Web-Tools for Streamlining Ballroom Dance Competition Management

by

Eric D. Nielsen

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

February 2003

Author .....
Department of Electrical Engineering and Computer Science
February, 2003

Certified by .....
Gerald Sussman
Matsushita Professor of Electrical Engineering
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Web-Tools for Streamlining Ballroom Dance Competition Management

by

Eric D. Nielsen

Submitted to the Department of Electrical Engineering and Computer Science
on February, 2003, in Partial Fulfillment of the
Requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis presents the design and development of the initial offerings in a potential suite of computer aids for ballroom dance competitions. The suite of tools is known by the name CompInaBox; while the initial module which handles on-line registration for competitions is named SlidingDoors. The tools have been developed using the PHP programming language – a scripting language that is easily embeddable within the HTML used to create web-pages. The PostGreSQL database system is used to handle all data storage requirements. The registration component was selected as the initial tool because it provides the data needed by practically all of the other possible tools. SlidingDoors has been coded from scratch twice, under two radically different coding methodologies. The first was in accordance with "common" web-programming views and a completely dynamic page generation model. The second was along the more traditional software engineering model and used more static page generation. As the story of the tool development unfolds, trade-offs between the two approaches are discussed.

Thesis Supervisor: Gerald Sussman
Title: Matsushita Professor of Electrical Engineering

Acknowledgments

There are many people who assisted me in this endeavor. Dave Leung and Warren Dew for their projects that showed the need for Comp-ina-Box. Tom and Elizabeth Nugent for answering tireless questions on all the possible permutations that a true "universal" ballroom tool should accomodate. Kristin Parent of the Univserity of Connecticut, Kristin Streilein of the University of Michigan, and Stephanie Chin of Tufts University for using the software at their home competitions. Derek Greenfield of Harvard Univserity and Chen Ling of the University of Maryland for their competiting tools that sparked many feature debates. My parents for coninually pushing me to finish and the folks at the Chomotomographic Imaging group of the Air Force Research Laboratory for giving me time and encouragement to wrap this up.

Contents

List of Figures

Chapter 1

Introduction

This project, CompInaBox (CIB), is focused around creating a suite of tools to assist in the planning and execution of Ballroom Dance (DanceSport) competitions. It is a complicated enough field to be rich with opportunities, relatively unexplored, and beyond the limit of one implementor. It is hoped that that work described here will form the core upon which others can continue to build.

1.1 Domain Space

To understand the technical design presented later it will be important to understand the domain of ballroom competitions, which come in a myriad of forms as well as the typical processes involved with planning and running a competition.

There are numerous types of competitions: "adult" vs. collegiate; amateur vs. pro-am vs. professional. The following discussion focuses primarily on amateur competitions both collegiate and adult, but the design will be general enough to cover the other categories.

Common collegiate competitions in New England attract two to six hundred participants and require 10-40 VIPs – judging and officiating – along with a staff of 20-40 volunteers to keep things moving. Competition organizers have to deal with parking, changing rooms and costume storage, food, and housing for the competitors and VIPs. Most groups also need to arrange for audio equipment rentals along with

"large function" rentals of such things as risers, tables, chairs, fans, and, in some case, sectioned dance floors. Operating budgets are typically in the range of $5-$15 thousand for the collegiate scene and significant higher in the adult arena. However, the complexity of these logistical concerns is often overshadowed by the issues surrounding scheduling and registration of competitors.

A typical competition offers numerous events, broken down into proficiency levels, styles, and age levels. Proficiency levels are used to measure a competitor's progress and ensure that they are competing against the "right" set of competitors. There are two proficiency level designations commonly in use in the collegiate and amateur competitions: "Medal" – Pre-Bronze, Bronze, Silver, Gold, Novice, Pre-Championship, and Championship – and "Experience" – Newcomer, Beginner, Intermediate, Advanced, and Open. Typically a correspondence can be drawn between the two with Open mapping to Novice and above, as shown in Figure 1-1. Many competitions mix and match the level labels so the two tracks are not distinct.

Styles refer to groupings of roughly similar dances, Standard (or Modern), Latin, Smooth, and Rhythm. The first two are also called International Style; the latter two are called American style. American style is only danced competitively in the United States of America, while International is competed world-wide. The Standard and Smooth dances are what people normally think of as Ballroom – Waltz, Tango, Foxtrot, Viennese Waltz, and Quickstep – dances that float, stalk, and swirl around a ballroom with the leader in a tail suit and the follower in a ballgown. Latin and Rhythm are the "earthier" or "sexier" dances of Rumba, Cha-Cha, Swing/Jive, Samba, Paso Doble, Bolero and Mambo – dances marked by skimpy outfits and very active hips. Figure 1-2 captures the mapping of styles and dances in a chart form.

Within each style/level combination a competition will offer one or more events. An events contains one or more dances. The realm of possible dances for a given event is dictated by the style. In USABDA (United States Amateur Ballroom Dance Association – the governing body of adult amateur competitors) competitions the minimum offerings within a style is set by the level. Events are uniquely identified, at collegiate competitions, by the level-style-dance(s) combination, such as "Bronze

American Waltz" or "Advanced International Cha-Cha/Rumba." The larger USABDA competitions have to prepend the age level to the event name to further identify the event. Age levels start at Pre-Teen I (9 & 10 year old) and step up every two years until reaching Adult; above adult there are two levels – Senior I and Senior II in 15 years intervals. The "Adult" age-level is the default level to which college competitions are equivalent and is typically the largest of the categories. These age level categories are closely aligned with the levels adopted overseas.

Competitions that host events for both amateurs and professions require one more label to fully qualify an event name. The common "classifications" are "Professional", "Amateur", and "Pro-Am", where the last refers to events where one member of the couple is a dance professional and the other is an amateur, in some cases these are called "Student-Teacher" events. These competitions will sometimes specify "Pro-Am, Am leading" or similar to further restrict and/or balance the entries, but this tends to occur more rarely and often only at the higher levels. Competitions with Pro-Am events also tend to subdivide the proficiency levels more than Figure 1-1 indicates. Within a given level, such as "Gold" you might have "Open Gold", "Closed Gold", and "Gold Star" where each of these has slightly different rules on how strict the syllabus restrictions are.

The CompInaBox project is targeting first the standard collegiate scene, then the greater USABDA/amateur arena, and lastly the Pro/Pro-Am/Am realm. This focus was chosen because the author is a member of the first two communities, their requirements are slightly simpler, and, with the organizers' limited budgets, they are the most receptive to using new tools if they are cheap or free.

Competitions tend to require registration in advance. Various competitions will have different rules for determining for which events a couple is eligible. Competitions will also commonly set limits on the number of events a given person or couple may enter. There are two common systems in the US for determining event eligibility, the YCN (Youth College Network) Proficiency Point system and the USABDA Proficiency Point system. Both are designed to bar competitors from competing at or below levels in which they have had measurable success; they only differ in how they

define success, with YCN being more "generous" as its designed to accommodate the collegiate practice of advancing dancers about one level a year, as opposed to the USABDA practice of taking several years per level.

Competitions almost always allow competitors to enter only one or two level within a given style – the lowest level for which they are eligible and sometimes the next highest level. As an exception, many USABDA competitions do not count the "Novice" level as a level for the two level restriction. Competitions vary on the ability for a person to enter widely different levels in different styles. Most college competitions allow competitors to enter any number of events in their allowed levels for the same fee. Adult competitions tend to restrict either the number of entries or over-restrict the number of level-style combinations for a set fee, or charge per event.

Most competitions want to do their registration process on-line and the above discussion was intended to both familiarize the reader with common ballroom terms as well as show the possible complexities involved in the on-line registration tool requirements. In addition most competitions use a computer based scrutineering tool to score the competitions. Scrutineering is the term used to describe the act of compiling the marks of judges into call-backs and final placements and will be discussed in the next section. The scrutineering tool should be pre-loaded with the initial registrations and ideally would be kept up to date with the late registrations and the day-of the competition adds and drops.

1.1.1 Judging & Scrutineering

Ballroom dancing is judged according to a system referred to as the "Skating System of Judging" which reflects its roots in the method of judging as used in figure skating, about two iterations ago.

The leader of a couple(typically the gentlemen, but in some same-sex couples, one of the ladies) wears a number of his back for identification purposes. In early rounds, typically quarter-finals and earlier, the competition field is broken up into multiple heats of about 12-20 couples. An odd number of judges form a panel who will "mark" the round. The MC or the Chairman of the Judges will direct the judges

to "Call-back n from m couples, in p heats" where n is usually about 1/2 m and p is m/heat size. The judges then watch the 90-120 seconds of dancing in each heat and "mark", by writing down the couple's number, the couples they think should be in the next round – an approval voting system. There is no requirements to allocate the marks evenly between the heats, but most judges do, as the initial heats are close to random.

The judges' sheets are then collected and brought to the scrutineer who tabulates the marks to determine which couples will advanced to the next round. Typically a couple needs to get a majority of judges' marks to be called back, but its possible with various mark distribution to make the cutoff ± 1.

For final rounds, the judges are instructed to "Please rank (or place) all n couples"; they must assign a unique placement from 1 to n to each couple. Again the sheets are brought to the scrutineer who determines the final placement. The short version is for each place, starting with 1st, the couple that receives the majority of that place or better wins that place. There are a series of rules for breaking ties or dealing with no couple receiving a majority initially. The rules for tie breaking can become very involved, requiring numerous calculations on specified subsets of the total marks and is one of the places computers have previously helped with the execution of dance competitions.

Scrutineering is the fundamental driver of the timeliness of the competition as later rounds can not be started until the previous round can be scrutineered. Computer tools assist in this process, but wise competition organizers plan their events so that there is enough built-in time between rounds of events, either by interleaving enough different events or by inserting general – non-competitive – dances between rounds where a bottleneck is expected.

Various ideas have been considered for how to further streamline the data entry portion of scrutineering – such as having the judges use palm pilots or long range bar code readers, etc. However none of these options looks feasible from a social perspective, even if the technology would work. These proposals are not addressed again in this project.

1.1.2 Master of Ceremonies

The Master of Ceremonies is the "voice" of the head table and runs the competition. At times some of the duties of the MC are split off and given to the "Chairman of Judges". Typically the MC is responsible for announcing events and call-backs; splitting the rounds into heats; assigning judging panels; and directing the scrutineer to call back more/fewer competitors when the desired number isn't possible. The MC is also responsible for coordinating with the DJ when an extra round of an event is needed in order to insert the additional music.

The MC, like the scrutineer, tends to be swamped with pieces of paper – different lists of lists, as well as scraps of paper from the scrutineer for call-backs and results. A computer tool here again could be very useful, but presupposes the existence of registration and scrutineering tools to drive the data needs.

1.1.3 DJ

The DJ is responsible for planning and playing the music for a competition. Many songs need to be sped up or slowed down to match the specified number of measures per minute(mpm) as specified in the rules outlined in the different dance organizations' rule books. Other songs need to be queued up to a certain point to avoid either a long intro or a change of timing. The beat of some songs is much harder to hear than others; these songs should not be played during the early rounds of the lower level events. Finally some songs can grate on the audiences nerves if overplayed and should not be used in events that have several heats.

The DJ is responsible for taking all these considerations and scheduling a song for each round of every dance. Typically the DJ will also prepare a list of extra songs of each dance type in case of unexpected additional rounds and another list of songs to use during the "General Dances" commonly used between sessions or as filler if the scrutineer needs a break or is running behind.

Most competitions DJ using multiple CD player systems. Some are starting to use computer-based MP3 systems. Many competitions will pre-record their competition

play list onto a small number of CDs to limit the disk switching demands.

Here we see that there is a possible need for two different computer aids. The first could assist in the pre-planning. A simple spreadsheet/database program could be used to help track which songs have been assigned from an organizers's library. Furthermore this tool could track the needed pitch change/cue point. A more complicated version could do some audio analysis to figure out the needed pitch change, etc without making the planner count measures. This program could also archive multiple people's ratings of the song and could use heuristics to generate a play list on its own.

The other application could be a day-of program that could remove the need for a DJ, assuming that either an MP3 system, pre-burned CDs, or a large CD juke box are available and could be computer controlled. The DJ program could then be slaved to the MC's terminal, moving towards the possibility of a one-person head table.

1.1.4 Information to Competitors

Competitions need to be able to accurately inform the competitors on who has made call-backs to given rounds and to post the results of all events for public display in a timely manner. The latter is typically done by posting an extra copy of the scrutineer's sheets; the former is accomplished by spoken announcements made by the MC. While posting the scrutineer sheets works well, it tends to impose an additional burden on the head table to find time to generate these extra sheets. Announcing competitors numbers is similar, while the system works, there are always a few people who misheard or are unsure if they were called or not. These people typically run up to the head table and pester the MC as he is trying to do his job. In both cases, if the data is available electronically it should be possible to offload some of the tasks to non-head table personnel, possibly to the registration & information desk that has very low stress once the competition is actually underway. Video projectors or slaved televisions, similar to airport arrival/departure information, could also be used in parallel with the MC's announcements to help reduce confusion. Also under consideration is publication of data to a wireless network filling the competition venue.

Another similar type of information requirement is the generation of the printed program for the competition and the competition web site. Both of these products could be largely auto-generated with the data already required to run the registration and scrutineering tools. As the other tools discussed are added, the gap becomes even smaller, until the only missing data is sponsor advertisements and textual amplification of the rules. As a first version, this "printed product" tool could output raw html for the web site and partial or whole LaTeX files for the printed program, focusing on the "common" data. Later versions could developed into a tool that can do rudimentary layout and font management for both print and web as well as exporting data sets in specified format for use in other more fully featured publishing tools.

1.1.5 General Event Planning

There are a number of useful tools that have not been captured above, but some have been alluded to. The MC/Event Planner would find a tool to help pre-schedule events and rounds useful, especially if the tool included heuristics based on past competitions for time requirements. Some competitions run multiple "floors" at a single time so either multiple heats of a single event or multiple levels of the same dance are run simultaneously. This complexity would definitely benefit from a computer aid as many competitions that run such as system cut corners and break the official regulations, as all the constraints for adequate time between rounds can be hard to satisfy while still fitting into the limited time available. This same tool could be used to pre-plan the changing of judging panels and changing of shifts for competitions staff.

Other possible useful tools would be simplistic budgeting tools for both projected and actual income and expenditures as well as tracking reimbursements needed. While these tools could be implemented via general spreadsheets, and many schools currently take this approach, there is some "domain knowledge" that would be useful to provide to less experienced event organizers. In addition a centralized, web-tool is useful for many competitions that are planned by a committee of people who seldom have the opportunity for face-to-face meetings and passing files around over email can cause versioning problems.

1.2 Computing Model

This project targets the Web as the delivery/computing medium for all these tools. While registration is the only tool that practically requires a web philosophy to reach its audience, the others could be simple stand-alone programs. The main arguments in favor of the web based implementation are portability, code-reuse from the registration tool to other future components, and familiar user interface. The last point is one of the most important; many of the event planners that the future components would like to help are not especially computer-savvy, so limiting developers to the common web-interface acts as a limit against "cool innovations" that are not intuitive except to power users. Furthermore most of the components would likely have a pool of potential data enterers and so some method of remote collaboration is needed.

1.3 Similar Projects

Computer tools in the ballroom domain space are relatively young.

There are at present three scrutineering tools in use. The first is Chester, a program written by Warren Dew. Chester is a Mac program and has not been ported to other platforms, but a JAVA port is planned. Due to the increased usage of local schools using CompInaBox, Dew has added an simple event and registration import facility to allow CompInaBox to completely setup and pre-populate Chester's data fields. Its only data export are text files. Chester is partially responsible for sparking the CompInaBox project; Chester is also the preferred scrutineering tool for use with CompInaBox.

The second is a commercial program called COMPMNGR. COMPMNGR is actually more of a complete solution; similar to CompInaBox as a whole, but designed specifically for the National Dance Council of America (NDCA) and other Professional and Pro-Am competition needs. It includes a scrutineering tool; this tool is normally the only part of COMPMNGR that is found useful by collegiate and USABDA competitions. COMPMNGR is currently being offered for free to YCN and USABDA

competitions. While it is possible to export registration data to COMPMNGR, it is a messy process involving first setting up the competition events in COMPMNGR and then telling the registration tool about the COMPMNGR assigned event identifiers before generating files to load the registrations into COMPMNGR. The tool also handles the assignment of competitors numbers itself and does not support independent assignment of numbers or non-alphabetical orderings.

The third scrutineering tool is a commercial product called Scrutiny3 developed by scrutineers in South Africa. While COMPMNGR is becoming the default tool used America, Scrutiny3 is widely used in the international arena. Its feature list looks more appropriate to the collegiate scene than COMPMNGR's and allows simpler information interchange than COMPMNGR. However, this is based solely on reading feature lists and extremely brief glances at the tool when used at the Cornell DanceSport Spectacular in the fall of 2002. It also appears to have some difficultly printing scrutineer sheets in the format preferred in the United States, or possibly is too confusing to novice scrutineers.

Most collegiate competitions offer on-line registration; most USABDA and NDCA competitions do not. A few competitions use email or cgi-to-email gateways, but many are moving towards custom built applications.

Dave Leung's (MIT) OpenTelemark was one of the first and was a JAVA based tool; OpenTelemark was created in 1998. Mr Leung switched languages shortly thereafter and wrote OpenImpetus, a Perl port of OT. OI saw sporadic development until early 2000 and was used by competitions at MIT, Brown, and Northeastern during its reign. OI is the direct parent of the CompInaBox project. The OpenTelemark/Impetus family was hindered at times by its internal, home-grown log-file/database storage system and often had data integrity issues.

After the CompInaBox project started several other tools began to appear; brief descriptions of each follow.

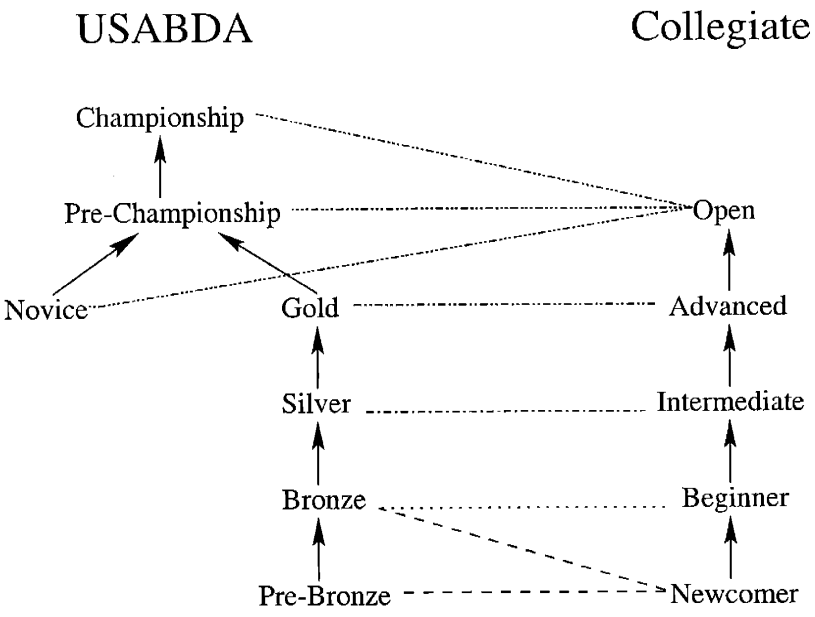
Derek Greenfield (Harvard University) began writing his Wasabi program in 2001 for use at Harvard's two competitions. It is a Perl implementation with a MySQL back-end. Wasabi is not being pushed strongly at this time outside the two Harvard

competitions, but does have on-going development. One of Wasabi's distinguishing features over OI was it automated creation of invoices for college teams based on their registration. Wasabi and CIB have both benefited from informal, "feature-borrowing" as the developers work off each other's latest bright idea. Unfortunately to date, different development models and drives have precluded a more combined approach.

Brown University, after fighting with a local install of OI the previous year, wrote their own system in 2001 in LISP. Overall the system worked, but lacked many of the registration statistics features common to the other tools of its type. It was a custom tool and has not been pushed on the community. The LISP tool is being used again this year, with a reduced function set.

Chen Ling (University of Maryland) also wrote a tool which has been used at various DC area competitions and at the Intercontinental DanceSport Festival. Its noticeable feature is an integration with PayPal to process on-line payments as well as registration. Chen Ling has expressed interest in joining the CIB effort so his tool and the in-house SlidingDoors will begin to merge in the future.

Proficiency Levels



Age Levels

Senior II	50 and over
Senior I	35 and over
Adult	19 and over
Youth	16-18
Junior I	14-15
Junior II	12-13
Pre-Teen I	10-11
Pre-Teen II	9 and under

Figure 1-1: The two commonly used proficiency level systems and the common age level designations.

	Ballroom	Latin American
International	Standard (Modern) Waltz, Tango, Viennese Waltz, Foxtrot, Quickstep	Latin Cha-Cha, Samba, Rumba, Paso Doble, Jive
American	Smooth Waltz, Tango, Foxtrot Viennese Waltz (Peabody, Polka, Texas Two-Step)	Rhythm (Nightclub) Cha-Cha, Rumba, Swing Bolero, Mambo (Nightclub Two-Step, Hustle, Lindy Hop, Merengue)

Figure 1-2: The dances are grouped into four styles.

Chapter 2

Design

2.1 Requirements

Early in 2000 a group of MIT Ballroom Dance Team members, interested in furthering the state of computer aids for ballroom dance, met to discuss what would become the CompInaBox project. At the meeting were Boris Berdnikov, Alex Bernstein, Warren Dew, Dave Leung, Eric Nielsen, Tom Nugent, and Tuan Phan. Dave Leung was the original author of the OpenTelemark/Impetus family of registration programs, while Warren Dew coded the Chester scrutineering tool. Tuan Phan was representing the interests of integrating parts of registration into his team/club/studio level portal system. The author was the emerging heir-apparent maintainer for OpenImpetus. After the initial code, bug list, and wish list review and discussions with Tom Nugent, the author and Nugent felt a redesign was required. The others were there to see what could be done and if they were interested in taking a larger role.

Two main decisions came out of this meeting. The first was to proceed with the often called "Unix" model of systems of cooperating "simple" systems. The team was not going to try to write a single massive ballroom dance application, but many smaller tools. The second was that the primary means of intra-program communication would be through a Relational Database Management System (RDBMS). Figure 2-1 depicts the initial design plan with its layers.

The "base" layer for the system would be the RDBMS itself, with varied API's

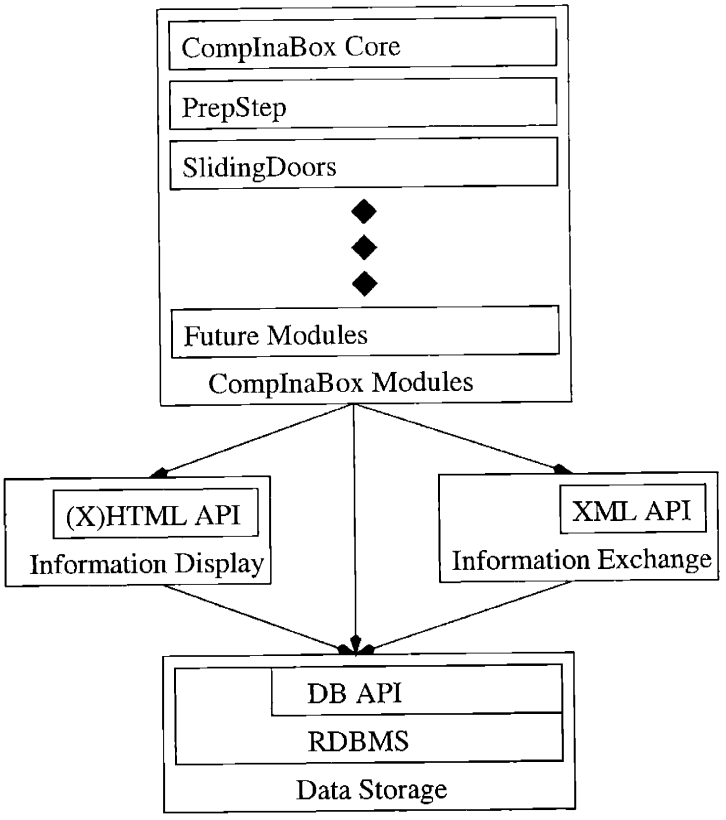


Figure 2-1: Preliminary Comp-Ina-Box Design

layered on top. The DB API layer can be split into two sub-layers. The lower is simply the DB API calls provided by the programming languages used in this project. The higher layer is the CIB-DB API with functions such as RetreivePerson-ByID/Name/Email and similar. The individual Comp-Ina-Box component applications would be built on top of this base. Typically this components would be broken down into at least two layers of their own: an information manipulation layer and an information display layer. However no set architecture is dictated for component applications, aside from a requirement for data storage in the RDBMS and support for interoperability with non-CIB tools through XML as a minimum data format.

2.1.1 Data Requirements

The author took on the task of defining the schema for the database, with a great deal of assistance from Tom and Elizabeth Nugent, especially with regards to exploring the

different needs of the different types of competitions. The database scheme definition took several months.

Many types of competitions were considered over the course of schema development.

College Competitions These competitions typically occur over 1-2 days; have multiple events at different levels and styles; are attended by teams and often have a social dance associated.

USABDA Competitions These competitions tend to be 1-3 days; have multiple events at different age levels/proficiency levels, and styles; are attended by individuals; typically have vendors and sponsors; and might use multiple dance floors.

NDCA Competitions As USABDA Competitions, but include events for both professions, amateurs, and pro-am.

Dance Festivals A week long USABDA/NDCA competition with workshops, private lessons, and social dances scattered throughout

A more detailed description of competition types may be found in the Chapter 1. The inclusion of the Dance Festival class of competitions was especially useful for figuring out the proper cardinalities of the database relationships as it tends to have the most demanding needs for similar, but unique entities. This was needed when dealing with the inter-relation between event sessions, floors, and events.

The existence of the multi-style dance events (such as the International 10 Dance, or the 19 Dance Championships) also imposed very strict constraints on the database schema. More than ninety-five percent of the events at ballroom competitions tend to be single-style events. In the rare cases when multi-style events occur, they almost always occur spread out of multiple sessions, ie the Latin 5 dances are danced in the Latin session and the Standard 5 dances are danced in the Standard session. As a result a given "round" of an event may not be tied to a single session. This means that

round-event combinations are the lowest level item that may be uniquely scheduled into a session.

The developed schema was then shown to Elizabeth Dew, an MITBDT member who works as a DBA. Her comments – which consisted mainly of breaking up a few large, but seldom used tables and standardizing some of the foreign key names – were folded into the next version of the schema which stayed relatively consistent for the next year of development. The schema is provided as Appendix A.2. Discussions of changes to this baseline scheme will occur in Chapter 3.

2.1.2 System Tools

At the initial meeting the team decided on a few tools to use for the project. An open-source RDBMS system was desired and PostGreSQL was selected, primarily for the reasons of being more SQL standard compliant than MySQL. The team of developers interested in working on the various components spanned a wide variety of programming experience ranging from almost pure-HTML only to programmers skilled in multiple languages. PHP was chosen as the primary language for development as it would allow the easiest incorporation of the "web designers" into the team. PHP is an HTML-embeddable, scripting language that shares many features with both C and Perl, but specifically targeting the web server environment. JAVA was identified as a desired language for components that require either client code or a more powerful user interface than a web form can provide.

2.1.3 Data Access Layer

The design and implementation of the interface between low level DB API calls and CompInaBox component applications was put on hold due to developmental constraints. The arguments for skipping this layer, and thereby likely duplicating effort in creating the temporary data pulls, are presented below:

Faster Prototyping It was hoped that the new tools would be available for use at the 2001 MIT Open Ballroom Dance Competition

More Developers Flagging developer interest; having individual projects to work on would help motivate people

Fire-Test the Schema The initial schema was bound to have omissions, prototyping the applications would help identify these omissions before the intermediate layer had been coupled to the schema.

The author began work on the Web Registration Component, later code-named "SlidingDoors". Tom Nugent began work on a scheduling aid and, later, on a Competition Setup Tool, later code-named "PrepStep."

2.1.4 SlidingDoors Design

SlidingDoors(SD) was the name given to the web-based registration component that is regarded as the main application of the CompInaBox project. It is the face seen by the majority of the users and provides the data needed to drive many of the other applications. The name SlidingDoors was chosen to continue the ballroom dance step naming trend to show its lineage from OpenTelemark and OpenImpetus. The changing from Standard to Latin steps showed that the new program was a very different application.

Two radically different versions of SlidingDoors were developed sequentially. The first version was highly dynamic and retrieved everything from the database that might change between competitions. It was designed so that a single instance of SlidingDoors could simultaneously service multiple competitions. The various pages displaying statistics were generated anew on every visit. In this manner SlidingDoors would achieve the data integrity on which OpenImpetus was often criticized. It would also make the job of setting up a new competition as simple as inserting data into the database.

The user interface development was driven by a desire to better support dancers who dance with multiple partners, a common occurrence in collegiate competitions. This occurrence also typically generates a problem of the same person being entered under multiple names, ie Tom or Tommy Smith. This can cause problems with invoice

generation, competitor number assignment, and in the printed program. SlidingDoors would attempt to catch these close matches and, where appropriate, prompt the user for a resolution. Simplicity, appearance, and easy of use also entered into the design.

The final criteria for the user interface was a desire to accommodate both leaders and followers as the "prime" registrant. Most of the other systems have the registrant enter either both the leader and follower on the same page or start with the leader and ask for the follower's information on a later page. While the first system is unbiased to the person's role, it greatly complicates form validation and processing. The second system is biased and tends to result in systems where the user may only retrieve couple information based on the leader's identity. It also makes it hard to retrieve all the leaders for a given follower. As a result SlidingDoors asks for a "prime" registrant on the first page of registration and partners on the later pages with the given partner's role (leader/follower) being listed explicitly.

Where possible information display and processing have been separated. User displayed pages submit to an "interstitial" page that handles the form validation and processing and then redirects to the appropriate next user viewable page. This breaks the common PHP practice of having forms self-submit in order to keep the user viewable pages easy to update and more cleanly separates the job of Web Page Designer/Architect from application programmer.

Figure 2-2 depicts the initial expected page transitions. The home or index page of the site would display top level information about the competition as well as a summary of the registrations to date. From this main page the user may view more detailed statistics on registration, register, update their registration, or conduct team level registration matters. Registration begins at the prime.php page with the entered data being validated by the prime-check.php. Control is passed back to prime if a field is left blank or the data is invalid. Likewise, if a name conflict is detected control would be passed to a page to resolve the conflict. Once a prime has been established the partners.php page is reached which is the "heart" of the registration process, displaying the prime's current registration as well as allowing editing, adds and drops. New information follows a similar path – first validate the data, then check

for name and event conflicts and, on success, return the user to partners.php with the new data displayed for confirmation. Name conflicts are handled on a separate page; event conflicts or invalid data return the user to the partner.php page with error messages and entered data preserved.

This first version was designed to be prototyped quickly as a collection of scripts. This version was completed in time for the 2001 MIT Open Ballroom Dance Competition in mid April and underwent field testing in early March. The stress test proved the fully dynamic page creation to be too slow for use under on the team's Pentium I server and the user interface proved troublesome to many regular users. As a result the application was not used at the 2001 competition and it was redesigned.

This second build started with an assumption of using an object model in addition to the script model. The separation of information processing and display as begun by the paired scripts of the prior version was continued into appropriate classes. The class diagram is presented in Figure 2-3.

The *StoredObject* family of classes are primary data management classes, through some do have rudimentary data processing capability. The stored object base class handles all loading and saving of information to the database. The other classes simply register the correspondence between their member variables and the columns and tables of the database. The *UIBase* and *HTMLDisplay* family of classes are primarily due to an earlier, aborted research effort investigating multi-modal interactions. The *UIBase* is effectively deprecated and is not used. The *DB* class provides a simple database abstraction layer and is the only file that contains RDBMS-specific commands. The *CompDB* subclass of the *DB* class is an utility class that adds tools for retrieving information from the database that do not fit into the main named objects such as *Person*, *Couple*, etc. Typically its functions generate aggregate statistics or return list of options to use as input to *HTMLDisplay* function to generate tables, select boxes, etc. The functions of *CompDB* and *StoredObject* are likely candidates for inclusion into the CompInaBox level DB API already discussed.

An even greater difference in this SlidingDoors compared to the previous was a decision to abandon the complete dynamic model and pre-generate as much information

as possible in order to speed up the response time. As a result a given SlidingDoors install will only be able to accommodate a single competition, however a given server could have multiple instances of SlidingDoors installed. The planned PrepStep and "Master Control Program" applications will be used to setup/update the new, more static SlidingDoors.

The initial plan for this static SlidingDoors was for PrepStep to output the display and processing scripts after doing some amount of variable substitution as well as outputting large code snipets for inclusion into the "standard" code base where large customizations were needed. Later it was re-thought to have PrepStep output a "constants" file and leave the variable placeholders in the code for substitution at run time, as the true issue was not dynamic versus static, but in the number and complexity of database queries. PrepStep would still output a few of the complicated code fragments that can not be easily generated at runtime.

2.2 Security

Like most multi-user web systems there are a number of security and privacy concerns that need to be address. First the threats will be detailed, then a security model will be presented.

2.2.1 Environment Threats

While most of the threats are relatively common, the requirements of the site often lead to different prioritizations of security vs convenience. The following paragraphs outline the type of threats considered.

Server Compromise

A server compromise is the system administrator's worst nightmare. The application should not allow a user, legitimate or otherwise, to gain access above which he has already been granted. Servers may be compromised through a variety of means. "Rootkits" exist that probe machines for known vulnerability in common software

and seek to exploit them. Outsiders may craft specially designed URLs and input strings seeking to trick the server into executing arbitrary code that gives them ever increasing access to the system, or deletes information. It should be the goal of the application to open no new holes, while limiting the ability of the attacker to exploit potential holes in the supporting programs.

Web Site Defacement & Cross Site Scripting

Web site defacement can happen inadvertently, if for instance, a user forgets the closing tag of the bold entity in HTML. It could be a simple malicious attack – entering a long series of unbreakable characters causing the page's formatting to be destroyed. The craftier adversary employs Cross Site Scripting (XSS) which often leaves little visible sign, but results in private information — email address, cookie, and session information — being leaked to the adversary allowing for account/session hijacking and worse. XSS focuses on getting custom client-side script, such as JavaScript, added to the page. This script could redirect the user to a forged copy of the site and collect passwords in addition to the privacy leaks discussed above. The best defenses against these types of attacks are twofold: never trust the user and always initialize variables to known values. These same principles apply to data being stored into a back end database to avoid executing unintended queries that could wipe the database.

Harvesting of Personal Data

Harvesting of emails for mailing lists is the most obvious and common of such attacks; yet many others are possible. Some competitions request address, phone numbers, date of birth and citizenship information, with the latter two used to determine eligibility for championship/scholarship events. While this data may be available elsewhere; an application should still not make it available to those who do not need to know it. Ideally the information should be encrypted to and from the server and encrypted in the database while stored to avoid both network snooping and unscrupulous local users. The encryption at the database level offers another small obstacle in the case of a server compromise, but probably not one that will hold the

attacker for long. SlidingDoors often needs to resolve an "Identity" conflict — is "Bob Jones" the same person as "Robert Jones"; in these cases enough information needs to be revealed to allow the user to make a ruling, but without leaking the privileged information. Common solutions here involve displaying the minimum amount of information possible, encrypting what you can, and using authentication techniques such as passwords where appropriate

Denial of Service

Denial of Service(DoS) refers to the ability of an attacker to consume enough of some crucial resource of the server, such as connection bandwidth or processor time, so that the server can not handle real users' requests. The server may not be compromised or crashed, but it is still unusable. In some cases a DoS attack may be indistinguishable from a surge in user activity. In most cases protecting against a DoS attack is matter for the system administrator of the server, configuring firewalls, web servers, and specially designed software and not the realm of the application. However, some of the parts of the application need to be especially safeguarded. For instance, the administrator of a competition should have the ability to close registration early, or prolong it as needed. However, should an attacker gain the ability to close registration early, that would be a successful DoS exploit. Some of the considered methods for keeping the registration statistics current use a considerable of processor power and if implemented poorly could result in a DoS exploit becoming available.

2.2.2 Security Tradeoffs

The following sections look at some of the trade-offs involved in designing the security infrastructure for the project. The two main trade-off spaces are security vs either speed or usability.

Security versus Usability

Security and usability tend to be directly opposed to each other. Requiring a user name/password system for access adds an immediate barrier to the usability of a site. Without a user name/password system all simple authentication/authorization systems fall apart. More complex systems involving PKI certification and/or security tokens, may be slightly more user-friendly, but they require a greater infrastructure that is well beyond the scope of this project. However, full client-certificate PKI should be explored as an eventual improvement. Furthermore, the average user of SlidingDoors only visits the protected portion of the site a small number of times. The statistics pages that receive high traffic would not be in the protected area. Given this, it seems unreasonable to ask a person to get an account for a single visit and, therefore, a password-based security system is inappropriate for the general registration tool.

In the future it is hoped, that as multiple competitions begin to use this system, that a central datastore can be established to expedite form fill out. Whether this system is part of CompInaBox or not will depend on how schools and organizations use CompInaBox. The originally envisioned model was that a few organizations would run a server covering a geographic region, for instance, the MIT Ballroom Dance Team would run the server for the Northeast Region. However, there appears to be significant interest from many schools in running it locally. If this model prevails the central datastore and authentication engine should lie outside the project. If the earlier model prevails a simple system where each of the servers knows about the others and can pass tokens around could work. With a small number of servers, its easier to ensure that all the proper keys are established for securely passing authentication information around, which would not be possible with a myriad of installs, often run by inexperienced system administrators.

For places where the site requires login, such as the administration pages for SlidingDoors and PrepStep, the competition setup module, the security model can afford to go the far extreme. Most users will be visiting often enough to require a

login. The data stored and services provided are significant and a successful attack would be very destructive. Therefore all efforts should be taken to avoid account compromise. This will show up in the design to be presented shortly; this is especially highlighted in the portion concerning password "reminders" that are common on web based forums and other services.

Security versus Speed

Encryption, decryption, hashing, MAC'ing, etc are all computationally expensive compared to most most other operations involved, though some database queries may take more CPU resources. As practically all security is built on an effective combination of the various techniques just mentioned, security will always impose a performance hit on the application. Most of SlidingDoors, the main tool under discussion in this thesis, is not security conscious for reasons discussed in the preceding section. The parts that are administrative areas which need to be well protected and are only used by a few people. Given this it seems reasonable that security win over speed on most tradeoffs in this space. Furthermore the author wanted to learn more about security systems and decided to take a rather paranoid approach. This decision will be reflected in the detailed design to follow.

2.2.3 Security Design

Most of the site is not heavily secured. Anyone can register/update anyone else's registration, given the target's email address. At present confirmation emails are not sent on registration or updates and to date this has not been abused. Nightly back-ups are kept to remedy any malicious attacks should they occur. One recent competition using a new home-grown tool assigned all entrants a random password required to modify their registration and also assigned a school-level master password for team captains or registration coordinators to use. In the long-run this is likely where CompInaBox will go, but there are a number of usability issues that will need to be resolved through various tests.

The administration areas of the site, however are locked down rather strongly using a role-based access control paradigm. There are three "families" of roles – Site, Competition, and Team. Within the site family the available roles are Member, Staff, and Administrator; within the competition family the roles are Staff, Comptroller, Registrar, and Coordinator; team offers Member, Treasurer, Registration Coordinator, and Captain. While Staff and Member appear in two lists they are distinct roles, as roles always include the family as well as the role. At present Site Member includes all visitors to the site, logged-in or not, in the future an anonymous role will likely be dictated by evolving requirements. Within each family users may grant their role or lower to other users. Only Site Staff or higher may grant the initial Competition Coordinator/Team Captain limited "superuser" privileges who can then delegate as needed. In affect each user has a key ring of roles. Pages are secured with a set of locks, such that if any lock is opened the page is viewable by the current user.

User accounts require a user name, email address, and user chosen password that includes at least one each of uppercase, lowercase, numbers, and "special" characters with a total length of at least 6 characters. Activating the account requires following directions mailed to the email address. During login, a few second delay is added before testing for successful login to slow a brute force attack to only several tries a minute. The delay before the test is to ensure that the presence of the delay doesn't amount to failure and allowing a brute-force tool to retry earlier.

At all times throughout the new account creation process and while logged in, the user name, client IP and session identifiers are MAC'd and compared to detect any attempts at session hijacking.

In a departure from conventional password protection wisdom, the login system advises the user to record their password somewhere rather than memorize it only and the system does not provide a "I forgot my password" auto-email feature as is common on most community web sites. This decision was made in light of the expected source of attacks. The most likely attacker is an intruder interested in taking over any particular server for use in further malicious activities – there is very limited personal data stored and no financial data stored that would entice a directed

attack. Therefore we need to make it difficult for the average malicious web user, with no relation to our regular users, to impersonate a user to begin an escalation of privileges type attack. When all that is required to reset and mail a password is knowledge of either the user's email or user name, it is very easy to generate a reset email. Furthermore it is likely trivial for most intruders of this nature to intercept an email sent in plain text. Therefore account compromise is only an email away.

As the project grows, the security offerings will as well. A password reset service will be added, but will likely require that the user provide a public key with which to encrypt the email at account creation time or answer a randomly selected question provided at account creation time and request the password reset from an IP address from which they had previously visited.

2.3 File System Layout

Now that the application design and security model has been discussed, it is possible to begin to specify the locations of the different components within the file system. The main concerns here are to ensure that the resulting layout is consistent with both the Linux Filesystem Hierarchy Standard and common practices in the field, with the practices of GNU/Debian Linux being used as the primary common practice data point as it diverges from most other interpretations of the standard.

Fundamentally a design was chosen so that it fit the FHS and was then tweaked so that it could also accommodate Debian's policies. The main distinguishing feature of Debian is the practice of treating all installed packages as "system-level" tools, ie outside of the /usr/local directory tree. As a result the software must be able to be split across many top level directories.

At the top-most level, the application may be broken down into documentation, source code, configuration files, and auxiliary support.

Documentation may be broken down into two main categories: installation/ developer instructions and user documentation. The former category are the omnipresent README, COPYING, AUTHORS, and INSTALL files that most Open Source soft-

ware packages use. User documentation often takes several forms – common ones are man pages for use by local users and web-accessible pages.

The source code versus configuration file demarcation is slightly artificial as the configuration files in question are PHP scripts containing variable name-value pairs, and at times contain a small amount of embedded logic. However a quick look at the file typically suffices to be able to tell the two apart. The "real" source code files can be further broken down into back end scripts, front-end scripts, viewable pages, class definitions, and general utilities. Only front-end scripts and viewable pages need to be housed within the web-serveable directory tree. The other files only need to be in a web-server readable location.

Auxiliary support includes files needed for interprocess communication such as lock files, log files, temporary files, etc. These files often need to be web-server writable, but do not, and should not, be directly web accessible.

Figure 2-4 shows the file layout used by this package. `bin/` contains the back-end scripts, each of which will have a man page in `doc/man`. Doc itself will hold user documentation both HTML files for use in local installations and either HTML or PDF file as user guides for using the system. The web-server will reach these user guides either through symlinks from the web tree or as direct includes, depending on the templating system developed. The installation/ developer instructions are placed at the root level of the package tree, as seen in most other packages. `doc/` is easily relocatable to the typical `/usr/share/doc/` hierarchy under Debian.

All the include files, whether they are class definitions, configuration files, or utility functions, resided within the `include/` hierarchy – `classes/`, `conf/`, and `general/` respectively. The configuration component may be relocated to the `/etc/` hierarchy as directed under Debian. A pair of files, `paths.inc` and `include_others.inc`, resides outside of the sub-directories of the `include/` tree and contains the path information for reaching the possibly relocated files. These files are also responsible for taking a URL and determining which configuration files and utility functions to load.

`var/` contains the "build" area for PrepStep generated customization files for SlidingDoors as well as the lock files used by the statistic generating code to avoid

extraneous statistic rebuilds as well as ensure only one build occurs at a time. This directory may be easily moved and, like the others, its location is recorded in the `paths.inc` include file.

`src/` contains the compressed source files (`tar`-ed and `gzip`-ed) for the individual components. These files are used both by the installer and also by curious users/developers who wish to see how the system works or to improve it. Because PHP is an interpreted language there is not a difference between source and executable as in compiled languages. The complete "pristine" source is still provided in this directory to aid in recovering from possibly harmful local edits.

`www/` contains all the web-accessible pages. It should either be configured as the destination of an "Alias" from within the web-server or should be a symlink from the main web-tree. Either method works depending on the host machine's policy. The main point of interest is that the directory names do not directly match the names of the components that provide the functionality, for instance `www/register/` and `www/stats/` are the realms of the SlidingDoors component, rather than having a `www/SlidingDoors` directory. This was done to attempt to simplify the URLs. Each component is free to subdivide its area as needed. Components will commonly have a `SCRIPTS/` subdirectory for front-end, non-viewable pages, and an `Admin/` directory for administration tools. All tools in this administration directory need to check for user permissions as the location of the tools will be relatively common knowledge. The maximum number of files possible have been moved out of the web document root for security reasons.

Each hosted competition will have a subdirectory under each of the functional areas of `www/` — `register/`, `stats/`, `results/`, etc. It is felt that distributing the competition specific level pages to the leaves of the Uniform Resource Locator(URL) space will assist in navigation as well as simplify the catching of typos in URLs. Furthermore it eases the eventual URL rewriting task for avoiding the per competition installation of the tool directories.

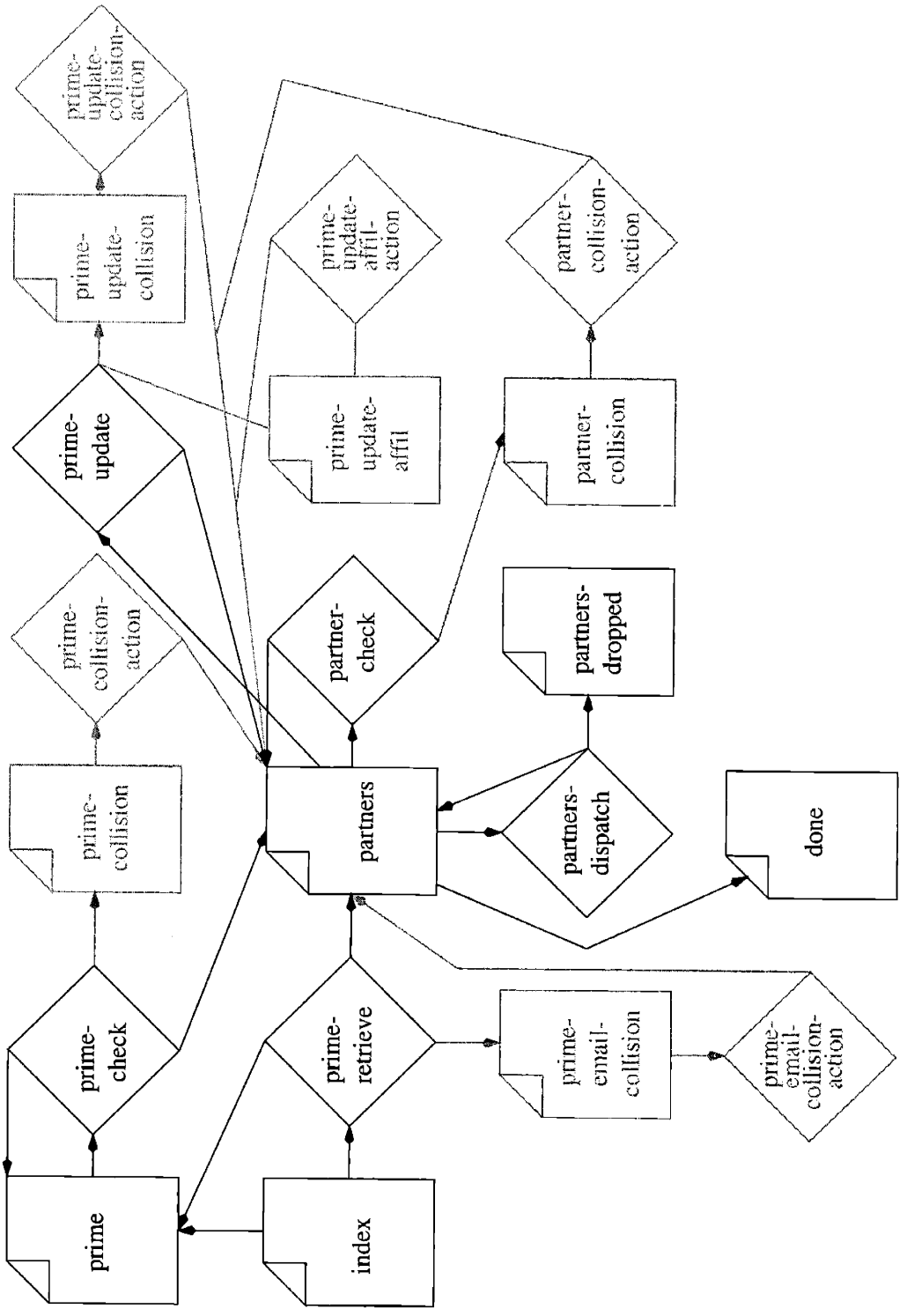


Figure 2-2: The page flow through the SlidingDoors system. Error handling pages are grayed out to help highlight the main path. Admin and group pages are not showed. Diamonds are "interstitual" scripts, folded pages are web-viewable.

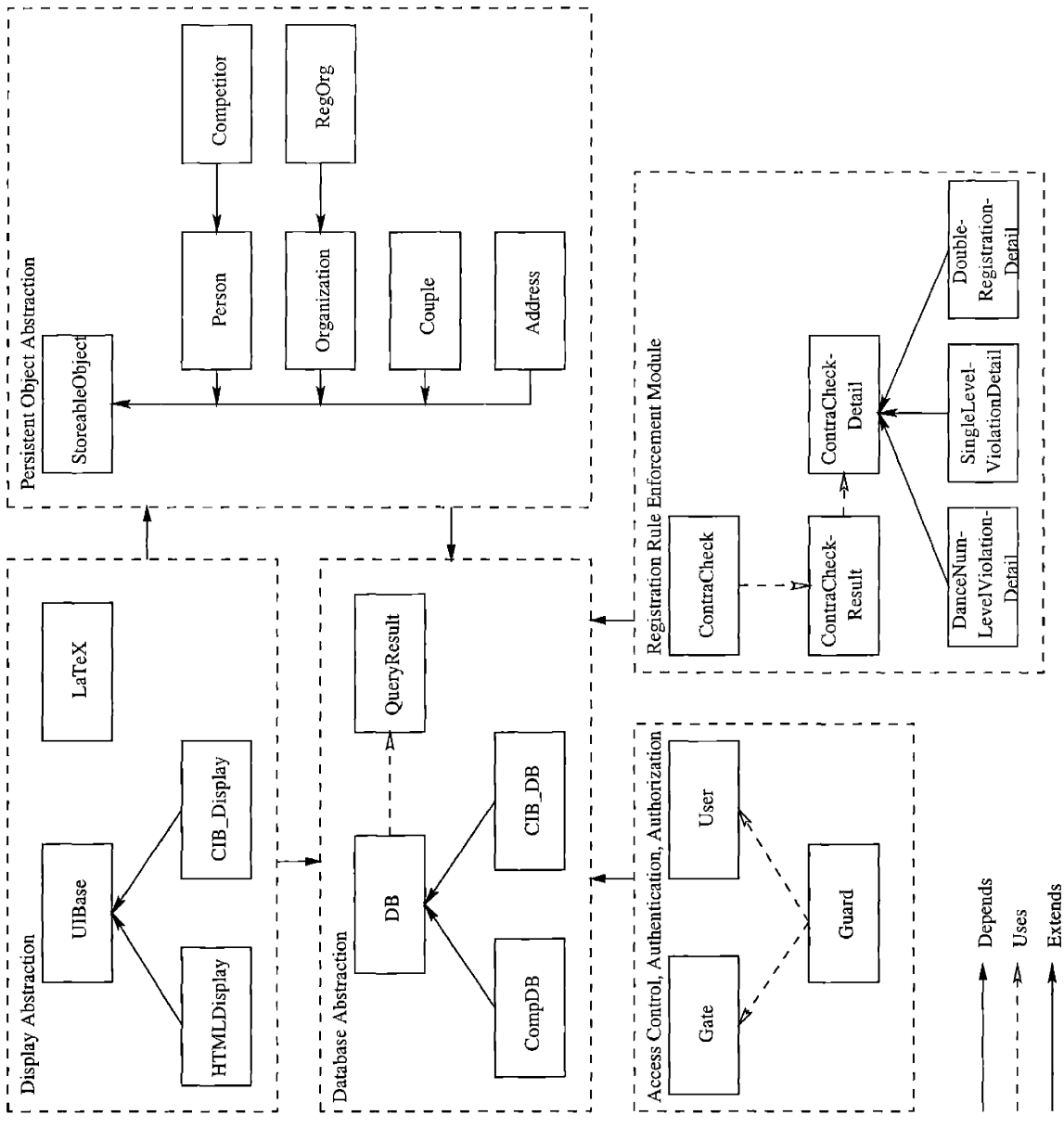


Figure 2-3: Class hierarchy as used in SlidingDoors and CompInaBox

Figure 2-4: The layout of the filesystem as used within the CompInaBox project. The dashed and dotted lines show where configuration files are pulled from depending on the location within the web serveable pages.

Chapter 3

Implementation

This chapter chronicles the difficulties encountered during the implementation of the design discussed in the preceding chapter. Chronologically speaking the design and implementation chapters are interleaved as the design chapter details the redesign that followed the first failed implementation. Not every difficulty is discussed here; rather, the most interesting, in the author's opinion, solution space problems are detailed.

3.1 User Interface Forms

The display and handling of the HTML forms to interact with the user proved to be one of the most difficult sections to implement. In order to keep the tool useful across the range of possible competitions, the user interface could not be hardwired. Most college competition in the New England/New York area will ask for first/last name, email address, and school affiliation. USABDA/NDCA competitions will ask for the above plus address, home/work telephone numbers, USABDA or NDCA identification numbers, birth date, etc. College competitions in the Midwest often ask for the USABDA number. Some local competitions will request a telephone number.

The simplest solution would be to create form that does everything and allow the competition coordinator to mark the unneeded fields as optional. This, however, would lead to a very "busy" form that would be confusing to the user and would be

seen as intrusive for many college competitions, even if the information is optional. The code for form generation and handling has gone through the most changes during the projects history and has almost returned full circle to the simplest solution with a few modifications.

3.1.1 Forms from Database

The first version of Sliding Doors, written in early 2001, attempted to postpone the problem in many ways. A mechanism was created to allow pair of code snipets for sub-form display and processing to be written and placed in a specified directory. A competition coordinator would then pick from this list of form elements to create their form. Their choices would be stored in the database and then retrieve at run-time to display. An initial set of choices would be developed for use and others could be coded by interested users if they needed more control.

This version worked well on the development machine and was deemed "good enough" for the first beta test on the Ballroom Dance Team's server. Table 3-1 lists the relevant features of the two machines. The development machine is substantially more powerful than the production environment. At no point during development was a processing lag observed between page request and display. However, it was quickly found that this dynamic form generation was significantly over-taxing the production server and pages were taking a second or two to load for non-simultaneous requests. This delay coupled with the poor response of the statistics pages to be discussed later this chapter is what led to the redesign and the change from generally dynamic to generally static. Most of the processing time was used to generate the event listing check-boxes, as opposed to the text fields and similar which while dynamic were fast.

3.1.2 Forms coded by PrepStep

The redesign of SlidingDoors featured a new form generation and handling system. While the 2002 MIT competition featured an in-line system, two other systems have

Component	Devel.	Prod. (old)	Prod. (new Aug/02)
Processor	Pentium III	Pentium I	Pentium IV
Processor Speed	733 MHz	133 MHz	2.4 GHz
RAM	128 MB	32 MB	1 GB
OS	RH Linux 7.1	RH Linux 7.1	Debian Linux 3.0
Supporting Tools	Apache 1.3.22	Apache 1.3.22	1.3.26
	PHP 4.1.0	PHP 4.1.2	PHP 4.1.2
	PostGreSQL 7.1.3	PostGreSQL 7.1.3	PostGreSQL 7.2.1

Figure 3-1: Comparison of Development and Production machines

been implemented to better accommodate the envisioned SlidingDoors and PrepStep interaction; both are presented below.

While it was realized that this new static approach was going to greatly complicate the creation of the as yet unwritten PrepStep, it was also accepted that this version of SlidingDoors was more of a prototype of what PrepStep should generate, but not an exact replica of the code that PrepStep will generate. As such, the initial coding of the forms was not divided up in a manner that would facilitate the easy creation of the code by PrepStep.

In-line

In this system it was envisioned that the competition coordinator, after going through the PrepStep "Twenty Questions" style of customization would press a "Generate On-Line Registration" button that would perform variable substitution on the template code and then fill-in the blank functions for form display and processing with computer crafted code. The version used at the 2002 MIT competition had this code generated by the author at the appropriate places in the script and functions. The code for event display and event registration handling is separated out to an include file due to its complexity. This system created pages that were quickly loadable and also easy to tweak in response to user feedback during the beta test, as befits a prototype. However, code that would need to be computer generated at setup time was scattered throughout the source code.

Pseudo-Functions

The downside of the implemented system was that it would be almost impossible to code a realistic PrepStep that could generate the desired functions. PrepStep was envisioned as being a rather simpler program on the whole compared to SlidingDoors and simply passing a problem from one tool to another is not much of a solution. The first option explored for simplifying PrepStep's code creation was to pull the display and processing code completely out of the scripts, similar to the way the the event display and processing had their own include file.

To simplify PrepStep's job it was decided that nothing other than these code snipets should be affected by changing the form elements. This lead to requiring code snipets for the following:

Person Constructor Required to itemize the mapping between database fields and object variables, had the constructor simply listed the entire realm of possible, the object retrieve and search methods would have had an extreme number of un-needed database queries

Person Identification Form Can be reused for both Prime and Partner identification information gathering, as well as the Prime Update

Person Identification Validation Validates the information entered on the Person Identification From

Person Identification Processing If validated the information needs to be added to the database and the appropriate objects create or updated as needed before the script can perform its form-independent operations on the Person and Couple objects involved

Event Listing Form/Validation/Processing Events need the same types of code snipets as people

The implementation of these code snipets took the form of code to be included in-line at the needed locations using the PHP `include()` function, which injects the

named file into the source and then executes it before continuing with the reminder of the given source script. They could not be coded easily as traditional functions because of the variable number of arguments needed based on the form data required for a given competition and the design requirement to not require changes to the calling script — there is no way to enumerate the variables to pass in without changing the calling script. While the variable passing could be overcome through liberal use of global variables, that was also rejected as inelegant.

As the injected code will share the same scope as the "calling" function it avoids the need to use globals to pull all the submitted variables into scope. However as the form elements want to be reused for both prime, prime update, and partner some degree of argument passing is required and in the case of validation or processing a "return value" may be needed. Thus the following calling convention was created:

```
$variable1=value1;
$variable2=value2;
...
include(''code_snippet.inc'');
if ($returnVal==''value'')
...
```

This remains constant with all the customizations occurring within code_snippet.inc

.

3.1.3 Forms from Configuration File

While the system discussed above gives at some of the benefits of functions, and also allows for ease of creation by PrepStep, the code produced by it begins to look quite jumbled and inelegant. The code within the pseudo- function makes copious use of variable variable-name as it doesn't have the regular argument to parameter bindings. In it was later realized that appropriate use of default arguements could reduce the variable number of variables to a "regular" function and avoid all the extra complexity. As a result one more version was designed.

In this version the "constants.inc" configuration file would include a list of form

elements to include and then standard and fixed functions would handle the form display and validation as well as the object constructor and update methods. This approach had been avoided at first primary because of the difficulty involved in creating the generalized code for these sections, especially for form display and object update. An added concern is that this method greatly limits the flexibility of form layout for different choices of form elements. However this inflexibility may be seen as a positive as it ensures a semi-consistent interface across all SlidingDoors hosted competitions. As a proof of concept the form display piece was written and was found to be much easier and cleaner than expected. At present it is felt that this approach makes the most sense and is the approach selected for further refinement.

3.2 Statistic Pages

One of the users' favorite features of the original OpenImpetus on-line registration tool was the collection of statistics available on the current registrations. These statistic pages were also the place where the data integrity issues with OI's home-grown database most routinely surfaced. As a result much care was taken in designing the new statistic pages. These pages provide snapshots of the current registration along different slices, such as number of entries by event or by school or a complete listing by name. Drill-down within a category can pull up the competitors names, affiliations, and alternate events. SlidingDoors added two new statistic views, a competition summary view and a style summary page. The competition summary provided a small table showing the average number of entries in each level/ style. The entries in the competition summary link to the more detailed style summary pages. These style summary pages show for a given style/level the "coverage" of the competitors across all the events. This is useful in cases where a competition has multiple events within a level-style combination and might present "best of" awards for combined placements. These pages link the event names to the actual event drill down pages.

Currently the pages are not as hyper-linked as possible. It would be nicer if the names of partners were linked to allow quick jumping between view by leader and view

by follower and similar. These hyper-links are expected to be added in the near future in the post-academic development of the tool as the generation of statistic pages has been optimized to the point that adding a small amount of extra processing won't be too taxing on the CPU.

In most cases the output of the different methods are unchanged, only the method used to generate the output.

3.2.1 Dynamic Statistics Pages

The older version of SlidingDoors, following the general design model of "be as dynamic as possible" created each statistic page on view request. This version was therefore always up to date with respect to the database. This version was also "button" heavy, as "POST" was used instead of "GET." POST was used first to avoid name-mangling URL and also to keep these dynamic pages from being cached. This method however proved to be too computationally intensive for both client and server. The dynamic generation of every page on view time consumed massive amounts of resources for a partially populated competition. The button laden form caused some browsers to mis- render the page and run out of memory.

3.2.2 Static Statistic Pages

Following the redesign, the statistic pages were changed from fully dynamic to fully static, along with the rest of the design. Of course, the statistic pages can not be truly static and still keep up with changes to the database so the pages have to be rebuilt periodically. This also adds the complication of possibly allowing at least part of the SlidingDoors module of CIB to have write access to the web tree to create the statistic pages.

Rebuilt on DB change

The first approach taken here was to have the statistics pages regenerated on every database change. This approach ensures that the statistics are always up to date

and it appeared fast enough during most of the tests. However, on the team's old production server, once the competition size reached about 200 event entries, the complete rebuild began to take about 2-3 minutes, on a tuned database As this delay was happening on every page submit that involved updating the database, this proved to be a great inconvenience to the users and a change was required.

Rebuilt on "Commit"

The next method moved the rebuild from every database activity to only on a "Commit and Exit" button on the registration form being pressed. After the user presses the button, the pages will be rebuilt and the user will be placed at a "Thank you for registering" type page to let them know they are through with the process. So this moved the 2-3 minute delay on every page to a 2-3 minute delay once per registration, which was slightly better. However, the length of time continued to grow and soon was reaching 10-15 minutes of full CPU utilization. No locking was being performed, so if two users' registration commits overlapped the time was doubled and the machine was swapping itself to inactivity.

Rebuilt as Cron job

At this point the rebuilding of the statistic pages was removed from the realm of user initiated actions and placed into an automated (cron) script to rebuild the pages every hour. It was now possible for the data to be slightly out of date with respect to the database, but, with a simple note that recent registrations won't show up for a short while, most users seemed content. This solved the simultaneous submit problem without the need for a locking system. However the length of statistic generation time continued to grow until it was taking a little over an hour. Not only was this heavily loading the server, but it was also causing inconsistencies to appear in the statistics as the pages were built from the live database, not from a static snapshot.

However, at this point the team's new server was purchased and the problem resolved itself with more power. The full rebuild on the new machine took less than 3 minutes. The cron method was used for three competitions hosted in the fall of 2002

with no problems aside from a few complaints of non-instant registration confirmation.

Further refinements of Cron-based rebuilds

The cron based method has several logical improvements. The first improvement was making sure that the statistics did not get rebuilt needlessly. If a database last-updated-timestamp is available and can be compared to the timestamp on the statistic pages the script could exit early if the pages are currently up to date without taxing the CPU for the rebuild, which can become an important consideration when hosting multiple competitions on a single server with similar registration deadlines. The only question is what causes the database timestamp (a lockfile) to be updated. The decision reached was to only update the timestamp on "Commit and Exit" and to run the cron job much more frequently, for instance once a minute. In this manner the statistic pages get rebuild after every completed registration, but not in the middle of one. However, adding the possibly frequent rebuilds means that it is possible for two or more rebuilds to try to occur simultaneously which in extreme cases can heavily load the server almost to a DoS level. As a result a second lockfile is used to ensure that at most one rebuild happens at a time. The last-updated-timestamp will continue cause the cron job to trigger until it catches up, but without doubling up. This improvement is in place and is planned for use in the spring season of competitions.

The next improvements involves taking a snapshot of the database and parsing it into an internal structure for all the ensuing manipulations. At present each statistic page tends to be generated from one primary database query that often involves a high number of joins and sometimes has auxiliary queries embedded in the result set processing code. However, many of these queries are extremely similar and it might be possible to develop a cache-like system to avoid rerunning the almost identical queries on many pages. These improvements have begun by constructing a affiliation id number to name/ abbreviation lookup table. This eliminated the need for a inefficient pair of outer joins on almost all the other queries used. Similar look up tables have been constructed for competitor names. A final look up table is planned to map

couple numbers to their entered events.

The third technique would be to keep a log of changes since the last rebuild and only rebuild those pages, or even portions of pages, that have changed. If done right, this might allow rebuild to move back to on commit or earlier as the expected load would be significantly lower. However, it would require the name-based break-out pages to be completely changed. At present the application automatically splits the master list of competitors into break-out pages of roughly 20 couples per page, while trying to keep all competitors with the same initial letter of last name together. As a result, small changes in the number of people within a given "letter bin" can result in wide- scale shifts of the binning of letters to pages and could require rebuilding almost all of the break-out pages. One solution to this problem is to return to the method used by OpenImpetus was was pre-determined break-out pages, one per letter. While this eliminates the cascade effect it does mean that their will be many empty pages and many over-filled pages.

3.3 Schema Changes

At the time of writing there are 87 tables defined in the single competition database schema. Thirty of these tables are used by SlidingDoors in some capacity. Another 20 will be populated by the scheduling and scrutineering tools under development by other individuals. The remaining tables deal primarily with things of interest to either creating the printed program and competition information web-site, tracking finances and sponsorship, or dealing with non-competitive adjuncts to the competitions such as socials, workshops, and private lessons.

Twenty-one of the eighty-seven tables have been added to the schema during the development process, Four tables were added to further refine the allowed events given a selected package; six were added to deal with team matches and other situations were either multiple regular events are combined into one "super" event for scoring purposes or multiple couples dances as an entrant in a single event. Three were added to deal with scheduling concerns and three more to deal with creating proper

normalization of some tables that were getting cumbersome. The remainder of the new tables have varied purposes.

Nine tables have changed since the initial specs. Most of these were changed to add start times and estimated end times to support the scheduling application. Two were changes in the maximum length restrictions on given fields.

Seventeen tables have been marked for revision. In most cases these are places where the rules of normalization were broken, in the interest of efficiency, and the break has turned out to be less of a savings and more of a hassle. Some tables while still properly normalized could be split into two tables for easier use. Three tables are marked for possible removal, only one of which is in use, abbreviations. The abbreviations tables introduces significant overhead on almost all queries by requiring an outer join to perform the abbreviation lookup. Rather than consolidating all the abbreviations in one table, it would be better to move the abbreviation field to the tables that are contain the full text, notably styles, levels, styles_dances, and organizations. The rules, packages, levels series of tables all need extensive rework to provide a more intutive naming system as well as a more general purpose ruleset.

A full-scale scheme review and revision is planned following the implementation of scrutineering data inport to test the majority of the un-used tables.

Chapter 4

Lessons Learned

Web-programming paradigms, security considerations, database design, and version control were all topics that were explored over the course of the development. A few of the lessons that were not discussed in-line with design and implementation are presented here. This section also contains some anecdotal accounts of the acceptance of the CompInaBox project by the ballroom dance community at large.

4.1 Web-Programming

For all the hype over "web-enabled","web-technology", etc there is much less difference between web and "regular" programming than "web-programmers" claim. Over the course of the software rewrites more and more conventional software wisdom was embraced over the looser web-programming standard practices. Involvement in a PHP user community has strengthen this belief. Many people come to the various web language, PHP and ASP, for example, from a variety of backgrounds. Many are high school age or younger and many are graphic designers turned web-programmers.

Few, at least on the forum, were computer science or engineering trained. Most of these seem to start using the web-scripting language for doing minor tasks like templating or simple user accounts – typically by slightly modifying existing scripts and taking the code as a black box that they don't want to heavily modify. As they get more comfortable with it, they start to ask questions about the why's not the

how's of a given program and often develop very strong beliefs over the answers they get. Two areas where this progression were quite apparent were in the matter of hashing passwords for security and use of object oriented programming in PHP.

On the hashed password front, beginners are constantly warned away from the `crypt` function, which does a salted 3DES hash, throwing away input past the eighth character, in favor of the `md5` function which does a MD5 hash, using all the characters of the input, but is unsalted. They understand the insecurities of `crypt` and embrace `md5` and become the strongest anti-`crypt` proponents for a while and begin to believe that `MD5` is the end-all, be-all of encryption. When presented with a situation that should use symmetric, aka reversible, encryption technology they seek a way to re-mold the problem into a form that uses the asymmetric tool they've learned and state that other advice is "wrong." A similar dogmatic stance is taken when people suggest that they try `crypt` again, but invoke it in MD5 mode, which performs a salted, MD5 hash giving the best of both hashes with a single, built-in function call.

On the object oriented front, we see a similar behavior. People start off coding without classes, as the author did, either because they don't know about them or have heard that "web-programming" is different. Yet reusable code is still a worthwhile goal and is even more common in the web-environment than in stand-alone projects. The average web site developer is called upon to build dynamic sites with user accounts, shopping carts, etc on a fairly regular basis and most of these tools are almost interchangeable between web-sites, as numerous on-line tutorials point out. These tutorials develop a sample class for doing some common task. However, the object-oriented mind-set isn't taught and classes grow and grow and grow until each coder has their own single massive class that does everything often with many static functions — its become a library.

Some coders justify this with the perceived inefficiencies of PHP's object model, in that all object calls require an additional function call as opposed to a single call for a regular function. They don't want to slow down their code with nested classes, helper classes, and the like. However, none of the projects I've seen being developed would be adversely affect by this small additional overheard, in fact, I would suspect

they would never notice the difference. However, being conscientious programmers they avoid classes because they heard that classes are slow; they then spread this belief.

Most of the advice and tutorials on the web are written by these self-educated web-programmers with their biases and taken-for-granted beliefs and to a great extent appear to define what is "web-programming".

However in both cases of encryption and object-oriented programming, when the doubters are shown a simple use that achieves a needed goal; most of them convert instantly, becoming very vocal about their support of this "new way". These observations seem to imply that most of what the greater community of php developers consider "web-programming practices" are less formal rules and more "what worked for them", but slowly become rules when not challenged.

4.2 Development Tools

The services of SourceForge.net were used to provide bug-tracking, software release, project web-space, forums, etc. Overall, SourceForge.net provides a decent set of free services. While SourceForge.net provides version control software, this feature was not used during most of the project's development due to networking limitations of the primary development machine. A local version of the same tool (CVS) was used, however, and in the fall of 2002 the code moved to the CVS service on SourceForge.net when the initial development machine began to show signs of impending death.

Over the course of the project, 91 bugs were reported and 25 features requested through the SourceForge.net interface. Many of these were emails the author received and then manually entered into the system, however. Those that were entered directly by the user often lacked sufficient detail and didn't provide an email address for additional information requests. The bug reporting interface is also a little cluttered with too many selector boxes for the average user — one who is not an user-developer. Partially this is due to SourceForge.net role as an central point for Open Source developers to collaborate on projects. Most of the tools are written from a developers

standpoint for developers. However, most projects do not seem to maintain a separate bug tracker for user submitted versus developer submitted bugs. The Mantis Bug Tracker was also briefly investigated and while it is more fully features it suffers from most of the same problems – overly complex interface for reporting a bug, typically requiring a user account, though there are ways around that.

The experience with the version control software was only mediocre, but due mainly to the networking difficulties with the development laptop. Version history was maintained on the development laptop throughout the project. However, complete snapshots of the project were used to update either the testing or live site, and these upgrades took more time than they should have.

4.3 Acceptance by the Dance Community

As CompInaBox enters its post-academic role, it is hoped that other developers will join the project to help it grow to its full potential. It is also hoped that the project can attract the current developers of the competing registration software to eliminate the redundancy and let the joint developers focus on adding new tools, features, and robustness instead of re-implementing the same features over and over.

As part of this transition process, the CompInaBox effort was presented to the fall meeting of the YCN (Youth/College Network) Council of USABDA (United States Amateur Ballroom Dance Association that met in Cambridge in September. Attending were regional representatives for the YCN and some of the top executives of USABDA, including the association's president. Over the course of the one hour presentation the change in people's expressions was remarkable as they realized that CompInaBox is not just another registration tool, but an effort aimed at competition planning and execution in general. YCN and USABDA are both interested in helping further the development.

Shortly after the meeting, requests from the Carnegie Mellon University and University of Pennsylvania ballroom teams arrived for using CompInaBox at their spring competitions, adding to those from the Universities of Connecticut and Michigan,

Tufts University, and Yale University who used the software this fall. The University of Pennsylvania is planning to install a local copy for their spring competition and Boston University is planning to use the MIT server for their competition.

There have been a few tentative offers of assistance with development, but most have not materialized past a few emails exchanged. However, Chen Ling, who wrote the tool used in the DC area, has asked to officially join the project and is expected to take an active role after the close of the Ohio Star Ball that he is hosting with his software.

4.4 Conclusion

While the author had developed several moderate size programs before, none were of the complexity, size, or scope of this project. Bringing a project from conception to prototype to production model and fielding has been a very enlightening and rewarding experience. Especially useful were the explorations of web-script based utilities versus back-end cron controlled processes as well, development of a security model, and optimization work on the statistic page generation process. Future development on CompInaBox and SlidingDoors promises to continue growth as a software developer.

Appendix A

Database Schemas

A.1 Per Compeitition Database aka "Ragu"

```
-------------------------------------------------------------------
-- This file is part of CompInaBox.                              --
-- Copyright 2002. Eric D. Nielsen                               --
-- CompInaBox is availible for license under the GPL, see        --
-- the COPYING file in the root directory of the install         --
-- for the full terms of GPL.                                    --
--                                                               --
-- This file creates the CompInaBox comp databases used          --
-- by each competition hosted on the CIB server.                 --
-------------------------------------------------------------------
CREATE USER ''OIuser'';
CREATE USER ''PrepStepUser'';
CREATE TABLE abbreviations (
    fulltext     VARCHAR(80) NOT NULL PRIMARY KEY,
    abbreviation VARCHAR(10)
);
REVOKE ALL ON abbreviations FROM public;
GRANT INSERT,UPDATE,SELECT ON abbreviations TO ''OIuser'';
GRANT INSERT,UPDATE,SELECT ON abbreviations TO ''PrepStepUser'';

CREATE TABLE addresses (
    addressid SERIAL NOT NULL PRIMARY KEY,
    street1   VARCHAR(80),
    street2   VARCHAR(80),
    city      VARCHAR(40),
    state     VARCHAR(40),
    zip_code  VARCHAR(10),
    country   VARCHAR(40),
    type      VARCHAR(10)
);
REVOKE ALL ON addresses FROM public;
GRANT INSERT, UPDATE, SELECT ON address                TO ''OIuser'';
GRANT UPDATE,SELECT          ON address_addressid_seq TO ''OIuser'';

CREATE TABLE booths (
    boothid       SERIAL NOT NULL PRIMARY KEY,
```

```
    owner           INTEGER REFERENCES organizations
      ON DELETE CASCADE ON UPDATE CASCADE,
    spacereq        INTEGER,
    socketsneeded   INTEGER,
    name            VARCHAR(80),
    blurbmarkup     VARCHAR(10),
    blurb           TEXT
);
REVOKE ALL ON booths FROM public;

CREATE TABLE booths_locations (
    boothid         INTEGER NOT NULL REFERENCES booths
        ON DELETE CASCADE ON UPDATE CASCADE,
    date            date,
    roomid          INTEGER REFERENCES rooms
        ON DELETE RESTRICT ON UPDATE CASCADE,
    open            time,
    close           time,
    spacegranted    INTEGER,
    socketsavail    INTEGER,
    location        VARCHAR(80),
    PRIMARY KEY (boothid,date,room)
);
REVOKE ALL ON booths_locations FROM public;

CREATE TABLE buildings (
    buildingid  SERIAL NOT NULL PRIMARY KEY,
    addressid   INTEGER REFERENCES addresses
        ON DELETE SET NULL ON UPDATE CASCADE,
    floorplan   INTEGER REFERENCES images
        ON DELETE SET NULL ON UPDATE CASCADE,
    map         INTEGER REFERENCES images
        ON DELETE SET NULL ON UPDATE CASCADE,
    name        VARCHAR(40),
    markup      VARCHAR(10),
    directions  TEXT
);
REVOKE ALL ON buildings FROM public;

CREATE TABLE buildings_maps (
    buildingid INTEGER REFERENCES buildings,
    mapid      INTEGER REFERENCES images,
    PRIMARY KEY (buildingid, mapid)
);
REVOKE ALL ON buildings_maps FROM public;

CREATE TABLE comp (
    compid          INTEGER NOT NULL PRIMARY KEY,
    venue           INTEGER REFERENCES buildings
        ON DELETE RESTRICT ON UPDATE CASCADE,
    hostingorg      INTEGER REFERENCES organizations
        ON DELETE RESTRICT ON UPDATE CASCADE,
    director        INTEGER REFERENCES people
        ON DELETE RESTRICT ON UPDATE CASCADE,
    webmaster       INTEGER REFERENCES people
        ON DELETE RESTRICT ON UPDATE CASCADE,
    registrar       INTEGER REFERENCES people
        ON DELETE RESTRICT ON UPDATE CASCADE,
    comp_logo_large INTEGER REFERENCES images
        ON DELETE SET NULL ON UPDATE CASCADE,
    comp_logo_small INTEGER REFERENCES images
```

```
        ON DELETE SET NULL ON UPDATE CASCADE,
    name            VARCHAR(80),
    url             VARCHAR(80)
)
REVOKE ALL ON comp FROM public;
GRANT SELECT ON comp TO ''OIuser'';
GRANT SELECT,INSERT,UPDATE ON comp TO ''PrepStepUser'';


CREATE TABLE comp_shifts (
    shiftid     INTEGER NOT NULL,
    worker      INTEGER REFERENCES people
        ON DELETE SET NULL ON UPDATE CASCADE,
    start       timestamp,
    estend      timestamp,
    reportsto   INTEGER REFERENCES people
        ON DELETE RESTRICT ON UPDATE CASCADE,
    description VARCHAR(80),
    PRIMARY KEY (shiftid,worker)
);
REVOKE ALL ON comp_shifts FROM public;

CREATE TABLE comp_times (
    day         date NOT NULL PRIMARY KEY,
    doorsOpen   time,
    eventsstart time,
    estclose    time
);
REVOKE ALL ON comp_times FROM public;

CREATE TABLE couples
   (coupleid SERIAL NOT NULL PRIMARY KEY,
    leader   INTEGER REFERENCES people
        ON DELETE SET NULL ON UPDATE CASCADE,
    follower INTEGER REFERENCES people
        ON DELETE SET NULL ON UPDATE CASCADE
);
REVOKE ALL ON couples FROM public;
GRANT SELECT,INSERT,UPDATE,DELETE ON couples TO "OIuser";
GRANT UPDATE          ON couples_coupleid_seq TO "OIuser";
GRANT SELECT,DELETE   ON couples              TO "PrepStepUser";

CREATE TABLE dance_heats (
    heatid          SERIAL NOT NULL PRIMARY KEY,
    atomicroundid INTEGER NOT NULL REFERENCES round_dances
        ON DELETE CASCADE ON UPDATE CASCADE
    floorid         INTEGER REFERENCES floors
        ON DELETE RESTRICT ON UPDATE CASCADE,
    heatnumber      INTEGER,
    starttime       timestamp,
    endtime         timestamp,
    shared_floor    BOOLEAN
);
REVOKE ALL ON dance_heats FROM public;

CREATE TABLE dance_marks (
    atomicroundid INTEGER NOT NULL REFERENCES round_dances
        ON DELETE RESTRICT ON UPDATE CASCADE,
    judge           INTEGER REFERENCES people
        ON DELETE RESTRICT ON UPDATE CASCADE,
    couple          INTEGER REFERENCES couples
```

```
        ON DELETE RESTRICT ON UPDATE CASCADE,
    mark            CHAR(1),
    PRIMARY KEY (atomicroundid,judge,couple)
);
REVOKE ALL on dance_marks FROM public;

CREATE TABLE events (
    eventid         SERIAL NOT NULL PRIMARY KEY,
    programnumber INTEGER,
    categoryid      INTEGER REFERENCES events_categories
       ON DELETE RESTRICT ON UPDATE CASCADE,
    starttime       timestamp,
    estendtime      timestamp
    estawardtime    timestamp,
    specrules       VARCHAR(80)
);
REVOKE ALL ON EVENTS FROM public;
GRANT SELECT ON                         events TO "OIuser";
GRANT INSERT,SELECT,UPDATE,DELETE ON events TO "PrepStepUser";
GRANT UPDATE ON             events_eventid_seq TO "PrepStepUser";

CREATE TABLE events_announced (
    eventid INTEGER NOT NULL PRIMARY KEY REFERENCES events
       ON DELETE RESTRICT ON UPDATE CASCADE,
    printed BOOLEAN
);
REVOKE ALL ON events_announced FROM public;

CREATE TABLE events_categories (
    categoryID    SERIAL NOT NULL PRIMARY KEY,
    class           VARCHAR(20),
    level           VARCHAR(20),
    style           VARCHAR(20),
    age             VARCHAR(20),
    type            VARCHAR(20),
    costumerules TEXT,
    syllabirules TEXT
);
REVOKE ALL ON events_categories FROM public;
GRANT SELECT ON                         events_categories TO "OIuser";
GRANT SELECT,UPDATE,DELETE,INSERT ON events_categories TO "PrepStepUser";
GRANT UPDATE ON          events_categorie_categoryid_seq TO "PrepStepUser";

CREATE TABLE events_dances (
    eventid     INTEGER NOT NULL REFERENCES events
        ON DELETE CASCADE ON UPDATE CASCADE,
    danceorder INTEGER,
    dancename  VARCHAR(20),
    PRIMARY KEY (eventid, dancename)
);
REVOKE ALL ON events_dances FROM public;
GRANT SELECT ON                         events_dances TO "OIuser";
GRANT SELECT,UPDATE,DELETE,INSERT ON events_dances TO "PrepStepUser";


CREATE TABLE events_est_reg (
    eventid INTEGER NOT NULL PRIMARY KEY REFERENCES events
        ON DELETE CASCADE ON UPDATE CASCADE,
    estreg   INTEGER,
    curreg   INTEGER
);
```

```
REVOKE ALL ON events_est_reg FROM public;
GRANT SELECT, UPDATE ON events_est_reg TO "OIuser";
GRANT SELECT,UPDATE,INSERT,DELETE ON events_est_reg TO "PrepStepUser";

CREATE TABLE events_registration (
    eventid          INTEGER NOT NULL REFERENCES events
        ON DELETE CASCADE ON UPDATE CASCADE,
    coupleid         INTEGER NOT NULL REFERENCES couples
        ON DELETE CASCADE ON UPDATE CASCADE,
    competitornumber VARCHAR(4),
    PRIMARY KEY (eventid,coupleid)
);
REVOKE ALL ON events_registration FROM public;
GRANT SELECT, INSERT, UPDATE, DELETE ON events_registration TO "OIuser";
GRANT SELECT,DELETE,INSERT,UPDATE ON events_registration TO "PrepStepUser";

CREATE TABLE events_results (
    eventid      INTEGER NOT NULL REFERENCES events
        ON DELETE RESTRICT ON UPDATE CASCADE,
    coupleid     INTEGER NOT NULL REFERENCES couples,
        ON DELETE restict ON UPDATE CASCADE,
    place        INTEGER,
    uptoFirst    VARCHAR(10),
    uptoSecond   VARCHAR(10),
    uptoThird    VARCHAR(10),
    uptoFourth   VARCHAR(10),
    uptoFifth    VARCHAR(10),
    uptoSixth    VARCHAR(10),
    uptoSeventh VARCHAR(10),
    uptoEighth   VARCHAR(10),
    uptoNinth    VARCHAR(10),
    uptoTenth    VARCHAR(10),
    ruleused     TEXT,
    PRIMARY KEY (eventid, coupleid)
);
REVOKE ALL ON events_results FROM public;

CREATE TABLE events_rounds (
    roundid    SERIAL NOT NULL PRIMARY KEY,
    eventid    INTEGER NOT NULL REFERENCES events
        ON DELETE CASCADE ON UPDATE CASCADE,
    cutnumber INTEGER NOT NULL,
    session    INTEGER REFERENCES sessions,
    starttime time,
    roundname VARCHAR(40)
);
REVOKE ALL ON events_rounds FROM public;

CREATE TABLE expenditure (
    expenditureid SERIAL NOT NULL PRIMARY KEY,
    amount          float,
    purchasedby     INTEGER REFERENCES people
        ON DELETE RESTRICT ON UPDATE CASCADE,
    reimbursed      date,
    number          INTEGER,
    countsas        INTEGER REFERENCES expenditure_budget
        ON DELETE RESTRICT ON UPDATE CASCADE,
    item            VARCHAR(40),
    means           VARCHAR(20)
);
REVOKE ALL ON expenditure FROM public;
```

```
CREATE TABLE expenditure_budget (
    expenditureitemid SERIAL NOT NULL PRIMARY KEY,
    amount            float,
    category          VARCHAR(40)
);
REVOKE ALL ON expenditure_budget FROM public;

CREATE TABLE floors (
    floorid  SERIAL NOT NULL PRIMARY KEY,
    room     INTEGER REFERENCES rooms
        ON DELETE RESTRICT ON UPDATE CASCADE,
    location VARCHAR(40),
    type     VARCHAR(20)
);
REVOKE ALL ON floors FROM public;

CREATE TABLE floor_capacity (
    category    INTEGER REFERENCES events_categories
        ON DELETE RESTRICT ON UPDATE CASCADE,
    numcouples INTEGER,
    dancename  VARCHAR(20),
    type       VARCHAR(20),
    PRIMARY KEY (category,type)
);
REVOKE ALL ON floor_capacity FROM public;

CREATE TABLE heat_couples (
    heatid INTEGER NOT NULL REFERENCES dance_heats
       ON DELETE CASCADE ON UPDATE CASCADE,
    coupleid INTEGER reference couples,
    PRIMARY KEY (heatid,coupleid)
);
REVOKE ALL ON heat_couples FROM public;

CREATE TABLE images (
    imageid     SERIAL NOT NULL PRIMARY KEY,
    filename    VARCHAR(80),
    description VARCHAR(80)
);
REVOKE ALL ON images FROM public;

CREATE TABLE income (
    incomeid    SERIAL NOT NULL PRIMARY KEY,
    amount      FLOAT,
    number      INTEGER,
    paidby      INTEGER REFERENCES people
        ON DELETE RESTRICT ON UPDATE CASCADE,
    countsas    INTEGER REFERENCES income_budget
        ON DELETE RESTRICT ON UPDATE CASCADE,
    received    date,
    deposited   date,
    subsource   VARCHAR(40),
    means       VARCHAR(20)
);
REVOKE ALL ON income FROM public;

CREATE TABLE income_budget (
    incomeitemid SERIAL NOT NULL PRIMARY KEY,
    amount       FLOAR,
```

```
    source          VARCHAR(40)
);
REVOKE ALL ON income_budget FROM public;

CREATE TABLE judges_info
    (judgeid              INTEGER NOT NULL PRIMARY KEY REFERENCES people
        ON DELETE CASCADE ON UPDATE CASCADE,
     honorariumAd         INTEGER REFERENCES images,
     timestyle            char(1),
     abbrev               VARCHAR(2),
     foodpref             VARCHAR(80),
     foodrestrict         VARCHAR(80),
     beveragerestrict     VARCHAR(80),
     misc                 VARCHAR(80),
     qualstyles           VARCHAR(80),
     availtimes           VARCHAR(80),
     prefstyles           VARCHAR(80),
     preftimes            VARCHAR(80),
     lodgingpref          VARCHAR(80),
     markup               VARCHAR(5),
     bio                  TEXT
);
REVOKE ALL ON judges_info FROM public;

CREATE TABLE local_food (
     foodid      SERIAL NOT NULL PRIMARY KEY,
     resturant INTEGER REFERENCES organizations
         ON DELETE CASCADE ON UPDATE CASCADE,
     costlow     INTEGER,
     costhigh    INTEGER,
     distance    FLOAT,
     seatingavailible INTEGER,
     takeout     BOOLEAN,
     delivery    BOOLEAN,
     type        VARCHAR(80)
);
REVOKE ALL ON local_food FROM public;

CREATE TABLE local_lodging (
     lodgeid      SERIAL NOT NULL PRIMARY KEY,
     hotel        INTEGER REFERENCES organizations
        ON DELETE CASCADE ON UPDATE CASCADE,
     costlow      INTEGER,
     costhigh     INTEGER,
     distance     FLOAT,
     capacity     INTEGER,
     ammenities   VARCHAR(80)
);
REVOKE ALL ON local_lodgings FROM public;

CREATE TABLE local_lodging_booked (
     lodgebookedid SERIAL PRIMARY KEY,
     personid        INTEGER REFERENCES people,
     lodgeid         INTEGER REFERENCES local_lodging,
     bedassigned     VARCHAR(20)
);
REVOKE ALL ON local_lodging FROM public;

CREATE TABLE local_lodging_offered (
     lodgeofferedid SERIAL PRIMARY KEY,
     offeror          INTEGER REFERENCES people
```

```
        ON DELETE RESTRICT ON UPDATE CASCADE,
    building        INTEGER REFERENCES buidlings
        ON DELETE RESTRICT ON UPDATE CASCADE,
    capacity        INTEGER,
    distance        FLOAT,
    gender          CHAR(1),
    room            VARCHAR(40)
);
REVOKE ALL ON local_lodging_offered;

CREATE TABLE music_guidelines (
    category    INTEGER NOT NULL REFERENCES events_categories
        ON DELETE RESTRICT ON UPDATE CASCADE,
    mintempo    INTEGER,
    maxtempo    INTEGER,
    length      INTEGER,
    dancename VARCHAR(20),
    PRIMARY KEY (category,dancename)
);
REVOKE ALL ON music_guidelines FROM public;

CREATE TABLE organizations (
    orgid           SERIAL NOT NULL PRIMARY KEY,
    building        INTEGER REFERENCES buildings
        ON DELETE SET NULL ON UPDATE CASCADE,
    representative INTEGER REFERENCES people
        ON DELETE RESTRICT ON UPDATE CASCADE,
    org_logo         INTEGER REFERENCES images
        ON DELETE SET NULL ON UPDATE CASCADE,
    name            VARCHAR(80),
    rep_title       VARCHAR(80)
    url             VARCHAR(80)
);
REVOKE ALL ON organizations FROM public;
GRANT INSERT,UPDATE,SELECT ON organizations TO "OIuser"
GRANT INSERT,UPDATE,DELETE,SELECT ON organizations TO "PrepStepUser";

CREATE TABLE packages (
    packageid       SERIAL NOT NULL PRIMARY KEY,
    unlimevents     BOOLEAN,
    numevents       INTEGER,
    unlimsocials    BOOLEAN,
    numsocials      INTEGER,
    unlimworkshops BOOLEAN,
    numworkshops    INTEGER,
    timespan        INTERVAL,
    name            VARCHAR(80)
);
REVOKE ALL ON packages FROM public;
GRANT SELECT ON pacakges TO "OIuser";
GRANT SELECT,INSERT,UPDATE,DELETE ON packages to "PrepStepUser";
GRANT UPDATE ON packages_packageid_seq TO "PrepStepUser";

CREATE TABLE packages_cost (
    packageid INTEGER REFERENCES packages,
    cost        FLOAT,
    category    VARCHAR(80),
    PRIMARY KEY (packageid, category)
);
REVOKE ALL ON packages_cost FROM public;
GRANT SELECT ON packages_cost TO "OIuser";
```

```
GRANT SELECT,INSERT,UPDATE,DELETE ON pakcages_cost To "PrepStepUser";

CREATE TABLE packages_purchased (
    packageid  INTEGER NOT NULL REFERENCES packages
        ON DELETE RESTRICT ON UPDATE CASCADE,
    personid      INTEGER NOT NULL REFERENCES people
        ON DELETE CASCADE ON UPDATE CASCADE.
    PRIMARY KEY (packageid, personid)
);
REVOKE ALL ON packages_purchaed FROM public;
GRANT SELECT,INSERT,UPDATE,DELETE ON packages_purchased TO "OIuser";
GRANT SELECT,INSERT,UPDATE,DELETE ON pakcages_purchased To "PrepStepUser";

CREATE TABLE people (
    peopleid       SERIAL NOT NULL PRIMARY KEY,
    firstname      VARCHAR(20),
    middleinitial VARCHAR(1),
    lastname       VARCHAR(40),
    organization   INTEGER REFERENCES organzations,
    address        INTEGER REFERENCES addresses
        ON DELETE SET NULL ON UPDATE CASCADE
);
REVOKE ALL ON people FROM public;
GRANT INSERT,UPDATE,DELETE,SELECT ON people TO "OIuser";
GRANT UPDATE ON people_peopleid_seq TO "OIuser";
GRANT INSERT,UPDATE,DELETE,SELECT ON people TO "PrepStepUser";
GRANT UPDATE ON people_peopleid_seq TO "PrepStepUser";

CREATE TABLE people_contact (
    peopleid INTEGER PRIMARY KEY REFERENCES people
        ON DELETE CASCADE ON UPDATE CASCADE,
    phoneday VARCHAR(30),
    phoneeve VARCHAR(30),
    phonefax VARCHAR(30),
    emailday VARCHAR(80),
    emaileve VARCHAR(80)
);
REVOKE ALL ON people_contact FROM public;
GRANT INSERT,UPDATE,SELECT,DELETE ON people_contact TO "OIuser";
GRANT INSERT,UPDATE,SELECT,DELETE ON people_contact TO "PrepStepUser";

CREATE TABLE people_classification(
    peopleid  INTEGER REFERENCES people
        ON DELETE CASCADE ON UPDATE CASCADE,
    agelevel  VARCHAR(20),
    class     VARCHAR(20)
);
REVOKE ALL ON people_classification FROM public;
GRANT INSERT,UPDATE,SELECT,DELETE ON people_classification TO "OIuser";
GRANT INSERT,UPDATE,SELECT,DELETE ON people_classification TO "PrepStepUser";

CREATE TABLE people_id_numbers (
    peopleid      INTEGER REFERENCES people
        ON DELETE CASCADE ON UPDATE CASCADE,
    organization INTEGER REFERENCES organizations,
    idnumber      VARCHAR(20),
    PRIMARY KEY (peopleid, organization)
);
REVOKE ALL ON people_id_numbers FROM public;
GRANT INSERT,UPDATE,SELECT,DELETE ON people_id_numbers TO "OIuser";
GRANT INSERT,UPDATE,SELECT,DELETE ON people_id_numbers TO "PrepStepUser";
```

```
CREATE TABLE people_roles (
    peopleid INTEGER REFERENCES people
        ON DELETE CASCADE ON UPDATE CASCADE,
    role     VARCHAR(20),
    PRIMARY KEY (peopleid, role)
);
REVOKE ALL ON people_roles FROM public;

CREATE TABLE privates (
    privateid  INTEGER NOT NULL PRIMARY KEY,
    instructor INTEGER REFERENCES people
        ON DELETE CASCADE ON UPDATE CASCADE,
    session    INTEGER REFERENCES sessions,
    start      timestamp,
    end        timestamp,
    room       INTEGER REFERENCES rooms
        ON DELETE RESTRICT ON UPDATE CASCADE
);
REVOKE ALL ON privates FROM public;

CREATE TABLE private_booked (
    privateid INTEGER NOT NULL PRIMARY KEY REFERENCES privates
        ON DELETE RESTRICT ON UPDATE CASCADE,
    peopleid INTEGER NOT NULL REFERENCES people
        ON DELETE RESTRICT ON UPDATE CASCADE,
    PRIMARY KEY (privateid, peopleid)
);
REVOKE ALL ON privates_booked FROM public;

CREATE TABLE privates_costs (
    instructor INTEGER NOT NULL REFERENCES people
        ON DELETE CASCADE ON UPDATE CASCADE,
    category   VARCHAR(20),
    cost       FLOAT,
    PRIMARY KEY (instructor, category)
);
REVOKE ALL ON privates_costs FROM public;

CREATE TABLE rooms (
    roomid   INTEGER NOT NULL PRIMARY KEY,
    building INTEGER REFERENCES buildings
        ON DELETE CASCADE ON UPDATE CASCADE,
    maxocc   INTEGER,
    sockets  INTEGER,
    mapint   INTEGER REFERENCES images
        ON DELETE SET NULL ON UPDATE CASCADE,
    mapext   INTEGER reference images
        ON DELETE SET NULL ON UPDATE CASCADE,
    name     VARCHAR(40),
    number   VARCHAR(20)
);
REVOKE ALL ON rooms FROM public;

CREATE TABLE round_dances (
    atomicroundid INTEGER NOT NULL PRIMARY KEY,
    roundid       INTEGER REFERENCES events_rounds
        ON DELETE CASCADE ON UPDATE CASCADE,
    music         INTEGER REFERENCES songs
        ON DELETE RESTRICT ON UPDATE CASCADE,
    starttime     timestamp,
```

```
    endtime         timestamp,
    dancename       VARCHAR(20)
);
REVOKE ALL ON round_dances FROM public;

CREATE TABLE round_invigilating_panel (
    atomicroundid INTEGER REFERENCES round_dances
        ON DELETE CASCADE ON UPDATE CASCADE,
    invigilator INTEGER REFERENCES people
        ON DELETE RESTRICT ON UPDATE CASCADE,
    PRIMARY KEY (atomicroundid, invigilator)
);
REVOKE ALL ON round_invigilating_panel FROM public;

CREATE TABLE round_judging_panel (
    atomicroundid INTEGER REFERENCES round_dances
        ON DELETE CASCADE ON UPDATE CASCADE,
    judge INTEGER REFERENCES people
        ON DELETE RESTRICT ON UPDATE CASCADE,
    PRIMARY KEY (atomicroundid, judge)
);
REVOKE ALL ON round_judging_panel FROM public;

CREATE TABLE rules_age (
   (ruleid    INTEGER NOT NULL PRIMARY KEY,
    cutoff    INTEGER REFERENCES rules_age_level
        ON DELETE RESTRICT ON UPDATE CASCADE,
    agemod    char(1),
    levelmod  char(1),
    relationship VARCHAR(2)
);
REVOKE ALL ON rules_age FROM public;

CREATE TABLE rules_age_level
    index INTEGER NOT NULL PRIMARY KEY,
    low   INTEGER,
    high  INTEGER,
    name  VARCHAR(20)
);
REVOKE ALL ON rules_age_level FROM public;
GRANT SELECT ON rules_age_level TO "OIuser";
GRANT SELECT, INSERT, DELETE ON rules_age_level TO "PrepStepUser";

CREATE TABLE rules_eligibility
    event INTEGER PRIMARY KEY REFERENCES events
        ON DELETE CASCADE ON UPDATE CASCADE,
    points INTEGER,
    date   date
);
REVOKE ALL ON rules_eligibility FROM public;

CREATE TABLE rules_level (
    index INTEGER NOT NULL,
    date          date,
    levelthere  VARCHAR(20),
    PRIMARY KEY (date, levelthere)
);
REVOKE ALL ON rules_level FROM public;

CREATE TABLE rules_level_names (
    index INTEGER PRIMARY KEY,
```

```
    name  VARCHAR(20)
);
REVOKE ALL ON rules_level_names FROM public;
GRANT SELECT ON rules_level_names TO "OIuser";
GRANT SELECT,INSERT,DELETE ON rules_level_names TO "PrepStepUser";

CREATE TABLE rules_points (
    place       INTEGER NOT NULL PRIMARY KEY,
    points      INTEGER,
    closeness   char(1),
    leveldiff   VARCHAR(2),
    size        VARCHAR(20)
);
REVOKE ALL ON rules_points FROM public;

CREATE TABLE rules_text (
    level       INTEGER NOT NULL PRIMARY KEY REFERENCES rules_level
        ON DELETE RESTRICT ON UPDATE CASCADE,
    markup      VARCHAR(5),
    rule        TEXT
);
REVOKE ALL ON rules_text FROM public;

CREATE TABLE sessions (
    sessionid INTEGER NOT NULL PRIMARY KEY,
    start       timestamp,
    estend      timestamp,
    name        VARCHAR(80)
);
REVOKE ALL ON session FROM public;

CREATE TABLE socials (
    socialID INTEGER NOT NULL PRIMARY KEY,
    session INTEGER REFERENCES sessions,
    start       timestamp,
    end         timestamp,
    location INTEGER REFERENCES rooms
        ON DELETE RESTRICT ON UPDATE CASCADE,
    poc         INTEGER REFERENCES people
        ON DELETE RESTRICT ON UPDATE CASCADE,
    name        VARCHAR(80)
);
REVOKE ALL ON socials FROM public;

CREATE TABLE songs (
    songid      INTEGER NOT NULL PRIMARY KEY,
    track       INTEGER,
    mpm         INTEGER,
    pitchadjust FLOAT,
    offset      FLOAT,
    title       VARCHAR(40),
    artist      VARCHAR(40),
    cd          VARCHAR(40),
    mp3         VARCHAR(80)
);
REVOKE ALL ON songs FROM public;

CREATE TABLE sponsors (
    sponsorid INTEGER NOT NULL PRIMARY KEY,
    sponsor   INTEGER REFERENCES organizations
        ON DELETE RESTRICT ON UPDATE CASCADE,
```

```
    person     INTEGER reference people,
    advert INTEGER REFERENCES images
        ON DELETE RESTRICT ON UPDATE CASCADE,
    level VARCHAR(20)
);
REVOKE ALL ON sponsors FROM public;

CREATE TABLE workshops (
    workshopid INTEGER NOT NULL PRIMARY KEY,
    room INTEGER REFERENCES rooms
        ON DELETE RESTRICT ON UPDATE CASCADE,
    session INTEGER REFERENCES sessions,
    start timestamp,
    end   timestamp,
    number INTEGER,
    estreg INTEGER,
    curreg INTEGER,
    name VARCHAR(40)
);
REVOKE ALL ON workshops FROM public;

CREATE TABLE workshops_instructors (
    workshopid INTEGER NOT NULL REFERENCES workshops
        ON DELETE CASCADE ON UPDATE CASCADE,
    instructor INTEGER NOT NULL REFERENCES people
        ON DELETE RESTRICT ON UPDATE CASCADE,
    PRIMARY KEY (workshopid,instructor)
);
REVOKE ALL ON workshops_instructors FROM public;

CREATE TABLE workshops_registration (
    workshopid INTEGER NOT NULL PRIMARY KEY REFERENCES workshops
        ON DELETE RESTRICT ON UPDATE CASCADE,
    peopleid INTEGER NOT NULL REFERENCES people
        ON DELETE RESTRICT ON UPDATE CASCADE,
    PRIMARY KEY (workshopid,peopleid)
);
REVOKE ALL ON workshops_registration FROM public;
```

A.2 CIB Central Database

```
-- This file is part of CompInaBox.                                    --
-- Copyright 2002. Eric D. Nielsen                                     --
-- CompInaBox is availible for license under the GPL, see              --
-- the COPYING file in the root directory of the install               --
-- for the full terms of GPL.                                          --
--                                                                     --
-- This file creates the CompInaBox central database used              --
-- chiefly for authentication of users and for providing               --
-- navigation to the main areas of this server.                        --
--                                                                     --
-- :TODO: automate creation of required users                          --
--                                                                     --
-- This file requires the following postgres users:                    --
-- cib_unauth_user: used for browsing, creating users                  --
-- cib_auth_user:   used for creating/adminning comps                  --
-- cib_admin_user:  used for adminning the site                        --
--                                                                     --
-- These users do not need permission to create users or               --
-- new databases.  These accounts should be password                   --
-- protected.  The passwords should be added to                        --
-- compinabox/include/cib_constants.inc, if you change                 --
-- the account names, be sure to change them in the above              --
-- mentioned file as well.                                             --
-----------------------------------------------------------------------

-- Authentication groups
-- Created and populated in two steps on purpose.  If the group already
-- exists a one-step create and populate won't ensure the users are added.
CREATE GROUP cib_all;
CREATE GROUP cib_trusted;
ALTER GROUP cib_all ADD USER cib_unauth_user, cib_auth_user, cib_admin_user;
ALTER GROUP cib_trusted ADD USER              cib_auth_user, cib_admin_user;


-- Core tables to operation of CompInaBox

CREATE TABLE comp_status (
    statusid SERIAL PRIMARY KEY,
    name     TEXT);
REVOKE ALL PRIVILEGES       ON                 comp_status FROM PUBLIC;
GRANT SELECT                ON                 comp_status TO GROUP cib_all;
GRANT INSERT,UPDATE,DELETE ON                  comp_status TO cib_admin_user;
GRANT UPDATE                ON comp_status_statusid_seq TO cib_admin_user;
INSERT INTO comp_status (name) VALUES ('CONFIGURATION');
INSERT INTO comp_status (name) VALUES ('PENDING_ADMIN');
INSERT INTO comp_status (name) VALUES ('PLANNING');
INSERT INTO comp_status (name) VALUES ('TESTING');
INSERT INTO comp_status (name) VALUES ('OPEN');
INSERT INTO comp_status (name) VALUES ('CLOSED');
INSERT INTO comp_status (name) VALUES ('RUNNING');
INSERT INTO comp_status (name) VALUES ('POST-COMP');
INSERT INTO comp_status (name) VALUES ('ARCHIVE');

CREATE TABLE comps (
    compid    SERIAL PRIMARY KEY,
    statusid  INT REFERENCES comp_status ON UPDATE CASCADE ON DELETE RESTRICT,
    realcomp  BOOLEAN,
    occurs_on TIMESTAMP,
    compname  TEXT,
```

```
    compunix  TEXT);
CREATE INDEX comps_multi_idx ON comps (realcomp,statusid,compname);
REVOKE ALL PRIVILEGES ON              comps FROM PUBLIC;
GRANT SELECT          ON              comps TO GROUP cib_all;
GRANT INSERT,UPDATE   ON              comps TO GROUP cib_trusted;
GRANT UPDATE          ON comps_compid_seq TO GROUP cib_trusted;
GRANT DELETE          ON              comps TO cib_admin_user;

-- Tables relating to the security system
-- Roles define the access-levels, tools are responsible for enforcing their
-- security model
CREATE TABLE roles (
    roleid      SERIAL PRIMARY KEY,
    rolename    TEXT,
    description TEXT);
REVOKE ALL PRIVILEGES       ON              roles FROM PUBLIC;
GRANT SELECT                ON              roles TO GROUP cib_all;
GRANT INSERT,UPDATE,DELETE ON               roles TO cib_admin_user;
GRANT UPDATE                ON roles_roleid_seq TO cib_admin_user;
INSERT INTO roles (rolename) VALUES ('Competition_Coordinator');
INSERT INTO roles (rolename) VALUES ('Competition_Registrar');
INSERT INTO roles (rolename) VALUES ('Competition_Comptroller');
INSERT INTO roles (rolename) VALUES ('Competition_Staff');
INSERT INTO roles (rolename) VALUES ('Affiliation_Coordinator');

create TABLE user_status(
    statusid SERIAL PRIMARY KEY,
    name     TEXT);
REVOKE ALL PRIVILEGES ON user_status FROM PUBLIC;
GRANT SELECT          ON user_status TO GROUP cib_all;
INSERT INTO user_status (name) VALUES ('PENDING');
INSERT INTO user_status (name) VALUES ('ACTIVE');
INSERT INTO user_status (name) VALUES ('BANNED');
INSERT INTO user_status (name) VALUES ('DELETED');

-- NOTE: Email should be stored encrypted with a two way method, to minimize
-- the chance of inadvertant leakage, email is used seldomly so the added
-- processor expense shouldn't be a problem
CREATE TABLE users (
    userid     SERIAL PRIMARY KEY,
    statusid   INT REFERENCES user_status ON UPDATE CASCADE ON DELETE CASCADE,
    siteadmin  BOOLEAN,
    username   TEXT,
    hashed     TEXT,
    email      TEXT);
REVOKE ALL PRIVILEGES ON              users FROM PUBLIC;
GRANT SELECT,INSERT   ON              users TO GROUP cib_all;
GRANT UPDATE          ON users_userid_seq TO GROUP cib_all;
GRANT UPDATE          ON              users TO GROUP cib_trusted;
GRANT DELETE          ON              users TO cib_admin_user;

-- Note that role based permission are granted on a per comp basis
CREATE TABLE user_roles (
   user_roleid SERIAL PRIMARY KEY,
   userid      INT    REFERENCES users ON UPDATE CASCADE ON DELETE CASCADE,
   compid      INT    REFERENCES comps ON UPDATE CASCADE ON DELETE CASCADE,
   roleid      INT    REFERENCES roles ON UPDATE CASCADE ON DELETE CASCADE);
REVOKE ALL PRIVILEGES       ON                   user_roles FROM PUBLIC;
GRANT SELECT                ON                   user_roles TO GROUP cib_all;
GRANT INSERT,UPDATE,DELETE ON                    user_roles TO GROUP cib_trusted;
GRANT UPDATE                ON user_roles_user_roleid_seq TO GROUP cib_trusted;
```

```
-- Track the ips the user has visited from, only allow password reset
-- attempts from these IPs
-- NOTE: This table should be stored encrypted with a two way method,
-- This information is pushing the bounds of user privacy, however, it is
-- also needed to protect their account if we wish to offer a password reset
-- ability. I'm not usre how this will interact with dialups and other
-- dynamic ip assingments (quasi-dynamic such as DSL and cable work fine).
CREATE TABLE user_ips (
    relid  SERIAL PRIMARY KEY,
    userid INT REFERENCES users ON UPDATE CASCADE ON DELETE CASCADE,
    client_ip INET);
REVOKE ALL PRIVILEGES       ON                  user_ips FROM PUBLIC;
GRANT SELECT                ON                  user_ips TO GROUP cib_all;
GRANT INSERT,UPDATE,DELETE ON                   user_ips TO GROUP cib_trusted;
GRANT UPDATE                ON user_ips_user_relid_seq TO GROUP cib_trusted;


-- NOTE: This table should be stored encrypted with a two way method
-- This information should be considered highly personal and protected as
-- such. Users will commonly use the same q/a pairs on other sites that
-- use this method. However it does add one more layer to the security.
CREATE TABLE user_questions (
   questionid SERIAL PRIMARY KEY,
   userid     INT    REFERENCES users ON UPDATE CASCADE ON DELETE CASCADE,
   question   TEXT,
   answer     TEXT);
REVOKE ALL PRIVILEGES       ON              user_qucstions FROM PUBLIC;
GRANT SELECT                ON              user_questions TO GROUP cib_all;
GRANT INSERT,UPDATE,DELETE ON               user_questions TO GROUP cib_trusted;
GRANT UPDATE             ON user_questions_questionid_seq TO GROUP cib_trusted;

-- Store user provided public keys, if a user provides a public key we can
-- forgo the tracking of ip/questions nothing in this table is private.
CREATE TABLE user_keys (
   keyid      SERIAL PRIMARY KEY,
   userid     INT    REFERENCES users ON UPDATE CASCADE ON DELETE CASCADE,
   key_type   TEXT,
   public_key TEXT);
REVOKE ALL PRIVILEGES       ON            user_keys FROM PUBLIC;
GRANT SELECT                ON            user_keys TO GROUP cib_all;
GRANT INSERT,UPDATE,DELETE ON             user_keys TO GROUP cib_trusted;
GRANT UPDATE                ON user_keys_keyid_seq TO GROUP cib_trusted;


-- Tables for controlling our Document (Help) system
CREATE TABLE categories (
   categoryid SERIAL PRIMARY KEY,
   cat_order  INT,
   name       TEXT,
   blurb      TEXT);
REVOKE ALL PRIVILEGES       ON                  categories FROM PUBLIC;
GRANT SELECT                ON                  categories TO GROUP cib_all;
GRANT INSERT,UPDATE,DELETE ON                   categories TO cib_admin_user;
GRANT UPDATE                ON categories_categoryid_seq TO cib_admin_user;
INSERT INTO categories (cat_order,name,blurb) VALUES (1,'Help_Navigation_Bar',
       'Documents_in_this_category_are_displayed_in_the_main_CIB_"help_me"_navigation_bar');
INSERT INTO categories (cat_order,name,blurb) VALUES (2,'FAQs',
       'Documents_in_this_category_are_lists_of_Frequently_Asked_Questions');
INSERT INTO categories (cat_order,name,blurb) VALUES (3,'User_Guides',
       'User_guides_and_manuals_for_the_tools');
```

```
INSERT INTO categories (cat_order,name,blurb) VALUES (4,'Administrator_Guides',
       'Documents_to_help_new_administrators_learn_their_way_around');

CREATE TABLE documents (
   documentid SERIAL PRIMARY KEY,
   name       TEXT,
   summary    TEXT,
   url        TEXT);
REVOKE ALL PRIVILEGES       ON                documents FROM PUBLIC;
GRANT SELECT                ON                documents TO GROUP cib_all;
GRANT INSERT,UPDATE,DELETE ON                 documents TO cib_admin_user;
GRANT UPDATE                ON document_documentid_seq TO cib_admin_user;

CREATE TABLE document_categories (
   relid       SERIAL PRIMARY KEY,
   documentid INT    REFERENCES  documents ON UPDATE CASCADE ON DELETE CASCADE,
   categoryid INT    REFERENCES categories ON UPDATE CASCADE ON DELETE CASCADE,
   doc_order  INT);
REVOKE ALL PRIVILEGES       ON           document_categories FROM PUBLIC;
GRANT SELECT                ON           document_categories TO GROUP cib_all;
GRANT INSERT,UPDATE,DELETE ON            document_categories TO cib_admin_user;
GRANT UPDATE                ON document_categories_relid_seq TO cib_admin_user;

-- Tables for lists, used to provide navigation to related sites
-- for our visitors, not fundamental to our site

CREATE TABLE cib_servers (
    serverid SERIAL PRIMARY KEY,
    host     TEXT,
    url      TEXT,
    comments TEXT);
REVOKE ALL PRIVILEGES       ON               cib_servers FROM PUBLIC;
GRANT SELECT                ON               cib_servers TO GROUP cib_all;
GRANT INSERT,UPDATE,DELETE ON                cib_servers TO cib_admin_user;
GRANT UPDATE                ON cib_servers_serverid_seq TO cib_admin_user;
INSERT INTO cib_servers (host,url,comments) VALUES
('MITBDT','https://ballroom.mit.edu/competitions/',
'The_birthplace_of_CIB,_primarily_hosts_competitions_in_the_New_England_area.');

CREATE TABLE outside_comp_type (
    typeid SERIAL PRIMARY KEY,
    name   TEXT);
REVOKE ALL PRIVILEGES       ON           outside_comp_type FROM PUBLIC;
GRANT SELECT                ON           outside_comp_type TO GROUP cib_all;
GRANT INSERT,UPDATE,DELETE ON            outside_comp_type TO cib_admin_user;
GRANT UPDATE                ON outside_comp_type_typeid_seq TO cib_admin_user;
INSERT INTO outside_comp_type (name) values ('List');
INSERT INTO outside_comp_type (name) values ('Wasabi');
INSERT INTO outside_comp_type (name) values ('OpenImpetus');
INSERT INTO outside_comp_type (name) values ('Other');


CREATE TABLE outside_comps (
    outsideid SERIAL PRIMARY KEY,
    type      INT REFERENCES outside_comp_type ON UPDATE CASCADE
                                               ON DELETE RESTRICT,
    name      TEXT,
    url       TEXT);
REVOKE ALL PRIVILEGES       ON               outside_comps FROM PUBLIC;
GRANT SELECT                ON               outside_comps TO GROUP cib_all;
GRANT INSERT,UPDATE,DELETE ON                outside_comps TO cib_admin_user;
```

```
GRANT UPDATE                    ON outside_comps_outsideid_seq TO cib_admin_user;
```

Appendix B

Source Code

B.1 Back-end Scripts

bin/assign_numbers.php

```
#!/usr/local/bin/php
<?php
include("paths.inc");
if ($argv[1]=="") die("No Comp Name provided\n");
if ($argv[2]=="") die("No assignment method selected\n");
$startingNumber = $argv[3];
if ($startingNumber=="") $startingNumber=100;
$unixname=$argv[1];
if (!ereg("^[-a-zA-Z0-9_]*$",$unixname)) die ("Badly formed compname\n");
$method=$argv[2];
$CIB_TOOL_ROLE="register";
include "$CIB_CONFIG_PATH/comps/{$unixname}_config.inc";
if (!isset($SD_compname)) die ("Invalid Comp Name.");

include "$CIB_INCLUDE_PATH/tools/SlidingDoors_header.inc";
ini_alter("include_path",
"$CIB_WEB_PATH/$CIB_WEB_INSTALL/$CIB_TOOL_ROLE/$unixname/PREPARSED");

switch($method)
{
    case "Alphabetical": $query =<<<END_QUERY
SELECT DISTINCT leader, coupleid
FROM  couples JOIN
people ON (leader=peopleid)
ORDER BY lastname,firstname, leader, coupleid;
END_QUERY;
    $result= $db->query($query);
    $numCouples = $result->numrows();
    $couples = array();
    for ($i=0;$i<$numCouples;$i++)
    {
        list($aLeader,$aCouple)= $result->getRowAt($i);
        $couples[$aLeader][]=$aCouple;
```

```
    }
    $numLeaders=count($couples);
    $usedNumbers=$startingNumber;
    foreach ($couples as $leader => $listOfCouples)
    {
        $competitorNumber=$usedNumbers++;

        $numToUpdate = count($listOfCouples);
        for ($j=0; $j<$numToUpdate; $j++)
        {
            $aCouple = $listOfCouples[$j];
            $query=<<<END_QUERY
UPDATE events_registration
SET competitornumber='$competitorNumber'
WHERE coupleid=$aCouple;
END_QUERY;
            $db->query($query);
        }
    }
    case "Random": $query =<<<END_QUERY
SELECT DISTINCT leader, coupleid, firstname, lastname
FROM  couples JOIN
people ON (leader=peopleid)
ORDER BY lastname,firstname, leader, coupleid;
END_QUERY;
    $result= $db->query($query);
    $numCouples = $result->numrows();
    $couples = array();
    $index=array();
    for ($i=0;$i<$numCouples;$i++)
    {
        list($aLeader,$aCouple,$dum,$dum)= $result->getRowAt($i);
        if (!in_array($aLeader,$index))
          $index[]=$aLeader;
        $couples[$aLeader][]=$aCouple;

    }
    $numLeaders=count($couples);
    srand((float)microtime()*1000000);
    for ($i=0;$i<4;$i++)
      shuffle($index);

    $usedNumbers=$startingNumber;
    for ($i=0;$i<$numLeaders;$i++)
    {
      $leader = $index[$i];
      $listOfCouples = $couples[$leader];
      $competitorNumber=$usedNumbers++;
      $numToUpdate = count($listOfCouples);
      for ($j=0; $j<$numToUpdate; $j++)
      {
          $aCouple = $listOfCouples[$j];
          $query=<<<END_QUERY
UPDATE events_registration
SET competitornumber='$competitorNumber'
WHERE coupleid=$aCouple;
END_QUERY;
          $db->query($query);
      }
    }
```

```
    default :break;
}
?>
```

bin/chester_export.php

```
#!/usr/local/bin/php
<?php
include("paths.inc");
if ($argv[1]=="") die("No Comp Name provided\n");
if ($argv[2]=="") die("No Comp File provided\n");
if ($argv[3]=="") die("No email address provided for document delivery\n");
$unixname=$argv[1];
if (!ereg("^[-a-zA-Z0-9_]*$",$unixname)) die ("Badly formed compname\n");
$mailto = $argv[3];


$CIB_TOOL_ROLE="register";
include "$CIB_CONFIG_PATH/comps/{$unixname}_config.inc";
if (!isset($SD_compname)) die ("Invalid Comp Name.");

include "$CIB_INCLUDE_PATH/tools/SlidingDoors_header.inc";
ini_alter("include_path",
"$CIB_WEB_PATH/$CIB_WEB_INSTALL/$CIB_TOOL_ROLE/$unixname/PREPARSED");

$partSeperator=md5(uniqid(time()));

$body.=<<<END_BODY
This is a MIME Encoded Email\n\n--$partSeperator
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

This email was sent via SlidingDoors, the on-line registration
component of CompInaBox.  Attached are raw text files containing the
registration data, suitible for importing directly into the Chester
scrutineering program.

Reply to this message to get in touch with the CompInaBox Server Administrator.

$CIB_SERVER_ADMIN_NAME
$CIB_SERVER_HOST_NAME CompInaBox Server Administrator


END_BODY;

$body.=<<<END_PART_HEADER
--$partSeperator
Content-Type: text/plain; charset=us-ascii; name="by-school.txt"
Content-Transfer-Encoding: 7bit


END_PART_HEADER;


$query =<<<END_QUERY
SELECT COUNT(*) FROM judges_info;
END_QUERY;

$result=$db->query($query);
list($numJudges)=$result->getRowAt(0);
$file = "Adjudicators\t$numJudges\tEvery export will reassign numbers to judges\n";
$query =<<<END_QUERY
SELECT judgeid,lastname,firstname
FROM judges_info JOIN people ON (peopleid=judgeid)
ORDER BY lastname, firstname;
```

```
END_QUERY;
$result=$db->query($query);
for ($i=0;$i < $numJudges;$i++)
{
  list($id,$lastname,$firstname) = $result->getRowAt($i);
  $subQuery ="UPDATE judges_info SET abbrev='$i' WHERE judgeid=$id;";
  $db->query($subQuery);
  $file.="$i\t$firstname $lastname\n";
}

$query =<<<END_QUERY
SELECT eventid, programnumber, class, level,
       style, age, type
FROM events NATURAL JOIN
     events_categories
ORDER BY programnumber;
END_QUERY;
$result = $db->query($query);
$numEvents=$result->numrows();
$file.="\nEvents\t$numEvents\n";

for ($i=0;$i < $numEvents;$i++)
{
  list ($eventID, $eventNumber, $class, $level,
        $style, $age, $type) = $result->getRowAt($i);
  // HACKED
  include ($argv[2]); // very insecure
  $file .="\nEvent $eventNumber $eventCat $eventLevel $eventStyle ";
  $query = "SELECT dancename FROM events_dances WHERE eventid=$eventID ORDER BY danceorder;";
  $danceResult = $db->query($query);
  $numDances = $danceResult->numrows();
  $file .="$numDances ";
  $query = "SELECT competitornumber FROM events_registration WHERE
eventid=$eventID ORDER BY competitornumber;";
  $entrantsResult = $db->query($query);
  $numEntrants = $entrantsResult->numrows();
  $file .= "$numEntrants\tDB EventID:$eventID\n";
  for ($j=0;$j<$numDances;$j++)
    {
      list($dance) = $danceResult->getRowAt($j);
      $file .= "$dance\n";
    }
  for ($j=0;$j<$numEntrants;$j++)
    {
      if ($j!=0 && ($j%10)==0)
        $file .= "\n";
      else if ($j!=0) $file .=" ";
      list($number) = $entrantsResult->getRowAt($j);
      $file .= $number;
    }
  $file .= "\n";
}
$body.="$file\n";

$body.="\n--$partSeperator--\n";
$headers="";
$headers.=<<<END_HEADERS
From: $CIB_SERVER_ADMIN_EMAIL\r
X-mailer: CompInaBox (via PHP)\r
MIME-Version: 1.0\r
Content-Type: multipart/mixed;\r
```

```
        boundary="$partSeperator"
END_HEADERS;
mail("$mailto","[$unixname Competition] Chester Export",$body,$headers);



?>
```

bin/compmngr_export.php

```
#!/usr/local/bin/php
<?php
function getFormattedTime($input)
{
  return date("n.j.Y.H.i.s",$input);
}

include("paths.inc");
if ($argv[1]=="") die("No Comp Name provided\n");
if ($argv[2]=="") die("No Comp File provided\n");
if ($argv[3]=="") die("No email address provided for document delivery\n");
$unixname=$argv[1];
include("$argv[2]"); // INSECURE VERY!!!!!!!!!!!!
if (!ereg("^[-a-zA-Z0-9_]*$",$unixname)) die ("Badly formed compname\n");
$mailto = $argv[3];


$CIB_TOOL_ROLE="register";
include "$CIB_CONFIG_PATH/comps/{$unixname}_config.inc";
if (!isset($SD_compname)) die ("Invalid Comp Name.");

include "$CIB_INCLUDE_PATH/tools/SlidingDoors_header.inc";
ini_alter("include_path",
"$CIB_WEB_PATH/$CIB_WEB_INSTALL/$CIB_TOOL_ROLE/$unixname/PREPARSED");


$query =<<<END_QUERY
SELECT DISTINCT eventid,
                l.firstname, l.lastname, lo.name,
                f.firstname, f.lastname, fo.name
FROM events_registration NATURAL JOIN
     couples JOIN
     people AS l ON (leader=l.peopleid) LEFT JOIN
     organizations AS lo ON (l.organization=lo.orgid) JOIN
     people AS f ON (follower=f.peopleid) LEFT JOIN
     organizations AS fo ON (f.organization=fo.orgid)
ORDER BY eventid, l.lastname, l.firstname, f.lastname, f.firstname;
END_QUERY;

$result=$db->query($query);
$numrows=$result->numrows();
$partSeperator=md5(uniqid(time()));

$body.=<<<END_BODY
This is a MIME Encoded Email\n\n--$partSeperator
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

This email was sent via SlidingDoors, the on-line registration
component of CompInaBox.  Attached are raw text files containing the
registration data, suitible for importing directly into COMPMNGR.  Pipes
('|') are used to delimit the data.

Reply to this message to get in touch with the CompInaBox Server Administrator.

$CIB_SERVER_ADMIN_NAME
$CIB_SERVER_HOST_NAME CompInaBox Server Administrator
```

```
END_BODY;

$body.=<<<END_PART_HEADER
--$partSeperator
Content-Type: text/plain; charset=us-ascii; name="by-school.txt"
Content-Transfer-Encoding: 7bit


END_PART_HEADER;
$file1="";
$time=time();
for ($i=0;$i<$numrows;$i++)
{
    list ($eventID, $lfname, $llname, $lorg, $ffname, $flname, $forg) =
        $result->getRowAt($i);
    $danceCategory = $danceCats[$eventID];
    $danceCode     = $danceCodes[$eventID];
    $time1=getFormattedTime($time++);
    $time2=getFormattedTime($time++);
    $time3=getFormattedTime($time++);
    $time4=getFormattedTime($time++);
    if ($lorg==$forg) $studio=$lorg; else $studio="$lorg/$forg";
    $file1.="EN|$danceCategory|$danceCode|0|0|A||$userCode|$time1|";
    $file1.="ST|$studio||||||||||1|$userCode|$time2|"; // studio
    $file1.="PR||A|$lfname|$llname|||1|$userCode|$time3|"; // leader
    $file1.="PR||A|$ffname|$flname|||1|$userCode|$time4|"; // follower
    $file1.="PR||||||||||"; // coach
    $file1.="\n";
}
$body.="$file1\n";

$body.="\n--$partSeperator--\n";
$headers="";
$headers.=<<<END_HEADERS
From: $CIB_SERVER_ADMIN_EMAIL\r
X-mailer: CompInaBox (via PHP)\r
MIME-Version: 1.0\r
Content-Type: multipart/mixed;\r
        boundary="$partSeperator"
END_HEADERS;
mail("$mailto","[$unixname Competition] COMPMNGR Export",$body,$headers);



?>
```

bin/head-table-lists.php

```
<?php
include "header.inc";

$LUT = $db->buildLUT();

set_time_limit(900);
## Generate MC List by Events

$query = "SELECT eventid, programnumber, longname from events natural
join events_names order by programnumber;";
$mcText="\\documentclass{article}\n";
$mcText.="\\usepackage{fullpage,graphics,array,color,colortbl,nopageno}\n";
$mcText.="\\definecolor{light}{gray}{.8}\n";
$mcText.="\\begin{document}\n";
$mcText.="\\Huge MC Lists by Event \\normalsize\n";
$scrutText="\\documentclass{article}\n";
$scrutText.="\\usepackage{fullpage,graphics,array,color,colortbl,nopageno}\n";
$scrutText.="\\definecolor{light}{gray}{.8}\n";
$scrutText.="\\begin{document}\n";
$scrutText.="\\Huge Scrut Update Forms \\normalsize\n";
$outerResult = $db->query($query);
$numevents =$outerResult->numrows();
for ($i=0;$i<$numevents;$i++)
{
    list($id,$num,$name)=$outerResult->getRowAt($i);




    $query = "SELECT competitornumber, l.firstname, l.lastname,
l.organization, f.firstname, f.lastname, f.organization from couples natural
join events_registration join people as l on (l.peopleid = leader) join people
as f on (f.peopleid=follower) where eventid=$id and l.firstname<>'TBA' order
by competitornumber;";
    $innerResult=$db->query($query);
    $numEntries = $innerResult->numrows();
    if ($numEntries==0) continue;

    $scrutText .= "\\newpage \n\\center{\\Large Event $num -- $name}\\\\ \n";
    $scrutText .= "\\begin{tabular}{c|c}\n";
    $scrutText .= "\\hspace{20ex}\\textbf{Adds} \\hspace{20ex} & ";
    $scrutText .= "\\hspace{20ex}\\textbf{Drops} \\hspace{20ex} \\\\\n";
    for ($silly=0;$silly<48;$silly++)
        $scrutText.= " & \\\\ \n";
    $scrutText .= "\\end{tabular}";


    $tba=0;
    for ($j=0;$j<$numEntries;$j++)
    {
        list($couple,$leaderfirst,$leaderlast,$leaderorg,$followfirst,
                $followlast,$followorg) = $innerResult->getRowAt($j);
        $leaderAffil = $LUT[$leaderorg];
        $followAffil = $LUT[$followorg];

        if ($j % 48 ==0)
        {
```

```
            if ($j!=0)
                $mcText.="\\end{tabular}\n";
            $mcText.="\\newpage\n";
            if ($j==0)
$mcText.="\\center{\\Large Event $num -- $name ($numEntries Entries)}\\\\\ \n";
            else
$mcText.="\\center{\\Large Event $num -- $name (cont.)}\\\\\ \n";
            $mcText.="\\begin{tabular}{lllr}\\hline\n";

        }
        if ($couple=="")
        {
            continue;
        }
        $line ="\\underline{\\hspace{0.4in}} $couple & $leaderfirst
$leaderlast ($leaderAffil) & \\& $followfirst $followlast
($followAffil) &".($j+1)."\\\\\ \n";
        if ($j % 2 ==0)
            $mcText.=$line;
        else
            $mcText .= "\\rowcolor{light}$line";
        if ($j % 6 ==5)
            $mcText .= "\\hline \n";

    }
    $startnumber=$j;
    $endnumber=(int)($j*1.10)+2;
    for ($k=$startnumber;$k<$endnumber;$k++)
    {
        if ($k % 48 ==0)
        {
            if ($k!=0)
                $mcText.="\\end{tabular}\n";
            $mcText.="\\newpage\n";
            if ($k==0)
$mcText.="\\center{\\Large Event $num -- $name ($numEntries Entries)} \n";
            else
$mcText.="\\center{\\Large Event $num -- $name (cont.)}\n";
            $mcText.="\\begin{tabular}{lllr}\\hline\n";

        }
        $line = "\\underline{\\hspace{0.4in}} &\hspace{40ex}
&\hspace{40ex} & ".($k+1)."\\\\\ \n";

        if ($k % 2 ==0)
            $mcText.=$line;
        else
            $mcText .= "\\rowcolor{light}$line";
        if ($k % 6 ==5)
            $mcText .= "\\hline \n";

    }

    $mcText .= "\\end{tabular}\n";
}
$scrutText .= "\\end{document}\n";
$mcText .= "\\end{document}\n";
#mail("nielsene@mit.edu","Test","Hi");
#mail("nielsene@mit.edu","MCTest",$mcText);
#mail("nielsene@mit.edu","ScrutList",$scrutText);
#echo $text;
```

```
#die("done");
#    echo str_replace("\n","<br>",htmlspecialchars($mcText));

## Generate Master MC List


$query = "SELECT distinct competitornumber, l.firstname, l.lastname, ";
$query .= "l.organization, f.firstname, f.lastname, f.organization ";
$query .= "FROM events_registration NATURAL JOIN couples JOIN ";
$query .= "people AS l ON (leader=l.peopleid) JOIN ";
$query .= "people AS f ON (follower=f.peopleid)  ";
$query .= "WHERE l.firstname<>'TBA' ";
$query .= "ORDER BY competitornumber,l.lastname, l.firstname, ";
$query .= "f.lastname, f.firstname;";
#echo $query;
$result = $db->query($query);
$numRows=$result->numrows();
$mcText="\\documentclass{article}\n";
$mcText.="\\usepackage{fullpage,graphics,array,color,colortbl,nopageno}\n";
$mcText.="\\definecolor{light}{gray}{.8}\n";
$mcText.="\\begin{document}\n";
$mcText.="\\Huge Master MC Lists \\normalsize\n";
for ($j=0;$j<$numRows;$j++)
{
    list($couple,$leaderfirst,$leaderlast,$leaderorg,$followfirst,
            $followlast,$followorg) = $result->getRowAt($j);
        $leaderAffil = $LUT[$leaderorg];
        $followAffil = $LUT[$followorg];

        if ($j % 48 ==0)
        {
            if ($j!=0)
                $mcText.="\\end{tabular}\n";
            $mcText.="\\newpage\n";
            if ($j==0)
                $mcText.="\\center{\\Large Master List of Competitors}\\\\ \n";
            else
                $mcText.="\\center{\\Large Master List of Competitors (cont.)}\\\\ \n";
            $mcText.="\\begin{tabular}{lllr}\\hline\n";

        }
        if ($couple=="")
        {
            continue;
        }
        $line ="$couple & $leaderfirst $leaderlast ($leaderAffil) & \\&
$followfirst $followlast ($followAffil) &".($j+1)."\\\\ \n";
        if ($j % 2 ==0)
            $mcText.=$line;
        else
            $mcText .= "\\rowcolor{light}$line";
        if ($j % 6 ==5)
            $mcText .= "\\hline \n";

}
$mcText .= "\\end{tabular}\n";
$mcText .= "\\end{document}\n";
#mail("nielsene@mit.edu","Test","Hi");
#mail("nielsene@mit.edu","MCTestMaster",$mcText);
#echo $text;
#die("done");
```

```
#    echo str_replace("\n","<br>",htmlspecialchars($mcText));

$query = "SELECT distinct competitornumber, l.firstname, l.lastname, ";
$query .= "l.organization, f.firstname, f.lastname, f.organization ";
$query .= "FROM events_registration NATURAL JOIN couples JOIN ";
$query .= "people AS l ON (leader=l.peopleid) JOIN ";
$query .= "people AS f ON (follower=f.peopleid)  ";
$query .= "WHERE l.firstname<>'TBA' ";
$query .= "ORDER BY l.lastname, l.firstname, ";
$query .= "f.lastname, f.firstname;";
#echo $query;
$result = $db->query($query);
$numRows=$result->numrows();
$mcText="\\documentclass{article}\n";
$mcText.="\\usepackage{fullpage,graphics,array,color,colortbl,nopageno}\n";
$mcText.="\\definecolor{light}{gray}{.8}\n";
$mcText.="\\begin{document}\n";
$mcText.="\\Huge Master MC Lists, by Name \\normalsize\n";
for ($j=0;$j<$numRows;$j++)
{
    list($couple,$leaderfirst,$leaderlast,$leaderorg,$followfirst,
            $followlast,$followorg) = $result->getRowAt($j);
        $leaderAffil = $LUT[$leaderorg];
        $followAffil = $LUT[$followorg];

        if ($j % 48 ==0)
        {
            if ($j!=0)
                $mcText.="\\end{tabular}\n";
            $mcText.="\\newpage\n";
            if ($j==0)
$mcText.="\\center{\\Large Master List of Competitors}\\\\ \n";
            else
$mcText.="\\center{\\Large Master List of Competitors (cont.)}\\\\ \n";
            $mcText.="\\begin{tabular}{lllr}\\hline\n";

        }
        if ($couple=="")
        {
            continue;
        }
        $line ="$couple & $leaderfirst $leaderlast ($leaderAffil) & \\&
$followfirst $followlast ($followAffil) &".($j+1)."\\\\ \n";
        if ($j % 2 ==0)
            $mcText.=$line;
        else
            $mcText .= "\\rowcolor{light}$line";
        if ($j % 6 ==5)
            $mcText .= "\\hline \n";

}
$mcText .= "\\end{tabular}\n";
$mcText .= "\\end{document}\n";
#mail("nielsene@mit.edu","Test","Hi");
mail("nielsene@mit.edu","MCTestMaster",$mcText);
#echo $text;
#die("done");
#    echo str_replace("\n","<br>",htmlspecialchars($mcText));
```

```
$query = "SELECT categoryid , level , style from events_categories where level
is not NULL and style is not NULL and level <> 'Newcomer';";
$scrutText="\\documentclass{article}\n";
$scrutText.="\\usepackage{fullpage ,graphics ,array ,color ,colortbl ,nopageno}\n";
$scrutText.="\\definecolor{light}{gray}{.8}\n";
$scrutText.="\\setlength{\\topmargin}{0in}\n";
$scrutText.="\\setlength{\\evensidemargin}{0in}\n";
$scrutText.="\\setlength{\\oddsidemargin}{0in}\n";
$scrutText.="\\setlength{\\textwidth}{9in}\n";
$scrutText.="\\setlength{\\textheight}{6.5in}\n";

$scrutText.="\\begin{document}\n";
$scrutText.="\\Huge Scrut 4/5 Dance Forms \\normalsize\n";
$outerResult = $db->query($query);
$numevents =$outerResult->numrows();
for ($i=0;$i<$numevents;$i++)
{
    list($catid ,$level ,$style)=$outerResult->getRowAt($i);

    $query="SELECT count(eventid) from events where categoryid=$catid;";
    $countResult=$db->query($query);
    list($numDances)=$countResult->getRowAt(0);



    $query = "SELECT competitornumber, count(*) as num, l.firstname,
l.lastname, l.organization, f.firstname, f.lastname, f.organization from
couples natural join events_registration join people as l on (l.peopleid =
leader) join people as f on (f.peopleid=follower) natural join events where
categoryid=$catid and l.firstname<>'TBA' group by competitornumber,
l.firstname, l.lastname, l.organization, f.firstname, f.lastname,
f.organization order by num desc,competitornumber asc;";
    $innerResult=$db->query($query);
    $numEntries = $innerResult->numrows();
    if ($numEntries==0) continue;

    $tba=0;
    for ($j=0;$j<$numEntries;$j++)
    {
        list($couple ,$count ,$leaderfirst ,$leaderlast ,$leaderorg ,$followfirst ,
                $followlast ,$followorg) = $innerResult->getRowAt($j);
        $leaderAffil = $LUT[$leaderorg];
        $followAffil = $LUT[$followorg];
        if ($count!=$numDances) break;
        if ($j % 36 ==0)
        {
            if ($j!=0)
                $scrutText.="\\end{tabular}\n";
            $scrutText.="\\newpage\n";
            if ($j==0)
$scrutText.="\\center{\\Large $level $style $numDances-Dancers}\\\\ \n";
            else
$scrutText.="\\center{\\Large $level $style (cont.)}\\\\ \n";
            $scrutText.="\\begin{tabular}{lll";
            for ($silly=0;$silly<$numDances;$silly++)
                $scrutText .= "c";
            $scrutText.="r}\\hline\n";

        }
        $line ="\\underline{\\hspace{0.4in}} $couple & $leaderfirst
```

```
$leaderlast ($leaderAffil) & \\& $followfirst $followlast ($followAffil) &";
            for ($silly=0;$silly<$numDances;$silly++)
                $line .= "\\underline{\\hspace{4ex}} & ";
            $line .= ($j+1)."\\\\ \n";
        if ($j % 2 ==0)
            $scrutText.=$line;
        else
            $scrutText .= "\\rowcolor{light}$line";
        if ($j % 6 ==5)
            $scrutText .= "\\hline \n";

    }
    $startnumber=$j;
    $endnumber=(int)($j*1.10)+2;
    for ($k=$startnumber;$k<$endnumber;$k++)
    {
        if ($k % 36 ==0)
        {
            if ($k!=0)
                $scrutText.="\\end{tabular}\n";
            $scrutText.="\\newpage\n";
            if ($k==0)
$scrutText.="\\center{\\Large $level $style $numDances-Dancers}\\\\ \n";
            else
                $scrutText.="\\center{\\Large $level $style (cont.)}\\\\ \n";
            $scrutText.="\\begin{tabular}{lll";
            for ($silly=0;$silly<$numDances;$silly++)
                $scrutText .= "c";
            $scrutText.="r}\\hline\n";

        }
        $line = "\\underline{\\hspace{0.4in}} &\hspace{40ex} &\hspace{40ex} & ";
            for ($silly=0;$silly<$numDances;$silly++)
                $line .= "\\underline{\\hspace{4ex}} & ";
            $line .= ($k+1)."\\\\ \n";

        if ($k % 2 ==0)
            $scrutText.=$line;
        else
            $scrutText .= "\\rowcolor{light}$line";
        if ($k % 6 ==5)
            $scrutText .= "\\hline \n";

    }

    $scrutText .= "\\end{tabular}\n";
}
    list($level,$style)=array("Newcomer","");

    $query="SELECT count(eventid) from events where categoryid in (1,2,3,4);";
    $countResult=$db->query($query);
    list($numDances)=$countResult->getRowAt(0);



    $query = "SELECT competitornumber, count(*) as num, l.firstname,
l.lastname, l.organization, f.firstname, f.lastname, f.organization from
couples natural join events_registration join people as l on (l.peopleid =
leader) join people as f on (f.peopleid=follower) natural join events where
categoryid in (1,2,3,4)  and l.firstname<>'TBA' group by competitornumber,
```

```
l.firstname, l.lastname, l.organization, f.firstname, f.lastname,
f.organization order by num desc,competitornumber asc;";
    $innerResult=$db->query($query);
    $numEntries = $innerResult->numrows();
    if ($numEntries==0) continue;

    $tba=0;
    for ($j=0;$j<$numEntries;$j++)
    {
        list($couple,$count,$leaderfirst,$leaderlast,$leaderorg,
        $followfirst,$followlast,$followorg) = $innerResult->getRowAt($j);
        $leaderAffil = $LUT[$leaderorg];
        $followAffil = $LUT[$followorg];
        if ($count!=$numDances) break;
        if ($j % 36 ==0)
        {
            if ($j!=0)
                $scrutText.="\\end{tabular}\n";
            $scrutText.="\\newpage\n";
            if ($j==0)
                $scrutText.="\\center{\\Large $level $style
$numDances-Dancers}\\\\\ \n";
            else
                $scrutText.="\\center{\\Large $level $style (cont.)}\\\\\ \n";
            $scrutText.="\\begin{tabular}{lll";
            for ($silly=0;$silly<$numDances;$silly++)
                $scrutText .= "c";
            $scrutText.="r}\\hline\n";

        }
        $line ="\\underline{\\hspace{0.4in}} $couple & $leaderfirst
$leaderlast ($leaderAffil) & \\& $followfirst $followlast ($followAffil) &";
            for ($silly=0;$silly<$numDances;$silly++)
                $line .= "\\underline{\\hspace{4ex}} & ";
            $line .= ($j+1)."\\\\\ \n";
        if ($j % 2 ==0)
            $scrutText.=$line;
        else
            $scrutText .= "\\rowcolor{light}$line";
        if ($j % 6 ==5)
            $scrutText .= "\\hline \n";

    }
    $startnumber=$j;
    $endnumber=(int)($j*1.10)+2;
    for ($k=$startnumber;$k<$endnumber;$k++)
    {
        if ($k % 36 ==0)
        {
            if ($k!=0)
                $scrutText.="\\end{tabular}\n";
            $scrutText.="\\newpage\n";
            if ($k==0)
                $scrutText.="\\center{\\Large $level $style
$numDances-Dancers}\\\\\ \n";
            else
                $scrutText.="\\center{\\Large $level $style (cont.)}\\\\\ \n";
            $scrutText.="\\begin{tabular}{lll";
            for ($silly=0;$silly<$numDances;$silly++)
                $scrutText .= "c";
            $scrutText.="r}\\hline\n";
```

```
        }
        $line = "\\underline{\\hspace{0.4in}} &\hspace{40ex}
&\hspace{40ex} & ";
            for ($silly=0;$silly<$numDances;$silly++)
                $line .= "\\underline{\\hspace{4ex}} & ";
            $line .= ($k+1)."\\\\ \n";

        if ($k % 2 ==0)
            $scrutText.=$line;
        else
            $scrutText .= "\\rowcolor{light}$line";
        if ($k % 6 ==5)
            $scrutText .= "\\hline \n";

    }

    $scrutText .= "\\end{tabular}\n";
$scrutText .= "\\end{document}\n";
#mail("nielsene@mit.edu","ScrutMultiDances",$scrutText);

?>
```

bin/initial-lists.php

```
#!/usr/local/bin/php
<?php
function latexHeader($title)
{
$header=<<<END_HEADER
\\documentclass{report}
\\usepackage{fullpage,graphics,array,color,colortbl,nopageno,longtable}
\\definecolor{light}{gray}{.8}
\\renewcommand{\\thechapter}{}
\\renewcommand{\\thesection}{\\Roman{section}}
\\renewcommand{\\thesubsection}{}
\\renewcommand{\\thesubsubsection}{Event \\addtocounter{event}{1}\\theevent :}
\\newcommand{\\awards}{\\vspace{2ex}1st place:\\underline{\\hspace{0.4in}}
2nd place:\\underline{\\hspace{0.4in}} 3rd place:\\underline{\\hspace{0.4in}}
4th place:\\underline{\\hspace{0.4in}} 5th place:\\underline{\\hspace{0.4in}}
6th place:\\underline{\\hspace{0.4in}}\\par}
\\newcommand{\\awardsshort}{\\centerline{1st place:\\underline{\\hspace{0.4in}}
2nd place:\\underline{\\hspace{0.4in}}
3rd place:\\underline{\\hspace{0.4in}}}\\vspace{2ex}}
\\begin{document}
\\Huge $title \\normalsize
\\newpage
END_HEADER;
 return $header;
}



include("paths.inc");
if ($argv[1]=="") die("No Comp Name provided\n");
if ($argv[2]=="") die("No email address provided for document delivery\n");
$unixname=$argv[1];
if (!ereg("^[-a-zA-Z0-9_]*$",$unixname)) die ("Badly formed compname\n");
$mailto = $argv[2];

$CIB_TOOL_ROLE="register";
include "$CIB_CONFIG_PATH/comps/{$unixname}_config.inc";
if (!isset($SD_compname)) die ("Invalid Comp Name.");

include "$CIB_INCLUDE_PATH/tools/SlidingDoors_header.inc";
ini_alter("include_path",
"$CIB_WEB_PATH/$CIB_WEB_INSTALL/$CIB_TOOL_ROLE/$unixna
me/PREPARSED");


/*************************************************************/
/* Event Name, number of entrants, and competitor numbers */
/* plain text, comma/tab seperated, and xml                  */
/* PDF/LaTeX version generated latter, (minus TBAs)          */
/*************************************************************/

$query =<<<END_QUERY
SELECT eventid, programnumber, longname
FROM events NATURAL JOIN events_names
ORDER BY programnumber;
END_QUERY;

$textEventCouples="";
$csvEventCouples="";
```

```
$tsvEventCouples="";
$xmlEventCouples="<event_registration>\n";
$outerResult = $db->query($query);
$numevents =$outerResult->numrows();
for ($i=0;$i < $numevents;$i++)
{
    list($id,$num,$name)=$outerResult->getRowAt($i);
    $query = <<<END_QUERY
SELECT competitornumber
FROM couples NATURAL JOIN events_registration
WHERE eventid=$id
ORDER BY competitornumber;
END_QUERY;
    $innerResult=$db->query($query);
    $numEntries = $innerResult->numrows();
    if ($numEntries==0) continue;
    $textEventCouples.="Event $num -- $name ($numEntries Entries)";
    $csvEventCouples=$tsvEventCouples=$textEventCouples;
    $xmlEventCouples="<event name=\"$name\" number=\"$num\">\n";
    $xmlEventCouples="\t<number_registered>$numEntries</number_registered>\n";
    $tba=0;
    for ($j=0;$j < $numEntries;$j++)
    {
        list($couple) = $innerResult->getRowAt($j);
        if ($j % 15 ==0)
          {
            $textEventCouples .= "\n";
            $tsvEventCouples .= "\n";
            $csvEventCouples .= "\n";
          }
        else
          {
            $textEventCouples .= "    ";
            $tsvEventCouples .= "\t";
            $csvEventCouples .= ",";
          }
        if ($couple=="")
            $tba++;
        else
          {
            $textEventCouples .= "$couple";
            $csvEventCouples  .= "$couple";
            $tsvEventCouples  .= "$couple";
            $xmlEventCouples  .= "\t<entrant>$couple</entrant>\n";
          }
    }
    if ($tba)
      {
        $textEventCouples .= "\n($tba TBA's)";
        $csvEventCouples .= "\n($tba TBA's)";
        $tsvEventCouples  .= "\n($tba TBA's)";
        $xmlEventCouples .= "\t<tba_count>$tba</tba_count>\n";
      }
    $textEventCouples .= "\n\n";
    $textEventCouples .= "\n\n\awards\n\n";
    $csvEventCouples .= "\n\n";
    $tsvEventCouples .= "\n\n";
    $xmlEventCouples .= "</event>\n";

}
$xmlEventCouples .= "</event_registration>\n";
```

```
/* Retrieve data for fancier outputs */

$query =<<<END_QUERY
SELECT programnumber, longname, l.firstname, l.lastname,
       lo.abbreviation, f.firstname, f.lastname, fo.abbreviation,
       competitornumber
FROM events_registration NATURAL JOIN events NATURAL JOIN
     events_names NATURAL JOIN couples JOIN
     people AS l ON (leader=l.peopleid) JOIN
     people AS f ON (follower=f.peopleid) LEFT OUTER JOIN
     organizations AS la ON (l.organization=la.orgid) LEFT OUTER JOIN
     abbreviations AS lo ON (la.name=lo.fulltext) LEFT OUTER JOIN
     organizations AS fa ON (f.organization=fa.orgid) LEFT OUTER JOIN
     abbreviations AS fo ON (fa.name=fo.fulltext)
WHERE l.firstname<>'TBA'
ORDER_BY_programnumber,_l.lastname,_l.firstname,_f.lastname,_f.firstname;
END_QUERY;

$programEventArray=array();
$masterListByNumber=array();

$result_=_$db->query($query);
$numRows=$result->numrows();
for($i=0;$i_<_$numRows;$i++)
{
____list_($progNum,_$eventName,
__________$lFirst,_$lLast,_$lAbbrev,
__________$fFirst,_$fLast,_$fAbbrev,
__________$compNum)_=_$result->getRowAt($i);
____$lAbbrev_=_str_replace('&','\&',$lAbbrev);
____$fAbbrev_=_str_replace('&','\&',$fAbbrev);
____if_(isset($programEventArray[$progNum]))
____{
________array_push($programEventArray[$progNum]["Entries"],
__________________array($compNum,_"$lFirst_$lLast_($lAbbrev)",
____________________"$fFirst_$fLast_($fAbbrev)"));
____}
____else
________$programEventArray[$progNum]=array("Name"=>$eventName,
__________________________________"Entries"=>array(array($compNum,
____________________"$lFirst_$lLast_($lAbbrev)",
____________________"$fFirst_$fLast_($fAbbrev)")));
____if_(isset($masterListByNumber[$compNum]))
____{
________if_(!in_array("$fFirst_$fLast_($fAbbrev)",
________________$masterListByNumber[$compNum]["Followers"]))
____________array_push($masterListByNumber[$compNum]["Followers"],
________________"$fFirst_$fLast_($fAbbrev)");
____}
____else
________$masterListByNumber[$compNum]=array(
________________"Name"=>"$lFirst_$lLast_($lAbbrev)",
____________"Followers"=>array("$fFirst_$fLast_($fAbbrev)"));
}


###_List_of_Competitors_by_School
ksort($masterListByNumber);

/*********************************************************/
```

```
/*_Master_List_of_Couples_by_Number_____________________________*/
/* PDF/Latex only right now                                      */
/****************************************************************/
$coupleLaTeX = latexHeader("Master Competitor List");
$coupleLaTeX .= "\\begin{center}\n";
$coupleLaTeX .= "\\begin{longtable}{lll}\n";
$coupleText = "";
reset($masterListByNumber);
while($aCompetitor = each($masterListByNumber))
{
    $number = $aCompetitor["key"];
    $name = $aCompetitor["value"]["Name"];
    $follower = $aCompetitor["value"]["Followers"][0];
    $coupleLaTeX .= "$number & $name & \\& $follower \\\\\n";
    $coupleText .= "$number\t$name\t& $follower\n";
    $numFollowers = count($aCompetitor["value"]["Followers"]);
    for ($i=1;$i<$numFollowers;$i++)
      {
        $coupleLaTeX .= "\t&\t& \\& ".$aCompetitor["value"]["Followers"][$i] .
                "\\\\\n";
        $coupleText .= "\t\t& $follower\n";
      }
}
$coupleLaTeX .= "\\end{longtable}\n";
$coupleLaTeX .= "\\end{center}\n";
$coupleLaTeX .= "\\end{document}\n";

/****************************************************************/
/* Event by event list of names                                 */
/* PDF/Latex only right now                                     */
/****************************************************************/
$LaTeXEventCouples = latexHeader("Event Registration - Short Form");
$LaTeXEventCouples .= "\\begin{center}\n";
$LaTeXEventCouples .= "\\begin{longtable}{lll}\n";
reset($programEventArray);
while ($anEvent = each($programEventArray))
{
    $progName = $anEvent["value"]["Name"];
    $entries = $anEvent["value"]["Entries"];
        sort($entries);
    $numEntries = count($entries);
    $LaTeXEventCouples .= '\subsubsection{'.$progName_."($numEntries_entries)".'}';
____for_($i=0;$i<$numEntries;$i++)
____{
________if_($i_%_12_==_0)
____________$LaTeXEventCouples_.=_"\n";
________$LaTeXEventCouples_.=_"__".$entries[$i][0];
____}
____$LaTeXEventCouples_.=_"\n\n\\awards\n\n";
}
$LaTeXEventCouples_.=_"\\end{document}\n";


/****************************************************************/
/*_Couples_by_School,_by_name_____________________________*/
/*_PDF/Latex_only_right_now_____________________________*/
/****************************************************************/
$LaTeXSchoolCouples_=_latexHeader("Couple_Listing_by_School,_by_name");
$LaTeXSchoolCouples_.=_"\\begin{center}\n";
$LaTeXSchoolCouples_.=_"\\begin{longtable}{lll}\n";
_$textSchoolCouples_="";
```

```
$query =<<<END_QUERY
SELECT DISTINCT la.name, l.firstname, l.lastname,
                fa.name, f.firstname, f.lastname,
                competitornumber
FROM events_registration NATURAL JOIN couples JOIN
     people AS l ON (leader=l.peopleid) JOIN
     people AS f ON (follower=f.peopleid) LEFT OUTER JOIN
     organizations AS la ON (l.organization=la.orgid) LEFT OUTER JOIN
     organizations AS fa ON (f.organization=fa.orgid)
ORDER BY la.name, fa.name, l.lastname, l.firstname, f.lastname, f.firstname;
END_QUERY;

$result = $db->query($query);
$numRows= $result->numrows();
$schoolArray=array();
for ($i=0;$i < $numRows;$i++)
{
        list($leadAffil, $lFirst, $lLast,
             $followAffil, $fFirst, $fLast,
             $num) = $result->getRowAt($i);
        $leadAffil=str_replace('&','\&',$leadAffil);
        $followAffil=str_replace('&','\&',$followAffil);
        if(isset($schoolArray[$leadAffil][$followAffil]))
        {
            if(isset($schoolArray[$leadAffil][$followAffil][$num]))
                array_push($schoolArray[$leadAffil][$followAffil][$num],
                           array("Leader"=>"$lFirst_$lLast",
                                 "Follower"=>"$fFirst_$fLast"));
            else
                $schoolArray[$leadAffil][$followAffil][$num]=
                    array(array("Leader"=>"$lFirst_$lLast",
                                "Follower"=>"$fFirst_$fLast"));
        }
        else
            $schoolArray[$leadAffil][$followAffil][$num]=
                array(array("Leader"=>"$lFirst_$lLast",
                            "Follower"=>"$fFirst_$fLast"));
}
$LaTeXSchoolCouples_=_latexHeader("Competitor_Listing_--_By_School");
$LaTeXSchoolCouples_.=_"\\chapter*{Competitor_Listing_--_By_School}\n";
$LaTeXSchoolCouples_.=_"\\begin{longtable}{lll}\n";
$textSchoolCouples_.=_"Competitor_Listing_--_By_School\n";
reset($schoolArray);
while($aSchool_=_each_($schoolArray))
{
__$schoolName_=_$aSchool['key'];
__$otherSchools_=_$aSchool['value'];
__reset($otherSchools);
__while($bSchool_=_each($otherSchools))
____{
______$bSchoolName_=_$bSchool['key'];
______$leaders_=_$bSchool['value'];
______if_($bSchoolName==$schoolName)
________{
__________$LaTeXSchoolCouples_.=_'\\multicolumn{3}{l}{{'."\\large_\\textbf".
__________'{'.$schoolName.'}}}'."\\\\\n";
__________$textSchoolCouples_.=_"$schoolName\n";
________}
______else
________{
```

```
          $LaTeXSchoolCouples .= '\\multicolumn{3}{1}{{ '."\\large\\textbf".
                    '{'.$schoolName."/".$bSchoolName.'}}}'."\\\\\n";
          $textSchoolCouples .= "$schoolName/$bSchoolName\n";
        }
      reset($leaders);
      while($aLeader=each($leaders))
        {
          $num=$aLeader['key'];
          $partners=count($aLeader['value']);
          for($i=0;$i<$partners;$i++)
            {
              $lName=$aLeader['value'][$i]['Leader'];
              $fName=$aLeader['value'][$i]['Follower'];
              if($i==0)
                {
                  $LaTeXSchoolCouples .= "$num\t&\t$lName\t& \\&\t$fName\t\\\\\n";
                  $textSchoolCouples .= "$num\t$lName\t& $fName\n";
                }
              else
                {
                  $LaTeXSchoolCouples .= "& \t & \\&\t$fName\t\\\\\n";
                  $textSchoolCouples .= "\t\t& $fName\n";
                }
            }
        }
      $LaTeXSchoolCouples .= "\\multicolumn{3}{1}{}\\\\\n";
    }
}
$LaTeXSchoolCouples .= "\\end{longtable}\n\\end{document}\n";

/*************************************************************/
/* Event Listing with Names, Add/Drop Summaries              */
/* PDF/Latex Only                                            */
/*************************************************************/

$query =<<<END_QUERY
SELECT eventid, programnumber, longname
FROM events NATURAL JOIN events_names
ORDER BY programnumber;
END_QUERY;
$LaTeXFullEvents = latexHeader("Event by Event Listing with Full Names");
$LaTeXAddDropSummary = latexHeader("Add/Drop Summary");
$outerResult = $db->query($query);
$numevents = $outerResult->numrows();
for($i=0;$i<$numevents;$i++)
{
  list($id,$num,$name)=$outerResult->getRowAt($i);
  $query =<<<END_QUERY
SELECT competitornumber, l.firstname, l.lastname, lo.abbreviation,
       f.firstname, f.lastname, fo.abbreviation
FROM couples NATURAL JOIN events_registration JOIN
     people AS l ON (l.peopleid = leader) JOIN
     people AS f ON (f.peopleid=follower) LEFT OUTER JOIN
     organizations AS la ON (l.organization=la.orgid) LEFT OUTER JOIN
     abbreviations AS lo ON (la.name=lo.fulltext) LEFT OUTER JOIN
     organizations AS fa ON (f.organization=fa.orgid) LEFT OUTER JOIN
     abbreviations AS fo ON (fa.name=fo.fulltext)
WHERE eventid=$id AND l.firstname<>'TBA'
ORDER BY competitornumber;
END_QUERY;
  $innerResult=$db->query($query);
```

```
    $numEntries = $innerResult->numrows();
    if ($numEntries==0) continue;

    $LaTeXAddDropSummary .= "\\newpage \n\\center{\\Large Event $num -- $name}\\\\\ \n";
    $LaTeXAddDropSummary .= "\\begin{tabular}{c|c}\n";
    $LaTeXAddDropSummary .= "\\hspace{20ex}\\textbf{Adds} \\hspace{20ex} & ";
    $LaTeXAddDropSummary .= "\\hspace{20ex}\\textbf{Drops} \\hspace{20ex} \\\\\n";
    for ($silly=0;$silly<48;$silly++)
      $LaTeXAddDropSummary.= " & \\\\ \n";
    $LaTeXAddDropSummary .= "\\end{tabular}";


    $tba=0;
    for ($j=0;$j<$numEntries;$j++)
      {
        list($couple,$leaderfirst,$leaderlast,
             $leaderAffil,$followfirst,
             $followlast,$followAffil) = $innerResult->getRowAt($j);
        $leaderAffil=str_replace('&','\&',$leaderAffil);
        $followAffil=str_replace('&','\&',$followAffil);

        if ($j % 48 ==0)
        {
          if ($j!=0)
            $LaTeXFullEvents.="\\end{tabular}\n";
          $LaTeXFullEvents.="\\newpage\n";
          if ($j==0)
            $LaTeXFullEvents.="\\center{\\Large Event $num --- $name
($numEntries Entries)}\\\\\ \n";
          else
            $LaTeXFullEvents.="\\center{\\Large Event $num --- $name (cont.)}\\\\\ \n";
          $LaTeXFullEvents.="\\begin{tabular}{lllr}\\hline\n";

        }
        if ($couple=="")
        {
          continue;
        }
        $line="\\underline{\\hspace{0.4in}} $couple & $leaderfirst
$leaderlast ($leaderAffil) & \\& $followfirst $followlast
($followAffil) &".($j+1)."\\\\\ \n";
        if ($j % 2 ==0)
          $LaTeXFullEvents.=$line;
        else
          $LaTeXFullEvents.="\\rowcolor{light}$line";
        if ($j % 6 ==5)
          $LaTeXFullEvents.="\\hline \n";

      }
    $startnumber=$j;
    $endnumber=(int)($j*1.10)+2;
    for ($k=$startnumber;$k<$endnumber;$k++)
    {
      if ($k % 48 ==0)
      {
        if ($k!=0)
          $LaTeXFullEvents.="\\end{tabular}\n";
        $LaTeXFullEvents.="\\newpage\n";
        if ($k==0)
          $LaTeXFullEvents.="\\center{\\Large Event $num --
$name ($numEntries Entries)} \n";
```

```
            else
                $LaTeXFullEvents.="\\center{\\Large Event $num --
$name (cont.)}\n";
              $LaTeXFullEvents.="\\begin{tabular}{llll}\\hline\n";

            }
          $line = "\\underline{\\hspace{0.4in}} &\hspace{40ex} &\hspace{40ex} & ".
  ($k+1)."\\\\ \n";

          if ($k % 2 ==0)
            $LaTeXFullEvents.=$line;
          else
            $LaTeXFullEvents .= "\\rowcolor{light}$line";
          if ($k % 6 ==5)
            $LaTeXFullEvents .= "\\hline \n";

        }

    $LaTeXFullEvents .= "\\end{tabular}\n";
}
$LaTeXAddDropSummary .= "\\end{document}\n";
$LaTeXFullEvents .= "\\end{document}\n";

$query =<<<END_QUERY
SELECT abbreviation , fulltext
FROM abbreviations JOIN
            organizations ON (name=fulltext)
ORDER BY abbreviation;
END_QUERY;
$result = $db->query($query);
$LaTeXAbbreviations = latexHeader("Abbreviations");
$numAffils =$result->numrows();
for ($i=0;$i < $numAffils;$i++)
{
  list($abbrev,$name)=$result->getRowAt($i);
        $abbrev=str_replace('&','\&',$abbrev);
        $name=str_replace('&','\&',$name);

  if ($i%48==0)
  {
    if($i!=0)
      $LaTeXAbbreviations.="\\end{tabular}\n";
    $LaTeXAbbreviations.="\\newpage\n";
    if($i==0)
      $LaTeXAbbreviations.="\\center{\\Large Abbreviations}\\\\\n";
    else
      $LaTeXAbbreviations.="\\center{\\Large Abbreviations (cont.)}\\\\\n";
    $LaTeXAbbreviations.="\\begin{tabular}{ll}\\hline\n";
  }
  $line="$abbrev & $name \\\\";
  if ($i%2==0)
    $LaTeXAbbreviations.=$line;
  else
    $LaTeXAbbreviations.="\\rowcolor{light}$line";
  if ($i%6==5)
    $LaTeXAbbreviations.="\\hline \n";
}
$LaTeXAbbreviations.="\\end{tabular}\n";
$LaTeXAbbreviations.="\\end{document}\n";
```

```
$partSeperator=md5(uniqid(time));

$body.=<<<END_BODY
This is a MIME Encoded Email\n\n--$partSeperator
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7 bit

This email was sent via SlidingDoors, the on-line registration
component of CompInaBox. Attached are raw LaTeX files containing the
registration data, designed for use by the head table.

Reply to this message to get in touch with the CompInaBox Server Administrator.

$CIB_SERVER_ADMIN_NAME
$CIB_SERVER_HOST_NAME CompInaBox Server Administrator

END_BODY;
$body.=<<<END_PART_HEADER
--$partSeperator
Content-Type: text/plain; charset=us-ascii; name="by-number.txt"
Content-Transfer-Encoding: 7 bit

$coupleText
--$partSeperator
Content-Type: text/plain; charset=us-ascii; name="by-school.txt"
Content-Transfer-Encoding: 7 bit

$textSchoolCouples
--$partSeperator
Content-Type: text/plain; charset=us-ascii; name="adddrop.tex"
Content-Transfer-Encoding: 7 bit

$LaTeXAddDropSummary
--$partSeperator
Content-Type: text/plain; charset=us-ascii; name="by-event.tex"
Content-Transfer-Encoding: 7 bit

$LaTeXFullEvents
--$partSeperator
Content-Type: text/plain; charset=us-ascii; name="abbreviations.tex"
Content-Transfer-Encoding: 7 bit

$LaTeXAbbreviations
--$partSeperator
Content-Type: text/plain; charset=us-ascii; name="by-number.tex"
Content-Transfer-Encoding: 7 bit

$coupleLaTeX
--$partSeperator--

END_PART_HEADER;
$headers=<<<END_HEADERS
From: $CIB_SERVER_ADMIN_EMAIL\r
X-mailer: CompInaBox (via PHP)\r
MIME-Version: 1.0\r
Content-Type: multipart/mixed;\r
        boundary="$partSeperator"
END_HEADERS;
mail("$mailto","[$unixname Competition] Data Export",$body,$headers);
```

```
echo "done";
?>
```

bin/latex_invoices.php

```
#!/usr/local/bin/php
<?php
/*************************************************************
 * This file is part of SlidingDoors, a CompInaBox Component *
 * Copyright 2001-2002. Eric D. Nielsen, All Rights Reserved *
 * CompInaBox is availible for license under the GPL, see    *
 * the COPYING file in the root directory of the install for *
 * the full terms of the GPL.                                *
 *************************************************************/

function collapseRegArray($registrants,&$total)
{
  $total=0;
  $tempArray=array();
  $arrayIndex=0;
  reset($registrants);
  while ($package = each($registrants))
    {
      reset($package["value"]);
      while ($section=each($package["value"]))
        {
          $sectionData = $section["value"]["Sponsored"];
          $label = $section["key"] . "-".
        substr($sectionData["Name"],0,strpos($sectionData["Name"]," "));
          $cost = $sectionData["Cost"];
          $entries = $sectionData["Entries"];
          $numEntries = count($entries);
          for ($j=0;$j<$numEntries;$j++)
            {
              $name = $entries[$j]["Name"];
              $tempP = $entries[$j]["Partners"];
              $partners = arrayToCSL($tempP);
              $entryCost = ($entries[$j]["Paid"]?"Pre-Paid":$cost);
              $total+=($entries[$j]["Paid"]?0:$cost);
              $tempArray[$arrayIndex++] = array("Line"=>$arrayIndex,
                        "Name"=>LaTeX::escapeText($name),
                        "Partners"=>LaTeX::escapeText($partners),
                        "Package"=>LaTeX::escapeText($label),
                        "Cost"=>LaTeX::escapeText($entryCost));
            }
        }
    }
  $sortedArray = sortLastName($tempArray);
  $numPeople = count($sortedArray);
  for ($i=0;$i<$numPeople;$i++)
    {
      $sortedArray[$i]["Line"]=$i+1;
    }
  return $sortedArray;
}

function arrayColumn($arr,$col)
{
  $len=count($arr);
  for ($i=0;$i<$len;$i++)
    if ($col!="AutoNum")
      $temp[$i]=$arr[$i][$col];
    else
      $temp[$i]=$i+1;
```

```
  return $temp;
}

function sortLastName($tempArray)
{
  $num = count($tempArray)-1;
  for ($i=0;$i<$num;$i++)
    {
      $subNum = $num-$i;
      for ($j=0;$j<$subNum;$j++)
        {
          $cur = $tempArray[$j]["Name"];
          $next = $tempArray[$j+1]["Name"];
          if (strcmp($cur,$next)>0)
            {
              $temp = $tempArray[$j];
              $tempArray[$j]=$tempArray[$j+1];
              $tempArray[$j+1]=$temp;
            }
        }
    }
  return $tempArray;
}

ini_alter("include_path",".");
include("../include/paths.inc");
ini_alter("include_path","$CIB_CLASS_PATH:$CIB_PREFIX/include");
if ($argv[1]=="") die("No Comp Name Provided\n");
$unixname = $argv[1];
$CIB_TOOL_NAME="register";
include "$CIB_CONFIG_PATH/tools/CompInaBox_config.inc";
include "$CIB_CONFIG_PATH/comps/{$unixname}_config.inc";
include "$CIB_INCLUDE_PATH/tools/SlidingDoors_header.inc";
include "$CIB_CLASS_PATH/LaTeX.inc";

?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
        "http://www.w3.org/TR/1999/REC-html401-19991224/loose.dtd">
<html lang="en">
<meta http-equiv="content-type" content="text/html; charset=iso-8859-15">
        <title>test</title>
</head>
<body>
<?php

set_time_limit(900);
$query =<<<END_QUERY
SELECT orgid, name, abbreviation
FROM organizations JOIN
     abbreviations ON (name=fulltext)
WHERE orgid>1;
END_QUERY;
$result = $db->query($query);
$numAffiliations = $result->numrows();
$directory="{$CIB_WEB_PATH}/register/$unixname/group/Invoices/";
chdir($directory);
for ($i=0;$i<$numAffiliations;$i++)
{
  list( $affilID, $name, $abbreviation) = $result->getRowAt($i);

  echo "$name started at " . date("r", time()) . " -- $abbreviation";
```

```
$org = new RegOrg($db);
$org->setID($affilID);
$org->retrieve();
$rep = $org->getRepString();
$tempPos = strpos($rep,";");
if ($tempPos)
  $rep=substr($rep,0,$tempPos);
$addr = $org->getAddress();
if ($addr!="")
  {
    $addrPieces = explode(";",$addr);
    $addr1=$addrPieces[0];
    $addr2=$addrPieces[1];
  }
else
  {
    $addr1="No Address Provided";
    $addr2="";
  }
$total=0;
$regArrayByPackage = $db->produceRegistrantArray($affilID);
$regArray = collapseRegArray($regArrayByPackage,$total);
if ($total==0) continue;
$filename = "$abbreviation"."_invoice";
$filename=str_replace(" ","_",$filename);
$LaTeX = LaTeX::beginDocument("$SD_compname","SlidingDoors",10);
$summaryCells=array();
$summaryCells[] = array(LaTeX::formatText("Invoice For:","Bold"),
                        LaTeX::escapeText($name),
                        LaTeX::formatText("Amount Due:","Bold"),
                        LaTeX::escapeText('$'.$total));

$summaryCells[] = array(LaTeX::formatText("Contact Person:","Bold"),
                        LaTeX::escapeText($rep),
                        LaTeX::formatText("Postmarked by:","Bold"),
                        LaTeX::escapeText("$SD_postmarked_by"));

$summaryCells[] = array(LaTeX::formatText("Contact Address:","Bold"),
                        LaTeX::escapeText($addr1),
                        LaTeX::formatText("Mail to:","Bold"),
                        LaTeX::escapeText($SD_payment_name));

$summaryCells[] = array("",
                        LaTeX::escapeText($addr2),
                        "",
                        LaTeX::escapeText($SD_payment_addr1));
$summaryCells[] = array("",
                        LaTeX::escapeText($addr2),
                        "",
                        LaTeX::escapeText($SD_payment_addr2));


$LaTeX .= LaTeX::produceLongtable($summaryCells,"|rp{1.5in}rp{1.5in}|",
                                  FALSE,TRUE);

$header = LaTeX::formatText("Line \\#","Bold") . " & " .
  LaTeX::formatText("Member Name","Bold") . " & " .
  LaTeX::formatText("Partners", "Bold") . " & ".
  LaTeX::formatText("Package", "Bold") . " & ".
```

```
    LaTeX::formatText("Cost", "Bold") . " \\\\\\n\\hline\n";
  $headerFirst = $header;
  $footer = "\\hline\n".LaTeX::spanRow(
        LaTeX::formatText("Continued on Next Page","SmallCaps"),5);
  $footerLast = "\\hline\n".LaTeX::spanRow(
        LaTeX::formatText("Itemized Listing Finished","SmallCaps"),5);



  $LaTeX .= LaTeX::produceLongtable($regArray,"lp{2.0in}p{2.0in}ll",
                                    TRUE,FALSE,
                                    $header,$headerFirst,
                                    $footer,$footerLast);
  $LaTeX .= LaTeX::formatText("Payment Instructions:","Bold") .
    " Send a check made ".
    "out to '$SD_payment_to' to the address listed at the top of the ".
    "first page.  If you need to make changes you may line out items ".
    "and/or write-in.\n\n";
  $LaTeX .= LaTeX::endDocument(TRUE);

#  echo "<pre><br /><br />".htmlspecialchars($LaTeX)."</pre>";
  $fp = fopen("{$directory}{$filename}.tex", "w");
  fwrite($fp,$LaTeX);
  fclose($fp);
  $safeFile = escapeshellarg("{$directory}{$filename}.tex");
  exec("pdflatex $safeFile");
  exec("pdflatex $safeFile"); // needs to be run twice
}
?>
</body>
</html>
```

bin/newresults.php

```
#!/usr/local/bin/php
<?php
$chesterToEventName["Ch"]="Cha Cha";
$chesterToEventName["Sa"]="Samba";
$chesterToEventName["Ru"]="Rumba";
$chesterToEventName["Pa"]="Paso Doble";
$chesterToEventName["Ji"]="Jive";
$chesterToEventName["Wa"]="Waltz";
$chesterToEventName["Ta"]="Tango";
$chesterToEventName["Vi"]="Viennese Waltz";
$chesterToEventName["Fo"]="Foxtrot";
$chesterToEventName["Qu"]="Quickstep";
$chesterToEventName["Sw"]="Swing";
$chesterToEventName["Bo"]="Bolero";
$chesterToEventName["Ma"]="Mambo";

function filterBlanks($var)
{
  $val = str_replace(" ","",$var);
  return ($var!="");
}

function buildAffilLut($db)
{
  $query =<<<END_QUERY
    SELECT DISTINCT orgid, name, abbreviation
    FROM organizations
    LEFT OUTER JOIN abbreviations ON (name=fulltext)
    RIGHT OUTER JOIN people ON (orgid=organization)
    JOIN couples ON (leader=peopleid OR follower=peopleid)
    ORDER BY name;
END_QUERY;
$result=$db->query($query);
 $lut=array();
 $numAffils=$result->numrows();
 for ($i=0;$i<$numAffils;$i++)
   {
     list($orgid,$name,$abbreviation) = $result->getRowAt($i);
     if ($name=="")$name="Independent";
     if ($abbreviation=="") $abbreviation=$name;
     $lut["$orgid"]["name"]=$name;
     $lut["$orgid"]["abbrev"]=$abbreviation;
     $lut["-1"]["name"]="Independent";
     $lut["-1"]["abbrev"]="Independent";
   }
 return $lut;
}



ini_alter("include_path",".");
include("../include/paths.inc");
ini_alter("include_path","$CIB_CLASS_PATH:$CIB_PREFIX/include");

if ($argv[1]=="") die("No Comp Name provided\n");
$unixname = $argv[1];
$CIB_TOOL_ROLE="register";
include "$CIB_CONFIG_PATH/comps/{$unixname}_config.inc";
```

```
include "$CIB_INCLUDE_PATH/tools/SlidingDoors_header.inc";
ini_alter("include_path",
"$CIB_WEB_PATH/$CIB_WEB_INSTALL/$CIB_TOOL_ROLE/$unixname/PREPARSED");


$judgeList = array("Mark Nocera","Douglas Banks","Judit Gaspar",
                   "Suzanne Hamby","Christine Harvey","Richard Hutton",
                   "Armin Kappacher","Tibor Kerekes","Peter Kovacs",
                   "Russell Monk","Dawna Nocera","Anne Marie Paul",
                   "Dan Radler","Gail Rundlett","Helle Rusholt");
$competitorList=array();
$LUT=buildAffilLut($db);
$inputFilename = $argv[2];
$outputFilename = $argv[3];

if ($inputFilename=="" || $outputFilename=="")
        die ("Please provide both an input and an output filename");

$fp = fopen("$inputFilename","r");
$file = fread($fp,filesize("$inputFilename"));
$lines = explode("\r",$file);
if (count($lines)==1)
        $lines = explode("\n",$file);
$output = "<html><head><title>$lines[0]</title></head>\n";
$output .= "<body bgcolor=white><h1>$lines[0]</h1>\n";
$numLines = count($lines);
$i=1;
while ($i<$numLines)
{
        $line = $lines[$i];

        if (substr($line,0,1)=='E')
        {
            $temp = explode('_',$line);
            $eventid = substr($temp[2],0,strlen($temp[2])-1));
            $temp = explode(',',$line);
            $output .= "<h2>$temp[1]</h2>\n";
            $i+=3;
            $judges=array();
            $judgeLine1 = $lines[$i-1];
            $judgeLine2 = $lines[$i];
            $lineLength = strlen($judgeLine2);
            $dances=array();
            $danceName="";
            for ($j=0;$j<$lineLength;$j++)
            {
                $ones = $judgeLine2[$j];
                $tens = $judgeLine1[$j];;
                if (!is_numeric($ones))
                  {
                    if ($danceName!="" && $ones==" ")
                      {
                        $dances[]=$chesterToEventName[$danceName];
                        $danceName="";
                      }
                    if ($ones>='A' && $ones<='Z')
                      $danceName.=$ones;
                    else if ($danceName!="" && $ones!=" ")
                      $danceName .=$ones;
                    continue;
                  }
```

```
            if (is_numeric($tens))
                $judge = $tens * 10 + $ones;
            else
                $judge = $ones;
            $judges[]=$judge;
        }
        $numJudges = count($judges);
        if ($numJudges==0)
        {
            # begin final output
            $line = $lines[$i-1];
            $pieces = explode(" ",trim($line));
            $pieces=array_values(array_filter($pieces,"filterBlanks"));
            $numPieces = count($pieces);
            $numDances = $numPieces-3;
            $output .= "<table cellspacing=0 cellpadding=4 border=1><tr>
                    <th>Couple</th>";
            $output .= "<th nowrap=\"nowrap\">Competitor Names</th>
<th>Overall Place</th><td></td><th>Total</th><td></td>\n";

            for ($index=0;$index<$numDances;$index++)
              {
                $output .= "<th>{$chesterToEventName[$pieces[$index+1]]}</th>";
              }
            $output .= "</tr>";
            $i+=1;
            $line = $lines[$i];
            $pieces = array_values(array_filter(explode(" ",trim($line)),
                        "filterBlanks"));
            $numPieces = count($pieces);
            while ($numPieces==1 ||is_numeric($pieces[0]))
            {
                if ($numPieces==1)
                {
                    $i++;
                    $line = $lines[$i];
                    $pieces = explode(" ",trim($line));
                    $numPieces = count($pieces);
                    continue;
                }
                $output .= "<tr><td align=\"center\">$pieces[0]</td>";
                $num=$pieces[0];
                if (!in_array($num,array_keys($competitorList)))
                  {
                $query = "SELECT l.firstname, l.lastname, l.organization, ";
                $query .= "f.firstname, f.lastname, f.organization FROM ";
                $query .= "events_registration natural join couples ";
                $query .= "JOIN people as l ON (leader=l.peopleid) ";
                $query .= "JOIN people AS f ON (follower=f.peopleid) ";
                $query .= "Natural Join events ";
                $query .= "WHERE competitornumber=$num AND programnumber=$eventid;";
                        $result = $db->query($query);
                        if ($result->numrows())
                          {
                            list($lFirst, $lLast, $lAffilID,
                                 $fFirst, $fLast, $fAffilID) = $result->getRowAt(0);
                            if ($lFirst=="Mark")echo":$lAffilID?$fAffilID:";
                            if ($lAffilID=="")$lAffilID=-1;
                            if ($fAffilID=="")$fAffilID=-1;
                            if ($lFirst=="Mark")echo":$lAffilID?$fAffilID:";
```

```
            $lAffil=$LUT["$lAffilID"]["abbrev"];
            $fAffil=$LUT["$fAffilID"]["abbrev"];
            $competitorList[$num]=
"$lFirst $lLast ($lAffil) & $fFirst $fLast ($fAffil)";
          }
        else
          $competitorList[$num]="No names available at present";
      }

    $output .= "<td nowrap=\"nowrap\"> ".
        $competitorList[$pieces[0]]."</td>\n";

    if (substr($pieces[$numPieces-1],0,1)=='R')
      {
        $place = $pieces[$numPieces-2] . " " .$pieces[$numPieces-1];
        $total = $pieces[$numPieces-3];
      }
    else
      {
        $place = $pieces[$numPieces-1];
        $total = $pieces[$numPieces-2];
      }

    $output .= "<td align=\"center\">$place</td>";
    $output .= "<td></td>";
    $output .= "<td align=\"center\">$total</td>";
    $output .= "<td></td>";
    for ($index=0;$index<$numDances;$index++)
      {
        $output .= "<td align=\"center\">{$pieces[$index+1]}</td>";
      }
    $output .= "</tr>";
    $i++;
    $line = $lines[$i];
    $pieces = array_values(array_filter(explode(" ",
        trim($line))," filterBlanks"));
    $numPieces = count($pieces);
}
$i++;
$line = $lines[$i];
$modLine = ereg_replace('[[:space:]]+',' ',$line);
$pieces = explode(" ",$modLine);
$dance = $pieces[0];
$numPieces=count($pieces);
while ($numPieces!=3)
  {
    echo $line."\n";
    $i++;
    $line = $lines[$i];
    $modLine = ereg_replace('[[:space:]]+',' ',$line);
    $pieces = explode(" ",$modLine);
    $numPieces=count($pieces);
    $judges=array();
    $low=0;

    for ($j=1;$j<count($pieces);$j++)
      {
        if ($pieces[$j]>$low)
          {
            $low = $pieces[$j];
            $judges[]=$pieces[$j];
```

```
                    }
                  else
                    break;
                }
              $places=array();
              $low=0;
              for (;$j<count($pieces);$j++)
                {
                  if ($pieces[$j]>$low)
                    {
                      $low = $pieces[$j];
                      $places[]=$pieces[$j];
                    }
                  else
                    break;
                }
              $numJudges = count($judges);

              $numPlaces = count($places)-1;
              $output .= "</table><br />";

              $output .= "<table cellspacing=0 cellpadding=4 border=1>";
              $output .= "<tr><th>$dance</th><th rowspan=2>
Competitor Names</th>";
              $output .= "<th colspan=$numJudges>Marks by Judge</th>
<th>  </th><th colspan=$numPlaces>Cumulative Marks</th><th></th></tr>\n";


              $output .= "<tr><th>Couple</th>";
              for($j=0;$j<$numJudges;$j++)
                $output .= "<th><a href=\"judges.html\" title=\"".
$judgeList[$judges[$j]-1]."\">$judges[$j]</a></th>";
              $output.="<th>  </th>";
              for($j=0;$j<$numPlaces;$j++)
                $output .= "<th>$places[$j]</th>";
              $output.="<th>Place</th></tr>";
              $i+=2;;

              $line = $lines[$i];
              $modLine = ereg_replace('[[:space:]]+',",",$line);
              $pieces = explode(",",$modLine);
              $numPieces= count($pieces);
              while ($numPieces!=1 && $numPieces!=3)
                {
                  $output.="<tr>";
                  for ($j=1;$j<$numPieces;$j++)
                    {
                      if ($j==2+$numJudges)
                        $output.="<td></td>";
                      $output .= "<td align=\"center\">$pieces[$j]</td>";
                      if ($j==1) $output.=
        "<td nowrap=\"nowrap\">{$competitorList[$pieces[$j]]}</td>";
                    }
                  $output.="</tr>";
                  $i++;
                  $line = $lines[$i];
                  $modLine = ereg_replace('[[:space:]]+',",",$line);
                  $pieces = explode(",",$modLine);
                  $numPieces= count($pieces);
                }
              $i++;
```

```
                $line = $lines[$i];
                $modLine = ereg_replace('[[:space:]]+',' ',$line);
                $pieces = explode(" ",$modLine);
                $numPieces= count($pieces);
                $dance=$pieces[0];
              }
            break; // file is done being processed
          }
          else
          {
            #begin regular round output
            $i+=1;
            $output .= "<table border=1 cellspacing=0 cellpadding=3>";
                $output .= "<tr><th colspan=3></th>";
#           if (count($dances)==1)
#             {
#               $output .= "<th colspan=$numJudges>Judge</th></tr>";
#             }
#           else
#             {
                $numDances = count($dances);
                $numJudgesDance=$numJudges/$numDances;
                for ($index=0;$index<$numDances;$index++)
                  {
                    $colSpan=$numJudgesDance+1;
                    $output .= "<th colspan=$colSpan>{$dances[$index]}</th>";

                  }
                $output .= "</tr>";

#             }
            $output .= "<tr><th>Couple</th><th>Names</th><th>Total</th>";

            for ($j=0;$j<$numJudges;$j++)
{
                if (($j % $numJudgesDance)==0)
                  $output .= "<td align=\"center\">Marks</td>";
                $output .= "<th><a href=\"judges.html\" title=\"".
$judgeList[$judges[$j]-1]."\">". ($judges[$j]<10 ? " " : "") .
"$judges[$j]</a></th>";
}
            $output .= "</tr>";
          }
        }
        if ($line=="")
        {
            $i++;
            continue;
        }
        if (in_array($line[0],array('A','S')))
        {
            $output .= "</table>";
            $i++;
            continue;
        }
        if (strstr($line,"Couple")) break;
        if (is_numeric(substr($line,0,2)))
        {
            if (is_numeric(substr($line,0,1)))
            {
                $j=1;
```

```
            $num=$line[0];
        }
        else
        {
            $j=2;
            $num=$line[1];
        }
        while (is_numeric($line[$j]))
        {
            $num = $num *10 + $line[$j];
            $j++;
        }
        if (!in_array($num,array_keys($competitorList)))
        {
            $query = "SELECT l.firstname, l.lastname, l.organization, ";
            $query .= "f.firstname, f.lastname, f.organization FROM ";
            $query .= "events_registration natural join couples ";
            $query .= "JOIN people as l ON (leader=l.peopleid) ";
            $query .= "JOIN people AS f ON (follower=f.peopleid) ";
            $query .= "Natural Join events ";
            $query .= "WHERE competitornumber=$num AND programnumber=$eventid;";
            $result = $db->query($query);
            if ($result->numrows())
            {
                list($lFirst, $lLast, $lAffilID,
                     $fFirst, $fLast, $fAffilID) = $result->getRowAt(0);
                if ($lFirst=="Mark")echo":$lAffilID?$fAffilID:";
                if ($lAffilID=="")$lAffilID=-1;
                if ($fAffilID=="")$fAffilID=-1;
                if ($lFirst=="Mark")echo":$lAffilID?$fAffilID:";

                $lAffil=$LUT["$lAffilID"]["abbrev"];
                $fAffil=$LUT["$fAffilID"]["abbrev"];
                $competitorList[$num]="$lFirst $lLast ($lAffil) &
$fFirst $fLast ($fAffil)";
            }
            else
                $competitorList[$num]="No names available at present";
        }

        $output .= "<tr><td><center>$num</center></td>";
        $output.="<td nowrap=\"nowrap\">$competitorList[$num]</td>";

        while ($line[$j]==" ") $j++;
        $total="";
        while($line[$j]!=" " || $total=="")
        {
          if (is_numeric($line[$j]))
            $total.=$line[$j];
          $j++;
        }
        $output .= "<td align=\"center\">$total</td>";
        $numDances=count($dances);
        $numMarksDance= $numJudges/$numDances;
        $lineLength=strlen($line);
        for ($index=0;$index<$numDances;$index++)
          {

            $subTotal="";
            while($line[$j]!=" " || $subTotal=="")
              {
```

```
                if (is_numeric($line[$j]))
                  $subTotal.=$line[$j];
                $j++;
                if ($j > $lineLength) break;
              }
            $output.="<td align=\"center\">$subTotal</td>";
            $end = $j+$numMarksDance;
            $j++;
            while ($j<=$end)
              {

                if ($line[$j]!=" " && $j <$lineLength)
                  {
                    $output .= "<td align=\"center\" bgcolor=\"#ccddee\">X</td>";
                  }
                else
                    {
                      $output .= "<td> </td>";
                    }
                $j++;
              }
          }
        $output.="</tr>\n\n";
      }
      $i++;
}

$fp = fopen("$CIB_WEB_PATH/results/$unixname/$outputFilename","w");
fwrite($fp,$output);
?>
```

bin/postgres-analyze

```
#!/bin/sh
COMPNAME=$1
su - postgres -c "echo vacuum analyze | psql -e $COMPNAME"
exit 0;
```

bin/postgres-backup

```
#!/bin/sh
COMPNAME=$1
timeinfo=`date +%Y-%m-%d`
su - postgres -c "pg_dump $COMPNAME | gzip > \
/usr/local/compinabox/var/backups/$COMPNAME-$timeinfo.gz"
exit 0;
```

bin/postgres-vacuum

```
#!/bin/sh
COMPNAME=$1
su - postgres -c "echo vacuum | psql -e $COMPNAME"
exit 0;
```

bin/rebuild_sd_stats.php

```
#!/usr/local/bin/php
<?php
#####################################################################
# This file is part of SlidingDoors                                 #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd        #
# SlidingDoors is availible for license under the GPL, see          #
# the COPYING file in the root directory of the install for         #
# the full terms of the GPL.                                        #
#                                                                   #
# Please Note: a revised version of the stats system is in use      #
# on a hacked production system.  The changes will be merged        #
# back to this tree shortly.                                        #
#####################################################################
include("paths.inc");
if ($argv[1]=="") die("No Comp Name provided\n");
$unixname = $argv[1];
$CIB_TOOL_ROLE="register";
include "$CIB_CONFIG_PATH/comps/{$unixname}_config.inc";
include "$CIB_INCLUDE_PATH/tools/SlidingDoors_header.inc";
ini_alter("include_path",
"$CIB_WEB_PATH/$CIB_WEB_INSTALL/$CIB_TOOL_ROLE/$unixname/PREPARSED");
$db->rebuildStats();
?>
```

bin/simple_export.php

```
#!/usr/local/bin/php
<?php
    include("paths.inc");
if ($argv[1]=="") die("No Comp Name provided\n");
if ($argv[2]=="") die("No email address provided for document delivery\n");
$unixname=$argv[1];
if (!ereg("^[-a-zA-Z0-9_]*$",$unixname)) die ("Badly formed compname\n");
$mailto = $argv[2];


$CIB_TOOL_ROLE="register";
include "$CIB_CONFIG_PATH/comps/{$unixname}_config.inc";
if (!isset($SD_compname)) die ("Invalid Comp Name.");

include "$CIB_INCLUDE_PATH/tools/SlidingDoors_header.inc";
ini_alter("include_path",
"$CIB_WEB_PATH/$CIB_WEB_INSTALL/$CIB_TOOL_ROLE/$unixname/PREPARSED");


$query =<<<END_QUERY
SELECT DISTINCT competitornumber,
                l.firstname, l.lastname, lo.name,
                f.firstname, f.lastname, fo.name
FROM events_registration NATURAL JOIN
     couples JOIN
     people AS l ON (leader=l.peopleid) LEFT JOIN
     organizations AS lo ON (l.organization=lo.orgid) JOIN
     people AS f ON (follower=f.peopleid) LEFT JOIN
     organizations AS fo ON (f.organization=fo.orgid)
ORDER BY lo.name, l.lastname, l.firstname, f.lastname, f.firstname;
END_QUERY;

$result=$db->query($query);
$numrows=$result->numrows();
$partSeperator=md5(uniqid(time));

$body.=<<<END_BODY
This is a MIME Encoded Email\n\n--$partSeperator
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

This email was sent via SlidingDoors, the on-line registration
component of CompInaBox.  Attached are raw text files containing the
registration data, suitible for formatting into the printed program or
manually entering into a scrutineering program.  Other formats may eventaully
be offered upon demand.

Reply to this message to get in touch with the CompInaBox Server Administrator.

$CIB_SERVER_ADMIN_NAME
$CIB_SERVER_HOST_NAME CompInaBox Server Administrator

END_BODY;

$body.=<<<END_PART_HEADER
--$partSeperator
Content-Type: text/plain; charset=us-ascii; name="by-school.txt"
Content-Transfer-Encoding: 7bit
```

```
END_PART_HEADER;
$file1="";
for ($i=0;$i<$numrows;$i++)
{
    list ($number, $lfname, $llname, $lorg, $ffname,
$flname, $forg) = $result->getRowAt($i);

    $file1.="\n$number\t$llname, $lfname ($lorg) & $flname, $ffname ($forg)";
}
$body.="$file1\n";

$body.=<<<END_PART_SEPERATOR
--$partSeperator
Content-Type: text/plain; charset=us-ascii; name="by-event.txt"
Content-Transfer-Encoding: 7bit


END_PART_SEPERATOR;

$query = "Select eventid, longname from events_names;";
$result = $db->query($query);
$numEvents = $result->numrows();
for ($i=0; $i<$numEvents; $i++)
{
    list($eventid,$eventName)= $result->getRowAt($i);
    $body .="\n\n$eventName";
    $query = "Select competitornumber from events_registration where
eventid=$eventid order by competitornumber;";
    $subresult=$db->query($query);
    $numEntrants = $subresult->numrows();
    for ($j=0;$j<$numEntrants;$j++)
    {
        if (($j % 12) ==0) $body.="\n";
        list($number)=$subresult->getRowAt($j);
        $body .= "  $number";
    }
}

$body.="\n--$partSeperator--\n";
$headers="";
$headers.=<<<END_HEADERS
From: $CIB_SERVER_ADMIN_EMAIL\r
X-mailer: CompInaBox (via PHP)\r
MIME-Version: 1.0\r
Content-Type: multipart/mixed;\r
        boundary="$partSeperator"
END_HEADERS;
mail("$mailto","[$unixname Competition] Data Export",$body,$headers);



?>
```

B.2 Configuration Files

include/include_others.inc

```
<?php
/*****************************************************************
 * This file is part of the Core module of CompInaBox            *
 * Copyright 2002. Eric D. Nielsen, All rights reserverd         *
 * CompInaBox is availible for license under the GPL, see        *
 * the COPYING file in the root directory of the install for     *
 * the full terms of the GPL.                                    *
 *                                                               *
 * File: include_others.inc                                      *
 * Type: Include file                                            *
 * Author: Eric D. Nielsen                                       *
 * Description: Master include file for CIB components, parses   *
 *              the URI to determin which other include files    *
 *              are needed by the current script                 *
 *****************************************************************/

require("paths.inc");

// redirect to a secure page, if ssl should be used for all transactions and
// currently on an insecure page.
if ($CIB_USES_SSL=="ALWAYS"
    && strpos(".".$HTTP_SERVER_VARS["SERVER_PROTOCOL"],"HTTP/"))
{
  header("Location: https://$CIB_HOSTNAME".".$CIB_DOMAIN/$PHP_SELF");
  exit;
}

// redirect to a insecure page, if ssl should not be used at all and
// currently on a secure page.
if ($CIB_USES_SSL=="NEVER"
    && strpos(".".$HTTP_SERVER_VARS["SERVER_PROTOCOL"],"HTTPS/"))
{
  header("Location: http://$CIB_HOSTNAME".".$CIB_DOMAIN/$PHP_SELF");
  exit;
}

// if we only use ssl sometimes is the responsibilities of ssl requiring
// pages to handle the redirect if reached insecurely

// Now we need to grab the toolname and the compname: these will be the first
// and second directory components of the URL.

$CIB_REQUEST_PATH = substr($PHP_SELF,strlen($CIB_WEB_INSTALL_DIR)+2);
$CIB_PATH_COMPONENTS = explode('/',$CIB_REQUEST_PATH);

$CIB_NUM_PATH_COMPONENTS = count($CIB_PATH_COMPONENTS);
if ($CIB_NUM_PATH_COMPONENTS==1 || $CIB_PATH_COMPONENTS[0]=="SCRIPTS" ||
    $CIB_PATH_COMPONENTS[0]=="accounts")
{
  $CIB_TOOL_ROLE="core";
}
else
{
  $CIB_TOOL_ROLE = $CIB_PATH_COMPONENTS[0];
}
```

```
$CIB_TOOL_NAME = $CIB_ROLE_TO_NAME_LUT[$CIB_TOOL_ROLE];
if ($CIB_NUM_PATH_COMPONENTS>2)
{
  $CIB_COMP_UNIXNAME = $CIB_PATH_COMPONENTS[1];
  if ($CIB_COMP_UNIXNAME=="SCRIPTS")
    $CIB_COMP_UNIXNAME = "";
}
else
{
  $CIB_COMP_UNIXNAME="";
}

include("$CIB_CONFIG_PATH/tools/{$CIB_TOOL_NAME}_config.inc");
if (""!=$CIB_COMP_UNIXNAME)
{
  include("$CIB_CONFIG_PATH/comps/{$CIB_COMP_UNIXNAME}_config.inc");
}
include("$CIB_INCLUDE_PATH/tools/{$CIB_TOOL_NAME}_header.inc");
srand(microtime());
session_start();
if (isset($authorizationLevels)) $authenticationRequired=TRUE;
if ($authenticationRequired)
{
  include_once("$CIB_INCLUDE_PATH/session_user_setup.inc");
  newTimestamp("Found user permissions");
  $gate = new Gate();
  $guard = new Guard();
  if ($authenticationRequired)
    $gate->lowerPortcullis();
  if (isset($authorizationLevels))
    $gate->lockGate($authorizationLevels);
  newTimestamp("Locked gate");
  if (isset($user) && $guard->recognizesUser($user->getID()))
    {
      $guard->unlockGateWithKeys($gate,$user->getKeys(),
                                 "$CIB_BASE_URL/access-denied.php");
      newTimestamp("Successful unlocked gate");
    }
  else
    {
      $guard->referToScribe();
      exit;
    }
}

?>
```

include/paths.inc

```
<?php
/*************************************************************
 * This file is part of the Core module of CompInaBox        *
 * Copyright 2002. Eric D. Nielsen, All rights reserverd     *
 * CompInaBox is availible for license under the GPL, see    *
 * the COPYING file in the root directory of the install for *
 * the full terms of the GPL.                                *
 *                                                           *
 * File: paths.inc                                           *
 * Type: Include file                                        *
 * Author: Eric D. Nielsen                                   *
 * Description: Set up paths needed for both command-line and *
 *              web php.                                     *
 *************************************************************/

$CIB_PREFIX        = "/usr/local/compinabox";
$CIB_CLASS_PATH    = "$CIB_PREFIX/include/classes";
$CIB_CONFIG_PATH   = "$CIB_PREFIX/include/conf";
$CIB_INCLUDE_PATH = "$CIB_PREFIX/include/general";
require("$CIB_CONFIG_PATH/tools/CompInaBox_config.inc");
?>
```

include/pre_main_include.inc

```
<?php
function getCompNameFromURL($url,$section)
{
  $path = explode("/",$url);
  $i=0;
  $numDirs = count($path)-1;
  while($path[$i]!=$section && $i<$numDirs)
    $i++;
  if ($path[$i]==$section)
    return $path[$i+1];
  die("No competition name found in a competition URL, something is very wrong.");
}


?>
```

include/conf/comps/SD_config.inc

```
<?php
# This file is generated by PrepStep do not edit by hand

##################################################################
## Comp Identification                                          ##
##################################################################
$unixname             = "SD";
$SD_compname          = "SlidingDoors Reference Install";
$SD_comp_url          = "compinabox.sourceforge.net/";
$SD_venue             = "Building; Street Addr; City, State";
$SD_payment_addr1     = "5 Casco Drive Apt E";
$SD_payment_addr2     = "Nashua NH, 03062";
$SD_date              = "Tomorrow";
$SD_deadline          = "Yesterday";
$SD_latefee           = "Today";
$SD_compcoord_title   = "Author";
$SD_compcoord_name    = "Eric Nielsen";
$SD_compcoord_email   = "nielsene@mit.edu";
$SD_registrar_title   = "Need hosting?";
$SD_registrar_name    = "Eric Nielsen";
$SD_registrar_email   = "nielsene@mit.edu";

##################################################################
## Sliding Doors Configuration Options                          ##
##################################################################
$SD_default_org_pays        = TRUE;
$SD_default_age             = 1; #Student
$SD_default_package         = 2; #Regular
$SD_elements                = array("Name",
                                    "Email",
                                    "Fee Category",
                                    "Package",
                                    "Affiliation");
$SD_lookup_by               = array("Email");
$SD_allow_TBAs              = TRUE;
$SD_prime_is_always_leader  = FALSE;
$SD_contraCheck_URL         = "NULL";

##################################################################
## Sliding Doors Appearance Options                             ##
##################################################################
$SD_style_sheet             = "SD_MITBDT2002EDN.css";
$SD_background_color        = "#ccddee";
$SD_text_background_color   = "white";
$SD_text_color              = "black";
$SD_error_background_color  = "#f0f090";
$SD_error_text_color        = "red";
$SD_done_background_color   = "#eeeeee";
$SD_title_color             = "#555555";
$SD_title_font              = "font-family: Verdana, sans-serif";
$SD_button_style            = "background: #d9a";

##################################################################
## Stats Page Congiguration options                             ##
##################################################################
$peoplePerPage = 20;

##########Do not change the values after this line###########
```

```
#####################################################################
## Comp-ina-Box Globals, do not touch                               ##
## These values are generated when CiB is installed                 ##
#####################################################################
$compdbName = "SDdemo";
$compWebUser = "OIuser";
$compWebPass = "L3tsR3g!";

#####################################################################
## Derived Variables, do not touch                                  ##
## These variables are built out of the one assigned above          ##
#####################################################################
$baseURL    = "$CIB_BASE_URL/$CIB_TOOL_ROLE/$unixname";
$secureURL = "$CIB_SECURE_URL/$CIB_TOOL_ROLE/$unixname";
$basePath   = "$CIB_WEB_PATH/$CIB_TOOL_ROLE/$unixname";

#####################################################################
## End Configuration File                                           ##
#####################################################################
?>
```

include/conf/tools/CompInaBox_config.inc

```
<?php
// Hostname, protocols, paths
$CIB_HOSTNAME                    = "ballroom";
$CIB_DOMAIN                      = ".mit.edu";
$CIB_USES_SSL                    = "LOGIN"; // NEVER, LOGIN, ALWAYS
$CIB_WEB_ROOT                    = "/var/www/";
$CIB_WEB_INSTALL_DIR             = "competitions";

// Sponsor/hosting org name/link
$CIB_SERVER_HOST_NAME            = "MITBDT";
$CIB_SERVER_HOST_LINK            = "http://ballroom.mit.edu";
$CIB_SERVER_ADMIN_NAME           = "Eric Nielsen";
$CIB_SERVER_ADMIN_EMAIL          = "nielsene@mit.edu";

// Database access information
$CIB_DB_NAME                     = 'cib_central';
$CIB_UNAUTH_USER                 = 'cib_unauth_user';
$CIB_UNAUTH_PASS                 = 'N3WCh@ng3Me!';
$CIB_AUTH_USER                   = 'cib_auth_user';
$CIB_AUTH_PASS                   = 'N3WUpd@t3Me!';
$CIB_ADMIN_USER                  = 'cib_admin_user';
$CIB_ADMIN_PASS                  = 'N3WSup3rS3kr!t';

// Combined values of the above, shouldn't need editting, if they do, please
// let us know what options should be added to the above set to make these
// not require hand editting

switch($CIB_USES_SSL)
{
 case "NEVER":
   $CIB_BASE_URL="http://$CIB_HOSTNAME"."$CIB_DOMAIN/$CIB_WEB_INSTALL_DIR";
   $CIB_SECURE_URL=$CIB_BASE_URL;
   break;
 case "ALWAYS" :
   $CIB_SECURE_URL="https://$CIB_HOSTNAME"."$CIB_DOMAIN/$CIB_WEB_INSTALL_DIR";
   $CIB_BASE_URL=$CIB_SECURE_URL;
   break;
 case "LOGIN":
 default:
   $CIB_BASE_URL="http://$CIB_HOSTNAME"."$CIB_DOMAIN/$CIB_WEB_INSTALL_DIR";
   $CIB_SECURE_URL="https://$CIB_HOSTNAME"."$CIB_DOMAIN/$CIB_WEB_INSTALL_DIR";
   break;
}

$CIB_WEB_PATH="$CIB_WEB_ROOT$CIB_WEB_INSTALL_DIR";
$CIB_ADMIN_EMAIL="$CIB_SERVER_ADMIN_EMAIL";

$CIB_ROLE_TO_NAME_LUT["core"]="CompInaBox";
$CIB_ROLE_TO_NAME_LUT["register"]="SlidingDoors";
$CIB_ROLE_TO_NAME_LUT["stats"]="SlidingDoors";
$CIB_ROLE_TO_NAME_LUT["plan"]="PrepStep";



?>
```

include/conf/tools/SlidingDoors_config.inc

```
<?php
# This file is generated by PrepStep do not edit by hand

##################################################################
## Comp Identification                                          ##
##################################################################
$unixname             = "";
$SD_compname          = "";
$SD_comp_url          = "";
$SD_venue             = "";
$SD_payment_addr1     = "";
$SD_payment_addr2     = "";
$SD_date              = "";
$SD_deadline          = "";
$SD_latefee           = "";
$SD_compcoord_title = "Competition Coordinator";
$SD_compcoord_name    = "";
$SD_compcoord_email = "";
$SD_registrar_title = "Registration Questions";
$SD_registrar_name    = "";
$SD_registrar_email = "";

##################################################################
## Sliding Doors Configuration Options                          ##
##################################################################
$SD_default_org_pays          = TRUE;
$SD_default_age               = 1;
$SD_default_package           = 1;
$SD_elements                  = array("Name",
                                      "Email",
                                      "Fee Category",
                                      "Package",
                                      "Affiliation");
$SD_lookup_by                 = array("Email");
$SD_allow_TBAs                = TRUE;
$SD_prime_is_always_leader = FALSE;
$SD_contraCheck_URL           = "NULL";

##################################################################
## Sliding Doors Appearance Options                             ##
##################################################################
$SD_style_sheet               = "SD_MITBDT2002EDN.css";
$SD_background_color          = "#ccddee";
$SD_text_background_color     = "white";
$SD_text_color                = "black";
$SD_error_background_color = "#f0f090";
$SD_error_text_color          = "red";
$SD_done_background_color     = "#eeeeee";
$SD_title_color               = "#555555";
$SD_title_font                = "font-family: Verdana, sans-serif";
$SD_button_style              = "background: #d9a";

##################################################################
## Stats Page Congiguration options                             ##
##################################################################
$peoplePerPage = 20;

##########Do not change the values after this line##########
```

```
$compdbName  = "";
$compWebUser = "";
$compWebPass = "";
#################################################################
## Derived Variables, do not touch                             ##
## These variables are built out of the one assigned above     ##
#################################################################

$baseURL    = "$CIB_BASE_URL/$CIB_TOOL_ROLE/$unixname";
$secureURL  = "$CIB_SECURE_URL/$CIB_TOOL_ROLE/$unixname";
$basePath   = "$CIB_WEB_PATH/$CIB_TOOL_ROLE/$unixname";

#################################################################
## End Configuration File                                      ##
#################################################################
?>
```

include/general/session_user_setup.inc

```
<?php
/*************************************************************
 * This file is part of CompInaBox.                          *
 * Copyright 2001-2002. Eric D. Nielsen, All rights reserverd *
 * CompInaBox is availible for license under the GPL, see    *
 * the COPYING file in the root directory of the install for *
 * the full terms of the GPL.                                *
 *                                                           *
 * This file setups the user object for use by the security  *
 * system.  It should be included at the top level of files  *
 * that require authentication/authorization services.       *
 *************************************************************/
#include_once("Logger.inc");
#$logger = new Logger();

$userID=$HTTP_SESSION_VARS["security_userID"];
if (isset($userID))
{
  $sessionID=session_id();
  // Yes this isn't perfect, a better solution should use
  // X-Forwarded-For if present.
  $clientIP = $HTTP_SERVER_VARS["REMOTE_ADDR"];
  $stored_session_hash=$HTTP_SESSION_VARS["security_hash"];
  $test_hash = md5($userID . $sessionID . $clientIP);
  if ($stored_session_hash != $test_hash)
    {
      // OK Bad guys doing something
#      $logger->post("Intrusion","Session Hijack attempt at ".
#date()."IP:$clientIP user: $userID");
      sleep(2);
      unset($security_userID);
      session_unregister("security_userID");
      die("Naughty");
      localRedirect("Location: $CIB_BASE_URL/");
      exit;
    }
  $SECURITY_CODE_DB = new CIB_DB($CIB_DB_NAME,$CIB_AUTH_USER,$CIB_AUTH_PASS);
  if (!$SECURITY_CODE_DB)
    {
#      $logger->post("Database",CIB_DB_NAME." unavailible at ".date());
      localRedirect("Location: $CIB_BASE_URL/service_outtage.html");
      exit;
    }
  $user = new User($SECURITY_CODE_DB);
  $user->private_retrieve($userID);
  if ($user->isSiteAdmin())
    {
      $SECURITY_CODE_DB = new CIB_DB($CIB_DB_NAME,$CIB_ADMIN_USER,
                $CIB_ADMIN_PASS);
      if (!$SECURITY_CODE_DB)
        {
#          $logger->post("Database",CIB_DB_NAME." unavailible at ".date());
          localRedirect("Location: $CIB_BASE_URL/service_outtage.html");
          exit;
        }
      $user = new User($SECURITY_CODE_DB);
      $user->private_retrieve($userID);
    }
 }
```

```
else
{
  $SECURITY_CODE_DB = new CIB_DB($CIB_DB_NAME,$CIB_UNAUTH_USER,
                $CIB_UNAUTH_PASS);
  if (!$SECURITY_CODE_DB)
    {
#       $logger->post("Database",CIB_DB_NAME." unavailible at ".date());
      localRedirect("Location: $CIB_BASE_URL/service_outtage.html");
      exit;
    }
  unset($user);
}
?>
```

include/general/utilities.inc

```
<?php
#####################################################################
# File: utilities.inc                                               #
# Author: Eric D. Nielsen                                           #
# Description: A collection of simple PHP wrappers around SQL       #
#              calls for the Sliding Doors application.             #
#              General page functions are also found here.          #
# Defined Functions: getDBHandle                                    #
#                    errorPage                                      #
#                    formValue                                      #
#                    getFormErrorMessage                            #
# Change Log: 6/12/01 -- created -- edn                             #
#             7/02/02 -- validation function -- edn                 #
#####################################################################

function getDB()
#####################################################################
# Function: getDBHandle                                             #
# Arguments: none                                                   #
# Returns:  The handle to the Ragu database, the connection         #
#           information is held in constants.inc.  If the           #
#           connection can not be opened, an error page is          #
#           generated.                                              #
# Requires: nothing                                                 #
# Modifies: nothing                                                 #
#####################################################################
{
    global $PHP_SELF;
    global $compdbName, $compWebUser, $compWebPass;
    $db = new CompDB($compdbName, $compWebUser, $compWebPass);
    if (!$db)
        errorPage($PHP_SELF ,"utilities::getDBHandle",
"Database inaccessible",
"Sliding Doors could not connect to the competition database.",TRUE);
    return $db;
}

function localRedirect($url)
{
    GLOBAL $HTTP_COOKIE_VARS;
    if (isset($HTTP_COOKIE_VARS["PHPSESSID"]))
        header($url);
    else
        header($url . "?" .SID);
}

function primeIDSet()
{
    GLOBAL $HTTP_SESSION_VARS, $HTTP_POST_VARS;
    $tempSess = $HTTP_SESSION_VARS["primeID"];
    $tempPost = $HTTP_POST_VARS["primeID"];
    return ($tempSess!="" || $tempPost!="");
}

function requirePrimeID()
{
    GLOBAL $baseURL;
    if (!primeIDSet())
    {
```

```
        localRedirect("Location: $baseURL/back.php");
        exit;
    }
}

function restrictNavigation($referers)
{
    GLOBAL $HTTP_REFERER, $HTTP_SESSION_VARS, $HTTP_POST_VARS;
    GLOBAL $baseURL;
    $url = $HTTP_REFERER;

    $protocol = strpos($url,'://');
    if ($protocol)
    {
        $url = substr($url,$protocol+3,strlen($url));
        $url = "http://".$url;
    }
    $hash = strpos($url,'#');
    if ($hash)
        $url = substr($url,0,$hash);
    $ques = strpos($url,'?');
    if ($ques)
        $url = substr($url,0,$ques);
    reset($referers);
    while ($aLegalPage = each($referers))
    {

        $page = $aLegalPage["value"]["PAGE"];
        $requiredVars = $aLegalPage["value"]["VARS"];
        if ($page==$url)
        {
            $numVars = count($requiredVars);
            for ($i=0;$i<$numVars;$i++)
            {
                $varName = $requiredVars[$i];
                $tempPost = $HTTP_POST_VARS["$varName"];
                $tempSess = $HTTP_SESSION_VARS["$varName"];
                if ($tempPost=="" && $tempSess=="")
                {
                    localRedirect("Location: $baseURL/back.php");
                    exit;
                }
            }
            return true;
        }
    }
    localRedirect("Location: $baseURL/back.php");
    exit;
}


function errorPage($page, $func, $title, $msg, $sendEmail)
#################################################################
# Function: errorPage                                           #
# Arguments: page -- the name of the page throwing the error,   #
#            func -- the function throwing the error,           #
#            title -- the title for the error page,             #
#            msg -- a short descriptive error message,          #
#            sendEmail -- boolean, auto send email              #
# Returns:  nothing                                             #
# Requires: nothing                                             #
```

```
# Modifies: Output buffer, program flow, issues a die command  #
#################################################################
{
#    obClear(); // need to find right command to empty the buffer if content
#               // has already been sent.
    global $onErrorMail;
    $errorText  = "<HTML>\n<HEAD>\n<TITLE>ERROR: $title </TITLE>\n</HEAD\n";
    $errorText .= "<BODY>\n<H1>$msg</H1>\n<P>\n";
    $errorText .= "This error was generated from $func ";
    $errorText .= "called from $page.\n<p>\n";
        $sendEmail=FALSE;
    if ($sendEmail)
    {
        $errorText .= "An email was automatically sent to ";
        $errorText .= "<A HREF=mailto:\\$onErrorMail>$onErrorMail</A>. ";
        $errorText .= " If you wish to be informed when the error has been ";
        $errorText .= "fixed, feel free to send you own email to this ";
        $errorText .=" address as well.";

        $mailBody = "$date\n$msg\npage:$page\nfuntion:$func\n";
        mail($onErrorMail,"OI ERROR: $title","$mailBody");
    }
    else
    {
        $errorText .= "This is considered a non-critical error and it ";
        $errorText .= "can likely be fixed by going BACK and trying again.";
        $errorText .= "\n<BR>\n";
        $errorText .= "If the problem persists, please email ";
        $errorText .= "<A HREF=mailto:\\$onErrorMail>$onErrorMail</A>.";
    }
    $errorText .="</BODY>\n</HTML>\n";
    die($errorText);
}

function formValue($var)
#################################################################
# Function: formVars                                            #
# Arguments: var -- what variable do we want the value of       #
# Returns:  The value of the variable submitted to the form,    #
#           returns the empty string if the form wasn't_______#
#___________submitted_or_the_variable_weren't in the form.      #
# Requires: nothing                                             #
# Modifies: nothing                                             #
#################################################################
{
    return $GLOBALS["formVars"]["$var"];
}

function formValidate($label,$value,$msg,$required=TRUE)
{
    global $validated;
    global $formVars;
    if ($value!="")
        $formVars["$label"]=$value;
    else if ($required)
    {
        $formVars["$label"."Msg"]=$msg;
        $validated=FALSE;
    }
}
```

```
function getFormErrorMessage($var, $cols,$options="")
#####################################################################
# Function: getFormErrorMessage                                     #
# Arguments: var -- what variable's error are we checking           #
#            cols -- how many columns to span                       #
#            options --- arguements to the <th> element             #
# Returns:  A row to insert into the table with the error           #
#           message for the value spanning the specified            #
#           columns.  Returns nothing if no error.                  #
# Requires: nothing                                                 #
# Modifies: nothing                                                 #
#####################################################################
{
    global $HTTP_SESSION_VARS;

    $varMsg = $var."Msg";
    $msg = $HTTP_SESSION_VARS["formVars"][$varMsg];
    $result="";
    if($msg!="")
        $result="<TR><TD COLSPAN=$cols $options>$msg</TD></TR>\n";
    return $result;
}

function spamList($addresses,$subject,$body,$from)
{
  global $SD_hostname;
  $addresses = array_unique($addresses);
  $numAddresses = count($addresses);
  $to = "no-reply@$SD_hostname";
  $bcc = "";
  for ($i = 0; $i < $numAddresses; $i++)
    {
      if ($i%25==24)
        {
          mail($to,$subject,$body,
               "From: $from\r\nbcc: $bcc\r\nX-mailer: SlidingDoors");
          $bcc="";
        }
      $bcc .= ($bcc==""?"":",").$addresses[$i];
    }
  if ($bcc!="")
    mail($to,$subject,$body,
         "From: $from\r\nbcc: $bcc\r\nX-mailer: SlidingDoors");
}

function sanitizeXHTML($string,$allowedTags=array())
{
  foreach($allowedTags as $tag)
    {
      // handle all allowed, paired tags
      $string = ereg_replace("<$tag(.*)>(.*)</$tag>",
                             "[$tag\\1]\\2[/$tag]",$string);
      // handle all empty tags
      $string = ereg_replace("<$tag(.*)\\>",
                             "[$tag\\1\\]",$string);
    }
  //string all disallowed paired tag, leave tagged text
  $string = ereg_replace("<.*>(.*)</.*>","\\1",$string);
  //string all disallowed empty tags AND all allowed paired without pair
  $string = ereg_replace("<.*>","",$string);
  foreach($allowedTags as $tag)
```

```
    {
        // convert paired back to html
        $string = ereg_replace("\[$tag(.*)\](.*)\[/$tag\]",
                               "<$tag\\1>\\2</$tag>",$string);
        // convert empty back to html
        $string = ereg_replace("\[$tag(.*) \\\]",
                               "<$tag\\1\\>",$string);
    }
  return $string;
}


function nameConflicts($db, $first, $last, $email, $affil)
##################################################################
# Function: nameConflicts                                        #
# Arguments: db -- handle to the competition database            #
#            first -- the person's first name to test            #
#            last -- the person's last name to test              #
#            email -- the person's email to test                 #
#            affil -- the person's affiliation to test           #
# Returns: Returns true if there is another person in the db     #
#          whose identification information matches closely      #
#          Returns false otherwise, even if db error.            #
# Requires: nothing                                              #
# Modifies: nothing                                              #
##################################################################
{
    return (count(getNameConflicts($db,
                                   $first,
                                   $last,
                                   $email,
                                   $affil)) > 0 ? TRUE : FALSE);
}

function nameCollisionTable($people, $formVars, $target)
##################################################################
# Function: nameCollisionTable                                   #
# Arguments: people -- the list of colliding IDs                 #
#            formVars -- the entered information                 #
#            target -- where to send the user's choice           #
# Returns: A table/form string that gives the name collisions    #
#          to the user for user clarification.                   #
# Requires: nothing                                              #
# Modifies: nothing                                              #
##################################################################
{
    $first = $formVars["FirstName"];
    $last = $formVars["LastName"];
    $email = $formVars["Email"];
    $affil = $formVars["Affil"];
    $newAffil = $formVars["NewAffil"];
    $affilTemp = ($affil == "" ? $newAffil : $affil);

    $result  ="<TABLE>\n";
    $result .= "\t<TR>\n";
    $result .= "\t\t<TH>Name</TH><TH>Email</TH>\n";
    $result .= "\t\t<TH>Affiliation</TH><TH>Partners</TH><TH>Select</TH>\n";
    $result .= "\t</TR>\n";
    $result .= "\t<TR>\n";
    $result .= "\t\t<TD>$first $last</TD><TD>$email</TD><TD>$affilTemp</TD>\n";
    $result .= "\t\t<TD>None entered yet</TD>\n";
```

```
        $result .= "\t\t<TD><FORM ACTION=$target METHOD=POST>\n";
        $result .= "\t\t\t<INPUT TYPE=HIDDEN NAME=CHOICE VALUE=-1>\n";
        $result .= "\t\t\t<INPUT TYPE=SUBMIT NAME=USE VALUE=\"Use Me\"></FORM></TD>\n";
        $result .= "\t</TR>\n";

        $num = count($people);
        for ($i=0;$i<$num;$i++)
        {
            $person = new Person($db,$people[$i]);
            $partners = getPartnersByID($db, $people[$i]);
            $first = $person->getFirstName();
            $last = $person->getLastName();
            $email = $person->getEmail();
            $affil = $person->getAffil();

            $result .= "\t<TR>\n";
            $result .= "\t\t<TD>$first $last</TD><TD>$email</TD><TD>$affilTemp</TD>\n";
            $result .= "\t\t<TD>";
            $numPartners = count($partners);
            if ($numPartners==0)
            {
                $result .= "None entered yet";
            }
            for ($j=0;$j<$numPartners;$j++)
            {
                $aPartner = new Person($db,$partners[$j]);
                $partnerFirst = $aPartner->getFirstName();
                $partnerLast = $aPartner->getLastName();
                $result .= "$partnerFirst $partnerLast <BR>";
            }
            $reult .= "\t\t<\TD>\n";
            $result .= "\t\t<TD><FORM ACTION=$target METHOD=POST>\n";
            $result .= "\t\t\t<INPUT TYPE=HIDDEN NAME=CHOICE VALUE=$people[$i]>\n";
            $result .= "\t\t\t<INPUT TYPE=SUBMIT NAME=USE VALUE=\"Use Me\"></FORM></TD>\n";
            $result .= "\t</TR>\n";
        }
        $result .= "</TABLE>\n";
        return $result;
}

function getEmailConflicts($db, $email)
#################################################################
# Function: getEmailConflicts                                   #
# Arguments: db -- handle to the competition database           #
#            email -- the person's email to test                #
# Returns: Returns an array containing the personID of the      #
#          matches                                              #
# Requires: nothing                                             #
# Modifies: nothing                                             #
#################################################################
{
    $people = array();

    $query = "SELECT peopleid ";
    $query .= "FROM people_contact ";
    $query .= "WHERE emailday='$email' or emaileve='$email';";
    $result = pg_exec($db,$query);
    $count = pg_numrows($result);
    for ($i=0; $i < $count; $i++)
    {
        list ($temp) = pg_fetch_array($result, $i);
```

```
        array_push($people,$temp);
    }
    return array_unique($people);
}

#function emailCollisionTable($db,$email,$target)
#################################################################
# Function: emailCollisionTable                                 #
# Arguments: db -- handle to the competition database           #
#            email -- the information colliding                 #
#            target -- where to send the user's choice          #
# Returns: A table/form string that gives the name collisions   #
#          to the user for user clarification.                  #
# Requires: nothing                                             #
# Modifies: nothing                                             #
#################################################################
#{
#    $people = getEmailConflicts($db, $email);#
#
#    $result  ="<TABLE>\n";
#    $result .= "\t<TR>\n";
#    $result .= "\t\t<TH>Name</TH><TH>Email</TH>\n";
#    $result .= "\t\t<TH>Affiliation</TH><TH>Partners</TH><TH>Select</TH>\n";
#    $result .= "\t</TR>\n";
#
#    $num = count($people);
#    for ($i=0;$i<$num;$i++)
#    {
#        $person = getPersonByID($db,$people[$i]);
#        $partners = getPartnersByID($db, $people[$i]);
#        $first = getFirstName($person);
#        $last = getLastName($person);
#        $email = getEmail($person);
#        $affil = getAffil($person);
#
#        $result .= "\t<TR>\n";
#        $result .= "\t\t<TD>$first $last</TD><TD>$email</TD><TD>$affilTemp</TD>\n";
#        $result .= "\t\t<TD>";
#        $numPartners = count($partners);
#        if ($numPartners==0)
#        {
#            $result .= "None entered yet";
#        }
#        for ($j=0;$j<$numPartners;$j++)
#        {
#            $aPartner = getPersonByID($db,$partners[$j]);
#            $partnerFirst = getFirstName($aPartner);
#            $partnerLast = getLastName($aPartner);
#            $result .= "$partnerFirst $partnerLast <BR>";
#        }
#        $reult .= "\t\t<\TD>\n";
#        $result .= "\t\t<TD><FORM ACTION=$target METHOD=POST>\n";
#        $result .= "\t\t\t<INPUT TYPE=HIDDEN NAME=CHOICE VALUE=$people[$i]>\n";
#        $result .= "\t\t\t<INPUT TYPE=SUBMIT NAME=USE VALUE=\"Use Me\"></FORM></TD>\n";
#        $result .= "\t</TR>\n";
#    }
#    $result .= "</TABLE>\n";
#    return $result;
#}

function arrayToCSL($values)
```

```
#################################################################
# Function: arrayToCSL                                          #
# Arguments: values -- the list to change to a CSL              #
# Returns: a string with the values in comma seperated list     #
# Requires: nothing                                             #
# Modifies: nothing                                             #
#################################################################
{
    $result = "";
    $num = count($values)-1;
    for ($i=0;$i<$num;$i++)
    {
#       echo "<br>$result";
        $result .= ($values[$i] . ", ");
    }
#       echo "<br>$result";
    $result .= $values[$i];
#       echo "<br>$result";
    return $result;
}

function arrayRemove($arr,$item)
#################################################################
# Function: arrayRemove                                         #
# Arguments: arr  -- the first array                            #
#            item -- the item to remove                         #
# Returns: arr without item, designed for numerically indexed   #
#          arrays                                               #
# Requires: nothing                                             #
# Modifies: nothing                                             #
# Notes: No promises about ordering of the resulting array      #
#        designed to work best with numericall indexed arrays   #
#################################################################
{
    $isArray = is_array($arr);
    $isItem = !is_array($item);
    if (!$isArray || !$isItem) return $arr;
    if (!in_array($item,$arr)) return $arr;
    $num = count($arr);
    $found=FALSE;
    for ($i=0;$i<$num;$i++)
    {
        if ($arr[$i]==$item)
        {
            $found=TRUE;
            continue;
        }
        $newArray[($found ? $i-1 : $i)] = $arr[$i];
    }
    return $newArray;
}

function arrayJoin($arr1,$arr2)
#################################################################
# Function: arrayJoin                                           #
# Arguments: arr1 -- the first array to join                    #
#            arr2 -- the second array to join                   #
# Returns: an array with all the elements elements of arr1 &    #
#          arr2, without duplicates.                            #
# Requires: nothing                                             #
# Modifies: nothing                                             #
```

```
# Notes: No promises about ordering of the resulting array      #
#        designed to work best with numericall indexed arrays  #
#################################################################
{
    $is1 = is_array($arr1);
    $is2 = is_array($arr2);
    if ($is1 && $is2)
    {
        $result = $arr1;
        reset($arr2);
        while ($cell = each($arr2))
        {
            if (!in_array($cell['value'],$result))
                array_push($result,$cell['value']);
        }
        return $result;
    }
    elseif ($is1 && !$is2)
        return $arr1;
    elseif (!$is2 && $is2)
        return $arr2;
    else
        return array();
}


function cslToArray($str, $prefix="")
#################################################################
# Function: cslToArray                                          #
# Arguments: str -- the comma seperated list to convert         #
#            prefix -- to prepend to all labels                 #
# Returns: str in array form with prefix prepended, order is    #
#          maintained between the two forms                     #
# Requires: nothing                                             #
# Modifies: nothing                                             #
# Notes: prefix is prepended to each label, seperated by "--",  #
#        if not the empty string (no seperator then)            #
#################################################################
{
   $arr = array();
    $temp = explode(",",$str);
    $num = count($temp);
    for ($i=0;$i<$num;$i++)
    {
        if ($temp[$i]=="") continue;
        $name = ($prefix != "" ? ("$prefix" . "--") : "");
        $name .= trim($temp[$i]);
        array_push($arr,$name);
    }
    return $arr;
}

function buildArray($arr,$prefix,$lbls,$values,$empty)
#################################################################
# Function: buildArray                                          #
# Arguments: arr -- the array to build onto                     #
#            prefix -- to prepend to all labels                 #
#            lbls -- the names to associate with values         #
#            values -- the values to add to the array           #
#            empty --  determines behavoir if value=null        #
# Returns: Returns an array with the label->value pairs added   #
```

```
#           Creates an new empty array if arr is not an array.  #
#           If empty=TRUE will add labels with NULL values, if  #
#           FALSE it will not add the labels to the list.       #
# Requires: nothing                                             #
# Modifies: arr (changes don't seem to propagate back though    #
# Notes: prefix is prepended to each label, seperated by "_"    #
#################################################################
{
    $labels = cslToArray($lbls, $prefix);
    if (!is_array($arr))
        $arr = array();
    $numLabels = count($labels);
    for ($i=0;$i<$numLabels; $i++)
        if ($values[$i] || $empty)
            $arr[$labels[$i]]=$values[$i];

    return $arr;
}


function printArray($arr, $str="")
#################################################################
# Function: printArray                                          #
# Arguments: arr -- the array to print                          #
#            str -- identation needed (nested arrays)           #
# Returns: A string ready to print containing the array in      #
#          list form. "Not an Array" if arr is not an array     #
# Requires: nothing                                             #
# Modifies: nothing                                             #
# Notes: This function is recursive, don't use on extremely     #
#        deeply nested arrays.                                  #
#################################################################
{
    $result = "";
    reset($arr);
    while ($cell = each($arr))
    {
        $value = $cell['value'];
        $key = $cell['key'];
        $result .= "$str $key -> ";
        if (is_array($value))
            $result .= "<BR>" . printArray($value, $str . "...");
        else
            $result .= "$value <BR>";
    }
    return $result;
}

function getPartnersByID($db, $id)
#################################################################
# Function: getPartnersByID                                     #
# Arguments: db -- handle to the competition database           #
#            id -- the the id of the person                     #
# Returns: An array of peopleID who dance with id, empty list   #
#          if db inaccessible OR no partners                    #
# Requires: nothing                                             #
# Modifies: nothing                                             #
#################################################################
{
    $partners = array();
```

```
    $query  = "SELECT leader  ";
    $query .=   "FROM couples ";
    $query .=   "WHERE follower=$id;";
    $result = pg_exec($db,$query);
    if (!$result)
    {
        $partners=array();
        return $partners;
    }
    $num = pg_numrows($result);
    for ($i=0;$i<$num;$i++)
    {
        list($temp) = pg_fetch_array($result,$i);
        array_push($partners, $temp);
    }
    $query  = "SELECT follower  ";
    $query .=   "FROM couples ";
    $query .=   "WHERE leader=$id;";
    $result = pg_exec($db,$query);
    if (!$result)
    {
        $partners=array();
        return $partners;
    }
    $num = pg_numrows($result);
    for ($i=0;$i<$num;$i++)
    {
        list($temp) = pg_fetch_array($result,$i);
        array_push($partners, $temp);
    }
    return array_unique($partners);
}


function newTimestamp($label)
{
    global $timeStamps;
    $timeStamps[$label]=microtime();
}

function printTiming($beginTime)
{
    global $timeStamps;
    $initial = explode(" ", $beginTime);
    $last = $initial;
    reset($timeStamps);
    while ($cell = each($timeStamps))
    {
        $label = $cell['key'];
        $time = explode(" ",$cell['value']);
        $partial = $time[1] - $last[1] + $time[0] - $last[0];
        $total = $time[1] - $initial[1] + $time[0] - $initial[0];
        echo "<BR> The script had run for $total seconds at $label.
($partial seconds since last timestamp.)";
        $last=$time;
    }
}

#################################################################
# Function: clean_up_session                                    #
# Arguments: none                                               #
```

```
# Returns: nothing                                                    #
# Requires: nothing                                                   #
# Modifies: the session                                               #
# Notes: This function unregisters all registered session             #
#        variables.  Called by most display pages to clean            #
#        up the namespace.  It appeared once registered a             #
#        was staying registered on subsequent pages, contrary         #
#        to the manual.   I haven't checked if the manual has         #
#        changed or there is a bug report on this topic.              #
#####################################################################
function clean_up_session()
{
    GLOBAL $HTTP_SESSION_VARS;

    $vars = array_keys($HTTP_SESSION_VARS);
    $numVars=count($vars);
    for ( $i=0;$i<$numVars;$i++)
    {
        if (strpos($vars[$i],"security")===FALSE &&
            strpos($vars[$i],"propagate")===FALSE)
        {
            session_unregister("$vars[$i]");
            unset($$vars[$i]);
        }
    }
}
#####################################################################
# Function: copyArray                                               #
# Arguments: source   the source array                              #
#            prefix   prepend to all copied keys                    #
#            prefixToRemove   remove from all source keys,          #
#                             before prepending prefix              #
# Returns: an array, dest, such that:                               #
#          dest[prefix.(key(source)[i]-prefixToRemove)]=            #
#                    source[i]                                      #
# Requires: nothing                                                 #
# Modifies: nothing                                                 #
#####################################################################
function copyArray($source, $prefix="",$prefixToRemove="")
{
    $dest=array();
    reset($source);
    while ($anItem = each($source))
    {
        $key=$anItem["key"];
        if (""!=$prefixToRemove)
            if (0===strpos($key,$prefixToRemove))
                $key = substr($key,strlen($prefixToRemove),strlen($key));
        $key = $prefix . $key;
        $value=$anItem["value"];
        $dest[$key]=$value;
    }
    return $dest;
}

function equalOrGreaterKeyLevel($a,$b,$type)
{
  switch($type)
  {
    case "Site": switch($a)
    {
```

```
      case "Admin" : return in_array($b,array("Admin","Member")); break;
      case "Member" : return in_array($b,array("Member")); break;
      default: return FALSE;
      }
    case "Comp" : switch($a)
      {
      case "Coordinator" : return in_array($b,array("Coordinator",
                                                    "Registrar",
                                                    "Comptroller",
                                                    "Staff")); break;

      case "Registrar" : return in_array($b,array("Registrar",
                                                  "Staff")); break;

      case "Comptroller" : return in_array($b,array("Comptroller",
                                                    "Staff")); break;
      case "Staff" : return in_array($b,array("Staff")); break;
      default: return FALSE;
      }
    case "Team" : switch($a)
      {
      case "Captain" : return in_array($b,array("Captain",
                                                "Registration Coordinator",
                                                "Treasurer",
                                                "Member")); break;
      case "Registration Coordinator" :
        return in_array($b,array("Registration Coordinator",
                                 "Member")); break;
      case "Treasurer" : return in_array($b,array("Treasurer",
                                                  "Member")); break;
      case "Member" : return in_array($b,array("Member")); break;
      default: return FALSE;
      }
    default:return FALSE;
    }
}

function keyString($key)
{
  switch ($key["Type"])
    {
    case "Site" : return "Site {$key["Level"]}"; break;
    case "Comp" : return "Comp {$key["Compname"]} {$key["Level"]}";break;
    case "Team" : return "Team {$key["Teamname"]} {$key["Level"]}";break;
    default:return "Bad Key";
    }
}


function keyCompare($a,$b)
{
  switch($a["Type"])
    {
    case "Site":
      if ($b["Type"]!="Site") return -1;
      else
        switch ($a["Level"])
          {
          case "Admin": if ($b["Level"]!="Admin") return -1;
          else                                    return 0; break;
```

```
          case "Staff": if ($b["Level"]=="Admin") return 1;
          else if ($b["Level"]=="Staff")          return 0;
          else                                   return -1; break;

          case "Member": if ($b["Level"]!="Member")  return 1;
          else                                       return 0; break;
          default: break;
          }
        break;
      case "Comp":
        if ($b["Type"]=="Site") return 1;
        else if ($b["Type"]=="Team") return -1;
        else
          switch ($a["Level"])
            {
            case "Coordinator": if ($b["Level"]!="Coordinator") return -1;
            else return ($a["Compname"]<$b["Compname"]); break;

            case "Registrar": if ($b["Level"]=="Coordinator")  return 1;
            else if ($b["Level"]=="Registrar")   return ($a["Compname"]<$b["Compname"]);
            else  return -1; break;

            case "Comptroller": if (in_array($b["Level"],array("Coordinator",
                                                    "Registrar")))
              return 1;
            else if ($b["Level"]="Comptroller")
              return ($a["Compname"]<$b["Compname"]);
                else return -1; break;

            case "Staff": if (in_array($b["Level"],array("Coordinator",
                                                  "Registrar",
                                                  "Comptroller")))
              return 1;
            else return ($a["Compname"]<$b["Compname"]);break;
            default: break;
            }
        break;
      case "Team":
        if ($b["Type"]!="Team") return 1;
        else
          switch ($a["Level"])
            {
            case "Captain": if ($b["Level"]!="Captain") return -1;
            else return ($a["Teamname"]<$b["Teamname"]); break;

            case "Registration Coordinator": if ($b["Level"]=="Captain")  return 1;
            else if ($b["Level"]=="Registration Coordinator")
return ($a["Teamname"]<$b["Teamname"]);
            else  return -1; break;

            case "Treasurer": if (in_array($b["Level"],array("Captan",
                                                  "Registration Coordinator")))
              return 1;
            else if ($b["Level"]="Treasurer")
              return ($a["Teamname"]<$b["Teamname"]);
                else return -1; break;

            case "Member": if (in_array($b["Level"],array("Captain",
                                                  "Registration Coordinator",
                                                  "Treasurer")))
              return 1;
```

```
            else return ($a["Teamname"]<$b["Teamname"]);break;
            default: break;
            }
        break;
    default: break;
    }
  return 0;
}

?>
```

include/general/tools/CompInaBox_header.inc

```
<?php
/*************************************************************
 * This file is part of CompInaBox.                          *
 * Copyright 2001-2002. Eric D. Nielsen, All rights reserverd *
 * CompInaBox is availible for license under the GPL, see    *
 * the COPYING file in the root directory of the install for *
 * the full terms of the GPL.                                *
 *                                                           *
 * This file sets up three objects for use by the rest of the *
 * applications require authentication services. The three   *
 * objects are logger (for logging to files or email),       *
 * user (user permissions)                                   *
 * db (database connection, with appropriate permissions)    *
 *************************************************************/
include_once("include_others.inc");
include_once("$CIB_CLASS_PATH/database/DB.inc");
include_once("$CIB_CLASS_PATH/database/CIB_DB.inc");
include_once("$CIB_CLASS_PATH/database/QueryResult.inc");
include_once("$CIB_CLASS_PATH/access-control/User.inc");
include_once("$CIB_CLASS_PATH/access-control/Gate.inc");
include_once("$CIB_CLASS_PATH/access-control/Guard.inc");
include_once("$CIB_CLASS_PATH/html-formatting/UIBase.inc");
include_once("$CIB_CLASS_PATH/html-formatting/CIB_Display.inc");
include_once("$CIB_INCLUDE_PATH/utilities.inc");
session_start();
include_once("$CIB_INCLUDE_PATH/session_user_setup.inc");
$db = $SECURITY_CODE_DB;
?>
```

include/general/tools/SlidingDoors_header.inc

```
<?php
    if (count($HTTP_GET_VARS)>1)
die("Malicious User attempting to hand code URL string.
Email has been sent to the site administrators.");
$beginTime=microtime();
$timeStamps = array();
$onErrorMail = "nielsene@localhost"; //:TODO: move to conf file

#turn outputBuffering on
include "database/QueryResult.inc";
include "database/DB.inc";
include "database/CompDB.inc";
include "database/CIB_DB.inc";
include "general/utilities.inc";

include "storeable-objects/StoredObject.inc";
include "storeable-objects/Person.inc";
include "storeable-objects/Competitor.inc";
include "storeable-objects/Couple.inc";
include "storeable-objects/Organization.inc";
include "storeable-objects/Address.inc";
include "storeable-objects/RegOrg.inc";

include "html-formatting/UIBase.inc";
include "html-formatting/HTMLDisplay.inc";

include "contra-check/ContraCheckDetail.inc";
include "contra-check/DoubleRegistrationDetail.inc";
include "contra-check/DanceNumLevelViolationDetail.inc";
include "contra-check/SingleLevelViolationDetail.inc";
include "contra-check/ContraCheckResult.inc";
include "contra-check/ContraCheck.inc";

include "access-control/User.inc";
include "access-control/Gate.inc";
include "access-control/Guard.inc";

newTimestamp("Finished Including Files");
$db=getDB();
newTimestamp("Established DB connection");



?>
```

B.3 Class Defination Files

Paths start from `include/classes`.

LaTeX.inc

```
<?php

class LaTeX
{

  function escapeText($text)
    {
      $text = str_replace('\\','',$text);
    ------$text = str_replace('~','',$text);
    ------$text = str_replace('^','',$text);
    ------$text = str_replace('$','\\$',$text);
    ------$text = str_replace('&','\\&',$text);
    ------$text = str_replace('%','\\%',$text);
    ------$text = str_replace('#','\\#',$text);
    ------$text = str_replace('_','\\_',$text);
    ------$text = str_replace('{','\\{',$text);
    ------$text = str_replace('}','\\}',$text);
    ------return $text;
    ----}

  --function centerText($text)
    ----{
    ------if (is_array($text))
    --------{
    ----------$numRows = count($text);
    ----------$latex = "\\begin{center}\n";
    ----------for ($i=0; $i<$numRows; $i++)
    ------------$latex .= "$text\n";
    ----------$latex .= "\\end{center}\n";
    --------}
    ------else
    --------$latex = "\\centerline\{$text}\n";
    ------return $latex;
    ----}

  --function spanRow($text,$numRows,$format="c")
    ----{
    ------return "\\multicolumn\{$numRows}\{$format}\{$text}\\\\\n";
    ----}

  --function formatText($text, $type)
    ----{
    ------if (is_array($text)) die("Text must be scalar in formatText");
    ------switch ($type)
    --------{
    --------case "Bold":
    --------case "BF":
    --------case "bf": $text = "\\textbf\{$text}"; break;
    --------case "Teletype":
    --------case "TT":
    --------case "tt": $text = "\\texttt\{$text}"; break;
    --------case "Italic":
```

```
        case "IT":
        case "it": $text = "\\textit\{$text}"; break;
        case "SmallCaps":
        case "SC":
        case "sc": $text = "\\textsc\{$text}"; break;
        case "Large": $text = "\\Large\{$text}"; break;
        case "small": $text = "\\small\{$text}"; break;
        default : break;
        }
      return $text;
    }

  function beginDocument($title,$author,$size=10)
    {
      $size = ceil($size);
      if ($size < 10) $size = 10;
      if ($size > 12) $size = 12;
      $latex  = "\\documentclass[{$size}pt]{article}\n";
      $latex .= "\\usepackage{fullpage,graphics,longtable}\n";
      $latex .= "\\usepackage{array,color,colortbl,nopageno}\n";
      $latex .= "\\definecolor{light}{gray}{.8}\n";
      $latex .= "\\title\{$title}\n";
      $latex .= "\\author\{$author}\n";
      $latex .= "\\begin{document}\n";
      $displayTitle =LaTeX::formatText(LaTeX::formatText($title,"Large"),
                                       "SmallCaps");
      $latex .= LaTeX::centerText($displayTitle);
      return $latex;
    }


  function produceLongtable($cells,$format="",$colored=TRUE,$border=FALSE,
                            $firstHead="",$otherHead="",
                            $otherFoot="",$lastFoot="")
    {
      $latex = "\\begin{longtable}\{$format}\n";
      if ($border) $latex .= "\\hline\n";
      if ("" != $firstHead) $latex .= "$firstHead\n\\endfirsthead\n";
      if ("" != $otherHead) $latex .= "$otherHead\n\\endhead\n";
      if ("" != $otherFoot) $latex .= "$otherFoot\n\\endfoot\n";
      if ("" != $lastFoot)  $latex .= "$lastFoot\n\\endlastfoot\n";
      $numrows = count ($cells);
      $rowKeys = array_keys($cells[0]);
      for ($i = 0; $i < $numrows; $i++)
        {
          if ($i!=0)
            $latex .= "\\\\\n";
          $oneRowOfData = $cells[$i];
          $oneLineOfLaTeX = "";
          if ($colored && $i % 2 == 0)
            $oneLineOfLaTeX .= "\\rowcolor{light}";
          $numColumns = count($oneRowOfData);
          for ($j = 0; $j < $numColumns; $j++)
            {
              if ($j!=0) $oneLineOfLaTeX .= " & ";
              $oneLineOfLaTeX .= $oneRowOfData[$rowKeys[$j]];
            }
          $latex .= $oneLineOfLaTeX;
        }
      if ($border) $latex .= "\\\\\n\\hline\n";
      else         $latex .= "\n";
```

```
        $latex .= "\\end{longtable}\n";
        return $latex;
      }

    function endDocument($includeColophon=FALSE)
      {
        if ($includeColophon)
          {
            $latex = LaTeX::formatText("Colophon:","Bold");
            $latex .= " This document was produced using ";
            $latex .= LaTeX::formatText("latexmk","Teletype");
            $latex .= " The source \\LaTeX\\hspace{1ex} was computer generated
by SlidingDoors, the on-line registration component of the CompInaBox project.
For more information on CompInaBox visit http://compinabox.sourceforge.net/.";
          }
        $latex .= "\\end{document}\n";
        return $latex;
      }
}
?>
```

access-control/Gate.inc

```
<?php
/*************************************************************
 * This file is part of the Core module of CompInaBox        *
 * Copyright 2002. Eric D. Nielsen, All rights reserverd     *
 * CompInaBox is availible for license under the GPL, see    *
 * the COPYING file in the root directory of the install for *
 * the full terms of the GPL.                                *
 *                                                           *
 * File: Gate.inc                                            *
 * Author: Eric D. Nielsen                                   *
 * Description: Access control class, stores information about *
 *              who is allowed to view a given page          *
 * Attributes:  isOpen -- a closed gate requires Authentication *
 *              isLocked -- locked gates test authorization  *
 *              locks -- access levels that may pass the gate *
 * Constructor: Gate -- created an open, unlocked gate       *
 * Methods:     isOpen -- may unauthenticated users pass     *
 *              isLocked -- must users present a key to pass *
 *              raise/lowerPortcullis -- toggle open status  *
 *              lockGate -- lock the gate, provide list of keys *
 *                          that may unlock it, always includes *
 *                          the site admin key               *
 *              unlockGate -- attempts to unlock the gate    *
 *                            gate is open/unlocked on success *
 * Private:     test{Site/Comp/Team}Key -- trys to unlock the *
 *                                         gate using a key  *
 * Notes: All member functions require the class to be        *
 *        instantiated, no static functions are provided, the *
 *        instantiation requirement is not explicitly mentioned *
 *        in the function descriptions.                      *
 *        Locks are currently identical to keys, in the future *
 *        it is expected that Locks will be a subset of Keys, *
 *        as keys evolve into an accountable token that may be *
 *        shared between servers.                            *
 *************************************************************/

class Gate
{
  var $_isOpen;
  var $_isLocked;
  var $_locks;

  /*************************************************************
   * Function: Gate                                            *
   * Description: The Gate constructor creates an open and     *
   *              unlocked gate that does not restrict access. *
   * Modifies: this                                            *
   *************************************************************/
  function Gate()
    {
      $this->_isOpen = TRUE;
      $this->_isLocked= FALSE;
      $this->_locks = "";
    }

  /*************************************************************
   * Function: isOpen                                          *
   * Description: accessor function used to inspect the gate   *
   * Returns:   TRUE if unauthenticated users may pass         *
```

```
 ************************************************************/
function isOpen()
  {
    return $this->_isOpen;
  }

/************************************************************
 * Function: isLocked                                       *
 * Description: accessor function used to inspect the gate  *
 * Returns:   TRUE if this gate requires specified access   *
 *            levels to pass                                *
 ************************************************************/
function isLocked()
  {
    return $this->_isLocked;
  }

/************************************************************
 * Function: lowerPortcullis                                *
 * Description: mutator used to restrict access to          *
 *              authenticated users                         *
 * Modifies: this                                           *
 ************************************************************/
function lowerPortcullis()
  {
    $this->_isOpen = FALSE;
  }

/************************************************************
 * Function: raisePortcullis                                *
 * Description: mutator used to open the gate               *
 * Modifies: this                                           *
 * Notes: as a gate begins open, this function is used to   *
 *        signal that the required authoization has been met*
 ************************************************************/
function raisePortcullis()
  {
    if (!$this->isLocked())
      $this->_isOpen = TRUE;
  }

/************************************************************
 * Function: lockGate                                       *
 * Description: mutator used to lock the gate and set the   *
 *              keys that may unlock the gate               *
 * Arguments: keys -- a listing of allowed access levels,   *
 *                    defaults to Site Admin                *
 * Modifies: this                                           *
 * Notes: if no key of type "Site" is given, the Site Admin *
 *        key is automatically added.                       *
 ************************************************************/
function lockGate($keys=array(array("Type"=>"Site","Level"=>"Admin")))
  {
    if ($this->isOpen())
      {
        $this->lowerPortcullis();
      }
    $numKeys = count($keys);
    if ($numKeys > 0)
      {
        $this->_isLocked=TRUE;
```

```
        $siteKey = FALSE;
        $this->_locks=array();
        for ($i=0;$i<$numKeys;$i++)
          {
            $aKey = $keys[$i];
            $keyType = $aKey["Type"];
            switch ($keyType)
              {
              case "Site" :
                $lock=array("Type"=>"$keyType","Level"=>$aKey["Level"]);
                $siteKey = TRUE;
                break;
              case "Comp" :
                $lock=array("Type"=>"$keyType","Level"=>$aKey["Level"],
                            "CompName"=>$aKey["CompName"]);
                break;
              case "Team":
                $lock=array("Type"=>"$keyType","Level"=>$aKey["Level"],
                            "TeamID"=>$aKey["TeamID"]);
                break;
              default:
                break;
              }
            $this->_locks[$i]=$lock;
          }
        if (!$siteKey)
          {
            // ensure that the admin's skeleton key always works
            $lock = array("Type"=>"Site","Level"=>"Admin");
            $this->_locks[]=$lock;
          }
        return true;
      }
    else
      {
        return false; // gate not locked, no key provided
      }
  }

/*************************************************************
 * Function: unlockGate                                      *
 * Description: Attempts to unlock the gate using the keys   *
 *              provided, should a match occur the gate is   *
 *              unlocked and the portcullis raised.          *
 * Arguments: presentedKeys -- the access levels to test     *
 * Modifies: this                                            *
 * Notes: As the gate is raised in sucess the caller should  *
 *        call isOpen to test for success.  Leaving the gate *
 *        open after a successful authorization is not an    *
 *        issue as each gate exists for one attempt only.    *
 *************************************************************/
function unlockGate($presentedKeys)
  {
    $numKeys = count($presentedKeys);
    $trialKeyIndex = 0;
    while ($this->isLocked() && $trialKeyIndex<$numKeys)
      {
        $aKey = $presentedKeys[$trialKeyIndex];
        $keyType = $aKey["Type"];
        // make sure this is a valid key type
        if (in_array($keyType,array("Site","Comp","Team")))
```

```
          {
            // build the function name to test the key
            $testFunc = "_test".$keyType."Key";
            if ($this->$testFunc($aKey))
              {
                // on match we unlock the gate, causing the loop to end
                $this->_isLocked=FALSE;
              }
          }
        $trialKeyIndex++;
      }
    // on success, raise the portcullis
    if (!$this->isLocked())
      $this->raisePortcullis();
  }


/***************** PRIVATE MEMBER FUNCTIONS *****************/

/*************************************************************
 * Function: testSiteKey                                     *
 * Description: Tests a site key in each of the locks        *
 * Arguments: trialKey -- the key containing the user's      *
 *                        access level                       *
 * Notes: The family of test{Site/Comp/Team}Key are all very *
 *        similar but were broken up to avoid requiring      *
 *        either messy logic or database queries.            *
 *        This function holds the hierarchial data for which *
 *        site levels supersede others.                      *
 *        Current hierarchy is Staff < Admin                 *
 *************************************************************/
function _testSiteKey($trialKey)
  {
    $locks = $this->_locks;
    $numLocks = count($locks);
    for ($i=0;$i<$numLocks;$i++)
      {
        if ($locks[$i]["Type"]!="Site") continue;
        switch ($locks[$i]["Level"])
          {
          case "Admin":
            if ($trialKey["Level"]=="Admin") return TRUE;
            break;
          case "Staff":
            if (in_array($trialKey["Level"],array("Admin","Staff")))
              return TRUE;
            break;
          default: break;
          }
      }
    return FALSE;
  }

/*************************************************************
 * Function: testCompKey                                     *
 * Description: Tests a comp key in each of the locks        *
 * Arguments: trialKey -- the key containing the user's      *
 *                        access level                       *
 * Notes: This function holds the hierarchial data for which *
 *        comp levels supersede others.                      *
 *        Current hierachy is                                *
```

```
 *         Staff < [Comptroller | Registrar] < Coordinator      *
 ************************************************************/
function _testCompKey($trialKey)
  {
    $locks = $this->_locks;
    $numLocks = count($locks);

    for ($i=0;$i<$numLocks;$i++)
      {
        if ($locks[$i]["Type"]!="Comp") continue;
        if ($locks[$i]["CompName"]!=$trialKey["Compname"]) continue;
        switch ($locks[$i]["Level"])
          {
          case "Coordinator":
            if ($trialKey["Level"]=="Coordinator") return TRUE;
            break;
          case "Registrar":
            if (in_array($trialKey["Level"],array("Coordinator",
                                                  "Registrar")))
              return TRUE;
            break;
          case "Comptroller":
            if (in_array($trialKey["Level"],array("Coordinator",
                                                  "Comptroller")))
              return TRUE;
            break;
          case "Staff":
            if (in_array($trialKey["Level"],array("Coordinator",
                                                  "Registrar",
                                                  "Comptroller",
                                                  "Staff")))
              return TRUE;
            break;
          default: break;
          }
      }
    return FALSE;
  }

/************************************************************
 * Function: testTeamKey                                    *
 * Description: Tests a team key in each of the locks       *
 * Arguments: trialKey -- the key containing the user's     *
 *                        access level                      *
 * Notes: This function holds the hierarchial data for which *
 *         team levels supersede others.                    *
 *         Current hierarchy is                             *
 *         Member < [Treasurer | Registration Coordinator]  *
 *                < Captain                                 *
 ************************************************************/
function _testTeamKey($trialKey)
  {
    $locks = $this->_locks;
    $numLocks = count($locks);
    for ($i=0;$i<$numLocks;$i++)
      {
        if ($locks[$i]["Type"]!="Team") continue;
        if ($locks[$i]["TeamID"]!=$trialKey["TeamID"]) continue;
        switch ($locks[$i]["Level"])
          {
          case "Captain":
```

```
            if ($trialKey["Level"]=="Captain") return TRUE;
            break;
          case "Registration Coordinator":
            if (in_array($trialKey["Level"],
                         array("Captain",
                               "Registration Coordinator")))
              return TRUE;
            break;
          case "Treasurer":
            if (in_array($trialKey["Level"],array("Captain",
                                                  "Treasurer")))
              return TRUE;
            break;
          case "Member":
            if (in_array($trialKey["Level"],array("Captain",
                                                  "Treasurer",
                                                  "Registration Coordinator",
                                                  "Member")))
              return TRUE;
            break;
          default: break;
          }
      }
    return FALSE;
  }

}
?>
```

access-control/Guard.inc

```
<?php
/*****************************************************************
 * This file is part of the Core module of CompInaBox            *
 * Copyright 2002. Eric D. Nielsen, All rights reserverd         *
 * CompInaBox is availible for license under the GPL, see        *
 * the COPYING file in the root directory of the install for     *
 * the full terms of the GPL.                                    *
 *                                                               *
 * File: Guard.inc                                               *
 * Author: Eric D. Nielsen                                       *
 * Description: Access control class, interacts with both User   *
 *              and Gate to control access, at present Guard is  *
 *              a simple class, but if distributed servers ever  *
 *              occur it will be crucial to negotiating passage  *
 * Notes: This class, while currently static, is expected to     *
 *        require instantiation in the future.  For upward       *
 *        compatibility, always instantiate a Guard.             *
 *****************************************************************/

class Guard
{

  /*************************************************************
   * Function: Guard                                           *
   * Description: The Guard constructor does nothing           *
   * Modifies:  this                                           *
   *************************************************************/
  function Guard()
    {
    }

  /*************************************************************
   * Function: recognizesUser                                  *
   * Description: is this a local, authenticated user          *
   * Arguements: the current userid                            *
   * Returns:   TRUE if the userid is numeric and not zero     *
   *************************************************************/
  function recognizesUser($userid)
    {
      return isset($userid) && is_numeric($userid) && $userid>0;
    }

  /*************************************************************
   * Function: examineCredentials                              *
   * Description: used to accept a user from another server,   *
   *              detect and delete forged keys                *
   * Arguements: the current user's credentials                *
   * Returns:   TRUE if this user has valid credintials        *
   *************************************************************/
  function examineCredentials($credentials)
    {
      return FALSE; // not implemented
    }

  /*************************************************************
   * Function: unlockGateWithKeys                              *
   * Description: Tries the user's keys in the gate            *
   * Arguments: gate -- the gate to attempt to unlock          *
   *            keys -- the user's keys                        *
```

```
    * Notes: Eventually this may include some intrustion          *
    *          detection code and the ability to exile/execute    *
    *          users caught acting in bad faith                   *
    *          If this function returns, the user may pass        *
    ***************************************************************/
  function unlockGateWithKeys($gate,$keys,$failurePage)
    {
      $gate->unlockGate($keys);
      if (!$gate->isOpen())
        {
          header("Location: $failurePage");
          exit;
        }
      return;
    }

  /***************************************************************
    * Function: referToScribe                                     *
    * Description: redirects the user to a login/create account *
    *              screen, returns to this page after login       *
    ***************************************************************/
  function referToScribe()
    {
      GLOBAL $HTTP_SESSION_VARS, $PHP_SELF,$CIB_SECURE_URL;
      GLOBAL $Retries, $ReturnTo;
      $Retries=0;
      $ReturnTo=$PHP_SELF;
      session_register("Retries");
      session_register("ReturnTo");
      header("Location: $CIB_SECURE_URL/accounts/login.php");
      exit;
      return;
    }
}
?>
```

access-control/User.inc

```
<?php
#################################################################
# This file is part of the Core module of CompInaBox            #
# Copyright 2002. Eric D. Nielsen, All rights reserverd         #
# CompInaBox is availible for license under the GPL, see        #
# the COPYING file in the root directory of the install for     #
# the full terms of the GPL.                                    #
#                                                               #
# File: User.inc                                                #
# Author: Eric D. Nielsen                                       #
# Description: Authenticated user class                         #
# Attributes:                                                   #
# Constructor:                                                  #
# Methods:                                                      #
#################################################################

class User
{
  var $db;
  var $username;
  var $userid;
  var $permissions;
  var $isSuperUser;

  function User($db)
    {
      $this->db=$db;
      $this->username = "";
      $this->userid="";
      $this->permissions=array();
    }

  function isSiteAdmin()
    {
      return $this->isSuperUser;
    }

  function getKeys()
    {
      return $this->permissions;

    }




  function login($username,$password)
    {
      GLOBAL $HTTP_SERVER_VARS;
      $query = "SELECT userid, hashed FROM users NATURAL JOIN user_status
WHERE username='$username' AND name='ACTIVE';";
      $result = $this->db->query($query);
      if ($result->numrows()!=1)
        return FALSE;
      list($id,$ref_pass)=$result->getRowAt(0);
      $test_pass = crypt($password,$ref_pass);
      if ($test_pass==$ref_pass)
        {
          $this->private_retrieve($id);
```

```
        return TRUE;
      }
    else
      return FALSE;
  }

function retrieve($userid)
  {
    GLOBAL $HTTP_SESSION_VARS,$HTTP_REFERER;
    $ref_hash = $HTTP_SESSION_VARS["security_hash"];
    $test_hash = md5($userid . session_id() . $HTTP_REFERER . $ref_hash);
    if ($ref_hash==$test_hash)
      {
        private_retrieve($userid);
        return TRUE;
      }
    else
      return FALSE;
  }

function getID()
  {
    return $this->userid;
  }
function getUsername()
  {
    return $this->username;
  }

function isLoggedIn()
  {
    return $this->username!="";
  }

function mayAuthorizeKey($key)
  {
    if ($this->isSiteAdmin()) return TRUE;
    $userKeys = $this->getKeys();
    switch($key["Type"])
      {
      case "Site" : return FALSE;break;
      case "Comp" :
        foreach ($userKey as $testKey)
          {
            if ($testKey["Type"]=="Comp" &&
                $testKey["Compname"]==$key["Compname"] &&
                equalOrGreaterKeyLevel($key["Level"],
                                       $testKey["Level"],"Comp"))
              return TRUE;
          }
        return FALSE;
        break;
      case "Team" :
        foreach ($userKey as $testKey)
          {
            if ($testKey["Type"]=="Team" &&
                $testKey["Teamname"]==$key["Teamname"] &&
                equalOrGreaterKeyLevel($testkey["Level"],
                                       $key["Level"],"Team"))
              return TRUE;
          }
```

```
        return FALSE;
        break;
      default: return FALSE;
      }
  }

function getComps()
  {
    $comps = array();
    $id = $this->getID();
    $isSiteAdmin = $this->isSiteAdmin();
    $compList = $this->db->getComps();
    foreach ($compList as $aComp)
      {
        $name = $aComp["Label"];
        $compunix = $aComp["Link"];
        $tasks = array();
        $query = "SELECT * FROM comps, comp_status WHERE compname='$name' ";
        $query .= "AND comp_status.name='OPEN';";
        $result = $this->db->query($query);
        if ($result->numrows())
          $tasks[]="Register";
        $query = "SELECT * FROM user_comp_roles NATURAL JOIN comps ";
        $query .= "WHERE userid=$id AND compname='$name';";
        $result = $this->db->query($query);
        if ($result->numrows() || $isSiteAdmin)
          {
            $tasks[] = "Reg Admin";
            $tasks[] = "Setup";
          }
        $query = "SELECT * FROM user_team_roles NATURAL JOIN team_roles ";
        $query .= "WHERE userid=$id AND rolename <> 'Affiliation_Member';";
        $result = $this->db->query($query);
        if ($result->numrows() || $isSiteAdmin)
          {
            $tasks[] = "Affil";
          }
        $comps[] = array("CompName"=>$name,
                         "CompUnixname"=>$compunix,
                         "Tasks"=>$tasks);
      }
    return $comps;
  }



function private_retrieve($userid)
  {
    $this->userid=$userid;
    $query = "SELECT username,siteadmin FROM users WHERE userid=$userid;";
    $result = $this->db->query($query);
    if ($result->numrows())
      {
        list($username,$siteadmin) = $result->getRowAt(0);
        $this->username=$username;
        $this->permissions=array();
        if ($siteadmin=='t')
          {
            $this->isSuperUser=TRUE;
            $this->permissions[]=array("Type"=>"Site",
```

```
                                        "Level"=>"Admin");
            }
          $query = "SELECT compunix,rolename FROM users ";
          $query .= "NATURAL JOIN comp_roles ";
          $query .= "NATURAL JOIN user_comp_roles ";
          $query .= "JOIN comps USING (compid) ";
          $query .= "WHERE username='{$this->username}' ";
          $query .= "ORDER BY compunix, rolename;";
          $result = $this->db->query($query);
          $numPermissions = $result->numrows();
          for ($i=0;$i< $numPermissions;$i++)
            {
              list($comp,$role) = $result->getRowAt($i);
              $roleSpecificIndex = strpos($role,'_');
              $role = substr($role,$roleSpecificIndex+1);
              $this->permissions[]=array("Type"=>"Comp",
                                         "Compname"=>"$comp",
                                         "Level"=>"$role");
            }
          $query = "SELECT teamid,rolename FROM users ";
          $query .= "NATURAL JOIN teams NATURAL JOIN team_roles ";
          $query .= "WHERE username='{$this->username}' ";
          $query .= "ORDER BY teamid, rolename;";
          $result = $this->db->query($query);
          $numPermissions = $result->numrows();
          for ($i=0;$i< $numPermissions;$i++)
            {
              list($team,$role) = $result->getRowAt($i);
              $roleSpecificIndex = strpos($role,'_');
              $role = substr($role,$roleSpecificIndex+1);
              $this->permissions[]=array("Type"=>"Team",
                                         "TeamID"=>"$team",
                                         "Level"=>"$role");
            }
          return TRUE;
        }
      else
        return FALSE;
    }

  // OBE: Gate/Guard combo handles this now
#  function mayView($page,$roleRestrict="")
#    {
#      // if the page asking for permission isn't a CIB page return false
#      if (FALSE===strpos(COMPINABOX_SECURE_URL,$page))
#       return FALSE;
#      // superuser doesn't need to use permission table
#      if ($this->isSuperUser)
#       return TRUE;##
#
#      // strip the common base url
#      $baseLen = strlen(COMPINABOX_SECURE_URL);
#      $dispatchString = substr($page,$baseLen);#
#
#      // everyone may view the CompInaBox index page
#      $lastSlash = strrpos($dispatchString,'/');
#      if ($lastSlash==0)
#       return TRUE;
#
#      $requestPage = substr($dispatchString,$lastSlash);
#      $path = substr($dispatchString,0,strlen($dispatchString)-
```

```
#                                       strlen($requestPage));#
#
#
#       $directories = explode("/",$path);
#       // directories[0]=""
#       // directories[1]=toolname, if present
#       // directories[2]=compname, if present#
#
#       // All top level pages are public
#       if (count($directories) < 3)
#        return TRUE;#
#
#       if (isset($this->permissions[$directories[2]]))
#        {
#          // if the user has this permission, and the calling page hasn't
#          // imposed additional restrictions, the user may view the page
#          if ($roleRestrict=="")
#              return TRUE;
#          else
#            {
#              $perms = $this->permissions[$directories[2]];
#              if (count(array_intersect($perms,$roleRestrict)))
#                  return TRUE;
#              else
#                  return FALSE;
#            }
#        }
#       else
#        return FALSE;
#    }

  function mail_username($email)
    {
      $query = "SELECT username FROM users WHERE email='$email';";
      $result = $this->db->query($query);
      if ($result->numrows()==1)
        {
          list($username)=$result->getRowAt(0);
          mail($email,"[CompInaBox] Username reminder",
               "Someone, hopefully you, requested the username associated with
this email address. The username is $username . If you did not request this
mailing please contact ".COMPINABOX_ADMIN_EMAIL." immediately.",
               "From: noreply@".COMPINABOX_DOMAIN."\r\n");
          return TRUE;
        }
      else
          return FALSE;
    }

  //:NOTE: siteadmin may NOT request their password in this manner
  function reset_and_mail_password($username)
    {
      $query = "SELECT email FROM users where username='$username' AND
siteadmin=FALSE;";
      $result = $this->db->query($query);
      if ($result->numrows()==1)
        {
          list($email)=$result->getRowAt(0);
          $password="";
#:TODO: SEED the random number generate somewhere....
          for ($i=0;$i<8;$i++)
```

```
            $password.=$this->encode(random(70));
          mail($email,"[CompInaBox] Password reset",
               "Somebody, hopefully you, requested that the password associated
with your accoun be reset. The new password is $password . Please log in
immediately and change it as this message was sent in the clear. If you did
not request this please contact ".COMPINABOX_ADMIN_EMAIL." immediately.",
               "From: noreply@".COMPINABOX_DOMAIN."\r\n");
          return $this->private_change_password($password,$username);
        }
      else
        {
          return FALSE;
        }
    }

  function private_change_password($password,$username="")
    {
      if ($username=="" && $this->username=="") return FALSE;
      if ($username=="") $username=$this->username;
      // generate an MD5-style salt
      $salt = '$1$'.substr(md5(microtime().getmypid()),0,12);
      $hashed = crypt($password,$salt);
      $query = "UPDATE users SET hashed='$hashed' WHERE username='$username';";
      $this->db->query($query);
      return TRUE;
    }

  function createPendingAccount($username,$email,$password)
    {
      GLOBAL $CIB_SERVER_HOST_NAME, $CIB_ADMIN_EMAIL, $CIB_SECURE_URL;
      GLOBAL $CIB_DOMAIN, $CIB_HOSTNAME;
      $query = "SELECT statusid FROM user_status WHERE name='PENDING';";
      $result = $this->db->query($query);
      if (!$result->numrows()) die ("Could not find the PENDING status.");
      list($status_id)=$result->getRowAt(0);
      // generate an MD5-style salt
      $salt = '$1$'.substr(md5(microtime().getmypid()),0,12);
      $hashed = crypt($password,$salt);
      $query = "INSERT INTO users (statusid,username,hashed,email,siteadmin) ";
      $query.= " VALUES ($status_id,'$username','$hashed','$email',FALSE);";
      $result = $this->db->query($query);
      $lastOID = $this->db->getLastOID();
      $query = "SELECT userid FROM users WHERE oid=$lastOID;";
      $result = $this->db->query($query);
      list($userID) = $result->getRowAt(0);

      $confirmation_hash = MD5($userID . $username);
      $query = "INSERT INTO user_hashes (userid, hash) VALUES ";
      $query.= "($userID,'$confirmation_hash')";
      $result = $this->db->query($query);
      $mail_to = $email;
      $title = "[CompInaBox-$CIB_SERVER_HOST_NAME] New Account";
      /* The following block is a HEREDOC, make sure you understand how
         they work before you edit it, don't event add spaces/tabes */
      $body =<<<END_EMAIL
This Email is sent to you to confirm your new account on the
$CIB_SERVER_HOST_NAME CompInaBox Server. To complete your registration
simply follow the following link (or cut and paste it into a browser).
$CIB_SECURE_URL/accounts/confirm_account.php?hash=$confirmation_hash

If you did not request this account, you may safely ignore this
```

```
email or you may contact $CIB_ADMIN_EMAIL to report this.

-- The $CIB_SERVER_HOST_NAME CompInaBox staff
END_EMAIL;

        $headers='From: noreply@'.$CIB_HOSTNAME.$CIB_DOMAIN."\r\n";
        mail($mail_to,$title,$body,$headers);
    }

  function activate_user($userid)
    {
      GLOBAL $CIB_DB_NAME, $CIB_AUTH_USER,$CIB_AUTH_PASS;
      GLOBAL $CIB_ADMIN_USER,$CIB_ADMIN_PASS;
      if (""==$userid) return;
      $query = "SELECT statusid FROM user_status WHERE name='PENDING';";
      $result = $this->db->query($query);
      if (!$result->numrows()) die ("Could not find the PENDING status.");
      list($pending_id)=$result->getRowAt(0);
      $query = "SELECT statusid FROM user_status WHERE name='ACTIVE';";
      $result = $this->db->query($query);
      if (!$result->numrows()) die ("Could not find the ACTIVE status.");
      list($active_id)=$result->getRowAt(0);
      // upgrade database connection to an authenticated user
      $this->db = new CIB_DB($CIB_DB_NAME,$CIB_AUTH_USER,$CIB_AUTH_PASS);

      $query = "UPDATE users SET statusid=$active_id WHERE userid=$userid;";
      $this->db->query($query);
      $query= "DELETE FROM user_hashes WHERE userid=$userid;";
      $tempDB = new CIB_DB($CIB_DB_NAME,$CIB_ADMIN_USER,$CIB_ADMIN_PASS);
      $tempDB->query($query);
      return;
    }

}
?>
```

contra-check/ContraCheck.inc

```
<?php
#################################################################
# This file is part of SlidingDoors                             #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    #
# SlidingDoors is availible for license under the GPL, see      #
# the COPYING file in the root directory of the install for     #
# the full terms of the GPL.                                    #
#                                                               #
# File: ContraCheck.inc                                         #
# Author: Eric D. Nielsen                                       #
# Description: Eligibility/Rule Checker                         #
# Attributes: db -- the local comp db                           #
#             url -- the location of a regional/national info   #
#                    store holding the results of past comps    #
# Constructor: ContraCheck                                      #
# Methods: get/set.URL/DB                                       #
# Change Log: 2/19/02 -- created -- edn                         #
#################################################################
class ContraCheck
{
    var $db;
    var $url;

    ###########################################################
    # Function: ContraCheck                                   #
    # Arguments: none                                         #
    # Returns: nothing                                        #
    # Modifies: this                                          #
    # Requires: nothing                                       #
    ###########################################################
    function ContraCheck($db=0,$url="")
    {
        $this->db=$db;
        $this->url="";
    }

    ###########################################################
    # Function: check                                         #
    # Arguments: couple -- the couple to check for valid reg  #
    # Returns: a ContraCheckResult with the results           #
    # Modifies: this                                          #
    # Requires: nothing                                       #
    ###########################################################
    function check($couple,$events)
    {
        $result = new ContraCheckResult();
        if ($this->db)
        {
            $result->mergeResult($this->checkCoupleConflicts($couple,$events));
            $result->mergeResult($this->checkCoupleCompRules($couple,$events));
            if ($this->url!="")
                $result->mergeResult($this->checkCoupleEligilibity($couple,$events));
        }
        return $result;
    }

    function mayCombine($oldReg,$newReg)
    {
        #:TODO: WRITE THIS FUNCTION (ContraCheck:mayCombine)
```

```
    return new ContraCheckResult();
}

function checkCoupleConflicts($couple,$events)
{
    $result= new ContraCheckResult();
    if (!is_array($events))
      return $result;
    $tempCouple = new Couple();
    $tempCouple->setEvents($events);
    $eventIDs = $tempCouple->getGeneralData("eventid");
    $events = arrayToCSL($eventIDs);
    $result->mergeResult($this->checkPersonConflicts("leader",$couple,$events));
    $result->mergeResult($this->checkPersonConflicts("follower",$couple,$events));
    return $result;
}

function checkPersonConflicts($role,$couple,$events)
{
    $otherRole = "leader" == $role ? "follower" : "leader";
    $OtherRole = "leader" == $role ? "Follower" : "Leader";
    $Role      = "leader" == $role ? "Leader"   : "Follower";
    $getRole = "get".$Role;
    $getOtherRole = "get".$OtherRole;
    $primeID = $couple->$getRole();
    $partnerID = $couple->$getOtherRole();
    $query  = "SELECT coupleid, eventid FROM couples ";
    $query .= "NATURAL JOIN events_registration WHERE ";
    $query .= "$otherRole <> $partnerID AND ";
    $query .= "($role = $primeID OR $otherRole = $primeID) AND ";
    $query .= "eventid IN ($events);";
    $dbResult = $this->db->query($query);
    $numRows = $dbResult->numrows();
    $result = new ContraCheckResult();
    if ($numRows!=0)
    {
        $details=array();
        for ($i=0;$i<$numRows;$i++)
        {
            list($coupleID, $event) = $dbResult->getRowAt($i);
            if (isset($details[$coupleID]))
                $details[$coupleID]->addEvents(array($event));
            else
                $details[$coupleID]=new DoubleRegistrationDetail($this->db,
                                                                 $coupleID,
                                                                 array($event));
        }
        reset($details);
        while ($aDetail = each($details))
        {
            $tempDetail = $aDetail["value"];
            $result->addResult(FALSE,$tempDetail);
        }
    }
    return $result;
}

function checkCoupleCompRules($couple,$events)
{
    $tempCouple = new Couple();
    $tempCouple->setEvents($events);
```

```
        $eventIDs = $tempCouple->getGeneralData("eventid");
        $events = arrayToCSL($eventIDs);
        $result= new ContraCheckResult();
        $tempPerson = new Person($this->db);
        $tempPerson->setID($couple->getLeader());
        $tempPerson->retrieve();
        $result->mergeResult($this->checkPersonPackage($couple->getLeader(),
                                                $events,
                                                $couple->getFollower(),
                                                $tempPerson->getPackage(),"leader"));
        $tempPerson = new Person($this->db);
        $tempPerson->setID($couple->getFollower());
        $tempPerson->retrieve();
        $result->mergeResult($this->checkPersonPackage($couple->getFollower(),
                                                $events,
                                                $couple->getLeader(),
                                                $tempPerson->getPackage(),"follower"));
        return $result;
    }
    function checkPersonPackageBothRoles($personID,$events,$partnerID,$newPackage)
      {
        $result = new ContraCheckResult();
        $result->mergeResult($this->checkPersonPackage($personID,$events,
                                                       $partnerID,$newPackage,
                                                       "leader"));
        $result->mergeResult($this->checkPersonPackage($personID,$events,
                                                       $partnerID,$newPackage,
                                                       "follower"));
        return $result;
      }

    function checkPersonPackage($personID,$events,$partnerID,$newPackage,$role)
    {
      $otherRole = ($role=="leader" ? "follower" : "leader");
      $eventLevels=array();
      $styleLevels=array();
        $ccResult = new ContraCheckResult();
        $person = new Person($this->db);
        $person->setID($personID);
        $person->retrieve();
        $pid=$newPackage;
        $query  = "SELECT lvls_per_dance, lvls_overall, usesignoreevents, ";
        $query .= "usesexcludelevels, usesstylelevels, allowonlylevel, ";
        $query .= "whichlevelonly, breakwithaltrole FROM comp_reg_rules
WHERE packageid=$pid";
        $result = $this->db->query($query);
        if ($result->numrows()!=1) return $ccResult;
        list($levelsPerDance, $levelsOverall, $ignoreEvents,
             $excludeLevels, $usesStyleLevels, $allowOnlyLevel,
             $whichLevel, $breakWithAltRole) =  $result->getRowAt(0);

        if ($ignoreEvents=='t')
        {
            $query = "SELECT eventid FROM comp_reg_rules_exclude WHERE ";
            $query .= "packageid=$pid;";
            $result = $this->db->query($query);
            $numEvents = $result->numrows();
            for ($i=0;$i<$numEvents;$i++) {
                list($eventid)=$result->getRowAt($i);
                if ($i==0)
                    $ignoreEventsClause = " AND eventid NOT IN ($eventid";
```

```
            else
                $ignoreEventsClause .= ", $eventid";
        }
        if (""!=$ignoreEventsClause) $ignoreEventsClause .= ") ";
    }

    $query = "SELECT shortname, level, style FROM events_categories ";
    $query.= "NATURAL JOIN events NATURAL JOIN events_names ";
    $query.= "NATURAL JOIN events_registration NATURAL JOIN couples, ";
    $query.= "people WHERE peopleid=$personID AND ";
    if ($breakWithAltRole=='t')
      {
        $query .= "$role=peopleid AND $otherRole<>$partnerID ";
      }
    else
      {
        $query .= "((leader=peopleid AND follower<>$partnerID) OR ";
        $query .= "(leader<>$partnerID AND follower=peopleid)) ";
      }
    $query.= "$ignoreEventsClause;";
    $result = $this->db->query($query);
    $numEvents = $result->numrows();

    for ($i=0;$i<$numEvents;$i++)
    {
        list($anEvent, $aLevel, $aStyle) = $result->getRowAt($i);
        $anEvent = substr($anEvent,strpos($anEvent,'-')+1);
        $eventStyle = substr($anEvent,0,strpos($anEvent,'-'));
        $dances = substr($anEvent,strpos($anEvent,'-')+1);
        $danceStart=0;
        $j=1;
        while ($danceStart<strlen($dances))
        {
            $testDance = substr($dances,$danceStart,$j);
            $query = "SELECT * from abbreviations JOIN styles_dances
ON (fulltext=dancename) ";
            $query .= "where abbreviation='$testDance';";
            $subresult=$this->db->query($query);
            if ($subresult->numrows())
            {
                $danceStart+=$j;
                $j=0;
                $eventLevels[$eventStyle.$testDance][$aLevel]=1;
            }
            else
                $j++;
        }
        $styleLevels[$aStyle][$aLevel]=1;
        $levels[$aLevel]=1;
    }
#   echo printArray(array("EVENTS"=>$eventLevels,
#                         "STYLES"=>$styleLevels,
#                         "LEVELS"=>$levels));
    if ($events!="")
    {
        $events=explode(',',$events);
        $numEvents = count($events);
    }
    else $numEvents=0;
    for ($i=0;$i<$numEvents;$i++)
    {
```

```
            $tempEventID=$events[$i];
            $query = "SELECT shortname, level, style FROM events_names ";
            $query.= "NATURAL JOIN events NATURAL JOIN events_categories ";
            $query.= "WHERE eventid=$tempEventID;";
            $rcsult=$this->db->query($query);
            list($anEvent, $aLevel, $aStyle) = $result->getRowAt(0);
            $anEvent = substr($anEvent,strpos($anEvent,'-')+1);
            $eventStyle = substr($anEvent,0,strpos($anEvent,'-'));
            $dances = substr($anEvent,strpos($anEvent,'-')+1);
            $danceStart=0;
            $j=0;
            while ($danceStart<strlen($dances))
            {
                $testDance = substr($dances,$danceStart,$j);
                $query = "SELECT * from abbreviations  JOIN styles_dances
ON (fulltext=dancename) ";
                $query .= "where abbreviation='$testDance';";
                $subresult=$this->db->query($query);
                if ($subresult->numrows())
                {
                    $danceStart+=$j;
                    $j=0;
                    $eventLevels[$eventStyle.$testDance][$aLevel]=1;
                }
                else
                    $j++;
            }
            $styleLevels[$aStyle][$aLevel]=1;
            $levels[$aLevel]=1;
        }
#       echo printArray(array("EVENTS"=>$eventLevels,
#                             "STYLES"=>$styleLevels,
#                             "LEVELS"=>$levels));
        if ($excludeLevels=='t')
        {
            $query = "SELECT name, ignorealways, ignoreadjacentto FROM ";
            $query .= "comp_reg_rules_ignore NATURAL JOIN levels ";
            $query .= "WHERE packageid=$pid;";
            $result = $this->db->query($query);
            $numrows = $result->numrows();
            for ($i=0;$i<$numrows;$i++)
            {
                list($level,$always,$adjacent) = $result->getRowAt($i);
                if ($always=='t' && isset($levels[$level]))
                {
                    unset($levels[$level]);
                    reset($styleLevels);
                    while($aStyle= each($styleLevels))
                    {
                        $styleName = $aStyle["key"];
                        if (isset($styleLevels[$styleName][$level]))
                            unset($styleLevels[$styleName][$level]);
                    }
                    reset($eventLevels);
                    while($anEvent= each($eventLevels))
                    {
                        $eventName = $anEvent["key"];
                        if (isset($eventLevels[$eventName][$level]))
                            unset($eventLevels[$eventName][$level]);
                    }
                }
```

```
        if ($adjacent!="")
        {
            $query ="SELECT name FROM levels WHERE levelid=$adjacent;";
            $result = $this->db->query($query);
            list($adjacent)=$result->getRowAt(0);
            if ($levels[$levelName] && $levels[$adjacent])
                unset($levels[$levelName]);
            reset($styleLevels);
            while($aStyle= each($styleLevels))
            {
                $styleName = $aStyle["key"];
                if (isset($styleLevels[$styleName][$level]) &&
                    isset($styleLevels[$styleName][$adjacent]))
                    unset($styleLevels[$styleName][$level]);
            }
            reset($eventLevels);
            while($anEvent= each($eventLevels))
            {
                $eventName = $anEvent["key"];
                if (isset($eventLevels[$eventName][$level]) &&
                    isset($eventLevels[$eventName][$adjacent]))
                    unset($eventLevels[$eventName][$level]);
            }
        }
    }
}

if ($allowOnlyLevel=='t')
{
    $query = "SELECT name FROM levels WHERE levelid=$whichLevel;";
    $result = $this->db->query($query);
    list($levelName)= $result->getRowAt(0);
    if (count($levels)>1 || (count($levels)!=0 && !$levels[$levelName]))
    {
        $detail = new SingleLevelViolationDetail($this->db,$personID,
                                                 $levelName,$levels);
        $ccResult->addResult(FALSE,$detail);
    }
}

if ($levelsOverall>0)
{
    if (count($levels) > $levelsOverall)
    {
        $detail = new MaxNumLevelViolationDetail($db,$personID,
                                                 $levelsOverall,
                                                 $levels);
        $ccResult->addResult(FALSE,$detail);
    }
}
if ($levelsPerDance>0)
{
    if (isset($detail)) unset($detail);
    reset($eventLevels);
    while ($anEvent = each($eventLevels))
    {
        $eventName=$anEvent["key"];
        if (count($eventLevels[$eventName]) > $levelsPerDance)
        {
            if (isset($detail))
                $detail->addViolation($eventName,
```

```
                                                $eventLevels[$eventName]);
                    else
                        $detail = new DanceNumLevelViolationDetail($this->db,
                                                $personID,
                                                $eventName,
                                                $levelsPerDance,
                                                $eventLevels[$eventName]);
                }
            }
            if (isset($detail))
                $ccResult->addResult(FALSE, $detail);
        }
        if ($usesStyleLevels=='t')
        {
            if (isset($detail)) unset($detail);
            $query = "SELECT name, lvls_this_style FROM comp_reg_rules_styles ";
            $query.= "NATURAL JOIN styles WHERE packageid=$pid";
            $result = $this->db->query($query);
            $numStyles = $result->numrows();
            for ($i=0;$i<$numStyles;$i++)
            {
                list($style,$numLevels) = $result->getRowAt($i);
                if (count($styleLevels[$style])>$numLevels)
                {
                    if (isset($detail))
                        $detail->addViolation($style, $styleLevels[$style],
                                              $numLevels);
                    else
                        $detail = new StyleNumLevelViolationDetail($this->db,
                                              $personID,
                                              $style,
                                              $styleLevels[$style],
                                              $numLevels);
                }
            }
            if (isset($detail))
                $ccResult->addResult(FALSE, $detail);
        }
        return $ccResult;
    }

    ##########################################################
    # Function: setURL                                       #
    # Arguments: url -- the home of the info store           #
    # Returns: nothing                                       #
    # Modifies: this                                         #
    # Requires: nothing                                      #
    ##########################################################
    function setURL($url)
    {
        $this->url=$url;
    }
    ##########################################################
    # Function: getURL                                       #
    # Arguments: none                                        #
    # Returns: the url of the infostore of comp results      #
    # Modifies: this                                         #
    # Requires: nothing                                      #
    ##########################################################
    function getURL()
```

```
    {
        return $this->url;
    }
    ##########################################################
    #                                                        #
    # Function: setDB                                        #
    # Arguments:db -- the local comp database                #
    # Returns: nothing                                       #
    # Modifies: this                                         #
    # Requires: nothing                                      #
    ##########################################################
    function setDB($db)
    {
        $this->db=$db;
    }
    ##########################################################
    #                                                        #
    # Function: getDB                                        #
    # Arguments: none                                        #
    # Returns: the local comp's database handle              #
    # Modifies: this                                         #
    # Requires: nothing                                      #
    ##########################################################
    function getDB()
    {
        return $this->db;
    }

}
?>
```

contra-check/ContraCheckDetail.inc

```
<?php
#############################################################
# This file is part of SlidingDoors                         #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd #
# SlidingDoors is availible for license under the GPL, see  #
# the COPYING file in the root directory of the install for #
# the full terms of the GPL.                                #
#                                                           #
# File: ContraCheckDetail.inc                               #
# Author: Eric D. Nielsen                                   #
# Description: The abstract base class of a ContraCheckDetail #
# Attributes: details -- textual description of the reason of #
#                        a false                            #
# Constructor: ContraCheckDetail                            #
# Methods:     getText                                      #
# Change Log: 3/03/02 -- created -- edn                     #
#############################################################
class ContraCheckDetail
{
var $db;

    ###########################################################
    # Function: ContraCheckDetail                             #
    # Arguments: db -- the db to interact with                #
    # Returns: nothing                                        #
    # Modifies: this                                          #
    # Requires: nothing                                       #
    ###########################################################
    function ContraCheckDetail($db)
    {
        $this->db=$db;
    }

    ###########################################################
    # Function: getText                                       #
    # Arguments: none                                         #
    # Returns: nothing                                        #
    # Modifies: this                                          #
    # Requires: nothing                                       #
    ###########################################################
    function getText()
    {
        return "";
    }
}
?>
```

contra-check/ContraCheckResult.inc

```
<?php
#################################################################
# This file is part of SlidingDoors                             #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    #
# SlidingDoors is availible for license under the GPL, see      #
# the COPYING file in the root directory of the install for     #
# the full terms of the GPL.                                    #
#                                                               #
# File: ContraCheckResult.inc                                   #
# Author: Eric D. Nielsen                                       #
# Description: The helper class for the ContraCheckModule       #
# Attributes: result -- boolean flag, true if no problems       #
#             details -- textual description of the reason of   #
#                        a false                                #
# Constructor: ContraCheckResult                                #
# Methods:     AddResult                                        #
#              MergeResult                                      #
# Change Log: 2/19/02 -- created -- edn                         #
#################################################################
class ContraCheckResult
{
    var $details;
    var $result;

    ###############################################################
    # Function: ContraCheckResult                                 #
    # Arguments: none                                             #
    # Returns: nothing                                            #
    # Modifies: this                                              #
    # Requires: nothing                                           #
    ###############################################################
    function ContraCheckResult()
    {
        $this->details=array();
        $this->result=TRUE;
    }

    ###############################################################
    # Function: addResult                                         #
    # Arguments: result -- boolean                                #
    #            detail -- the detail object to add               #
    # Returns: nothing                                            #
    # Modifies: this                                              #
    # Requires: nothing                                           #
    ###############################################################
    function addResult($result,$detail)
    {
        $this->result &= $result;
        $this->details[count($this->details)]=$detail;
    }

    ###############################################################
    # Function: mergeResult                                       #
    # Arguments: ccResult -- another ccr to combine with this     #
    # Returns: nothing                                            #
    # Modifies: this                                              #
    # Requires: nothing                                           #
    ###############################################################
    function mergeResult($ccr)
```

```
    {
        $this->result &= $ccr->result;
        $numDetails = count($ccr->details);
        for ($i=0;$i<$numDetails;$i++)
            $this->details[count($this->details)]= $ccr->details[$i];
    }

    ############################################################
    # Function: getErrorMessage                                #
    # Arguments:                                               #
    # Returns: a english readable string describing the error  #
    # Modifies: this                                           #
    # Requires: nothing                                        #
    ############################################################
    function getErrorMessage()
    {
        if (!$this->result)
        {

            $returnString="";
            $numDetails = count($this->details);
            for ($i=0;$i<$numDetails;$i++)
            {
                $returnString.=$this->details[$i]->getText();
            }
        } else $returnString= "";
        return $returnString;
    }
}
?>
```

contra-check/DanceNumLevelViolationDetail.inc

```
<?php
#################################################################
# This file is part of SlidingDoors                             #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    #
# SlidingDoors is availible for license under the GPL, see      #
# the COPYING file in the root directory of the install for     #
# the full terms of the GPL.                                    #
#                                                               #
# File: DanceNumLevelViolationDetail.inc                        #
# Author: Eric D. Nielsen                                       #
# Description: A class that hold data describing when a         #
#              user tries to compete in a given event at too    #
#              many levels                                      #
# Attributes: details -- textual description of the reason of   #
#                        a false                                #
# Constructor: ContraCheckDetail                                #
# Methods:     getText                                          #
# Change Log: 3/03/02 -- created -- edn                         #
#################################################################
class DanceNumLevelViolationDetail extends ContraCheckDetail
{
    var $personID;
    var $personName;
    var $eventNames;
    var $levelsAllowed;
    var $levelsEntered;

    ###############################################################
    # Function: DanceNumLevelViolationDetail                      #
    # Arguments: db -- the db to interact with                    #
    #            personID --   the person generating the error    #
    #            eventName -- the events generating the error     #
    #            levelsAllowed -- the number of levels allowed    #
    #                             at this comp                    #
    #            levelsEntered -- the levels this person          #
    #                             entered                         #
    # Returns: nothing                                            #
    # Modifies: this                                              #
    # Requires: nothing                                           #
    ###############################################################
    function DanceNumLevelViolationDetail($db,$personID,$eventName,
                                          $levelsAllowed, $levelsEntered)
    {
      ContraCheckDetail::ContraCheckDetail($db);
        $this->personID = $personID;
        $person = new Person($db);
        $person->setID($personID);
        $person->retrieve();
        $this->personName = $person->getFirstName() . " " .
            $person->getLastName();
        $this->eventNames[0]=$eventName;
        $this->levelsAllowed=$levelsAllowed;
        $this->levelsEntered[$eventName]=$levelsEntered;
    }

    ###############################################################
    # Function: addViolation                                      #
    # Arguments: eventName -- the additional bad event            #
    #            eventLevels -- the levels of the bad event       #
```

```
    # Returns: nothing                                                #
    # Modifies: this                                                  #
    # Requires: nothing                                               #
    ##################################################################
    function addViolation($eventName, $eventLevels)
    {
        $this->eventNames[count($this->eventNames)]=$eventName;
        $this->levelsEntered[$eventName]=$eventLevels;
    }
    ##################################################################
    # Function: getText                                               #
    # Arguments: none                                                 #
    # Returns: nothing                                                #
    # Modifies: this                                                  #
    # Requires: nothing                                               #
    ##################################################################
    function getText()
    {
        $returnString = "<br><font color=\"red\">This competition only allows each competitor ";
        $returnString .="to register for $this->levelsAllowed level";
        $returnString .= ($this->levelsAllowed==1 ? "" :"s"). " ";
        $returnString .= " in any given dance.  Currently $this->personName is ";
        $returnString .= "attempting to break this rule with the following ";
        $returnString .= "event". (count($this->eventNames)==1?"":"s") ." ";
        $numEvents = count($this->eventNames);
        for ($i=0;$i<$numEvents;$i++)
            $events[$i]=$this->getDanceAndLevels($i);
        $returnString .= arrayToCSL($events);
        $returnString .= "</font>";
        return $returnString;
    }

    ##################################################################
    # Function: getDanceAndLevels                                     #
    # Arguments: i -- index                                           #
    # Returns: nothing                                                #
    # Modifies: this                                                  #
    # Requires: nothing                                               #
    ##################################################################
    function getDanceAndLevels($i)
    {
        $event = $this->eventNames[$i];
        $levels = array_keys($this->levelsEntered[$event]);
        $numLevels = count($levels);
        for ($i=0;$i<$numLevels;$i++)
            $shortLevels[$i]=substr($levels[$i],0,1);
        $levels = arrayToCSL($shortLevels);
        return "$event:($levels)";
    }
}
?>
```

contra-check/DoubleRegistrationDetail.inc

```
<?php
#####################################################################
# This file is part of SlidingDoors                                 #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd        #
# SlidingDoors is availible for license under the GPL, see          #
# the COPYING file in the root directory of the install for         #
# the full terms of the GPL.                                        #
#                                                                   #
# File: ContraCheckDetail.inc                                       #
# Author: Eric D. Nielsen                                           #
# Description: The abstract base class of a ContraCheckDetail       #
# Attributes: details -- textual description of the reason of       #
#                        a false                                    #
# Constructor: ContraCheckDetail                                    #
# Methods:     getText                                              #
# Change Log: 3/03/02 -- created -- edn                             #
#####################################################################
class DoubleRegistrationDetail extends ContraCheckDetail
{
    var $coupleid;
    var $leaderName;
    var $followerName;
    var $eventIDs;
    var $eventName;

    ###############################################################
    # Function: DoubleRegistrationDetail                          #
    # Arguments: db -- the db to interact with                    #
    #            coupleID --  the couple generating the error     #
    #            eventsIDs -- the events generating the error     #
    #            isPrime -- should text say you or your partner   #
    # Returns: nothing                                            #
    # Modifies: this                                              #
    # Requires: nothing                                           #
    ###############################################################
    function DoubleRegistrationDetail($db,$coupleID,$eventsIDs)
    {
      ContraCheckDetail::ContraCheckDetail($db);
        $this->coupleid = $coupleID;
        $couple = new Couple($db);
        $couple->setID($coupleID);
        $couple->retrieve();
        $leader = $couple->getLeader();
        $follower = $couple->getFollower();
        $query = "Select firstname, lastname from people where peopleid=$leader";
        $result = $db->query($query);
        list($tFirst,$tLast) = $result->getRowAt(0);
        $this->leaderName = "$tFirst $tLast";
        $query = "Select firstname, lastname from people where peopleid=$follower";
        $result = $db->query($query);
        list($tFirst,$tLast) = $result->getRowAt(0);
        $this->followerName = "$tFirst $tLast";
        if (is_array($eventsIDs))
            $numEvents = count($eventsIDs);
        else
            $numEvents = 0;
        $this->eventIDs=array();
        $numEvents=count($eventsIDs);
        for ($i=0;$i<$numEvents;$i++) {
```

```
            $tid = $eventsIDs[$i];
            $this->eventIDs[$i]=$tid;
            $query = "Select shortname from events_names where eventid=$tid;";
            $result = $db->query($query);
            list($tname) = $result->getRowAt(0);
            if ($i!=0)
                $this->eventNames.=", ";
            $this->eventNames.= $tname;
        }

    }

    ##########################################################
    # Function: addEvents                                    #
    # Arguments: eventIDs                                    #
    # Returns: nothing                                       #
    # Modifies: this                                         #
    # Requires: nothing                                      #
    ##########################################################
    function addEvents($eventIDs)
    {
        $numEvents = count($eventIDs);
        for ($i=0;$i<$numEvents;$i++) {
            $tid = $eventIDs[$i];
            $query = "Select shortname from events_names where eventid=$tid;";
            $result = $this->db->query($query);
            list($tname) = $result->getRowAt(0);
            if ($this->eventNames!="")
                $this->eventNames.=", ";
            $this->eventNames.= $tname;
            $this->eventIDs[count($this->eventIDs)]=$tid;
        }
        return "";
    }
    ##########################################################
    # Function: getText                                      #
    # Arguments: none                                        #
    # Returns: nothing                                       #
    # Modifies: this                                         #
    # Requires: nothing                                      #
    ##########################################################
    function getText()
    {
        $returnString = "<br><font color=\"red\">";
        $returnString .= "The couple ($this->leaderName, $this->followerName)
is already dancing $this->eventNames.  They must drop those events before you
may register for those events.</font>";
        return $returnString;
    }
}
?>
```

contra-check/SingleLevelViolationDetail.inc

```
<?php
#############################################################
# This file is part of SlidingDoors                         #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd #
# SlidingDoors is availible for license under the GPL, see  #
# the COPYING file in the root directory of the install for #
# the full terms of the GPL.                                #
#                                                           #
# File: SingleLevelViolationDetail.inc                    #
# Author: Eric D. Nielsen                                   #
# Description: A class that hold data describing when a     #
#              user trics to compete in a given event at too #
#              many levels                                  #
# Attributes: details -- textual description of the reason of #
#                        a false                            #
# Constructor: ContraCheckDetail                            #
# Methods:     getText                                      #
# Change Log: 3/03/02 -- created -- edn                     #
#############################################################
class SingleLevelViolationDetail extends ContraCheckDetail
{
    var $personID;
    var $personName;
    var $levelAllowed;
    var $levelsEntered;

    ###########################################################
    # Function: SingleLevelViolationDetail                    #
    # Arguments: db -- the db to interact with                #
    #            personID --   the person generating the error #
    #            levelAllowed -- the name of thelevels allowed #
    #                            at this comp                 #
    #            levelsEntered -- the levels this person      #
    #                            entered                      #
    # Returns: nothing                                        #
    # Modifies: this                                          #
    # Requires: nothing                                       #
    ###########################################################
    function SingleLevelViolationDetail($db,$personID,
                                        $levelAllowed, $levelsEntered)
    {
      ContraCheckDetail::ContraCheckDetail($db);
        $this->personID = $personID;
        $person = new Person($db);
        $person->setID($personID);
        $person->retrieve();
        $this->personName = $person->getFirstName() . " " .
            $person->getLastName();
        $this->levelAllowed=$levelAllowed;
        $this->levelsEntered=$levelsEntered;
    }

    ###########################################################
    # Function: getText                                       #
    # Arguments: none                                         #
    # Returns: nothing                                        #
    # Modifies: this                                          #
    # Requires: nothing                                       #
    ###########################################################
```

```
    function getText()
    {
        $returnString = "<br><font color=\"red\">The package selected by $this->personName ";
        $returnString .="only includes entries in the $this->levelAllowed ";
        $returnString .= "level.  If $this->personName wants to compete in (";
        $returnString .= arrayToCSL(array_keys($this->levelsEntered));
        $returnString .= ") $this->personName will have to select a different
package or drop events first.";
        $returnString .= "</font>";
        return $returnString;
    }

}
?>
```

database/CIB_DB.inc

```
<?php
/*****************************************************************
 * This file is part of CompInaBox                               *
 * Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    *
 * CompInaBox is availible for license under the GPL, see        *
 * the COPYING file in the root directory of the install for     *
 * the full terms of the GPL.                                    *
 *                                                               *
 * File: CIB_DB.inc                                              *
 * Type: Class File                                              *
 * Author: Eric D. Nielsen                                       *
 * Description: Database wrapper for the central CIB database    *
 * Attributes: from base class (DB)                              *
 * Constructor: CIB_DB                                           *
 * Methods:                                                      *
 * Change Log: 8/12/02 -- created -- edn                         *
 *****************************************************************/
class CIB_DB extends DB
{
  // NO NON-DERIVED ATTRIBUTED

  // CONSTRUCTOR
  /*****************************************************************
   * Function: CIB_DB                                              *
   * Arguments: dbname -- what local db to connect to              *
   *            user -- who is connecting                          *
   *            pass -- with what password                         *
   * Modifies: this                                                *
   *****************************************************************/
  function CIB_DB($dbname, $user, $pass)
    {
      DB::DB($dbname, $user, $pass);
    }

  // PUBLIC METHODS
  function getDocumentList($type="")
    {
      $query = "SELECT categories.name, documents.name, url FROM ";
      $query.= "documents NATURAL JOIN document_categories ";
      $query.= "JOIN categories USING (categoryid) ";
      if ($type!="")
        {
          $listType = arrayToCSL($type);
          $query.= "WHERE categories.name IN ($listType) ";
        }
      $query .= "ORDER BY cat_order, doc_order;";

      $items = $this->helper_queryToArrayCategories($query);
      if (is_array($type) && count($type)==1)
        {
          $index = substr($type[0],1,strlen($type[0])-2);;
          return $items[$index];
        }
      else
        return $items;
    }

  function getComps($status="",$order="occurs_on", $limit="",$offset="")
    {
```

```
    $query = "SELECT compname, compunix FROM ";
    $query.= "comps JOIN comp_status USING (statusid) ";
    if ($status!="") $query.="WHERE comp_status.name='$status'";
    $query .= "ORDER BY $order ";
    $query.= $this->helper_limitOffset($limit,$offset).";";
    return $this->helper_queryToArray($query);
  }

function getTeams()
  {
    $query = "SELECT name, teamid FROM teams ORDER BY name;";
    return $this->helper_queryToArray($query);
  }

function getCibServers($excludeSelf=TRUE,$limit="",$offset="")
  {
    global $CIB_BASE_URL;
    $query = "SELECT host, url FROM ";
    $query.= "cib_servers ";
    if ($excludeSelf)
      $query .= "WHERE url <> '$CIB_BASE_URL/' ";
    $query.= $this->helper_limitOffset($limit,$offset).";";
    return $this->helper_queryToArray($query);
  }

function getOutsideComps($limit="",$offset="")
  {
    $query = "SELECT name, url FROM ";
    $query.= "outside_comps ";
    $query.= $this->helper_limitOffset($limit,$offset).";";
    return $this->helper_queryToArray($query);
  }
function getUserIdByName($username)
  {
    $query ="SELECT userid FROM users WHERE username='$username';";
    return $this->helper_singleValueQuery($query);
  }

function getUserPasswordById($userid)
  {
    $query = "SELECT hashed FROM users WHERE userid=$userid";
    return $this->helper_singleValueQuery($query);
  }

function getCompStatusFromStatusID($id)
  {
    $query = "SELECT name FROM comp_status WHERE statusid=$id;";
    return $this->helper_singleValueQuery($query);
  }

function getCompStatusIDFromStatus($status)
  {
    $query = "SELECT statusid FROM comp_status WHERE name='$status';";
    return $this->helper_singleValueQuery($query);
  }

function getCompStatusChoices()
  {
    $choices=array();
    $query = "SELECT statusid, name FROM comp_status ORDER BY statusid;";
    $result = $this->query($query);
```

```
    $numChoices = $result->numrows();
    for ($i=0;$i<$numChoices;$i++)
      {
        list($id,$name)= $result->getRowAt($i);
        $choices[$name]=$id;
      }
    return $choices;
  }

function getCompUnixNames()
  {
    $choices=array();
    $query = "SELECT compunix FROM comps ORDER BY compunix;";
    $result = $this->query($query);
    $numChoices = $result->numrows();
    for ($i=0;$i < $numChoices;$i++)
      {
        list($name)= $result->getRowAt($i);
        $choices[$i]=$name;
      }
    return $choices;
  }

function getTeamIDs()
  {
    $choices=array();
    $query = "SELECT teamid FROM teams ORDER BY teamid;";
    $result = $this->query($query);
    $numChoices = $result->numrows();
    for ($i=0;$i < $numChoices;$i++)
      {
        list($name)= $result->getRowAt($i);
        $choices[$i]=$name;
      }
    return $choices;
  }

function getUserRoles($type="comp")
  {
    $choices=array();
    $query = "SELECT rolename FROM {$type}_roles ORDER BY rolename;";
    $result = $this->query($query);
    $numChoices = $result->numrows();
    for ($i=0;$i < $numChoices;$i++)
      {
        list($name)= $result->getRowAt($i);
        $name = substr($name,strpos($name," ")+1);
        $choices[$i]=$name;
      }
    return $choices;
  }

function getCompStatusChoices()
  {
    $choices=array();
    $query = "SELECT statusid , name FROM comp_status ORDER BY statusid;";
    $result = $this->query($query);
    $numChoices = $result->numrows();
    for ($i=0;$i<$numChoices;$i++)
      {
        list($id,$name)= $result->getRowAt($i);
```

```
            $choices[$name]=$id;
          }
        return $choices;
      }


    function expandTeamNamesFromID($keys)
      {
        $numKeys = count($keys);
        for ($i=0;$i<$numKeys;$i++)
          {
            if ($keys[$i][$type]!="Team") continue;
            $id = $keys[$i]["TeamID"];
            $query = "SELECT name FROM teams WHERE teamid=$id;";
            $keys[$i]["Teamname"]=$this->helper_singleValueQuery($query);
          }
        return $keys;
      }

    function getPendingKeys($userid=0)
      {
        $query = "SELECT type,level,details FROM pending_permissions WHERE
requestor=$userid;";
        $result = $this->query($query);
        $numResults = $result->numrows();
        $keys=array();
        for ($i=0;$i<$numResults;$i++)
          {
            list($type,$level,$details)=$result->getRowAt($i);
            switch ($type)
              {
              case "Site" : $keys[]=array("Type"=>"Site","Level"=>$level);break;
              case "Comp" : $keys[]=array("Type"=>"Comp",
                                          "Level"=>$level,
                                          "Compname"=>$details);break;
              case "Team" : $keys[]=array("Type"=>"Team",
                                          "Level"=>$level,
                                          "Teamname"=>$details);break;
              default:break;
              }
          }
        return $keys;
      }
    function getPendingKey($permid=0)
      {
        $query = "SELECT type,level,details FROM pending_permissions WHERE
pendingid=$permid;";
        $result = $this->query($query);
        if ($result->numrows()==1)
          {
            list($type,$level,$details)=$result->getRowAt(0);
            switch ($type)
              {
              case "Site" : $key=array("Type"=>"Site","Level"=>$level);break;
              case "Comp" : $key=array("Type"=>"Comp",
                                       "Level"=>$level,
                                       "Compname"=>$details);break;
              case "Team" : $key=array("Type"=>"Team",
                                       "Level"=>$level,
                                       "Teamname"=>$details);break;
              default:break;
```

```
            }
          }
        else
          $key=array();
        return $key;
      }

    function grantPermissionToUser($permid,$userid)
      {
        GLOBAL $CIB_SERVER_HOST_NAME,$CIB_HOSTNAME,$CIB_DOMAIN;
        $query = "SELECT type,level,details FROM pending_permissions WHERE
pendingid=$permid;";
        $result = $this->query($query);
        if ($result->numrows()==1)
          {
            list($type,$level,$details)=$result->getRowAt(0);
            switch ($type)
              {
              case "Site" : switch ($level)
                {
                case "Admin" : $query = "UPDATE users SET siteadmin=true WHERE
user=$userid;";
                  $result = $this->query($query);
                  break;
                default : break;
                }
              case "Comp" :
                $query = "SELECT roleid FROM comp_roles WHERE
rolename='Competition_$level';";
                $roleid = $this->helper_singleValueQuery($query);
                $query = "SELECT compid FROM comps WHERE compunix='$details';";
                $compid = $this->helper_singleValueQuery($query);
                $query = "INSERT INTO user_comp_roles (userid,compid,roleid)
VALUES ($userid,$compid,$roleid);";
                $this->query($query);
                break;
              case "Team" :
                $query = "SELECT roleid FROM team_roles WHERE rolename LIKE
'Affiliation_$level%';";
                $roleid = $this->helper_singleValueQuery($query);
                $query = "SELECT teamid FROM teams WHERE name='$details';";
                $teamid = $this->helper_singleValueQuery($query);
                $query = "INSERT INTO user_team_roles (userid,teamid,roleid)
VALUES ($userid,$compid,$roleid);";
                $this->query($query);
                break;
              default: break;
              }
            $query = "DELETE FROM pending_permissions WHERE pendingid=$permid;";
            $this->query($query);
            $query = "SELECT email FROM users WHERE userid=$userid;";
            $to_email = $this->helper_singleValueQuery($query);
            $header="From: noreply@$CIB_HOSTNAME"."$CIB_DOMAIN\r\n";
            $title="[CompInaBox-$CIB_SERVER_HOST_NAME] Permission Grant";
            $body=<<<END_EMAIL
This email is sent to you to inform you that you have been granted
increased permissions as designated by $type $details $level.

-- The $CIB_SERVER_HOST_NAME CompInaBox staff
END_EMAIL;
            mail($to_email,$title,$body,$headers);
```

```
        }
    }


  function denyPermissionToUser($permid,$userid)
    {
      GLOBAL $CIB_SERVER_HOST_NAME,$CIB_HOSTNAME,$CIB_DOMAIN;
      $query = "SELECT type,level,details FROM pending_permissions WHERE
pendingid=$permid;";
      $result = $this->query($query);
      if ($result->numrows()==1)
        {
          list($type,$level,$details)=$result->getRowAt(0);
          $query = "DELETE FROM pending_permissions WHERE pendingid=$permid;";
          $this->query($query);
          $query = "SELECT email FROM users WHERE userid=$userid;";
          $to_email = $this->helper_singleValueQuery($query);
          $header="From: noreply@$CIB_HOSTNAME"."$CIB_DOMAIN\r\n";
          $title="[CompInaBox-$CIB_SERVER_HOST_NAME] Permission Denied";
          $body=<<<END_EMAIL
This email is sent to you to inform you that your request for the
increased permissions as designated by $type $details $level has been denied.

-- The $CIB_SERVER_HOST_NAME CompInaBox staff
END_EMAIL;
          mail($to_email,$title,$body,$headers);
        }
    }



  function getPendingPermissions($user)
    {
      if (!$user) return array();
      $query = "SELECT pendingid,requestor,username,type,level,details ";
      $query.= "FROM pending_permissions JOIN users ON (requestor=userid) ";
      $query.= "ORDER BY username,type,level,details;";
      $result = $this->query($query);
      $numPerms = $result->numrows();
      $resultSet = array();
      for ($i=0;$i<$numPerms;$i++)
        {
          list($permID,$reqID,$username,$type,$level,$details) =
            $result->getRowAt($i);
          unset($key);
          switch ($type)
            {
            case "Site" : $key = array("Type"=>$type,"Level"=>$level);break;
            case "Comp" : $key = array("Type"=>$type,
                                       "Level"=>$level,
                                       "Compname"=>$details); break;
            case "Team" : $key = array("Type"=>$type,
                                       "Level"=>$level,
                                       "Teamname"=>$details); break;
            default:break;
            }
          if (isset($key))
            if ($user->mayAuthorizeKey($key))
              $resultSet[] = array("Username"=>$username,
                                   "UserID"=>$reqID,
                                   "PermID"=>$permID,
```

```
                                    "Key"=>$key);
        }
      return $resultSet;
    }

  function addPending($userid,$key)
    {
      $type = $key["Type"];
      $level = $key["Level"];
      $query = "INSERT INTO pending_permissions (requestor,type,level";
      if ($type!="Site") $query .= ",details";
      $query .= ") VALUES ($userid,'$type','$level'";
      $details="";
      if ($type=="Team") $details = $key["TeamID"];
      else if ($type=="Comp") $details = $key["Compname"];
      if (""!=$details) $query .= ",'$details'";
      $query .= ");";
      $this->query($query);
    }



  // PRIVATE METHODS
  function helper_singleValueQuery($query)
    {
      $result = $this->query($query);
      if ($result->numrows())
        list($value)  = $result->getRowAt(0);
      else
        $value="";
      return $value;
    }

  function helper_queryToArray($query)
    {
      $result = $this->query($query);
      $items=array();
      $numItems = $result->numrows();
      for ($i = 0; $i < $numItems; $i++)
        {
          list($name,$link) = $result->getRowAt($i);
          $items[$i]=array("Label"=>$name,"Link"=>$link);
        }
      return $items;
    }

  function helper_queryToArrayCategories($query)
    {
      $result = $this->query($query);
      $items=array();
      $numItems = $result->numrows();
      for ($i = 0; $i < $numItems; $i++)
        {

          list($cat,$name,$link) = $result->getRowAt($i);
          $items[$cat][]=array("Label"=>$name,"Link"=>$link);
        }
      return $items;
    }

  function helper_limitOffset($limit,$offset)
```

```
    {
      $query = "";
      if ($limit!="")
        $query .= "LIMIT $limit ";
      if ($offset!="")
        $query .= "OFFSET $offset ";
      return $query;
    }

}
?>
```

database/CompDB.inc

```
<?php
#####################################################################
# This file is part of SlidingDoors                                 #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd        #
# SlidingDoors is availible for license under the GPL, see          #
# the COPYING file in the root directory of the install for         #
# the full terms of the GPL.                                        #
#                                                                   #
# File: CompDB.inc                                                  #
# Author: Eric D. Nielsen                                           #
# Description: Descendant of DB, with common stat query's in        #
#_____________method form_____________________________________      #
#_Attributes:__only inheireted_____________________________________ #
#_Constructor:_CompDB______________________________________________ #
#_Methods:_____getNumAffiliations__________________________________ #
#_____________connectionValid______________________________________ #
#_Change_Log:_6/12/01___created__edn_______________________________ #
#####################################################################
class_CompDB_extends_DB
{
____var_$lastUsed;

__#################################################################
____#_Function:_CompDB_____________________________________________#
____#_Arguments:_dbname___where_to_connect_____________________    #
____#___________user___who_is_connecting_____________________     #
____#___________pass___with_what_password____________________     #
____#_Returns:_nothing_____________________________________________#
____#_Modifies:_this_______________________________________________#
____#_Requires:_nothing____________________________________________#
__#################################################################
____function_CompDB($dbname,_$user,_$pass)
____{
________DB::DB($dbname,_$user,_$pass);
____}

____function_getStatus()
____{
________$query_=_"SELECT_status_FROM_comp_status;";
________$result_=_$this->query($query);
________list($res)_=_$result->getRowAt(0);
________return_$res;
____}
____function_setStatus($status)
____{
________$query_=_"UPDATE_comp_status_SET_status='$status';";
________$result_=_$this->query($query);
____}

function_getHost()
{
__$query_=_"SELECT_hostingorg_FROM_comp;";
__$result_=_$this->query($query);
__if_($result->numrows())
____list($orgid)=$result->getRowAt(0);
__return_$orgid;
}

__#################################################################
```

```
----#-Function:-getNumAffiliations----------#
    # Arguments: nothing                     #
    # Returns:  The number of unique affiliations associated  #
    #           with at least one competitor registered for at #
    #           least one event.  If the database can not be  #
    #           accessed a zero will be returned.             #
    # Requires: nothing                      #
    # Modifies: nothing                      #
    ##########################################################
    function getNumAffiliations()
    {
        $query =  "SELECT count(distinct organization) ";
        $query .=    "FROM people, couples, events_registration ";
        $query .=    "WHERE (peopleid=follower or peopleid=leader) ";
        $query .=        "AND couples.coupleid=events_registration.coupleid;";
        $result = $this->query($query);
        if ($result->getError())
            $numAffil=0;
        else
            list($numAffil)=$result->getRowAt(0);
        return $numAffil;
    }

    ##########################################################
    # Function: getNumCompetitors                            #
    # Arguments: nothing                                     #
    # Returns:  The number of unique individuals registered  #
    #           for at least one event.  If the database can #
    #           not be accessed a zero will be returned.     #
    # Requires: nothing                                      #
    # Modifies: nothing                                      #
    ##########################################################
    function getNumCompetitors()
    {
        $query =  "SELECT count(distinct peopleid) ";
        $query .=    "FROM people, couples, events_registration ";
        $query .=    "WHERE (peopleid=follower or peopleid=leader) ";
        $query .=        "AND couples.coupleid=events_registration.coupleid;";
        $result = $this->query($query);
        if ($result->getError())
            $numCompetitors=0;
        else
            list($numCompetitors)=$result->getRowAt(0);
        return $numCompetitors;
    }


    ##########################################################
    # Function: getNumCouples                                #
    # Arguments: db -- a handle to the database              #
    # Returns:  The number of unique couples registered for at #
    #           least one event.  If the database can not be  #
    #           accessed a zero will be returned.             #
    # Requires: nothing                                      #
    # Modifies: nothing                                      #
    # Note: A unique couple is an ordered pair of lead/follow  #
    ##########################################################
    function getNumCouples()
    {
        $query =  "SELECT count(distinct coupleid) ";
        $query .=    "FROM events_registration;";
```

```
        $result = $this->query($query);
        if ($result->getError())
            $numCouples=0;
        else
            list($numCouples)=$result->getRowAt(0);
        return $numCouples;
    }

    ##############################################################
    # Function: getNumEntries                                    #
    # Arguments: nothing                                         #
    # Returns:  The sum, across all the events, of the couples   #
    #           entered.  If the database can not be accessed    #
    #           a zero will be returned.                         #
    # Requires: nothing                                          #
    # Modifies: nothing                                          #
    ##############################################################
    function getNumEntries($db)
    {
        $query =  "SELECT count(coupleid) ";
        $query .=     "FROM events_registration;";
        $result = $this->query($query);
        if ($result->getError())
            $numEntries=0;
        else
            list($numEntries)=pg_fetch_array($result,0);
        return $numEntries;
    }


    ##############################################################
    # Function: getAvgEntries                                    #
    # Arguments: level -- what level are we summarizing          #
    #             stlyle -- which style are we summarizing       #
    #                       defaults to "All"                    #
    # Returns:  The average, across the requested events, of     #
    #           the couples entered.  If the database can not    #
    #           be accessed a zero will be returned.             #
    # Requires: nothing                                          #
    # Modifies: nothing                                          #
    # Notes:  This function does not include Team Matches or     #
    #         Fun Dance registrations in its calculations.       #
    ##############################################################
    function getAvgEntries($level, $style="All")
    {
        if ($style=="All")
        {
            $query =  "SELECT count (distinct eventid) ";
            $query .=     "FROM events, events_categories ";
            $query .=     "WHERE level =  '$level' ";
____________$query .=______"AND type IS NULL ";
____________$query .=______"AND events.categoryid = events_categories.categoryid ";
____________$result = $this->query($query);
____________if ($result->getError())
________________$numEvents=0;
____________else
________________list($numEvents)=$result->getRowAt(0);
____________$query =  "SELECT sum (curreg) ";
____________$query .=    "FROM events_est_reg, events, events_categories ";
____________$query .=    "WHERE events_est_reg.eventid = events.eventid ";
____________$query .=    "AND level = '$level' ";
```

```
            $query .=       "AND events.categoryid = events_categories.categoryid;";
            $result = $this->query($query);
            if ($result->getError())
                $numEntries=0;
            else
                list($numEntries)=$result->getRowAt(0);
        }
        else
        {
            $query =  "SELECT count (distinct eventid) ";
            $query .=     "FROM events, events_categories ";
            $query .=     "WHERE level = '$level' ";
            $query .=     "AND style = '$style' ";
            $query .=     "AND type IS NULL ";
            $query .=     "AND events.categoryid = events_categories.categoryid;";
            $result = $this->query($query);
            if ($result->getError())
                $numEvents=0;
            else
                list($numEvents)=$result->getRowAt(0);
            $query = "SELECT sum(curreg) ";
            $query .= "FROM events_est_reg, events, events_categories ";
            $query .= "WHERE level = '$level' ";
            $query .= "AND style = '$style' ";
            $query .= "AND type IS NULL ";
            $query .= "AND events.categoryid = events_categories.categoryid ";
            $query .= "AND events_est_reg.eventid = events.eventid; ";
            $result = $this->query($query);
            if ($result->getError())
                $numEntries=0;
            else
                list($numEntries)=$result->getRowAt(0);
        }
        return ($numEvents!=0?round($numEntries/$numEvents):0);
    }


    function getTotalCompFees($paid=FALSE,$excludeOrg="")
    {
        $query="SELECT SUM(cost) FROM packages_purchased NATURAL JOIN
people_classification JOIN rules_age_level ON (people_classification.agelevel=
rules_age_level.index) JOIN packages_cost ON (packages_purchased.packageid=
packages_cost.packageid AND packages_cost.category=rules_age_level.name) ";
        if ($excludeOrg!="" || $paid)
            $query .="NATURAL JOIN people_paid ";
        if ($excludeOrg!="" || $paid)
            $query .="WHERE ";
        if ($excludeOrg!="")
            $query .=" org_paying <> $excludeOrd ";
        if ($excludeOrg!="" && $paid)
            $query .="AND ";
        if ($paid)
            $query .=" paid=TRUE";
        $query .=";";
        $result = $this->query($query);
        list ($total) = $result->getRowAt(0);
        return $total;
    }

function getIncompleteReg()
```

```
{
  $badRegs = array();
  $query = "SELECT distinct peopleid FROM people_paid left outer join couples
on (peopleid=follower OR peopleid=leader) WHERE leader IS NULL;";
  $reuslt = $this->query($query);
  $numBad = $result->numrows();
  for ($i=0; $i < $numBad; $i++)
    list($badRegs[$i])=$result->getRowAt($i);
  return $badRegs;
}

function getMOTD($label="Main")
  {
    $query = "SELECT message FROM motd WHERE label='$label';";
____$result_=_$this->query($query);
____if_($result->numrows())
______list($message)_=_$result->getRowAt(0);
____else
______$message="";
____$message_=_stripslashes($message);
____return_$message;
__}

function_setMOTD($message,$label="Main")
__{
____$message_=_addslashes($message);
____$query_=_"UPDATE_motd_SET_message='$message'_WHERE_label='$label';";
____$this->query($query);
__}


__#################################################################
____#_Function:_getAffilListChoices_____________________________#
____#_Arguments:_db_--_a_handle_to_the_database_________________#
____#_Returns:__A_string_with_all_the_affiliations_formatted____#
____#___________between_<OPTION>_tags_for_use_in_<SELECT>_tag._#
____#___________Returns_the_empty_string_if_no_db_handle._____#
____#_Requires:_nothing_______________________________________#
____#_Modifies:_nothing_______________________________________#
__#################################################################
____function_getAffilListChoices()
____{
________$query_=__"SELECT_distinct_orgid,_name_";
________$query_.=____"FROM_organizations,_people_";
________$query_.=____"WHERE_organization=orgid;";
________$result_=_$this->query($query);
________if_($result->getError())
____________$listString="";
________else
________{
__________$listString="<OPTION_value=\"-1\">No_Affiliation</OPTION>\n";
____________$count_=_$result->numRows();
____________for_($i=0;$i<$count;$i++)
__________{
________________$temp_=_$result->getRowAt($i);
#_______________echo_printArray($temp)_."<BR>";
________________list($orgid,_$name)=$result->getRowAt($i);
#_______________echo_$name;
________________$nameVal_=_$name;
________________$query_=__"SELECT_abbreviation_FROM_abbreviations_";
________________$query_.=_"WHERE_fulltext='$name';";
```

```
                $abbrevResult = $this->query($query);
                if (!$abbrevResult->getError())
                    if ($abbrevResult->numRows()==1)
                        list($name)=$abbrevResult->getRowAt(0);
                $listString .="\n\t<OPTION VALUE=\"$orgid\">$name</OPTION>";
            }
        }
        return $listString;
    }

    ###########################################################
    # Function: getAffilListChoicesSelect                     #
    # Arguments: chosen -- the value of the option to set     #
    # Returns:  A string with all the affiliations formatted  #
    #           between <OPTION> tags for use in <SELECT> tag.#
    #           Returns the empty string if no db handle.     #
    # Requires: nothing                                       #
    # Modifies: nothing                                       #
    # Note: Should probably be rewritten to use the above func#
    ###########################################################
    function getAffilListChoicesSelect($chosen)
    {
        $query =  "SELECT distinct orgid, name ";
        $query .=    "FROM organizations, people ";
        $query .=    "WHERE organization=orgid;";
        $result = $this->query($query);
        if ($result->getError())
            $listString="";
        else
        {
            if ($chosen==-1)
                $listString="<OPTION SELECTED value=\"-1\">No Affiliation</OPTION>\n";
            else
                $listString="<OPTION value=\"-1\">No Affiliation</OPTION>\n";
            $count = $result->numRows();
            for ($i=0;$i<$count;$i++)
            {
                $selected=FALSE;
                list($orgid, $name)=$result->getRowAt($i);
                $nameVal = $name;
                if ($name==$chosen)
                    $selected=TRUE;
                $query =  "SELECT abbreviation FROM abbreviations ";
                $query .= "WHERE fulltext='$name';";
                $abbrevResult = $this->query($query);
                if (!$abbrevResult->getError())
                    if ($abbrevResult->numRows()==1)
                        list($name)=$abbrevResult->getRowAt(0);
                if ($name==$chosen)
                    $selected=TRUE; # just in case I change how it works later
                $listString .="\n\t<OPTION VALUE=\"$orgid\"";
                $listString .=($selected ? " SELECTED" : "");
                $listString .=">$name</OPTION>";
            }
        }
        return $listString;
    }

    ###########################################################
    # Function: getAgeCategories                              #
    # Arguments: selName --- the name of the radio button     #
```

```
    #              chosen --- the value of the option to set      #
    # Returns:  A set of radio buttons for selecting what age     #
    #           category a competitor falls into.                 #
    # Requires: nothing                                           #
    # Modifies: nothing                                           #
    ###############################################################
    function getAgeCategories($chosen,$selName="age")
    {
        if (!isset($chosen) || ($chosen==""))
            $chosen="Student";
        $query =  "SELECT name FROM rules_age_level ORDER BY index";
        $result = $this->query($query);
        if ($result->getError())
            $listString="";
        else
        {
            $listString="";
            $count = $result->numRows();
            for ($i=0;$i<$count;$i++)
            {
                $selected=FALSE;
                list($name)=$result->getRowAt($i);
                if ($name==$chosen)
                    $selected=TRUE;
                $listString .="\n\t$name:<input type=\"radio\"
name=\"$selName\" value=\"$name\"";
                $listString .= ($selected ? " CHECKED ": "") . ">\n";
            }
        }
        return $listString;
    }

    ###############################################################
    # Function: getAgeCategoriesSelect                            #
    # Arguments: selName -- the name of the select box            #
    #            chosen -- the value of the option to set         #
    # Returns:  A select box for selecting what age               #
    #           category a competitor falls into.                 #
    # Requires: nothing                                           #
    # Modifies: nothing                                           #
    ###############################################################
    function getAgeCategoriesSelect($chosen,$selName="age")
    {
        if (!isset($chosen) || ($chosen==""))
            $chosen="Student";
        $query =  "SELECT name FROM rules_age_level ORDER BY index";
        $result = $this->query($query);
        if ($result->getError())
            $listString="";
        else
        {
            $listString="<SELECT name=\"$selName\">";
            $count = $result->numRows();
            for ($i=0;$i<$count;$i++)
            {
                $selected=FALSE;
                list($name)=$result->getRowAt($i);
                if ($name==$chosen)
                    $selected=TRUE;
                $listString .="\n\t<option value=\"$name\" ";
                $listString .= ($selected ? " SELECTED ": "");
```

```
                $listString .= ">$name</option>";
            }
            $listString .= "</SELECT>";
        }
        return $listString;
    }

    ##########################################################
    # Function: getPackages                                  #
    # Arguments: chosen -- the value of the option to set    #
    # Returns:  A set of radio buttons for selecting what    #
    #           package a competitor is purchasing           #
    # Requires: nothing                                      #
    # Modifies: nothing                                      #
    ##########################################################
    function getPackages($chosen,$selName="package")
    {
        GLOBAL $baseURL;
        if (!isset($chosen) || ($chosen==""))
            $chosen="2";#HACK
        $query =  "SELECT packageid, name FROM packages order by packageid";
        $result = $this->query($query);
        if ($result->getError())
            $listString="";
        else
        {
            $listString="";
            $count = $result->numRows();
            for ($i=0;$i<$count;$i++)
            {
                $selected=FALSE;
                list($pid, $name)=$result->getRowAt($i);

                $subResult = $this->query("SELECT distinct cost from
packages_cost where packageid=$pid order by cost;");
                $subNumRows = $subResult->numRows();
                $costString="";
                for ($j=0;$j<$subNumRows;$j++)
                {
                    list($cost)=$subResult->getRowAt($j);
                    if ($costString=="")
                        $costString= "\$" . $cost;
                    else
                        $costString.="/$cost";
                }
                if ($pid==$chosen)
                    $selected=TRUE;
                $listString .="\n\t<a href=\"$baseURL/PREPARSED/packages.html#
$name\">$name</a>($costString):<input type=\"radio\" name=\"$selName\"
value=\"$pid\"";
                $listString .= ($selected ? " CHECKED ": "") . "><br />\n";;
            }
        }
        return $listString;
    }


    ##########################################################
    # Function: getPackagesSelect                            #
    # Arguments: selName -- the name of the select box       #
    #            chosen -- the value of the option to set    #
```

```
    # Returns:  A select box for selecting what package        #
    #           a competitor falls into.                       #
    # Requires: nothing                                        #
    # Modifies: nothing                                        #
    ############################################################
    function getPackagesSelect($chosen,$selName="package")
    {
        if (!isset($chosen) || ($chosen==""))
            $chosen="Student";
        $query =  "SELECT packageid, name FROM packages order by packageid";
        $result = $this->query($query);
        if ($result->getError())
            $listString="";
        else
        {
            $listString="<SELECT name=\"$selName\">";
            $count = $result->numRows();
            for ($i=0;$i<$count;$i++)
            {
                $selected=FALSE;
                list($pid, $name)=$result->getRowAt($i);

                $subResult = $this->query("SELECT distinct cost from
packages_cost where packageid=$pid order by cost;");
                $subNumRows = $subResult->numRows();
                $costString="";
                for ($j=0;$j<$subNumRows;$j++)
                {
                    list($cost)=$subResult->getRowAt($j);
                    if ($costString=="")
                        $costString= "\$" . $cost;
                    else
                        $costString.="/$cost";
                }
                if ($pid==$chosen)
                    $selected=TRUE;
                $listString .="\n\t<option value=\"$pid\" ";
                $listString .= ($selected ? " SELECTED ": "") . ">\n";
                $listString .= "$name($costString)</option>\n";
            }
        }
        $listString .= "</SELECT>";
        return $listString;
    }

    ############################################################
    function getSelectMembers($affil,$chosen="0")
    {
        $query =  "SELECT peopleid, firstname, lastname FROM people ";
        $query .= "WHERE organization=$affil ORDER BY lastname, firstname";
        $result = $this->query($query);
        $listString="<option value=\"0\">New Member</option>";
        if (!$result->getError())
        {
            $count = $result->numRows();
            for ($i=0;$i<$count;$i++)
            {
                $selected=FALSE;
                list($id, $first,$last)=$result->getRowAt($i);
                if ($id==$chosen)
                    $selected=TRUE;
```

```
            $listString .="<option value=\"$id\" ";
            $listString .= ($selected ? " SELECTED ": "") . ">\n";;
            $listString .= "$last, $first</option>";
        }
    }
    return $listString;
}

function personConflicts($person,$includeEmail=TRUE, $exact=FALSE)
#############################################################
# Function: getNameConflicts                                #
# Arguments: person -- the person who might conflict        #
#            includeEmail -- false if email matches should  #
#                            generate a false               #
#            exact -- first, last, and email must all       #
#                     match for a valid match               #
# Returns:  Returns an array containing the personID of     #
#           the close matches.                              #
# Requires: nothing                                         #
# Modifies: nothing                                         #
#############################################################
{
    $first = $person->getFirstName();
    $last = $person->getLastName();
    $email = $person->getEmail();
    $emailMatches = array();
    $firstMatches = array();
    $lastMatches = array();

    if ($includeEmail)
    {
        $query =  "SELECT peopleid ";
        $query .=    "FROM people_contact ";
        $query .=    "WHERE emailday='$email';";
        $result = $this->query($query);
        $count = $result->numRows();
        for ($i=0; $i<$count; $i++)
        {
            list ($temp) = $result->getRowAt($i);
            array_push($emailMatches, $temp);
        }
    }

    $temp = substr($last,0,2);
#   echo $temp;
    $query = "SELECT peopleid ";
    $query .= "FROM people ";
    $query .= "WHERE firstname='$first' ";
    $query .= "AND lastname LIKE '$temp%';";
    $result = $this->query($query);
    $count = $result->numRows();
    for ($i=0; $i<$count; $i++)
    {
        list ($temp) = $result->getRowAt($i);
        array_push($firstMatches, $temp);
    }

    $temp = substr($first,0,2);
#   echo $temp;
    $query = "SELECT peopleid ";
```

```
        $query .=    "FROM people ";
        $query .=    "WHERE lastname='$last' ";
        $query .=    "AND firstname LIKE '$temp%';";
        $result = $this->query($query);
        $count = $result->numRows();
        for ($i=0; $i < $count; $i++)
        {
            list ($temp) = $result->getRowAt($i);
            array_push($lastMatches, $temp);
        }
        if ($exact)
        {
            $people = array_intersect($emailMatches,
                                      $firstMatches, $lastMatches);

        }
        else
            $people = array_unique(array_merge($emailMatches,
                                               $firstMatches,
                                               $lastMatches));
        return $people;
    }

function nameCollisionTable($people, $formVars, $target, $prefix="")
#####################################################################
# Function: nameCollisionTable                                      #
# Arguments: people --- the list of colliding IDs                   #
#            formVars --- the entered information                   #
#            target --- where to send the user's choice             #
#            prefix -- to prepend to the var in formVars            #
    # Returns: A table/form string that gives the name collisions    #
    #          to the user for user clarification.                   #
    # Requires: nothing                                              #
    # Modifies: nothing                                              #
    ##################################################################
    {
        $first = $formVars[$prefix."FirstName"];
        $last = $formVars[$prefix."LastName"];
        $email = $formVars[$prefix."Email"];
        $affil = $formVars[$prefix."Affil"];
        $newAffil = $formVars[$prefix."NewAffil"];
        if ($affil!="" && $affil!=-1)
        {
            $org= new Organization($this, $affil);
            $org->retrieve();
            $affil=$org->getAbbrev();
        }
        $affilTemp = ($affil == "" ? $newAffil : $affil);

        $result  ="<TABLE>\n";
        $result .= "\t<TR>\n";
        $result .= "\t\t<TH>Status</TH><TH>Name</TH><TH>Email</TH>\n";
        $result .= "\t\t<TH>Affiliation</TH><TH>Partners</TH><TH>Options</TH>\n";
        $result .= "\t</TR>\n";
        $result .= "\t<TR>\n";
        $result .= "\t\t<TD>Just Entered</TD><TD>$first $last</TD>
<TD>$email</TD><TD>$affilTemp</TD>\n";
        $result .= "\t\t<TD>None entered yet</TD>\n";
        $result .= "\t\t<TD><FORM ACTION=$target METHOD=POST>\n";
        $result .= "\t\t\t<INPUT TYPE=HIDDEN NAME=CHOICE VALUE=-1>\n";
        $result .= "\t\t\t<INPUT TYPE=SUBMIT NAME=USE VALUE=\"Use Me\"
```

```
style=\"background: #d9a\"></FORM></TD>\n";
        $result .= "\t</TR>\n";
        reset($people);
        while ($aPerson = each($people))
        {
            $person = new Competitor($this);
            $person->setID($aPerson['value']);
            $person->retrieve();
#           echo "<br>CompDB::nameCollision:".printArray($person->fields);
            $aPersonID=$person->getID();
            $partners = $person->getPartners();
            $first = $person->getFirstName();
            $last = $person->getLastName();
            $email = $person->getEmail();
            $affilID = $person->getAffilID();
            $org = new Organization($this,$affilID);
            $org->retrieve();
            $affil = $org->getAbbrev();
            $result .= "\t<TR>\n";
            $result .= "\t\t<TD>From Database</TD><TD>$first $last</TD>
<TD>$email</TD><TD>$affil</TD>\n";
            $result .= "\t\t<TD> ";
            $numPartners = count($partners);
            if ($numPartners==0)
            {
                $result .= "None entered yet";
            }
            for ($j=0;$j<$numPartners;$j++)
            {
                $aPartner = new Person($this);
                $aPartner->setID($partners[$j]);
                $aPartner->retrieve();
                $partnerFirst = $aPartner->getFirstName();
                $partnerLast = $aPartner->getLastName();
                $result .= "$partnerFirst $partnerLast <BR>";
            }
            $result .= "\t\t</TD>\n";
            $result .= "\t\t<TD><FORM ACTION=$target METHOD=POST>\n";
            $result .= "\t\t\t<INPUT TYPE=HIDDEN NAME=CHOICE VALUE=$aPersonID>\n";
            $result .= "\t\t\t<INPUT TYPE=SUBMIT NAME=USE VALUE=\"Use Me\"
style=\"background: #d9a\">\n";
            $result .= "\t\t</FORM></TD>\n";
            $result .= "\t</TR>\n";
        }
        $result .= "</TABLE>\n";
        return $result;
    }

    function getNextTBA()
    {
        if (isset($this->lastUsed))
            $this->lastUsed++;
        else
        {
            $query = "SELECT MAX(to_number(lastname,'00000009')) FROM
people WHERE firstname='TBA'";
            $result = $this->query($query);
            list($this->lastUsed) = $result->getRowAt(0);
            $this->lastUsed++;
        }
        return $this->lastUsed;
```

```
    }

    function rebuildStats()
    {
#       $this->rebuildSummary();
        ignore_user_abort(1);
        set_time_limit(180);
        $this->rebuildEvents();
        $this->rebuildSchools();
        $this->rebuildPeople("leader");
        $this->rebuildPeople("follower");
        $this->rebuildCategories();
        ignore_user_abort(0);
        return 0;
    }

    function getStatsHeader($title)
    {
        global $CIB_BASE_URL,$unixname;
        $display = new HTMLDisplay($this);
        $temp = $display->beginPage("",$title);


        $temp .= "<table bgcolor=\"white\" border=\"1\ width=\"95%\"
cellpadding=\"8\">\n";
        $temp .= "<tr><td>\n";

        $temp .= "<div id=\"NavBar\" align=\"center\">\n";
        $temp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/events.html\">
by Event</a> | \n";
        $temp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/summary.html\">
by Summary</a> | \n";
        $temp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/affils.html\">
by Affiliation</a> |\n ";
        $temp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/leaders.html\">
by Leaders</a> |\n ";
        $temp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/followers.html\">
by Followers</a>\n";
        $temp .= "<br />" . $this->getStatsSummary();
        $temp .= "</div>\n</td>\n</tr>\n</table>";
        $temp .= "<br>\n";
        $temp .= "<table bgcolor=\"white\" border=\"1\" width=\"95%\"
cellpadding=\"8\">\n";
        $temp .= "<tr><td>\n";
        $temp .= "<h3>$title</h3>\n";
        return $temp;
    }

    function getStatsSummary()
    {
        $query = "SELECT count(*) FROM events_registration;";
        $result = $this->query($query);
        list($entries)=$result->getRowAt(0);
        $query = "SELECT count(*) FROM couples;";
        $result = $this->query($query);
        list($couples)=$result->getRowAt(0);
        $query = "SELECT count(distinct orgid) FROM organizations, people, couples ";
        $query .=       "WHERE organizations.orgid=people.organization AND ";
        $query .=             "(peopleid=follower OR peopleid=leader);";
        $result = $this->query($query);
        list($affiliations)=$result->getRowAt(0);
```

```
        $temp = "$entries Entries, $couples Couples, $affiliations Affiliations\n";
        return $temp;
    }

    function getStatsFooter()
    {
        GLOBAL $baseURL;
        $temp .= "</td></tr></table><br \>\n";
        $temp .= HTMLDisplay::compinaboxBox();
        $temp .= HTMLDisplay::endPage();
        return $temp;
    }

    function getStatsEventListing($id, $fullname)
    {
        $query  = "SELECT l.peopleid, l.firstname, l.lastname, lo.name, ";
        $query .=       "f.peopleid, f.firstname, f.lastname, fo.name ";
        $query .=   "FROM events_registration NATURAL JOIN couples ";
        $query .=                   "JOIN people AS l ON (l.peopleid=leader) ";
        $query .=                   "JOIN people AS f ON (f.peopleid=follower) ";
        $query .=         "LEFT OUTER JOIN organizations AS lo ON
(l.organization=lo.orgid) ";
        $query .=         "LEFT OUTER JOIN organizations AS fo ON
(f.organization=fo.orgid) ";
        $query .=  "WHERE eventid='$id' ";
        $query .=  "ORDER BY l.lastname, l.firstname, f.lastname, f.firstname;";
#       echo $query;
        $result = $this->query($query);
        $numRows = $result->numRows();
        if (!$numRows)
            return "No couples registered.";
        $temp = "$numRows Couples Registered for <b>$fullname</b>
<br>\n<table><tr><th>Leader</th><th>Follower</th></tr>\n";
        for ($i=0;$i<$numRows;$i++)
        {
            list($leaderid, $leaderFirst, $leaderLast, $leaderAffil,
                 $followid, $followFirst,
                 $followLast, $followAffil) = $result->getRowAt($i);
            $temp .= "<tr><td>$leaderFirst $leaderLast ($leaderAffil)</td>
<td>$followFirst $followLast ($followAffil)</td></tr>\n";
        }
        $temp .= "</table>";
        return $temp;
    }

    function getStatsAffilListing($id, $fullname)
    {
#       echo ":::$id ---------------$fullname<br>";
        if ($id!="")
        {
            $query  = "SELECT ldr.peopleid, ldr.firstname, ldr.lastname, ldr.name, ";
            $query .=       "flw.peopleid, flw.firstname, flw.lastname,
flw.name, coupleid ";
            $query .=       "FROM (people LEFT JOIN organizations
ON organization=orgid) AS ldr, ";
            $query .=            "(people LEFT JOIN organizations
ON organization=orgid) AS flw, ";
            $query .=            "couples ";
            $query .=       "WHERE ldr.peopleid=leader and
flw.peopleid=follower AND ";
```

```
            $query .=               "(ldr.orgid='$id' OR flw.orgid='$id') ";
            $query .=         "ORDER BY ldr.lastname, ldr.firstname,
flw.lastname, flw.firstname;";
        }
        else
        {
            $query  = "SELECT ldr.peopleid, ldr.firstname, ldr.lastname, ldr.name, ";
            $query .=         "flw.peopleid, flw.firstname, flw.lastname,
flw.name, coupleid ";
            $query .=         "FROM (people LEFT JOIN organizations ON
organization=orgid) AS ldr, ";
            $query .=               "(people LEFT JOIN organizations ON
organization=orgid) AS flw, ";
            $query .=               "couples ";
            $query .=         "WHERE ldr.peopleid=leader and flw.peopleid=
follower AND ";
            $query .=                "(ldr.orgid IS NULL OR flw.orgid IS NULL) ";
            $query .=         "ORDER BY ldr.lastname, ldr.firstname,
flw.lastname, flw.firstname;";
        }
#       echo "<br>".$query."<BR>";
        $result = $this->query($query);
        $numRows = $result->numRows();
        if (!$numRows)
            return "No couples registered.";
        $temp = "$fullname is represented by $numRows Couples<br>\n<table>
<tr><th>Leader</th><th>Follower</th><th>Events</th></tr>\n";
        for ($i=0;$i<$numRows;$i++)
        {
            list($leaderid, $leaderFirst, $leaderLast, $leaderAffil,
                $followid, $followFirst,
$followLast, $followAffil,$coupleid) = $result->getRowAt($i);
            if ($leaderAffil=="$fullname") $leaderAffil="";
            else $leaderAffil="($leaderAffil)";
            if ($followAffil=="$fullname") $followAffil="";
            else $followAffil="($followAffil)";
            $couple= new Couple($this);
            $couple->setID($coupleid);
            $couple->retrieve();
            $display = new HTMLDisplay($this);
            $events = $display->colorDances($couple->getDances());
            $temp .= "<tr><td>$leaderFirst $leaderLast $leaderAffil</td>
<td>$followFirst $followLast $followAffil</td><td>$events</td></tr>\n";
        }
        $temp .= "</table>";
        return $temp;
    }

    function rebuildPeople($role)
    {
        global $CIB_WEB_PATH,$unixname;
        global $CIB_BASE_URL;
        global $peoplePerPage;
        $PEOPLE_PER_PAGE=10;
        $display = new HTMLDisplay($this);

        $Roles = ($role=="leader" ? "Leaders" : "Followers");
        $Role = substr($Roles,0,strlen($Roles)-1);
        $otherRole = ($role == "leader" ? "Follower" : "Leader");
        $otherrole = ($role == "leader" ? "follower" : "leader");
        $query  = "SELECT DISTINCT coupleid, firstname, lastname, name ";
```

```
        $query .=    "FROM couples ";
        $query .=    "JOIN people ON $role=peopleid ";
        $query .=    "LEFT OUTER JOIN organizations ON organization=orgid ";
        $query .=    "NATURAL JOIN events_registration  ";
        $query .=    "ORDER BY lastname , firstname , name;";
        $result = $this->query($query);
        $alphabet=array("0","1","2","3","4","5","6","7","8","9",
"A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S",
"T","U","V","W","X","Y","Z"," ");
        $alphaCount=array("0"=>0,"1"=>0,"2"=>0,"3"=>0,"4"=>0,
                          "5"=>0,"6"=>0,"7"=>0,"8"=>0,"9"=>0,
                          "A"=>0,"B"=>0,"C"=>0,"D"=>0,"E"=>0,"F"=>0,
                          "G"=>0,"H"=>0,"I"=>0,"J"=>0,"K"=>0,"L"=>0,
                          "M"=>0,"N"=>0,"O"=>0,"P"=>0,"Q"=>0,"R"=>0,
                          "S"=>0,"T"=>0,"U"=>0,"V"=>0,"W"=>0,"X"=>0,
                          "Y"=>0,"Z"=>0);
        $uncountedCount=0;
        $numLines = $result->numRows();
        for ($i=0;$i<$numLines;$i++)
        {
            list($dum1, $dum2, $lastname, $dum3) = $result->getRowAt($i);
            $temp = strtoupper(substr($lastname,0,1));
            if (($temp >= 'A' && $temp <= 'Z') ||
                ($temp >= "0" && $temp <= "9")
)
                $alphaCount["$temp"]++;
            else
                $uncountedCount++;
        }
        $blocks=array();
        $beginning=0;
        $runningTally=0;
        for ($i=0;$i<=35;$i++)
        {
            if (($runningTally+$alphaCount[$alphabet[$i]]) < $PEOPLE_PER_PAGE)
            {
                $runningTally+=$alphaCount[$alphabet[$i]];
                $end=$i;
            }
            else if ($runningTally >($PEOPLE_PER_PAGE/2))
            {
                array_push($blocks,array($alphabet[$beginning],$alphabet[$end]));
                $beginning = ++$end;
                $end=$beginning;
                $runningTally=$alphaCount[$alphabet[$i]];
            }
            else
            {
                $end=$i;
                array_push($blocks,array($alphabet[$beginning],$alphabet[$end]));
                $beginning = $end+1;
                $end=$beginning;;
                $runningTally=0;
            }
        }
        $dummyTemp = $alphaCount[" "];
        if (($runningTally+$alphaCount[" "]) < $PEOPLE_PER_PAGE)
        {
            array_push($blocks,array($alphabet[$beginning],"Z"));
        }
        else if ($runningTally >($PEOPLE_PER_PAGE/2))
```

```
        {
            array_push($blocks,array($alphabet[$beginning],"Z"));
        }
        else
        {
            array_push($blocks,array($alphabet[$beginning],$alphabet[$end]));
            array_push($blocks,array("Z","Z"));
        }
        $numBlocks=count($blocks);
        $tableHeadings = "<tr><th>$role</th><th>$otherRole</th><th></th></tr>";
        $indexBody  = $this->getStatsHeader("Stats: by $Role");
        $fullBody   = $this->getStatsHeader("Stats: by $Role");

        $indexBody .= "<a href=
\"$CIB_BASE_URL/stats/$unixname/$Roles/$role-full.html\">
Full Listing (may be quite large)</a><br>\n";
        $indexBody .= "Or view the breakout pages with ~$PEOPLE_PER_PAGE
couples per page.<br><br>";
        $statsNavRel ="";
        $statsNavAbs ="";
        for ($i=0;$i<$numBlocks;$i++)
        {
            $block0= $blocks[$i][0]; $block1=$blocks[$i][1];
            $blockBegin = ($block0>="0" && $block0<="9") ? "TBA's" : $block0;
            $blockEnd   = ($block1>="0" && $block1<="9") ? "TBA's" : $block1;
            $blockName  = ($blockBegin==$blockEnd ? $blockBegin :
"$blockBegin-$blockEnd");
            $blockNames[$i]=$blockName;
            $indexBody    .= "<a href=
\"$CIB_BASE_URL/stats/$unixname/$Roles/$role-$i.html\">$blockName</a><br>\n";
            $statsNavRel .= "<a href=\"#$blockName\">$blockName</a>";
            $statsNavAbs .= "<a href=
\"$CIB_BASE_URL/stats/$unixname/$Roles/$role-$i.html\">$blockName</a>";
            $commaNeeded = ($i != ($numBlocks-1));
            if ($commaNeeded)
            {
                $statsNavRel .= ", ";
                $statsNavAbs .= ", ";
            }
        }
        $indexBody .= $this->getStatsFooter();
        $fp = fopen ("$CIB_WEB_PATH/stats/$unixname/$role"."s"."."html", "w");
        if (!fwrite($fp,$indexBody))
        {
            mail($onErrorMail,"OI ERROR: Stats","Could not write to $role.html");
        }
        fclose($fp);
        $statsNavRel .= "<br>";
        $statsNavAbs .= "<br>";
        $blockIndex = 0;
        $fullBody .= "<a name=\"$blockNames[$blockIndex]\">
<h3>$blockNames[$blockIndex]</h3></a><br>\n";
        $fullBody .= $display->colorLegend();
        $fullBody .= "<table>\n";
        $fullBody .= $tableHeadings;
        $breakoutBody  = $this->getStatsHeader("Stats: by $Role: $blockNames[$blockIndex]");
        $breakoutBody .= $statsNavAbs;
        $breakoutBody .= $display->colorLegend();
        $breakoutBody .= "<table>\n";
        $breakoutBody .= $tableHeadings;
        $procName = "get" . $otherRole;
```

```
        for ($i=0;$i<$numLines;$i++)
        {
            list($coupleID, $primeFirst, $primeLast, $primeOrg) = $result->getRowAt($i);
            $temp = strtoupper(substr($primeLast,0,1));
            if ($temp > $blocks[$blockIndex][1])
            {
                $filename = $role . "-" . $blockNames[$blockIndex];
                $breakoutBody .= "</table>";
                $breakoutBody .= $this->getStatsFooter();
                $fp= fopen(
"$CIB_WEB_PATH/stats/$unixname/$Roles/$role-$blockIndex.html","w");
                if (!fwrite($fp,$breakoutBody))
                {
                    mail($onErrorMail,"OI ERROR: Stats",
"Could not write to $role-$blockNames[$blockIndex].html");
                }
                fclose($fp);
                $blockIndex++;
                $breakoutBody  = $this->getStatsHeader(
"Stats: by $Role: $blockNames[$blockIndex]");
                $breakoutBody .= $statsNavAbs;
                $breakoutBody .= $display->colorLegend();
                $breakoutBody .= "<table>\n";
                $breakoutBody .= $tableHeadings;
                $fullBody .= "</table>\n";
                $fullBody .= $statsNavRel;
                $fullBody .= "<a name=\"$blockNames[$blockIndex]\">
<h3>$blockNames[$blockIndex]</h3></a><br>\n";
                $fullBody .= $display->colorLegend();
                $fullBody .= "<table>\n";
                $fullBody .= $tableHeadings;
            }
            $couple = new Couple($this, $coupleID);
            $couple->setID($coupleID);
            $couple->retrieve();
            $partnerID = $couple->$procName();
            $partner = new Person($this, $partnerID);
            $partner->setID($partnerID);
            $partner->retrieve();
            $partnerFirst = $partner->getFirstName();
            $partnerLast = $partner->getLastName();
            $partnerOrg = ($partner->getAffilID() ?
$partner->getAffil() : "none");
            $events = $display->colorDances($couple->getDances());
            $coupleRow  = "<tr>\n\t<td>$primeLast,
$primeFirst ($primeOrg)</td>\n";
            $coupleRow .=       "\t<td>$partnerLast,
$partnerFirst ($partnerOrg)</td>\n";
            $coupleRow .=       "\t<td>$events</td></tr>\n";
            $breakoutBody .= $coupleRow;
            $fullBody .= $coupleRow;
        }
        $breakoutBody .= "</table>\n";
        $breakoutBody .= $this->getStatsFooter();
        $fp= fopen(
"$CIB_WEB_PATH/stats/$unixname/$Roles/$role-$blockIndex.html","w");
        if (!fwrite($fp,$breakoutBody))
        {
            mail($onErrorMail,"OI ERROR: Stats",
"Could not write to $role-$blockIndex.html");
        }
```

```
        fclose($fp);
        $fullBody .= "</table>\n";
        $fullBody .= $this->getStatsFooter();
        $fp= fopen("$CIB_WEB_PATH/stats/$unixname/$Roles/$role-full.html","w");
        if (!fwrite($fp,$fullBody))
        {
            mail($onErrorMail,"OI ERROR: Stats","Could not write to $role-full.html");
        }
        fclose($fp);

    }

    function rebuildSchoolInclude()
    {
        GLOBAL $CIB_WEB_PATH,$unixname;
        $includeBody = "<?php\n";
        $includeBody .= '$incAffils["No_Affil"]=-1;'."\n";
        $query  = "SELECT DISTINCT orgid, name, abbreviation ";
        $query .=   "FROM organizations ";
        $query .=   "LEFT OUTER JOIN abbreviations ON (name=fulltext) ";
        $query .=   "WHERE orgid>1 ";
        $query .=   "ORDER BY abbreviation;";

        $result = $this->query($query);
        $numAffils = $result->numRows();
        for ($i=0;$i < $numAffils;$i++)
        {
            list($id,$name,$abbreviation) = $result->getRowAt($i);
            $abbreviation = ($abbreviation=="" ?
($name=="" ? "none" : $name) : $abbreviation);
            if ($id=="")
                $tid=-1;
            else
                $tid=$id;
            $includeBody .= '$incAffils["'.$abbreviation.'"]='.$tid.";\n";
        }
        $includeBody .= "?>";
        $fp = fopen("$CIB_WEB_PATH/stats/$unixname/affils.inc","w");
        fwrite($fp,$includeBody);
        fclose($fp);
    }


    function rebuildSchools()
    {
        global $CIB_WEB_PATH,$unixname;
        global $CIB_BASE_URL;
        #first we need to make the master affiliation page
        $this->rebuildSchoolInclude();
        $filebody = $this->getStatsHeader("Stats: by Affiliations");
        $filebody .= "<h3>Number of Competitors from each Affiliation</h3>\n";
        $query  = "SELECT DISTINCT orgid, name, abbreviation ";
        $query .=   "FROM organizations ";
        $query .=   "LEFT OUTER JOIN abbreviations ON (name=fulltext) ";
        $query .=   "RIGHT OUTER JOIN people ON (orgid=organization) ";
        $query .=   "JOIN couples ON (leader=peopleid OR follower=peopleid) ";
        $query .=   "ORDER BY name;";

        $result = $this->query($query);
        $numAffils = $result->numRows();
        $includedAffils=array();
```

```
        $includedText = "";
        for ($i=0;$i<$numAffils;$i++)
        {
            list($id,$name,$abbreviation) = $result->getRowAt($i);
            $abbreviation = ($abbreviation=="" ?
($name=="" ? "none" : $name) : $abbreviation);
            $name = ($name=="" ? "No Affil" : $name);
            array_push($includedAffils,array($id,$name,$abbreviation));
            if ($id)
            {
                $query = "SELECT COUNT(DISTINCT peopleid) FROM people, couples ";
                $query .=       "WHERE (leader=peopleid OR follower=peopleid)
AND organization='$id';";
            }
            else
            {
                $query = "SELECT COUNT(DISTINCT peopleid) FROM people, couples ";
                $query .=       "WHERE (leader=peopleid OR follower=peopleid)
AND (organization IS NULL OR organization=-1);";
            }
            $countResult = $this->query($query);
            list($num)= $countResult->getRowAt(0);
            $includedText .= "
<a href=\"$CIB_BASE_URL/stats/$unixname/Affils/$abbreviation.html\">$name</a>:
$num<br \>\n";
        }
        $filebody .= $includedText;
        $filebody .= $this->getStatsFooter();
        $fp = fopen ("$CIB_WEB_PATH/stats/$unixname/affils.html", "w");
        if (!fwrite($fp,$filebody))
        {
            mail($onErrorMail,"OI ERROR:Stats","Could not write to affils.html");
        }
        # now we need to make the drill down pages
        #this is all done using included code to isolate the comp specifics
#       echo printArray($includedAffils);
        reset($includedAffils);
        while ($aTempAffil = each($includedAffils))
        {
            $anAffil = $aTempAffil['value'];
            $affilText = $this->getStatsHeader($anAffil[1]);

            $affilText .= $this->getStatsAffilListing($anAffil[0],$anAffil[1]);
            $affilText .= $this->getStatsFooter();
            $filename = $anAffil[2];
            $fp = fopen(
"$CIB_WEB_PATH/stats/$unixname/Affils/" . $filename . ".html", "w");
            fwrite($fp,$affilText);
            fclose($fp);
        }

        return 0;
    }


    function rebuildEvents()
    {
        global $CIB_WEB_PATH,$unixname;
        global $CIB_BASE_URL;
        #first we need to make the master events page
        $filebody = $this->getStatsHeader("Stats: by Events");
```

```
        include ("compDB__rebuildEvents__eventsSummary.inc");
        $filebody .= $includedTemp;
        $filebody .= $this->getStatsFooter();
        $fp = fopen ("$CIB_WEB_PATH/stats/$unixname/events.html", "w");
        if (!fwrite($fp,$filebody))
        {
            mail($onErrorMail,"OI ERROR: Stats","Could not write to events.html");
        }
        # now we need to make the drill down pages
        #this is all done using included code to isolate the comp specifics
        include ("compDB__rebuildEvents__eventsList.inc");
        reset($includedEvents);
        while ($aTempEvent = each($includedEvents))
        {
            $anEvent = $aTempEvent['value'];
            $eventText = $this->getStatsHeader($anEvent[2]);
            $eventText .= $this->getStatsEventListing($anEvent[0],$anEvent[2]);
            $eventText .= $this->getStatsFooter();
            $filename = $anEvent[1];
            $fp = fopen(
"$CIB_WEB_PATH/stats/$unixname/Events/" . $filename . ".html", "w");
            fwrite($fp,$eventText);
            fclose($fp);
        }

        return 0;
    }

    function rebuildCategories()
    {
        global $CIB_WEB_PATH,$unixname;
        global $CIB_BASE_URL;
        global $SD_background_color;
        $query = "SELECT level, style FROM events_categories join levels on
(level=levels.name) join styles on (style=styles.name) WHERE
events_categories.type IS NULL order by levelorder,styleorder;";
        $result = $this->query($query);
        $numCats = $result->numrows();
        $filebody = $this->getStatsHeader("Stats: Summary");
        $filebody .= "<ul>";
        $incbody = "There are currently ". $this->getNumCouples();
        $incbody .=" couples (".$this->getNumCompetitors()." people) ";
        $incbody .= "registered from ". $this->getNumAffiliations();
        $incbody .=" schools, studios,\n and independents.\n ";
        $incbody .= "The average number of entries, grouped by style and level,
are shown below.\n";
        $incbody .= "<center><table border=\"1\" cellspacing=\"0\"
cellpadding=\"2\">\n";
        $incbody .= "\t<tr>\n\t\t<th><font size=\"-1\">Level</font></th>\n
<th><font size=\"-1\">  Overall  </font></th>\n";
        $subQuery = "SELECT DISTINCT style,styleorder FROM events_categories ";
        $subQuery.= "JOIN styles ON (style=styles.name) ";
        $subQuery.= "WHERE events_categories.type IS NULL ORDER BY styleorder;";
        $subResult = $this->query($subQuery);
        $styles = array();
        $numStyles=$subResult->numrows();
        for ($i=0;$i<$numStyles;$i++)
        {
            list($style)=$subResult->getRowAt($i);
            $styles[$i]=$style;
            $incbody .= "\t\t<th><font size=\"-1\">  $style  </font>
```

```
</th>\n";
        }
        $incbody .= "\t</tr>\n";
        $numStyles=count($styles);
        $styleCounter=0;
        $curLevel="";
        $first=TRUE;
        for ($i=0;$i <$numCats;$i++)
        {
            list($level, $style) = $result->getRowAt($i);
            $levelLink=str_replace("/","-",$level);
            $this->rebuildCategorySummary($style,$level);
            $numComp = round($this->getAvgEntries($level,$style),1);
            if ($curLevel!=$level)
            {
                if (!$first)
                  {
                    while($numStyles>$styleCounter)
                      {
                        $incbody .=
"\t\t<td bgcolor=\"$SD_background_color\"> </td>\n";
                        $styleCounter++;
                      }
                    $styleCounter =0;
                    $incbody .= "\t</tr>\n";
                    $filebody .= "<br />\n";
                }

                $incbody .= "\t<tr>\n\t\t<th><font size=\"-1\"> 
$level  </font></th>\n";
                $incbody .= "\t\t<td><font size=\"-1\"><center>" .
round($this->getAvgEntries($level),1) . "</center></font></td>\n";
                $curLevel=$level;
            }
            $filebody .= "<li><a href=\"$CIB_BASE_URL/stats/$unixname/Summary/".
"$levelLink-$style.html\">$level $style</a> ($numComp Couples)</li>\n";

            while($styles[$styleCounter]!=$style)
              {
                $incbody .= "\t\t<td bgcolor=\"$SD_background_color\"> </td>\n";
                $styleCounter++;
              }
            $incbody .= "\t\t<td><font size=\"-1\"><center>
<a href=\"$CIB_BASE_URL/stats/$unixname/Summary/$levelLink-$style.html\">
$numComp</a></center></font></td>\n";
            $styleCounter++;
            $first=FALSE;
        }
        if (!$first)
          {
            while ($numStyles>$styleCounter)
            {
                $incbody .= "\t\t<td bgcolor=\"$SD_background_color\"> </td>\n";
                $styleCounter++;
            }
            $incbody .= "\t</tr>\n";
          }
        $incbody .= "</table></center>";
        $filebody .= "</ul>";
        $filebody .= $this->getStatsFooter();
        $fp = fopen("$CIB_WEB_PATH/stats/$unixname/summary.html","w");
```

```
        fwrite($fp,$filebody);
        fclose($fp);
        $fp = fopen("$CIB_WEB_PATH/stats/$unixname/summary.inc","w");
        fwrite($fp,$incbody);
        fclose($fp);
        return 0;
    }

    function rebuildCategorySummary($style,$level)
    {
        global $CIB_WEB_PATH,$unixname;
        global $CIB_BASE_URL;
        global $SD_background_color;
        $levelLink=str_replace("/","-",$level);
        #first we need to make the master events page
        $filebody = $this->getStatsHeader("Stats: Summary of $level $style");

        $query = "Select eventid, shortname from events_categories natural
join events natural join events_names natural join events_dances join styles
on (style=styles.name) join styles_dances on (styles.styleid=
styles_dances.styleid AND events_dances.dancename=styles_dances.dancename)
where level='$level' and style='$style' and events_dances.danceorder=1 order
by styles_dances.danceorder;";
        $result = $this->query($query);
        $numDances = $result->numrows();
        $filebody .= "<table><tr><th>Couple</th>";
        for ($i=0;$i<$numDances;$i++)
        {
            list($eventid,$eventName) = $result->getRowAt($i);
            $eventShort= substr($eventName,strrpos($eventName,'-')+1,
strlen($eventName));
//          $eventName=strtolower($eventName);
            $filebody .= "<th> 
<a href=\"$CIB_BASE_URL/stats/$unixname/Events/".$eventName.".html\">
$eventShort</a>  </th>";
            $idToOrder[$eventid]=$i;
        }
        $filebody .= "</tr>\n";
        $query = "SELECT distinct l.firstname, l.lastname, f.firstname,
f.lastname, coupleid, la.abbreviation, fa.abbreviation FROM people AS l
JOIN couples ON (l.peopleid=leader) JOIN people AS f ON (f.peopleid=follower)
NATURAL JOIN events_registration NATURAL JOIN events NATURAL JOIN
events_categories LEFT OUTER JOIN organizations AS lo ON
(l.organization=lo.orgid) LEFT OUTER JOIN abbreviations AS la ON
(lo.name=la.fulltext) LEFT OUTER JOIN organizations AS fo ON
(f.organization=fo.orgid) LEFT OUTER JOIN abbreviations AS fa ON
(fo.name=fa.fulltext) WHERE level='$level' and style='$style' ORDER BY
l.lastname, l.firstname;";
        $result=$this->query($query);
        $numCouples = $result->numrows();
        for ($i=0;$i<$numCouples;$i++)
        {
            list($lFirst, $lLast, $fFirst,$fLast,
                $coupleid, $lAffil, $fAffil) =
                $result->getRowAt($i);
            $filebody .= "<tr>";
            $filebody .= "<td>$lFirst $lLast ($lAffil) &amp; $fFirst $fLast
($fAffil) </td>\n";
            $query = "SELECT events.eventid FROM couples, events_registration,
events, events_categories WHERE couples.coupleid=$coupleid and couples.coupleid
 = events_registration.coupleid and events_registration.eventid=events.eventid
```

```
and events.categoryid=events_categories.categoryid and level='$level' and
style='$style';";
            $subResult = $this->query($query);
            $numEntries = $subResult->numrows();
            for ($j=0;$j<$numDances;$j++)
                $entries[$j]=FALSE;
            for ($j=0;$j<$numEntries;$j++)
            {
                list($eventid) = $subResult->getRowAt($j);
                $entries[$idToOrder[$eventid]]=TRUE;
            }
            $curFlag=$entries[0];
            $j=1;
            $colSpan=1;
            while ($j<$numDances)
            {
                if ($entries[$j]!=$curFlag)
                {
                    if ($curFlag)
                        $filebody .= "<td colspan=\"$colSpan\"
bgcolor=\"$SD_background_color\"> </td>";
                    else
                        $filebody .= "<td colspan=\"$colSpan\"
bgcolor=\"white\"> </td>";
                    $colSpan=1;
                    $curFlag=!$curFlag;
                }
                else
                    $colSpan++;
                $j++;
            }
            if ($curFlag)
                $filebody .= "<td colspan=\"$colSpan\"
bgcolor=\"$SD_background_color\"> </td>";
            else
                $filebody .= "<td colspan=\"$colSpan\"
bgcolor=\"white\"> </td>";
            $filebody .= "</tr>";
        }
        $filebody .= "</table>";

        $filebody .= $this->getStatsFooter();
        $fp = fopen (
"$CIB_WEB_PATH/stats/$unixname/Summary/$levelLink-$style.html", "w");
        if (!fwrite($fp,$filebody))
        {
            mail($onErrorMail,"OI ERROR: Stats",
"Could not write to $level-$style.html");
        }
        return 0;
    }


    function assignCompNumbers($method, $starting=100, $ending=999)
    {
        switch($method) {
            case "Alpha" : return $this->assignCompNumbersAlpha(
$starting,$ending); break;
            case "AlphaWithinAffil" : return $this->assignCompNumbersAffil(
$starting,$ending); break;
            default : return 0; break;
```

```
        }
    }

    function assignCompNumbersAlpha($starting=100, $ending=999)
    {
        $query = "SELECT DISTINCT leader FROM couples NATURAL JOIN ";
        $query .= "events_registration; ";
        $result = $this->query($query);
        $numNumbers = $result->numrows();
        if (($ending-$starting+1) > $numNumbers) return 0;
        $query = "SELECT DISTINCT leader, firstname, lastname FROM people ";
        $query .= "JOIN couples ON (peopleid=leader) NATURAL JOIN ";
        $query .= "events_registration ";
        $query .= "ORDER BY lastname, firstname;";
        $result = $this->query($query);
        $j=$starting;
        for ($i=0;$i<$numNumbers;$i++)
        {
            list($curLeader, ,$dum1, $dum2) = $result->getRowAt($i);
            $query = "UPDATE events_registration SET competitornumber='";
            $query .= $j+$i;
            $query .= "' WHERE leader=$curLeader;";
            $this->query($query);
        }
        return 1;
    }

    function assignCompNumbersAffil($starting=100, $ending=999)
    {
        $query = "SELECT DISTINCT leader FROM couples NATURAL JOIN ";
        $query .= "events_registration; ";
        $result = $this->query($query);
        $numNumbers = $result->numrows();
        if (($ending-$starting+1) > $numNumbers) return 0;

        $query = "SELECT organization, COUNT(organization) FROM people JOIN ";
        $query .= "couples ON (peopleid=leader) NATURAL JOIN ";
        $query .= "events_registration GROUP BY organization ";
        $query .= "ORDER BY count DESC;";
        $result = $this->query($query);
        $numAffils = $result->numrows();
        $baseNum = ceil($starting/10);
        $affils = array();
        for ($i=0;$i<$numAffils;$i++)
        {
           #populate the affil structure
        }
        $query = "SELECT DISTINCT leader, firstname, lastname, organization FROM people ";
        $query .= "JOIN couples ON (peopleid=leader) NATURAL JOIN ";
        $query .= "events_registration ";
        $query .= "ORDER BY organization, lastname, firstname;";
        $result = $this->query($query);
        for ($i=0;$i<$numNumbers;$i++)
        {
            list($curLeader, ,$dum1, $dum2, $curOrg) = $result->getRowAt($i);
            $query = "UPDATE events_registration SET competitornumber='";

            $numberToAssign = $baseNum + $affils[$curOrg]["NumAssigned"];
            $numberToAssign = $numberToAssign . $affils[$curOrg]["Suffix"];
            $affils[$curOrg]["numAssigned"]++;
```

```
            $query .= " WHERE leader=$curLeader;";
            $this->query($query);
        }
        return 1;
    }


function produceRegistrantArray($affilID)
/*****************************************************************
* Function: produceRegistrantArray                               *
* Arguments: affilID -- what school are we interested in         *
* Returns: An array broken up by packages, and sponsored         *
*          sponsored status for use in creating web/pdf          *
*          invoices.                                             *
* Requires: nothing                                              *
* Modifies: nothing                                              *
*****************************************************************/
{
$query =<<<END_QUERY
SELECT packages.packageid, name, category, cost
FROM packages NATURAL JOIN packages_cost
ORDER BY cost, category
END_QUERY;
$result = $this->query($query);
$numPackages = $result->numRows();
$packagesPurchased = array();
for ($i=0;$i < $numPackages;$i++)
  {
    list($id, $name, $cat, $cost) = $result->getRowAt($i);
    $packagesPurchased[$id][$cat]["Sponsored"] = array("Name"=>$name,
                                                       "Cost"=>$cost,
                                                       "Entries"=>array());
    $packagesPurchased[$id][$cat]["Non"] = array("Name"=>$name,
                                                 "Cost"=>$cost,
                                                 "Entries"=>array());
    $packagesPurchased[$id][$cat]["Outsider"] = array("Name"=>$name,
                                                      "Cost"=>$cost,
                                                      "Entries"=>array());
  }
$query =<<<END_QUERY
 SELECT DISTINCT firstname, lastname, peopleid, packages_purchased.packageid,
                 rules_age_level.name, org_pays, paid, o1.name, o2.name
FROM people NATURAL JOIN
     packages_purchased NATURAL JOIN
     people_classification JOIN
     rules_age_level ON (agelevel=rules_age_level.index) NATURAL JOIN
     people_paid JOIN couples ON (peopleid=follower OR
                                  peopleid=leader) NATURAL JOIN
     events_registration JOIN organizations AS o1 ON (org_paying=o1.orgid) JOIN
     organizations AS o2 ON (organization=o2.orgid)
WHERE o1.orgid='$affilID'
END_QUERY;
if ($affilID==-1)
  $query .= "OR organization IS NULL ";
$query .= "ORDER BY lastname, firstname;";
$result = $this->query($query);
$numPeople = $result->numRows();
for ($i=0;$i < $numPeople;$i++)
  {
    list($first, $last, $id, $pid, $cat, $orgPays, $paid,
         $orgPayingName,$orgName) = $result->getRowAt($i);
```

```
      $orgPaying=($orgPays=='t'?TRUE:FALSE);
      $paid=($paid=="t"?TRUE:FALSE);
      $query =<<<END_QUERY
SELECT DISTINCT coupleid, peopleid, firstname, lastname, o1.name, o2.name,
                packages_purchased.packageid, rules_age_level.name, paid
FROM people JOIN
      couples ON ((follower=peopleid AND leader=$id) OR
                (leader=peopleid AND follower=$id)) NATURAL JOIN
      events_registration LEFT JOIN
      organizations AS o1 ON (organization=o1.orgid) NATURAL JOIN
      people_classification JOIN
      rules_age_level ON (agelevel=rules_age_level.index) NATURAL JOIN
      packages_purchased NATURAL JOIN
      people_paid LEFT JOIN
      organizations AS o2 ON (org_paying=o2.orgid)
ORDER BY lastname, firstname;
END_QUERY;
      $partnerResult = $this->query($query);
      $numbers = array();
      $partners = array();
      $numPartners = $partnerResult->numRows();
      $numbers=array();
      for ($j=0;$j < $numPartners;$j++)
        {
          list($cid, $pId, $pFirst, $pLast, $pAffil, $pPaidAffil, $pPid, $pCat,
               $pPaid) = $partnerResult->getRowAt($j);
          $pPaid=($pPaid=="t"?TRUE:FALSE);

         if ($this->getStatus()!="OPEN")
           {
             $query =<<<END_QUERY
SELECT DISTINCT competitornumber
FROM events_registration NATURAL JOIN
     couples
WHERE coupleid=$cid and leader=$id;
END_QUERY;
             $numbersResult = $this->query($query);
             $numNumbers = $numbersResult->numRows();
             for ($k=0;$k < $numNumbers;$k++)
               {
                 list($num) = $numbersResult->getRowAt($k);
                 if (!in_array($num,$numbers) && $num!="")
                     array_push($numbers,$num);
               }
                 sort($numbers);
           }
         if ($pAffil==-1 || $pAffil=="") $pAffil="Independant";
         if ($pPaidAffil==-1 || $pPaidAffil=="") $pPaidAffil="Independant";
         $pName = substr($pFirst,0,1) . ". $pLast";
         $pName = stripslashes($pName);
         array_push($partners,$pName);

         if ($pPaidAffil!=$orgPayingName && $pFirst != "TBA")
           {
             $primeName = substr($first,0,1);
             $primeName .= ". $last";
             $pFullName = "$pLast, $pFirst ($pAffil)";
             $pFullName = stripslashes($pFullName);
             array_push($packagesPurchased[$pPid][$pCat]["Outsider"]["Entries"],
                        array("Name"=>"$pFullName",
```

```
                                "ID"=>$pId,
                                "Number"=>"",
                                "Partners"=>array($primeName),
                                "Paid"=>$pPaid,
                                "Org"=>$pPaidAffil));
            }

        }
        $fullName = "$last, $first";
        $fullName .= ($orgPayingName!=$orgName ? " ($orgName)":"");
        $fullName = stripslashes($fullName);
        if ($orgPaying)
          {
            array_push($packagesPurchased[$pid][$cat]["Sponsored"]["Entries"],
                       array("Name"=>"$fullName",
                             "ID"=>$id,
                             "Partners"=>$partners,
                             "Numbers"=>$numbers,
                             "Paid"=>$paid,
                             "Org"=>$orgName));
          }
        else
          {
            array_push($packagesPurchased[$pid][$cat]["Non"]["Entries"],
                       array("Name"=>"$fullName",
                             "ID"=>$id,
                             "Partners"=>$partners,
                             "Numbers"=>$numbers,
                             "Paid"=>$paid,
                             "Org"=>$orgName));
          }
    }
}

return $packagesPurchased;
} // end func

} // end class
?>
```

database/DB.inc

```
<?php
#################################################################
# This file is part of SlidingDoors                             #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    #
# SlidingDoors is availible for license under the GPL, see      #
# the COPYING file in the root directory of the install for     #
# the full terms of the GPL.                                    #
#                                                               #
# File: DB.inc                                                  #
# Author: Eric D. Nielsen                                       #
# Description: The database abstraction class                   #
# Attributes: db -- the handle to the DB                        #
# Constructor: DB                                               #
# Methods:      query                                           #
#               connectionValid                                 #
#               getLastOID                                      #
# Change Log: 6/12/01 -- created -- edn                         #
#             3/6/02 -- added getLastOID functionality --edn    #
#################################################################
class DB
{
    var $db;
    var $lastOID;

    ###############################################################
    # Function: DB                                                #
    # Arguments: dbname -- what local db to connect to            #
    #            user -- who is connecting                        #
    #            pass -- with what password                       #
    # Returns: nothing                                            #
    # Modifies: this                                              #
    # Requires: nothing                                           #
    ###############################################################
    function DB($dbname, $user, $pass)
    {
        $this->db = @pg_connect("dbname=$dbname user=$user password=$pass");
        $this->lastOID=0;
    }

    ###############################################################
    # Function: query                                             #
    # Arguments: query -- he query to run on the database         #
    # Returns: a QueryResult object (numRows and all results)     #
    # Modifies: nothing                                           #
    # Requires: nothing                                           #
    # Notes: on error the QueryResult has a -1 set as its rows    #
    ###############################################################
    function query($query)
    {
        $r = new QueryResult();
#       echo "<br>$query";
        $result = pg_exec($this->db, $query);
        if (!$result)
        {
            $r->setError("Error executing query");
            return $r;
        }
        if (stristr($query,"INSERT"))
        {
```

```
            $this->lastOID=pg_getlastoid($result);
        }
        $num = pg_numrows($result);
        for ($i=0;$i<$num;$i++)
        {
            $temp = pg_fetch_row($result,$i);
            if (!$temp)
            {
                $r->setError("Error retrieving row $i");
                return $r;
            }
            $r->addRow($temp);
        }
        return $r;
    }

    ##############################################################
    # Function: getLastOID                                       #
    # Arguments: none                                            #
    # Returns: the OID of the last inserted row or zero          #
    # Modifies: this                                             #
    # Requires: nothing                                          #
    ##############################################################
    function getLastOID()
    {
        return $this->lastOID;
    }

}
?>
```

database/QueryResult.inc

```
<?php
#################################################################
# This file is part of SlidingDoors                             #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    #
# SlidingDoors is availible for license under the GPL, see      #
# the COPYING file in the root directory of the install for     #
# the full terms of the GPL.                                    #
#                                                               #
# File: QueryResult.inc                                         #
# Author: Eric D. Nielsen                                       #
# Description: The QueryResult abstraction class                #
# Attributes: rows -- how many results are stored here          #
#             results -- an array of arrays of results          #
#             error -- hold the error status of the Query       #
# Constructor: QueryResult                                      #
# Methods:     numRows                                          #
#              addRow                                           #
#              setError                                         #
#              getError                                         #
#              getRowAt                                         #
# Change Log: 6/12/01 -- created -- edn                         #
#################################################################
class QueryResult
{
    var $rows;
    var $results;
    var $error;

    ###############################################################
    # Function: QueryResult                                       #
    # Arguments: nothing                                          #
    # Returns: nothing                                            #
    # Modifies: this                                              #
    # Requires: nothing                                           #
    ###############################################################
    function QueryResult()
    {
        $this->rows=0;
        $this->results=array();
        $this->error="";
    }

    ###############################################################
    # Function: setError                                          #
    # Arguments: error -- the error message                       #
    # Returns: nothing                                            #
    # Modifies: this                                              #
    # Requires: nothing                                           #
    ###############################################################
    function setError($error)
    {
        $this->error=$error;
    }

    ###############################################################
    # Function: getError                                          #
    # Arguments: nothing                                          #
    # Returns: 0 if no error, the error string otherwise          #
    # Modifies: nothing                                           #
```

```
    # Requires: nothing                                                  #
    ##########################################################
    function getError()
    {
        $temp = $this->error;
        return ($temp!="" ? $temp : 0);
    }

    ##########################################################
    # Function: numRows                                                  #
    # Arguments: nothing                                                 #
    # Returns: the number of rows stored here                            #
    # Modifies: nothing                                                  #
    # Requires: nothing                                                  #
    # Note: If there is an error there could still be rows               #
    #       stored here.                                                 #
    ##########################################################
    function numRows()
    {
        return $this->rows;
    }

    ##########################################################
    # Function: addRow                                                   #
    # Arguments: row -- the row to add                                   #
    # Returns: notihing                                                  #
    # Modifies: this                                                     #
    # Requires: nothing                                                  #
    ##########################################################
    function addRow($row)
    {
        $this->results[$this->rows++]=$row;
    }

    ##########################################################
    # Function: getRowAt                                                 #
    # Arguments: index                                                   #
    # Returns: the index'th row of the query                             #
    # Modifies: nothing                                                  #
    # Requires: nothing                                                  #
    ##########################################################
    function getRowAt($index)
    {
        return $this->results[$index];
    }

}
?>
```

html-formatting/CIB_Display.inc

```
<?php
/*************************************************************
 * This file is part of CompInaBox                           *
 * Copyright 2001-2002. Eric D. Nielsen, All rights reserverd *
 * CompInaBox is availible for license under the GPL, see    *
 * the COPYING file in the root directory of the install for *
 * the full terms of the GPL.                                *
 *                                                           *
 * File: CLASSES/CIB_Display.inc                             *
 * Type: Class Files                                         *
 * Author: Eric D. Nielsen                                   *
 * Description: HTML formatting for CIB Core                 *
 * Attributes: only inherited                                *
 * Constructor: CIB_Display                                  *
 * Public Methods: userPage, loginForm, loginCell, helpBar,  *
 *                 hostedComps, compResults, cibServers,     *
 *                 outsideComps, compinaboxLogo              *
 * Private Methods: helper_formatLinks                       *
 * Change Log: 8/11/02 -- created -- edn                     *
 *************************************************************/
class CIB_Display extends UIBase
{
  // NO NON-DERIVED ATTRIBUTES

  // CONSTRUCTOR
  /*************************************************************
   * Function: CIB_Display                                     *
   * Arguments: db -- the queriable database object            *
   * Modifies: this                                            *
   *************************************************************/
  function CIB_Display($db="")
    {
      UIBase::UIBase($db);
    }


  // Public Methods
  /*************************************************************
   * Function: userPage                                        *
   * Arguments: user -- display this users comps, teams, admin *
   * Returns: a 2 column wide summary of this user's options   *
   *          on this server, like a "my" page on portals      *
   * Requires: user is a valid, logged in user                 *
   *************************************************************/
  function userPage($user)
    {
      $cell = "<td colspan=\"2\">\n";
      //:TODO: CIB_Display::userPage
      $username .= $user->getUsername();
      $cell .= "Welcome back, $username <a href=\"accounts/SCRIPTS/logout.php\">
[Logout]</a><br />\n";
      $cell .= $this->pendingActions($user);
      $cell .= $this->competitionMenu($user->getComps());

      $cell .= "I want to:\n";
      $cell .= "<ul><li><a href=\"setup/\">Set Up a New Competition</a></li>\n";
#      $cell .= "<li><a href=\"comingsoon.html\">Advertise a new CIB server,
competition, etc</a></li></ul>";
```

```
        $cell .= "</td>";
        return $cell;
      }

    function displayUserKeys($user)
      {
        $table = "\n<table>";
        $table .="\n\t<tr>\n\t\t<th>Key Type</th><th>Key Level</th>";
        $table .= "<th>Key Label</th>";
        $table .= "\n\t</tr>";
        $keys = $this->db->expandTeamNamesFromID($user->getKeys());
        usort($keys,"keyCompare");
        $table .= "\n\t<tr><th colspan=\"3\">Granted Keys</th></tr>";
        $table .= $this->helper_displayKeys($keys, "You have no keys");
        $keys = $this->db->expandTeamNamesFromID(
$this->db->getPendingKeys($user->getID()));
        usort($keys,"keyCompare");
        $table .= "\n\t<tr><th colspan=\"3\">Pending Requests</th></tr>";
        $table .= $this->helper_displayKeys($keys, "You have no pending key requests");
        $table .= "</table>";
        return $table;
      }

    function helper_displayKeys($keys,$noLabel)
      {
        $table ="";
        if (count($keys)==0)
          $table .= "<tr>\n\t\t<td colspan=\"3\"><em>$noLabel</em></td>\n\t</tr>";
        foreach ($keys as $aKey)
          {
            $type = $aKey["Type"];
            $level =$aKey["Level"];
            switch ($type)
              {
              case "Site" : $label="<em>unused</em>";break;
              case "Comp" : $label=$aKey["Compname"];break;
              case "Team" : $label=$aKey["Teamname"];break;
              default: break;
              }
            $table .= "<tr>\n\t\t<td>$type</td><td>$level</td>";
            $table .= "<td>$label</td>\n\t</tr>";
          }
        return $table;
      }


    function displayKeyRequest($user)
      {
        $table = "<table border=\"1\">\n";
        $table .= "<tr><th>Site</th><th>Comp</th><th>Team</th></tr>\n";
        $table .= "<tr><td><select name=\"site_level\" size=\"1\">\n";
        $table .= "\t<option value=\"0\" selected=\"selected\">No Change</option>\n";
        if (!$user->isSiteAdmin())
          $table .= "\t<option \"1\">Admin</option>\n";
        $table .= "</select></td>\n";
        $table .= "<td>You must select an option from both boxes to generate a
valid request.<br /><select name=\"comp_name\" size=\"1\">\n";
        $table .= "\t<option value=\"0\" selected=\"selected\">No Change</option>\n";
        $comps = $this->db->getComps();
        foreach ($comps as $aComp)
          $table .= "\t<option value=\"{$aComp["Link"]}\">{$aComp["Label"]}</option>\n";
```

```
        $table .= "</select><select name=\"comp_level\" size=\"1\">\n";
        $compLevels = array("Staff","Comptroller","Registrar","Coordinator");
        $table .= "<option value=\"0\" selected=\"selected\">No Change</option>\n";
        foreach ($compLevels as $aLevel)
          $table .= "\t<option value=\"$aLevel\">$aLevel</option>\n";

        $table .= "</select></td>\n";
        $table .= "<td>You must select an option from both boxes to generate a
valid request.<br /><select name=\"team_name\" size=\"1\">\n";
        $table .= "\t<option value=\"0\" selected=\"selected\">No Change</option>\n";
        $teams = $this->db->getTeams();
        foreach ($teams as $aTeam)
          $table .= "\t<option value=\"{$aTeam["Link"]}\">{$aTeam["Label"]}</option>\n";
        $table .= "</select><select name=\"team_level\" size=\"1\">\n";
        $teamLevels = array("Member","Treasurer","Registration Coordinator","Captain");
        $table .= "<option value=\"0\" selected=\"selected\">No Change</option>\n";
        foreach ($teamLevels as $aLevel)
          $table .= "\t<option value=\"$aLevel\">$aLevel</option>\n";

        $table .= "</select></td><td></td></tr></table>\n";
        return $table;
      }

    /*************************************************************
     * Function: pendingActions                                  *
     * Arguments: user -- who's pending action list to generate  *
     * Returns: a table of formatted links                       *
     * Requires: nothing                                         *
     *************************************************************/
    function pendingActions($user)
      {
        GLOBAL $CIB_SECURE_URL;
        $compTable ="";

        if ($user->isSiteAdmin())
          {
            $comps = $this->db->getComps("Pending Admin");
            if (count($comps))
              {
                $compTable = "<b>There are competitions awaiting approval</b><br />";
                $compTable .= "<table>\n";
                foreach($comps as $aComp)
                  $compTable .= "<tr><th>{$aComp["Name"]}</th><td>
[Preview]</td><td>[Approve]</td></tr>\n";
                $compTable .= "</table>";
              }
          }
        $pendingPerms =$this->db->getPendingPermissions($user);
        if (count($pendingPerms))
          {
            $permTable = "<b>The following users are requesting
permission escalation</b><br />";
            $permTable.=
"<form action=\"$CIB_SECURE_URL/accounts/SCRIPTS/grant_permission.php\"
method=\"POST\">\n";
            $permTable.="<table>\n";
            $permTable.="<tr><th>Username</th><th>Permission</th><th>Action</th></tr>\n";
            $i=0;
            foreach($pendingPerms as $aPerm)
              {
                $username=$aPerm["Username"];
```

```
                $userID=$aPerm["UserID"];
                $permID=$aPerm["PermID"];
                $key=$aPerm["Key"];
                $key=keyString($key);
                $permTable.="<tr><td>$username</td><td>$key</td><td>";
                $permTable.="<input type=\"radio\" name=\"actions$i\"
value=\"Unchanged-$userID-$permID\" checked=\"checked\" />No action\n";
                $permTable.="<input type=\"radio\" name=\"actions$i\"
value=\"Grant-$userID-$permID\" />Grant\n";
                $permTable.="<input type=\"radio\" name=\"actions$i\"
value=\"Deny-$userID-$permID\"  />Deny\n";
                $permTable.="</td></tr>";
                $i++;

            }
          $permTable .= "<tr><td colspan=\"2\"> </td><td>";
          $permTable .= "<input type=\"submit\" value=\"Process Requests\">\n";
          $permTable .= "</td></tr></table></form>\n";
        }
      return $compTable.$permTable;
    }

  function competitionMenu($comps)
  /*************************************************************
   * Function: competitionMenu                                 *
   * Arguments: comps -- a list of competitions and tools the  *
   *                     user has access to                    *
   * Returns: a table of formatted links                       *
   * Requires: nothing                                         *
   *************************************************************/
    {
      GLOBAL $baseURL;
      $numComps = count($comps);
      if ($numComps != 0)
        {
          $menu = "<table>\n";
          $menu .= "\t<tr>\n\t\t<th>Competition</th>\n";
          $menu .= "\t\t<th>Register</th><th>Affiliation Admin</th>\n";
          $menu .= "\t\t<th>Registration Admin</th>
<th>Pre-Registration Setup</th></tr>\n";
          for ($i=0;$i < $numComps; $i++)
            {
              $compName = $comps[$i]["CompName"];
              $compUnix = $comps[$i]["CompUnixname"];
              $tasks    = $comps[$i]["Tasks"];
              $menu .= "\t<tr>\n\t\t<th>$compName</th>\n";
              if (in_array("Register",$tasks))

                $menu .= "\t\t<td><a href=\"register/$compUnix\">
[Register]</a></td>\n";
              else
                $menu .= "\t\t<td> </td>\n";
              if (in_array("Affil",$tasks))
                $menu .= "\t\t<td>[Team Management]</td>\n";
              else
                $menu .= "\t\t<td> </td>\n";
              if (in_array("Reg Admin",$tasks))
                $menu .= "\t\t<td><a href=\"register/$compUnix/admin/\">
[Comp Admin]</a></td>\n";
              else
                $menu .= "\t\t<td> </td>\n";
```

```
            if (in_array("Setup",$tasks))
              $menu .= "\t\t<td>[Comp Setup]</td>\n";
            else
              $menu .= "\t\t<td> </td>\n";
            $menu .= "</tr>\n";
          }
        $menu .= "</table>\n";
      }
    else
      $menu = "";
    return $menu;
  }


/*************************************************************
 * Function: loginForm                                       *
 * Arguments: returnTo -- return the user to this location   *
 * Returns: a simple form to authenticate the user that can  *
 *          be embedded in other pages                       *
 *************************************************************/
function loginForm($returnTo)
  {
    GLOBAL $HTTP_SESSION_VARS, $HTTP_REFERER;
    GLOBAL $CIB_SECURE_URL;
    $username =$HTTP_SESSION_VARS["username"];
    if ($username!="")
      {
        $nonce = $HTTP_SESSION_VARS["nonce"];
        $challenge =$HTTP_SESSION_VARS["challenge"];
        $test = MD5(session_id().$username.$nonce.$HTTP_REFERER);
        if ($test!=$challenge)
          {
            // handling session hijacki
            return "You're being Naughty!";
__________}
________}
______$login_msg = $HTTP_SESSION_VARS["login_msg"];
______if ($login_msg!="")
________$login_msg = "<font color=\"red\">$login_msg</font><br/>\n";
______$form.="<form action=\"$CIB_SECURE_URL/accounts/SCRIPTS/login.php\"
method=\"post\">\n";
______$form.="<input type=\"hidden\" name=\"returnTo\" value=\"$returnTo\"/>\n";
______$form.=$login_msg;
______$form.="Username:<input type=\"text\" name=\"username\"
value=\"$username\"/><br/>\n";
______$form.="Password:<input type=\"password\" name=\"password\"/><br/>\n";
______$form.="<input type=\"submit\" value=\"Login\"/>\n";
______$form.="</form>\n";
______$form.="Create a <a href=\"$CIB_SECURE_URL/accounts/new_account.php\">
New Account</a>\n";
______return $form;
____}


__/*************************************************************
___* Function: loginCell                                      *
___* Returns: a login form embedded in a table cell. Returns  *
___*          the user to the index/home page                  *
___*************************************************************/
__function loginCell()
```

```
    {
        $cell ="<td colspan=\"2\"><h3>Login</h3>\n";
        $cell.="You only need to login if you using this server to host your
competition or are the \"registration coordinator\" for your team/studio.
You do <b>not</b> need to login to register for a competition.<p />\n";
        $cell .= $this->loginForm("/");
        $cell.="</td>";
        return $cell;
    }


  /*************************************************************
   * Function: helpBar                                         *
   * Returns: a complete table row, with links to site defined *
   *          help documentation.                              *
   *************************************************************/
  function helpBar()
    {
      $helpItems = $this->db->getDocumentList(array("'Help Navigation Bar'"));
      $numItems = count($helpItems);
      if ($numItems>0)
      {
        $row ="<tr valign=\"top\"><td colspan=\"3\">\n";
        $row .="\t<b>Looking for Help:</b>\n";
        for ( $i = 0; $i < $numItems; $i++)
        {
          $row.=( $i > 0 ?"\t":"\t|");
          $url = $helpItems[$i]["Link"];
          $label = $helpItems[$i]["Label"];
          $row .="<a href=\"$url\">$label</a>\n";
        }
        $row .="</td></tr>\n";
      }
      return $row;
    }


  /*************************************************************
   * Function: hostedComps                                     *
   * Returns: a tableCell, with links to the registration page *
   *          of all the competitions with open registration.  *
   *************************************************************/
  function hostedComps()
    {
      $hostedComps = $this->db->getComps("OPEN");
      $cell ="<td>\n";
      $cell .="\t<h3>Trying to Register?</h3>\n";
      $numItems = count($hostedComps);
      if ($numItems>0)
      {
        $cell .="Simply click on the name of the competition to start the
registration process.<p/>\n";
        $cell .= $this->helper_formatLinks($hostedComps,$numItems,"register/");
        $cell .="[<a href=\"comp_summary.php?type=OPEN\">Competition Summary</a>]\n";
      }
      else
      {
        $cell .="There are no open competitions hosted on this server.";
      }
      $cell .="</td>\n";
      return $cell;
```

```
----}


  /*************************************************************
   * Function: compResults                                     *
   * Returns: a tableCell, with links to the results page of   *
   *          the most recent 10 competitions and a link to    *
   *          full archive                                     *
   *************************************************************/
  function compResults()
    {
      $hostedComps = $this->db->getComps("ARCHIVE","occurs_on",10);
      $cell = "<td>\n";
      $cell .= "\t<h3>View Past Results</h3>\n";
      $numItems = count($hostedComps);
      if ($numItems>0)
        {
          $cell .= "The most recent 10 competitions are shown below, for
older competitions follow the link at the bottom of the list:<p />\n";
          $cell .= $this->helper_formatLinks($hostedComps,$numItems," results/");
          $cell .= "[<a href=\"comp_summary.php?type=RESULTS\">Results Archive</a>]\n";
        }
      else
        {
          $cell .= "This feature is coming soon.";
        }
      $cell .= "</td>\n";
      return $cell;
    }


  /*************************************************************
   * Function: cibServers                                      *
   * Returns: a tableCell, with links to other CIB servers     *
   *************************************************************/
  function cibServers()
    {
      $cibServers = $this->db->getCibServers(TRUE);
      $cell = "<td>\n";
      $cell .= "\t<h3>Other CIB Servers</h3>\n";
      $numItems = count($cibServers);
      if ($numItems>0)
        {
          $cell .= "If you expected to find a competition that isn't listed
above, you should check some of the other servers, especially if there is one
closer to you.<p />";
          $cell .= $this->helper_formatLinks($cibServers,$numItems);
          $cell .= "[<a href=\"comingsoon.html\">More Info on CIB Servers</a>]\n";
        }
      else
        {
          $cell .= "No other CIB Server has asked to be listed.";
        }
      $cell .= "</td>\n";
      return $cell;
    }


  /*************************************************************
   * Function: outsideComps                                    *
   * Returns: a tableCell, with links to competitions on other *
```

```
    *           registration services, the next 10 are shown as  *
    *           links, with the full listing on a breakout page  *
    ************************************************************/
  function outsideComps()
    {
      $otherComps = $this->db->getOutsideComps(10);
      $cell = "<td>\n";
      $cell .= "\t<h3>Other Competitions</h3>\n";
      $numItems = count($otherComps);
      if ($numItems>0)
        {
          $cell .= "CIB may not be the answer for all competitions; following
is a list of upcoming competitions using alternate registration tools.<p />";
          $cell .= $this->helper_formatLinks($otherComps,$numItems);
          $cell .= "[<a href=\"outside_summary.php\">Other Competitions</a>]\n";
        }
      else
        {
          $cell .= "We don't know of any outside competitions.";
        }
      $cell .= "</td>\n";
      return $cell;
    }


  /************************************************************
    * Function: compinaboxLogo ------------------------------*
    * Returns: a tableCell, with the CompInaBox logo image ---*
    ************************************************************/
  function compinaboxLogo()
    {
      //:TODO: get a real image made
      //:TODO: get the real pixel dimensions
      $logo = "<img src=\"images/compinaboxLogo.jpg\" alt=\"CompInaBox Logo\"\n";
      $logo .="\theight=\"260\" width=\"250\">\n";
      return $logo;
    }

  //Private Methods
  /************************************************************
    * Function: helper_formatLinks --------------------------*
    * Arguements: items---an array of URLS and labels -------*
    * ------------numItems---if the count has already been ---*
    * -------------------------calculated you may pass it in to*
    * -------------------------save this function from redoing it*
    * Returns: a<br/>-seperated series of links -------------*
    ************************************************************/
  function helper_formatLinks($items,$numItems="",$path="")
    {
      if ($numItems=="")$numItems=count($items);
      $list ="<ul>\n";
      for ($i=0;$i<$numItems;$i++)
        {
          $url = $items[$i]["Link"];
          $label = $items[$i]["Label"];
          $url = str_replace("&","&amp;",$url);
          $list .="<li><a href=\"$path"."$url\">$label</a></li>\n";
        }
      $list .="</ul>\n";
      return $list;
    }
```

```
}
?>
```

html-formatting/HTMLDisplay.inc

```
<?php
#####################################################################
# This file is part of SlidingDoors                                 #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd        #
# SlidingDoors is availible for license under the GPL, see          #
# the COPYING file in the root directory of the install for         #
# the full terms of the GPL.                                        #
#                                                                   #
# File: HTMLDisplay.inc                                             #
# Author: Eric D. Nielsen                                           #
# Description: Module containing function calls for generating      #
#              HTML pages                                           #
# Attributes:  from superclass                                      #
# Constructor: HTMLDisplay                                          #
# Methods:     need to list                                         #
# Change Log: 8/08/01 -- created -- edn                             #
#             6/26/02 -- GPL and constants.inc/forms                #
#             7/19/02 -- prime update handling, partner -- edn      #
#####################################################################
class HTMLDisplay extends UIBase
{

    function HTMLDisplay($db=0)
    {
      UIBase::UIBase($db);
    }


    function beginPage($pageTitle="",$pageHeader="")
    {
        GLOBAL $SD_compname, $baseURL, $SD_style_sheet, $SD_comp_url;
        GLOBAL $SD_title_color, $SD_title_font, $SD_background_color;
        GLOBAL $SD_background_color, $SD_text_color, $basePath;
        GLOBAL $CIB_WEB_PATH,$unixname,$CIB_BASE_URL;
        if ($pageTitle=="")
            $pageTitle=$SD_compname;
        if ($pageHeader=="")
            $pageHeader=$pageTitle;
#       $page = "<!-- created by HTMLDisplay::beginPage($pageTitle,\n";
#       $page .= "\t $pageHeader) -->\n";
#       $page .= "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\"\n";
#        $page .=  "\t\"http://www.w3.org/TR/1999/REC-html401-19991224/loose.dtd\">\n";
#       $page .= "<html lang=\"en\">\n";
#       $page .= "<head>\n";
#       $page .= "\t<meta http-equiv=\"content-type\"
content=\"text/html; charset=iso-8859-1\">\n";
#       $page .= "\t<title>$pageTitle</title>\n";
#       $page .= "\t<link rel=\"stylesheet\" href=\"$baseURL/$SD_style_sheet\">\n";
#       $page .= "</head>\n\n";
        $page = $this->documentHead($pageTitle,"$baseURL/$SD_style_sheet","HTML");
        $page .= "<body bgcolor=\"$SD_background_color\" text=\"$SD_text_color\">\n";
        $page .= "<!-- This is the Comp Name which should appear at the top
of every page -->\n";
        $page .= "<center>\n";
        $page .= "<font color=\"$SD_title_color\" style=\"$SD_title_font\">\n";
        $page .= "<h1>$pageHeader</h1>\n";
        $page .= "</font>\n";
        $page .= "</center>\n";
        $page .= "<!-- End Comp Name -->\n\n";
```

```
        $page .= "<!-- Navigation bar -->\n";
        $page .= "<div id=\"NavBar\" align=\"center\">\n";
        $page .= "\t<a href=\"$baseURL\">Registration Home</a> | \n";
        $page .= "\t<a href=\"$CIB_BASE_URL/stats/$unixname/\">
Data on Registrations</a> | \n";
        $page .= "\t<a href=\"http://$SD_comp_url\">Competition Home</a>\n";
        $page .= "</div>\n";
        $page .= "<!-- End Navigation bar -->\n\n";
        $page .= "<center>\n";
        if (is_object($this->db))
                $page .= $this->db->getMOTD();
        $page .= "<!-- End HTMLDisplay::beginPage -->\n\n";

        return $page;
    }

    function instructionBox($heading, $text)
    {
        $box = "<!-- HTMLDisplay::instructionBox($heading,[text]) -->\n";
        $box .="<!-- This is the section/page number and instructions section -->\n";
        $box .= "\t<h2>$heading</h2>\n";
        $box .= "\t<p \>\n";
        $box .= "$text\n";
        $box .= "<!-- End section/page number and instructions -->\n";
        $box .= "<!-- End HTMLDisplay::instructionBox -->\n\n";
        return HTMLDisplay::textBox($box);
    }

    function errorBox($text)
    {
        GLOBAL $SD_error_background_color, $SD_error_text_color;
        if ($text=="") return "";
        $box = "<!-- HTMLDisplay::errorBox([text]) -->\n";
        $box .= "<!-- This is box alerting the user to errors on the page -->\n";
        $box .= "<center>\n";
        $box .= "<table bgcolor=\"$SD_error_background_color\" border=\"1\"
cellpadding=\"8\" width=\"75%\">\n";
        $box .= "<tr><td>\n";
        $box .= "<font color=\"$SD_error_text_color\">\n";
        $box .= "$text\n";
        $box .= "</font>";
        $box .= "</td></tr></table>\n";
        $box .= "<!-- End error box -->\n";
        $box .= "<!-- End HTMLDisplay::errorBox() -->\n\n";
        return $box;
    }

    function compinaboxBox()
    {
        $beginBox = "<!-- HTMLDisplay::compinaboxBox() -->\n";
        $endBox = "<!-- End HTMLDisplay::compinaboxBox() -->\n\n";
        GLOBAL $beginTime;
        return $beginBox .
          HTMLDisplay::textBox(UIBase::developerContactInfo("SlidingDoors",
                                                            "sd.html")) .
          $endBox;
    }

    function endPage()
    {
        $page = "<!-- HTMLDisplay::endPage() -->\n";
```

```
        $page .= "</center>\n";
        $page .= UIBase::endPage();
        $page .= "<!-- End HTMLDisplay::endPage() -->\n";
        return $page;
    }

    function textBox($text,$width="95%", $comment="")
    {
        GLOBAL $SD_text_background_color;
        if ($text=="") return "";
        if ($comment=="")
            $comment = "This is a general purpose box displaying content";
        $beginBox =  "<!-- HTMLDisplay::textBox([text],$width,[comment]) -->\n";
        $box = "<!-- $comment -->\n";
        $box .= "<table bgcolor=\"$SD_text_background_color\" border=\"1\"
cellpadding=\"8\" width=\"$width\">\n";
        $box .= "<tr><td>\n";
        $box .= "$text\n";
        $box .= "</td></tr></table>\n";
        $box .= "<!-- End ($comment) -->\n";
        $endBox = "<!-- End HTMLDisplay::textBox -->\n\n";
        return $beginBox . $box . $endBox;
    }

    function registerBox()
    {
        GLOBAL $SD_button_style,$basePath;
        $status = $this->db->getStatus();
        $beginBox = "<!-- HTMLDisplay::registerBox() -->\n";
        $box = "\t\t<h3>Register</h3>\n";
        $motd = $this->db->getMOTD("Register");
        if (""!=$motd)
          $box .= $motd . "<br />";
        if ($status=="OPEN" || $status=="TESTING")
          {
            $box .= "\t\t<form action=\"prime.php\" method=\"POST\">\n";
            $box .= "\t\tTo begin the registration process, press the \"register\" ";
            $box .= "button.<br />";
            $box .= "\t\t<input type=submit name=\"Register\" value=\"Register\"\n";
            $box .= "\t\t\tstyle=\"$SD_button_style\">\n";
            $box .= "\t\t</form>\n";
          }
        else if ($status=="PLANNING")
          {
            $box .= "\t\tRegistration is not yet open.\n";
          }
        else
          $box .= "\t\tRegistration is closed.\n";
        $endBox = "<!-- End HTMLDisplay::registerBox() -->\n\n";
        return $beginBox .
            $this->textBox($box,"100%") .
            $endBox;
    }

    function browseBox()
    {
        GLOBAL $SD_button_style, $CIB_BASE_URL,$unixname;
        $beginBox = "<!-- HTMLDisplay::browseBox() -->\n";
        $box = "\t\t<h3>Browse Current Registrations</h3>\n";
        $box .= "\t\t<form action=\"$CIB_BASE_URL/stats/$unixname/events.php\"
method=\"POST\">\n";
```

```
        $motd = $this->db->getMOTD("Stats");
        if (""!=$motd)
          $box .= $motd . "<br />";
        $box .= "\t\tDrill down or search the current registration by \n";
        $box .= "\t\taffiliation, events, and names.\n";
        $box .= "\t\t<br />\n";
        $box .= "\t\t<input type=submit name=\"Browse\" value=\"Browse\"\n";
        $box .= "\t\t\tstyle=\"$SD_button_style\">\n";
        $box .= "\t\t</form>\n";
        $endBox = "<!-- End HTMLDisplay::browseBox() -->\n\n";
        return $beginBox .
            $this->textBox($box,"100%") .
            $endBox;
    }

    function retrieveBox()
    {
        GLOBAL $SD_button_style, $basePath;
        $status = $this->db->getStatus();
        $beginBox = "<!-- HTMLDisplay::retrieveBox() -->\n";
        $box = "\t\t<h3>View/Modify Existing Registrations</h3>\n";
        $motd = $this->db->getMOTD("Modify");
        if (""!=$motd)
          $box .= $motd . "<br />";
        if ($status == "OPEN" || $status=="TESTING")
          {
            $box .= "\t\t<form action=\"SCRIPTS/prime-retrieve.php\"
method=\"POST\">\n";
            $box .= "\t\tEnter your email address used to register, or\n";
            $box .= "\t\tchosen identifier if you do not have an email address\n";
            $box .= "\t\t<br />\n";
            $box .= "\t\t<input type=\"text\" name=\"Search\">\n";
            $box .= "\t\t<input type=\"submit\" name=\"RETREIVE\" value=\"Retrieve\"\n";
            $box .= "\t\t\tstyle=\"$SD_button_style\">\n";
            $box .= "\t\t</form>\n";
          }
        else if ($status=="PLANNING")
          {
            $box .= "\t\tRegistration is not yet open.\n";
          }
        else
          $box .= "\t\tRegistration is closed.\n";

        $endBox = "<!-- End HTMLDisplay::retrieveBox() -->\n\n";
        return $beginBox .
            $this->textBox($box,"100%") .
            $endBox;
    }

    function statsBox()
    {
      GLOBAL $CIB_WEB_PATH, $unixname;
        $beginBox = "<!-- HTMLDisplay::statsBox() -->\n";
        $fp = fopen("$CIB_WEB_PATH/stats/$unixname/summary.inc","r");
        if (!$fp)
            $box="No stats availible;";
        else
        {
            $box = fread($fp,filesize("$CIB_WEB_PATH/stats/$unixname/summary.inc"));
            fclose($fp);
        }
```

```
    $motd = $this->db->getMOTD("Stats");
    if (""!=$motd)
      $box .= $motd;
    $endBox = "<!-- End HTMLDisplay::statsBox() -->\n\n";
    return $beginBox .
        $this->textBox($box,"100%") .
        $endBox;
}

function groupAdminBox()
{
    GLOBAL $SD_button_style,$CIB_WEB_PATH,$unixname;
    include("$CIB_WEB_PATH/stats/$unixname/affils.inc");
    $beginBox = "<!-- HTMLDisplay::groupAdminBox() -->\n";
    $box .= "\t\t<h3>Studio/Team Options</h3>\n";
    $box .= "\t\t<form action=\"group/index.php\" method=\"POST\">\n";
    $motd = $this->db->getMOTD("Group");
    if (""!=$motd)
      $box .= $motd."<br />";
    $box .= "\t\t\tSelect the affiliation you wish to administer.\n";
    $box .= "\t\t\tOptions include Team Match Registration, reviewing\n";
    $box .= "\t\t\tteam registration, and other administration.\n";
    $box .= "<input type=\"hidden\" name=\"REQUESTTYPE\" value=\"CHARGE\">\n";
    $box .= "\t\t\t<br />\n";
    $box .=$this->arrayToSelect("Affil",$incAffils,"",1,1);
    $box .= "<input type=\"submit\" name=\"Retrieve\" value=\"Retrieve\"\n";
    $box .= "\t\t\tstyle=\"$SD_button_style\">\n";
    $box .= "\t\t</form>\n";
    $endBox = "<!-- End HTMLDisplay::groupAdminBox() -->\n\n";
    return $beginBox .
        $this->textBox($box,"100%") .
        $endBox;
}

function compInfoBox()
{
    GLOBAL $SD_text_background_color, $SD_venue, $SD_comp_url;
    GLOBAL $SD_compcoord_title, $SD_compcoord_name, $SD_compcoord_email;
    GLOBAL $SD_registrar_title, $SD_registrar_name, $SD_registrar_email;
    GLOBAL $SD_date, $SD_deadline, $SD_latefee;

    $beginBox = "<!-- HTMLDisplay::compInfoBox() -->\n";
    $box = "<!-- Begin Competition Information Box -->\n";
    $box .= "<table>\n";
    $box .= "<tr valign=\"top\">\n";
    $box .= "<td width=\"45%\">\n";
    $box .= "\t<table>\n";
    $box .= "\t\t<tr>\n";
    $box .= "\t\t\t<td><font size=\"-1\"><strong>Location</strong></font></td>\n";
    $box .= "\t\t\t<td><font size=\"-1\">\n";
    $box .= "\t\t\t\t$SD_venue</font></td>\n";
    $box .= "\t\t</tr><tr>\n";
    $box .= "\t\t\t<td><font size=\"-1\"><strong>\n";
    $box .= "\t\t\t\t$SD_compcoord_title";
    $box .= "</strong></font></td>\n";
    $box .= "\t\t\t<td><font size=\"-1\"><a href=\"mailto:$SD_compcoord_email\">\n";
    $box .= "\t\t\t\t$SD_compcoord_name</a></font></td>\n";
    $box .= "\t\t</tr><tr>\n";
    $box .= "\t\t\t<td><font size=\"-1\"><strong>$SD_registrar_title\n";
    $box .= "\t\t\t</strong></font></td>\n";
    $box .= "\t\t\t<td><font size=\"-1\"><a href=\"mailto:$SD_registrar_email\">\n";
```

```
        $box .= "\t\t\t\t$SD_registrar_name</a></font></td>\n";
        $box .= "\t\t</tr><tr>\n";
        $box .= "\t\t\t<td><font size=\"-1\"><strong>Competition Webpage\n";
        $box .= "\t\t\t\t</strong></font></td>\n";
        $box .= "\t\t\t<td><font size=\"-1\"><a href=\"http://$SD_comp_url\">\n";
        $box .= "\t\t\t\thttp://$SD_comp_url</a></font></td>\n";
        $box .= "</tr></table></td>\n";
        $box .= "<td align=\"right\" width=\"45%\">\n";
        $box .= "\t<table>\n";
        $box .= "\t\t<tr>\n";
        $box .= "\t\t\t<td><font size=\"-1\"><strong>Date</strong></font></td>\n";
        $box .= "\t\t\t<td><font size=\"-1\">$SD_date</font></td>\n";
        $box .= "\t\t<tr></tr>\n";
        $box .= "\t\t\t<td><font size=\"-1\"><strong>Registration Deadline\n";
        $box .= "\t\t\t\t</strong></font></td>\n";
        $box .= "\t\t\t<td><font size=\"-1\">$SD_deadline</font></td>\n";
        $box .= "\t\t<tr></tr>\n";
        $box .= "\t\t\t<td><font size=\"-1\"><strong>Deadline with Late Fee\n";
        $box .= "\t\t\t\t</strong></font></td>\n";
        $box .= "\t\t\t<td><font size=\"-1\">$SD_latefee</font></td>\n";
        $box .= "</tr></table></td>\n";
        $box .= "</tr></table>\n";
        $box .= "<!-- End Comp Information Box -->\n";
        $endBox = "<!-- End HTMLDisplay::compInfoBox() -->\n\n";
        return $beginBox . HTMLDisplay::textBox($box) . $endBox;
    }

    function doneBox($affilID=-1,$ignoreEmail="")
    {
        GLOBAL $SD_done_background_color, $SD_button_style, $SD_elements;
        $beginBox = "<!-- HTMLDisplay::doneBox() -->\n";
        $box .= "<table bgcolor=\"$SD_done_background_color\" border=\"1\"
cellpadding=\"8\" width=\"95%\">\n";
        $box .= "<tr><td>\n";
        $box .= "<table><tr><td colspan=\"2\">When you have added all the
partners for the current person, please \n";
        $box .= "</td></tr><tr><td>\n";
        $box .= "<form action=\"prime.php\" method=\"POST\">\n";
        if (in_array("Affiliation",$SD_elements))
            $box .= "<input type=\"hidden\" name=\"propagateAffil\"
value=\"$affilID\" />\n";
        if ($ignoreEmail!="")
          {
            $box .= "<input type=\"hidden\" name=\"propagateIgnoreEmail\"
value=\"TRUE\" />\n";
            $box .= "<input type=\"hidden\" name=\"propagateEnteredEmail\"
value=\"$ignoreEmail\" />\n";
          }
        else
          {
            $box .= "<input type=\"hidden\" name=\"propagateIgnoreEmail\"
value=\"FALSE\" />\n";
          }
        $box .= "<input type=\"submit\" value=\"Register Another Couple\"
style=\"$SD_button_style\"></form>\n";
        $box .= "</td><td>\n";

        $box .= "<form action=\"done.php\" method=\"POST\">\n";
        if (in_array("Affiliation",$SD_elements))
            $box .= "<input type=\"hidden\" name=\"propagateAffil\"
value=\"$affilID\" />\n";
```

```
        if ($ignoreEmail!="")
          {
            $box .= "<input type=\"hidden\" name=\"propagateIgnoreEmail\"
value=\"TRUE\" />\n";
            $box .= "<input type=\"hidden\" name=\"propagateEnteredEmail\"
value=\"$ignoreEmail\" />\n";
          }
        else
          {
            $box .= "<input type=\"hidden\" name=\"propagateIgnoreEmail\"
value=\"FALSE\" />\n";
   }
        $box .= "<input type=\"submit\" value=\"Commit and Exit\"
style=\"$SD_button_style\"></form></td></tr></table>\n";
        $box .= "</td></tr></table>\n";
        $endBox = "<!-- End HTMLDisplay::doneBox -->\n\n";
        return $beginBox . $box . $endBox;
    }

    function conflictResolutionBox($conflicts,$formVars,$primeID="")
    {
        GLOBAL $SD_done_background_color, $SD_button_style;
        $beginBox =  "<!-- HTMLDisplay::conflictResolutionBox() -->\n";
        $box = "<table bgcolor=\"$SD_done_background_color\" border=\"1\"
cellpadding=\"8\" width=\"95%\">\n";
        $box .= "<tr><td>\n";
        $box .= "<table width=\"100%\">\n";
        $numMatches = count($conflicts);
        for ($i=0;$i<$numMatches;$i++)
        {
            $box.= "\t".'<tr><td><b>Possible Match: #'.($i+1)."</b><br />";
            $box .= $this->conflictInfoCell($conflicts[$i]);
            $box .= "\t</td>";
            $box .= '<form action="SCRIPTS/prime-collision-action.php"
method="POST">'."<td>\n";
            $box .= '<input type="hidden" name="primeid" value="'.
$conflicts[$i]."\">\n";
            if (""!=$primeID)
                $box .= "<input type=\"hidden\" name=\"oldPrime\"
value=\"$primeID\">\n";
            $box .= $this->conflictInfoUpdate($conflicts[$i],
                                              $formVars);
            $box .= '<input type="submit" value="Update Identification">to ';
            $box .= '</td></form>';
            $box .= "<td>";
            $box .= $this->conflictInfoCell($conflicts[$i],
                                            $formVars);
            $box .= "</td>";
            $box .= '</tr><tr>';
            $box .= "<tr><td colspan=\"3\" bgcolor=\"$SD_background_color\"> ";
            $box .= "</td></tr>";
        }
        $box .= '<tr><td>I am not any of the <b>BOLDED</b> people.</td>
<td><form action="SCRIPTS/prime-collision-action.php" method="post">';
        if (""!=$primeID)
            $box .= "<input type=\"hidden\" name=\"oldPrime\" value=\"$primeID\">\n";
        $box .= $this->conflictInfoUpdate("",$formVars);
        $box .= '<input type="submit" value="Create My Entry">';
        $box .= "</form></td></tr></table>";
        $box .= "</td></tr></table>\n";
        $endBox = "<!-- End HTMLDisplay::conflictResolutionBox() -->\n\n";
```

```
        return $beginBox . $box . $endBox;
    }


    function personalInfoUpdateBar($person)
    {
        global $HTTP_SESSION_VARS,$SD_elements,$formVars;

        $primeID = $person->getID();
        $bar ="<h3>Personal Information Summary</h3>If your name is mispelled
or you wish to change the listed affiliation please enter the corrections and
press the update button.  If you need to create a new affiliation, please
select 'New_Affiliation'.\n";
        if (isset($HTTP_SESSION_VARS["formVars"]))
            $updated = array_keys($HTTP_SESSION_VARS["formVars"]);
        else
            $updated = array();
        if (in_array("Name",$SD_elements))
        {
            if (in_array("UpdateFirstName",$updated))
                $fname= $updated["UpdateFirstName"];
            else
                $fname=$person->getFirstName();
            if (in_array("UpdateLastName",$updated))
                $lname= $updated["UpdateLastName"];
            else
                $lname=$person->getLastName();
        }
        if (in_array("Email",$SD_elements))
            if (in_array("UpdateEmail",$updated))
                $email=$updated["UpdateEmail"];
            else
                $email=$person->getEmail();
        if (in_array("Fee Category",$SD_elements))
            if (in_array("UpdateAge",$updated))
                $age=$updated["UpdateAge"];
            else
                $age=$person->getAgeLevel();
        if (in_array("Package",$SD_elements))
            if (in_array("UpdatePackage",$updated))
                $package = $updated["UpdatePackage"];
            else
                $package = $person->getPackage();
        if (in_array("Affiliation",$SD_elements))
            if (in_array("UpdateAffil",$updated))
                $affilID = $updated["UpdatedAffil"];
            else
              $affilID = $person->getAffilID();
        $bar .= "<form action=\"SCRIPTS/prime-update.php\" method=\"POST\">";
        $bar .= "<input type=\"hidden\" name=\"primeID\" value=\"$primeID\">\n";
        $bar .= "<table>\n\t<tr>\n";
        if (in_array("Name",$SD_elements))
        {
            $bar .= "\t\t<td><span class=\"FormItemNames\">First Name:</span></td>\n";
            $bar .= "\t\t<td><span class=\"FormItemNames\">Last Name:</span></td>\n";
        }
        if (in_array("Email",$SD_elements))
            $bar .= "\t\t<td><span class=\"FormItemNames\">Email:</span></td>\n";
        if (in_array("Affiliation",$SD_elements))
            $bar .= "\t\t<td><span class=\"FormItemNames\">Affiliation:</span></td>\n";
```

```
        if (in_array("Name",$SD_elements) || in_array("Email",$SD_elements) ||
            in_array("Affiliation",$SD_elements))
            $bar .= "\t</tr><tr>\n";
        if (in_array("Name",$SD_elements))
        {
            $bar .= "<td><input type=\"text\" tabindex=1
name=\"updateFirstName\" size=\"15\" value=\"$fname\"></td>\n";
            $bar .= "<td><input type=\"text\" tabindex=2
name=\"updateLastName\" size=\"20\" value=\"$lname\"></td>\n";
        }
        if (in_array("Email",$SD_elements))
            $bar .= "<td><input type=\"text\" tabindex=3
name=\"updateEmail\" value=\"$email\"></td>\n";
        if (in_array("Affiliation",$SD_elements))
        {
            if ($affilID!="" && $affilID!=-1)
            {
                $org = new Organization($this->db,$affilID);
                $org->retrieve($affilID);
                $affil = $org->getName();
            }
            else
                $affil=-1;
            $bar .= "<td><select tabindex=4 name=\"updateAffilID\">\n
<option value=\"New\">New Affiliation...</option>" .
$this->db->getAffilListChoicesSelect($affil) . "</select>\n";
        }
        if (in_array("Name",$SD_elements) || in_array("Email",$SD_elements) ||
            in_array("Affiliation",$SD_elements))
            $bar .= "\t</tr>\n";
        if (in_array("Fee Category",$SD_elements) ||
            in_array("Package",$SD_elements))
            $bar .= "\t<tr>\n";
        if (in_array("Fee Category",$SD_elements))
            $bar .= "\t<tr>".$this->displayFormAgeOneLiner("updateAge",$age,5)."\n";
        if (in_array("Package",$SD_elements))
            $bar .= "\t".$this->displayFormPackageOneLiner("updatePackage",
$package,6)."\n";
        if (in_array("Fee Category",$SD_elements) ||
            in_array("Package",$SD_elements))
            $bar .= "\t</tr>\n";
        $bar .= "\t\t<td align=\"bottom\"><input type=\"reset\"
value=\"Undo Edits\"></td><td colspan=\"3\" align=\"botton\">
<input type=\"submit\" value=\"Update\" style=\"background: #d9a\"></td>\n";
        if ($HTTP_SESSION_VARS["formVars"]["UpdateEventsMsg"]!="")
            $bar .= "<tr><td colspan=\"5\">".
$HTTP_SESSION_VARS["formVars"]["UpdateEventsMsg"]."</td></tr>";
        $bar .= "</table>\n</form>";
        return $bar;
    }

    function personFormBlock($partner=FALSE)
    {
# TODO: Tabbing order

        GLOBAL $SD_elements;
        GLOBAL $SD_allow_TBAs;
        GLOBAL $SD_prime_is_always_leader;
        GLOBAL $formVars;
        GLOBAL $CIB_WEB_PATH,$unixname;
        if (!is_array($SD_elements))
```

```
            return "The competition coordinator must run PrepStep to setup
on-line registration";
        $cols = array();
        $tab=1;
        $form="";
        if ($partner)
        {
          $tab+=10;
            if ($SD_allow_TBAS || $SD_prime_is_always_leader)
            {
                $form .= "<tr>\n";
            }
            if ($SD_allow_TBAs)
            {
                $form .= "\t<tr><td>".$this->displayFormTBAElement(
$formVars["FirstName"],$tab++) . "</td>\n";
                $cols[0]++;
            }
            if ($SD_allow_TBAs && !$SD_prime_is_always_leader)
            {
                $form .= "\t<td> </td>\n";
                $cols[0]++;
            }
            if (!$SD_prime_is_always_leader)
            {
              $form.= "\t<td>".$this->displayFormLeadingElement(
$formVars["Leading"],$tab++)."</td>\n";
              $cols[0]++;
            }

            # remember role is defined from the PRIME'S_POV
____________if_($SD_prime_is_always_leader)
________________$form.="\t<input_type=\"hidden\"_name=\"role\"
value=\"leading\">\n";
____________if_($SD_allow_TBAS_||_$SD_prime_is_always_leader)
____________{
________________$form_.="</tr>\n";
____________}
________}
________$baseRow_=_isset($cols[0])?0:1;

________$needsName_=_in_array("Name",$SD_elements);
________$needsEmail_=_in_array("Email",$SD_elements);
________$needsFee_=_in_array("Fee_Category",$SD_elements);
________$needsPackage_=_in_array("Package",$SD_elements);
________$needsID_=_in_array("ID_number",$SD_elements);
________$needsAffil_=_in_array("Affiliation",$SD_elements);
________$needsPhone_=_in_array("Phone_Number",$SD_elements);
________$needsAddress_=_in_array("Address",$SD_elements);

________if_($needsName_||_$needsEmail)
________{
____________$cols[$baseRow]++;
____________$form_.="<tr><td_valign=\"top\"><table>\n";
________}
________if_($needsName)
________{
____________$form_.="\t<tr><td><span_class=\"FormItemNames\">
First_Name</span></td></tr>\n";
____________$form_.="\t<tr><td><input_type=\"text\"
tabindex=".$tab++."_name=\"FirstName\"_value=\"";
```

```
            $form .= $formVars["FirstName"]."\"></td></tr>\n";
            $form .= "\t<tr><td><span
class=\"FormItemNames\">Last Name</span></td></tr>\n";
            $form .= "\t<tr><td><input type=\"text\" tabindex=".$tab++.
" name=\"LastName\" value=\"";
            $form .= $formVars["LastName"]."\"></td></tr>\n";

        }
        if ($needsEmail)
        {
            global $preloadEmail;
            if ($formVars["Email"]=="" && $preloadEmail!="")
                $formVars["Email"]=$preloadEmail;
            $form .= "\t<tr><td><span class=\"FormItemNames\">
Email</span></td></tr>\n";
            $form .= "\t<tr><td><input type=\"text\" tabindex=".$tab++.
" name=\"Email\" value=\"";
            $form .= $formVars["Email"]."\">\n";
            $form .= "</td></tr>\n";
            $form .= "\t<tr><td><input type=\"checkbox\" name=\"ignoreEmail\" ";
            $form .= ($preloadEmail=="" ? "" : "checked=\"checked\" ");
            $form .= "/> Partner/team uses same email</td></tr>\n";

        }
        if ($needsName || $needsEmail)
        {
            $form.="</table></td>\n";
        }
        if (isset($cols[$baseRow]))
        {
            $form.="<td> </td>";
            $cols[$baseRow]++;
        }
        if ($needsFee || $needsPackage || $needsID)
        {
            if (!isset($cols[$baseRow]))
                $form .= "<tr>";
            $cols[$baseRow]++;
            $form .= "<td valign=\"top\"><table>\n";
        }
        if ($needsFee)
        {
            $feeElement = $this->displayFormAgeElement("Age",
$formVars["Age"],$tab++);
            if (strpos($feeElement,"hidden")===FALSE)
            {
                $form .= "\t<tr><td><span class=\"FormItemNames\">
Fee Category</span></td></tr>\n";
                $form .= "\t<tr><td>$feeElement</td></tr>\n";
            }
            else
                $form .= $feeElement;

        }
        if ($needsPackage)
        {
            $packageElement = $this->displayFormPackageElement("Package",
$formVars["Package"],$tab++);
            if (strpos($packageElement,"hidden")===FALSE)
            {
                $form .= "\t<tr><td><span class=\"FormItemNames\">
```

```
Package</span></td></tr>\n";
                $form .= "\t<tr><td>$packageElement</td></tr>\n";
            }
            else
                $form .= $packageElement;

        }
        if ($needsID)
        {
            $form .= "This feature not set up yet";
        }
        if ($needsFee || $needsPackage || $needsID)
        {
            $cols[$baseRow]++;
            $form .= "</table></td>\n";
        }
        if ($cols[$baseRow])
        {
            $form.="<td> </td>";
            $cols[$baseRow]++;
        }
        if ($needsAffil)
        {
            $form .= "<td valign=\"top\"><table>";
            $form .= "<tr><td colspan=\"2\"><span class=\"FormItemNames\">
Affiliation</span></td></tr>\n";
            $incAffils["Select your affiliation"]="choose";
            include("$CIB_WEB_PATH/stats/$unixname/affils.inc");
            $form .= "<tr><td colspan=\"2\">";
            $form .= $this->arrayToSelect("AffilID",$incAffils,
$formVars["AffilID"],1,$tab++);
            $form .= "</td></tr>";
            $form .= "<tr><td colspan=\"2\"><span class=\"FormItemNames\">
New Affiliation</span></td></tr>\n";
            $form .= "<tr><td>  </td><td><span
class=\"FormItemNames\">Full Name</span></td></tr>\n";
            $form .= "<tr><td>  </td><td><input type=\"text\"
tabindex=".$tab++." name=\"NewAffil\" value=\"".$formVars["NewAffil"].
"\"></td></tr>\n";
            $form .= "<tr><td>  </td><td><span
class=\"FormItemNames\">Abbreviation/Short form (10 characters or
less)</span></td></tr>\n";
            $form .= "<tr><td>  </td><td><input type=\"text\"
tabindex=".$tab++." name=\"AffilShort\" value=\"".$formVars["AffilShort"].
"\"></td></tr>\n";
            $form .= "</table></td></tr>";
        }
        $mainBody = $cols[$baseRow];
        if ($needsAddress)
        {
            $form .= "<tr><td colspan=$mainBody>This feature not
implemented</td></tr>\n";
        }
        if ($needsPhone)
        {
            $form .= "<tr><td colspan=$mainBody>This feature not
implemented</td></tr>\n";
        }
        return $form;
    }
```

```
function conflictInfoCell($personID,$formVars="")
{
    GLOBAL $SD_elements;

    $update = ($formVars!="");
    $person = new Competitor($this->db);
    $person->setID($personID);
    $person->retrieve();
    if (in_array("Name",$SD_elements))
    {
        if ($update)
        {
            $cell  = $formVars["FirstName"] . " ";
            $cell .= $formVars["LastName"] ."<br>";
        }
        else
        {
            $cell  ="<b>".$person->getFirstName()." ";
            $cell .= $person->getLastName() . "</b><br>";
        }
    }
    if (in_array("Affiliation",$SD_elements))
    {
        if ($update)
        {
            $org = new Organization($this->db);
            $org->setID($formVars["AffilID"]);
            $org->retrieve();
            $cell .= $org->getAbbrev() . "<br>";
        }
        else
        {
            $cell .= $person->org->getAbbrev() . "<br>";
        }
    }
    if (in_array("Email",$SD_elements))
    {
        if ($update)
            $cell .= $formVars["Email"] . "<br><br>";
        else
        {
            $email = $person->getEmail();
            $cell .= substr($email,0,strpos($email,"@")+1) . "...<br><br>";
        }
    }
    $cell .= "Dancing with:<br>";
    $partners = $person->getPartners();
    $numPartners = count($partners);
    for ($i=0;$i<$numPartners;$i++)
    {
        $partner = new Person($this->db);
        $partner->setID($partners[$i]);
        $partner->retrieve();
        $cell .= $partner->getFirstName() ." ";
        $cell .= $partner->getLastName() . "<br>";
    }
    if ($numPartners==0 && !$update)
    {
        $cell .= "No partner's entered yet, select the ";
        $cell .= "'Register/Add' button below to add a partner.";
    }
```

```
    if (!$update)
    {
        $cell .= '<form action="partners.php" method="POST">'."\n";
        $cell .= '<input type="hidden" name="primeID" value="'.$personID."\">\n";
        $cell .= '<input type="submit" value="Review/Add Registrations">';
        $cell .= "</form>";
    }
    return $cell;
}

function conflictInfoUpdate($primeID,$formVars="")
{
    GLOBAL $SD_elements;

    $cell = "<input type=\"hidden\" name=\"primeID\" value=\"$primeID\">\n";
    if (in_array("Name",$SD_elements))
    {
        $cell .="<input type=\"hidden\" name=\"FirstName\" value=\"";
        $cell .= $formVars["FirstName"]."\">\n";
        $cell .="<input type=\"hidden\" name=\"LastName\" value=\"";
        $cell .= $formVars["LastName"]."\">\n";
    }
    if (in_array("Email",$SD_elements))
    {
        $cell .="<input type=\"hidden\" name=\"Email\" value=\"";
        $cell .= $formVars["Email"]."\">\n";
    }
    if (in_array("Fee Category",$SD_elements))
    {
        $cell .="<input type=\"hidden\" name=\"Age\" value=\"";
        $cell .= $formVars["Age"]."\">\n";
    }
    if (in_array("Package",$SD_elements))
    {
        $cell .="<input type=\"hidden\" name=\"Package\" value=\"";
        $cell .= $formVars["Package"]."\">\n";
    }
    if (in_array("Affiliation",$SD_elements))
    {
        $cell .="<input type=\"hidden\" name=\"AffilID\" value=\"";
        $cell .= $formVars["AffilID"]."\">\n";
        $cell .="<input type=\"hidden\" name=\"NewAffil\" value=\"";
        $cell .= $formVars["NewAffil"]."\">\n";
        $cell .="<input type=\"hidden\" name=\"AffilShort\" value=\"";
        $cell .= $formVars["AffilShort"]."\">\n";
    }
    return $cell;
}

function partnerTable($person,$needForm=0,$selectedCouple=0)
{
    GLOBAL $SD_elements, $formVars,$SD_button_style;
    $table = "";
    $table .= "<h3>Registration Summary</h3>";
    $primeID = $person->getID();
    #TODO: INFO CELL needed here
    $fname = $person->getFirstName();
    $lname = $person->getLastName();
    if (count($person->getPartners()))
    {
        $table .= "You are currently registered with the following
```

```
partners.  The Drop Partner button is used to cancel your registration with
that row's partner.  The Update button will load the partner's identification
information and current registration into the form below for editting.\n";
            $table .= "<table>\n";
               include("htmlDisplay_partnerTable_header.inc");
            $partners = $person->getPartners("follow");
            $numPartners = count($partners);
            if ($needForm)
                $totalDances=array();
            for ($i=0;$i<$numPartners;$i++)
            {
                $partner = new Person($this->db);
                $partner->setID($partners[$i]);
                $partner->retrieve();
        #TODO: INFO CELL needed here
                $partnerFirst = $partner->getFirstName();
                $partnerLast = $partner->getLastName();
                $org = new Organization($this->db);
                $org->setID($partner->getAffilID());
                $org->retrieve();
                $partnerAffil = $org->getAbbrev();
                $tempCouple = new Couple($this->db);
                $tempCouple->setLeader($primeID);
                $tempCouple->setFollower($partners[$i]);
                $coupleid = $tempCouple->searchDB();
                if (count($coupleid)!=1) die ("Should never happen");
                $tempCouple->setID($coupleid[0]);
                $tempCouple->retrieve();
                if ($needForm)
                    $totalDances=arrayJoin($totalDances,
$tempCouple->getDances());
                $table .= "\t<tr>\n\t\t<td><form
action=\"SCRIPTS/partners-dispatch.php\"
method =\"POST\">\n\t\t\t<input type=\"hidden\" name=\"action\"
value=\"update\">\n\t\t\t<input type=\"hidden\" name=\"primeID\"
value=\"$primeID\"><input type=\"hidden\" name=\"coupleid\"
value=\"$coupleid[0]\">\n\t\t\t<input type=\"submit\"
value=\"Update\" style=\"background: #d9a\"></form></td>\n";
                $table .= "\t\t<td><form action=\"SCRIPTS/partners-dispatch.php\"
method =\"POST\">\n\t\t\t<input type=\"hidden\" name=\"action\"
value=\"drop\">\n\t\t\t<input type=\"hidden\" name=\"coupleid\"
value=\"$coupleid[0]\"><input type=\"hidden\" name=\"primeID\"
value=\"$primeID\">\n\t\t\t<input type=\"submit\"
value=\"Drop\" style=\"background: #d9a\"></form></td>\n";
                $table .= "\t\t<td>leading <b>$partnerFirst 
$partnerLast</b> ($partnerAffil)</td>\n";
                include("htmlDisplay_partnerTable_row.inc");
                $table .= "\t</tr>\n";
            }
            $partners = $person->getPartners("lead");
            $numPartners = count($partners);
            for ($i=0;$i<$numPartners;$i++)
            {
                $partner = new Person($this->db);
                $partner->setID($partners[$i]);
                $partner->retrieve();
        #TODO: INFO CELL needed here
                $partnerFirst = $partner->getFirstName();
                $partnerLast = $partner->getLastName();
                $org = new Organization($this->db);
                $org->setID($partner->getAffilID());
```

```
                $org->retrieve();
                $partnerAffil = $org->getAbbrev();
                $tempCouple = new Couple($this->db);
                $tempCouple->setLeader($partners[$i]);
                $tempCouple->setFollower($primeID);
                $coupleid = $tempCouple->searchDB();
                if (count($coupleid)!=1) die ("Should never happen");
                $tempCouple->setID($coupleid[0]);
                $tempCouple->retrieve();
                if ($needForm)
                    $totalDances=arrayJoin($totalDances,$tempCouple->getDances());
                $table .= "\t<tr>\n\t\t<td><form
action=\"SCRIPTS/partners-dispatch.php\" method =\"POST\">\n\t\t\t<input
type=\"hidden\" name=\"action\" value=\"update\">\n\t\t\t<input
type=\"hidden\" name=\"coupleid\" value=\"$coupleid[0]\">\n\t\t\t
<input type=\"hidden\" name=\"primeID\" value=\"$primeID\">
<input type=\"submit\" value=\"Update\" style=\"background: #d9a\">
</form></td>\n";
                $table .= "\t\t<td><form
action=\"SCRIPTS/partners-dispatch.php\" method =\"POST\">\n\t\t\t
<input type=\"hidden\" name=\"action\" value=\"drop\">\n\t\t\t<input
type=\"hidden\" name=\"coupleid\" value=\"$coupleid[0]\">\n\t\t\t<input
type=\"hidden\" name=\"primeID\" value=\"$primeID\"><input
type=\"submit\" value=\"Drop\" style=\"background: #d9a\"></form></td>\n";
                $table .= "\t\t<td>following <b>$partnerFirst
$partnerLast</b> ($partnerAffil)</td>\n";
                include("html1Display--partnerTable--row.inc");
                $table .= "\t</tr>\n";
            }
            $table .= "\t<tr>\n";
            $table .= "\t\t<td colspan=\"3\"><form
action=\"SCRIPTS/partners-dispatch.php\" method=\"POST\">\n\t\t\t<input
type=\"hidden\" name=\"primeID\" value=\"$primeID\"><input type=\"hidden\"
name=\"action\" value=\"cancel\">\n\t\t\t<input type=\"submit\"
value=\"Cancel All Registrations\" style=\"background: #d9a\"></form>
</td><td colspan=\"4\">This button will withdraw you from the competition,
but leave any events your partners compete in with other partners untouched.
</td>\n\t</tr>\n";

            $table .= '</table>New partners may be added in the form below
without using any buttons from this table.</td></tr></table><br>';
        }
        else
            $table .= "Welcome to our comp.  You are currently not registered
with any partners.</td></tr></table>";
        if (!$needForm) return $table;
        global $HTTP_SESSION_VARS;
        $table .= "<a name=\"reg\"> </a>";
        #:TODO: Fix error handling
        if (isset($HTTP_SESSION_VARS["formVars"]["FirstNameMsg"]) ||
            isset($HTTP_SESSION_VARS["formVars"]["LastNameMsg"]) ||
            isset($HTTP_SESSION_VARS["formVars"]["AffilMsg"]) ||
            isset($HTTP_SESSION_VARS["formVars"]["NewAffilMsg"]) ||
            isset($HTTP_SESSION_VARS["formVars"]["AgeMsg"]) ||
            isset($HTTP_SESSION_VARS["formVars"]["PackageMsg"]) ||
            isset($HTTP_SESSION_VARS["formVars"]["EventsMsg"]) ||
            isset($HTTP_SESSION_VARS["formVars"]["EmailMsg"]))
        {
            $table.= "<br />";
            $table.=  "<table bgcolor=#f0f090 border=1 cellpadding=8
 width=75%><tr><td>";
```

```
    $table.=  $HTTP_SESSION_VARS["formVars"]["FirstNameMsg"];
    $table.=  $HTTP_SESSION_VARS["formVars"]["LastNameMsg"];
    $table.=  $HTTP_SESSION_VARS["formVars"]["EmailMsg"];
    $table.=  $HTTP_SESSION_VARS["formVars"]["AgeMsg"];
    $table.=  $HTTP_SESSION_VARS["formVars"]["PackageMsg"];
    $table.=  $HTTP_SESSION_VARS["formVars"]["AffilMsg"];
    $table.=  $HTTP_SESSION_VARS["formVars"]["NewMsg"];
    $table.=  $HTTP_SESSION_VARS["formVars"]["EventsMsg"];
    $table.=  '</td></tr></table>';
}

$table .= '<table bgcolor="white" border=1 cellpadding=8 width=95%>';
$table .= "<tr><td>";
$table .= "<h3>Registration for $fname $lname with ";
if ($selectedCouple!=0)
{
    $couple = new Couple($this->db, $selectedCouple);
    $couple->setID($selectedCouple);
    $couple->retrieve();
    if ($couple->getLeader()==$person->getID())
    {
        $leading = TRUE;
        $partner = new Person($this->db);
        $partner->setID($couple->getFollower());
    }
    elseif ($couple->getFollower()==$person->getID())
    {
        $leading = FALSE;
        $partner = new Person($this->db);
        $partner->setID($couple->getLeader());
    }
    else die("This should never happend.  HTMLDisplay::Partners");
    $partner->retrieve();
    $pID = $partner->getID();
    $GLOBALS["formVars"]["Leading"]=$leading;
    $needName = in_array("Name",$SD_elements);
    $needEmail = in_array("Email",$SD_elements);
    $needFee = in_array("Fee",$SD_elements);
    $needPackage = in_array("Package",$SD_elements);
    $needAffiliation = in_array("Affiliation",$SD_elements);
    $needPhone = in_array("Phone",$SD_elements);
    $needID = in_array("ID",$SD_elements);
    $needAddress = in_array("Address",$SD_elements);
    if ($needName)
    {
        $pFName  = $partner->getFirstName();
        $pLName = $partner->getLastName();
        $idLabel = "<i>$pFName $pLName</i></h3>";
        if (""==$formVars["FirstName"])
            $formVars["FirstName"]=$pFName;
        if (""==$formVars["LastName"])
            $formVars["LastName"]=$pLName;
    }
    if ($needEmail)
    {
        $pEmail = $partner->getEmail();
        if (""==$idLabel)
            $idLabel .= "<i>$pEmail</i></h3>";
        if (""==$formVars["Email"])
            $formVars["Email"]=$pEmail;
    }
```

```
            if ($needFee)
                if (""==$formVars["Age"])
                    $formVars["Age"]=$partner->getAgeLevel();
            if ($needPackage)
                if (""==$formVars["Package"])
                    $formVars["Package"]=$partner->getPackage();
            if ($needAffiliation)
                if (""==$formVars["AffilID"])
                    $formVars["AffilID"]=$partner->getAffilID();
#TODO: ADD ID, address, phone support
        }
        else
        {
            $partner=0;
            $idLabel = "<i>New Partner</i></h3>";
            $role = $HTTP_SESSION_VARS["formVars"]["Role"];
            if ($role=="") $role="leading";

            $formVars["Leading"]=("leading"==$role);
            $prime = new Person($this->db);
            $prime->setID($primeID);
            $prime->retrieve();
            if (in_array("Affiliation",$SD_elements))
                $primeAffil = $prime->getAffilID();
            if ($formVars["AffilID"]=="") $formVars["AffilID"]=$primeAffil;
        }
        $table .= $idLabel;
        $table .= "<form action=\"SCRIPTS/partner-check.php\" method=\"POST\">";
        $table .= "<input type=\"hidden\" name=\"primeID\" value=\"$primeID\">";
        $table .= "<table>";
        if ($selectedCouple!=0)
        {
            $table .= "<input type=\"hidden\" name=\"partnerID\" value=\"$pid\">";
            $table .= "<input type=\"hidden\" name=\"coupleid\"
value=\"$selectedCouple\">\n";
        }

        $table .= $this->personFormBlock(TRUE);
        $table .= "</table>";
        $table .= $this->displayFormEventsTable($totalDances,$person,$couple);
        $table .= "<table><tr><td><input type=\"submit\" name=\"action\" value=\"";
        if ($selectedCouple!=0)
            $table .= "Update This Partner";
        else
            $table .= "Register This Partner";
        $table .= "\" style=\"$SD_button_style\"><input type=\"reset\"
value=\"Clear Partner Registration Form\" style=\"$SD_button_stlye\"></td></tr>";
        $table .= "</table>";
        $table .= "</form>";
        return $table;

    }

    function teamNavBar($components,$currentRequest)
    {
        $bar ="";
        reset($components);
        $first=TRUE;
        while ($aComponent = each($components))
        {
            if (!$first)
```

```
            $bar .= " | ";
        $first = FALSE;
        $name = $aComponent["value"];
        switch ($name)
        {
            case "Top" :
                if ($currentRequest=="top")
                $bar .= "<b>Top</b>";
            else
                $bar .= "<a href=\"#top\">Top</a>";
            break;
            case "Contact" :
                if ($currentRequest=="contact")
                $bar .= "<b>Contact Information</b>";
            else
                $bar .= "<a href=\"#contact\">Contact Information</a>";
            break;
            case "Setup" :
                if ($currentRequest=="setup")
                $bar .= "<b>Update/Setup</b>";
            else
                $bar .= "<a href=\"#setup\">Update/Setup</a>";
            break;
            case "TeamMatch" :
                if ($currentRequest=="teammatch")
                $bar .= "<b>Team Match</b>";
            else
                $bar .= "<a href=\"#teammatch\">Team Match</a>";
            break;
            case "Invoice" :
                if ($currentRequest=="invoice")
                $bar .= "<b>Registrations</b>";
            else
                $bar .= "<a href=\"#invoice\">Registrations</a>";
            break;
            case "Results" :
                if ($currentRequest=="results")
                $bar .= "<b>Results</b>";
            else
                $bar .= "<a href=\"#results\">Results</a>";
            break;
          default: break;
        }
    }
    $bar .= "<br>";
    return $bar;
}

function teamInfoUpdateBar($org)
{
    $orgID = $org->getID();
    $updateType = ($orgID==0?"INSERT":"UPDATE");
    $name = $org->getName();
    $abbrev = $org->getAbbrev();
    $repTitle = $org->getRepTitle();
    $repID = $org->getRep();
#   $coordinator = $org->getCoordinator();
    $repName = $org->getRepString();
    $orgAddr=$org->getBuildingString();
    $addrType="business";
    $repAddr=$org->getRepString();
```

```
        if ($orgAddr=="")
        {
            $orgAddr=$org->getRepAddr();
            $addrType="personal";
        }
        else
            $orgAddr=$org->getOrgAddr();

        $line1 = $orgAddr->getStreet1();
        $line2 = $orgAddr->getStreet2();
        $city = $orgAddr->getCity();
        $state = $orgAddr->getState();
        $zip = $orgAddr->getZip();
        $bar = "<form action=\"SCRIPTS/group-info-update.php\" method=\"POST\">\n";
        $bar .= "<input type=\"hidden\" name=\"type\" value=\"$updateType\">\n";
        $bar .= "<input type=\"hidden\" name=\"orgid\" value=\"$orgID\">\n";
        $bar .= "<a name=\"setup\"><h3>Affiliation Setup</h3></a>";
        $bar .= "<table>\n";
        $bar .= "<tr><td colspan=\"2\">The full name is the name used in
formal listing and announcements, the abbreviation is a short form used in
program listings and on the web page. Some Examples: Massachusetts Institute
of Technology-->MIT;Harvard University-->Harvard; Fred Astaire Dance Studio,
Boston-->FADS Boston</td>";
        $bar .= getFormErrorMessage("TeamName",2,"align=\"center\"");
        $bar .= "\t<tr>\n\t\t<th>Team Name</th>";
        $bar .= "\n\t\t<td><input type=\"text\" name=\"teamname\"
value=\"$name\" size=\"40\"></td></tr>";
        $bar .= getFormErrorMessage("TeamAbbrev",2,"align=\"center\"");
        $bar .= "\t<tr>\n\t\t<th>Team Abbreivation</th>";
        $bar .= "\n\t\t<td><input type=\"text\" name=\"teamabbrev\"
value=\"$abbrev\"></td></tr>";
        $bar .= "<tr><td colspan=\"2\">The Representative is the person that
Sliding Doors will contact with announcements about competition
changes/deadlines to be passed onto their members.</td>";
        $bar .= getFormErrorMessage("RepTitle",2,"align=\"center\"");
        $bar .= "\t<tr>\n\t\t<th>Representative title(President, Owner,
 Captain, etc)</th>";
        $bar .= "\n\t\t<td><input type=\"text\" name=\"reptitle\"
value=\"$repTitle\"></td></tr>";
        $bar .= "\n\t<tr>\n\t\t<th>Representative:</th>";
        $bar .= "\n\t\t<td><select name=\"rep\">";
        $bar .= $this->db->getSelectMembers($orgID,$repID);
        $bar .= "</select></td></tr>";
        $bar .= "<tr><td colspan=\"2\">The affiliation mailing address may be
used to send photo or results after the comp, or registration materials before
the competition. Please indicate if this is a business address or the
representative's home address.</td>";
        ________$bar .= "<tr><th>Address Type:</th><td>Business<input type=\"radio\"
name=\"addrType\" value=\"business\" ".($addrType=="business"?"CHECKED>":">");
        ________$bar .= "Personal<input type=\"radio\" name=\"addrType\"
value=\"personal\" ".($addrType=="personal"?"CHECKED>":">");
        ________$bar .= getFormErrorMessage("Street1",2,"align=\"center\"");
        ________$bar .= "\t<tr>\n\t\t<th>Street, Line 1</th>";
        ________$bar .= "\n\t\t<td><input type=\"text\" name=\"street1\"
value=\"$line1\"></td></tr>";
        ________$bar .= "\t<tr>\n\t\t<th>Street, Line 2</th>";
        ________$bar .= "\n\t\t<td><input type=\"text\" name=\"street2\"
value=\"$line2\"></td></tr>";
        ________$bar .= getFormErrorMessage("City",2,"align=\"center\"");
        ________$bar .= "\t<tr>\n\t\t<th>City</th>";
        ________$bar .= "\n\t\t<td><input type=\"text\" name=\"city\"
```

```
value=\"$city\"></td></tr>";
        $bar .= getFormErrorMessage("State",2,"align=\"center\"");
        $bar .= "\t<tr>\n\t\t<th>State</th>";
        $bar .= "\n\t\t<td><input type=\"text\" name=\"state\"
value=\"$state\"></td></tr>";
        $bar .= getFormErrorMessage("Zip",2,"align=\"center\"");
        $bar .= "\t<tr>\n\t\t<th>Zip</th>";
        $bar .= "\n\t\t<td><input type=\"text\" name=\"zip\"
value=\"$zip\"></td></tr>";
#       $bar .= "<tr><td colspan=\"2\">Registration Coordinators are people
from the affiliation who track the team's registrations.  They can setup
Sliding Doors to email them when members register/change registration and
can remove people from the list of people the team is paying for with the
 team's entrance fee.  Some affiliations may have their Representative do
this job, others may have a seperate person, or even a group of people.
To add additional people simple add the first, and update, when the screen
returns you'll have a block to add a second (and third on the next submit).
If the representative is doing this job, please select them in this box as
well.</td></tr>";
#       if (is_array($coordinator))
#           $numCoords = count($coordinator);
#       else
#           $numCoords=1;
#       for ($i=0;$i<$numCoords+1;$i++)
#       {
#           if ($i!=$numCoords)
#               if ($numCoords==1)
#               {
#                   $coordID=$coordinator;
#                   $email = $org->getEmail();
#                   $digest = $org->getDigest();
#               }
#               else
#               {
#                   $coordID=$coordinator[$i];
#                   $email=$org->getEmail($coordID);
#                   $digest=$org->getDigest($coordID);
#               }
#           else
#           {
#               unset($coordID);
#               unset($email);
#               unset($digest);
#           }
#           $bar .= "\n\t<tr>\n\t\t<th>Registration Coordinator:</th>";
#           $bar .= "\n\t\t<td><select name=\"coord-$i\">
<option value=\"-1\">Nobody</option>";
#           $bar .= $this->db->getSelectMembers($orgID,$coordID);
#           $bar .= "</select></td></tr>";
#           $bar .= "<tr><td>Email Options(must check both boxes for digest):
</td>";
#           $bar .= "<td>Email:<input type=\"checkbox\" name=\"email-$i\" ";
#           $bar .= ($email?"CHECKED>":">");
#           $bar .= " Digest:<input type=\"checkbox\" name=\"digest-$i\" ";
#           $bar .= ($digest?"CHECKED>":">");
#           $bar .= "</td></tr>";
#       }
        $bar .= "<tr><td><input type=\"submit\" name=\"Submit\"
value=\"Submit\" style=\"background: #d9a\"></td></tr></table></form>";
        return $bar;
    }
```

```
    function teamInfoLoginBar($org)
    {
        GLOBAL $SD_compcoord_email, $SD_compcoord_title;
        $orgID = $org->getID();
        $name = $org->getName();
        $abbrev = $org->getAbbrev();
        $repTitle = $org->getRepTitle();
        $repID = $org->getRep();
        $coordinator = $org->getCoordinator();
        $repName = $org->getRepString();
        $orgAddr=$org->getBuildingString();
        $bar="<!-- begin HTMLDisplay::teamInfoLoginBar -->\n";
        $bar.="<h3>$name ($abbrev)</h3>";
        $bar .= "<b><a name=\"contact\">You</a> must login to change any of
the information on this page.</b>\n";
        $bar .= "<table>";
        $infoCell ="<table>";
        if ($repName=="")
            $infoCell .= "<tr><th colspan=\"2\">Organization not set up
yet</th></tr>\n";
        else
        {
            $repName .= "; " . $orgAddr;
            $repName = str_replace(";","<br>",$repName);
            $infoCell.="<tr><th>$repTitle:</th><td>$repName</td></tr>\n";
        }
        if ($coordinator!=0)
        {
                $infoCell .= "<tr><th>Registration Coordinator";
                $infoCell .= (is_array($coordinator)?"s":"") .":</th>";
                $infoCell .= "<td>".$org->getCoordString()."</td></tr>";;
        }
        else
            $infoCell .= "<tr><th colspan=\"2\">No registration coordinators
setup</th></tr>";
        $infoCell .= "</table>";
        $bar .= "<tr><td>$infoCell</td><td>$loginCell</td></tr>";
        $bar .= "</table>";
        $bar .= "<!-- End HTMLDisplay::teamLoginBar -->\n";
        return $bar;
    }

    function teamMatchSummary($affil)
    {
        $affilID = $affil->getID();
        $bar = "";
        $query = "SELECT team_name, shortname FROM team_match_registrations
JOIN events_names ON (eventid=teammatchid) WHERE orgid=$affilID ORDER BY
shortname;";
        $result = $this->db->query($query);
        $numTeams = $result->numRows();
        $bar .= "<a name=\"teammatch\">You</a> have $numTeams team".
($numTeams!=1?"s":"").  " registered.";
        $bar .= "You can edit your teams from the organization login page.";
        for ($i=0;$i < $numTeams;$i++)
        {
            list($teamName,$teamMatch)=$result->getRowAt($i);
            $bar .="<br><b>$teamName</b> ($teamMatch)";
        }
```

```
        return $bar."<br>";
    }


    function teamRegistration($affil,$update=FALSE)
    {   $bar="";
        $affilID = $affil->getID();
        $status=$this->db->getStatus();
        $shortName = $affil->getAbbrev();
        $safeShortName = str_replace(" ","_",$shortName);
#       if ($status=="OPEN")
#           $bar .= "<b>Check back after the registration closes for a link
to a finalized invoice</a>.</b>";
#       elseif ($status=="CLOSED")
            $bar.= "Click here for a <font size=+1><b><a href=\"Invoices/".
$safeShortName."_invoice.pdf\">Printer Friendly Invoice</a></b></font>";

        $packagesPurchased=$this->db->produceRegistrantArray($affilID);

        $sponsored=FALSE;
        $nonSponsored=FALSE;
        $outsider=FALSE;
        $query=<<<END_QUERY
SELECT packages.packageid, category
FROM packages NATURAL JOIN packages_cost
ORDER BY category;
END_QUERY;
        $result = $this->db->query($query);
        $numOptions = $result->numRows();
        for ($i=0;$i<$numOptions;$i++)
          {
            list($id,$cat)=$result->getRowAt($i);
            if (!$sponsored &&
                count($packagesPurchased[$id][$cat]["Sponsored"]["Entries"]))
              $sponsored=TRUE;
            if (!$nonSponsored &&
                count($packagesPurchased[$id][$cat]["Non"]["Entries"]))
              $nonSponsored=TRUE;
            if (!$outsider &&
                count($packagesPurchased[$id][$cat]["Outsider"]["Entries"]))
              $outsider=TRUE;
          }
        if ($update && ($sponsored || $nonSponsored || $outsider))
        {
            $bar .= "<form action=\"SCRIPTS/group-reg-update.php\"
method=\"POST\">";
            $bar .= "This page let you \"remove\" inidividuals from your
team's invoice.  This should be used only when there are people from your
affiliation who are not part of your team.  If you team isn't paying for
its team members, we still ask that the team collect the registration fees
and submits the fee along with the invoice.";
        }
        $bar .= "<table><tr><th colspan=\"4\"><a name=\"invoice\">Current</a>
Entries and Preliminary Invoice</th></tr>";
        if (!($sponsored || $nonSponsored || $outsider))
        {
            $bar .= "<tr><th align=\"center\" colspan=\"4\">
No entries at this time</th></tr>";
        }
        if ($sponsored)
        {
```

```
            $bar .="<tr><th colspan=\"4\">Affiliation Sponsored Individuals</th></tr>";
            $bar .= $this->teamRegSections($packagesPurchased,"Sponsored",$update);
        }
        if ($nonSponsored)
        {
            $bar .= "<tr><th colspan=\"4\">Affiliated, non-Sponsored
Individuals</th></tr>";
            $bar .= $this->teamRegSections($packagesPurchased,"Non",$update);
        }
        if ($outsider)
        {
            $bar .= "<tr><th colspan=\"4\">Different Affiliation Partners</th></tr>";
            $bar .= $this->teamRegSections($packagesPurchased,"Outsider",$update);
        }
        if ($update && ($sponsored || $nonSponsored || $outsider))
            $bar .= "<tr><td><input type=\"hidden\" name=\"affil\"
value=\"$affilID\"><input type=\"submit\" name=\"Submit\" value=\"Submit\"
style=\"background: #d9a\"></td></tr>";
        $bar .= "</table>";
        if ($update && ($sponsored || $nonSponsored || $outsider))
            $bar .= "</form>";
        $status=$this->db->getStatus();
#       if ($status=="OPEN")
#           $bar .= "<b>Check back after the registration closes for a link
to a formal invoice.</b>";
#       elseif ($status=="CLOSED")
            $bar.= "Click here for a <font size=+1><b><a href=\"Invoices/".
$safeShortName."_invoice.pdf\">Printer Friendly Invoice</a></b></font>";
#       echo htmlspecialchars($bar);
        return $bar."<br>";
    }

    function teamRegSections($registrants,$type,$update)
    {
#       echo printArray($registrants);
        $sponsored = ($type=="Sponsored");
        $bar="";
        reset($registrants);
        $total=0;
        while ($package=each($registrants)) {
            reset($package["value"]);
            while ($section=each($package["value"])) {
                $sectionData = $section["value"][$type];
                $label = $sectionData["Name"] . " -- " .$section["key"];
                $cost = $sectionData["Cost"];
                $entries = $sectionData["Entries"];
                $numEntries = count($entries);
                if ($numEntries)
                {
                    $bar .= "<tr><th>Competitor</th><th colspan=\"2\">
$label Entries</th><th>Cost</th></tr>\n";
                    $bar .= "<tr><th>Number</th><th>Name</th><th>
Partners</th><th></th>";
                    if ($update)
                        $bar .= "<th>Team Pays</th><th>Individual Pays</th></tr>";
                    else
                        $bar .= "</tr>";
                    $subtotal=0;
                    for ($j=0;$j<$numEntries;$j++)
                    {
                        $person = $entries[$j];
```

```
                        $name = $person["Name"];
                        $paid = $person["Paid"];
                        $id = $person["ID"];
                        if (!$paid)
                            $subtotal+=$cost;
                        $partners = arrayToCSL($person["Partners"]);
                        $numbers = arrayToCSL($person["Numbers"]);
                        $bar .= "<tr><td>$numbers</td><td>$name</td>
<td>$partners</td><td align=\"right\">".($paid?"PAID":"\$$cost")."</td>";
                        if ($update)
                        {
                            $bar.="<td><input type=\"radio\"
name=\"people-$id\" value=\"sponsored\" ";
                            $bar.=($sponsored?"CHECKED":"").">";
                            $bar.="<td><input type=\"radio\"
name=\"people-$id\" value=\"not-sponsored\" ";
                            $bar .=(!$sponsored?"CHECKED":"").">";
                        }
                        else
                            $bar .="</tr>";
                    }
                    if ($sponsored)
                        $bar .= "<tr><th colspan=\"3\" align=\"right\">
$label Subtotal</th><th align=\"right\">\$$subtotal</th></tr>\n";
                    $total+=$subtotal;
                }
                else # no entries with this package
                {
                    $bar .= "<tr><th colspan=\"4\">No $label entries</th></tr>";
                }
            }
        }
        if ($sponsored)
            $bar .= "<tr><th colspan=\"3\" align=\"right\">
Affiliation Owes:</th><th align=\"right\">\$$total</th></tr>\n";
        return $bar;
    }

    function teamResults($afiil)
    {
        if ($this->db->getStatus()!="FINISHED")
            return "<b><a name=\"results\">Check</a> back after the comp for
your team's results</b><br>";
        else
            return "<b><a name=\"results\">This</a> feature not implemented
yet</b><br>";
    }

    function nameCollisionTable($person)
    {
        return "";
    }

#############################################################
# Function: emailCollisionTable                             #
# Arguments: matches --- the ids of the matching emails     #
#            target --- where the form submits to           #
# Returns: A table/form string that gives the email collisions#
#          to the user for user clarification               #
# Requires: nothing                                         #
# Modifies: nothing                                         #
```

```
#############################################################
    function emailCollisionTable($matches, $target)
    {
        $result  ="<TABLE>\n";
        $result .= "\t<TR>\n";
        $result .= "\t\t<TH>Name</TH>\n";
        $result .= "\t\t<TH>Affiliation</TH><TH>Partners</TH><TH>Select</TH>\n";
        $result .= "\t</TR>\n";

        $num = count($matches);
        for ($i=0;$i<$num;$i++)
        {
            $person = new Competitor($this->db);
            $person->setID($matches[$i]);
            $person->retrieve();
            $partners = $person->getPartners();
            $first = $person->getFirstName();
            $last = $person->getLastName();
            $affil = $person->getAffil();

            $result .= "\t<TR>\n";
            $result .= "\t\t<TD>$first $last</TD><TD>$affil</TD>\n";
            $result .= "\t\t<TD>";
            $numPartners = count($partners);
            if ($numPartners==0)
            {
                $result .= "None entered yet";
            }
            for ($j=0;$j<$numPartners;$j++)
            {
                $aPartner = new Person($this->db);
                $aPartner->setID($partners[$j]);
                $aPartner->retrieve();
                $partnerFirst = $aPartner->getFirstName();
                $partnerLast = $aPartner->getLastName();
                $result .= "$partnerFirst $partnerLast <BR>";
            }
            $result .= "\t\t</TD>\n";
            $result .= "\t\t<TD><FORM ACTION=$target METHOD=POST>\n";
            $result .= "\t\t\t<INPUT TYPE=HIDDEN NAME=CHOICE VALUE=$matches[$i]>\n";
            $result .= "\t\t\t<INPUT TYPE=SUBMIT NAME=USE VALUE=\"Use Me\"
style=\"background: #d9a\"></FORM></TD>\n";
            $result .= "\t</TR>\n";
        }
        $result .= "</TABLE>\n";
        return $result;
    }

    function getAffilListChoices()
    {
        return "";
    }

    function getAffilListChoicesSelect($chosen)
    {
        return "";
    }

    function colorDances($dances)
    {
        $result = "";
```

```
        $numDances= count($dances);
        $first=TRUE;
        for ($i=0;$i<$numDances;$i++)
        {
            $levelSep = strpos($dances[$i],'-');
            $level=substr($dances[$i],0,$levelSep);
            $color="";
            include"htmlDisplay_colorDances.inc";
            if(!$first)$result.=",";
            $result.="<font color=\"$color\">$dances[$i]</font>";
            $first=FALSE;
        }
        return $result;
    }

    function colorLegend()
    {
        $result="";
        $result.='<table width="100%"><tr>';
        include("htmlDisplay_colorLegend.inc");
        $result.='</tr></table>';
        return $result;
    }

    function displayFormTBAElement($name,$tab)
    {
        $element="";
        $element.="<span class=\"FormItemNames\">Partner is</span>
<input tabindex=$tab type=\"checkbox\" name=\"tba\" value=\"true\" ";
        $element.=$name=="TBA"?"CHECKED":"";
        $element.=">TBA";
        return $element;
    }

    function displayFormLeadingElement($leading,$tab)
    {
        if($leading==="")$leading=TRUE;
        $element="";
        $element.="<span class=\"FormItemNames\"><b>Partner</b> is</span>
<input tabindex=$tab type=\"radio\" name=\"role\" value=\"leading\" "
($leading?"CHECKED":"")." >Following<input type=\"radio\"
name=\"role\" value=\"following\" ($leading?"":"CHECKED")." >Leading";
        return $element;
    }

    function displayFormElement($name,$value,$tab,$displayName="",$span=2)
    {
        if($displayName=="")
            $displayName=$name;
        $element.="<tr>";
        $element.="<td><span class=\"FormItemNames\">$displayName</span></td>";
        $element.="<td><input type=\"text\" tabindex=$tab name=\"$name\"
value=\"$value\"></td>";
        $element.="</tr>\n";
        return $element;
    }

    function displayFormAgeElement($name,$chosen,$tab)
    {
        global $baseURL,$basePath;
        GLOBAL $SD_default_age;
```

```
        if ($chosen=="") $chosen=$SD_default_age;
        $ages="";
        include("htmlDisplay__feeCategory.inc");
        if (count($ages)>1)
            $element = $this->arrayToRadio($name,$ages,
                                           $chosen,1,$tab);
        else
            $element = "<input type=\"hidden\" name=\"$name\"
value=\"$chosen\">\n";

        return $element;
    }

    function displayFormAgeOneLiner($name,$chosen,$tab)
    {
        global $baseURL,$basePath;
        GLOBAL $SD_default_age;
        if ($chosen=="") $chosen=$SD_default_age;

        $ages="";
        include("htmlDisplay__feeCategory.inc");
        if (count($ages)>1)
        {
            $element = "<td><span class=\"FormItemNames\">Fee Category:</span></td>";
            $element .= "<td>" . $this->arrayToSelect($name,$ages,
                                                      $chosen,1,$tab);
            $element .= "</td>";
        }
        else
           $element = "<input type=\"hidden\" name=\"$name\" value=\"$chosen\">\n";
        return $element;
    }


    function displayFormPackageElement($name,$chosen,$tab)
    {
        global $baseURL,$basePath;
        GLOBAL $SD_default_package;
        if ($chosen=="") $chosen=$SD_default_package;

        $packages="";
        include("htmlDisplay__packages.inc");
        if (count($packages)>1)
            $element = $this->arrayToRadio($name, $packages, $chosen,1,$tab);
        else
            $element = "<input type=\"hidden\" name=\"$name\" value=\"$chosen\">\n";
        return $element;
    }

    function displayFormPackageOneLiner($name,$chosen,$tab)
    {
        GLOBAL $baseURL,$basePath;
        GLOBAL $SD_default_package;
        $packages="";
        include("htmlDisplay__packages.inc");
        if ($chosen=="") $chosen=$SD_default_package;

        if (count($packages)>1)
        {
            $element = "<td><span class=\"FormItemNames\">Package:</span></td>";
            $element .= "<td>" . $this->arrayToSelect($name,$packages,
```

```
$chosen,1,$tab) . "</td>";
        }
        else
            $element = "<input type=\"hidden\" name=\"$name\" value=\"$chosen\">\n";
        return $element;
    }


    function displayFormAffilElement($name,$chosen , $tab , $partner=FALSE)
    {
        GLOBAL $CIB_WEB_PATH,$unixname;
        $element .= "<tr>";
        $element .= "<td>Affiliation:</td>";
#       $element .= "<td><select name=\"$name\" tabindex=$tab size=\"1\">\n\t";
        $element .= "<td>";
        $newValue = "choose";
        $newName = "Select ";
        if ($partner)
            $newName .= "partner's ";
        else
            $newName .= "your ";
        $newName .= "affiliation";
        $incAffils[$newName]=$newValue;
        include("$CIB_WEB_PATH/stats/$unixname/affils.inc");

        if (formValue("Affil")!="" && formValue("Affil")!=-1)
        {
            $chosen=formValue("Affil");
            $org = new Organization($this->db);
            $org->setID($chosen);
            $org->retrieve();
            $chosen=$org->getName();
        }
        $element .= $this->arrayToSelect($name,$incAffils,$chosen,1);
#       $element .= $this->db->getAffilListChoicesSelect($chosen) . "</select></td>";
        $element .= "</td><td>If your affiliation isn't listed, enter it below.</td>";
        $element .="</tr>";
        $element .="<tr><td>New affiliation:</td>";
        $tab++;
        $element .="<td><input type=\"text\" tabindex=$tab name=\"NewAffil\"
value=\"" .formValue("NewAffil")."\"></td>";
        $element .="<td>Abbreviation/short form(10 characters or less)</td>";
        $tab++;
        $element .="<td><input type=\"text\" tabindex=$tab
name=\"AffilShort\" value=\"" .formValue("AffilShort")."\"></td>";
        $element .="</tr>";
        return $element;
    }

    function displayFormEventsTable($dances,$person,$couple=0)
    {
        global $basePath,$baseURL;
        $table ="Check the event you wish to enter with this partner. Gray
events are events you are already registered for with this or another partner.";
        if($couple!=0)
        {
#           $table .="Yellow events are events that your partner(the one you
#are updating now) is currently registered for with another partner.(not implmented)";
        }
        GLOBAL $formVars;
        include("html1Display_displayFormEventsTable.inc");
```

```
        return $table;
    }


    function arrayToSelect($selName, $choices, $chosen="", $size=1,$tab)
    {
        $selectBox = "<select name=\"$selName\" tabindex=$tab size=\"$size\">\n";
        reset($choices);
        while ($aChoice=each($choices))
        {
            $name = $aChoice["key"];
            $value = $aChoice["value"];
            $selectBox .= "\t<option value=\"$value\" ";
            if ($chosen==$name || $chosen==$value)
                $selectBox .= " selected=\"selected\"";
            $selectBox .= ">$name</option>\n";
        }
        $selectBox .= "</select>\n";
        return $selectBox;
     }

    function arrayToRadio($selName, $choices, $chosen="", $onePerLine=1,$tab)
    {
        $radioBox="";
        reset($choices);
        while ($aChoice=each($choices))
        {
            $name = $aChoice["key"];
            $value = $aChoice["value"];
            $radioBox .= "\t<input type=\"radio\" tabindex=$tab name=\"$selName\" ";
            $radioBox .=  "value=\"$value\"";
            if ($chosen==$name || $chosen==$value)
                $radioBox .= " checked=\"checked\"";
            $radioBox .= ">$name";
            if ($onePerLine)
                $radioBox .= "<br />";
            $radioBox.="\n";
         }
         return $radioBox;
     }


}
?>
```

html-formatting/UIBase.inc

```
<?php
#####################################################################
# This file is part of CompInaBox                                   #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd        #
# CompInaBox is availible for license under the GPL, see            #
# the COPYING file in the root directory of the install for         #
# the full terms of the GPL.                                        #
#                                                                   #
# File: UIBase.inc                                                  #
# Author: Eric D. Nielsen                                           #
# Description: Abstract base class for User Interface               #
#              functionality                                        #
#              This class need to be backfilled later....           #
# Attributes:  db object                                            #
# Constructor: UIBase                                               #
# Methods:     documentHead -- everything before <body>             #
#              beginPage    -- including top heading                #
#              endPage      -- closes page elements                 #
#              sectionBreak -- default <br />                       #
#              developerContactInfo -- developer information        #
# Change Log: 8/05/01 -- created -- edn                             #
#             6/26/02 -- GPL -- edn                                 #
#             8/11/02 -- refactored HTMLDisplay into UIBase         #
#####################################################################
class UIBase
{
    var $db;

    function UIBase($db="")
      {
        $this->db=$db;
      }

    function documentHead($pageTitle="",$styleSheet="",$type="XML")
      {
        if ($type=="XML")
{
        $page = "<?xml version=\"1.0\" encoding=\"iso-8859-1\"?>\n";
        $page.= "<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0 Transitional//EN\"\n";
        $page .= "\t\"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd\">\n";
        $page .= "<html xmlns=\"http://www.w3.org/1999/xhtml\" xml:lang=\"en\" lang=\"en\">\n";
}
else
{
        $page = "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN
 \"http://www.w3.org/TR/html4/loose.dtd\">\n";
        $page .= "<html lang=\"en\">\n";
}
        $page.="<!-- UIBase::documentHead() -->\n";
        $page.="<head>\n";
        $page.="\t<meta http-equiv=\"content-type\"
content=\"text/html; charset=iso-8859-1\" />\n";
        $page.="\t<title>$pageTitle</title>\n";
        if ($styleSheet!="")
          $page.="\t<link rel=\"stylesheet\" href=\"$styleSheet\" />\n";
        $page .= "<style type=\"text/css\">\n";
        $page .= "<!--\n";
        $page .= ".developerinfo {font-size: smaller;}\n";
        $page .= "-->\n";
```

```
        $page .= "</style>\n";
        $page.="</head>\n\n";
        $page.="<!-- End UIBase::documentHead() -->\n";
        return $page;
      }

    function beginPage($pageTitle="",$pageHeader="")
      {
        if ($pageTitle=="")
          $pageTitle="CompInaBox";
        if ($pageHeader=="")
          $pageHeader=$pageTitle;
        $page = $this->documentHead($pageTitle,"");
        $page.="<!-- UIBase::beginPage() -->\n";
        $page.="<!-- Begin Page -->\n";
        $page.="<body bgcolor=\"white\">\n";
        $page.="<h1>$pageHeader</h1>\n";
        return $page;
      }

    function endPage()
      {
        $page ="<!-- UIBase::endPage() -->\n";
        $page.="</body>\n";
        $page.="<!-- End Page -->\n";
        $page.="</html>\n";
        $page.="<!-- End UIBase::endPage() -->\n";
        return $page;
      }

    function sectionBreak()
      {
        $break = "<!-- UIBase::sectionBreak() -->\n<br />\n";
        $break .= "<!-- End UIBase::sectionBreak() -->\n\n";
        return $break;
      }

    function compinaboxBox()
      {
        return $this->developerContactInfo();
      }

    function developerContactInfo($toolname="CompInaBox", $toolURL="")
      {
        $box = "<!-- This is the SD/CIB Developer information block -->\n";
//      $box .= "<font size=\"-1\">\n";
        $box .= "<div class=\"developerinfo\">\n";
        $box .= "Please use the following links to get in touch with the
CompInaBox Developers:\n";
        $box .= "<ul>\n";
        $box .= "\t<li><a
href=\"http://sourceforge.net/tracker/?func=add&amp;group_id=40795&amp;atid=429121\">
\n\t\tBugs</a> -- to report a bug in the s/w</li>\n";
        $box .= "\t<li><a
href=\"http://sourceforge.net/tracker/?func=add&amp;group_id=40795&amp;atid=429122\">
\n\t\tSupport</a> -- for information on hosting your comp on CompInaBox</li>\n";
        $box .= "\t<li><a
href=\"http://sourceforge.net/tracker/?func=add&amp;group_id=40795&amp;atid=429124\">
\n\t\tWishlist</a> -- for new features you'd like to see in the future</li>\n";
        $box.="</ul>\n\n";
        $box.="This page was generated by\n";
```

```
        $box .= "$toolname</a>,\n";
        $box .= "a <a href=\"http://sourceforge.net/projects/compinabox\">";
        $box .= "CompInaBox</a>\ncomponent.  Copyright &copy; 2001-2002,\n";
        $box .= "<a href=\"mailto:nielsene@alum.mit.edu\">Eric D. Nielsen";
        $box .= "</a>\nAll Rights Reserved.\n\n";
        $box .= "<br />\n";
        $box .= "CompInaBox was inspired by Dave Leung's ";
        $box .= "<a href=\"http://www.mit.edu/~dcltdw/OpenImpetus/\">";
        $box .= "OpenImpetus</a>\n competition registration software and ";
        $box .= "Warren Dew's \"Chester\" competition scrutineering software.";
//      $box .= "</font>\n";
        $box .= "</div>\n";
        $box .= "<!-- End Developer Contact -->\n";
        return $box;
    }


}
?>
```

storeable-objects/Address.inc

```
<?php
#####################################################################
# This file is part of SlidingDoors                                 #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd        #
# SlidingDoors is availible for license under the GPL, see          #
# the COPYING file in the root directory of the install for         #
# the full terms of the GPL.                                        #
#                                                                   #
# File: Address.inc                                                 #
# Author: Eric D. Nielsen                                           #
# Description: The Address class                                    #
# Attributes: only inheireted                                       #
# Constructor: person                                               #
# Accessors/Mutators: get/set[street[1|2]|City|State|Zip            #
# Change Log: 11/08/01 -- created -- edn                            #
#####################################################################
class Address extends StoredObject
{

    function Address($db=0, $id=0)
    ###############################################################
    # Method: Address (constructor)                               #
    # Arguments: db -- the database to load/save this address     #
    #            id -- the id of the address, from the db         #
    # Returns: the new address object                             #
    ###############################################################
    {
        $this->primeField="id";
        $this->setID($id);
      StoredObject::StoredObject($db);
        $this->addTable("addresses",
                        "addressid","id",
                        array("street1","street2","city",
                              "state","zip_code"),
                        array("street1","street2","city","state","zip"));
        if ($id)
            $this->retrieve();
    }

    function setStreet1($data)
    {
        return  $this->setGeneralData("street1",$data);
    }
    function setStreet2($data)
    {
        return  $this->setGeneralData("street2",$data);
    }
    function setCity($data)
    {
        return  $this->setGeneralData("city",$data);
    }
    function setState($data)
    {
        return  $this->setGeneralData("state",$data);
    }
    function setZip($data)
    {
        return  $this->setGeneralData("zip",$data);
    }
```

```
    function getStreet1()
    {
        return  $this->getGeneralData("street1");
    }
    function getStreet2()
    {
        return  $this->getGeneralData("street2");
    }
    function getCity()
    {
        return  $this->getGeneralData("city");
    }
    function getState()
    {
        return  $this->getGeneralData("state");
    }
    function getZip()
    {
        return  $this->getGeneralData("zip");
    }
    function getBlock()
    {
        $line1=$this->getStreet1();
        $line2=$this->getStreet2();
        $line3=$this->getCity() . ", " .  $this->getState() . " ". $this->getZip();
        $string = $line1 . ($line2!=""?"\n$line2\n":"\n") . $line3 . "\n";
        return $string;
    }
    function getString()
    {
        $line1=$this->getStreet1();
        $line2=$this->getStreet2();
        $line3=$this->getCity() . ", " .  $this->getState() . " ". $this->getZip();
        $string = $line1 . ($line2!=""?"; $line2; ":"; ") . $line3 . "";
        return $string;
    }
}
?>
```

storeable-objects/Competitor.inc

```
<?php
#################################################################
# This file is part of SlidingDoors                             #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    #
# SlidingDoors is availible for license under the GPL, see      #
# the COPYING file in the root directory of the install for     #
# the full terms of the GPL.                                    #
#                                                               #
# File: Competitor.inc                                          #
# Author: Eric D. Nielsen                                       #
# Description: The competitor class                             #
# Attributes: only inheireted                                   #
# Constructor: competitor                                       #
# Accessors/Mutators: getPartners                               #
# Change Log: 7/27/01 -- created -- edn                         #
#################################################################
class Competitor extends Person
{
    function Competitor($db=0, $id=0)
    ###########################################################
    # Method: Competitor (constructor)                        #
    # Arguments: db -- the database to load/save competitor   #
    #            id -- the id of the competitor, from the db  #
    # Returns: the new competitor object                      #
    ###########################################################
    {
      Person::Person($db,$id);
    }


    function getPartners($role="")
    ###########################################################
    # Function: getPartners                                   #
    # Arguments: role -- "lead"/"follow"/*, * for             #
    #                    unrestricted, lead to get partners   #
    #                    who are leading this competitor      #
    # Returns: list of peopleids                              #
    # Requires: nothing                                       #
    # Modifies: nothing                                       #
    ###########################################################
    {
        $db = $this->db;
        $temp = array();
        $queryID= $this->getID();

        $isFollow = (($role == "follow") ? TRUE : FALSE);
        if (!$isFollow)
        {
            $query = "SELECT leader FROM COUPLES where follower=$queryID;";
            $result = $db->query($query);
            $numLeads = $result->numRows();
            for ($i=0;$i<$numLeads;$i++)
            {
                $lead = $result->getRowAt($i);
                array_push($temp,$lead[0]);
            }
        }
        $isLead = (($role == "lead") ? TRUE : FALSE);
```

```
        if (!$isLead)
        {
            $query = "SELECT follower FROM COUPLES where leader=$queryID;";
            $result = $this->db->query($query);
            $numFollowss = $result->numRows();
            for ($i=0;$i<$numFollowss;$i++)
            {
                $follow = $result->getRowAt($i);
                array_push($temp, $follow[0]);
            }
        }
        return $temp;
    }

}
?>
```

storeable-objects/Couple.inc

```
<?php
#################################################################
# This file is part of SlidingDoors                             #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    #
# SlidingDoors is availible for license under the GPL, see      #
# the COPYING file in the root directory of the install for     #
# the full terms of the GPL.                                    #
#                                                               #
# File: Couple.inc                                              #
# Author: Eric D. Nielsen                                       #
# Description: The couple class                                 #
# Attributes: only inheireted                                   #
# Constructor: person                                           #
# Accessors/Mutators: get/setLeader                             #
#                     get/setFollower                           #
#                     get/add/dropDances                        #
# Change Log: 8/10/01 -- created -- edn                         #
#################################################################
class Couple extends StoredObject
{
    var $eventNames;

    function Couple($db=0, $id=0)
    ###########################################################
    # Method: Couple (constructor)                            #
    # Arguments: db -- the database to load/save this Couple  #
    #            id -- the id of the Couple, from the db      #
    # Returns: the new Couple object                          #
    ###########################################################
    {
        $this->primeField="id";
        $this->setID($id);
      StoredObject::StoredObject($db);
        $this->addTable("couples",
                        "coupleid","id",
                        array("leader","follower"),
                        array("leader","follower"));
        $this->addTable("events_registration",
                         "coupleid","id",
                         array("eventid","competitorNumber"),
                         array("eventid", "number"));
    }

    function getLeader()
    ###########################################################
    # Function: getLeader                                     #
    # Arguments: none                                         #
    # Returns: leader, if no leader found returns  ""         #
    # Requires: nothing                                       #
    # Modifies: nothing                                       #
    ###########################################################
    {
        return $this->getGeneralData("leader");
    }

    function getFollower()
    ###########################################################
    # Function: getFollower                                   #
    # Arguments: none                                         #
```

```
# Returns: Follower , if no Follower  found returns ""      #
# Requires: nothing                                         #
# Modifies: nothing                                         #
#############################################################
{
    return $this->getGeneralData("follower");
}

function setLeader($leader)
#############################################################
# Function: setLeader                                       #
# Arguments: leader -- the leader to set                    #
# Returns: 1 if set, 0 otherwise (never)                    #
# Requires: nothing                                         #
# Modifies: nothing                                         #
#############################################################
{
    return  $this->setGeneralData("leader",$leader);
}

function setFollower($follower)
#############################################################
# Function: setFollower                                     #
# Arguments: Follower -- the follower to set                #
# Returns: 1 if set, 0 otherwise                            #
# Requires: nothing                                         #
# Modifies: nothing                                         #
#############################################################
{
    return $this->setGeneralData("follower",$follower);
}

function getNames($events)
  {
    if (!is_array($events)) return array();
    $numEvents = count($events);
    $temp = array();
    for ($i=0;$i<$numEvents;$i++)
      {
        $query = "SELECT shortname FROM events_names WHERE eventid='$events[$i]';";
        $result = $this->db->query($query);
        if ($result->numRows()!=1)
    die ("Corrupted DB (Couple::getNames) no name for event $events[$i]");
        list($danceName) = $result->getRowAt(0);
        array_push($temp,$danceName);
      }
    return $temp;
  }

function getDances($style="")
  {
    $tempEvents = $this->fields["eventid"];
    if (!is_array($tempEvents))
      $tempEvents = array($tempEvents);
    $events = $this->getNames($tempEvents);
    if ($style=="") return $this->orderDances($events); # need to order;
    $result = "";
    include("couple__getDances.inc");
    if (count($result)>0)
      $result = $this->orderDances($result);
    return $result;
```

```
  }

function styleFilter($events,$style,$dances)
  {
    $result = array();
    $numEvents = count($events);
    for ($i=0;$i<$numEvents;$i++)
      {
        $eventStyleStart = strpos($events[$i],"-");
        $eventStyleEnd = strrpos($events[$i],"-");
        $eventStyleLen = $eventStyleEnd-$eventStyleStart-1;
        $eventStyle = substr($events[$i],$eventStyleStart+1,$eventStyleLen);
        if ($style!=$eventStyle) continue;
        $eventDance = substr($events[$i],$eventsStyleEnd+1);
        $testDance = $eventDance[0];
        $eventDanceLength = strlen($eventDance);
        $j=1;
        $foundOne=FALSE;
        while ($j<$eventDanceLength &&!$foundOne)
          {
            if ('a' <= $eventDance[$j] && $eventDance[$j] <= 'z')
              $testDance .= $eventDance[$j];
            else
              if (in_array($testDance,$dances))
                  $foundOne=TRUE;
              else
                  $testDance = $eventDance[$j];
            $j++;
          }
        if (in_array($testDance,$dances))
          $foundOne=TRUE;
        if (!$foundOne) continue;
        array_push($result,$events[$i]);
      }
    return $result;
  }

function orderDances($events)
  {
    $numEvents = count($events)-1;
    for ($i=0;$i<$numEvents;$i++)
      {
        $done = TRUE;
        for ($j=0;$j<$numEvents-$i;$j++)
          {
            if ($this->eventBefore($events[$j+1],$events[$j]))
              {
                $temp = $events[$j+1];
                $events[$j+1] = $events[$j];
                $events[$j] = $temp;
                $done=FALSE;
              }
          }
        if ($done) break;
      }
    return $events;
  }

function eventBefore($event2,$event1)
  {
    $event2Sep1 = strpos($event2,'-');
```

```
    $event2Sep2 = strrpos($event2,'-');
    $event1Sep1 = strpos($event1,'-');
    $event1Sep2 = strrpos($event1,'-');
    $event2Lvl = substr($event2,0,$event2Sep1);
    $event1Lvl = substr($event1,0,$event1Sep1);
    $event2Style = substr($event2,$event2Sep1+1,$event2Sep2-2);
    $event1Style = substr($event1,$event1Sep1+1,$event1Sep2-2);
    $event2Dance = substr($event2,$event2Sep2+1,strlen($event2));
    $event1Dance = substr($event1,$event1Sep2+1,strlen($event1));
    if (strlen($event2Dance)>1)
      {
        $temp = substr($event2Dance,0,1);
        $length = strlen($event2Dance);
        $done=FALSE;
        $i=1;
        while (!$done && $i<$length)
          {
            $test = substr($event2Dance,$i,$i+1);
            if ("A"<$test && $test<"Z")
              $done=TRUE;
            else
              $temp .= $test;
            $i++;
          }
        $event2Dance=$temp;
      }
    if (strlen($event1Dance)>1)
      {
        $temp = substr($event1Dance,0,1);
        $length = strlen($event1Dance);
        $done=FALSE;
        $i=1;
        while (!$done && $i<$length)
          {
            $test = substr($event2lance,$i,$i+1);
            if ("A"<$test && $test<"Z")
              $done=TRUE;
            else
              $temp .= $test;
            $i++;
          }
        $event1Dance=$temp;
      }
    $levelTest =  include("couple__eventBeforeLevel.inc");
    if ($levelTest) return TRUE;
    if ($event2Lvl!=$event1Lvl) return FALSE;
    $styleTest =  include("couple__eventBeforeStyle.inc");
    if ($styleTest) return TRUE;
    if ($event2Style!=$event1Style) return FALSE;
    $danceTest = include("couple__eventBeforeDance.inc");
    if ($danceTest) return TRUE;
    return FALSE;
  }

function setEvents($events)
  {
    include("couple__setEventsLut.inc");
    $temp=array();
    $numEvents = count($events);
    for ($i=0;$i<$numEvents;$i++)
      {
```

```
            $fallthrough=FALSE;
            include("couple__setEventsSet.inc");
          }
        $this->setGeneralData("eventid",array_unique($temp));
      }
}
?>
```

storeable-objects/Organization.inc

```
<?php
#################################################################
# This file is part of SlidingDoors                             #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    #
# SlidingDoors is availible for license under the GPL, see      #
# the COPYING file in the root directory of the install for     #
# the full terms of the GPL.                                    #
#                                                               #
# File: Organization.inc                                        #
# Author: Eric D. Nielsen                                       #
# Description: The organization class                           #
# Attributes: only inheireted                                   #
# Constructor: organization                                     #
# Accessors/Mutators: get/setName                               #
#                     get/setAbbrev                             #
# Change Log: 6/27/01 -- created -- edn                         #
#################################################################
class Organization extends StoredObject
{

    function Organization($db=0, $id=0)
    ###########################################################
    # Method: Person (constructor)                            #
    # Arguments: db -- the database to load/save this org     #
    #            id -- the id of the organization, from the db#
    # Returns: the new organization object                    #
    ###########################################################
    {
      StoredObject::StoredObject($db,$id);
        $this->primeField="id";

        $this->addTable("organizations",
                        "orgid","id",
                        array("name"),array("fullName"));
        $this->addTable("abbreviations",
                        "fulltext","fullName",
                        array("abbreviation"),array("shortName"));

        $this->fields[$this->primeField]=$id;
        if ($id!=0)
            $this->retrieve();
    }

   function postToDB()
   {
       $retVal = StoredObject::postToDB();
       $this->db->rebuildSchoolInclude();
       return $retVal;
   }

    function getName()
    ###########################################################
    # Function: getName                                       #
    # Arguments: none                                         #
    # Returns: first name, if no name found returns  ""       #
    # Requires: nothing                                       #
    # Modifies: nothing                                       #
    ###########################################################
    {
```

```
        return $this->getGeneralData("fullName");
    }

    function getAbbrev()
    ##############################################################
    # Function: getAbbrev                                        #
    # Arguments: none                                            #
    # Returns: abbreviation, if no abbrev. found returns ""      #
    # Requires: nothing                                          #
    # Modifies: nothing                                          #
    ##############################################################
    {
        return $this->getGeneralData("shortName");
    }

    function setName($name)
    ##############################################################
    # Function: setName                                          #
    # Arguments: name -- the name to set                         #
    # Returns: 1 if set, 0 otherwise (never)                     #
    # Requires: nothing                                          #
    # Modifies: nothing                                          #
    ##############################################################
    {
        return $this->setGeneralData("fullName",$name);
    }

    function setAbbrev($abbrev)
    ##############################################################
    # Function: setAbbrev                                        #
    # Arguments: abbrev -- the abbreviation to set               #
    # Returns: 1 if set, 0 otherwise                             #
    # Requires: nothing                                          #
    # Modifies: nothing                                          #
    ##############################################################
    {
        return $this->setGeneralData("shortName",$abbrev);
    }
}
?>
```

storeable-objects/Person.inc

```
<?php
#####################################################################
# This file is part of SlidingDoors                                 #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd        #
# SlidingDoors is availible for license under the GPL, see          #
# the COPYING file in the root directory of the install for         #
# the full terms of the GPL.                                        #
#                                                                   #
# File: Person.inc                                                  #
# Author: Eric D. Nielsen                                           #
# Description: The person class                                     #
# Attributes: org -- linked to another storeable object             #
# Constructor: person                                               #
# Methods: validateInputs                                           #
#          populateFromFormVars                                     #
# Accessors/Mutators: get/setFirstName                              #
#                     get/setLastName                               #
#                     get/setEmail                                  #
#                     get/setAffil                                  #
# Change Log: 6/12/01 -- created -- edn                             #
#             7/02/02 -- GPL, PrepStep enabled -- edn               #
#####################################################################
class Person extends StoredObject
{

    var $org;

    function Person($db=0, $id=0)
    ################################################################
    # Method: Person (constructor)                                 #
    # Arguments: db -- the database to load/save this person       #
    #            id -- the id of the person, from the db           #
    # Returns: the new person object                               #
    ################################################################
    {
        $this->primeField="id";
        $this->setID($id);
      StoredObject::StoredObject($db);
        GLOBAL $SD_elements;


        if (in_array("Name",$SD_elements) &&
            in_array("Affiliation",$SD_elements))
            $this->addTable("people",
                            "peopleid","id",
                            array("firstname","lastname","organization"),
                            array("firstname","lastname","organizationID"));
        else if (in_array("Name",$SD_elements))
            $this->addTable("people",
                            "peopleid","id",
                            array("firstname","lastname"),
                            array("firstname","lastname"));
        else if (in_array("Affiliation",$SD_elements))
            $this->addTable("people",
                            "peopleid","id",
                            array("organization"),
                            array("organizationID"));
        if (in_array("Email",$SD_elements))
            $this->addTable( "people_contact",
```

```
                        "peopleid","id",
                        array("emailday"),array("email"));
    if (in_array("Fee Category",$SD_elements))
        $this->addTable( "people_classification",
                        "peopleid", "id",
                        array("agelevel","class"),
                        array("ageLevel","class"));
    if (in_array("Package",$SD_elements))
    {
        $this->addTable("people_paid",
                        "peopleid","id",
                        array("org_pays","paid","org_paying"),
                        array("orgPays","paid","orgPaying"));
        $this->addTable("packages_purchased",
                        "peopleid","id",
                        array("packageid"),array("packageid"));
    }
    if ($id)
        $this->retrieve();
}

function retrieve()
{
  StoredObject::retrieve();
  $this->org = new Organization($this->db,$this->fields["organizationID"]);
}


#################################################################
# Function: validateInputs                                      #
# Arguements: formVars -- will contain the validated inputs     #
#                         along with error messages for         #
#                         invalid inputs                        #
#             type -- <PRIME | PARTNER | PRIME-UPDATE>          #
#                     used in determining which fields are      #
#                     required and where the inputs are         #
#                     stored                                    #
# Returns: TRUE iff the inputs are valid, FALSE otherwise       #
# Modifies: formVars                                            #
#################################################################
function validateInputs($formVars,$type="PRIME")
{
    GLOBAL $SD_elements;
    GLOBAL $SD_allow_TBAs;
    GLOBAL $SD_prime_is_always_leader;
    GLOBAL $SD_error_text_color;
    GLOBAL $HTTP_POST_VARS;

    switch ($type)
    {
        case "PRIME" : $prime=TRUE; $update=FALSE; $partner=FALSE;
                       $prefix=""; $Prefix=""; $label="your "; break;
        case "PARTNER" : $prime=FALSE; $update=FALSE; $partner=TRUE;
                         $prefix=""; $Prefix=""; $label="your partner's ";
--------------------------------break;
----------case "PRIME-UPDATE": $prime=TRUE; $update=TRUE; $partner=FALSE;
--------------------------------$prefix="update";
--------------------------------$label="your "; break;
----------default: break;
--------}
--------$needsName = in_array("Name",$SD_elements);
```

```
        $needsEmail=in_array("Email",$SD_elements);
        $needsFee = in_array("Fee Category",$SD_elements);
        $needsPackage = in_array("Package",$SD_elements);
        $needsID = in_array("ID_number",$SD_elements);
        $needsAffil = in_array("Affiliation",$SD_elements);
        $needsPhone = in_array("Phone Number",$SD_elements);
        $needsAddress = in_array("Address",$SD_elements);

        $tbaPost = $Prefix ."TBA";
        $tba = $HTTP_POST_VARS["$tbaPost"];

        $role = $HTTP_POST_VARS["role"];


        $firstNamePost = $prefix."FirstName";
        $firstNameMsg =$firstNamePost."Msg";
        $firstName  = htmlspecialchars($HTTP_POST_VARS["$firstNamePost"]);

        $lastNamePost = $prefix."LastName";
        $lastNameMsg =$lastNamePost."Msg";
        $lastName   = htmlspecialchars($HTTP_POST_VARS["$lastNamePost"]);

        $newAffilPost = $prefix."NewAffil";
        $newAffilMsg =$newAffilPost."Msg";
        $newAffil   = htmlspecialchars($HTTP_POST_VARS["$newAffilPost"]);

        $affilShortPost = $prefix."AffilShort";
        $affilShortMsg =$affilShortPost."Msg";
        $affilShort = htmlspecialchars($HTTP_POST_VARS["$affilShortPost"]);

        $affilIDPost = $prefix."AffilID";
        $affilIDMsg =$affilIDPost."Msg";
        $affilID    = $HTTP_POST_VARS["$affilIDPost"]; # id's from db

        $emailPost = $prefix."Email";
        $emailMsg =$emailPost."Msg";
        $email      = strtolower(htmlspecialchars($HTTP_POST_VARS["$emailPost"]));

        $agePost = $prefix."Age";
        $ageMsg =$agePost."Msg";
        $age        = $HTTP_POST_VARS["$agePost"];

        $packagePost = $prefix."Package";
        $packageMsg =$packagePost."Msg";
        $package    = $HTTP_POST_VARS["$packagePost"];


        $validated= TRUE;

        # handle TBA special cases, over-ridden values are saved during the
        # regular processing of that field
        if ($SD_allow_TBAS)
        {
            if ($tba || $firstName=="TBA")
            {
                $tba=TRUE;
                if ($needsName)
                {
                    $firstName="TBA";
                    if (!is_num($lastName))
                        $lastName = $this->db->getNextTBA();
```

```
                }
                else if ($needsEmail)
                    $email = "TBA.".$this->db->getNextTBA();
                if ($needsAffil)
                    $affilID=-1;
            }
        } else $tba=FALSE;

        # handle first/last names; save value if non empty. Names are required
        # to be filled out, even if TBA, above special case code ensures this
        if ($needsName)
        {
            if ($firstName!="")
                $formVars["$firstNamePost"]=$firstName;
            else
            {
                $formVars[$firstNameMsg]="<font
color=\"$SD_error_text_color\">Please enter $label first name.</font><br />\n";
                $validated=FALSE;
            }
            if ($lastName!="")
                $formVars["$lastNamePost"]=$lastName;
            else
            {
                $formVars[$lastNameMsg]="<font
color=\"$SD_error_text_color\">Please enter $label last name.</font><br />\n";
                $validated=FALSE;
            }
        }

        # handle email, does not validate or even regexp check, email may
        # be empty if tba and name used
        if ($needsEmail)
            if ($email!="")
                $formVars["$emailPost"]=$email;
        else if (!$tba)
        {
            $formVars[$emailMsg]="<font color=\"$SD_error_text_color\">
Please enter $label email.</font><br />\n";
            $validated=FALSE;
        }

        # handle packages, should never be empty, but just in case
        if ($needsPackage)
            if ($package!="")
                $formVars["$packagePost"]=$package;
        else if (!$tba)
        {
            $formVars[$packageMsg]="<font color=\"$SD_error_text_color\">
Please select $label registration package.</strong><br />\n";
            $validated=FALSE;
        }

        # handles fee category, should never be empty
        if ($needsFee)
            if ($age!="")
                $formVars["$agePost"]=$age;
        else
        {
            $formVars[$ageMsg]="<font color=\"$SD_error_text_color\">
Please select $label fee category.</strong><br />\n";
```

```
            $validated=FALSE;
        }

        # handle the complicated affiliation cases
        if ($needsAffil)
        {
            if ($affilID=="choose")
                $affilID="";

            if ($affilID!="")
                $formVars["$affilIDPost"]=$affilID;

            if ($newAffil!="")
                $formVars["$newAffilPost"]=$newAffil;

            if ($affilShort!="")
                $formVars["$affilShortPost"]=$affilShort;

            # user selected from drop down box and entered text
            if ($affilID!="" && ($newAffil!="" || $affilShort != ""))
            {
                $validated=FALSE;
                $formVars[$affilIDMsg]="<font
color=\"$SD_error_text_color\">Please do not select an affiliation from the
list box and also enter information into the \"New Affiliation\" boxes.
</font><br />\n";
            }

            # user didn't select nor enter text
            if ($affilID==""&&($newAffil==""||$affilShort==""))
            {
                $validated=FALSE;
                $formVars[$affilIDMsg]="<font color=\"$SD_error_text_color\">
Please either select an affiliation from the box, or enter a new one with
long and short form in the text boxes below.</font><br/>";
            }
            #user didn't select, but only entered 1 of 2 text entries
            if (($newAffil!="" && $affilShort=="") ||
                ($newAffil=="" && $affilShort!=""))
            {
                $validated = FALSE;
                $formVars[$newAffilMsg]="<font color=\"$SD_error_text_color\">
Please provide both a long and short form of $label affiliation's name.  Here
are some examples:<br/>Massachusetts Institute of Technology -->MIT<br/>
Harvard University -->Harvard<br/>Fred Astaire Dance Studio, Boston -->
FADS Boston<br/></font>";
            }
            if ($newAffil!="" && $affilShort!="")
            {
                $query = "SELECT abbreviation from abbreviations where
fulltext='$newAffil';";
                $result = $this->db->query($query);
                if ($result->numrows()>0)
                {
                    $validated = FALSE;
                    $formVars[$newAffilMsg]="<font
color=\"$SD_error_text_color\">An affiliation with the name you
entered already exists, please select it from the
drop down select box.</font></br>";
                }
                $query = "SELECT fulltext from abbreviations where
```

```
abbreviation='$affilShort';";
                $result = $this->db->query($query);
                if ($result->numrows()>0)
                  {
                    list($tempName) = $result->getRowAt(0);
                    $validated = FALSE;
                    $formVars[$affilShortMsg]="<font
color=\"$SD_error_text_color\">$tempName is already using that
abbreviation.  You will have to choose another.</font></br>";
                  }
              }
        }

        $formVals["Role"]=$role;

        if ($needsID)
        {

        }
        if ($needsAddress)
        {
        }
        if ($needsPhone)
        {
        }
        return $validated;
    }


    function populateFromFormVars($formVars,$prefix="")
    ###############################################################
    # Function: populateFromFormVars                              #
    # Arguements: formVars -- contains the values to insert       #
    #             prefix -- an optional string to prefix to       #
    #                       names in FormVars before matching     #
    #                       with fields                           #
    #                       stored                                #
    # Returns: nothing                                            #
    # Modifies: this                                              #
    ###############################################################
    {
        GLOBAL $SD_elements;

        if (in_array("Name",$SD_elements))
        {
            $this->setFirstName($formVars[$prefix . "FirstName"]);
            $this->setLastName($formVars[$prefix . "LastName"]);
        }

        if (in_array("Email",$SD_elements))
            $this->setEmail($formVars[$prefix . "Email"]);
        if (in_array("Fee Category",$SD_elements))
            $this->setAgeLevel($formVars[$prefix . "Age"]);
        if (in_array("Package",$SD_elements))
            $this->setPackage($formVars[$prefix . "Package"]);
        if (in_array("Affiliation",$SD_elements))
        {
            $affilID= $formVars[$prefix . "AffilID"];
            if (!(($affilID=="") || ($affilID==-1)))
                $this->setAffilID($affilID);
            else if ($formVars[$prefix."NewAffil"]=="")
```

```
        {
            $this->setAffilID(-1);
            $this->setOrgPays("FALSE");
        }
        $this->setPaid($this->getPaid());
    }
}

function getPackage()
#############################################################
# Function: getPackage                                      #
# Arguments: none                                           #
# Returns: packageID, if no packageID found returns  ""     #
# Requires: nothing                                         #
# Modifies: nothing                                         #
#############################################################
{
    return $this->getGeneralData("packageid");
}

function setPackage($pid)
#############################################################
# Function: setPackage                                      #
# Arguments: pid                                            #
# Returns: nothing                                          #
# Requires: nothing                                         #
# Modifies: nothing                                         #
#############################################################
{
    return $this->setGeneralData("packageid",$pid);
}

function getFirstName()
#############################################################
# Function: getFirstName                                    #
# Arguments: none                                           #
# Returns: first name, if no first name found returns  ""   #
# Requires: nothing                                         #
# Modifies: nothing                                         #
#############################################################
{
    return $this->getGeneralData("firstname");
}

function getLastName()
#############################################################
# Function: getLastName                                     #
# Arguments: none                                           #
# Returns: last name, if no last name found returns ""      #
# Requires: nothing                                         #
# Modifies: nothing                                         #
#############################################################
{
    return $this->getGeneralData("lastname");
}

function getEmail()
#############################################################
# Function: getEmail                                        #
# Arguments: none                                           #
# Returns: email, if no email found returns ""              #
```

```
# Requires: nothing                                                    #
# Modifies: nothing                                                    #
#########################################################################
{
    return $this->fields["email"];
}

function getAffil()
#########################################################################
# Function: getAffil                                                   #
# Arguments: none                                                      #
# Returns: affiliation, if no affiliation found returns "" #
# Requires: nothing                                                    #
# Modifies: nothing                                                    #
#########################################################################
{
    return $this->org->getAbbrev();
}

function getAffilID()
#########################################################################
# Function: getAffil                                                   #
# Arguments: none                                                      #
# Returns: affiliation, if no affiliation found returns "" #
# Requires: nothing                                                    #
# Modifies: nothing                                                    #
#########################################################################
{
    return $this->getGeneralData("organizationID");
}

function getAgeLevel()
#########################################################################
# Function: getAgeLevel                                                #
# Arguments: none                                                      #
# Returns: age level, if no level found  returns ""                   #
# Requires: nothing                                                    #
# Modifies: nothing                                                    #
#########################################################################
{
    return $this->getGeneralData("ageLevel");
}

function getClass()
#########################################################################
# Function: getClass                                                   #
# Arguments: none                                                      #
# Returns: classification, if no class found returns ""    #
# Requires: nothing                                                    #
# Modifies: nothing                                                    #
#########################################################################
{
    return $this->getGeneralData("class");
}

function getOrgPays()
{
    $temp = $this->getGeneralData("orgPays");
    return ($temp=="t"? TRUE : FALSE);
}
```

```
function getOrgPaying()
{
    return $this->getGeneralData("orgPaying");
}

function setOrgPaying($data)
{
    return $this->setGeneralData("orgPaying",$data);
}

function getPaid()
{
    $temp = $this->getGeneralData("paid");
    return ($temp=="t"? TRUE : FALSE);
}


function setAgeLevel($ageLevel)
#############################################################
# Function: setAgeLevel                                     #
# Arguments: ageLevel -- the ageLevel to set                #
# Returns: 1 if set, 0 otherwise (never)                    #
# Requires: nothing                                         #
# Modifies: nothing                                         #
#############################################################
{
    return  $this->setGeneralData("ageLevel",$ageLevel);
}

function setClass($class)
#############################################################
# Function: setClass                                        #
# Arguments: class -- the class to set                      #
# Returns: 1 if set, 0 otherwise (never)                    #
# Requires: nothing                                         #
# Modifies: nothing                                         #
#############################################################
{
    return  $this->setGeneralData("class",$class);
}


function setFirstName($firstName)
#############################################################
# Function: setFirstName                                    #
# Arguments: firstName -- the firstName to set              #
# Returns: 1 if set, 0 otherwise (never)                    #
# Requires: nothing                                         #
# Modifies: nothing                                         #
#############################################################
{
    return  $this->setGeneralData("firstname",$firstName);
}

function setLastName($lastName)
#############################################################
# Function: setLastName                                     #
# Arguments: lastName -- the last name to set               #
# Returns: 1 if set, 0 otherwise                            #
# Requires: nothing                                         #
# Modifies: nothing                                         #
```

```
##########################################################
{
    return $this->setGeneralData("lastname",$lastName);
}

function setEmail($email)
##########################################################
# Function: setEmail                                     #
# Arguments: email -- the email to set                   #
# Returns: 1 if set, 0 otherwise                         #
# Requires: nothing                                      #
# Modifies: nothing                                      #
##########################################################
{
    return $this->setGeneralData("cmail", $email);
}

function setAffilID($affil)
##########################################################
# Function: setAffil                                     #
# Arguments: affil -- the affil to set                   #
# Returns: 1 if set, 0 otherwise                         #
# Requires: nothing                                      #
# Modifies: nothing                                      #
##########################################################
{
    if ($this->fields["orgPays"]=="" && $affil!=-1 && $affil!="")
    {
        $query = "SELECT org_pays FROM registration_supplement ";
        $query.= "WHERE orgid=$affil;";
        $result = $this->db->query($query);
        if ($result->numRows())
        {
            list($orgPays) = $result->getRowAt(0);
            $orgPays = ($orgPays=="t");
            $this->setOrgPays($orgPays);
            if ($orgPays)
                $this->setOrgPaying($affil);
            else
                $this->setOrgPaying(0);
        }
    }
    else if ($affil==-1 || $affil=="")
    {
        $this->setOrgPays("FALSE");
        $this->setOrgPaying(0);
    }
    return $this->setGeneralData("organizationID",$affil);
}

function setPaid($data)
{
    return $this->setGeneralData("paid",($data? "TRUE":"FALSE"));
}
function setOrgPays($data)
{
    return $this->setGeneralData("orgPays",($data ? "TRUE" : "FALSE"));
}

}
```

```
?>
```

storeable-objects/RegOrg.inc

```
<?php
#####################################################################
# This file is part of SlidingDoors                                 #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd        #
# SlidingDoors is availible for license under the GPL, see          #
# the COPYING file in the root directory of the install for         #
# the full terms of the GPL.                                        #
#                                                                   #
# File: Organization.inc                                            #
# Author: Eric D. Nielsen                                           #
# Description: The organization class                               #
# Attributes: only inheireted                                       #
# Constructor: organization                                         #
# Accessors/Mutators: get/setName                                   #
#                     get/setAbbrev                                 #
# Change Log: 6/27/01 -- created -- edn                             #
#####################################################################
class RegOrg extends Organization
{
    var $repPerson;
    var $coordPeople;
    var $buildingAddress;
    var $repAddress;

    function RegOrg($db=0, $id=0)
        ###############################################################
        # Method: RegOrg (constructor)                                #
        # Arguments: db -- the database to load/save this org         #
        #            id -- the id of the organization, from the db    #
        # Returns: the new organization object                        #
        ###############################################################
    {
      StoredObject::StoredObject($db,$id);
        $this->primeField="id";
        $this->addTable("organizations",
                        "orgid","id",
                        array("name","rep_title","representative","building"),
                        array("fullName","rep_title","rep","building"));
        $this->addTable("abbreviations",
                        "fulltext","fullName",
                        array("abbreviation"),array("shortName"));
        $this->addTable("registration_coordinators",
                        "orgid","id",
                        array("peopleid","email_on_update","email_digest"),
                        array("coordinator","email","digest"));
        $this->addTable("registration_supplement",
                        "orgid","id",
                        array("org_pays"), array("org_pays"));
        $this->addTable("buildings",
                        "buildingid","building",
                        array("addressid"),array("org_addressid"));
        $this->fields[$this->primeField]=$id;
        $this->repPerson = new Person($db);
        $this->coordPeople = array();
        $this->buildingAddress = new Address($db);
        $this->repAddress = new Address($db);

        if ($id!=0)
            $this->retrieve($id);
```

```
}

function retrieve($id=0)
{
  StoredObject::retrieve($id);

    $rep = $this->getRep();
    if (isset($rep) && $rep!=0 && $rep!="")
    {
        $this->repPerson->setID($rep);
        $this->repPerson->retrieve();
    }
    $coords = $this->getCoordinator();

    if ($coords != 0)
        if (is_array($coords))
        {
            $numCoords = count($coords);
            for ($i=0;$i<$numCoords;$i++)
            {
                $aPerson = new Person($this->db);
                $aPerson->setID($coords[$i]);
                $aPerson->retrieve();
                arrayPush($this->coordPeople,$aPerson);
            }
        }
    else
    {
        $aPerson = new Person($this->db);
        $aPerson->setID($coords);
        $aPerson->retrieve();
        array_push($this->coordPeople,$aPerson);
    }
    $building = $this->getBuilding();
    if (isset($buildings) && $building!=0 && $building!="")
    {
        $this->buildingAddress->setID($building);
        $this->buildingAddress->retrieve();
    }
    if (isset($rep) && $rep!=0 && $rep!="")
    {
        $query = "SELECT address FROM people WHERE peopleid=$rep;";
        $result = $this->db->query($query);
        if ($result->numRows())
            list($repAddr) =$result->getRowAt(0);
        else
            $repAddr=0;
        if ($repAddr!=0)
        {
            $this->repAddress->setID($repAddr);
            $this->repAddress->retrieve();
        }
    }
    return 1;
}

function postToDB()
{
  Organization::postToDB();

    $rep = $this->getRep();
```

```
    if ($rep!=0)
    {
        $this->repPerson->postToDB();
    }

    $coords = $this->getCoordinator();
    if ($coords != 0)
        if (is_array($coords))
        {
            $numCoords = count($coords);
            for ($i=0;$i<$numCoords;$i++)
            {
                $coordPeople[$i]->postToDB();
            }
        }
    else
    {
        $coordPeople[0]->post();
    }
    $building = $this->getBuilding();
    if ($building!=0)
    {
        $this->buildingAddress->postToDB();
    }
    if ($rep)
    {
        $query = "SELECT address FROM people WHERE peopleid=$rep;";
        $result = $this->db->query($query);
        if ($result->numRows())
            list($repAddr) =$result->getRowAt(0);
        else
            $repAddr=0;
        if ($repAddr!=0)
        {
            $this->repAddress->postToDB();
        }
    }
}

function getAddress()
  {
    if ($this->buildingAddress->getID()!=0)
      return $this->buildingAddress->getString();
    else if ($this->repAddress->getID()!=0)
      return $this->repAddress->getString();
    else
      return "";
  }

function setBuilding($data)
{
    $this->setGeneralData("building",$data);
    $this->buildingAddress->setID($data);
    $this->buildingAddress->retrieve();
    return 1;
}
function getBuilding()
{
    return $this->getGeneralData("building");
}
function getBuildingString()
```

```
{
    if ($this->buildingAddress->getID()!=0)
        return $this->buildingAddress->getString();
    else
        return "";
}
function getBuildingBlock()
{
    if ($this->buildingAddress->getID()!=0)
        return $this->buildingAddress->getBlock();
    else
        return "";
}
function getOrgAddr()
{
    return $this->buildingAddress;
}
function getRepAddr()
{
    return $this->repAddress;
}

function setRepTitle($data)
{
    return $this->setGeneralData("rep_title",$data);
}
function setRep($data)
{
    $this->setGeneralData("rep",$data);
    $this->repPerson->setID($data);
    $this->repPerson->retrieve();
    $query = "SELECT address FROM people WHERE peopleid='$data';";
    $result = $this->db->query($query);
    if ($result->numRows())
        list($repAddr) =$result->getRowAt(0);
    else
        $repAddr=0;
    if ($repAddr!=0)
    {
        $this->repAddress->setID($repAddr);
        $this->repAddress->retrieve();
    }
    return 1;
}
function setRepAddr($data)
{
    $repID = $this->getRep();
    if ($repID)
    {
        $query = "UPDATE people SET address=$data WHERE peopleid=$repID;";
        $result = $this->db->query($query);
        $this->repAddress->setID($data);
        $this->repAddress->retrieve();
    }
}

function getRepString()
{
    if ($this->repPerson->getID()!=0)
    {
        $string = $this->repPerson->getFirstName() . " ";
```

```
            $string .= $this->repPerson->getLastName() ;
        }
        else
            $string="";
        if ($this->repAddress->getID()!=0)
            return "$string;". $this->repAddress->getString();
        else
            return $string;
    }
    function getRepBlock()
    {
        if ($this->repPerson->getID()!=0)
        {
            $string = $this->repPerson->getFirstName . " ";
            $string .= $this->repPerson->getLastName . "\n ";
        }
        else
            $string="";
        if ($this->repAddress->getID()!=0)
            return $string . $this->repAddress->getBlock();
        else
            return $string;
    }
    function addCoordinator($data,$email=true,$digest=true)
    {
        $coords = $this->getCoordinator();
        $emails = $this->fields["email"];
        $digests = $this->fields["digest"];
        if (is_array($coords))
        {
            if (!in_array($data,$coords))
            {
                $numCoord = count($coords);
                $this->fields["coords"][$numCoord] = $data;
                $this->fields["email"][$numCoord] = $email;
                $this->fields["digest"][$numCoord] = $digest;
                $aPerson = new Person($this->db);
                $aPerson->setID($data);
                $aPerson->retrieve();
                $this->coordPeople[$numCoord]=$aPerson;
            }
        }
        else
        {
            $this->fields["coords"]=array($coords,$data);
            $this->fields["email"]=array($emails,$email);
            $this->fields["digest"]=array($digests,$digest);
            $aPerson = new Person($this->db);
            $aPerson->setID($data);
            $aPerson->retrieve();
            array_push($this->coordPeople,$aPerson);
        }
        return 1;
    }
    function removeCoordinator($data)
    {
        $coordsList = $this->getCoordinator();
        if ((is_array($coordsList) && !in_array($data,$coordsList)) ||
            (!is_array($coordsList) && $data!=$coordsList))
            return;
        $coordsPeople = $this->coordPeople;
```

```
    $numCoords = count($coordsPeople);
    switch ($numCoords)
    {
      case 0:
        break;
      case 1:
        $this->coordPeople=array();
        unset($this->fields["coordinator"]);
        unset($this->fields["email"]);
        unset($this->fields["digest"]);
        break;
      case 2:
        if ($coordList[0]==$data)
        {
            $this->fields["coordinator"]=$coordList[1];
            $this->fields["email"]=$this->fields["email"][1];
            $this->fields["digest"]=$this->fields["digest"][1];
            $this->coordPeople[0]=$this->coordPeople[1];
            unset($this->coordPeople[1]);
        }
        else
        {
            $this->fields["coordinator"]=$coordList[0];
            $this->fields["email"]=$this->fields["email"][0];
            $this->fields["digest"]=$this->fields["digest"][0];
            unset($this->coordPeople[1]);
        }
        break;
      default:
        $tempList=array();
        $tempEmail=array();
        $tempDigest=array();
        $tempPeople=array();
        for ($i=0;$i<$numCoords;$i++)
        {
            if ($coordsList[$i]==$data) continue;
            array_push($tempList,$coordsList[$i]);
            array_push($tempEmail,$this->fields["email"][$i]);
            array_push($tempDigest,$this->fields["digest"][$i]);
            array_push($tempPeople,$coordsPeople[$i]);
        }
        $this->coordPeople = $tempPeople;
        $this->fields["coordinator"]=$tempList;
        $this->fields["email"]=$tempEmail;
        $this->fields["digest"]=$tempDigest;
        break;
    }
    return;
}

function getCoordString()
{
    $coords = $this->coordPeople;
    $numCoords = count($coords);
    $string="";
    for ($i=0;$i<$numCoords;$i++)
    {
        $string .= $coords[$i]->getFirstName() . " ";
        $string .= $coords[$i]->getLastName();
        if ($numCoords==1 && $this->fields["email"])
            $string .= "(receives email on member registration)";
```

```
        else if ($numCoords!=1 && $this->fields["email"][$i])
            $string .= "(receives email on member registration)";
        $string .= ($i==($numCoords-1))?"":", ";
    }
    return $string;
}
function getCoordBlock()
{
    $coords = $this->coordPeople;
    $numCoords = count($coords);
    $string="";
    for ($i=0;$i<$numCoords;$i++)
    {
        $string .= $coords[$i]->getFirstName() . " ";
        $string .= $coords[$i]->getLastName();
        $string .= "\n";
    }
    return $string;
}

function setEmail($data,$id=0)
{
    $coords = $this->getCoordinator();
    $numCoords= count($coords);
    if ($numCoords=0) return;
    elseif ($numCoords==1) return $this->setGeneralData("email",$data);
    elseif ($id==0 || !in_array($id,$coords)) return;
    else
    {
        for ($i=0;$i<$numCoords;$i++)
            if ($coords[$i]==$id)
                return $this->fields["email"][$i]=$data;
    }
    return;
}
function setDigest($id,$data)
{
    $coords = $this->getCoordinator();
    $numCoords= count($coords);
    if ($numCoords=0) return;
    elseif ($numCoords==1) return $this->setGeneralData("digest",$data);
    elseif ($id==0 || !in_array($id,$coords)) return;
    else
    {
        for ($i=0;$i<$numCoords;$i++)
            if ($coords[$i]==$id)
                return $this->fields["digest"][$i]=$data;
    }
    return;
}
function setOrgPays($data)
{
    return $this->setGeneralData("org_pays",$data);
}
function getRepTitle()
{
    return $this->getGeneralData("rep_title");
}
function getRep()
{
    return $this->getGeneralData("rep");
```

```
    }
    function getCoordinator()
    {
        return $this->getGeneralData("coordinator");
    }
    function getEmail($id=0)
    {
        $numCoords=count($this->getCoordinator());
        if ($id=0)
            return $this->getGeneralData("email");
        else
            for ($i=0;$i<$numCoords;$i++)
                if ($coords[$i]==$id)
                    return $this->fields["email"][$i];
        return;
    }
    function getDigest($id=0)
    {
        $numCoords=count($this->getCoordinator());
        if ($id=0)
            return $this->getGeneralData("digest");
        else
            for ($i=0;$i<$numCoords;$i++)
                if ($coords[$i]==$id)
                    return $this->fields["digest"][$i];
        return;
    }
    function getOrgPays()
    {
        return $this->getGeneralData("org_pays");
    }
}
?>
```

storeable-objects/StoredObject.inc

```
<?php
#################################################################
# This file is part of SlidingDoors                             #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    #
# SlidingDoors is availible for license under the GPL, see      #
# the COPYING file in the root directory of the install for     #
# the full terms of the GPL.                                    #
#                                                               #
# File: StoredObject.inc                                        #
# Author: Eric D. Nielsen                                       #
# Description: A base class that contains tools for storing     #
#              and retrieving objects from a database           #
#              This is effectively an abstract class and        #
#              shuld never be instantiated directly.            #
# Attributes: id -- the ID from the DB                          #
#             values -- an associative array containing the     #
#                       class's data                            #
#             tables -- an array of TableInfo objects           #
#             detailing how this object relates to              #
#             the tables in the DB                              #
# Constructor: StoredObject                                     #
# Methods: fillFromDB                                           #
#          postToDB                                             #
#          searchDB                                             #
# Accessors/Mutators: getGeneralData                            #
#          setGeneralData                                       #
#          setDB                                                #
#          getDB                                                #
#          getID                                                #
#          setID                                                #
#          addTable                                             #
#          removeTable                                          #
#          getTable                                             #
# Change Log: 6/12/01 - - created -- edn                        #
#             8/03/01 - - rewritten - edn                       #
#             7/02/02 - - GPL Notice - - edn                    #
#################################################################
class StoredObject
{
    var $primeField;
    var $fields;
    var $tables;
    var $dirty;
    var $fieldsToTables;
    var $tablesToFields;
    var $tableOrder;
    var $db;

    ###############################################################
    # Method: StoredObject (constructor)                          #
    # Arguments: none                                             #
    # Returns: the new StoredObject                               #
    ###############################################################
    function StoredObject($db=0)
    {
        # note this is a abstract class, $primekey must be set
        $this->db=$db;
        $this->fields=array();
        $this->tables=array();
```

```
        $this->dirty=array();
        $this->fieldsToTables=array();
        $this->tablesToFields=array();
        $this->tableOrder=array();
    }


    function getSelectByIDString($tableName, $table,$multi,$m)
    {
        $result = "SELECT " . arrayToCSL($table[1]) . " FROM $tableName ";
        $tableKey = $table[0];
        $result .= "WHERE $tableKey = '";
#       echo printArray($this->tablesToFields);
#       echo printArray($this->fields);
        $fieldName = $this->tablesToFields[$tableKey][$tableName];
        if ($multi==1)
            $result .= $this->fields[$fieldName] . "';";
        else
            $result .= $this->fields[$fieldName][$m] . "';";
        return $result;
    }

    function remove()
    {
        $tableName = $this->tableOrder[0];
        $tableKey = $this->tables[$tableName][0];
        $prime = $this->fields[$this->primeField];
        $query = "DELETE FROM $tableName WHERE $tableKey=$prime;";
        $this->db->query($query);
    }

    ##########################################################
    # Method: retrieve                                       #
    # Arguments: none                                        #
    # Returns: 1 if load successful                          #
    #          0 if DB down or ID not set                    #
    # Modifies: fields                                       #
    ##########################################################
    function retrieve($id=0)
    {
        $db=$this->db;
        if ($id!=0)
            $this->setID($id);
        $currentID = $this->getID();
        if (!$currentID)
            return 0; #no ID set
        $numTables = count($this->tables);
        for ($i=0;$i<$numTables;$i++)
        {
            $tableName = $this->tableOrder[$i];
            $table = $this->tables[$tableName];
            $tableKey = $this->tablesToFields[$table[0]][$tableName];
            $tableKeyValue = $this->fields[$tableKey];
            if (!is_array($tableKeyValue))
                $tableKeyValue = array($tableKeyValue);
            $numValues = count($tableKeyValue);
            for ($m=0;$m<$numValues;$m++)
            {
                if (!isset($this->fields[$tableKey]) || $tableKeyValue[$m]=="") continue;
                $query = $this->getSelectByIDString($tableName,$table,$numValues,$m);
#               echo $query . "<br>";
```

```
            $result = $db->query($query);
            $numFields = count($table[1]);
            switch($numRows = $result->numRows())
            {
                case 0 : break;
                case 1 :
                    $temp = $result->getRowAt(0);
                for ($k=0;$k<$numFields;$k++)
                {
                    $level1 = $table[1][$k];
                    $field = $this->tablesToFields[$level1][$tableName];
                    if ($numValues==1)
                        $this->fields[$field] = $temp[$k];
                    else
                        $this->fields[$field][$m] = $temp[$k];
                }
                break;
              default:
                for ($j=0;$j<$numRows;$j++)
                {
                    $temp = $result->getRowAt($j);
                    for ($k=0;$k<$numFields;$k++)
                    {
                        $level1= $table[1][$k];
                        $field = $this->tablesToFields[$level1][$tableName];
                        if ($numValues==1)
                            $this->fields[$field][$j] = $temp[$k];
                        else
                            $this->fields[$field][$m][$j] = $temp[$k];
                    }
                }
            }
        }
    }
    return 1;
}

#############################################################
# Function: postToDB                                        #
# Arguments: none                                           #
# Returns: ID or 0 if database errors                       #
# Requires: nothing                                         #
# Modifies: the data stored in DB                           #
#############################################################
function postToDB()
{
    if ($this->fields[$this->primeField]=="")
        $result = $this->insertNew();
    else
        $result = $this->update($this->fields[$this->primeField]);
    reset($this->fields);
    while ($aField = each($this->fields))
    {
        $temp=$aField['key'];
        $this->dirty[$temp]=FALSE;
    }
    return $this->fields[$this->primeField];
}

#############################################################
# Function: insertNew                                       #
```

```
    # Arguments: id                                                    #
    # Returns: ID or 0 if database errors                              #
    # Requires: nothing                                                #
    # Modifies: the data stored in DB                                  #
    ###################################################################
    function insertNew()
    {
        reset($this->fields);
        $tablesInvolved = array();
        while ($field = each($this->fields))
        {
            if ($field['value']=="" ) continue;
            $tablesAffectedByField = $this->fieldsToTables[$field['key']];
            reset($tablesAffectedByField);
            while ($aTable = each($tablesAffectedByField))
            {
                if (in_array($aTable['key'],array_keys($tablesInvolved)))
                    array_push($tablesInvolved[$aTable['key']],
$this->fieldsToTables[$field['key']][$aTable['key']]);
                else
                    $tablesInvolved[$aTable['key']]=
array($this->fieldsToTables[$field['key']][$aTable['key']]);
            }
        }
        reset($tablesInvolved);
        while ($aTable = each($tablesInvolved))
        {
            $primeColumn = $this->tables[$aTable['key']][0];
            $tableName = $aTable['key'];
            $columns = $aTable['value'];
            $tablePrimeField = $this->tablesToFields[$primeColumn][$tableName];
            $arrayDepth=1;
            $numColumns = count($columns);
            for ($i=0;$i<$numColumns;$i++)
            {
                $tempFieldName = $this->tablesToFields[$columns[$i]][$tableName];
                $aColumn = $this->fields[$tempFieldName];
                if (is_array($aColumn) && (count($aColumn) > $arrayDepth))
                    $arrayDepth = count($aColumn);
            }
            for ($i=0;$i<$arrayDepth;$i++)
            {
                $query = "INSERT INTO $tableName (";
                $queryPart2 = " values (";
                $first = TRUE;
                $tablePrimeValue = $this->fields[$tablePrimeField];
                if ($tablePrimeValue!="")
                {
                    $needPrime=FALSE;
                    if (!in_array($primeColumn,$columns))
                    {
                        $query .= "$primeColumn";
    $keyValue = $this->fields[$this->tablesToFields[$primeColumn][$tableName]];
                        $queryPart2 .= "'$keyValue'";
                        $first=FALSE;
                    }
                }
                else
                {
                    $needPrime=TRUE;
#                   $primeSelect = "SELECT $primeColumn from $tableName WHERE ";
```

```
            }
            reset($columns);
            while ($aColumn = each($columns))
            {
                $columnName = $aColumn['value'];
        $fieldValue = $this->fields[$this->tablesToFields[$columnName][$tableName]];
                if (!$first)
                {
                    $queryPart2 .= ", ";
                    $query .= ", ";
                    if ($needPrime) $primeSelect .= "AND ";
                }
                else
                    $first=FALSE;
                $query .= "$columnName";
                if (is_array($fieldValue))
                    $queryPart2 .= "'$fieldValue[$i]'";
                else
                    $queryPart2 .= "'$fieldValue'";
#               if ($needPrime)
#               {
#                   if (is_array($fieldValue))
#                       $primeSelect .= "$columnNames='fieldValue[$i]'";
#                   else
#                       $primeSelect .= "$columnName='$fieldValue'";
#               }
            }
            $query .= ") $queryPart2);";
#           echo "<br>".$query;
            $result = $this->db->query($query);
            if ($needPrime)
            {
                $lastOID = $this->db->getLastOID();
                $query = "SELECT $primeColumn FROM $tableName WHERE oid=$lastOID;";
                $result = $this->db->query($query);
#               $primeSelect .= ";";
#               $result = $this->db->query($primeSelect);
                $temp = $result->getRowAt(0);
                $keyField = $this->tablesToFields[$primeColumn][$tableName];
                $this->fields[$keyField] = $temp[0];
            }
        }
    }
    return $this->fields[$this->primeFields];
}

#############################################################
# Function: update                                          #
# Arguments: id                                             #
# Returns: ID or 0 if database errors                       #
# Requires: nothing                                         #
# Modifies: the data stored in DB                           #
#############################################################
function update($id)
{
    $dirtyTables = array();
    reset($this->dirty);
    while ($aField = each($this->dirty))
    {
        if ($aField['value'])
        {
```

```
                $affectedTables = $this->fieldsToTables[$aField['key']];
                reset($affectedTables);
                while ($aTable = each($affectedTables))
                {
                    if (in_array($aTable['key'],array_keys($dirtyTables)))
                        array_push($dirtyTables[$aTable['key']],
$this->fieldsToTables[$aField['key']][$aTable['key']]);
                    else
                        $dirtyTables[$aTable['key']]=
array($this->fieldsToTables[$aField['key']][$aTable['key']]);
                }
            }
        }
        reset($dirtyTables);
        while ($aTable = each($dirtyTables))
        {
            $tableName = $aTable['key'];
            $primeColumn = $this->tables[$tableName][0];
            $columns = $aTable['value'];
            $tablePrimeField = $this->tablesToFields[$primeColumn][$tableName];
            $arrayDepth=0;
            $fieldIsArray=array();
            $numColumns= count($columns);
            for ($i=0;$i<$numColumns;$i++)
            {
                $tempFieldName = $this->tablesToFields[$columns[$i]][$tableName];
                $fieldIsArray[$tempFieldName]=FALSE;
                $aColumn = $this->fields[$tempFieldName];
                if (is_array($aColumn))
                {
                    $fieldIsArray[$tempFieldName]=TRUE;
                    if (count($aColumn) > $arrayDepth)
                        $arrayDepth = count($aColumn);
                }
            }
            if (!$arrayDepth) # normal update
            {
                $tableKey = $this->tables[$tableName][0];
                $query = "UPDATE $tableName SET ";
                $dirtyFields = $aTable['value'];
                reset($dirtyFields);
                $first=TRUE;
                while ($aField = each($dirtyFields))
                {
                    $columnName = $aField['value'];
                    $fieldValue =
$this->fields[$this->tablesToFields[$columnName][$tableName]];
                    if (!$first) $query .= ", ";
                    $query .= " $columnName ='$fieldValue' ";
                    $first=FALSE;
                }
                $query .= "WHERE $tableKey = '$id';";
#           echo $query;
                $this->db->query($query);
            }
            else
            {
                ######################
                $objectType = get_class($this);
                $tempObject = new $objectType($this->db);
                $tempObject->setID($this->getID());
```

```
                $tempObject->retrieve();
#               echo printArray($tempObject);
                # find the array(s)
                $storedFields = $tempObject->fields;
                $storedIsArray = array();
                $storedDepth =1;
                reset($storedFields);
                while ($aField = each($storedFields))
                {
                    if (is_array($aField['value']))
                    {
                        $storedIsArray[$aField['key']]=TRUE;
                        if (($tempCount=count($aField['value']))>$storedDepth)
                            $storedDepth=$tempCount;
                    }
                }
                $inserts = array(); # for now we'll delete everything
----------------$deletes = array();#and reinsert, later well also check
----------------#for updates to avoid delete/insert pairs
----------------for($i=0;$i<$storedDepth;$i++)
----------------------$deletes[$i][0]=$tempObject->fields[$tablePrimeField];
----------------reset($columns);
----------------$j=0;
----------------while($aColumn=each($columns))
----------------{
----------------------if($primeColumn==$aColumn['value'])continue;
----------------------$j++;
----------------------$columnName=$aColumn['value'];
----------------------$fieldValue=
$tempObject->fields[$this->tablesToFields[$columnName][$tableName]];
----------------------if(is_array($fieldValue))
----------------------------for($i=0;$i<$storedDepth;$i++)
----------------------------------$deletes[$i][$j]=$fieldValue[$i];
----------------------else
----------------------------for($i=0;$i<$storedDepth;$i++)
----------------------------------$deletes[$i][$j]=$fieldValue;
----------------}
----------------for($i=0;$i<$arrayDepth;$i++)
----------------------$inserts[$i][0]=$this->fields[$tablePrimeField];
----------------reset($columns);
----------------$j=0;
----------------while($aColumn=each($columns))
----------------{
----------------------if($primeColumn==$aColumn['value'])continue;
----------------------$j++;
----------------------$columnName=$aColumn['value'];
----------------------$fieldValue=
$this->fields[$this->tablesToFields[$columnName][$tableName]];
----------------------if(is_array($fieldValue))
----------------------------for($i=0;$i<$arrayDepth;$i++)
----------------------------------$inserts[$i][$j]=$fieldValue[$i];
----------------------else
----------------------------for($i=0;$i<$arrayDepth;$i++)
----------------------------------$inserts[$i][$j]=$fieldValue;
----------------}
----------------for($i=0;$i<$storedDepth;$i++)
----------------{
----------------------$query="DELETE FROM $tableName WHERE $primeColumn='";
                    $query .= $deletes[$i][0];
                    reset($columns);
                    $j=0;
```

```
                    while ($aColumn=each($columns))
                    {
                        if ($primeColumn==$aColumn['value']) continue;
                        $j++;
                        $query .= "' AND ";
        ----------$query .= $aColumn['value'] . "='";
                        $query .= $deletes[$i][$j];
                    }
                    $query .= "';";
        ----------$result = $this->db->query($query);
        ----------}
        ----------for ($i=0;$i<$arrayDepth;$i++)
        ----------{
        ----------$query = "INSERT INTO $tableName ($primeColumn";
        ----------$query2 = "('".$inserts[$i][0]."'";
        ----------reset($columns);
        ----------$j=0;
        ----------while ($aColumn=each($columns))
        ----------{
        ----------if ($primeColumn==$aColumn['value']) continue;
        ----------$j++;
        ----------$query .= ",".$aColumn['value'];
        ----------$query2 .= ",'" . $inserts[$i][$j] ."'";
        ----------}
        ----------$query .= ") values $query2 );";
        ----------$result = $this->db->query($query);
        ----------}
        ----------}

        --------return 1;
        ----}

    ##############################################################
    #  Function: searchDB                                        #
    #  Arguments: none                                           #
    #  Returns: ID(s) or 0 if DB inaccessible or not found       #
    #  Requires: nothing                                         #
    #  Modifies: the data stored in DB                           #
    ##############################################################
    function searchDB()
    {
        --------$matches = array();
        --------reset($this->fields);
        --------$tablesInvolved = array();
        --------while ($field = each($this->fields))
        --------{
        ----------if ($field['value']=="") continue;
        ----------$tablesAffectedByField = $this->fieldsToTables[$field['key']];
        ----------reset($tablesAffectedByField);
        ----------while ($aTable = each($tablesAffectedByField))
        ----------{
        ----------if (in_array($aTable['key'], array_keys($tablesInvolved)))
        ----------array_push($tablesInvolved[$aTable['key']],
$this->fieldsToTables[$field['key']][$aTable['key']]);
        ----------else
        ----------$tablesInvolved[$aTable['key']]=
array($this->fieldsToTables[$field['key']][$aTable['key']]);
        ----------}
        --------}
        --------reset($tablesInvolved);
```

```
        $intermediateKeys=array();
        while ($aTable = each($tablesInvolved))
        {
            $primeColumn = $this->tables[$aTable['key']][0];
        $tableName = $aTable['key'];
        $columns = $aTable['value'];

        $query = "SELECT $primeColumn FROM $tableName WHERE ";
        reset($columns);
        $first=TRUE;
        while ($aColumn = each($columns))
        {
            $columnName = $aColumn['value'];
            $fieldValue =
$this->fields[$this->tablesToFields[$columnName][$tableName]];
            if ($fieldValue=="")
                $fieldValue = $intermediateKeys[$columnName];
            if (!$first)
                $query .= "AND ";
            else
                $first=FALSE;
            $query .= "$columnName = '$fieldValue'";
        }
        $query .= ";";
        $result = $this->db->query($query);
        $numResults = $result->numRows();
        if ($numResults==0)continue;
        $intermediateMatches = array();
        for ($i=0;$i<$numResults;$i++)
        {
            $aRow = $result->getRowAt($i);
            array_push($intermediateMatches, $aRow[0]);
        }
        if ($tablesToFields[$primeColumn][$tableName]==$primeField)
        {
            if (count($matches))
                $matches = array_intersect($matches,$intermediateMatches);
            else
                $matches=$intermediateMatches;
        }
        else
            $intermediateKeys[$primeColumn] = $intermediateMatches[0];
    }
    return $matches;
}

##############################################################
#  Function: getGeneralData                                  #
#  Arguments: key --- the label of the data to retrieve      #
#  Returns: value of key, if no value found returns ""       #
#  Requires: nothing                                         #
#  Modifies: nothing                                         #
##############################################################
function getGeneralData($key)
{
    return $this->fields["$key"];
}

##############################################################
#  Function: getID                                           #
#  Arguments: none                                           #
```

```
----#_Returns:_the_ID_of_this_person_____________________________#
    # Requires: nothing                                           #
    # Modifies: nothing                                           #
    ##############################################################
    function getID()
    {
        return $this->fields[$this->primeField];
    }

    ##############################################################
    # Function: setGeneralData                                    #
    # Arguments: key -- the label of the data to set              #
    #            value -- the value to set                        #
    # Returns: 1 if set, 0 if error (key not registered)          #
    # Requires: nothing                                           #
    # Modifies: fields                                            #
    ##############################################################
    function setGeneralData($key, $value)
    {
        if (!in_array($key, array_keys($this->fields)))
            return 0;
        if ($this->fields[$key]!=$value)
        {
            $this->fields[$key]=$value;
            $this->dirty[$key]=TRUE;
        }
        return 1;
    }

    ##############################################################
    # Function: setID                                             #
    # Arguments: id -- the id of the person                       #
    # Returns: 1 if set, 0 otherwise                              #
    # Requires: nothing                                           #
    # Modifies: nothing                                           #
    ##############################################################
    function setID($id)
    {
        $this->fields[$this->primeField] = $id;
        $this->dirty[$this->primeField] = TRUE;
        return 1;
    }

    ##############################################################
    # Function: addTable                                          #
    # Arguments: table -- the table to add                        #
    # Returns: nothong                                            #
    # Requires: nothing                                           #
    # Modifies: nothing                                           #
    ##############################################################
    function addTable($tableName,
                      $keyName,$keyField,
                      $columnNames, $fieldNames)
    {
        $numColumns = count($columnNames);
        $numFields  = count($fieldNames);
        if ($numColumns != $numFields)
            return 0;
        if (!in_array($keyField,$this->fields))
        {
            $this->fields[$keyField]="";
```

```
            $this->dirty[$keyField]="";
        }
        $this->tables[$tableName]=array($keyName,$columnNames);
        $this->tablesToFields[$keyName][$tableName]=$keyField;
        $this->fieldsToTables[$keyField][$tableName]=$keyName;
        $this->tableOrder[count($this->tableOrder)]=$tableName;
        for ($i=0;$i<$numColumns;$i++)
        {
            if (!in_array($fieldNames[$i],$this->fields))
            {
                $this->fields[$fieldNames[$i]]="";
                $this->dirty[$fieldNames[$i]]="";
            }
            $this->tablesToFields[$columnNames[$i]][$tableName]=$fieldNames[$i];
            $this->fieldsToTables[$fieldNames[$i]][$tableName]=$columnNames[$i];
        }
        return 1;
    }

    ##############################################################
    # Function: removeTable                                      #
    # Arguments: table -- the name of the table to remove        #
    # Returns: the removed table                                 #
    # Requires: nothing                                          #
    # Modifies: nothing                                          #
    ##############################################################
    function removeTable($table)
    {
        list($keyName, $columns) = $this->tables[$table];
        reset($columns);
        while($aColumn = each($columns))
        {
            $fieldName = $this->tablesToFields[$aColumn['value']][$table];
----------unset($this->tablesToFields[$aColumn['value']][$table]);
----------unset($this->fieldsToTables[$fieldName][$table]);
----------if (!count($this->tablesToFields[$aColumn['value']]))
----------------unset($this->tablesToFields[$aColumn['value']]);
----------if (!count($this->fieldsToTables[$fieldName]))
----------{
----------------unset($this->fields[$fieldName]);
----------------unset($this->dirty[$fieldName]);
----------------unset($this->fieldsToTables[$fieldName]);
----------}
--------}
--------unset($this->tables[$table]);
--------$start = $this->tableOrder[$table];
--------$end = count($this->tableOrder);
--------for ($i=$start;$i<$end-1;$i++)
----------$this->tableOrder[$i]=$this->tableOrder[$i+1];
--------unset($this->tableOrder[$i]);

----}

}
?>
```

Appendix C

Source Code: Web

C.1 Comp-Ina-Box Core

www/access-denied.php

```
<?php
/*****************************************************************
 * This file is part of CompInaBox.                              *
 * Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    *
 * CompInaBox is availible for license under the GPL, see        *
 * the COPYING file in the root directory of the install for     *
 * the full terms of the GPL.                                    *
 *                                                               *
 *                                                               *
 * File: access-denied.php                                       *
 * Type: Serveable Page                                          *
 * Author: Eric D. Nielsen                                       *
 * Description: User tried to view a page for which they lack    *
 *              permissions.  Links to the user profile update   *
 *              to request additional permissions.               *
 * Change Log: 8/20/02 -- created -- edn                         *
 *****************************************************************/
// cib_header will setup $user if logged in, otherwise $user is unset
// cib_header also establishes the database connection
// cib_header does not create the CIB_Display object as many scripts
//            use this header as well as viewable pages
include("include_others.inc");

$display = new CIB_Display($db);
$page    = $display->beginPage("CompInaBox",
                               "Access Denied");
$page .=<<<END_TEXT
You lack the appropriate permissions to view the page you requested.\n
If you are either a member of a competition staff or a team officer/
coordinator you can request additional permissions in your
<a href="$CIB_SECURE_URL/accounts/profile_management.php">profile</a>.
<p />
If you are the competition coordinator and you received this message,
```

```
anywhere within your competition site, please contact
<a href="mailto:$CIB_SERVER_ADMIN_EMAIL">$CIB_SERVER_ADMIN_NAME</a>, the
Site Administrator to resolve this issue.  Similarly, if you are the
captain/owner of a team and are being block from your team pages.
<p />
Return to the <a href="$CIB_SECURE_URL/">CompInaBox Main Page</a>.
<p />
END_TEXT;
$page .= $display->sectionBreak();
$page .= $display->compinaboxBox();
$page .= $display->endPage();
echo $page;
?>
```

www/comingsoon.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3c.org/TR/1999/REC-html401-1991224/loose.dtd">
<html lang="en">
<head>
    <meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
    <title>Feature Coming Soon</title>
</head>
<body bgcolor="white">
<h1>Feature Coming Soon</h1>
This site is under active development, some links have been inserted by the
developers to help design the layout and plan the future growth. We hope to
avoid this in the future and most of these links should become operational in
the near future. We apologize for the inconvience.
</body>
</html>
```

www/comp_summary.php

```
<?php
/*************************************************************
 * This file is part of CompInaBox.                          *
 * Copyright 2001-2002. Eric D. Nielsen, All rights reserverd *
 * CompInaBox is availible for license under the GPL, see    *
 * the COPYING file in the root directory of the install for *
 * the full terms of the GPL.                                *
 *                                                           *
 *                                                           *
 * File: comp_summary.php                                    *
 * Type: Serveable Page                                      *
 * Author: Eric D. Nielsen                                   *
 * Description: Summary details for competitions on this server *
 * Change Log: 9/20/02 -- created -- edn                     *
 *************************************************************/
// cib_header will setup $user if logged in, otherwise $user is unset
// cib_header also establishes the database connection
// cib_header does not create the CIB_Display object as many scripts
//            use this header as well as viewable pages

$compStatus = $HTTP_GET_VARS["type"];
$compStatuses = array("CONFIGURATION","PENDING ADMIN", "PLANNING",
                      "TESTING","OPEN","CLOSED","RUNNING",
                      "POST-COMP","ARCHIVE");
$dbCompStatuses = $compStatuses;
$compStatuses[] = "PUBLIC";  // statuses > testing
$compStatuses[] = "PRIVATE"; // statuses <= testing, requires auth
$compStatuses[] = "ALL";     // all comps,          requires auth
if (!in_array($compStatus,$compStatuses))
{
  $compStatus="PUBLIC";
}
if (in_array($compStatus,array("ALL","PRIVATE","CONFIGURATION",
                               "PENDING ADMIN","TESTING")))
{
  $requiresAuthentication=TRUE;
  $authorizationLevels=array(array("Type"=>"Site","Level"=>"Admin"));
}
else
{
  $requiresAuthentication=FALSE;
}
include("include_others.inc");
$display = new CIB_Display($db);
$page  = $display->beginPage("CompInaBox: Comp Summary",
                             "Summary of Hosted Competitions");


$page .= "<table border=\"1\">\n";

// display the help navbar
$page .= $display->helpBar();

$page .= "<tr valign=\"top\">\n";
$page .= "<td colspan=\"3\">\n";
$query = "SELECT name, realcomp, occurs_on, compname, compunix, compurl ";
$query.= "FROM comps NATURAL JOIN comp_status ";
if (in_array($compStatus,$dbCompStatuses))
{
```

```
  $query .= "WHERE comp_status.name='$compStatus' ";
}
else if ($compStatus=="Private")
{
  $query.= "WHERE statusid <=4 ";
}
else if ($compStatus=="Public")
{
  $query.= "WHERE statusid >4 ";
}
$query .= "ORDER BY statusid , occurs_on , compname";
$result = $db->query($query);
$numComps = $result->numrows();
$page .= "<table>";
$page .= "<tr><th>Name</th><th>Date</th><th>Details</th><th>Status</th><th>Action</th></tr>\n";
for ($i=0;$i<$numComps;$i++)
{
  list($status,$realComp,$date,$name,$unixname,$externalURL) =
    $result->getRowAt($i);
  if ($realComp!='t')
    $name .= " (Fake Comp)";
  if ($externalURL!="")
    $externalURL="<a href=\"$externalURL\">$externalURL</a>";
  else
    $externalURL="<i>No link provided</i>";
  $page .= "<tr><th>$name</th><td>$date</td><td>$externalURL</td><td>$status</td><td>";
  switch($status)
    {
    case "CONNFIGURATION" : $page .= "[Configure]"; break;
    case "PENDING ADMIN" : $page .= "[Approve]"; break;
    case "PLANNING" : $page .= "[Plan]"; break;
    case "TESTING" : $page .= "[Test Registration]"; break;
    case "OPEN" : $page .= "<a href=\"register/$unixname/\">Register</a>"; break;
    case "CLOSED" : $page .= "<a href=\"stats/$unixname/\"View Registrations</a>"; break;
    case "RUNNING" : $page .= "[???]"; break;
    case "POST-COMP" : $page .= "[Comp Reporting]"; break;
    case "ARCHIVE" : $page .= "[Results]"; break;
    default: $page.="REPORT BUG";
    }
  $page .= "</td></tr>\n";

}
if ($numComps==0)
{
  $page .= "<tr><td colspan=\"5\"><i>There are no competitions on this server".
           " at the requested phase of hosting.</i></td></tr>\n";
}
$page .= "</table></td>\n";
$page .= "</tr>\n";
$page .= "</table>\n";

// footer matter, OK to be below the fold
$page .= $display->sectionBreak();
$page .= $display->compinaboxBox();
$page .= $display->endPage();
echo $page;
?>
```

www/index.php

```
<?php
/*************************************************************
 * This file is part of CompInaBox.                          *
 * Copyright 2001-2002. Eric D. Nielsen, All rights reserverd *
 * CompInaBox is availible for license under the GPL, see    *
 * the COPYING file in the root directory of the install for *
 * the full terms of the GPL.                                *
 *                                                           *
 *                                                           *
 * File: index.php                                           *
 * Type: Serveable Page                                      *
 * Author: Eric D. Nielsen                                   *
 * Description: Home/index page, directs users to the correct *
 *              sub-area of the site.                        *
 * Change Log: 8/11/02 -- created -- edn                     *
 *************************************************************/
// cib_header will setup $user if logged in, otherwise $user is unset
// cib_header also establishes the database connection
// cib_header does not create the CIB_Display object as many scripts
//            use this header as well as viewable pages
include("include_others.inc");
$display = new CIB_Display($db);
$page  = $display->beginPage("CompInaBox",
"Welcome to the <a href=\"$CIB_SERVER_HOST_LINK\">$CIB_SERVER_HOST_NAME</a> CompInaBox Server");
$page .= "CompInaBox is a suite of computer tools aimed at assisting ".
        "organizations in organizing and executing a ballroom dance competition.";
$page .= "<table border=\"1\">\n";

// display the help navbar
$page .= $display->helpBar();

// begin the main table: first row is the list of hosted comps and either
// the logo and login panel or a double wide user (my) page
$page .= "<tr valign=\"top\">\n";
$page .= $display->hostedComps();

// logged in check should be superfluous, but just in case
if (isset($user) && $user->isLoggedIn())
{
  $page .= $display->userPage($user);
}
else
{
//  $page .= "<td>".$display->compinaboxLogo()."</td>\n";
  $page .= $display->loginCell();
}

$page .= "</tr>\n";

// second row of the main table: links to other comps and past results
// should still be above the fold
//$page .= "<tr valign=\"top\">\n";
//$page .= $display->compResults();
//$page .= $display->cibServers();
//$page .= $display->outsideComps();
//$page .= "</tr>\n";
$page .= "</table>\n";

// footer matter, OK to be below the fold
```

```
$page .= $display->sectionBreak();
$page .= $display->compinaboxBox();
$page .= $display->endPage();
echo $page;
?>
```

www/service_outtage.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3c.org/TR/1999/REC-html401-1991224/loose.dtd">
<html lang="en">
<head>
    <meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
    <title>CompInaBox Service Unavailible</title>
</head>
<body bgcolor="white">
<h1>CompInaBox Service Unavailible</h1>
We are experiencing technical difficulties and service is unavailible. Our
volunteer administrators are aware of the problem and will resolve it as fast
as they can. We apologize for the inconvience.
</body>
</html>
```

www/accounts/confirm_account.php

```
<?php
/*************************************************************
 * This file is part of CompInaBox.                          *
 * Copyright 2001-2002. Eric D. Nielsen, All rights reserverd *
 * CompInaBox is availible for license under the GPL, see    *
 * the COPYING file in the root directory of the install for *
 * the full terms of the GPL.                                *
 *                                                           *
 *                                                           *
 * File: confirm_account.php                                 *
 * Type: Serveable Page                                      *
 * Author: Eric D. Nielsen                                   *
 * Description: Collect data from user to create a new account *
 * Change Log: 8/19/02 -- created -- edn                     *
 *************************************************************/
// cib_header will setup $user if logged in, otherwise $user is unset
// cib_header also establishes the database connection
// cib_header does not create the CIB_Display object as many scripts
//            use this header as well as viewable pages
include("include_others.inc");

if (isset($user)) localRedirect("$CIB_BASE_URL/");
sleep(1);
$hash=$HTTP_GET_VARS["hash"];
$query = "SELECT * FROM user_hashes WHERE hash='$hash';";
$result = $db->query($query);
if (!$result->numrows())
{
  sleep(2);
  die ("Please double-check the url, the hash value is incorrect.");
}
session_register("hash");

$username_msg   = $HTTP_SESSION_VARS["username_msg"];
if ($username_msg!="")
     $username_msg = "<br /><font color=\"red\">$username_msg</font>";
$password_msg = $HTTP_SESSION_VARS["password_msg"];
if ($password_msg!="")
     $password_msg = "<br /><font color=\"red\">$password_msg</font>";

$challenge      = $HTTP_SESSION_VARS["challenge"];
$username       = $HTTP_SESSION_VARS["username"];
$nonce          = $HTTP_SESSION_VARS["nonce"];
if ($challenge!="" || $username!="" || $nonce!="")
{
  $testChallenge = MD5(session_id() .$username. $email . $nonce);
  if ($testChallenge!=$challenge)
    {
#      $logger->post("Intrusion","Session Hijack attempt at ".date().
#       "IP:$clientIP user: $userID");
      sleep(2);
      unset($security_userID);
      session_unregister("security_userID");
      unset($challenge); unset($username);
      session_unregister("challenge"); session_unregister("username");
      localRedirect("Location: $CIB_BASE_URL/");
      exit;
    }
}
```

```
$display = new CIB_Display($db);
$page   = $display->beginPage("CompInaBox",
                              "Account Comfirmation");
$page .= "<form action=\"SCRIPTS/activate_account.php\" method=\"POST\">\n";
$page .= "<table border=\"1\">\n";
$page .= "<tr><th>Username$username_msg</th><td><input type=\"text\" ".
         "name=\"username\" value=\"$username\" /></td></tr>\n";
$page .= "<tr><th>Password$password_msg</th><td><input type=\"password\" ".
         "name=\"password\" /></td></tr>\n";
$page .= "<tr><td colspan=\"3\"><input type=\"submit\" ".
         "value=\"Activate My Account\" /></td></tr>\n";
$page .= "</table></form>\n";

$page .= $display->sectionBreak();
$page .= $display->compinaboxBox();
$page .= $display->endPage();
echo $page;
?>
```

www/accounts/login.php

```
<?php
/*************************************************************
 * This file is part of CompInaBox.                          *
 * Copyright 2001-2002. Eric D. Nielsen, All rights reserverd *
 * CompInaBox is availible for license under the GPL, see    *
 * the COPYING file in the root directory of the install for *
 * the full terms of the GPL.                                *
 *                                                           *
 *                                                           *
 * File: login.php                                           *
 * Type: Serveable Page                                      *
 * Author: Eric D. Nielsen                                   *
 * Description: simple login page, used when a requested page *
 *              requires login and the user is not logged in *
 * Change Log: 9/16/02 -- created -- edn                     *
 *************************************************************/
// cib_header will setup $user if logged in, otherwise $user is unset
// cib_header also establishes the database connection
// cib_header does not create the CIB_Display object as many scripts
//            use this header as well as viewable pages
include("include_others.inc");

$display = new CIB_Display($db);
$page    = $display->beginPage("CompInaBox",
                               "The requested page requires authentication.");
$returnTo = $HTTP_SESSION_VARS["ReturnTo"];
$page .= $display->loginForm($returnTo);

// footer matter, OK to be below the fold
$page .= $display->sectionBreak();
$page .= $display->compinaboxBox();
$page .= $display->endPage();
echo $page;
?>
```

www/accounts/new_account.php

```
<?php
/*****************************************************************
 * This file is part of CompInaBox.                              *
 * Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    *
 * CompInaBox is availible for license under the GPL, see        *
 * the COPYING file in the root directory of the install for     *
 * the full terms of the GPL.                                    *
 *                                                               *
 *                                                               *
 * File: new_account.php                                         *
 * Type: Serveable Page                                          *
 * Author: Eric D. Nielsen                                       *
 * Description: Collect data from user to create a new account   *
 * Change Log: 8/13/02 -- created -- edn                         *
 *****************************************************************/
// cib_header will setup $user if logged in, otherwise $user is unset
// cib_header also establishes the database connection
// cib_header does not create the CIB_Display object as many scripts
//            use this header as well as viewable pages
include("include_others.inc");

$username_msg  = $HTTP_SESSION_VARS["username_msg"];
if ($username_msg!="")
     $username_msg = "<br /><font color=\"red\">$username_msg</font>";
$email_msg     = $HTTP_SESSION_VARS["email_msg"];
if ($email_msg!="")
     $email_msg = "<br /><font color=\"red\">$email_msg</font>";
$password1_msg = $HTTP_SESSION_VARS["password1_msg"];
if ($password1_msg!="")
     $password1_msg = "<br /><font color=\"red\">$password1_msg</font>";
$password2_msg = $HTTP_SESSION_VARS["password2_msg"];
if ($password2_msg!="")
     $password2_msg = "<br /><font color=\"red\">$password2_msg</font>";

$challenge     = $HTTP_SESSION_VARS["challenge"];
$username      = $HTTP_SESSION_VARS["username"];
$email         = $HTTP_SESSION_VARS["email"];
$nonce         = $HTTP_SESSION_VARS["nonce"];
if ($challenge!="" || $username!="" || $email!="" || $nonce!="")
{
  $testChallenge = MD5(session_id() .$username. $email . $nonce);
  if ($testChallenge!=$challenge)
    {
#      $logger->post("Intrusion","Session Hijack attempt at ".date().
#       "IP:$clientIP user: $userID");
      sleep(2);
      unset($security_userID);
      session_unregister("security_userID");
      unset($challenge); unset($username); unset($email);
      session_unregister("challenge"); session_unregister("username");
      session_unregister("email");
      localRedirect("Location: $CIB_BASE_URL/");
      exit;
    }
}
$display = new CIB_Display($db);
$page  = $display->beginPage("CompInaBox",
                             "New User Account");
if (isset($user))
```

```
{
  $page .= "You are currently logged in.  You do not need a new account.";
}
else
{
  $page .= "<form action=\"SCRIPTS/validate_account.php\" method=\"POST\">\n";
  $page .= "<table border=\"1\">\n";
  $page.=<<<END_PAGE
<tr><th>Username$username_msg</th>
<td><input type="text" name="username" value="$username" /></td>
<td>Legal usernames are at least 6 characters, begin with a letter, and
contain only alpha-numberic character and hyphens.  Usernames are public on
the site.</td></tr>
<tr><th>Email$email_msg</th>
<td><input type="text" name="email" value="$email"/></td>
<td>This site requires you to supply a valid, active email address.  After you
submit this page an email with directions for finishing your account creation
 will be sent to the address you provide.  This email address is <b>not</b>
displayed anywhere on the site.</td></tr>
<tr><th>Password$password1_msg</th>
<td><input type="password" name="password1" /></td>
<td>Passwords must be at least 6 characters in length, with no maximum length
enforced.  Legal passwords can and must include at least a single character
from each of the following sets, upper-case characters, lower-case characters,
numbers, and special-characters.  All the printable special characters on the
keyboard are allowed.<p />Passwords are stored on the server after encryption
with a one-way function.  We can <b>not</b> tell you your password if you
forget it.  We are not providing an e-mail reminder service at present as
these expose your password in cleartext on the network. <p /> We would
rather you write down you password than e-mail it to you as cracker attacks
on our server are more likely from web denizens than your room/office-mates.
We hope to prvoide a public key encrypted password reminder option in the
near future for users with key-pairs.</td></tr>
<tr><th>Confirm Password$password2_msg</th>
<td><input type="password" name="password2" /></td>
<td>Retype the same password to make sure you typed what you think you
typed.</td></tr>
<tr><td colspan="3">
<input type="submit" value="Email Me the Next Step" /></td></tr>
</table></form>
END_PAGE;
}
$page .= $display->sectionBreak();
$page .= $display->compinaboxBox();
$page .= $display->endPage();
echo $page;
?>
```

www/accounts/pending_account.php

```
<?php
/*************************************************************
 * This file is part of CompInaBox.                          *
 * Copyright 2001-2002. Eric D. Nielsen, All rights reserverd *
 * CompInaBox is availible for license under the GPL, see    *
 * the COPYING file in the root directory of the install for *
 * the full terms of the GPL.                                *
 *                                                           *
 *                                                           *
 * File: pending_account.php                                 *
 * Type: Serveable Page                                      *
 * Author: Eric D. Nielsen                                   *
 * Description: Let the user know that the account was created *
 *              and that an email was sent with instructions *
 *              for finishing the registration process.      *
 * Change Log: 8/25/02 -- created -- edn                     *
 *************************************************************/
// cib_header will setup $user if logged in, otherwise $user is unset
// cib_header also establishes the database connection
// cib_header does not create the CIB_Display object as many scripts
//            use this header as well as viewable pages
include("include_others.inc");

$display = new CIB_Display($db);
$page  = $display->beginPage("CompInaBox",
                             "Account Creating, Pending Activation");
$page .=<<<END_HTML
Your account has been created. Instructions for activating the account have
been sent to the email address you provided. After you visit the web page
listed in the email your account will be operational.
<br />
Thank you for registering with CompInaBox.
END_HTML;
$page .= $display->sectionBreak();
$page .= $display->sectionBreak();
$page .= $display->compinaboxBox();
$page .= $display->endPage();
echo $page;
?>
```

www/accounts/profile_management.php

```
<?php
/*************************************************************
 * This file is part of CompInaBox.                          *
 * Copyright 2001-2002. Eric D. Nielsen, All rights reserverd *
 * CompInaBox is availible for license under the GPL, see    *
 * the COPYING file in the root directory of the install for *
 * the full terms of the GPL.                                *
 *                                                           *
 *                                                           *
 * File: profile_management.php                              *
 * Type: Serveable Page                                      *
 * Author: Eric D. Nielsen                                   *
 * Description: Form to request escalation of user permissions *
 * Change Log: 8/20/02 -- created -- edn                     *
 *************************************************************/
// cib_header will setup $user if logged in, otherwise $user is unset
// cib_header also establishes the database connection
// cib_header does not create the CIB_Display object as many scripts
//            use this header as well as viewable pages
$authenticationRequired=TRUE;
include("include_others.inc");

$display = new CIB_Display($db);
$page  = $display->beginPage("CompInaBox",
                             "Account Profile");
if (!isset($user))
{
  localRedirect("Location: $CIB_BASE_URL");
  exit;
}
else
{
  $page .= "<h3>User Permissions</h3>\n";
  $page .= "[<a href=\"$CIB_SECURE_URL\">Return to Main Menu</a>]<br />\n";
  $page .= "Your account is currently only used to control which areas of
the site you are allowed to visit. In the future accounts may also allow for
auto-registration for competitions, but that would require us storing more
data about you than just email/username. The following table displays the
permissions aka Keys that you have been granted. There are three types of
keys, Site: Comp, and Team. Site keys are used by the people who run this
server for approving new competitions or adding links to the menus. Comp keys
are used by the organizers of a given competition to configure, update, and
process their competition. Team keys are used by officials of teams or studios
to handle team-related matters across any or all competitions on this server.
If you are only here to register for competitions then you need no keys.";
  $page .= $display->displayUserKeys($user);
  $form = $display->displayKeyRequest($user);
$page .=<<<END_FORM
If you abuse(send lots of requests that get denied) this, your account may be
revoked. After submitting your request, the "approval" authority for the the
paticular comp/team will review the request and approve/deny at some point.
You should expect a delay of a few hours to days for them to respond. You
will be notified by email when the approval authority has dealt with your
request.
<form action="$CIB_SECURE_URL/accounts/SCRIPTS/request_permissions.php" method="POST">
$form
<input type="submit" value="Request Permission" />
</form>
END_FORM;
```

```
}
$page .= $display->sectionBreak();
$page .= $display->compinaboxBox();
$page .= $display->endPage();
echo $page;
?>
```

www/accounts/SCRIPTS/activate_account.php

```
<?php
/*************************************************************
 * This file is part of CompInaBox.                          *
 * Copyright 2001-2002. Eric D. Nielsen, All rights reserverd *
 * CompInaBox is availible for license under the GPL, see    *
 * the COPYING file in the root directory of the install for *
 * the full terms of the GPL.                                *
 *                                                           *
 *                                                           *
 * File: activate_account.php                                *
 * Type: Interstitual Page                                   *
 * Author: Eric D. Nielsen                                   *
 * Description: Performs the first-time login of an account  *
 *              changes account from pending to active, makes *
 *              sure that only one user can use each hash    *
 * Change Log: 8/26/02 -- created -- edn                     *
 *************************************************************/
include("include_others.inc");

$username = $HTTP_POST_VARS["username"];
// magic quotes have already escaped single quotes, so the only dangerous
// characters for the database is a trailing backslash which would escape the
// close quote encasing the user name
if ($username[strlen($username)-1]=="\\")
     $username = substr($username,0,strlen($username)-1);

$password = $HTTP_POST_VARS["password"];
$confirm_hash = $HTTP_SESSION_VARS["hash"];
// Initialize values that might get displayed in response to errors
$login_msg   = "";
$validated      = FALSE;

$userID       = $db->getUserIdByName($username);
$hashed_pass = $db->getUserPasswordById($userID);
if ($hashed_pass!="" && // this user exists
    crypt($password,$hashed_pass)==$hashed_pass && // the password matches
    MD5($userID . $username)==$confirm_hash) // its the correct user
{
  $validated=TRUE;
}


if (!$validated)
{
  // escape special html characters in username and email
  // valid username would be safe, but invalid may contain nasty script
  // tricks that would be redisplayed on the new_account form when passed back
  $username = htmlspecialchars($username);
  $login_msg<<<END_MSG
There was a problem with your login, please double check that you are using
the username you selected when you registered and re-enter your password.
END_MSG;
  $nonce = MD5(session_id().microtime());
  $challenge = MD5(session_id() . $username . $nonce);
  session_register("nonce");
  session_register("challenge");
  session_register("username");
  session_register("login_msg");
  header("Location: ../confirm_account.php?hash=$confirm_hash");
```

```
    exit;
}

$user = new User($db);
$user->activate_user($userID);
$user->login($username,$password);
header("Location: $CIB_SECURE_URL/");
exit;
?>
```

www/accounts/SCRIPTS/grant_permission.php

```
<?php
/*************************************************************
 * This file is part of CompInaBox.                          *
 * Copyright 2001-2002. Eric D. Nielsen, All rights reserverd*
 * CompInaBox is availible for license under the GPL, see    *
 * the COPYING file in the root directory of the install for *
 * the full terms of the GPL.                                *
 *                                                           *
 *                                                           *
 * File: grant_permission.php                                *
 * Type: Interstitual Page                                   *
 * Author: Eric D. Nielsen                                   *
 * Description: Process grant/deny permission requests       *
 * Change Log: 8/20/02 -- created -- edn                     *
 *************************************************************/
$authenticationRequired=TRUE;
include("include_others.inc");
$userid = $user->getID();
$postVars = $HTTP_POST_VARS;
foreach ($postVars as $name=>$value)
{
  if (strpos($name,"actions")!==FALSE)
    {
      $firstHyphen = strpos($value,"-");
      $secondHyphen = strrpos($value,"-");
      $operation = substr($value,0,$firstHyphen);
      $grantee    = substr($value,$firstHyphen+1,$secondHyphen-$firstHyphen-1);
      $permission = substr($value,$secondHyphen+1);
      switch ($operation)
        {
        case "Unchanged":break;
        case "Grant" : $key=$db->getPendingKey($permission);
          if ($user->mayAuthorizeKey($key))
            $db->grantPermissionToUser($permission,$grantee);
          break;
        case "Deny" : $key=$db->getPendingKey($permission);
          if ($user->mayAuthorizeKey($key))
            $db->denyPermissionToUser($permission,$grantee);
          break;
        default: break;
        }
    }
}
header("Location: $CIB_SECURE_URL");
exit;
?>
```

www/accounts/SCRIPTS/login.php

```
<?php
/*************************************************************
 * This file is part of CompInaBox.                          *
 * Copyright 2001-2002. Eric D. Nielsen, All rights reserverd *
 * CompInaBox is availible for license under the GPL, see    *
 * the COPYING file in the root directory of the install for *
 * the full terms of the GPL.                                *
 *                                                           *
 *                                                           *
 * File: login.php                                           *
 * Type: Interstitual Page                                   *
 * Author: Eric D. Nielsen                                   *
 * Description: Handle all logins, redirects back to page that *
 *              required login                               *
 * Change Log: 9/01/02 -- created -- edn                     *
 *************************************************************/
include("include_others.inc");

$username = $HTTP_POST_VARS["username"];
// magic quotes have already escaped single quotes, so the only dangerous
// characters for the database is a trailing backslash which would escape the
// close quote encasing the user name
if ($username[strlen($username)-1]=="\\")
    $username = substr($username,0,strlen($username)-1);

$password = $HTTP_POST_VARS["password"];


$returnTo = $HTTP_POST_VARS["returnTo"];
if ($returnTo=="/")
{
  $returnTo="/$CIB_WEB_INSTALL_DIR/";
}
// stop directory traversals
$returnTo = str_replace("..","",$returnTo);

// old apache vunerability with lots of trailing slashes
ereg_replace("//+$","",$returnTo);

// Merge returnTo and secure_url
if ($CIB_USES_SSL!="NEVER")
{
  $fullURL = "https://";
}
else
{
  $fullURL = "http://";
}
$fullURL .=$CIB_HOSTNAME.$CIB_DOMAIN.$returnTo;

// Initialize values that might get displayed in response to errors
$login_msg  = "";

sleep(1); // limit attacks to one a second
$user = new User($db);
if ($user->login($username,$password))
{
  $security_userID=$user->getID();
  session_register("security_userID");
```

```
  $sessionID = session_id();
  $clientIP = $HTTP_SERVER_VARS["REMOTE_ADDR"];
  $security_hash = md5($security_userID . $sessionID . $clientIP);
  session_register("security_hash");
  header("Location: $fullURL");
  exit;
}
else
{
  // escape special html characters in username and email
  // valid username would be safe, but invalid may contain nasty script
  // tricks that would be redisplayed on the new_account form when passed back
  $username = htmlspecialchars($username);
  $login_msg = "There was a problem with your login, please double check that
you are using the username you selected when you registered and re-enter your
password.";
  $nonce = MD5(session_id().microtime());
  $challenge = MD5(session_id().$username.$nonce.$HTTP_REFERER);
  session_register("nonce");
  session_register("challenge");
  session_register("username");
  session_register("login_msg");
  header("Location: $CIB_SECURE_URL");
  exit;
}
?>
```

www/accounts/SCRIPTS/logout.php

```
<?php
/*************************************************************
 * This file is part of CompInaBox.                          *
 * Copyright 2001-2002. Eric D. Nielsen, All rights reserverd *
 * CompInaBox is availible for license under the GPL, see    *
 * the COPYING file in the root directory of the install for *
 * the full terms of the GPL.                                *
 *                                                           *
 *                                                           *
 * File: logout.php                                          *
 * Type: Interstitual Page                                   *
 * Author: Eric D. Nielsen                                   *
 * Description: Destroy the current session and redirect to the *
 *              main page.                                   *
 * Change Log: 9/20/02 -- created -- edn                     *
 *************************************************************/
include("include_others.inc");
session_unset();
session_destroy();
header("Location: $CIB_BASE_URL");
exit;
?>
```

www/accounts/SCRIPTS/request_permissions.php

```
<?php
/*************************************************************
 * This file is part of CompInaBox.                          *
 * Copyright 2001-2002. Eric D. Nielsen, All rights reserverd *
 * CompInaBox is availible for license under the GPL, see    *
 * the COPYING file in the root directory of the install for *
 * the full terms of the GPL.                                *
 *                                                           *
 *                                                           *
 * File: requests_permissions.php                            *
 * Type: Interstitual Page                                   *
 * Author: Eric D. Nielsen                                   *
 * Description: Add user permission escalation requests to the *
 *              database for later review                    *
 * Change Log: 8/20/02 -- created -- edn                     *
 *************************************************************/
$authenticationRequired=TRUE;
include("include_others.inc");
$userid = $user->getID();
$site_level=$HTTP_POST_VARS["site_level"];
$comp       =$HTTP_POST_VARS["comp_name"];
$comp_level=$HTTP_POST_VARS["comp_level"];
$team       =$HTTP_POST_VARS["team_name"];
$team_level=$HTTP_POST_VARS["team_level"];

if ($site_level==1)
{
  $db->addPending($userid,array("Type"=>"Site","Level"=>"Admin"));
}
if ($comp && $comp_level)
{
  if (!in_array($comp,$db->getCompUnixNames()))
    die("Non existent competition name");
  if (!in_array($comp_level,$db->getUserRoles("comp")))
    die("Non existent competition access level");
  $db->addPending($userid,array("Type"=>"Comp","Compname"=>$comp,
                                "Level"=>$comp_level));
}
if ($team && $team_level)
{
  if (!in_array($team,$db->getTeamIDs()))
    die("Non existent team name");
  if (!in_array($team_level,$db->getUserRoles("team")))
    die("Non existent competition access level");
  $db->addPending($userid,array("Type"=>"Team","TeamID"=>$team,
                                "Level"=>$team_level));
}

header("Location: ../profile_management.php");
exit;
?>
```

www/accounts/SCRIPTS/validate_account.php

```
<?php
/*************************************************************
 * This file is part of CompInaBox.                          *
 * Copyright 2001-2002. Eric D. Nielsen, All rights reserverd *
 * CompInaBox is availible for license under the GPL, see    *
 * the COPYING file in the root directory of the install for *
 * the full terms of the GPL.                                *
 *                                                           *
 *                                                           *
 * File: validate_account.php                                *
 * Type: Interstitual Page                                   *
 * Author: Eric D. Nielsen                                   *
 * Description: Test the user inputs for validity before     *
 *              generating the confirmation email            *
 * Change Log: 8/13/02 -- created -- edn                     *
 *************************************************************/
include("include_others.inc");

$username  = $HTTP_POST_VARS["username"];
$email     = $HTTP_POST_VARS["email"];
$password1 = $HTTP_POST_VARS["password1"];
$password2 = $HTTP_POST_VARS["password2"];

// Initialize values that might get displayed in response to errors
$username_msg  = "";
$email_msg     = "";
$password1_msg = "";
$password2_msg = "";
$validated     = TRUE;

// Test minimum length of username requirement
if (strlen($username)<6)
{
  $validated = FALSE;
  $username_msg.="The username must be at least six characters in length.";
}

// Ensure that the username begins with an alphabetical character
if (!(("a"<=$username[0] && $username[0]<="z") ||
      ("A"<=$username[0] && $username[0]<="Z")))
{
  $validated = FALSE;
  $username_msg.="The username must begin with either an upper or lower case,
unaccented character (A-Za-z).";
}

// Ensure that the username contains only alphanumberics and hyphens
$numSpecials = strlen(ereg_replace("[-A-Za-z0-9]","",$username));
if ($numSpecials)
{
  $validated = FALSE;
  $username_msg.="The username must contain only alpha-numeric characters and
yphens.";
}

// Ensure that the username isn't taken.  Note: this isn't race safe
//:TODO: make race safe
#if (!$db->availibleUserName())
#{
```

```
#  $validated = FALSE;
#  $username_msg.="That username has already been taken.";
#}

if (strlen($email)=="")
{
  $validated = FALSE;
  $email_msg.="You must provide an email.";
}

// Ensure that the email isn't taken.  Note: this isn't race safe
//:TODO: make race safe
#if (!$db->availibleEmail())
#{
#  $validated = FALSE;
#  $email_msg.="There is already an account using that email address.";
#}


// Test for minimum password length
if (strlen($password1)<6)
{
  $validated = FALSE;
  $password1_msg.="The username must be at least six characters in length.";
}

// Count the number of characters of each character class in password
$numLowerCase = strlen(ereg_replace("[^a-z]","",$password1));
$numUpperCase = strlen(ereg_replace("[^A-Z]","",$password1));
$numNumbers = strlen(ereg_replace("[^0-9]","",$password1));
$numSpecials = strlen(ereg_replace("[a-zA-Z0-9]","",$password1));

// Ensure that the password has at least one character from each class
if (!$numLowerCase || !$numUpperCase || !$numNumbers || !$numSpecials)
{
  $validated = FALSE;
  $password1_msg.="The password must contain at least a single character from
the following four classes:Upper Case (A-Z), Lower Case (a-z), Numbers (0-9),
and Special Characters (-!@#\$%^&*()_=+[{]}\\|;:'\",<.>/?`~).";
}
// Ensure that the two passwords match
if ($password1!=$password2)
{
  $validated = FALSE;
  $password2_msg.="The two passwords entered do not match.";
}

if (!$validated)
{
  //escape special html characters in username and email
  //valid username would be safe, but invalid may contain nasty script
  //tricks that would be redisplayed on the new account form when passed back
  $username = htmlspecialchars($username);
  $email    = htmlspecialchars($email);
  $nonce = MD5(session_id().microtime());
  $challenge = MD5(session_id().$username.$email.$nonce);
  session_register("nonce");
  session_register("challenge");
  session_register("username");
  session_register("email");
  session_register("username_msg");
```

```
__session_register("email_msg");
  session_register("password1_msg");
  session_register("password2_msg");
  header("Location: ../new_account.php");
  exit;
}

// escape special html characters in username for display on webform
// username restrictions have already gotten rid of "naughty" characters
// email has not had any tests (almost anything is allowed in an email
// if you do a true test for all legal emails)
// However, both values are being fully escaped in case the username character
// restrictions change.
$username  = htmlspecialchars($username);
$email     = htmlspecialchars($email);

$user = new User($db);
$user->createPendingAccount($username,$email,$password1);
header("Location: ../pending_account.php");
exit;
?>
```

C.2 SlidingDoors

All paths start from `www/register/SD`.

SD_MITBDT2002EDN.css

```
<!--
This file is part of SlidingDoors.
Copyright 2001-2002. Eric D. Nielsen. All rights reserved.
SlidingDoors is availible for license under the GPL, see
the COPYING file in the root directory of the install for
the full terms of the GPL.

File: SD_MITBDT2002EDN.css
Author: Thomas J. Nugent, Jr.
Description: Style Sheet used by the SlidingDoors application
Modified heavily from SD_MITBDT2002.css by Eric D. Nielsen to
increase browser independence.
-->
:link {
        color: blue;
        text-decoration: none;
}

A:visited{
        color: #00f;
        text-decoration: none;
}

A:hover {
        color: #f00;
        text-decoration: underline;
}

A:active {
        color: #ccc;
        text-decoration: underline;
}

#NavBar{
        margin: 4px auto 10px auto;
        padding: 1px 5px;
        font-size: 0.8em;
}

#NavBar A:link {
        color: #17e;
}

#NavBar A:visited{
        color: #666;
}

#NavBar A:hover {
        color: #f66;
}

#NavBar A:active {
```

```
        color: #fff;
}

.FormItemNames{
        font-weight: bold;
        font-size: 0.8em;
        background-color: #ccc;
        padding: 1px 4px;
        display: block;
}
```

back.php

```
<?php
#####################################################################
# This file is part of SlidingDoors                                 #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd        #
# SlidingDoors is availible for license under the GPL, see          #
# the COPYING file in the root directory of the install for         #
# the full terms of the GPL.                                        #
#                                                                   #
# File: back.php                                                    #
# Author: Eric D. Nielsen                                           #
# Description: This page is diplayed when an interal error          #
#              occurs, typicall due to back arrow use.              #
# Internal Links: index.php                                         #
# External Links: 2002 MITBDT Open DanceSport Competition page      #
#                 SlidingDoors information page                     #
#                 Comp-in-a-box information page                    #
#                 OpenImpetus information page                      #
# Change Log: 2/22/02 -- created -- edn                             #
#####################################################################
include "include_others.inc";
session_start();
session_destroy();
$display = new HTMLDisplay();
$page = $display->beginPage();
$page .= $display->instructionBox("Navigation Error",
"This registration site is heavily dynamic and its operation depends upon the
user only navigating through the internal navigation features. Specifically,
<ul>
<li><b>Do NOT use the Back/Forward buttons on your browser</b></li>
<li>Do NOT bookmark pages past the home/index page, except for statistic pages
</li>
</ul>

We apologize for the inconvience. You can <a href=\"". $baseURL."\">
restart</a> your registration.
<br>
We look forward to seeing you at our competition.
");
$page .= $display->sectionBreak();
$page .= $display->compinaboxBox();
$page .= $display->endPage();
clean_up_session();
echo $page;
?>
```

done.php

```
<?php
#################################################################
# This file is part of SlidingDoors                             #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    #
# SlidingDoors is availible for license under the GPL, see      #
# the COPYING file in the root directory of the install for     #
# the full terms of the GPL.                                    #
#                                                               #
# File: done.php                                                #
# Author: Eric D. Nielsen                                       #
# Description: This is a "final" page to exit the add partner   #
#              page loop.  It provides links back to the        #
#              index page, comp homepage, and stats pages.      #
# Internal Links: index.php                                     #
#                 2002 MITBDT Open DanceSport Competition page  #
#                 by-school.php                                 #
# External Links: 2002 MITBDT Open DanceSport Competition page  #
#                 SlidingDoors information page                 #
#                 Comp-in-a-box information page                #
#                 OpenImpetus information page                  #
# Change Log: 2/22/02 -- created -- edn                         #
#################################################################
include "include_others.inc";
session_start();
$propagateEmail = $HTTP_POST_VARS["propagateEnteredEmail"];
$primeAffil = $HTTP_POST_VARS["propagateAffil"];
if (!ereg("^[-0-9]*$",$primeAffil)) $primeAffil="";
$display = new HTMLDisplay($db);
$page = $display->beginPage();
$page.=$display->instructionBox("Registration Complete",
"Your registration has been saved and the registration stats updated.
Thank you for registering.  Please do not
use the back arrow as the session information has been destroyed.  If you
wish to make corrections or review your registrations in the future please
log back in through the second option on the <a href=\"$baseURL\">main</a>
registration page.<p />
<form action=\"prime.php\" method=\"POST\">
<input type=\"hidden\" name=\"propagateAffil\" value=\"$primeAffil\" />
<input type=\"hidden\" name=\"propagateEnteredEmail\"
value=\"$propagateEmail\" />
<input type=\"submit\" value=\"Register Another\" />
</form><p />
For more information:<br />
<ul>
<li><a href=\"http://$SD_comp_url\">Competition Home Page</a> for rules,
logistics, etc</li>
<li><a href=\"$baseURL/Stats/events.html\">Registration Statistics</a> for
numbers of competitors entered in events/from various affiliations</li>
</ul>
<p>
We look forward to seeing you at our competition.");
$page .= $display->sectionBreak();
$page .= $display->compinaboxBox();
$page .= $display->endPage();
session_destroy();
clean_up_session();
echo $page;
?>
```

home.html

```
<!--
#################################################################
# This file is part of SlidingDoors                             #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    #
# SlidingDoors is availible for license under the GPL, see      #
# the COPYING file in the root directory of the install for     #
# the full terms of the GPL.                                    #
#                                                               #
# File: home.html                                               #
# Author: Eric D. Nielsen                                       #
# Description: This file contains links to the real pages and   #
#              is used primarily to capture directory divers.   #
# Internal Links: index.php                                     #
# External Links: 2002 MITBDT Open DanceSport Competition page  #
#                 SlidingDoors information page                 #
#                 Comp-in-a-box information page                #
#                 OpenImpetus information page                  #
# Change Log: 2/22/02 -- created -- edn                         #
#################################################################
-->
<HTML>
<HEAD>
<TITLE>
Sliding Doors: Redirection
</TITLE>
</HEAD>
<BODY>
The page you are looking for is <A HREF="index.php">here</A>.
<HR>
This page was generated by
<A HREF="http://mitbdt.mit.edu/CompInABox/SlidingDoors/">SlidingDoors</A> a
<A HREF="http://mitbdt.mit.edu/CompInABox/">Comp-in-a-Box</A> component.
Copyright 2001, <A HREF="mailto:ericnielsen@earthlink.net">Eric D. Nielsen</A>.
<BR>
Sliding Doors is an outgrowth of Dave Leung's
<A HREF="http://mitbdt.mit.edu/OI/">OpenImpetus</A>
competition registration software.
</BODY>
</HTML>
```

index.php

```
<?php
#################################################################
# This file is part of SlidingDoors                             #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    #
# SlidingDoors is availible for license under the GPL, see      #
# the COPYING file in the root directory of the install for     #
# the full terms of the GPL.                                    #
#                                                               #
# File: index.php                                               #
# Author: Eric D. Nielsen                                       #
# Description: The "Home Page" for on-line registration for     #
#              the 2002 MITBDT Open DanceSport Competition.     #
#              This page was generated by SlidingDoors and      #
#              care should be taken in modifying it by hand.    #
#              This page drops sessions.                        #
# Internal Links: prime.php                                     #
#                 prime-retreive.php                            #
#                 by-school.php                                 #
#                 charge.php (secure)                           #
# External Links: 2002 MITBDT Open DanceSport Competition page  #
#                 SlidingDoors information page                 #
#                 Comp-in-a-box information page                #
#                 OpenImpetus information page                  #
# Change Log: 6/12/01 -- created -- edn                         #
#             3/14/02 -- prepped for PrepStep -- edn            #
#             6/26/02 -- GPL -- edn
#################################################################
# Do not remove the following line

include_once "include_others.inc";
#include "simple_header.inc";
session_start();
clean_up_session();
$display = new HTMLDisplay($db);
$page = $display->beginPage($SD_compname,"ONLINE REGISTRATION FOR<br />$SD_compname");
$page .= $display->sectionBreak();
$page .= $display->compInfoBox();
$page .= $display->sectionBreak();
echo $page;
?>
<table bgcolor="<?php echo $SD_background_color;?>" cellspacing=4 cellpadding=6>
   <tr valign="top">
      <td align="left"><!-- column 1 -->
<?php echo $display->registerBox();?>
      </td>
      <td align="left"> <!-- column 2 -->
<?php  echo $display->browseBox();?>
      </td>
   </tr>
   <tr valign="top">
      <td><!-- column 1, row 2 -->
<?php  echo $display->retrieveBox();?>
      </td>
      <td align="left" rowspan=2><!-- column 1, rows 2 & 3 -->
<?php echo $display->statsBox(); ?>
      </td>
   </tr>
   <tr>
      <td><!-- column 1, row 3 -->
```

```
<?php echo $display->groupAdminBox(); ?>
      </td>
   </tr>
</table>
<?php
echo $display->sectionBreak();
echo $display->compinaboxBox();
echo $display->endPage();
clean_up_session();
?>
```

partner-collision-action.php

```
<?php
#################################################################
# This file is part of SlidingDoors                             #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    #
# SlidingDoors is availible for license under the GPL, see      #
# the COPYING file in the root directory of the install for     #
# the full terms of the GPL.                                    #
#                                                               #
# File: partner-collision-action.inc                            #
# Author: Eric D. Nielsen                                       #
# Description: An interstitial page that sets the proper        #
#              values based on the users response to the close  #
#              match page.  Adds the new person/affil if        #
#              needed.  Dispatches to partners.php to resume    #
#              Step Two.                                        #
# Internal Links: partners.php                                  #
# External Links: SlidingDoors information page (footer)        #
#                 Comp-in-a-box information page (footer)       #
#                 OpenImpetus information page (footer)         #
# Change Log: 6/12/01 -- created -- edn                         #
#################################################################

    session_start();
include "include_others.inc";
$contraCheck= new ContraCheck($db);
$firstName = $HTTP_SESSION_VARS["formVars"]["FirstName"];
$lastName = $HTTP_SESSION_VARS["formVars"]["LastName"];
$email = $HTTP_SESSION_VARS["formVars"]["Email"];
$affil = $HTTP_SESSION_VARS["formVars"]["Affil"];
$newAffil = $HTTP_SESSION_VARS["formVars"]["NewAffil"];
$affilShort = $HTTP_SESSION_VARS["formVars"]["AffilShort"];
$age = $HTTP_SESSION_VARS["formVars"]["age"];
$package = $HTTP_SESSION_VARS["formVars"]["Package"];
$primeID =$HTTP_SESSION_VARS["primeID"];
$role = $HTTP_SESSION_VARS["role"];
$events = $HTTP_SESSION_VARS["formVars"]["Events"];
$oldPartnerID = $HTTP_SESSION_VARS["oldPartnerID"];
$partnerID = $HTTP_POST_VARS["CHOICE"];
if ($partnerID != -1)
{
    session_unregister("conflicts");
    session_unregister("formVars");
    session_register("primeID");
    $couple = new Couple($db);
    $couple->setLeader(($role=="leading"? $primeID : $partnerID));
    $couple->setFollower(($role=="leading"? $partnerID : $primeID));
    $coupleCheck = $couple->searchDB();
    if (count($coupleCheck))
    {
        $couple->setID($coupleCheck[0]);
        $couple->retrieve();

        $multiPartnerConflicts = $contraCheck->checkCoupleConflicts($couple,$events);
        $compRulesConflicts = $contraCheck->checkCoupleCompRules($couple,$events);
        if (!$multiPartnerConflicts->result || !$compRulesConflicts->result)
        {
            # setup variables
            $couple=$coupleID;
            $formVars["Events"]=array();
```

```
            $formVars["EventsMsg"]=$multiPartnerConflicts->getErrorMessage() .
                $compRulesConflicts->getErrorMessage();
            session_register("primeID");
            session_register("couple");
            session_register("partnerID");
            session_register("formVars");
            localRedirect("Location: $baseURL/partners.php#reg");
            exit;
        }

        $currentEvents = $couple->getDances();
        if ($oldPartnerID!=$partnerID)
            $events = arrayJoin($currentEvents, $events);
    }
    $couple->postToDB();
    $couple->setEvents($events);
    $multiPartnerConflicts = $contraCheck->checkCoupleConflicts($couple,$events);
    $compRulesConflicts = $contraCheck->checkCoupleCompRules($couple,$events);
    if (!$multiPartnerConflicts->result || !$compRulesConflicts->result)
    {
        # setup variables
        $couple->remove();
        $formVars["EventsMsg"]=$multiPartnerConflicts->getErrorMessage() .
            $compRulesConflicts->getErrorMessage();
        session_register("primeID");
        session_register("formVars");
        localRedirect("Location: $baseURL/partners.php#reg");
        exit;
    }
    $couple->postToDB();
    if ($oldPartnerID!="" && $oldPartnerID!=$partnerID)
    {
        $oldCouple = new Couple($db);
        $oldCouple->setLeader(($role=="leading" ? $primeID : $oldPartnerID));
        $oldCouple->setFollower(($role=="leading" ? $oldPartnerID : $primeID));
        $oldCheck = $oldCouple->searchDB();
        $oldCouple->setID($oldCheck[0]);
        $oldCouple->remove();
        $oldPerson = new Person($db);
        $oldPerson->setID($oldPartnerID);
        $oldPerson->remove();
    }
#    $db->rebuildStats();
    localRedirect("Location: $baseURL/partners.php");
    exit;
}


if ($affil=="")
{
    $affilObj = new RegOrg($db);
    $affilObj->setName($newAffil);
    $affilObj->setAbbrev($affilShort);
        $affilObj->setOrgPays(TRUE);
    $affilObj->postToDB();
    $affilID = $affilObj->getID();
    if (!$affilID)
    {
        errorPage($PHP_SELF, "root","Database Inaccessible",
                "utilities::addAffiliation unable to access database",TRUE);
        exit;
```

```
    }
}
else if ($affil!=-1)
{
        $affilID=$affil;
#    $affilObj = new Organization($db);
#    $affilObj->setID($affil);
#    $temp = $affilObj->searchDB();
#        $affilID=$temp[0];
}
$person = new Person($db);
$person->setFirstName($firstName);
$person->setLastName($lastName);
$person->setEmail($email);
if ($affil!=-1)
    $person->setAffilID($affilID);
else
    $person->setOrgPays(FALSE);
$person->setPaid(FALSE);
$person->setAgeLevel($age);
$person->setPackage($package);
$person->postToDB();
$partnerID = $person->getID();
if (!$primeID)
{
    errorPage($PHP_SELF, "root","Database Inaccessible",
        "utilities::addPerson unable to access database",TRUE);
    exit;
}
$couple = new Couple($db);
$couple->setLeader(($role=="leading"? $primeID : $partnerID));
$couple->setFollower(($role=="leading"? $partnerID : $primeID));
$couple->postToDB();
$coupleID=$couple->getID();
$multiPartnerConflicts = $contraCheck->checkCoupleConflicts($couple,$events);
$compRulesConflicts = $contraCheck->checkCoupleCompRules($couple,$events);
if (!$multiPartnerConflicts->result || !$compRulesConflicts->result)
{
    # setup variables
    $couple->remove();
    $person->remove();
            $formVars["Events"]=array();
    $formVars["EventsMsg"]=$multiPartnerConflicts->getErrorMessage() .
        $compRulesConflicts->getErrorMessage();
    session_register("primeID");
    session_register("formVars");
    localRedirect("Location: $baseURL/partners.php#reg");
    exit;
}

$couple->setEvents($events);
$couple->postToDB();
#$db->rebuildStats();

session_register("primeID");
localRedirect("Location: $baseURL/partners.php");
exit;
?>
```

partner-collision.php

```
<?php
#################################################################
# This file is part of SlidingDoors                             #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    #
# SlidingDoors is availible for license under the GPL, see      #
# the COPYING file in the root directory of the install for     #
# the full terms of the GPL.                                    #
#                                                               #
# File: partner-collision.php                                   #
# Author: Eric D. Nielsen                                       #
# Description: This page queries the use when a near match on   #
#              their entered partner information occurs, ie     #
#              they are trying to register and their partner    #
#              already did either with correct or incorrect     #
#              spellings.                                       #
# Internal Links: partner-collision-action.php                  #
# External Links: SlidingDoors information page (footer)        #
#                 Comp-in-a-box information page(footer)        #
#                 OpenImpetus information page  (footer)        #
# Change Log: 6/12/01 -- created -- edn                         #
#################################################################
# Do not remove the following line
include_once "include_others.inc";
session_start();
restrictNavigation(array ( array("PAGE"=>"$baseURL/partners.php",
                                 "VARS"=>array("formVars","conflicts"))));

$formVars = $HTTP_SESSION_VARS["formVars"];
$role = $HTTP_SESSION_VARS["role"];
$primeID = $HTTP_SESSION_VARS["primeID"];
$action = $HTTP_SESSION_VARS["action"];
$oldPartnerID=$HTTP_SESSION_VARS["partnerID"];
?>
<HTML>
<HEAD>
<TITLE>Registration: Step Two and a Half</TITLE>
</HEAD>
<BODY BGCOLOR=WHITE>
<H1>Registration: Step Two and a Half: Close Matches</H1>
The information you entered for your partner closely matched an entry already
in the database.  If your partner has multiple partners it's likely that a
close match is, in fact, a real match.  If any of the lower rows contains an
entry for the partner you were entering, select that row.  Otherwise select
the top row and a new entry will be created for the partner you just entered.
<?php
    echo"<CENTER>\n"  $db->nameCollisionTable($HTTP_SESSION_VARS["conflicts"],
                                              $HTTP_SESSION_VARS["formVars"],
                                              "partner-collision-action.php");
echo"</CENTER>\n";
?>
<?php include"footer.inc";
session_register("formVars");
session_register("role");
session_register("primeID");
session_register("action");
session_register("oldPartnerID");
?>
```

partners-dropped.php

```
<?php
#################################################################
# This file is part of SlidingDoors                             #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    #
# SlidingDoors is availible for license under the GPL, see      #
# the COPYING file in the root directory of the install for     #
# the full terms of the GPL.                                    #
#                                                               #
# File: partners-dropped.php                                    #
# Author: Eric D. Nielsen                                       #
# Description: This page confirms a users request to cancel     #
#              registration.                                    #
# Internal Links: index.php                                     #
# External Links: SlidingDoors information page (footer)        #
#                 Comp-in-a-box information page(footer)        #
#                 OpenImpetus information page  (footer)        #
# Change Log: 8/23/01 -- created -- edn                         #
#             7/05/02 -- GPL, display -- edn                    #
#################################################################
# Do not remove the following line
include "include_others.inc";
session_start();
$display = new HTMLDisplay($db);
$page = $display->beginPage();
$retrievalMethod = (in_array("Email",$SD_lookup_by) ? "email" : "");
$retrievalMethod .=(in_array("Name",$SD_lookup_by) ?
                    ($retrievalMethod!="" ? " or name" : "name") :
                    "");
$retrievalMethod .=(in_array("ID Number",$SD_lookup_by) ?
                    ($retrievalMethod!="" ? " or ID number" : "ID number") :
                    "");

$page .= $display->InstructionBox(
"Registration Cancelled",
"All your registrations have been cancelled.  Your identification is
stored on file until after registration closes.  If you change your mind about
registration, just use your $retrievalMethod to retrieve your personal identification.

Back to the <a href=\"$baseURL/index.php\">Index</a> page.");
$page .= $display->sectionBreak();
$page .= $display->compinaboxBox();
$page .= $display->endPage();
clean_up_session();
echo $page;
?>
```

partners.php

```
<?php
#################################################################
# This file is part of SlidingDoors                             #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    #
# SlidingDoors is availible for license under the GPL, see      #
# the COPYING file in the root directory of the install for     #
# the full terms of the GPL.                                    #
#                                                               #
# File: partners.php                                            #
# Author: Eric D. Nielsen                                       #
# Description: This page contains the information about the     #
#              and all their partners and allows people to      #
#              add/drop/modify partners.                        #
# Internal Links: partners-action.php                           #
# External Links: SlidingDoors information page (footer)        #
#                 Comp-in-a-box information page(footer)        #
#                 OpenImpetus information page  (footer)        #
# Change Log: 6/12/01 -- created -- edn                         #
#################################################################
# Do not remove the following line
include "include_others.inc";
session_start();
requirePrimeID();
if ($HTTP_SESSION_VARS["propagateEnteredEmail"]!="")
   $preloadEmail=$HTTP_SESSION_VARS["propagateEnteredEmail"];
else
   $preloadEmail="";
$display = new HTMLDisplay($db);
$page = $display->beginPage();
$page .= $display->instructionBox("Registration: Step Two: Partners and Events",
"The table below lists all your partners and your current registration.
From this page you can modify your information to correct misspellings
or add/drop/modify your registration with your partners.");
$page .= $display->sectionBreak();
$formVars = $HTTP_SESSION_VARS["formVars"];
$primeID = $HTTP_SESSION_VARS["primeID"];
if ($primeID=="")
        $primeID = $HTTP_POST_VARS["primeID"];
$couple = $HTTP_SESSION_VARS["couple"];
if ($couple=="")
        $couple = $HTTP_POST_VARS["couple"];

$person = new Competitor($db);
$person->setID($primeID);
$person->retrieve();
if (in_array("Affiliation",$SD_elements))
   $primeAffil = $person->getAffilID();
else
   $primeAffil="";
$box = $display->errorBox($HTTP_SESSION_VARS["formVars"]["updateFirstNameMsg"].
                          $HTTP_SESSION_VARS["formVars"]["updateLastNameMsg"].
                          $HTTP_SESSION_VARS["formVars"]["updateEmailMsg"].
                          $HTTP_SESSION_VARS["formVars"]["updateAffilIDMsg"]);
if ($box!="")
    $page .= $box . $display->sectionBreak();
$page .= $display->textBox($display->personalInfoUpdateBar($person),"95%",
                           'This_is_the_prime_update_form');
$page .= $display->sectionBreak();
```

```
$page .= $display->doneBox($primeAffil,$preloadEmail);
$box = $display->partnerTable($person,TRUE,$couple);
$page .= $display->textBox($box,"95%","This is the partner/event registration table");
$page .= $display->doneBox($primeAffil,$preloadEmail);
$page .= $display->sectionBreak() . $display->compinaboxBox();
$page .= $display->endPage();
clean_up_session();
echo $page;
?>
```

prime-collision.php

```
<?php
#################################################################
# This file is part of SlidingDoors                             #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    #
# SlidingDoors is availible for license under the GPL, see      #
# the COPYING file in the root directory of the install for     #
# the full terms of the GPL.                                    #
#                                                               #
# File: prime-collision.php                                     #
# Author: Eric D. Nielsen                                       #
# Description: This page queries the use when a near match on   #
#              their entered personal information occurs, ie    #
#              they are trying to register than their partner   #
#              already did either with correct or incorrect     #
#              spellings.                                       #
# Internal Links: prime-collision-action.php                    #
# External Links: SlidingDoors information page (footer)        #
#                 Comp-in-a-box information page(footer)        #
#                 OpenImpetus information page  (footer)        #
# Change Log: 6/12/01 -- created -- edn                         #
#             4/23/02 -- modified for better interface          #
#################################################################
# Do not remove the following line
include "include_others.inc";
session_start();
session_register("formVars");
$conflicts = $HTTP_SESSION_VARS["conflicts"];
$display = new HTMLDisplay($db);

restrictNavigation(array(array("PAGE"=>"$baseURL/prime.php",
                               "VARS"=>array("formVars"))));
$page = $display->beginPage();
$page .= $display->instructionBox(
          "Registration: Step One and a Half: Close Matches",
"The information you have entered appears to match an entry already in our
database.  Each possible match is shown with their personal identification
information and any registrered partners.  You can update the spelling by
using the top button in the pair, or switch to the detailed view by using
the bottom button.  If none of the shown matches are actually you, select the
absolute bottom button of the table below.");
$page .= $display->sectionBreak();

$page .= $display->conflictResolutionBox($conflicts,
                                         $HTTP_SESSION_VARS["formVars"]);
$page .= $display->sectionBreak();
$page .= $display->compinaboxBox();
$page .= $display->endPage();
clean_up_session();
session_register("formVars");
echo $page;
?>
```

prime-email-collision.php

```
<?php
#################################################################
# This file is part of SlidingDoors                             #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    #
# SlidingDoors is availible for license under the GPL, see      #
# the COPYING file in the root directory of the install for     #
# the full terms of the GPL.                                    #
#                                                               #
# File: prime-email-collision.php                               #
# Author: Eric D. Nielsen                                       #
# Description: This page queries the use when a multiple match #
#              on the entered email occurs.                     #
# Internal Links: prime-email-collision-action.php              #
# External Links: SlidingDoors information page (footer)        #
#                 Comp-in-a-box information page(footer)        #
#                 OpenImpetus information page  (footer)        #
# Change Log: 6/12/01 -- created -- edn                         #
#################################################################
# Do not remove the following line
include "include_others.inc";
session_start();
session_register("email");
restrictNavigation(array (array("PAGE"=>"$baseURL/index.php",
                                "VARS"=>array("primeID")),
                          array("PAGE"=>"$baseURL/",
                                "VARS"=>array("primeID"))));
$display= new HTMLDisplay($db);
$page = $display->beginPage();
$page .= $display->instructionBox(
        "Registration: Step One and a Half: Close Matches",
"There are multiple people associated with the email you entered.  This usually
happens if a studio/school uses the same email for all entrants, or if a
couple uses the same email for both members.  Below are listed the people who
use this email address, please select the record you wish to retrieve.  If
the only choices are you and your partner, don't worry, selecting either
option will yield the same result.");
$page .= $display->sectionBreak();
$page .= $display->textBox($display->emailCollisionTable(
                $HTTP_SESSION_VARS["primeID"],
                "SCRIPTS/prime-email-collision-action.php"));
$page .= $display->sectionBreak();
$page.= $display->compinaboxBox();
$page .= $display->endPage();
clean_up_session();
echo $page;
?>
```

prime-update-affil.php

```
<?php
#####################################################################
# This file is part of SlidingDoors                                 #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd        #
# SlidingDoors is availible for license under the GPL, see          #
# the COPYING file in the root directory of the install for         #
# the full terms of the GPL.                                        #
#                                                                   #
# File: prime-update-affil.inc                                      #
# Author: Eric D. Nielsen                                           #
# Description: A displayed page to allow a user to enter a new      #
#              affiliation at a time later than initial             #
#              registration.                                        #
# Internal Links: prime.php                                         #
#                 prime-collision.php                               #
#                 partners.php                                      #
# External Links: SlidingDoors information page (footer)            #
#                 Comp-in-a-box information page (footer)           #
#                 OpenImpetus information page (footer)             #
# Change Log: 8/10/01 -- created -- edn                             #
#####################################################################

session_start();
include "include_others.inc";
$primeID = $HTTP_SESSION_VARS["primeID"];
$formVars = $HTTP_SESSION_VARS["formVars"];
session_register("primeID");
session_register("formVars");
$display = new HTMLDisplay($db);
$page = $display->beginPage();
$page .= $display->instructionBox("Create New Affiliation",
"Please enter both the full name and an abbreviated version of the affiliation.\n
If there is no good abbreviation, please just enter the same name both places.\n
If the affiliation was successfully created you will be returned to the\n
registration summary page and should see your new affiliation selected.\n
\n
<form action=\"SCRIPTS/prime-update-affil-action.php\" method=\"POST\">\n
<table>\n
<tr><td>Full Name:<input type=\"text\" name=\"affilName\">\n
Abbreviated Form:<input type=\"text\" name=\"affilShort\">\n
<input type=\"submit\" value=\"Create and Update\" style=\"$SD_button_style\">\n
</td></tr></table></form>\n");
$page .= $display->sectionBreak() . $display->compinaboxBox();
$page .= $display->endPage();
echo $page;
?>
```

prime-update-collision-action.php

```
<?php
#####################################################################
# This file is part of SlidingDoors                                 #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd        #
# SlidingDoors is availible for license under the GPL, see          #
# the COPYING file in the root directory of the install for         #
# the full terms of the GPL.                                        #
#                                                                   #
# THIS FILE IS BEING DEPRECATED/ COMBINED WITH                      #
# SCRIPTS/prime-collision-action.php                                #
#                                                                   #
# File: prime-update-collision-action.inc                           #
# Author: Eric D. Nielsen                                           #
# Description: An interstitial page that sets the proper            #
#              values based on the uses response to the close       #
#              match page. Adds the new person/affil if             #
#              needed. Dispatches to partners.php to start          #
#              Step Two.                                            #
# Internal Links: partners.php                                      #
# External Links: SlidingDoors information page (footer)            #
#                 Comp-in-a-box information page (footer)           #
#                 OpenImpetus information page (footer)             #
# Change Log: 6/12/01 -- created -- edn                             #
#             7/19/02 -- version 2 --edn                            #
#####################################################################

    session_start();
include "include_others.inc";
$contraCheck= new ContraCheck($db);
$firstName = $HTTP_SESSION_VARS["formVars"]["UpdateFirstName"];
$lastName = $HTTP_SESSION_VARS["formVars"]["UpdateLastName"];
$email = $HTTP_SESSION_VARS["formVars"]["UpdateEmail"];
$affil = $HTTP_SESSION_VARS["formVars"]["UpdateAffil"];
$newAffil = $HTTP_SESSION_VARS["formVars"]["UpdateNewAffil"];
$affilShort = $HTTP_SESSION_VARS["formVars"]["UpdateAffilShort"];
$age = $HTTP_SESSION_VARS["formVars"]["UpdateAge"];
$package = $HTTP_SESSION_VARS["formVars"]["UpdatePackage"];
$newPrimeID = $HTTP_POST_VARS["CHOICE"];
$primeID = $HTTP_SESSION_VARS["primeID"];
if ($newPrimeID != -1)
{
    session_unregister("conflicts");
    session_unregister("formVars");
    $query = "UPDATE couples SET follower=$newPrimeID WHERE follower=$primeID;";
    $db->query($query);
    $query = "UPDATE couples SET leader=$newPrimeID WHERE leader=$primeID;";
    $db->query($query);
    $query = "DELETE FROM people where peopleid=$primeID;";
    $db->query($query);

    $primeID=$newPrimeID;
    session_register("primeID");
    localRedirect("Location: $baseURL/partners.php");
    exit;
}
$person = new Person($db);
$person->setID($primeID);
$person->retrieve();
if ($person->getFirstName() != $firstName)
```

```
    $person->setFirstName($firstName);
if ($person->getLastName() != $lastName)
    $person->setLastName($lastName);
if ($person->getEmail() != $email)
    $person->setEmail($email);
if ($person->getAffilID() != $affil)
    $person->setAffilID($affil);
if ($person->getAgeLevel() != $age)
    $person->setAgeLevel($age);
if ($person->getPackage() != $package)
{
    $compRulesConflicts = $contraCheck->checkPersonPackage($primeID,"",$primeID,$package);
    if (!$compRulesConflicts->result)
    {
        # setup variables
        $formVars["UpdateEventsMsg"]=$compRulesConflicts->getErrorMessage();
        $formVars["UpdatePackage"]=$person->getPackage();
        session_register("primeID");
        session_register("formVars");
        localRedirect("Location: $baseURL/partners.php");
        exit;
    }

    ## need to insert the code here to check if this is allowed
    $person->setPackage($package);

}

$person->postToDB();
if ($primeID != $person->getID()) die ("ID changed on update");
if (!$primeID)
{
    errorPage($PHP_SELF, "root","Database Inaccessible",
        "utilities::addPerson unable to access database",TRUE);
    exit;
}
session_register("primeID");
localRedirect("Location: $baseURL/partners.php");
exit;
?>
```

prime-update-collision.php

```
<?php
#####################################################################
# This file is part of SlidingDoors                                 #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd        #
# SlidingDoors is availible for license under the GPL, see          #
# the COPYING file in the root directory of the install for         #
# the full terms of the GPL.                                        #
#                                                                   #
# File: prime-collision.php                                         #
# Author: Eric D. Nielsen                                           #
# Description: This page queries the use when a near match on       #
#              their entered personal information occurs, ie        #
#              they are trying to register than their partner       #
#              already did either with correct or incorrect         #
#              spellings.                                           #
# Internal Links: prime-collision-action.php                        #
# External Links: SlidingDoors information page (footer)            #
#                 Comp-in-a-box information page(footer)            #
#                 OpenImpetus information page  (footer)            #
# Change Log: 6/12/01 -- created -- edn                             #
#             7/19/02 -- GPL, display rework -- edn                 #
#####################################################################
# Do not remove the following line
include "include_others.inc";
session_start();
session_register("formVars");
$primeID=$HTTP_SESSION_VARS["primeID"];
$conflicts = $HTTP_SESSION_VARS["conflicts"];
$display = new HTMLDisplay($db);

$page = $display->beginPage();
$page .= $display->instructionBox(
"Registration: Name Clash on Update",
"Your entered information closely matches an entry already in the database.
It is possible one of your partner(s) already registered you, possibly
with a slightly misspelled name.  The information you entered appears as the
first option, the close matches with partner(s) names appear as options
below.  Selecting the top row updates your existing record to your newly
entered values.  Selecting a lower one, keeps the lower one's_data_but
merges_the_registration_of_your_current_entry_and_the_selected_lower_entry.");
$page_.=_$display->conflictResolutionBox($conflicts,
_____copyArray($HTTP_SESSION_VARS["formVars"],"","update"),
_____$primeID);
$page_.=_$display->sectionBreak();
$page_.=_$display->compinaboxBox();
$page_.=_$display->endPage();
clean_up_session();
session_register("formVars");
session_register("primeID");
echo_$page;
?>
```

prime.php

```
<?php
#####################################################################
# This file is part of SlidingDoors                                 #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd        #
# SlidingDoors is availible for license under the GPL, see          #
# the COPYING file in the root directory of the install for         #
# the full terms of the GPL.                                        #
#                                                                   #
# File: prime.php                                                   #
# Author: Eric D. Nielsen                                           #
# Description: This page allows the user to initially enter         #
#              their identification information. A different        #
#              page is used to modify this information later.       #
# Internal Links: prime-check.php                                   #
# External Links: SlidingDoors information page (footer)            #
#                 Comp-in-a-box information page(footer)            #
#                 OpenImpetus information page (footer)             #
# Change Log: 6/12/01 -- created -- edn                             #
#             3/14/02 -- change to use HTMLDisplay::boxes --edn     #
#             6/26/02 -- GPL, new person form -- edn                #
#####################################################################
# Do not remove the following line
include "include_others.inc";
session_start();
$display = new HTMLDisplay($db);
$formVars=$HTTP_SESSION_VARS["formVars"];
restrictNavigation(array(array("PAGE"=>"$baseURL/index.php",
                               "VARS"=>array()),
                         array("PAGE"=>"$baseURL/",
                               "VARS"=>array()),
                         array("PAGE"=>"$baseURL/done.php",
                               "VARS"=>array()),
                         array("PAGE"=>"$baseURL/prime-check.php",
                               "VARS"=>array("formVars")),
                         array("PAGE"=>"$baseURL/prime.php",
                               "VARS"=>array("formVars"))));

$preloadEmail = htmlspecialchars($HTTP_POST_VARS["propagateEnteredEmail"]);
if ($HTTP_POST_VARS["propagateAffil"]!="" && $formVars["AffilID"]=="" &&
        $formVars["NewAffil"]=="" && $formVars["ShortAffil"]=="" &&
        ereg("^[-0-9]*$",$HTTP_POST_VARS["propagateAffil"]))
    $formVars["AffilID"]=$HTTP_POST_VARS["propagateAffil"];

$propagateEnteredEmail=$preloadEmail;
session_register("propagateEnteredEmail");

$page = $display->beginPage();
$page .= $display->instructionBox("Registration Part 1: Personal Information",
"On this page you will enter the information used to identify yourself. You
will enter your partner's information on the next page; it doesn't matter
if you are a leader or follower. If you are registering an entire team or
studio, it will be slightly easier to enter people with multiple partners
 on this page.");

$page .= $display->sectionBreak();
#TODO: Fix error display
$box = $display->errorBox($HTTP_SESSION_VARS["formVars"]["GeneralMsg"] .
                          $HTTP_SESSION_VARS["formVars"]["FirstNameMsg"] .
                             $HTTP_SESSION_VARS["formVars"]["LastNameMsg"] .
```

```
                                $HTTP_SESSION_VARS["formVars"]["AgeMsg"] .
                                $HTTP_SESSION_VARS["formVars"]["PackageMsg"] .
                                $HTTP_SESSION_VARS["formVars"]["EmailMsg"] .
                                $HTTP_SESSION_VARS["formVars"]["AffilIDMsg"] .
                                $HTTP_SESSION_VARS["formVars"]["NewAffilMsg"]);
if ($box!="")
    $page .= $box . $display->sectionBreak();

# Begin Form Construction #
$form = "<form action=\"SCRIPTS/prime-check.php\" method=\"POST\">\n";
$form .= "<table cellspacing=\"0\" cellpadding=\"8\">\n";
$tab=1;
$form .= $display->personFormBlock(FALSE);
$form .= "\t<td colspan=\"5\" align=\"right\">\n";
$form .= "\t\t<input type=\"SUBMIT\" name=\"Next Page\" value=\"Next Page\"\n";
$form .= "\t\t\t style=\"$SD_button_style\">\n";
$form .= "\t</td>\n</tr></table>\n";
$form .= "</form>\n";
# End Form Construction
$page .= $display->textBox($form,"95%","This is the registration form");

$page .= $display->sectionBreak();
$page .= $display->compinaboxBox();
$page .= $display->endPage();
clean_up_session();
echo $page;
?>
```

C.3 PrepStep Generated Configuration Files

PREPARSED/compDB__rebuildEvents__eventsList.inc

```
<?php
#########################################################
# This file is generated by PrepStep and is required  #
# for proper operation of your SlidingDoor web site.  #
# Editing this file by hand will likely result in     #
# breaking your site.  You have been warned.          #
#                                                     #
# Generated By: parseEventsFile.php::eventsBuildFile  #
#########################################################

$includedEvents = array(
        array(2370,"N-A-F","Newcomer American Foxtrot"),
        array(2371,"N-A-Sw","Newcomer American Swing"),
        array(2372,"N-I-W","Newcomer International Waltz"),
        array(2373,"N-I-R","Newcomer International Rumba"),
        array(2374,"B-A-W","Beginner American Waltz"),
        array(2375,"B-A-T","Beginner American Tango"),
        array(2376,"B-A-F","Beginner American Foxtrot"),
        array(2377,"B-A-Vw","Beginner American Viennese Waltz"),
        array(2378,"I-A-W","Intermediate American Waltz"),
        array(2379,"I-A-T","Intermediate American Tango"),
        array(2380,"I-A-F","Intermediate American Foxtrot"),
        array(2381,"I-A-Vw","Intermediate American Viennese Waltz"),
        array(2382,"A-A-W","Advanced American Waltz"),
        array(2383,"A-A-T","Advanced American Tango"),
        array(2384,"A-A-F","Advanced American Foxtrot"),
        array(2385,"A-A-Vw","Advanced American Viennese Waltz"),
        array(2386,"O-A-W","Open American Waltz"),
        array(2387,"O-A-T","Open American Tango"),
        array(2388,"O-A-F","Open American Foxtrot"),
        array(2389,"O-A-Vw","Open American Viennese Waltz"),
        array(2390,"B-I-W","Beginner International Waltz"),
        array(2391,"B-I-T","Beginner International Tango"),
        array(2392,"B-I-Vw","Beginner International Viennese Waltz"),
        array(2393,"B-I-F","Beginner International Foxtrot"),
        array(2394,"B-I-Q","Beginner International Quickstep"),
        array(2395,"I-I-W","Intermediate International Waltz"),
        array(2396,"I-I-T","Intermediate International Tango"),
        array(2397,"I-I-Vw","Intermediate International Viennese Waltz"),
        array(2398,"I-I-F","Intermediate International Foxtrot"),
        array(2399,"I-I-Q","Intermediate International Quickstep"),
        array(2400,"A-I-W","Advanced International Waltz"),
        array(2401,"A-I-T","Advanced International Tango"),
        array(2402,"A-I-Vw","Advanced International Viennese Waltz"),
        array(2403,"A-I-F","Advanced International Foxtrot"),
        array(2404,"A-I-Q","Advanced International Quickstep"),
        array(2405,"O-I-W","Open International Waltz"),
        array(2406,"O-I-T","Open International Tango"),
        array(2407,"O-I-Vw","Open International Viennese Waltz"),
        array(2408,"O-I-F","Open International Foxtrot"),
        array(2409,"O-I-Q","Open International Quickstep"),
        array(2410,"B-I-C","Beginner International Cha Cha"),
        array(2411,"B-I-Sa","Beginner International Samba"),
        array(2412,"B-I-R","Beginner International Rumba"),
        array(2413,"B-I-P","Beginner International Paso Doble"),
```

```
        array(2414,"B-I-J","Beginner International Jive"),
        array(2415,"I-I-C","Intermediate International Cha Cha"),
        array(2416,"I-I-Sa","Intermediate International Samba"),
        array(2417,"I-I-R","Intermediate International Rumba"),
        array(2418,"I-I-P","Intermediate International Paso Doble"),
        array(2419,"I-I-J","Intermediate International Jive"),
        array(2420,"A-I-C","Advanced International Cha Cha"),
        array(2421,"A-I-Sa","Advanced International Samba"),
        array(2422,"A-I-R","Advanced International Rumba"),
        array(2423,"A-I-P","Advanced International Paso Doble"),
        array(2424,"A-I-J","Advanced International Jive"),
        array(2425,"O-I-C","Open International Cha Cha"),
        array(2426,"O-I-Sa","Open International Samba"),
        array(2427,"O-I-R","Open International Rumba"),
        array(2428,"O-I-P","Open International Paso Doble"),
        array(2429,"O-I-J","Open International Jive"),
        array(2430,"B-A-C","Beginner American Cha Cha"),
        array(2431,"B-A-R","Beginner American Rumba"),
        array(2432,"B-A-Sw","Beginner American Swing"),
        array(2433,"B-A-M","Beginner American Mambo"),
        array(2434,"B-A-B","Beginner American Bolero"),
        array(2435,"I-A-C","Intermediate American Cha Cha"),
        array(2436,"I-A-R","Intermediate American Rumba"),
        array(2437,"I-A-Sw","Intermediate American Swing"),
        array(2438,"I-A-M","Intermediate American Mambo"),
        array(2439,"I-A-B","Intermediate American Bolero"),
        array(2440,"A-A-C","Advanced American Cha Cha"),
        array(2441,"A-A-R","Advanced American Rumba"),
        array(2442,"A-A-Sw","Advanced American Swing"),
        array(2443,"A-A-M","Advanced American Mambo"),
        array(2444,"A-A-B","Advanced American Bolero"),
        array(2445,"O-A-C","Open American Cha Cha"),
        array(2446,"O-A-R","Open American Rumba"),
        array(2447,"O-A-Sw","Open American Swing"),
        array(2448,"O-A-M","Open American Mambo"),
        array(2449,"O-A-B","Open American Bolero"));
########################################################
# THIS FILE IS COMPUTER GENERATED CODE DO NOT EDIT BY  #
# HAND.  See top comment for information on what       #
# function produces it and what function includes it.  #
########################################################
?>
```

PREPARSED/compDB__rebuildEvents__eventsSummary.inc

```
<?php
##################################################
# This file is generated by PrepStep and is required  #
# for proper operation of your SlidingDoor web site.  #
# Editing this file by hand will likely result in     #
# breaking your site.  You have been warned.          #
#                                                     #
# Generated By: parseEventsFile.php::eventsStatsFile  #
##################################################

$includedTemp = "<a href=\"#Newcomer\">Newcomer</a>
(<a href=\"#Newcomer-Standard\">Standard</a>,
<a href=\"#Newcomer-Smooth\">Smooth</a>,
<a href=\"#Newcomer-Latin\">Latin</a>,
<a href=\"#Newcomer-Rhythm\">Rhythm</a>) |
<a href=\"#Beginner\">Beginner</a>
(<a href=\"#Beginner-Standard\">Standard</a>,
<a href=\"#Beginner-Smooth\">Smooth</a>,
<a href=\"#Beginner-Latin\">Latin</a>,
<a href=\"#Beginner-Rhythm\">Rhythm</a>) |
<a href=\"#Intermediate\">Intermediate</a>
(<a href=\"#Intermediate-Standard\">Standard</a>,
<a href=\"#Intermediate-Smooth\">Smooth</a>,
<a href=\"#Intermediate-Latin\">Latin</a>,
<a href=\"#Intermediate-Rhythm\">Rhythm</a>) |
<a href=\"#Advanced\">Advanced</a>
(<a href=\"#Advanced-Standard\">Standard</a>,
<a href=\"#Advanced-Smooth\">Smooth</a>,
<a href=\"#Advanced-Latin\">Latin</a>,
<a href=\"#Advanced-Rhythm\">Rhythm</a>) |
<a href=\"#Open\">Open</a>
(<a href=\"#Open-Standard\">Standard</a>,
<a href=\"#Open-Smooth\">Smooth</a>,
<a href=\"#Open-Latin\">Latin</a>,
<a href=\"#Open-Rhythm\">Rhythm</a>)<br />";
$includedTemp .= "<a name=\"Newcomer\"><h3>Newcomer</h3></a>\n";
$includedTemp .= "<a name=\"Newcomer-Standard\"><h4>Standard</h4></a>\n";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2372;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/N-I-W.html\">";
$includedTemp .= "Newcomer International Waltz</a>: $num<br>";
$includedTemp .= "<a name=\"Newcomer-Smooth\"><h4>Smooth</h4></a>\n";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2370;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/N-A-F.html\">";
$includedTemp .= "Newcomer American Foxtrot</a>: $num<br>";
$includedTemp .= "<a name=\"Newcomer-Latin\"><h4>Latin</h4></a>\n";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2373;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/N-I-R.html\">";
$includedTemp .= "Newcomer International Rumba</a>: $num<br>";
$includedTemp .= "<a name=\"Newcomer-Rhythm\"><h4>Rhythm</h4></a>\n";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2371;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/N-A-Sw.html\">";
```

```
$includedTemp .= "Newcomer American Swing</a>: $num<br>";
$includedTemp .= "<a name=\"Beginner\"><h3>Beginner</h3></a>\n";
$includedTemp .= "<a name=\"Beginner-Standard\"><h4>Standard</h4></a>\n";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2390;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/B-I-W.html\">";
$includedTemp .= "Beginner International Waltz</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2391;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/B-I-T.html\">";
$includedTemp .= "Beginner International Tango</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2392;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/B-I-Vw.html\">";
$includedTemp .= "Beginner International Viennese Waltz</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2393;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/B-I-F.html\">";
$includedTemp .= "Beginner International Foxtrot</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2394;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/B-I-Q.html\">";
$includedTemp .= "Beginner International Quickstep</a>: $num<br>";
$includedTemp .= "<a name=\"Beginner-Smooth\"><h4>Smooth</h4></a>\n";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2374;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/B-A-W.html\">";
$includedTemp .= "Beginner American Waltz</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2375;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/B-A-T.html\">";
$includedTemp .= "Beginner American Tango</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2376;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/B-A-F.html\">";
$includedTemp .= "Beginner American Foxtrot</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2377;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/B-A-Vw.html\">";
$includedTemp .= "Beginner American Viennese Waltz</a>: $num<br>";
$includedTemp .= "<a name=\"Beginner-Latin\"><h4>Latin</h4></a>\n";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2410;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/B-I-C.html\">";
$includedTemp .= "Beginner International Cha Cha</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2411;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/B-I-Sa.html\">";
$includedTemp .= "Beginner International Samba</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2412;";
```

```
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/B-I-R.html\">";
$includedTemp .= "Beginner International Rumba</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2413;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/B-I-P.html\">";
$includedTemp .= "Beginner International Paso Doble</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2414;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/B-I-J.html\">";
$includedTemp .= "Beginner International Jive</a>: $num<br>";
$includedTemp .= "<a name=\"Beginner-Rhythm\"><h4>Rhythm</h4></a>\n";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2430;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/B-A-C.html\">";
$includedTemp .= "Beginner American Cha Cha</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2431;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/B-A-R.html\">";
$includedTemp .= "Beginner American Rumba</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2432;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/B-A-Sw.html\">";
$includedTemp .= "Beginner American Swing</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2433;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/B-A-M.html\">";
$includedTemp .= "Beginner American Mambo</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2434;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/B-A-B.html\">";
$includedTemp .= "Beginner American Bolero</a>: $num<br>";
$includedTemp .= "<a name=\"Intermediate\"><h3>Intermediate</h3></a>\n";
$includedTemp .= "<a name=\"Intermediate-Standard\"><h4>Standard</h4></a>\n";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2395;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/I-I-W.html\">";
$includedTemp .= "Intermediate International Waltz</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2396;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/I-I-T.html\">";
$includedTemp .= "Intermediate International Tango</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2397;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/I-I-Vw.html\">";
$includedTemp .= "Intermediate International Viennese Waltz</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2398;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/I-I-F.html\">";
```

```
$includedTemp .= "Intermediate International Foxtrot</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2399;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/I-I-Q.html\">";
$includedTemp .= "Intermediate International Quickstep</a>: $num<br>";
$includedTemp .= "<a name=\"Intermediate-Smooth\"><h4>Smooth</h4></a>\n";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2378;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/I-A-W.html\">";
$includedTemp .= "Intermediate American Waltz</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2379;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/I-A-T.html\">";
$includedTemp .= "Intermediate American Tango</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2380;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/I-A-F.html\">";
$includedTemp .= "Intermediate American Foxtrot</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2381;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/I-A-Vw.html\">";
$includedTemp .= "Intermediate American Viennese Waltz</a>: $num<br>";
$includedTemp .= "<a name=\"Intermediate-Latin\"><h4>Latin</h4></a>\n";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2415;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/I-I-C.html\">";
$includedTemp .= "Intermediate International Cha Cha</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2416;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/I-I-Sa.html\">";
$includedTemp .= "Intermediate International Samba</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2417;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/I-I-R.html\">";
$includedTemp .= "Intermediate International Rumba</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2418;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/I-I-P.html\">";
$includedTemp .= "Intermediate International Paso Doble</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2419;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/I-I-J.html\">";
$includedTemp .= "Intermediate International Jive</a>: $num<br>";
$includedTemp .= "<a name=\"Intermediate-Rhythm\"><h4>Rhythm</h4></a>\n";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2435;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/I-A-C.html\">";
$includedTemp .= "Intermediate American Cha Cha</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2436;";
$result=$this->query($query);
```

```
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/I-A-R.html\">";
$includedTemp .= "Intermediate American Rumba</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2437;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/I-A-Sw.html\">";
$includedTemp .= "Intermediate American Swing</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2438;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/I-A-M.html\">";
$includedTemp .= "Intermediate American Mambo</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2439;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/I-A-B.html\">";
$includedTemp .= "Intermediate American Bolero</a>: $num<br>";
$includedTemp .= "<a name=\"Advanced\"><h3>Advanced</h3></a>\n";
$includedTemp .= "<a name=\"Advanced-Standard\"><h4>Standard</h4></a>\n";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2400;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/A-I-W.html\">";
$includedTemp .= "Advanced International Waltz</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2401;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/A-I-T.html\">";
$includedTemp .= "Advanced International Tango</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2402;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/A-I-Vw.html\">";
$includedTemp .= "Advanced International Viennese Waltz</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2403;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/A-I-F.html\">";
$includedTemp .= "Advanced International Foxtrot</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2404;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/A-I-Q.html\">";
$includedTemp .= "Advanced International Quickstep</a>: $num<br>";
$includedTemp .= "<a name=\"Advanced-Smooth\"><h4>Smooth</h4></a>\n";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2382;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/A-A-W.html\">";
$includedTemp .= "Advanced American Waltz</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2383;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/A-A-T.html\">";
$includedTemp .= "Advanced American Tango</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2384;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/A-A-F.html\">";
$includedTemp .= "Advanced American Foxtrot</a>: $num<br>";
```

```
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2385;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/A-A-Vw.html\">";
$includedTemp .= "Advanced American Viennese Waltz</a>: $num<br>";
$includedTemp .= "<a name=\"Advanced-Latin\"><h4>Latin</h4></a>\n";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2420;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/A-I-C.html\">";
$includedTemp .= "Advanced International Cha Cha</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2421;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/A-I-Sa.html\">";
$includedTemp .= "Advanced International Samba</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2422;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/A-I-R.html\">";
$includedTemp .= "Advanced International Rumba</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2423;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/A-I-P.html\">";
$includedTemp .= "Advanced International Paso Doble</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2424;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/A-I-J.html\">";
$includedTemp .= "Advanced International Jive</a>: $num<br>";
$includedTemp .= "<a name=\"Advanced-Rhythm\"><h4>Rhythm</h4></a>\n";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2440;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/A-A-C.html\">";
$includedTemp .= "Advanced American Cha Cha</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2441;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/A-A-R.html\">";
$includedTemp .= "Advanced American Rumba</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2442;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/A-A-Sw.html\">";
$includedTemp .= "Advanced American Swing</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2443;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/A-A-M.html\">";
$includedTemp .= "Advanced American Mambo</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2444;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/A-A-B.html\">";
$includedTemp .= "Advanced American Bolero</a>: $num<br>";
$includedTemp .= "<a name=\"Open\"><h3>Open</h3></a>\n";
$includedTemp .= "<a name=\"Open-Standard\"><h4>Standard</h4></a>\n";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2405;";
$result=$this->query($query);
```

```
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/O-I-W.html\">";
$includedTemp .= "Open International Waltz</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2406;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/O-I-T.html\">";
$includedTemp .= "Open International Tango</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2407;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/O-I-Vw.html\">";
$includedTemp .= "Open International Viennese Waltz</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2408;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/O-I-F.html\">";
$includedTemp .= "Open International Foxtrot</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2409;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/O-I-Q.html\">";
$includedTemp .= "Open International Quickstep</a>: $num<br>";
$includedTemp .= "<a name=\"Open-Smooth\"><h4>Smooth</h4></a>\n";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2386;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/O-A-W.html\">";
$includedTemp .= "Open American Waltz</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2387;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/O-A-T.html\">";
$includedTemp .= "Open American Tango</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2388;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/O-A-F.html\">";
$includedTemp .= "Open American Foxtrot</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2389;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/O-A-Vw.html\">";
$includedTemp .= "Open American Viennese Waltz</a>: $num<br>";
$includedTemp .= "<a name=\"Open-Latin\"><h4>Latin</h4></a>\n";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2425;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/O-I-C.html\">";
$includedTemp .= "Open International Cha Cha</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2426;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/O-I-Sa.html\">";
$includedTemp .= "Open International Samba</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2427;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/O-I-R.html\">";
$includedTemp .= "Open International Rumba</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2428;";
```

```
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/O-I-P.html\">";
$includedTemp .= "Open International Paso Doble</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2429;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/O-I-J.html\">";
$includedTemp .= "Open International Jive</a>: $num<br>";
$includedTemp .= "<a name=\"Open-Rhythm\"><h4>Rhythm</h4></a>\n";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2445;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/O-A-C.html\">";
$includedTemp .= "Open American Cha Cha</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2446;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/O-A-R.html\">";
$includedTemp .= "Open American Rumba</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2447;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/O-A-Sw.html\">";
$includedTemp .= "Open American Swing</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2448;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/O-A-M.html\">";
$includedTemp .= "Open American Mambo</a>: $num<br>";
$query = "SELECT COUNT(*) FROM events_registration WHERE eventid=2449;";
$result=$this->query($query);
list($num) = $result->getRowAt(0);
$includedTemp .= "<a href=\"$CIB_BASE_URL/stats/$unixname/Events/O-A-B.html\">";
$includedTemp .= "Open American Bolero</a>: $num<br>";
############################################################
# THIS FILE IS COMPUTER GENERATED CODE DO NOT EDIT BY      #
# HAND.  See top comment for information on what           #
# function produces it and what function includes it.      #
############################################################
?>
```

PREPARSED/couple__eventBeforeDance.inc

```
<?php
#######################################################
# This file is generated by PrepStep and is required  #
# for proper operation of your SlidingDoor web site.  #
# Editing this file by hand will likely result in     #
# breaking your site.  You have been warned.          #
#                                                     #
# Generated By:                                       #
#    parseEventsFile.php::couple__eventBefore         #
#                                                     #
# This code is directly embedded inside the function  #
# Couple::eventBefore and has access to its variables:#
# event1Lvl,event1Style,event1Dance                   #
#######################################################

switch($event2Style)
{
case "A" : switch ($event2Dance)
        {
                case "W": if (in_array($event1Dance,
        array("T", "F", "Vw", "C", "R", "Sw", "M", "B"))) return TRUE; break;
                case "T": if (in_array($event1Dance,
        array("F", "Vw", "C", "R", "Sw", "M", "B"))) return TRUE; break;
                case "F": if (in_array($event1Dance,
        array("Vw", "C", "R", "Sw", "M", "B"))) return TRUE; break;
                case "Vw": if (in_array($event1Dance,
        array("C", "R", "Sw", "M", "B"))) return TRUE; break;
                case "C": if (in_array($event1Dance,
        array("R", "Sw", "M", "B"))) return TRUE; break;
                case "R": if (in_array($event1Dance,
        array("Sw", "M", "B"))) return TRUE; break;
                case "Sw": if (in_array($event1Dance,
        array("M", "B"))) return TRUE; break;
                case "M": if (in_array($event1Dance,
        array("B"))) return TRUE; break;
                default: break;
        } break;
case "I" : switch ($event2Dance)
        {
                case "W": if (in_array($event1Dance,
        array("T", "Vw", "F", "Q", "C", "Sa", "R", "P", "J"))) return TRUE; break;
                case "T": if (in_array($event1Dance,
        array("Vw", "F", "Q", "C", "Sa", "R", "P", "J"))) return TRUE; break;
                case "Vw": if (in_array($event1Dance,
        array("F", "Q", "C", "Sa", "R", "P", "J"))) return TRUE; break;
                case "F": if (in_array($event1Dance,
        array("Q", "C", "Sa", "R", "P", "J"))) return TRUE; break;
                case "Q": if (in_array($event1Dance,
        array("C", "Sa", "R", "P", "J"))) return TRUE; break;
                case "C": if (in_array($event1Dance,
        array("Sa", "R", "P", "J"))) return TRUE; break;
                case "Sa": if (in_array($event1Dance,
        array("R", "P", "J"))) return TRUE; break;
                case "R": if (in_array($event1Dance,
        array("P", "J"))) return TRUE; break;
                case "P": if (in_array($event1Dance,
        array("J"))) return TRUE; break;
                default: break;
        } break;
```

```
default:
        break;
}
return TRUE;
#####################################################
# THIS FILE IS COMPUTER GENERATED CODE DO NOT EDIT BY #
# HAND.  See top comment for information on what      #
# function produces it and what function includes it. #
#####################################################
?>
```

PREPARSED/couple__eventBeforeLevel.inc

```
<?php
#########################################################
# This file is generated by PrepStep and is required  #
# for proper operation of your SlidingDoor web site.  #
# Editing this file by hand will likely result in     #
# breaking your site.  You have been warned.          #
#                                                     #
# Generated By:                                       #
#    parseEventsFile.php::couple__eventBefore         #
#                                                     #
# This code is directly embedded inside the function  #
# Couple::eventBefore and has access to its variables:#
# event1Lvl,event1Style,event1Dance                   #
#########################################################

switch($event2Lvl)
{
case "N": if (in_array($event1Lvl,array("B", "I", "A", "O"))) return TRUE; break;
case "B": if (in_array($event1Lvl,array("I", "A", "O"))) return TRUE; break;
case "I": if (in_array($event1Lvl,array("A", "O"))) return TRUE; break;
case "A": if (in_array($event1Lvl,array("O"))) return TRUE; break;
        default:
                break;
}
return FALSE;
#########################################################
# THIS FILE IS COMPUTER GENERATED CODE DO NOT EDIT BY #
# HAND.  See top comment for information on what      #
# function produces it and what function includes it. #
#########################################################
?>
```

PREPARSED/couple__eventBeforeStyle.inc

```
<?php
#########################################################
# This file is generated by PrepStep and is required  #
# for proper operation of your SlidingDoor web site.  #
# Editing this file by hand will likely result in     #
# breaking your site.  You have been warned.          #
#                                                     #
# Generated By:                                       #
#    parseEventsFile.php::couple__eventBefore         #
#                                                     #
# This code is directly embedded inside the function  #
# Couple::eventBefore and has access to its variables:#
# event1Lvl, event1Style, event1Dance                 #
#########################################################

switch($event2Style)
{
case "I": if (in_array($event1Style, array("A"))) return TRUE; break;
        default:
                break;
}
return FALSE;
#########################################################
# THIS FILE IS COMPUTER GENERATED CODE DO NOT EDIT BY #
# HAND.  See top comment for information on what      #
# function produces it and what function includes it. #
#########################################################
?>
```

PREPARSED/couple__getDances.inc

```
<?php
#########################################################
# This file is generated by PrepStep and is required  #
# for proper operation of your SlidingDoor web site.  #
# Editing this file by hand will likely result in     #
# breaking your site.  You have been warned.          #
#                                                     #
# Generated By:                                       #
#    parseEventsFile.php::couple__getDances           #
#                                                     #
# This code is directly embedded inside the function  #
# Couple::getDances and has access to its variables:  #
# this, events                                        #
#########################################################

switch($style)
{
case "Smooth" : $result = $this->styleFilter($events,"A",
        array("W", "T", "F", "Vw"));
break;
case "Standard" : $result = $this->styleFilter($events,"I",
        array("W", "T", "Vw", "F", "Q"));
break;
case "Latin" : $result = $this->styleFilter($events,"I",
        array("C", "Sa", "R", "P", "J"));
break;
case "Rhythm" : $result = $this->styleFilter($events,"A",
        array("C", "R", "Sw", "M", "B"));
break;
case "American" : $result = $this->styleFilter($events,"A",
        array("W", "T", "F", "Vw", "C", "R", "Sw", "M", "B"));
break;
case "International" : $result = $this->styleFilter($events,"I",
        array("W", "T", "Vw", "F", "Q", "C", "Sa", "R", "P", "J"));
break;
        default:
                $result = "";
}
#########################################################
# THIS FILE IS COMPUTER GENERATED CODE DO NOT EDIT BY #
# HAND.  See top comment for information on what      #
# function produces it and what function includes it. #
#########################################################
?>
```

PREPARSED/couple__setEventsLut.inc

```
<?php
#######################################################
# This file is generated by PrepStep and is required  #
# for proper operation of your SlidingDoor web site.  #
# Editing this file by hand will likely result in     #
# breaking your site.  You have been warned.          #
#                                                     #
# Generated By:                                       #
#    parseEventsFile.php::couple__setDances           #
#                                                     #
# This code is directly embedded inside the function  #
# Couple::setEvents and has access to its variables:  #
# temp,events                                         #
#######################################################

$eventLUT=array();
$eventLUT["N-A-F"]=2370;
$eventLUT["N-A-Sw"]=2371;
$eventLUT["N-I-W"]=2372;
$eventLUT["N-I-R"]=2373;
$eventLUT["B-A-W"]=2374;
$eventLUT["B-A-T"]=2375;
$eventLUT["B-A-F"]=2376;
$eventLUT["B-A-Vw"]=2377;
$eventLUT["I-A-W"]=2378;
$eventLUT["I-A-T"]=2379;
$eventLUT["I-A-F"]=2380;
$eventLUT["I-A-Vw"]=2381;
$eventLUT["A-A-W"]=2382;
$eventLUT["A-A-T"]=2383;
$eventLUT["A-A-F"]=2384;
$eventLUT["A-A-Vw"]=2385;
$eventLUT["O-A-W"]=2386;
$eventLUT["O-A-T"]=2387;
$eventLUT["O-A-F"]=2388;
$eventLUT["O-A-Vw"]=2389;
$eventLUT["B-I-W"]=2390;
$eventLUT["B-I-T"]=2391;
$eventLUT["B-I-Vw"]=2392;
$eventLUT["B-I-F"]=2393;
$eventLUT["B-I-Q"]=2394;
$eventLUT["I-I-W"]=2395;
$eventLUT["I-I-T"]=2396;
$eventLUT["I-I-Vw"]=2397;
$eventLUT["I-I-F"]=2398;
$eventLUT["I-I-Q"]=2399;
$eventLUT["A-I-W"]=2400;
$eventLUT["A-I-T"]=2401;
$eventLUT["A-I-Vw"]=2402;
$eventLUT["A-I-F"]=2403;
$eventLUT["A-I-Q"]=2404;
$eventLUT["O-I-W"]=2405;
$eventLUT["O-I-T"]=2406;
$eventLUT["O-I-Vw"]=2407;
$eventLUT["O-I-F"]=2408;
$eventLUT["O-I-Q"]=2409;
$eventLUT["B-I-C"]=2410;
$eventLUT["B-I-Sa"]=2411;
$eventLUT["B-I-R"]=2412;
```

```
$eventLUT["B-I-P"]=2413;
$eventLUT["B-I-J"]=2414;
$eventLUT["I-I-C"]=2415;
$eventLUT["I-I-Sa"]=2416;
$eventLUT["I-I-R"]=2417;
$eventLUT["I-I-P"]=2418;
$eventLUT["I-I-J"]=2419;
$eventLUT["A-I-C"]=2420;
$eventLUT["A-I-Sa"]=2421;
$eventLUT["A-I-R"]=2422;
$eventLUT["A-I-P"]=2423;
$eventLUT["A-I-J"]=2424;
$eventLUT["O-I-C"]=2425;
$eventLUT["O-I-Sa"]=2426;
$eventLUT["O-I-R"]=2427;
$eventLUT["O-I-P"]=2428;
$eventLUT["O-I-J"]=2429;
$eventLUT["B-A-C"]=2430;
$eventLUT["B-A-R"]=2431;
$eventLUT["B-A-Sw"]=2432;
$eventLUT["B-A-M"]=2433;
$eventLUT["B-A-B"]=2434;
$eventLUT["I-A-C"]=2435;
$eventLUT["I-A-R"]=2436;
$eventLUT["I-A-Sw"]=2437;
$eventLUT["I-A-M"]=2438;
$eventLUT["I-A-B"]=2439;
$eventLUT["A-A-C"]=2440;
$eventLUT["A-A-R"]=2441;
$eventLUT["A-A-Sw"]=2442;
$eventLUT["A-A-M"]=2443;
$eventLUT["A-A-B"]=2444;
$eventLUT["O-A-C"]=2445;
$eventLUT["O-A-R"]=2446;
$eventLUT["O-A-Sw"]=2447;
$eventLUT["O-A-M"]=2448;
$eventLUT["O-A-B"]=2449;
#######################################################
# THIS FILE IS COMPUTER GENERATED CODE DO NOT EDIT BY #
# HAND.  See top comment for information on what      #
# function produces it and what function includes it. #
#######################################################
?>
```

PREPARSED/couple__setEventsSet.inc

```
<?php
#######################################################
# This file is generated by PrepStep and is required  #
# for proper operation of your SlidingDoor web site.  #
# Editing this file by hand will likely result in     #
# breaking your site.  You have been warned.          #
#                                                     #
# Generated By:                                       #
#    parseEventsFile.php::couple__setDances           #
#                                                     #
# This code is directly embedded inside the function  #
# Couple::setEvents and has access to its variables:  #
# temp, events                                        #
#######################################################

switch($events[$i])
{
case "Newcomer-ALL" : $fallthrough=TRUE;
case "Newcomer-Standard-ALL" :array_push($temp,$eventLUT["N-I-W"]);
        if (!$fallthrough) break;
case "Newcomer-Smooth-ALL" :array_push($temp,$eventLUT["N-A-F"]);
        if (!$fallthrough) break;
case "Newcomer-Latin-ALL" :array_push($temp,$eventLUT["N-I-R"]);
        if (!$fallthrough) break;
case "Newcomer-Rhythm-ALL" :array_push($temp,$eventLUT["N-A-Sw"]);
        if (!$fallthrough) break;
case "Beginner-ALL" : $fallthrough=TRUE;
case "Beginner-Standard-ALL" :array_push($temp,$eventLUT["B-I-W"]);
        array_push($temp,$eventLUT["B-I-T"]);
        array_push($temp,$eventLUT["B-I-Vw"]);
        array_push($temp,$eventLUT["B-I-F"]);
        array_push($temp,$eventLUT["B-I-Q"]);
        if (!$fallthrough) break;
case "Beginner-Smooth-ALL" :array_push($temp,$eventLUT["B-A-W"]);
        array_push($temp,$eventLUT["B-A-T"]);
        array_push($temp,$eventLUT["B-A-F"]);
        array_push($temp,$eventLUT["B-A-Vw"]);
        if (!$fallthrough) break;
case "Beginner-Latin-ALL" :array_push($temp,$eventLUT["B-I-C"]);
        array_push($temp,$eventLUT["B-I-Sa"]);
        array_push($temp,$eventLUT["B-I-R"]);
        array_push($temp,$eventLUT["B-I-P"]);
        array_push($temp,$eventLUT["B-I-J"]);
        if (!$fallthrough) break;
case "Beginner-Rhythm-ALL" :array_push($temp,$eventLUT["B-A-C"]);
        array_push($temp,$eventLUT["B-A-R"]);
        array_push($temp,$eventLUT["B-A-Sw"]);
        array_push($temp,$eventLUT["B-A-M"]);
        array_push($temp,$eventLUT["B-A-B"]);
        if (!$fallthrough) break;
case "Intermediate-ALL" : $fallthrough=TRUE;
case "Intermediate-Standard-ALL" :array_push($temp,$eventLUT["I-I-W"]);
        array_push($temp,$eventLUT["I-I-T"]);
        array_push($temp,$eventLUT["I-I-Vw"]);
        array_push($temp,$eventLUT["I-I-F"]);
        array_push($temp,$eventLUT["I-I-Q"]);
        if (!$fallthrough) break;
case "Intermediate-Smooth-ALL" :array_push($temp,$eventLUT["I-A-W"]);
        array_push($temp,$eventLUT["I-A-T"]);
```

```
        array_push($temp,$eventLUT["I-A-F"]);
        array_push($temp,$eventLUT["I-A-Vw"]);
        if (!$fallthrough) break;
case "Intermediate-Latin-ALL" :array_push($temp,$eventLUT["I-I-C"]);
        array_push($temp,$eventLUT["I-I-Sa"]);
        array_push($temp,$eventLUT["I-I-R"]);
        array_push($temp,$eventLUT["I-I-P"]);
        array_push($temp,$eventLUT["I-I-J"]);
        if (!$fallthrough) break;
case "Intermediate-Rhythm-ALL" :array_push($temp,$eventLUT["I-A-C"]);
        array_push($temp,$eventLUT["I-A-R"]);
        array_push($temp,$eventLUT["I-A-Sw"]);
        array_push($temp,$eventLUT["I-A-M"]);
        array_push($temp,$eventLUT["I-A-B"]);
        if (!$fallthrough) break;
case "Advanced-ALL" : $fallthrough=TRUE;
case "Advanced-Standard-ALL" :array_push($temp,$eventLUT["A-I-W"]);
        array_push($temp,$eventLUT["A-I-T"]);
        array_push($temp,$eventLUT["A-I-Vw"]);
        array_push($temp,$eventLUT["A-I-F"]);
        array_push($temp,$eventLUT["A-I-Q"]);
        if (!$fallthrough) break;
case "Advanced-Smooth-ALL" :array_push($temp,$eventLUT["A-A-W"]);
        array_push($temp,$eventLUT["A-A-T"]);
        array_push($temp,$eventLUT["A-A-F"]);
        array_push($temp,$eventLUT["A-A-Vw"]);
        if (!$fallthrough) break;
case "Advanced-Latin-ALL" :array_push($temp,$eventLUT["A-I-C"]);
        array_push($temp,$eventLUT["A-I-Sa"]);
        array_push($temp,$eventLUT["A-I-R"]);
        array_push($temp,$eventLUT["A-I-P"]);
        array_push($temp,$eventLUT["A-I-J"]);
        if (!$fallthrough) break;
case "Advanced-Rhythm-ALL" :array_push($temp,$eventLUT["A-A-C"]);
        array_push($temp,$eventLUT["A-A-R"]);
        array_push($temp,$eventLUT["A-A-Sw"]);
        array_push($temp,$eventLUT["A-A-M"]);
        array_push($temp,$eventLUT["A-A-B"]);
        if (!$fallthrough) break;
case "Open-ALL" : $fallthrough=TRUE;
case "Open-Standard-ALL" :array_push($temp,$eventLUT["O-I-W"]);
        array_push($temp,$eventLUT["O-I-T"]);
        array_push($temp,$eventLUT["O-I-Vw"]);
        array_push($temp,$eventLUT["O-I-F"]);
        array_push($temp,$eventLUT["O-I-Q"]);
        if (!$fallthrough) break;
case "Open-Smooth-ALL" :array_push($temp,$eventLUT["O-A-W"]);
        array_push($temp,$eventLUT["O-A-T"]);
        array_push($temp,$eventLUT["O-A-F"]);
        array_push($temp,$eventLUT["O-A-Vw"]);
        if (!$fallthrough) break;
case "Open-Latin-ALL" :array_push($temp,$eventLUT["O-I-C"]);
        array_push($temp,$eventLUT["O-I-Sa"]);
        array_push($temp,$eventLUT["O-I-R"]);
        array_push($temp,$eventLUT["O-I-P"]);
        array_push($temp,$eventLUT["O-I-J"]);
        if (!$fallthrough) break;
case "Open-Rhythm-ALL" :array_push($temp,$eventLUT["O-A-C"]);
        array_push($temp,$eventLUT["O-A-R"]);
        array_push($temp,$eventLUT["O-A-Sw"]);
        array_push($temp,$eventLUT["O-A-M"]);
```

```
        array_push($temp,$eventLUT["O-A-B"]);
        if (!$fallthrough) break;
default: array_push($temp,$eventLUT[$events[$i]]);
}
#########################################################
# THIS FILE IS COMPUTER GENERATED CODE DO NOT EDIT BY   #
# HAND.  See top comment for information on what        #
# function produces it and what function includes it.   #
#########################################################
?>
```

PREPARSED/htmlDisplay__colorDances.inc

```
<?php
#########################################################
# This file is generated by PrepStep and is required  #
# for proper operation of your SlidingDoor web site.  #
# Editing this file by hand will likely result in     #
# breaking your site.  You have been warned.          #
#                                                     #
# Generated By:  parseEventsFile.php::                #
#          htmlDisplay__colorIncludes                 #
#                                                     #
# This code is directly embedded inside the function  #
# HTMLDisplay::colorDances and has access to it s     #
# variables:  color                                   #
#########################################################

switch($level)
{
        case "N" : $color = "BLUE"; break;
        case "B" : $color = "GREEN"; break;
        case "I" : $color = "BLACK"; break;
        case "A" : $color = "ORANGE"; break;
        case "O" : $color = "RED"; break;
        default:
                $color="BLACK"; break;
}
#########################################################
# THIS FILE IS COMPUTER GENERATED CODE DO NOT EDIT BY #
# HAND.  See top comment for information on what      #
# function produces it and what function includes it. #
#########################################################
?>
```

PREPARSED/htmlDisplay__colorLegend.inc

```
<?php
#########################################################
# This file is generated by PrepStep and is required  #
# for proper operation of your SlidingDoor web site.  #
# Editing this file by hand will likely result in     #
# breaking your site.  You have been warned.          #
#                                                     #
# Generated By:  parseEventsFile.php::                #
#          htmlDisplay__colorIncludes                 #
#                                                     #
# This code is directly embedded inside the function  #
# HTMLDisplay::colorLegend and has access to it s     #
# variables: result                                   #
#########################################################

$result .= '<td width="20%"><font color="BLUE">Newcomer: Blue</font></td>';
$result .= '<td width="20%"><font color="GREEN">Beginner: Green</font></td>';
$result .= '<td width="20%"><font color="BLACK">Intermediate: Black</font></td>';
$result .= '<td width="20%"><font color="ORANGE">Advanced: Orange</font></td>';
$result .= '<td width="20%"><font color="RED">Open: Red</font></td>';
#########################################################
# THIS FILE IS COMPUTER GENERATED CODE DO NOT EDIT BY #
# HAND.  See top comment for information on what      #
# function produces it and what function includes it. #
#########################################################
?>
```

PREPARSED/htmlDisplay__displayFormEventsTable.inc

```
<?php
#########################################################
# This file is generated by PrepStep and is required  #
# for proper operation of your SlidingDoor web site.  #
# Editing this file by hand will likely result in     #
# breaking your site.  You have been warned.          #
#                                                     #
# Generated By: parseEventsFile.php::eventsParsedFile #
#                                                     #
# This code is directly embedded inside the function  #
# HTMLDisplay::displayFormEventsTable and has access  #
# to its variables: table, dances, person, and couple #
#########################################################

if ($couple!=0)
        $events=$couple->getDances();
if (formValue("Events")!="")
        $events=formValue("Events");
if ($events=="") $events=array();
$Newcomer=FALSE;
$NewcomerStandard=FALSE;
$NewcomerSmooth=FALSE;
$NewcomerLatin=FALSE;
$NewcomerRhythm=FALSE;
$Beginner=FALSE;
$BeginnerStandard=FALSE;
$BeginnerSmooth=FALSE;
$BeginnerLatin=FALSE;
$BeginnerRhythm=FALSE;
$Intermediate=FALSE;
$IntermediateStandard=FALSE;
$IntermediateSmooth=FALSE;
$IntermediateLatin=FALSE;
$IntermediateRhythm=FALSE;
$Advanced=FALSE;
$AdvancedStandard=FALSE;
$AdvancedSmooth=FALSE;
$AdvancedLatin=FALSE;
$AdvancedRhythm=FALSE;
$Open=FALSE;
$OpenStandard=FALSE;
$OpenSmooth=FALSE;
$OpenLatin=FALSE;
$OpenRhythm=FALSE;
$numEvents=count($dances);
for ($i=0;$i<$numEvents;$i++)
{
        $registered[$dances[$i]]=TRUE;
        $NewcomerTest=FALSE;
        if(!NewcomerTest)
                $NewcomerTest = strpos(".".$dances[$i],"N"."-");
        if($NewcomerTest!=1) $NewcomerTest=0;
        if ($NewcomerTest)
                $Newcomer=TRUE;
        if ($Newcomer && !$NewcomerStandard && in_array($dances[$i],array("N-I-W")))
                $NewcomerStandard = TRUE;
        if ($Newcomer && !$NewcomerSmooth && in_array($dances[$i],array("N-A-F")))
                $NewcomerSmooth = TRUE;
        if ($Newcomer && !$NewcomerLatin && in_array($dances[$i],array("N-I-R")))
```

```
                $NewcomerLatin = TRUE;
        if ($Newcomer && !$NewcomerRhythm && in_array($dances[$i],array("N-A-Sw")))
                $NewcomerRhythm = TRUE;
        $BeginnerTest=FALSE;
        if(!BeginnerTest)
                $BeginnerTest = strpos(".".$dances[$i],"B"."-");
        if($BeginnerTest!=1) $BeginnerTest=0;
        if ($BeginnerTest)
                $Beginner=TRUE;
        if ($Beginner && !$BeginnerStandard && in_array($dances[$i],
array("B-I-W", "B-I-T", "B-I-Vw", "B-I-F", "B-I-Q")))
                $BeginnerStandard = TRUE;
        if ($Beginner && !$BeginnerSmooth && in_array($dances[$i],
array("B-A-W", "B-A-T", "B-A-F", "B-A-Vw")))
                $BeginnerSmooth = TRUE;
        if ($Beginner && !$BeginnerLatin && in_array($dances[$i],
array("B-I-C", "B-I-Sa", "B-I-R", "B-I-P", "B-I-J")))
                $BeginnerLatin = TRUE;
        if ($Beginner && !$BeginnerRhythm && in_array($dances[$i],
array("B-A-C", "B-A-R", "B-A-Sw", "B-A-M", "B-A-B")))
                $BeginnerRhythm = TRUE;
        $IntermediateTest=FALSE;
        if(!IntermediateTest)
                $IntermediateTest = strpos(".".$dances[$i],"I"."-");
        if($IntermediateTest!=1) $IntermediateTest=0;
        if ($IntermediateTest)
                $Intermediate=TRUE;
        if ($Intermediate && !$IntermediateStandard && in_array($dances[$i],
array("I-I-W", "I-I-T", "I-I-Vw", "I-I-F", "I-I-Q")))
                $IntermediateStandard = TRUE;
        if ($Intermediate && !$IntermediateSmooth && in_array($dances[$i],
array("I-A-W", "I-A-T", "I-A-F", "I-A-Vw")))
                $IntermediateSmooth = TRUE;
        if ($Intermediate && !$IntermediateLatin && in_array($dances[$i],
array("I-I-C", "I-I-Sa", "I-I-R", "I-I-P", "I-I-J")))
                $IntermediateLatin = TRUE;
        if ($Intermediate && !$IntermediateRhythm && in_array($dances[$i],
array("I-A-C", "I-A-R", "I-A-Sw", "I-A-M", "I-A-B")))
                $IntermediateRhythm = TRUE;
        $AdvancedTest=FALSE;
        if(!AdvancedTest)
                $AdvancedTest = strpos(".".$dances[$i],"A"."-");
        if($AdvancedTest!=1) $AdvancedTest=0;
        if ($AdvancedTest)
                $Advanced=TRUE;
        if ($Advanced && !$AdvancedStandard && in_array($dances[$i],
array("A-I-W", "A-I-T", "A-I-Vw", "A-I-F", "A-I-Q")))
                $AdvancedStandard = TRUE;
        if ($Advanced && !$AdvancedSmooth && in_array($dances[$i],
array("A-A-W", "A-A-T", "A-A-F", "A-A-Vw")))
                $AdvancedSmooth = TRUE;
        if ($Advanced && !$AdvancedLatin && in_array($dances[$i],
array("A-I-C", "A-I-Sa", "A-I-R", "A-I-P", "A-I-J")))
                $AdvancedLatin = TRUE;
        if ($Advanced && !$AdvancedRhythm && in_array($dances[$i],
array("A-A-C", "A-A-R", "A-A-Sw", "A-A-M", "A-A-B")))
                $AdvancedRhythm = TRUE;
        $OpenTest=FALSE;
        if(!OpenTest)
                $OpenTest = strpos(".".$dances[$i],"O"."-");
        if($OpenTest!=1) $OpenTest=0;
```

```
        if ($OpenTest)
                $Open=TRUE;
        if ($Open && !$OpenStandard && in_array($dances[$i],
array("O-I-W", "O-I-T", "O-I-Vw", "O-I-F", "O-I-Q")))
                $OpenStandard = TRUE;
        if ($Open && !$OpenSmooth && in_array($dances[$i],
array("O-A-W", "O-A-T", "O-A-F", "O-A-Vw")))
                $OpenSmooth = TRUE;
        if ($Open && !$OpenLatin && in_array($dances[$i],
array("O-I-C", "O-I-Sa", "O-I-R", "O-I-P", "O-I-J")))
                $OpenLatin = TRUE;
        if ($Open && !$OpenRhythm && in_array($dances[$i],
array("O-A-C", "O-A-R", "O-A-Sw", "O-A-M", "O-A-B")))
                $OpenRhythm = TRUE;
}
$table .= "<table width=\"100%\">\n";
$table .= "\t<tr>\n";
$table .= "\t\t<th " . ($Newcomer ? "bgcolor=\"silver\" " :"") .
"colspan=\"4\">Newcomer - <input type=\"checkbox\" name=\"events[]\"
value=\"Newcomer-ALL\">All</th></tr>";
$table .= "\t<tr>\n";
$table .= "\t\t<th " . ($Standard ? "bgcolor=\"silver\" " :"") .
"width=\"25%\" colspan=\"1\">Standard - <input type=\"checkbox\"
name=\"events[]\" value=\"Newcomer-Standard-ALL\">All</th>";
$table .= "\t\t<th " . ($Smooth ? "bgcolor=\"silver\" " :"") .
"width=\"25%\" colspan=\"1\">Smooth - <input type=\"checkbox\"
name=\"events[]\" value=\"Newcomer-Smooth-ALL\">All</th>";
$table .= "\t\t<th " . ($Latin ? "bgcolor=\"silver\" " :"") .
"width=\"25%\" colspan=\"1\">Latin - <input type=\"checkbox\"
name=\"events[]\" value=\"Newcomer-Latin-ALL\">All</th>";
$table .= "\t\t<th " . ($Rhythm ? "bgcolor=\"silver\" " :"") .
"width=\"25%\" colspan=\"1\">Rhythm - <input type=\"checkbox\"
name=\"events[]\" value=\"Newcomer-Rhythm-ALL\">All</th>";
$table .= "\t</tr>";
$table .= "\t<tr>\n";
$table .= "\t\t<td ". ($registered["N-I-W"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"N-I-W\"" .
(in_array("N-I-W",$events) ? " checked=\"true\"" :"") . ">Waltz</td>\n";
$table .= "\t\t<td ". ($registered["N-A-F"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"N-A-F\"" .
(in_array("N-A-F",$events) ? " checked=\"true\"" :"") . ">Foxtrot</td>\n";
$table .= "\t\t<td ". ($registered["N-I-R"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"N-I-R\"" .
(in_array("N-I-R",$events) ? " checked=\"true\"" :"") . ">Rumba</td>\n";
$table .= "\t\t<td ". ($registered["N-A-Sw"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"N-A-Sw\"" .
(in_array("N-A-Sw",$events) ? " checked=\"true\"" :"") . ">Swing</td>\n";
$table .= "\t</tr>\n";
$table .= "\t<tr>\n";
$table .= "\t\t<th " . ($Beginner ? "bgcolor=\"silver\" " :"") .
"colspan=\"4\">Beginner - <input type=\"checkbox\" name=\"events[]\"
value=\"Beginner-ALL\">All</th></tr>";
$table .= "\t<tr>\n";
$table .= "\t\t<th " . ($Standard ? "bgcolor=\"silver\" " :"") .
"width=\"25%\" colspan=\"1\">Standard - <input type=\"checkbox\"
name=\"events[]\" value=\"Beginner-Standard-ALL\">All</th>";
$table .= "\t\t<th " . ($Smooth ? "bgcolor=\"silver\" " :"") .
"width=\"25%\" colspan=\"1\">Smooth - <input type=\"checkbox\"
name=\"events[]\" value=\"Beginner-Smooth-ALL\">All</th>";
$table .= "\t\t<th " . ($Latin ? "bgcolor=\"silver\" " :"") .
"width=\"25%\" colspan=\"1\">Latin - <input type=\"checkbox\"
```

```
name=\"events[]\" value=\"Beginner-Latin-ALL\">All</th>";
$table .= "\t\t<th " . ($Rhythm ? " bgcolor=\"silver\" " :"") .
"width=\"25%\" colspan=\"1\">Rhythm - <input type=\"checkbox\"
 name=\"events[]\" value=\"Beginner-Rhythm-ALL\">All</th>";
$table .= "\t</tr>";
$table .= "\t<tr>\n";
$table .= "\t\t<td ". ($registered["B-I-W"] ? " bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"B-I-W\"" .
(in_array("B-I-W",$events) ? " checked=\"true\"" :"") . ">Waltz</td>\n";
$table .= "\t\t<td ". ($registered["B-A-W"] ? " bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"B-A-W\"" .
(in_array("B-A-W",$events) ? " checked=\"true\"" :"") . ">Waltz</td>\n";
$table .= "\t\t<td ". ($registered["B-I-C"] ? " bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"B-I-C\"" .
(in_array("B-I-C",$events) ? " checked=\"true\"" :"") . ">Cha Cha</td>\n";
$table .= "\t\t<td ". ($registered["B-A-C"] ? " bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"B-A-C\"" .
(in_array("B-A-C",$events) ? " checked=\"true\"" :"") . ">Cha Cha</td>\n";
$table .= "\t</tr>\n";
$table .= "\t<tr>\n";
$table .= "\t\t<td ". ($registered["B-I-T"] ? " bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"B-I-T\"" .
(in_array("B-I-T",$events) ? " checked=\"true\"" :"") . ">Tango</td>\n";
$table .= "\t\t<td ". ($registered["B-A-T"] ? " bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"B-A-T\"" .
(in_array("B-A-T",$events) ? " checked=\"true\"" :"") . ">Tango</td>\n";
$table .= "\t\t<td ". ($registered["B-I-Sa"] ? " bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"B-I-Sa\"" .
(in_array("B-I-Sa",$events) ? " checked=\"true\"" :"") . ">Samba</td>\n";
$table .= "\t\t<td ". ($registered["B-A-R"] ? " bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"B-A-R\"" .
(in_array("B-A-R",$events) ? " checked=\"true\"" :"") . ">Rumba</td>\n";
$table .= "\t</tr>\n";
$table .= "\t<tr>\n";
$table .= "\t\t<td ". ($registered["B-I-Vw"] ? " bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"B-I-Vw\"" .
(in_array("B-I-Vw",$events) ? " checked=\"true\"" :"") .
">Viennese Waltz</td>\n";
$table .= "\t\t<td ". ($registered["B-A-F"] ? " bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"B-A-F\"" .
(in_array("B-A-F",$events) ? " checked=\"true\"" :"") . ">Foxtrot</td>\n";
$table .= "\t\t<td ". ($registered["B-I-R"] ? " bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"B-I-R\"" .
(in_array("B-I-R",$events) ? " checked=\"true\"" :"") . ">Rumba</td>\n";
$table .= "\t\t<td ". ($registered["B-A-Sw"] ? " bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"B-A-Sw\"" .
(in_array("B-A-Sw",$events) ? " checked=\"true\"" :"") . ">Swing</td>\n";
$table .= "\t</tr>\n";
$table .= "\t<tr>\n";
$table .= "\t\t<td ". ($registered["B-I-F"] ? " bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"B-I-F\"" .
(in_array("B-I-F",$events) ? " checked=\"true\"" :"") . ">Foxtrot</td>\n";
$table .= "\t\t<td ". ($registered["B-A-Vw"] ? " bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"B-A-Vw\"" .
(in_array("B-A-Vw",$events) ? " checked=\"true\"" :"") .
">Viennese Waltz</td>\n";
$table .= "\t\t<td ". ($registered["B-I-P"] ? " bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"B-I-P\"" .
(in_array("B-I-P",$events) ? " checked=\"true\"" :"") . ">Paso Doble</td>\n";
$table .= "\t\t<td ". ($registered["B-A-M"] ? " bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"B-A-M\"" .
```

```
(in_array("B-A-M",$events) ? " checked=\"true\"" :"") . ">Mambo</td>\n";
$table .= "\t</tr>\n";
$table .= "\t<tr>\n";
$table .= "\t\t<td ". ($registered["B-I-Q"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"B-I-Q\"" .
(in_array("B-I-Q",$events) ? " checked=\"true\"" :"") . ">Quickstep</td>\n";
$table .= "\t\t<td></td>\n";$table .= "\t\t<td ".
($registered["B-I-J"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"B-I-J\"" .
(in_array("B-I-J",$events) ? " checked=\"true\"" :"") . ">Jive</td>\n";
$table .= "\t\t<td ". ($registered["B-A-B"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"B-A-B\"" .
(in_array("B-A-B",$events) ? " checked=\"true\"" :"") . ">Bolero</td>\n";
$table .= "\t</tr>\n";
$table .= "\t<tr>\n";
$table .= "\t\t<th " . ($Intermediate ? "bgcolor=\"silver\" " :"") .
"colspan=\"4\">Intermediate - <input type=\"checkbox\" name=\"events[]\"
value=\"Intermediate-ALL\">All</th></tr>";
$table .= "\t<tr>\n";
$table .= "\t\t<th " . ($Standard ? "bgcolor=\"silver\" " :"") .
"width=\"25%\" colspan=\"1\">Standard - <input type=\"checkbox\"
name=\"events[]\" value=\"Intermediate-Standard-ALL\">All</th>";
$table .= "\t\t<th " . ($Smooth ? "bgcolor=\"silver\" " :"") .
"width=\"25%\" colspan=\"1\">Smooth - <input type=\"checkbox\"
name=\"events[]\" value=\"Intermediate-Smooth-ALL\">All</th>";
$table .= "\t\t<th " . ($Latin ? "bgcolor=\"silver\" " :"") .
"width=\"25%\" colspan=\"1\">Latin - <input type=\"checkbox\"
name=\"events[]\" value=\"Intermediate-Latin-ALL\">All</th>";
$table .= "\t\t<th " . ($Rhythm ? "bgcolor=\"silver\" " :"") .
"width=\"25%\" colspan=\"1\">Rhythm - <input type=\"checkbox\"
name=\"events[]\" value=\"Intermediate-Rhythm-ALL\">All</th>";
$table .= "\t</tr>";
$table .= "\t<tr>\n";
$table .= "\t\t<td ". ($registered["I-I-W"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"I-I-W\"" .
(in_array("I-I-W",$events) ? " checked=\"true\"" :"") .
">Waltz</td>\n";
$table .= "\t\t<td ". ($registered["I-A-W"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"I-A-W\"" .
(in_array("I-A-W",$events) ? " checked=\"true\"" :"") .
">Waltz</td>\n";
$table .= "\t\t<td ". ($registered["I-I-C"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"I-I-C\"" .
(in_array("I-I-C",$events) ? " checked=\"true\"" :"") .
">Cha Cha</td>\n";
$table .= "\t\t<td ". ($registered["I-A-C"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"I-A-C\"" .
(in_array("I-A-C",$events) ? " checked=\"true\"" :"") .
">Cha Cha</td>\n";
$table .= "\t</tr>\n";
$table .= "\t<tr>\n";
$table .= "\t\t<td ". ($registered["I-I-T"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"I-I-T\"" .
(in_array("I-I-T",$events) ? " checked=\"true\"" :"") .
">Tango</td>\n";
$table .= "\t\t<td ". ($registered["I-A-T"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"I-A-T\"" .
(in_array("I-A-T",$events) ? " checked=\"true\"" :"") .
">Tango</td>\n";
$table .= "\t\t<td ". ($registered["I-I-Sa"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"I-I-Sa\"" .
```

```
(in_array("I-I-Sa",$events) ? " checked=\"true\"" :"") .
">Samba</td>\n";
$table .= "\t\t<td ". ($registered["I-A-R"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"I-A-R\"" .
(in_array("I-A-R",$events) ? " checked=\"true\"" :"") .
">Rumba</td>\n";
$table .= "\t</tr>\n";
$table .= "\t<tr>\n";
$table .= "\t\t<td ". ($registered["I-I-Vw"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"I-I-Vw\"" .
(in_array("I-I-Vw",$events) ? " checked=\"true\"" :"") .
">Viennese Waltz</td>\n";
$table .= "\t\t<td ". ($registered["I-A-F"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"I-A-F\"" .
(in_array("I-A-F",$events) ? " checked=\"true\"" :"") .
">Foxtrot</td>\n";
$table .= "\t\t<td ". ($registered["I-I-R"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"I-I-R\"" .
(in_array("I-I-R",$events) ? " checked=\"true\"" :"") .
">Rumba</td>\n";
$table .= "\t\t<td ". ($registered["I-A-Sw"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"I-A-Sw\"" .
(in_array("I-A-Sw",$events) ? " checked=\"true\"" :"") .
">Swing</td>\n";
$table .= "\t</tr>\n";
$table .= "\t<tr>\n";
$table .= "\t\t<td ". ($registered["I-I-F"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"I-I-F\"" .
(in_array("I-I-F",$events) ? " checked=\"true\"" :"") .
">Foxtrot</td>\n";
$table .= "\t\t<td ". ($registered["I-A-Vw"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"I-A-Vw\"" .
(in_array("I-A-Vw",$events) ? " checked=\"true\"" :"") .
">Viennese Waltz</td>\n";
$table .= "\t\t<td ". ($registered["I-I-P"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"I-I-P\"" .
(in_array("I-I-P",$events) ? " checked=\"true\"" :"") .
">Paso Doble</td>\n";
$table .= "\t\t<td ". ($registered["I-A-M"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"I-A-M\"" .
(in_array("I-A-M",$events) ? " checked=\"true\"" :"") .
">Mambo</td>\n";
$table .= "\t</tr>\n";
$table .= "\t<tr>\n";
$table .= "\t\t<td ". ($registered["I-I-Q"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"I-I-Q\"" .
(in_array("I-I-Q",$events) ? " checked=\"true\"" :"") .
">Quickstep</td>\n";
$table .= "\t\t<td></td>\n";$table .= "\t\t<td ".
($registered["I-I-J"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"I-I-J\"" .
(in_array("I-I-J",$events) ? " checked=\"true\"" :"") .
">Jive</td>\n";
$table .= "\t\t<td ". ($registered["I-A-B"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"I-A-B\"" .
(in_array("I-A-B",$events) ? " checked=\"true\"" :"") .
">Bolero</td>\n";
$table .= "\t</tr>\n";
$table .= "\t<tr>\n";
$table .= "\t\t<th " . ($Advanced ? "bgcolor=\"silver\" " :"") .
"colspan=\"4\">Advanced - <input type=\"checkbox\" name=\"events[]\"
```

```
value=\"Advanced-ALL\">All</th></tr>";
$table .= "\t<tr>\n";
$table .= "\t\t<th " . ($Standard ? "bgcolor=\"silver\" " :"") .
"width=\"25%\" colspan=\"1\">Standard - <input type=\"checkbox\"
name=\"events[]\" value=\"Advanced-Standard-ALL\">All</th>";
$table .= "\t\t<th " . ($Smooth ? "bgcolor=\"silver\" " :"") .
"width=\"25%\" colspan=\"1\">Smooth - <input type=\"checkbox\"
name=\"events[]\" value=\"Advanced-Smooth-ALL\">All</th>";
$table .= "\t\t<th " . ($Latin ? "bgcolor=\"silver\" " :"") .
"width=\"25%\" colspan=\"1\">Latin - <input type=\"checkbox\"
name=\"events[]\" value=\"Advanced-Latin-ALL\">All</th>";
$table .= "\t\t<th " . ($Rhythm ? "bgcolor=\"silver\" " :"") .
"width=\"25%\" colspan=\"1\">Rhythm - <input type=\"checkbox\"
name=\"events[]\" value=\"Advanced-Rhythm-ALL\">All</th>";
$table .= "\t</tr>";
$table .= "\t<tr>\n";
$table .= "\t\t<td ". ($registered["A-I-W"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"A-I-W\"" .
(in_array("A-I-W",$events) ? " checked=\"true\"" :"") .
">Waltz</td>\n";
$table .= "\t\t<td ". ($registered["A-A-W"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"A-A-W\"" .
(in_array("A-A-W",$events) ? " checked=\"true\"" :"") .
">Waltz</td>\n";
$table .= "\t\t<td ". ($registered["A-I-C"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"A-I-C\"" .
(in_array("A-I-C",$events) ? " checked=\"true\"" :"") .
">Cha Cha</td>\n";
$table .= "\t\t<td ". ($registered["A-A-C"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"A-A-C\"" .
(in_array("A-A-C",$events) ? " checked=\"true\"" :"") .
">Cha Cha</td>\n";
$table .= "\t</tr>\n";
$table .= "\t<tr>\n";
$table .= "\t\t<td ". ($registered["A-I-T"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"A-I-T\"" .
(in_array("A-I-T",$events) ? " checked=\"true\"" :"") .
">Tango</td>\n";
$table .= "\t\t<td ". ($registered["A-A-T"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"A-A-T\"" .
(in_array("A-A-T",$events) ? " checked=\"true\"" :"") .
">Tango</td>\n";
$table .= "\t\t<td ". ($registered["A-I-Sa"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"A-I-Sa\"" .
(in_array("A-I-Sa",$events) ? " checked=\"true\"" :"") .
">Samba</td>\n";
$table .= "\t\t<td ". ($registered["A-A-R"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"A-A-R\"" .
(in_array("A-A-R",$events) ? " checked=\"true\"" :"") .
">Rumba</td>\n";
$table .= "\t</tr>\n";
$table .= "\t<tr>\n";
$table .= "\t\t<td ". ($registered["A-I-Vw"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"A-I-Vw\"" .
(in_array("A-I-Vw",$events) ? " checked=\"true\"" :"") .
">Viennese Waltz</td>\n";
$table .= "\t\t<td ". ($registered["A-A-F"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"A-A-F\"" .
(in_array("A-A-F",$events) ? " checked=\"true\"" :"") .
">Foxtrot</td>\n";
$table .= "\t\t<td ". ($registered["A-I-R"] ? "bgcolor=\"silver\" " : "") .
```

```
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"A-I-R\"" .
(in_array("A-I-R",$events) ? " checked=\"true\"" :"") .
">Rumba</td>\n";
$table .= "\t\t<td ". ($registered["A-A-Sw"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"A-A-Sw\"" .
(in_array("A-A-Sw",$events) ? " checked=\"true\"" :"") .
">Swing</td>\n";
$table .= "\t</tr>\n";
$table .= "\t<tr>\n";
$table .= "\t\t<td ". ($registered["A-I-F"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"A-I-F\"" .
(in_array("A-I-F",$events) ? " checked=\"true\"" :"") .
">Foxtrot</td>\n";
$table .= "\t\t<td ". ($registered["A-A-Vw"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"A-A-Vw\"" .
(in_array("A-A-Vw",$events) ? " checked=\"true\"" :"") .
">Viennese Waltz</td>\n";
$table .= "\t\t<td ". ($registered["A-I-P"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"A-I-P\"" .
(in_array("A-I-P",$events) ? " checked=\"true\"" :"") .
">Paso Doble</td>\n";
$table .= "\t\t<td ". ($registered["A-A-M"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"A-A-M\"" .
(in_array("A-A-M",$events) ? " checked=\"true\"" :"") .
">Mambo</td>\n";
$table .= "\t</tr>\n";
$table .= "\t<tr>\n";
$table .= "\t\t<td ". ($registered["A-I-Q"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"A-I-Q\"" .
(in_array("A-I-Q",$events) ? " checked=\"true\"" :"") .
">Quickstep</td>\n";
$table .= "\t\t<td></td>\n";$table .= "\t\t<td ".
($registered["A-I-J"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"A-I-J\"" .
(in_array("A-I-J",$events) ? " checked=\"true\"" :"") .
">Jive</td>\n";
$table .= "\t\t<td ". ($registered["A-A-B"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"A-A-B\"" .
(in_array("A-A-B",$events) ? " checked=\"true\"" :"") .
">Bolero</td>\n";
$table .= "\t</tr>\n";
$table .= "\t<tr>\n";
$table .= "\t\t<th " . ($Open ? "bgcolor=\"silver\" " :"") .
"colspan=\"4\">Open - <input type=\"checkbox\" name=\"events[]\"
value=\"Open-ALL\">All</th></tr>";
$table .= "\t<tr>\n";
$table .= "\t\t<th " . ($Standard ? "bgcolor=\"silver\" " :"") .
"width=\"25%\" colspan=\"1\">Standard - <input type=\"checkbox\"
name=\"events[]\" value=\"Open-Standard-ALL\">All</th>";
$table .= "\t\t<th " . ($Smooth ? "bgcolor=\"silver\" " :"") .
"width=\"25%\" colspan=\"1\">Smooth - <input type=\"checkbox\"
name=\"events[]\" value=\"Open-Smooth-ALL\">All</th>";
$table .= "\t\t<th " . ($Latin ? "bgcolor=\"silver\" " :"") .
"width=\"25%\" colspan=\"1\">Latin - <input type=\"checkbox\"
name=\"events[]\" value=\"Open-Latin-ALL\">All</th>";
$table .= "\t\t<th " . ($Rhythm ? "bgcolor=\"silver\" " :"") .
"width=\"25%\" colspan=\"1\">Rhythm - <input type=\"checkbox\"
name=\"events[]\" value=\"Open-Rhythm-ALL\">All</th>";
$table .= "\t</tr>";
$table .= "\t<tr>\n";
$table .= "\t\t<td ". ($registered["O-I-W"] ? "bgcolor=\"silver\" " : "") .
```

```
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"O-I-W\"" .
(in_array("O-I-W",$events) ? " checked=\"true\"" :"") .
">Waltz</td>\n";
$table .= "\t\t<td " . ($registered["O-A-W"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"O-A-W\"" .
(in_array("O-A-W",$events) ? " checked=\"true\"" :"") .
">Waltz</td>\n";
$table .= "\t\t<td " . ($registered["O-I-C"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"O-I-C\"" .
(in_array("O-I-C",$events) ? " checked=\"true\"" :"") .
">Cha Cha</td>\n";
$table .= "\t\t<td " . ($registered["O-A-C"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"O-A-C\"" .
(in_array("O-A-C",$events) ? " checked=\"true\"" :"") .
">Cha Cha</td>\n";
$table .= "\t</tr>\n";
$table .= "\t<tr>\n";
$table .= "\t\t<td " . ($registered["O-I-T"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"O-I-T\"" .
(in_array("O-I-T",$events) ? " checked=\"true\"" :"") .
">Tango</td>\n";
$table .= "\t\t<td " . ($registered["O-A-T"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"O-A-T\"" .
(in_array("O-A-T",$events) ? " checked=\"true\"" :"") .
">Tango</td>\n";
$table .= "\t\t<td " . ($registered["O-I-Sa"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"O-I-Sa\"" .
(in_array("O-I-Sa",$events) ? " checked=\"true\"" :"") .
">Samba</td>\n";
$table .= "\t\t<td " . ($registered["O-A-R"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"O-A-R\"" .
(in_array("O-A-R",$events) ? " checked=\"true\"" :"") .
">Rumba</td>\n";
$table .= "\t</tr>\n";
$table .= "\t<tr>\n";
$table .= "\t\t<td " . ($registered["O-I-Vw"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"O-I-Vw\"" .
(in_array("O-I-Vw",$events) ? " checked=\"true\"" :"") .
">Viennese Waltz</td>\n";
$table .= "\t\t<td " . ($registered["O-A-F"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"O-A-F\"" .
(in_array("O-A-F",$events) ? " checked=\"true\"" :"") .
">Foxtrot</td>\n";
$table .= "\t\t<td " . ($registered["O-I-R"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"O-I-R\"" .
(in_array("O-I-R",$events) ? " checked=\"true\"" :"") .
">Rumba</td>\n";
$table .= "\t\t<td " . ($registered["O-A-Sw"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"O-A-Sw\"" .
(in_array("O-A-Sw",$events) ? " checked=\"true\"" :"") .
">Swing</td>\n";
$table .= "\t</tr>\n";
$table .= "\t<tr>\n";
$table .= "\t\t<td " . ($registered["O-I-F"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"O-I-F\"" .
(in_array("O-I-F",$events) ? " checked=\"true\"" :"") .
">Foxtrot</td>\n";
$table .= "\t\t<td " . ($registered["O-A-Vw"] ? "bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"O-A-Vw\"" .
(in_array("O-A-Vw",$events) ? " checked=\"true\"" :"") .
">Viennese Waltz</td>\n";
```

```
$table .= "\t\t<td ". ($registered["O-I-P"] ? " bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"O-I-P\"" .
(in_array("O-I-P",$events) ? " checked=\"true\"" :"") .
">Paso Doble</td>\n";
$table .= "\t\t<td ". ($registered["O-A-M"] ? " bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"O-A-M\"" .
(in_array("O-A-M",$events) ? " checked=\"true\"" :"") .
">Mambo</td>\n";
$table .= "\t</tr>\n";
$table .= "\t<tr>\n";
$table .= "\t\t<td ". ($registered["O-I-Q"] ? " bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"O-I-Q\"" .
(in_array("O-I-Q",$events) ? " checked=\"true\"" :"") .
">Quickstep</td>\n";
$table .= "\t\t<td></td>\n";$table .= "\t\t<td ".
($registered["O-I-J"] ? " bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"O-I-J\"" .
(in_array("O-I-J",$events) ? " checked=\"true\"" :"") .
">Jive</td>\n";
$table .= "\t\t<td ". ($registered["O-A-B"] ? " bgcolor=\"silver\" " : "") .
"colspan=\"1\"><input type=\"checkbox\" name=\"events[]\" value=\"O-A-B\"" .
(in_array("O-A-B",$events) ? " checked=\"true\"" :"") .
">Bolero</td>\n";
$table .= "\t</tr>\n";
$table .= "</table>\n";
#########################################################
# THIS FILE IS COMPUTER GENERATED CODE DO NOT EDIT BY   #
# HAND.  See top comment for information on what        #
# function produces it and what function includes it.   #
#########################################################
?>
```

PREPARSED/htmlDisplay_feeCategory.inc

```
<?php
$ages = array("<a href=\"$baseURL/PREPARSED/packages.html#student\"
target=\"_blank\">Student</a>"=>1,
              "<a href=\"$baseURL/PREPARSED/packages.html#adult\"
target=\"_blank\">Adult</a>"=>2);
?>
```

PREPARSED/htmlDisplay__packages.inc

```
<?php

#$packages = array("<a href=\"$baseURL\PREPARSED\packages.html#Newcomer Only\"
target=\"_blank\">Newcomer Only</a> (\$10)"=>1);

$packages = array("<a href=\"$baseURL\PREPARSED\packages.html#Newcomer Only\"
target=\"_blank\">Newcomer Only</a> (\$10)"=>1,
                  "<a href=\"$baseURL\PREPARSED\packages.html#Regular
Competitor\" target=\"_blank\">Regular Competitor</a> (\$25/\$50)"=>2);
?>
```

PREPARSED/htmlDisplay__partnerTable__header.inc

```
<?php
#########################################################
# This file is generated by PrepStep and is required  #
# for proper operation of your SlidingDoor web site.  #
# Editing this file by hand will likely result in     #
# breaking your site.  You have been warned.          #
#                                                     #
# Generated By:  parseEventsFile.php::                #
#          htmlDisplay__partnerTableIncludes          #
#                                                     #
# This code is directly embedded inside the function  #
# HTMLDisplay::partnerTable and has access to its     #
# variables: this, tempCouple, table                  #
#########################################################

$table .= "\t<tr>\n\t\t<th colspan=\"2\">Options</th><th>Partner</th>
<th>Standard</th><th>Smooth</th><th>Latin</th><th>Rhythm</th></tr>\n";
#########################################################
# THIS FILE IS COMPUTER GENERATED CODE DO NOT EDIT BY #
# HAND.  See top comment for information on what      #
# function produces it and what function includes it. #
#########################################################
?>
```

PREPARSED/htmlDisplay__partnerTable__row.inc

```
<?php
#########################################################
# This file is generated by PrepStep and is required  #
# for proper operation of your SlidingDoor web site.  #
# Editing this file by hand will likely result in     #
# breaking your site.  You have been warned.          #
#                                                     #
# Generated By:  parseEventsFile.php::                #
#         htmlDisplay__partnerTableIncludes           #
#                                                     #
# This code is directly embedded inside the function  #
# HTMLDisplay::partnerTable and has access to its     #
# variables: this, tempCouple, table                  #
#########################################################

$StandardDances = $this->colorDances($tempCouple->getDances("Standard"));
$table .= "\t\t<td>$StandardDances</td>\n";
$SmoothDances = $this->colorDances($tempCouple->getDances("Smooth"));
$table .= "\t\t<td>$SmoothDances</td>\n";
$LatinDances = $this->colorDances($tempCouple->getDances("Latin"));
$table .= "\t\t<td>$LatinDances</td>\n";
$RhythmDances = $this->colorDances($tempCouple->getDances("Rhythm"));
$table .= "\t\t<td>$RhythmDances</td>\n";
#########################################################
# THIS FILE IS COMPUTER GENERATED CODE DO NOT EDIT BY #
# HAND.  See top comment for information on what      #
# function produces it and what function includes it. #
#########################################################
?>
```

PREPARSED/packages.html

```
<html>
<head>
<title>Package Defination</title>
</head>
<body bgcolor="white">
<h1>Packages Offered</h1>
<a href="#Newcomer Only">Newcomer Only</a> |
<a href="#Regular Competitor">Regular Competitor</a><br />
<h3><a name="Newcomer Only">Newcomer Only</a></h3>
This package is availible only to competitors competeing only in the newcomer
events. The beginner team match and any fun dances are also open to holders
of this package. The registration package will not allow you to select non-
newcomer events if you choose this package. This package is offered at the
low price of $10 to all competitors.

<h3><a name="Regular Competitor">Regular Competitor</a></h3>
This is the package used by all non-newcomers at the MIT competition. Holders
of this package are allowed to register for upto one level per dance for
as many dances as they wish. Competitors must still obey eligibity
(YCN/USABDA proficiency points. Regular Competitors are eligible to compete
in any team matches or fun dances offered. This package is $50, $25 dollars
with a current student ID.

<h1>Fee Category Defination</h1>
<a href="#student">Student</a> | <a href="#adult">Adult</a><br />
<h3><a name="student">Student</a></h3>
Anyone attending school full-time, working towards a degree is considered
a student for fee purposes, regarless of age. (Yes this means that professors/
instructors do <b>not</b> count as students.) K-12 are cinsider workig
towards a High School degree.

<h3><a name="adult">Adult</a></h3>
Anyone whoe is no a <a href="#student">student</a> for pricing purposes is an
adult.

</body>
</html>
```

C.4 SlidingDoors Scripts

SCRIPTS/partner-check.php

```
<?php
#####################################################################
# This file is part of SlidingDoors                                 #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd        #
# SlidingDoors is availible for license under the GPL, see          #
# the COPYING file in the root directory of the install for         #
# the full terms of the GPL.                                        #
#                                                                   #
# File: partner-check.inc                                           #
# Author: Eric D. Nielsen                                           #
# Description: An interstitial page that handles the form data      #
#              passed from partners.php. This page will             #
#              validate the data, check for name collisions,        #
#              check for event collisions and insert valid          #
#              registrations into the database. It dispatches       #
#              to collision resolution pages or returns the         #
#              user to partners.                                    #
# Internal Links: partners.php                                      #
#                 prime-collision.php                               #
#                 partners.php                                      #
# External Links: SlidingDoors information page (footer)            #
#                 Comp-in-a-box information page (footer)           #
#                 OpenImpetus information page (footer)             #
# Change Log: 8/23/01 -- created -- edn                             #
#####################################################################

    session_start();
include "include_others.inc";
$contraCheck = new ContraCheck($db,"NULL");

unset($formVars);
$includeEmail = ($HTTP_POST_VARS["ignoreEmail"]!="on");
if (!$includeEmail)
{
  $propagateEnteredEmail=$HTTP_POST_VARS["Email"];
  session_register("propagateEnteredEmail");
}

$primeID = $HTTP_POST_VARS["primeID"];
$coupleID = $HTTP_POST_VARS["coupleid"];
$firstName = htmlspecialchars($HTTP_POST_VARS["FirstName"]);
$lastName = htmlspecialchars($HTTP_POST_VARS["LastName"]);
$email = strtolower(htmlspecialchars($HTTP_POST_VARS["Email"]));
$affilID = $HTTP_POST_VARS["AffilID"]; # not user entered, therefore not escaped
$newAffil = htmlspecialchars($HTTP_POST_VARS["NewAffil"]);
$affilShort = htmlspecialchars($HTTP_POST_VARS["AffilShort"]);
$age = $HTTP_POST_VARS["Age"];
$package = $HTTP_POST_VARS["Package"];
$action = $HTTP_POST_VARS["action"];
$action = substr($action,0,strpos($action,'_'));

$events = $HTTP_POST_VARS["events"];
$validated=TRUE;
$role = $HTTP_POST_VARS["role"];
if ($affilID=="choose")
```

```
        $affilID="";


if ($coupleID!="")
{
    $couple = new Couple($db);
    $couple->setID($coupleID);
    $couple->retrieve();
    $oldRole = ($primeID==$couple->getLeader() ? "leading" : "following");
    $getFunction = "get" . ($oldRole=="leading"? "Follower" : "Leader");
    $partnerID= $couple->$getFunction();
}


$formVars["Age"]=$age;
$formVars["Role"]=$role;



if ($tba)
{
    $firstName="TBA";
    if (!is_int($lastName))
        $lastName = $db->getNextTBA();
    $package=2;
}
if ($firstName=="TBA")
    $tba=TRUE;
if ($tba)
    $affilID=-1;

if ($firstName!="")
    $formVars["FirstName"]=$firstName;
else
{
    $formVars["FirstNameMsg"]="<font color=\"red\">
Please enter your partner's first name.</font><BR>\n";
    $validated=FALSE;
}
if ($lastName!="")
    $formVars["LastName"]=$lastName;
else
{
    $formVars["LastNameMsg"]="<font color=\"red\">
Please enter your partner's last name.</font><BR>\n";
    $validated=FALSE;
}

if ($package!="")
    $formVars["Package"]=$package;
else if (!$tba)
{
    $formVars["PackageMsg"]="<font color=\"red\">
Please select a package.</strong><br />\n";
    $validated=FALSE;
}
if ($tba || $email!="")
    $formVars["Email"]=$email;
else
{
    $formVars["EmailMsg"]="<font color=\"red\">
```

```
Please enter your partner's email.  If they do not have an email account
enter your own again.</font><BR>\n";
    $validated=FALSE;
}


if ($affilID!="")
    $formVars["Affil"]=$affilID;
if ($newAffil!="")
    $formVars["NewAffil"]=$newAffil;
if ($affilShort!="")
    $formVars["AffilShort"]=$affilShort;
if ($affilID!="" && ($newAffil!="" || $affilShort != ""))
{
    $validated=FALSE;
    $formVars["AffilMsg"]="<font color=\"red\">Please do not select an
affiliation from the list box and also enter information into the
\"New Affiliation\" boxes.<font color=\"red\"><BR>\n";
}
if (!$tba && $affilID=="" && ($newAffil=="" || $affilShort == ""))
{
    $validated=FALSE;
    $formVars["AffilMsg"]="<font color=\"red\">Please either select an
affiliation from the box, or enter a new one with long and short form in
the text boxes below.</font><BR>";
}
if (($newAffil!="" && $affilShort=="") || ($newAffil=="" && $affilShort!=""))
{
    $validated = FALSE;
    $formVars["NewAffilMsg"]="<font color=\"red\">Please provide both a
long and short form of your affiliation's name.  Here are some examples:
<BR>Massachusetts Institute of Technology --> MIT<BR>Harvard University -->
Harvard<BR>Fred Astaire Dance Studio, Boston --> FADS Boston<BR></font>";
}

if (count($events)==0)
{
    $validated= FALSE;
    $formVars["EventsMsg"]="<font color=\"red\">Please select at least one
event to compete in</font>";
} else
{
    $tempCouple = new Couple($db);
    $tempCouple->setEvents($events);
    $formVars["Events"]=$tempCouple->getDances();
}
if (!$validated)
{
    session_register("primeID");
    session_register("formVars");
    localRedirect("Location: $baseURL/partners.php#reg");
    exit;
}

if ($affilID!="" && $affilID!=-1)
{
    $affiliation = new Organization($db,$affil);
#    $affilName = $affiliation->getName();
}
if (!$tba && $affilID=="")
{
```

```
        $affiliation = new RegOrg($db);
        $affiliation->setName($newAffil);
        $affilName = $newAffil;
        $affiliation->setAbbrev($affilShort);
            $affiliation->setOrgPays(TRUE);
        $affiliation->postToDB();
        $affilID = $affiliation->getID();
        if (!$affilID)
        {
            errorPage($PHP_SELF, "root","Database Inaccessible",
            "Organization::postToDB unable to access database",TRUE);
            exit;
        }
}

if ($action=="Register")
{
        $person = new Person($db);
        $person->setFirstName($firstName);
        $person->setLastName($lastName);
        $person->setEmail($email);
        $person->setAgeLevel($age);
        $person->setPackage($package);
        if (!$tba && $affilID!=-1)
            $person->setAffilID($affilID);
else
        $person->setOrgPays(FALSE);

        if (!$tba && ($conflicts = $db->personConflicts($person,$includeEmail)))
        {
            session_register("role");
            session_register("primeID");
            session_register("conflicts");
            session_register("formVars");
            session_register("action");
            localRedirect("Location: $baseURL/partner-collision.php");
            exit;
        }
        $person->setPaid(FALSE);

}
else
{

        #check if anypersonal info changed, if so check for conflicts
        $partner = new Person($db,$partnerID);
        $partner->setID($partnerID);
        $partner->retrieve();
        $updatedPersonal=FALSE;
        if ($firstName != $partner->getFirstName())
        {
            $updatedPersonal=TRUE;
        }
        if ($lastName != $partner->getLastName())
        {
            $updatedPersonal=TRUE;
        }
        if ($email != $partner->getEmail())
        {
            $updatedPersonal=TRUE;
        }
```

```
    if (!$tba && $affilID != $partner->getAffilID())
    {
        $updatedPersonal=TRUE;
    }
    if ($age != $partner->getAgeLevel())
    {
        $updatedPersonal=TRUE;
    }
    if ($package != $partner->getPackage())
    {
        $updatedPersonal=TRUE;
        $tempCouple = new Couple();
        $tempCouple->setEvents($events);
        $eventIDs = $tempCouple->getGeneralData("eventid");
        $tempEvents = arrayToCSL($eventIDs);
        $compRulesConflicts = $contraCheck->checkPersonPackage($partnerID,
                                        $tempEvents,$primeID,$package);
        if (!$compRulesConflicts->result)
        {
            # setup variables
            $couple=$coupleID;
            $formVars["Events"]=array();
            $formVars["EventsMsg"]=$compRulesConflicts->getErrorMessage();
            session_register("primeID");
            session_register("couple");
            session_register("partnerID");
            session_register("formVars");
            localRedirect("Location: $baseURL/partners.php#reg");
            exit;
        }
    }
    if ($updatedPersonal)
    {
        $person = new Person($db);
        $person->setFirstName($firstName);
        $person->setLastName($lastName);
        $person->setEmail($email);
        $person->setAgeLevel($age);
        $person->setPackage($package);
        if (!$tba && $affilID!=-1)
            $person->setAffilID($affilID);
else
    $person->setOrgPays(FALSE);
        $conflicts = $db->personConflicts($person,$includeEmail);
        if (count($conflicts)==1)
            $conflicts = arrayRemove($conflicts,$partnerID);
        if (!$tba && count($conflicts))
        {
            session_register("role");
            session_register("primeID");
            session_register("partnerID");
            session_register("conflicts");
            session_register("formVars");
            session_register("action");
            localRedirect("Location: $baseURL/partner-collision.php");
            exit;
        }
        $person->setID($partnerID);
    }
    else
        $person = $partner;
```

```
    $storedRole = ($couple->getLeader()==$primeID ?
                   "leading" :
                   "following");
    if ($role!=$storedRole)
    {
        $tempCouple = new Couple($db);
        $tempCouple->setLeader(($role=="leading" ? $primeID : $partnerID));
        $tempCouple->setFollower(($role=="leading" ? $partnerID : $primeID));
        $tempCoupleCheck = $tempCouple->searchDB();
        if (count($tempCoupleCheck))
        {
            $tempCouple->setID($tempCoupleCheck[0]);
            $tempCouple->retrieve();
            $tempEvents = $tempCouple->getDances();


            ## Invoke CC

            $multiPartnerConflicts = $contraCheck->checkCoupleConflicts($couple,$events);
            $compRulesConflicts = $contraCheck->checkCoupleCompRules($couple,$events);
            if (!$multiPartnerConflicts->result || !$compRulesConflicts->result)
            {
                # setup variables
                $couple=$coupleID;
                $formVars["Events"]=array();
                $formVars["EventsMsg"]=$multiPartnerConflicts->getErrorMessage() .
                    $compRulesConflicts->getErrorMessage();
                session_register("primeID");
                session_register("couple");
                session_register("partnerID");
                session_register("formVars");
                localRedirect("Location: $baseURL/partners.php#reg");
                exit;
            }


            $tempEvents = arrayJoin($tempEvents,$events);
            $tempEvents = array_unique($tempEvents);
            $tempEvents->setEvents($tempEvents);
            $tempCouple->postToDB();
            $couple = new Couple($db);
            $couple->setID($coupleID);
            $couple->remove();
            session_register("primeID");
#           $db->rebuildStats();
            localRedirect("Location: $baseURL/partners.php");
            exit();
        }
    }
    $person->setID($partnerID);
}

if (!$tba && $affilID!=-1)
    $person->setAffilID($affilID);
else
    $person->setOrgPays(FALSE);

if ($action=="Register" || $updatedPersonal)
    $person->postToDB();

$partnerID=$person->getID();
```

```
if (!$partnerID)
{
    errorPage($PHP_SELF, "root","Database Inaccessible",
        "Person::PostToDB unable to access database",TRUE);
    exit;
}
if (!isset($couple))
{
        $couple = new Couple($db);
        $couple->setLeader(($role=="leading" ? $primeID : $partnerID));
        $couple->setFollower(($role=="leading" ? $partnerID : $primeID));
        $couple->postToDB();
        $coupleID=$couple->getID();
}
$multiPartnerConflicts = $contraCheck->checkCoupleConflicts($couple,$events);
$compRulesConflicts = $contraCheck->checkCoupleCompRules($couple,$events);
#$eligibilityConflicts = $contraCheck->getEligibilityConflicxts();
if (!$multiPartnerConflicts->result || !$compRulesConflicts->result ||
    count($eligibilityConflicts))
{
    # setup variables
    if ($action=="Register")
    {
        $couple->remove();
        $person->remove();
    }
    else
    {
        $couple=$coupleID;
        session_register("couple");
        session_register("partnerID");
    }
    session_register("primeID");
    $formVars["Events"]=array();
    $formVars["EventsMsg"]=$multiPartnerConflicts->getErrorMessage() .
        $compRulesConflicts->getErrorMessage();
    session_register("formVars");
    localRedirect("Location: $baseURL/partners.php#reg");
    exit;
}

if ($action=="Register")
{
    $couple->setEvents($events);
    $couple->postToDB();
}
else
{
    $storedRole = ($couple->getLeader()==$HTTP_POST_VARS["primeID"] ?
                "leading" :
                "following");
    if ($role != $storedRole)
    {
        $couple->setLeader(($role=="leading" ? $primeID : $partnerID));
        $couple->setFollower(($role=="leading" ? $partnerID : $primeID));
        $couple->setEvents($events);
        $couple->postToDB();
    }
    else
    {
        $couple->setEvents($events);
```

```
            $couple->postToDB();
        }
    }
    #$db->rebuildStats();
    unset($formVars);
    session_unregister("formVars");
    session_unregister("couple");
    session_unregister("person");
    session_register("primeID");
    localRedirect("Location: $baseURL/partners.php");
    exit;
    ?>
```

SCRIPTS/partners-dispatch.php

```
<?php
#################################################################
# This file is part of SlidingDoors                             #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    #
# SlidingDoors is availible for license under the GPL, see      #
# the COPYING file in the root directory of the install for     #
# the full terms of the GPL.                                    #
#                                                               #
# File: partners-dispatch.php                                   #
# Author: Eric D. Nielsen                                       #
# Description: An interstitial page that returns to handles     #
#              non registration requests from partners.php, ie  #
#              retrievals, drops, and cancels, nothing that     #
#              requires inserts or updates.                     #
# Internal Links: partners-action.php                           #
# External Links: SlidingDoors information page (footer)        #
#                 Comp-in-a-box information page(footer)        #
#                 OpenImpetus information page  (footer)        #
# Change Log: 8/22/01 -- created -- edn                         #
#             7/06/02 -- GPL -- edn                             #
#################################################################
# Do not remove the following line
include "include_others.inc";
session_start();
$couple=$HTTP_POST_VARS["coupleid"];
$dispatch=$HTTP_POST_VARS["action"];
$primeID=$HTTP_POST_VARS["primeID"];
switch($dispatch)
{
    case "update": session_unregister("formVars");
                   session_register("primeID");
                   session_register("couple");
                   localRedirect("Location: $baseURL/partners.php");
                   exit;
                   break;
    case "drop":
        $aCouple = new Couple($db);
        $aCouple->setID($couple);
        $aCouple->retrieve();
        $leader = $aCouple->getLeader();
        $follower =$aCouple->getFollower();
        if ($leader==$primeID)
            $partnerID=$follower;
        else
            $partnerID=$leader;
        $aCouple->remove();
        $aCompetitor = new Competitor($db);
        $aCompetitor->setID($partnerID);
        $result = $aCompetitor->getPartners();
        if (!count($result))
            $aCompetitor->remove();
        session_unregister("formVars");
        session_unregister("couple");
        session_register("primeID");
        localRedirect("Location: $baseURL/partners.php");
        exit;
        break;
     case "cancel":
        $competitor = new Competitor($db);
```

```
        $competitor->setID($primeID);
        $competitor->retrieve();
        $leaders = $competitor->getPartners("lead");
        $followers = $competitor->getPartners("follow");
        $numLeads = count($leaders);
        for ($i=0;$i<$numLeads;$i++)
        {
            $aCouple = new Couple ($db);
            $aCouple->setLeader($leaders[$i]);
            $aCouple->setFollower($competitor->getID());
            $id = $aCouple->searchDB();
            if (count($id) != 1) die ("Multiple couples");
            $aCouple->setID($id[0]);
            $aCouple->remove();
        }
        $numFollows = count($followers);
        for ($i=0;$i<$numFollows;$i++)
        {
            $aCouple = new Couple ($db);
            $aCouple->setFollower($followers[$i]);
            $aCouple->setLeader($competitor->getID());
            $id = $aCouple->searchDB();
            if (count($id) != 1) die ("Multiple couples");
            $aCouple->setID($id[0]);
            $aCouple->remove();
        }
        session_unregister("primeID");
        session_unregister("person");
        session_unregister("formVars");
        session_unregister("couple");
            localRedirect("Location: $baseURL/partners-dropped.php");
          default: die ("Unknown Dispatch code received");
}
?>
```

SCRIPTS/prime-check.php

```
<?php
#####################################################################
# This file is part of SlidingDoors                                 #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd        #
# SlidingDoors is availible for license under the GPL, see          #
# the COPYING file in the root directory of the install for         #
# the full terms of the GPL.                                        #
#                                                                   #
# File: prime-check.inc                                             #
# Author: Eric D. Nielsen                                           #
# Description: An interstitial page that handles the form data      #
#              passed from prime.php.  This page will validate      #
#              the data, check for name collisions, insert the      #
#              new person into the database and pass the PID        #
#              to the partners page if all goes well.  In case      #
#              of error it passed control back to prime.php to      #
#              handle missing/invalid data and to                   #
#              prime-collision.php to handle close matches.         #
# Internal Links: prime.php                                         #
#                 prime-collision.php                               #
#                 partners.php                                      #
# External Links: SlidingDoors information page (footer)            #
#                 Comp-in-a-box information page (footer)           #
#                 OpenImpetus information page (footer)             #
# Change Log: 6/12/01 -- created -- edn                             #
#             2/14/02 -- prepped for PrepStep -- edn                #
#             7/02/02 -- new validation system -- edn               #
#####################################################################

session_start();
include "include_others.inc";
unset($formVars);

$includeEmail = ($HTTP_POST_VARS["ignoreEmail"]!="on");
if ($includeEmail)
   {
      $propagateEnteredEmail=$HTTP_POST_VARS["Email"];
      session_register("propagateEnteredEmail");
   }
$formVars=array();
$person = new Person($db);

$validated = $person->validateInputs(&$formVars,"PRIME");
if (!$validated)
{
    session_register("formVars");
    localRedirect("Location: $baseURL/prime.php");  # Branch Works
    exit;
}

$person->populateFromFormVars($formVars);
if ($conflicts = $db->personConflicts($person,$includeEmail))
{
    $exact = $db->personConflicts($person,$includeEmail,TRUE);

    if (count($exact)!=1)
    {
        session_register("conflicts");
        session_register("formVars");
```

```
        localRedirect("Location: $baseURL/prime-collision.php"); # Branch works
        exit;
    }
    else
    {
        $keys = array_keys($exact);
        $primeID=$exact[$keys[0]];
        session_register("primeID");
        localRedirect("Location: $baseURL/partners.php"); # Branch works
        exit;
    }
}

if (in_array("Affiliation", $SD_elements))
{
    $newAffil = $HTTP_POST_VARS[$prefix . "NewAffil"];
    $affilShort = $HTTP_POST_VARS[$prefix . "AffilShort"];
    $affilID = $HTTP_POST_VARS[$prefix . "AffilID"];
    if ($affilShort!="")
    {
        $affiliation = new RegOrg($db);
        $affiliation->setName($newAffil);
        $affiliation->setOrgPays($SD_default_org_pays);
        $affiliation->setAbbrev($affilShort);
        $affiliation->postToDB();
        $affilID = $affiliation->getID();
        if (!$affilID)
        {
            errorPage($PHP_SELF, "root",
        "Database Inaccessible",
        "Organization::postToDB unable to access database",TRUE); #TODO,
            exit;
        }
    }
    if ($affilID!=-1)
        $person->setAffilID($affilID);
    else
        $person->setOrgPays(FALSE);
}
$person->postToDB();
$primeID=$person->getID();
if (!$primeID)
{
    errorPage($PHP_SELF, "root","Database Inaccessible",
        "Person::PostToDB unable to access database",TRUE);
    exit; #TODO, make prettier
}
session_register("primeID");
session_register("includeEmail");
localRedirect("Location: $baseURL/partners.php"); #branch works
exit;
?>
```

SCRIPTS/prime-collision-action.php

```
<?php
#####################################################################
# This file is part of SlidingDoors                                 #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd        #
# SlidingDoors is availible for license under the GPL, see          #
# the COPYING file in the root directory of the install for         #
# the full terms of the GPL.                                        #
#                                                                   #
# File: prime-collision-action.inc                                  #
# Author: Eric D. Nielsen                                           #
# Description: An interstitial page that sets the proper            #
#              values based on the uses response to the close       #
#              match page.  Adds the new person/affil if            #
#              needed.  Dispatches to partners.php to start         #
#              Step Two.                                            #
# Internal Links: partners.php                                      #
# External Links: SlidingDoors information page (footer)            #
#                 Comp-in-a-box information page (footer)           #
#                 OpenImpetus information page (footer)             #
# Change Log: 6/12/01 -- created -- edn                             #
#             7/19/02 -- GPL, unified prime collision -- edn        #
#####################################################################

    session_start();
include "include_others.inc";
$oldPrime=$HTTP_POST_VARS["oldPrime"];
$newPrime=$HTTP_POST_VARS["primeID"];
if (""!=$oldPrime)
{
    $contraCheck = new ContraCheck($db);
    $mergeResult = $contraCheck->mayCombine($newPrime,$oldPrime);
    if (!$mergeResult->result)
    {
        $formVars = copyArray($HTTP_POST_VARS,"update");
        $formVars["UpdateMergeMsg"]="Your old identity and the one you
                selected are not combineable:".$mergeResult->getErrorMessage();
        $primeID = $oldPrime;
        session_register("formVars");
        session_register("primeID");
        localRedirect("Location: $baseURL/partners.php");
        exit;
    }
}

$formVars = copyArray($HTTP_POST_VARS);
$person = new Person($db);
if (""==$newPrime)
    $person->setID($HTTP_POST_VARS[$oldPrime]);
else
    $person->setID($HTTP_POST_VARS[$newPrime]);
$person->retrieve();
$person->populateFromFormVars($formVars);

if (in_array("Affiliation",$SD_elements))
{
    if ($formVars["AffilShort"]!="")
    {
        $affilObj = new RegOrg($db);
        $affilObj->setName($formVars["NewAffil"]);
```

```
            $affilObj->setAbbrev($formVars["AffilShort"]);
            $affilObj->setOrgPays($SD_default_org_pays);
            $affilObj->postToDB();
            $affilID = $affilObj->getID();
            if (!$affilID)
            {
                errorPage($PHP_SELF, "root","Database Inaccessible",
            "utilities::addAffiliation unable to access database",TRUE);
                exit;
            }

            if ($affilID!=-1)
                $person->setAffilID($affilID);
            else
                $person->setOrgPays(FALSE);
        }
    }
    $person->postToDB();
    $primeID = $person->getID();
    if (!$primeID)
    {
        errorPage($PHP_SELF, "root","Database Inaccessible",
            "utilities::addPerson unable to access database",TRUE);
        exit;
    }

    if (""!=$oldPrime && ""!=$newPrime)
    {
        $query = "UPDATE couples SET follower=$primeID WHERE follower=$oldPrime;";
        $db->query($query);
        $query = "UPDATE couples SET leader=$primeID WHERE leader=$oldPrime;";
        $db->query($query);
        $query = "DELETE FROM people where peopleid=$oldPrime;";
        $db->query($query);
    }
    session_register("primeID");
    unset($formVars);
    session_unregister("formVars");
    session_unregister("conflicts");
    localRedirect("Location: $baseURL/partners.php");
    exit;
?>
```

SCRIPTS/prime-email-collision-action.php

```
<?php
#####################################################################
# This file is part of SlidingDoors                                 #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd        #
# SlidingDoors is availible for license under the GPL, see          #
# the COPYING file in the root directory of the install for         #
# the full terms of the GPL.                                        #
#                                                                   #
# File: prime-email-collision-action.inc                            #
# Author: Eric D. Nielsen                                           #
# Description: An interstitial page that sets the proper            #
#              values based on the uses response to the close       #
#              match page. Dispatches to partners.php to start      #
#              Step Two.                                            #
# Internal Links: partners.php                                      #
# External Links: SlidingDoors information page (footer)            #
#                 Comp-in-a-box information page (footer)           #
#                 OpenImpetus information page (footer)             #
# Change Log: 6/12/01 -- created -- edn                             #
#             7/04/02 -- GPL -- edn                                 #
#####################################################################

    session_start();
include "include_others.inc";
$primeID = $HTTP_POST_VARS["CHOICE"];

if ($primeID != -1);
{
    session_register("primeID");
    localRedirect("Location: $baseURL/partners.php");
    exit;
}
errorPage($PHP_SELF,"Line 28","Lost Database Connectivity","While retrieving
$email's_record,_the_database_became_inaccessible.",TRUE);
exit;
?>
```

SCRIPTS/prime-retrieve.php

```
<?php
#####################################################################
# This file is part of SlidingDoors                                 #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd        #
# SlidingDoors is availible for license under the GPL, see          #
# the COPYING file in the root directory of the install for         #
# the full terms of the GPL.                                        #
#                                                                   #
# File: prime-retrieve.inc                                          #
# Author: Eric D. Nielsen                                           #
# Description: An interstitial page that sets the proper            #
#              values based on the uses response to the index       #
#              page.   Dispatches to partners.php to start          #
#              Step Two.                                            #
# Internal Links: partners.php                                      #
# External Links: SlidingDoors information page (footer)            #
#                 Comp-in-a-box information page (footer)           #
#                 OpenImpetus information page (footer)             #
# Change Log: 6/12/01 -- created -- edn                             #
#####################################################################

    session_start();

include "include_others.inc";
$person = new Person($db);

# populate the test Person to search the database based on the allowed
# search methods and the apparent format of the query
$search = $HTTP_POST_VARS["Search"];
if (in_array("Email",$SD_lookup_by) && $search!="")
{
    if (strpos($search,"@"))
        $person->setEmail(strtolower(htmlspecialchars($search)));
}
if (in_array("ID_number",$SD_lookup_by) && $search !="")
{
}
if (in_array("Name",$SD_lookup_by) && $search!="")
{
    $lastName = substr($search,strpos($search," "),strlen($search));
    $firstName = substr($search,0,strpos($search, " "));
    $person->setFirstName(htmlspecialchars($firstName));
    $person->setLastName(htmlspecialchars($lastName));
}
$primeID = $person->searchDB();

# if no matches, redirect to the regular page, keeping the search values
if (count($primeID)==0)
{
    unset($formVars);
    session_register("formVars");
    if (in_array("Email",$SD_lookup_by))
        $formVars["Email"]=$person->getEmail();
    if (in_array("ID_number",$SD_lookup_by))
    {
    }
    if (in_array("Name",$SD_lookup_by))
    {
        $formVars["FirstName"]=$person->getFirstName();
```

```
        $formVars["FirstName"]=$person->getLastname();
    }
    $formVars["GeneralMsg"] = "There are no entries in the database matching
the query you submitted.  Please return to <a href=\"$baseURL\">Registration
Home</a> to try a new search, or create a new entry by filling out this form.";
    session_register("formVars");
    localRedirect("Location: $baseURL/prime.php");
    exit;
}

# multiple_matches, save matches and redirect to collision page
if (count($primeID) != 1)
{
    session_register("primeID");
    localRedirect ("Location: $baseURL/prime-email-collision.php");
    exit;
}

# exactly one match, redirect to partners.php
$primeID=$primeID[0];
session_register("primeID");
localRedirect("Location: $baseURL/partners.php");
exit;
?>
```

SCRIPTS/prime-update-affil-action.php

```
<?php
#################################################################
# This file is part of SlidingDoors                             #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    #
# SlidingDoors is availible for license under the GPL, see      #
# the COPYING file in the root directory of the install for     #
# the full terms of the GPL.                                    #
#                                                               #
# File: prime-update-affil-action.inc                           #
# Author: Eric D. Nielsen                                       #
# Description: An interstitial page that handles the form data  #
#              passed from prime-update-affil-action.php.       #
# Internal Links: prime.php                                     #
#                 prime-collision.php                           #
#                 partners.php                                  #
# External Links: SlidingDoors information page (footer)        #
#                 Comp-in-a-box information page (footer)       #
#                 OpenImpetus information page (footer)         #
# Change Log: 8/10/01 -- created -- edn                         #
#             7/06/02 -- GPL, generic -- edn                    #
#################################################################
session_start();
include "include_others.inc";

$primeID = $HTML_SESSION_VARS["primeID"];
if (in_array("Name",$SD_elements))
{
    $formVars["FirstName"] = $HTTP_SESSION_VARS["formVars"]["updateFirstName"];
    $formVars["LastName"] = $HTTP_SESSION_VARS["formVars"]["updateLastName"];
}
if (in_array("Email",$SD_elements))
    $formVars["Email"] = $HTTP_SESSION_VARS["formVars"]["updateEmail"];
if (in_array("Fee Category",$SD_elements))
    $formVars["Age"] = $HTTP_SESSION_VARS["formVars"]["updateAge"];
if (in_array("Package",$SD_elements))
    $formVars["Package"] = $HTTP_POST_VARS["updatePackage"];

$newAffil = htmlspecialchars($HTTP_POST_VARS["affilName"]);
$affilShort = htmlspecialchars($HTTP_POST_VARS["affilShort"]);

$validated=TRUE;

if ($newAffil!="")
    $formVars["updateAffilName"]=$newAffil;
else
{
    $formVars["updateAffilNameMsg"]="<STRONG>Please do not leave this
field blank.</STRONG><BR>\n";
    $validated=FALSE;
}
if ($affilShort!="")
    $formVars["updateAffilShort"]=$affilShort;
else
{
    $formVars["updateAffilShortMsg"]="<STRONG>Please do not leave this
field blank.</STRONG><BR>\n";
    $validated=FALSE;
}
```

```
if (!$validated)
{
    session_register("primeID");
    session_register("formVars");
    localRedirect("Location: $baseURL/prime-update-affil.php");
    exit;
}
$org = new RegOrg($db);
$org->setName($newAffil);
$org->setAbbrev($affilShort);
$org->setOrgPays($SD_default_org_pays);
$org->postToDB();
$affilID= $org->getID();
if (!$affilID)
{
    errorPage($PHP_SELF, "root","Database Inaccessible",
        "Organization::PostToDB unable to access database",TRUE);
    exit;
}
$formVars["updateAffilID"]=$affilID;
unset($formVars["updateAffilShort"]);
unset($formVars["updateAffilName"]);


$person = new Person($db);
$person->setID($primeID);
$person->retrieve();
$person->populateFromFormVars($formVars);
$person->setAffilID($affilID);

$conflicts = $db->personConflicts($person);
$conflicts = arrayRemove($conflicts,$primeID);
if ($conflicts = $db->personConflicts($person))
{
    $formVars["Affil"]=$affilID;
    session_register("primeID");
    session_register("conflicts");
    session_register("formVars");
    localRedirect("Location: $baseURL/prime-update-collision.php");
    exit;
}

$postResult = $person->postToDB();

if (!$postResult)
{
    errorPage($PHP_SELF, "root","Database Inaccessible",
        "Person::PostToDB unable to access database",TRUE);
    exit;
}
unset($formVars);
session_unregister("formVars");
session_register("primeID");
localRedirect("Location: $baseURL/partners.php");
exit;
?>
```

SCRIPTS/prime-update.php

```
<?php
#################################################################
# This file is part of SlidingDoors                             #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    #
# SlidingDoors is availible for license under the GPL, see      #
# the COPYING file in the root directory of the install for     #
# the full terms of the GPL.                                    #
#                                                               #
# File: prime-update.inc                                        #
# Author: Eric D. Nielsen                                       #
# Description: An interstitial page that handles the form data  #
#              passed from partners.php.  This page will        #
#              verify the update data request for an individual #
#              on a successful update an email is set to the    #
#              old email address.                               #
# Internal Links: prime.php                                     #
#                 prime-collision.php                           #
#                 partners.php                                  #
# External Links: SlidingDoors information page (footer)        #
#                 Comp-in-a-box information page (footer)       #
#                 OpenImpetus information page (footer)         #
# Change Log: 6/12/01 -- created -- edn                         #
#             7/06/02 -- GPL, generic form -- edn               #
#################################################################
session_start();
include "include_others.inc";
$primeID = $HTTP_POST_VARS["primeID"];
$contraCheck = new ContraCheck($db);
unset($formVars);
$formVars=array();
$person = new Person($db);

$validated = $person->validateInputs(&$formVars,"PRIME-UPDATE");
if (!$validated)
{
    session_register("formVars");
    session_register("primeID");
    localRedirect("Location: $baseURL/partners.php");
    exit;
}

if (in_array("Affiliation",$SD_elements))
    $affilID=$HTTP_POST_VARS["updateAffilID"];
    if ($affilID=="New")
    {
        session_register("primeID");
        session_register("formVars");
        localRedirect("Location: $baseURL/prime-update-affil.php");
        exit;
    }
if (in_array("Package",$SD_elements))
{
        $package = $HTTP_POST_VARS["updatePackage"];
    $compRulesConflicts = $contraCheck->checkPersonPackageBothRoles($primeID,
                                                    "",$primeID,$package);
    if (!$compRulesConflicts->result)
    {
        # setup variables
        $formVars["UpdateEventsMsg"]=$compRulesConflicts->getErrorMessage();
```

```
        $formVars["UpdatePackage"]=$person->getPackage();
        session_register("primeID");
        session_register("formVars");
        localRedirect("Location: $baseURL/partners.php");
        exit;
    }
}

$person = new Person($db);
$person->setID($primeID);
$person->retrieve();
$person->populateFromFormVars($formVars,"update");

$conflicts = $db->personConflicts($person);
$conflicts = arrayRemove($conflicts,$primeID);
if (count($conflicts))
{
    session_register("primeID");
    session_register("conflicts");
    session_register("formVars");
    localRedirect("Location: $baseURL/prime-update-collision.php");
    exit();
}

$postResult = $person->postToDB();
if (!$postResult)
{
    errorPage($PHP_SELF, "root","Database Inaccessible",
        "Person::PostToDB unable to access database",TRUE);
    exit;
}
#session_register("person");
session_register("primeID");
localRedirect("Location: $baseURL/partners.php");
exit;
?>
```

C.5 SlidingDoors Admin Area

admin/change-status.php

```
<?php
include "pre_main_include.inc";
$authenticationRequired=TRUE;
$authorizationLevels[]=array("Type"=>"Comp","Level"=>"Registrar",
                             "CompName"=>getCompNameFromURL($PHP_SELF,
                                                            "register"));
include "include_others.inc";
$cib_db = $SECURITY_CODE_DB;

// If the user has submitted the page, update all the entries
if (isset($HTTP_POST_VARS["submit"]))
{
  $statusID = $HTTP_POST_VARS["statusID"];
  if (is_numeric($statusID))
    {
      $status = $cib_db->getCompStatusFromStatusID($statusID);
      if ($status!="")
          $db->setStatus($status);
    }
}

$display = new HTMLDisplay($db);
$page = $display->beginPage();
$page.= $display->InstructionBox("Admin: Competition Status",
"From this page you can change registration from testing to open to close and
back to open as desired.  At the present changing from testing to open does not
purge the database of testing entries/affiliations.  You will have to manually
remove these via the other admin interfances.
pages and as additional instructions in the boxes on the index page.
<p />When satisified
with your updates return to the <a href=\"$secureURL/admin/\">Main Admin
Page</a>.");
$page .= $display->sectionBreak();

$form = "";
$form .= "<form action=\"$secureURL/admin/change-status.php\" method=\"POST\">\n";
$status = $db->getStatus();
$form .= "Your competition is currently <b>$status</b>.";
$form .= "You wish to change it to:";
$form .= $display->arrayToSelect("statusID",
                                 $cib_db->getCompStatusChoices(),
                                 $cib_db->getCompStatusIDFromStatus($status),
                                 1,1);
$form .= "\n<tr><td colspan=\"2\"><input type=\"submit\" name=\"submit\"
value=\"Change Status\" style=\"$SD_button_style\"></td></tr>\n";
$form .= "</form>\n";

$page .= $display->textBox("$form");
$page .= $display->sectionBreak();
$page .= $display->compinaboxBox();
$page .= $display->endPage();
echo $page;
?>
```

admin/index.php

```
<?php
/*****************************************************************
 * This file is part of SlidingDoors                             *
 * Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    *
 * SlidingDoors is availible for license under the GPL, see      *
 * the COPYING file in the root directory of the install for     *
 * the full terms of the GPL.                                    *
 *                                                               *
 * File: admin.php                                               *
 * Author: Eric D. Nielsen                                       *
 * Description: The "Admin Page" for on-line registration for    *
 *              the 2002 MITBDT Open DanceSport Competition.     *
 * Internal Links:                                               *
 * External Links: 2002 MITBDT Open DanceSport Competition page *
 *                 SlidingDoors information page                 *
 *                 Comp-in-a-box information page                *
 *                 OpenImpetus information page                  *
 * Change Log: 6/12/01 -- created -- edn                         *
 *****************************************************************/
include "pre_main_include.inc";
$authenicationRequired=TRUE;
$authorizationLevels[]=array("Type"=>"Comp","Level"=>"Staff",
                             "CompName"=>getCompNameFromURL($PHP_SELF,
                                                            "register"));

include "include_others.inc";
$status = $db->getStatus();
$display = new HTMLDisplay($db);
$page = $display->beginPage("Admin: $SD_compname");
$page .= $display->sectionBreak();
$page .= $display->instructionBox("Administration Pages",
"Welcome to the administration pages. Please be very careful here. There is
less error checking here than in the regular interface, because its possible
that you, as an Administrator, will have to do unusual things. Really \"
dangerous\" options are double checked, but its possible to do a lot of harm
with even regular operations.<p />
If you leave the administration pages, you will have to login again when you
return as the login information is not propagated on the regular
registration pages or on the stats pages.");
$page .= $display->sectionBreak();
$page .= "<table>\n";
$page .= "\t<tr valign=\"top\">\n\t\t<td width=\"50%\">\n";

$host = $db->getHost();
$totalDue = $db->getTotalCompFees();
$totalWithoutHost = $db->getTotalCompFees(FALSE,$host);
$totalPaid = $db->getTotalCompFees(TRUE);
$totalPaidWithoutHost =$db->getTotalCompFees(TRUE,$host);
$totalOutstanding = $totalDue-$totalPaid;
$totalOutstandingWithoutHost = $totalWithoutHost - $totalPaidWithoutHost;
$totalString = "\$$totalDue" . ($host ? "
               (\$$totalWithoutHost excluding host)" :"");
$totalPaidString = "\$$totalPaid" . ($host ? "
               (\$$totalPaidWithoutHost excluding host)" :"");
$totalOutstandingString = "\$$totalOutstanding" . ($host ? "
               (\$$totalOutstandingWithoutHost excluding host)" :"");

$page .= $display->textBox("<h3>Overview</h3>
Registration is $status. <a href=\"$secureURL/admin/change-status.php\">
[Change Status]</a> <p />
```

```
<center>
<table border=\"1\">
<tr><th>Registration Fees</th><th>Registration Options</th></tr>
<tr valign=\"top\"><td>
<table>
<tr><td>Fees Received</td><td>$totalPaidString</td></tr>
<tr><td>Fees Outstanding</td><td>$totalOutstandingString</td></tr>
<tr><th>Current Total</td><td>$totalString</td></tr></table>
<a href=\"$secureURL/admin/update-paid.php\">[Update Paid Fees]</a>
</td><td>
<a href=\"$secureURL/admin/incomplete-reg.php\">
[Detect Incomplete Registrations]</a><br /><br />
<a href=\"$secureURL/admin/lookup.php\">[Search/Edit Competitors]</a><br /><br />
<a href=\"$secureURL/admin/assign-group-password.php\">[Assign Group Password]</a>
</td></tr></table></center>");

$page .= "</td><td>";
$page .= $display->textBox("<h3>Messages for Competitors</h3>
<a href=\"$secureURL/admin/massmail.php\">[E-mail Competitors/Captions]</a><br />
<a href=\"$secureURL/admin/motd.php\">[Update Message-Of-The-Day]</a><br />
<a href=\"$secureURL/admin/template.php\">[Update Email Templates]</a>");
$page .= "</td></tr><tr valign=\"top\"><td>";

$page .= $display->textBox("<h3>Information Export</h3>
The information export page allows you to export competition information in a
variety of formats: LaTeX, PDF, XML, raw text.  There are specific options to
generate lists for the head table, registration table, program listings as
well as preloading event data into scrutineering tools.
<a href=\"$secureURL/admin/information-export.php\">[Export Information]</a>");
$page .= "</td><td>";
$page .= $display->textBox("<h3>Un-categorized Options</h3>
<a href=\"$secureURL/admin/assign-numbers.php\">[Assign Competitor Numbers]</a>
<br />
<a href=\"$secureURL/admin/edit-administrator.php\">
[Add/Remove Administrator]</a><br />
<a href=\"$secureURL/admin/edit-self.php\">[Update Your Admin Account]</a>");
$page .= "</td></tr></table>";
$page .= $display->sectionBreak();
$page .= $display->compinaboxBox();
$page .= $display->endPage();
echo $page;
?>
```

admin/motd.php

```
<?php
include "pre_main_include.inc";
$authenticationRequired=TRUE;
$authorizationLevels[]=array("Type"=>"Comp","Level"=>"Registrar",
                             "CompName"=>getCompNameFromURL($PHP_SELF,
                                                            "register"));

include "include_others.inc";

$motds = array("Main","Register","Stats","Modify","Group");

// If the user has submitted the page, update all the entries
if (isset($HTTP_POST_VARS["submit"]))
{
  foreach ($motds as $type)
    {
      $motd = $HTTP_POST_VARS[$type . "Msg"];
      $motd = sanitizeXHTML($motd,array("b","B","a","A"));
      $db->setMOTD($motd,$type);
    }
}

$display = new HTMLDisplay($db);
$page = $display->beginPage();
$page.= $display->InstructionBox("Admin: Message Of The Day",
"On this page you can update the messages displayed on the main registration
pages and as additional instructions in the boxes on the index page.  This page
submits back to itself to allow you to confirm your changes.  When satisified
with your updates return to the <a href=\"$secureURL/admin/\">Main Admin
Page</a>.<p />You are allowed to use &lt;b>Text to Bold&lt;/b>,
&lt;a href=\"link\">Text To Link&lt;/a> tags.  All other HTML
will be eliminated.  All tags must be paired, ie &lt;b>stuff&lt;/b>.  If
you leave off the closing tag, the opening tag will be stripped.");
$page .= $display->sectionBreak();

$form = "<table>\n";
$form .= "<form action=\"$secureURL/admin/motd.php\" method=\"POST\">\n";
foreach ($motds as $type)
  {
    $motd = $db->getMOTD($type);
    //    $motd = wrapword($motd,69); #:TODO: upgrade my php :)
    $form .= "\t<tr valign=\"center\">\n";
    $form .= "\t\t<td >$type Message</td>\n";
    $form .= "\t\t<td><textarea name=\"".$type."Msg"."\" cols=\"80\" ".
        "rows=\"4\">$motd</textarea></td></tr>\n";
  }
$form .= "\n<tr><td colspan=\"2\"><input type=\"submit\" name=\"submit\" ".
        "value=\"Update MOTDs\" style=\"$SD_button_style\"></td></tr>\n";
$form .= "</form></table>\n";

$page .= $display->textBox("$form");
$page .= $display->sectionBreak();
$page .= $display->compinaboxBox();
$page .= $display->endPage();
echo $page;
?>
```

C.6 SlidingDoors Group Area

group/group-add-person.php

```
<?php
#################################################################
# This file is part of SlidingDoors                             #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    #
# SlidingDoors is availible for license under the GPL, see      #
# the COPYING file in the root directory of the install for     #
# the full terms of the GPL.                                    #
#                                                               #
# File: group-add-person.php                                    #
# Author: Eric D. Nielsen                                       #
# Description: Adds a person to an organization, typically      #
#              used for non-competitors or during group setup   #
# Internal Links: group-add-person-action.php for handling      #
# External Links: SlidingDoors information page (footer)        #
#                 Comp-in-a-box information page(footer)        #
#                 OpenImpetus information page (footer)         #
# Change Log: 11/28/01 -- created -- edn                        #
#################################################################
# Do not remove the following line
include "include_others.inc";
session_start();


$peopleToAdd = $HTTP_SESSION_VARS["peopleToAdd"];
$orgID = $HTTP_SESSION_VARS["orgID"];
$addType = $HTTP_SESSION_VARS["addType"];
$display = new HTMLDisplay($db);
$page = $display->beginPage();
$page .= $display->instructionBox("Member Creation",
"This page at present does not do any error checking, please be sure
you already checked the list of people belonging to
your affiliation before adding someone as duplicated will not be caught");
$page .= $display->sectionBreak();

$form = "<form action=\"SCRIPTS/group-add-person-action.php\" method=\"POST\">\n";
$form .= "<table>\n";
$form .= "<input type=\"hidden\" name=\"orgID\" value=\"$orgID\">\n";
$form .= "<input type=\"hidden\" name=\"addType\" value=\"$addType\">\n";
reset($peopleToAdd);
while($aPerson = each($peopleToAdd))
{
    if ($aPerson["value"]["type"]=="REP")
    {
        $labelSuffix = "Rep";
        $labelPrefix = "Representative's ";
    }
    else
    {
        $index = $aPerson["value"]["index"];
        $displayIndex=$index+1;
        $labelSuffix="Coord".$index;
        $ordinalSuffix = ($displayIndex==1?"st":
                          ($displayIndex==2?"nd":"th"));
        $labelPrefix = $displayIndex . $ordinalSuffix . " Coordinator's ";
        $emailsendvar = "CoordEmailSend".$index;
```

```
        $digestvar = "CoordDigest".$index;
        $$emailsendvar = $aPerson["value"]["email"];
        $$digestvar = $aPerson["value"]["digest"];
    }
    $form .= "<tr><th colspan=2>$labelPrefix Information</th></tr>\n";
    $fnamevar = "fName" . $labelSuffix;
    $lnamevar = "lName" . $labelSuffix;
    $emailvar = "email" . $labelSuffix;
#    $phonevar = "phone" . $labelSuffix;
    if ($labelSuffix=="Rep")
    {
        $addressID = $aPerson["value"]["addrID"];
        if ($addressID!="")
        {
            $form.= "<input type=\"hidden\" name=\"addressRep\" value=\"$addressID\">\n";
        }
        if (isset($$emailsendvar))
            $form .= "<input type=\"hidden\" name=\"$emailsendvar\" value=\"$$emailsendvar\">\n";
        if (isset($$digestvar))
            $form .= "<input type=\"hidden\" name=\"$digestvar\" value=\"$$digestvar\">\n";


    }
    $form.= $display->displayFormElement("FirstName".$labelSuffix,$$fnamevar,1,"First Name");
    $form.= $display->displayFormElement("LastName".$labelSuffix,$$lnamevar,2,"Last Name");
    $form .= $display->displayFormElement("Email".$labelSuffix,$$emailvar,3,"Email");
#    echo $display->displayFormElement("Phone".$labelSuffix,$$phoneVar,4,"Phone Number");
}
$form .= "<tr><td><input type=\"submit\" value=\"Submit\" name=\"submit\"></td></tr>\n";
$form.= "</table>\n";
$form .= "</form>";
$page .= $display->textBox($form,"95%","This is the add person form");
$page .= $display->sectionBreak();
$page .= $display->compinaboxBox();
$page .= $display->endPage();
clean_up_session();
echo $page;
?>
```

group/group-setup.php

```
<?php
/*****************************************************************
 * This file is part of SlidingDoors                             *
 * Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    *
 * SlidingDoors is availible for license under the GPL, see      *
 * the COPYING file in the root directory of the install for     *
 * the full terms of the GPL.                                    *
 *                                                               *
 * File: group-setup.php                                         *
 * Author: Eric D. Nielsen                                       *
 * Description: This page deals with the creation of a new       *
 *              affiliation or the updating of data for an       *
 *              existing affiliation.  It also provides link     *
 *              to the team match registration pages.            *
 * Internal Links: no spec'd out yet                             *
 * External Links: SlidingDoors information page (footer)        *
 *                 Comp-in-a-box information page(footer)        *
 *                 OpenImpetus information page  (footer)        *
 * Change Log: 11/06/01 -- created -- edn                        *
 *****************************************************************/
# Do not remove the following line
include "pre_main_include.inc";
$authenticationRequired=TRUE;
$authorizationLevels[]=array("Type"=>"Team",
                             "Level"=>"Registration Coordinator",
                             "TeamID"=>$HTTP_SESSION_VARS["affil"]);
include "include_others.inc";
session_start();

$org = $HTTP_POST_VARS["affil"];
if ($org=="")
    $org = $HTTP_SESSION_VARS["affil"];
else
    localRedirect("$baseURL/back.php");

if (!$HTTP_SESSION_VARS["loggedin"])
    if (isset($HTTP_POST_VARS["pass"]))
    {
        $pass=$HTTP_POST_VARS["pass"];
        $query = "SELECT hashed_passwd FROM registration_supplement WHERE orgID=$org;";
        $result = $db->query($query);
        if ($result->numrows()!=0)
        {
            list($hashed) = $result->getRowAt(0);
            $test = crypt($pass,$hashed);
            if ($test==$hashed)
                $loggedin=TRUE;
            else
                $loggedin=FALSE;
        }
        else
            $loggedin=FALSE;
    } else $loggedin=FALSE;
else $loggedin=TRUE;

session_register("loggedin");
if (!$loggedin)
{
    session_register("org");
```

```
    localRedirect("Location: $baseURL/group/index.php");
    exit;
}
$orgObj = new RegOrg($db,0);
$orgObj->setID($org);

if (isset($HTTP_SESSION_VARS["formVars"]))
{
    $orgObj->setName($formVars["TeamName"]);
    $orgObj->setAbbrev($formVars["TeamAbbrev"]);
    $orgObj->setRepTitle($formVars["RepTitle"]);
    $orgObj->setRep($formVars["Rep"]);

    $tempAddr = new Address($db);
    $tempAddr->setStreet1($formVars["Street1"]);
    $tempAddr->setStreet1($formVars["Street2"]);
    $tempAddr->setStreet1($formVars["City"]);
    $tempAddr->setStreet1($formVars["State"]);
    $tempAddr->setStreet1($formVars["Zip"]);
}
else
    $orgObj->retrieve();
$display = new HTMLDisplay($db);
$page = $display->beginPage();
$text = "";
$navbar = "<center>".$display->teamNavBar(array("Top","Setup","Invoice",
"Results"),"setup")."</center><br>";
$text .= $navbar;
$text.= $display->teamInfoUpdateBar($orgObj);
$page .= $display->instructionBox(
                "<a name=\"top\"><H2>Affiliation Summary</h2></a>\n",$text);
$teamMatchText ="<center>".$display->teamNavBar(array("Top","Setup",
"Invoice","Results"),"teammatch")."</center><br>";
$teamMatchText .= $display->teamMatchDetails($orgObj);
$page .= $display->textBox($teamMatchText,"95%","Team Match Box");

$registrationBox ="";
$registrationBox.= "<center>".$display->teamNavBar(array("Top","Setup",
"Invoice","Results"),"invoice")."</center><br>";
$registrationBox.= $display->teamRegistration($orgObj,TRUE);
$page .= $display->textBox($registrationText,"95%","Registration Box");

$resultsBox ="";
$resultsBox.= "<center>".$display->teamNavBar(array("Top","Setup","Invoice",
"Results"),"results")."</center><br>";
$resultsBox .=  $display->teamResults($orgObj);
$page .= $display->textBox($resultsBox,"95%","Results Box");
$page .= $display->sectionBreak();
$page .= $display->compinaboxBox();
$page .= $display->endPage();
clean_up_session();
session_register("loggedin");
$affil=$org;
session_register("affil");
echo $page;
?>
```

group/index.php

```
<?php
#################################################################
# This file is part of SlidingDoors                             #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd    #
# SlidingDoors is availible for license under the GPL, see      #
# the COPYING file in the root directory of the install for     #
# the full terms of the GPL.                                    #
#                                                               #
# File: group.php                                               #
# Author: Eric D. Nielsen                                       #
# Description: This page allows a team to retrieve their        #
#              current registration summary, invoice, and       #
#              competition results after the comp.              #
# Internal Links: group-check.php                               #
# External Links: SlidingDoors information page (footer)        #
#                 Comp-in-a-box information page(footer)        #
#                 OpenImpetus information page  (footer)        #
# Change Log: 11/06/01 -- created -- edn                        #
#################################################################
# Do not remove the following line
include "include_others.inc";
session_start();
unset($formVars);
session_unregister($formVars);
unset($addType);
session_unregister($addType);
$org = $HTTP_POST_VARS["Affil"];
if ($org=="") $org=$HTTP_SESSION_VARS["org"];
if ($org=="NewAffil")
{
#       errorPage("index.php","selecting NewAffil","Feature Disabled",
#"This feature has been disabled temporarily due to myriads of bugs.  This is
#not a true bug, please do not email us (in spite of what the footer below
#says.",FALSE);
        $affil=0;
        session_register("affil");
        localRedirect("Location: $baseURL/group/group-setup.php");
        exit;
}
$display = new HTMLDisplay($db);
$page = $display->beginPage();
$page.= $display->instructionBox("<a name=\"top\">Affiliation Summary</a>",
                                 "This page allows captains/studio
representative and other similar affiliation organizers to quickly review the
status of their members'_registrations.__Registered,_team_authorized
individuals_may_update_most_of_the_information_here_for_their_team.
<p>
Relevant_Links:
<ul>
<li><a_href=\"group-setup.php\">Update_Information</a></li>
<li><a_href=\"$CIB_SECURE_URL/new_account.php\">Create_a_CompInaBox_Account</a></li>
<li><a_href=\"$CIB_SECURE_URL/profile_management.php\">Request_Team_Authorization</a></li></ul>
");

$affil_=_new_RegOrg($db,0);
$affil->setID($org);
$affil->retrieve();
$box_=_"<center>".$display->teamNavBar(array("Top","Login","TeamMatch",
"Invoice","Results"),"login")."</center><br>";
```

```
$box .= $display->teamInfoLoginBar($affil);
$page.= $display->textBox($box,"95%","Org Login in Box");
$text = "<center>".$display->teamNavBar(array("Top","Login","TeamMatch",
"Invoice","Results"),"teammatch")."</center><br>";
$text.= $display->teamMatchSummary($affil);
$page .= $display->textBox($text);

$text="<center>".$display->teamNavBar(array("Top","Login","TeamMatch",
"Invoice","Results"),"invoice")."</center><br>";
$text.= $display->teamRegistration($affil);
$page .= $display->textBox($text);

$text = "<center>".$display->teamNavBar(array("Top","Login","TeamMatch",
"Invoice","Results"),"results")."</center><br>";
$text .= $display->teamResults($affil);
$page .= $display->textBox($text);
$page .= $display->sectionBreak();
$page .= $display->compinaboxBox();
$page .= $display->endPage();
clean_up_session();
session_register("affil");
echo $page;
?>
```

group/SCRIPTS/group-add-person-action.php

```
<?php
#####################################################################
# This file is part of SlidingDoors                                 #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd        #
# SlidingDoors is availible for license under the GPL, see          #
# the COPYING file in the root directory of the install for         #
# the full terms of the GPL.                                        #
#                                                                   #
# File: group-add-person.php                                        #
# Author: Eric D. Nielsen                                           #
# Description: Adds a person to an organization, typically          #
#              used for non-competitors or during group setup       #
# Internal Links: group-add-person-action.php for handling          #
# External Links: SlidingDoors information page (footer)            #
#                 Comp-in-a-box information page(footer)            #
#                 OpenImpetus information page  (footer)            #
# Change Log: 11/28/01 -- created -- edn                            #
#####################################################################
# Do not remove the following line
include "inlclude_others.inc";
session_start();

$orgID = $HTTP_POST_VARS["orgID"];
$coordNum = array();
reset($HTTP_POST_VARS);
echo printArray($HTTP_POST_VARS);
while($aVar = each($HTTP_POST_VARS))
{
    $name = $aVar["key"];
    $isRep = strpos($name,"Rep");
    $isCoord = strpos($name,"Coord");
    if ($isCoord)
    {
        $lastChar = strrpos($name,"d");
        $digits = substr($name,$lastChar+1,strlen($name));
        if (!in_array($digits,$coordNum))
            array_push($coordNum,$digits);
    }
    if ($isRep || $isCoord)
    {
        $$name = $aVar["value"];
    }
}
if (isset($FirstNameRep))
{
    $person = new Person($db);
    $person->setFirstName($FirstNameRep);
    $person->setLastName($LastNameRep);
    $person->setEmail($EmailRep);
#    $person->setPhone($PhoneRep);
    $person->setAffilID($orgID);
    if (isset($addresRep) && $addressRep!="")
        $person->setAddress($HTTP_SESSION_VARS["addressRep"]);
    $person->postToDB();
    $repID = $person->getID();
}

$coordID=array();
$coordIndex=0;
```

```
reset($coordNum);
while($aCoord= each($coordNum))
{
     $person = new Person($db);
     $firstNameVar = "FirstNameCoord".$aCoord["value"];
#         echo "$firstNameVar = ".$$firstNameVar;
     $person->setFirstName($$firstNameVar);
     $lastNameVar = "LastNameCoord".$aCoord["value"];
     $person->setLastName($$lastNameVar);
     $emailVar = "EmailCoord".$aCoord["value"];
     $person->setEmail($$emailVar);
#     $phoneVar = "PhoneCoord".$aCoord["value"];
#     $person->setPhone($$phoneVar);
     $person->setAffilID($orgID);
     $person->postToDB();
     $coordID[$coordIndex++]=$person->getID();
}

$regOrg = new RegOrg($db);
$regOrg->setID($orgID);
$regOrg->retrieve();
$regOrg->setRep($repID);

$numCoords = count($coordID);
for ($i=0;$i<$numCoords;$i++)
{
     $emailVar = "CoordEmailSend".$index;
     $digestVar = "CoordDigest".$index;
     $regOrg->addCoordinator($coordID[$i],$emailVar,$digestVar);
#     echo "I'm here now.";
}
#echo "<br>".printArray($regOrg)."<br>";
$regOrg->postToDB();

session_register("orgID");
localRedirect("Location: $baseURL/group/group-setup.php");
exit;
?>
```

group/SCRIPTS/group-reg-update.php

```
<?php
#####################################################################
# This file is part of SlidingDoors                                 #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd        #
# SlidingDoors is availible for license under the GPL, see          #
# the COPYING file in the root directory of the install for         #
# the full terms of the GPL.                                        #
#                                                                   #
# File: group-reg-update.php                                        #
# Author: Eric D. Nielsen                                           #
# Description: Updates the org_pays fields of members               #
# Internal Links: redirects back to setup                           #
# External Links: SlidingDoors information page (footer)            #
#                 Comp-in-a-box information page(footer)            #
#                 OpenImpetus information page  (footer)            #
# Change Log: 11/06/01 -- created -- edn                            #
#####################################################################
# Do not remove the following line
include "include_others.inc";
session_start();

$affil = $HTTP_POST_VARS["affil"];
session_register("affil");

$form = $HTTP_POST_VARS;

reset($form);
while ($anElement = each($form))
{
    $label = $anElement["key"];
    $value = $anElement["value"];
    if ($label=="affil") continue;
    if ($label=="Submit") continue;
    $id = substr($label,strrpos($label,'-')+1,strlen($label));
    $update=(substr($value,0,3)!="not"?"TRUE":"FALSE");
    $query = "SELECT * FROM people_paid WHERE peopleid=$id;";
    $result = $db->query($query);
    if ($result->numRows())
    {
        $query = "UPDATE people_paid SET org_pays=$update WHERE peopleid=$id;";
    }
    else
    {
        $query = "INSERT into people_paid VALUES ($id,$update,FALSE);";
    }
    $result=$db->query($query);
}
$loggedin=$HTTP_SESSION_VARS["loggedin"];
session_register("loggedin");
localRedirect("Location: $baseURL/group/group-setup.php#invoice");
exit;
?>
```

group/SCRIPTS/group-info-update.php

```
<?php
#####################################################################
# This file is part of SlidingDoors                                 #
# Copyright 2001-2002. Eric D. Nielsen, All rights reserverd        #
# SlidingDoors is availible for license under the GPL, see          #
# the COPYING file in the root directory of the install for         #
# the full terms of the GPL.                                        #
#                                                                   #
# File: group-info-update.php                                       #
# Author: Eric D. Nielsen                                           #
# Description: Updates the general affiliation information          #
# Internal Links: redirects back to setup                           #
# External Links: SlidingDoors information page (footer)            #
#                 Comp-in-a-box information page(footer)            #
#                 OpenImpetus information page  (footer)            #
# Change Log: 11/06/01 -- created -- edn                            #
#####################################################################
# Do not remove the following line
include "include_others.inc";
session_start();
unset($formVars);
$type = $HTTP_POST_VARS["type"];
$orgID = $HTTP_POST_VARS["orgid"];
$affil=$orgID;
session_register("affil");
$teamname = htmlspecialchars($HTTP_POST_VARS["teamname"]);
$teamabbrev = htmlspecialchars($HTTP_POST_VARS["teamabbrev"]);
$reptitle = htmlspecialchars($HTTP_POST_VARS["reptitle"]);
$rep = $HTTP_POST_VARS["rep"];
$street1 = htmlspecialchars($HTTP_POST_VARS["street1"]);
$street2 = htmlspecialchars($HTTP_POST_VARS["street2"]);
$city = htmlspecialchars($HTTP_POST_VARS["city"]);
$state = htmlspecialchars($HTTP_POST_VARS["state"]);
$zip = htmlspecialchars($HTTP_POST_VARS["zip"]);
$addrType=$HTTP_POST_VARS["addrType"];
#$coords = array();
#$email = array();
#$digest = array();
#$index=0;
#$moreCoords=isset($HTTP_POST_VARS["coord-$index"]);
#while ($moreCoords)
#{
#    array_push($coords,$HTTP_POST_VARS["coord-$index"]);
#    array_push($email,$HTTP_POST_VARS["email-$index"]);
#    array_push($digest,$HTTP_POST_VARS["digest-$index"]);
#    $index++;
#    if (!isset($HTTP_POST_VARS["coord-$index"]))
#       $moreCoords = FALSE;
#}
#$formVars["Coords"]=$coords;
#$formVars["Email"]=$email;
#$formVars["Digest"]=$digest;
$formVars["Rep"]=$rep;
$formVars["OrgID"]=$orgID;
$formVars["Type"]=$type;
$formVars["AddrType"]=$addrType;

$validated=TRUE;
```

```
formValidate("TeamName",$teamname,
        "<STRONG>Please enter a full name for the affiliation</strong>");
formValidate("TeamAbbrev",$teamabbrev,
        "<STRONG>Please enter an abbreviation for the affiliation</strong>");
formValidate("RepTitle",$reptitle,
        "<STRONG>Please enter the title of the affiliation's representative</strong>");
formValidate("Street1",$street1,
        "<STRONG>Please enter the first line of the mailing address.</strong>",FALSE);
formValidate("City",$city,
        "<STRONG>Please enter the city for the affiliation's address.</strong>",FALSE);
formValidate("State",$state,
        "<strong>Please enter the state.</strong>",FALSE);
formValidate("Zip",$zip,
        "<strong>Please enter the zip code.</strong>",FALSE);

if (!$validated)
{
    session_register("formVars");
    localRedirect("Location: $baseURL/group/group-setup.php");
    exit;
}
if ($type=="INSERT")
{
    $regOrg = new RegOrg($db);
    $regOrg->setName($teamname);
    $regOrg->setAbbrev($teamabbrev);
    $regOrg->setRepTitle($reptitle);
    if ($rep)
        $regOrg->setRep($rep);

    $address= new Address($db);
    $address->setStreet1($street1);
    $address->setStreet2($street2);
    $address->setCity($city);
    $address->setState($state);
    $address->setZip($zip);
    $addressID=$address->searchDB();
    $addressAction = (count($addressID)?"UPDATE":"INSERT");
    if ($addressAction=="INSERT")
    {
        $address->postToDB();
        $addressID=$address->getID();
    }
    else
    {
        if (count($addressID)!=1)
        {
            die("Need to handle multiple address matches for the rep....");
        }
        else
            $addressID=$addressID[0];
    }
    if ($addrType=="business")
    {
        $regOrg->setBuilding($addressID);
    }
    else if($rep!=0)
    {
        $query  = "UPDATE people SET address=$addressID WHERE peopleid=$rep;";
        $db->query($query);
    }
```

```
#     $numCoords = count($coords);
#     for ($i=0;$i<$numCoords;$i++)
#         if ($coords[$i]>=1)
#         $regOrg->addCoordinator($coords[$i],$email[$i],$digest[$i]);
    $regOrg->postToDB();
    $orgID= $regOrg->getID();
}
else
{
    $regOrg = new RegOrg($db);
    $regOrg->setID($orgID);
    $regOrg->retrieve();
    $regOrg->setName($teamname);
    $regOrg->setAbbrev($teamabbrev);
    $regOrg->setRepTitle($reptitle);
    if ($rep)
        $regOrg->setRep($rep);

    $address= new Address($db);
    $address->setStreet1($street1);
    $address->setStreet2($street2);
    $address->setCity($city);
    $address->setState($state);
    $address->setZip($zip);
    $addressID=$address->searchDB();
    $addressAction = (count($addressID)?"UPDATE":"INSERT");
    if ($addressAction=="INSERT")
    {
        $address->postToDB();
        $addressID=$address->getID();
    }
    else
    {
        if (count($addressID)!=1)
        {
            die("Need to handle multiple address matches for the rep....");
        }
        else
            $addressID=$addressID[0];
    }
    if ($addrType=="business")
    {
        $regOrg->setBuilding($addressID);
    }
    else if($rep!=0)
    {
        $query  = "UPDATE people SET address=$addressID WHERE peopleid=$rep;";
        $db->query($query);
    }
    $regOrg->postToDB();
}


$peopleToAdd=array();
if (!$rep)
{
    if ($addrType=="business")
        array_push($peopleToAdd,
                   array("type"=>"REP", "orgID"=>$orgID));
    else
```

```
        array_push($peopleToAdd,
                    array("type"=>"REP", "orgID"=>$orgID,
                        "addrID"=>$addressID));
}
#$numCoords= count($coords);
#for ($i=0;$i<$numCoords;$i++)
#    if (!$coords[$i])
#    array_push($peopleToAdd,array(
#                                    "type"=>"COORD", "orgID"=>$orgID, "index"=>$i,
#                    "email"=>$email[$i], "digest"=>$digest[$i]));#

if (count($peopleToAdd))
{
    session_register("peopleToAdd");
    session_register("orgID");
    localRedirect("Location: $baseURL/group/group-add-person.php");
    exit;
}

$loggedin = $HTTP_SESSION_VARS["loggedin"];
session_register("loggedin");
localRedirect("Location: $baseURL/group/group-setup.php#setup");
exit;
?>
```

C.7 PrepStep Prototype

setup/parseEventsFile.php

```
<?php
$dbname = "Tufts02";
$dbuser = "PrepStepUser";
#$dbpass = "";
# The apache process needs write permission to the buildarea
#$buildArea = "/usr/local/compinabox/var/PrepStep";
$buildArea = "";
# after setup move the following output files
# eventsBuild.inc, eventsStats.inc should be moved to the PREPARSED directory
# eventsParsed.inc should be moved to the CLASSES directory

function purgeDatabase($db,$override=FALSE)
{
  $query = "SELECT count(*) FROM couples;";
  $result = pg_exec($db,$query);
  list ($num)=pg_fetch_row($result,0);
  if (0!=$num && !$override)
    die("This is a destructive installer; and you invoked it without the
?override=TRUE option. There are couples present in the database. Please
remove them manually and try again, or invoke it with the ?override=TRUE
option.");
  $query = "DELETE FROM couples;";
  $result = pg_exec($db,$query);
  $query = "DELETE FROM events_names;";
  $result = pg_exec($db,$query);
  $query = "DELETE FROM events_dances;";
  $result = pg_exec($db,$query);
  $query = "DELETE FROM events;";
  $result = pg_exec($db,$query);
  $query = "DELETE FROM events_categories;";
  $result = pg_exec($db,$query);
  $query = "DELETE FROM abbreviations;";
  $result = pg_exec($db,$query);
  $query = "DELETE FROM rules_text;";
  $result = pg_exec($db,$query);
  $query = "DELETE FROM rules_level_names;";
  $result = pg_exec($db,$query);
  $query = "DELETE FROM levels;";
  $result = pg_exec($db,$query);
  $query = "DELETE FROM packages_purchased;";
  $result = pg_exec($db,$query);
  $query = "DELETE FROM packages_cost;";
  $result = pg_exec($db,$query);
  $query = "DELETE FROM packages;";
  $result = pg_exec($db,$query);
  $query = "DELETE FROM styles;";
  $result = pg_exec($db,$query);
  $query = "DELETE FROM rules_age_level;";
  $result = pg_exec($db,$query);
  $query = "DELETE FROM comp_reg_rules;";
  $result = pg_exec($db,$query);
  $query = "DELETE FROM comp_reg_rules_exclude;";
  $result = pg_exec($db,$query);
  $query = "DELETE FROM comp_reg_rules_ignore;";
  $result = pg_exec($db,$query);
```

```
  $query = "DELETE FROM comp_reg_rules_styles;";
  $result = pg_exec($db,$query);
  $query = "DELETE FROM packages_purchased;";
  $result = pg_exec($db,$query);
}

# NOTE: can't use explode " " as dance names may include spaces
function getEventCat($db, $name, $styles, $dances)
{
    $firstSpace = strpos($name," ");
    $level = substr($name,0,$firstSpace);
    $rest = substr($name,$firstSpace+1,strlen($name));
    $secondSpace = strpos($rest," ");
    $styleFamily = substr($rest,0,$secondSpace);
    if (in_array($styleFamily,array_keys($dances)))
    {
        $styleFamily="MIXED";
        $eventDances=$rest;
    }
    else
        $eventDances = substr($rest,$secondSpace+1,strlen($rest));
    $style="";
    $eventDances = explode("/",$eventDances);
    $numDances = count($eventDances);
    for ($i=0;$i<$numDances;$i++)
    {
        $dance = $eventDances[$i];
        foreach($styles as $aStyle)
        {
            if ($styleFamily == $aStyle["Family"])
            {
                $offeredDances = $aStyle["Dances"];
                foreach ($offeredDances as $aDance)
                {
                    if ($aDance==$dance)
                        if (""==$style)
                            $style=$aStyle["Name"];
                    else if ($style!=$aStyle["Name"])
                        $style="MIXED";
                }
            }
        }
    }
    if ($style==""||$style=="MIXED")
        $query="SELECT categoryid FROM events_categories WHERE
level='$level' AND style IS NULL;";
    else
        $query="SELECT categoryid FROM events_categories WHERE
level='$level' AND style='$style';";
    $result=pg_exec($db,$query);
    if (pg_numrows($result)==0)
    {
        if ($style!="" && $style!="MIXED")
            $query="INSERT INTO events_categories (level, style)
VALUES ('$level','$style');";
        else
            $query="INSERT INTO events_categories (level) VALUES ('$level');";
        $result=pg_exec($db,$query);
        $oid=pg_getlastoid($result);
        $query="SELECT categoryid FROM events_categories WHERE oid=$oid;";
        $result=pg_exec($query);
```

```
        list($id)=pg_fetch_row($result,0);
    }
    else
        list($id)=pg_fetch_row($result,0);
    return $id;
}

function getEventAbbrev($db,$name,$dances)
{
    # NOTE: can't use explode " " as dance names may include spaces
    $firstSpace = strpos($name," ");
    $level = substr($name,0,$firstSpace);
    $rest = substr($name,$firstSpace+1,strlen($name));
    $secondSpace = strpos($rest," ");
    $style = substr($rest,0,$secondSpace);
    if (in_array($styleFamily,array_keys($dances)))
    {
        $style="";
        $eventDances=$rest;
    }
    else
        $eventDances =substr($rest,$secondSpace+1,strlen($rest));
    $eventDances = explode('/',$eventDances);
    $abbrev="";
    if ($level!="")
    {
        $query = "SELECT abbreviation FROM abbreviations WHERE fulltext='$level';";
        $result = pg_exec($db,$query);
        list($levelAbbrev)=pg_fetch_row($result,0);
        $abbrev .= $levelAbbrev . "-";
    }
    if ($style!="")
    {
        $query = "SELECT abbreviation FROM abbreviations WHERE fulltext='$style';";
        $result = pg_exec($db,$query);
        list($styleAbbrev)=pg_fetch_row($result,0);
        $abbrev .= $styleAbbrev . "-";
    }
    $numDances = count($eventDances);
    for ($i=0;$i<$numDances;$i++)
    {
        $abbrev .= $dances[$eventDances[$i]];
    }
    return $abbrev;
}

#comment
function levelLabels($db)
{
$query = "SELECT preciseorder, levelorder,name FROM levels ORDER by preciseorder;";
$result = pg_exec($db,$query);
$numLevels = pg_numrows($result);
$inverseTable = array();
for ($i=0; $i<$numLevels; $i++)
{
  list($precise, $order,$name)=pg_fetch_row($result,$i);
  if (isset($inverseTable[$order]))
    array_push($inverseTable[$order],$name);
  else
    $inverseTable[$order]=array($name);
}
```

```
  $numLabels = count($inverseTable);
  $keys = array_keys($inverseTable);
  $labelTable = array();
  for ($i=0; $i< $numLabels; $i++)
    {
      $numPossible = count($inverseTable[$keys[$i]]);
      $label = "";
      for ($j=0; $j<$numPossible; $j++)
        {
          if ($j!=0)
            $label.="/";
          $label .= $inverseTable[$keys[$i]][$j];
        }
      for ($j=0; $j<$numPossible; $j++)
        {
          $labelTable[$inverseTable[$keys[$i]][$j]]=$label;
        }
    }
  echo "<pre>";
  echo print_r($labelTable);
  echo "</pre>";
  return $labelTable;
}

function fillEventDances($db,$id,$name,$dances)
{
   $firstSpace = strpos($name," ");
   $level = substr($name,0,$firstSpace);
   $rest = substr($name,$firstSpace+1,strlen($name));
   $secondSpace = strpos($rest," ");
     $style = substr($rest,0,$secondSpace);
     if (in_array($styleFamily,array_keys($dances)))
     {
         $eventDances=$rest;
     }
     else
         $eventDances =substr($rest,$secondSpace+1,strlen($rest));
     $eventDances = explode('/',$eventDances);
     $numPieces = count($eventDances);
     for ($i=0;$i<$numPieces;$i++)
     {
         $text = $eventDances[$i];
         if (isset($dances[$text]))
         {
             $query = "INSERT INTO events_dances
(eventid,danceorder,dancename) VALUES ($id,".($i+1).
                 ",'$text');";
             pg_exec($db,$query);
         }
     }
}


function fillDbEvents($db, &$levels,&$dances,&$styles,&$events)
{
     foreach($levels as $key => $level)
     {
         $name = $level["Name"]; $abbrev=$level["Abbrev"];$order=$level["Order"];
         $desc = addslashes($level["Description"]);
         $query = "INSERT INTO abbreviations (fulltext,abbreviation)
VALUES ('$name', '$abbrev');";
```

```
        pg_exec($db,$query);
        $levelOrder = floor($order);
        $query = "INSERT INTO levels (name,levelorder,preciseorder)
VALUES ('$name',$levelOrder,$order);";
        $result = pg_exec($db,$query);
        $oid = pg_getlastoid($result);
        $query = "SELECT levelid FROM levels WHERE oid=$oid;";
        $result = pg_exec($query);
        list($id)=pg_fetch_row($result,0);
        $levels[$key]["ID"]=$id;
        $query = "INSERT into rules_level_names (index,name) VALUES ($id,'$name');";
        $result = pg_exec($db,$query);

        $query = "INSERT INTO rules_text (level, markup,rule)
VALUES ($id,'HTML','$desc');";
        $result = pg_exec($db,$query);

    }
    foreach($dances as $name => $abbrev)
    {
        $query = "INSERT INTO abbreviations (fulltext,abbreviation)
VALUES ('$name', '$abbrev');";
        pg_exec($db,$query);
    }
    foreach($styles as $key => $style)
    {
        $name = $style["Name"]; $abbrev=$style["Abbreviation"];
        $order=$style["Order"]; $type=$style["Family"];
        $styleDances = $style["Dances"];
        $query = "SELECT abbreviation FROM abbreviations WHERE fulltext='$type';";
        $result = pg_exec($db,$query);
        if (pg_numrows($result)==0)
        {
            $query = "INSERT INTO abbreviations (fulltext,abbreviation)
VALUES ('$type', '$abbrev');";
            pg_exec($db,$query);
        }
        $query = "INSERT INTO styles (name,type,styleorder)
VALUES ('$name','$type',$order);";
        $result = pg_exec($db,$query);
        $oid = pg_getlastoid($result);
        $query = "SELECT styleid FROM styles WHERE oid=$oid;";
        $result = pg_exec($query);
        list($id)=pg_fetch_row($result,0);
        $styles[$key]["ID"]=$id;
        foreach($styleDances as $key=>$name)
        {
            $query = "INSERT INTO styles_dances
(styleid,danceorder,dancename) VALUES ($id,$key,'$name');";
            pg_exec($db,$query);
        }
    }
    foreach($events as $key =>$event)
    {
        $name = $event["Name"]; $eventNum = $event["EventNum"];
        $shortName = getEventAbbrev($db,$name,$dances);
        $eventCat = getEventCat($db,$name,$styles,$dances);
        $query = "INSERT INTO events (programnumber, categoryid)
VALUES ($eventNum,$eventCat);";
        $result = pg_exec($db,$query);
        $oid = pg_getlastoid($result);
```

```
        $query = "SELECT eventid FROM events WHERE oid=$oid;";
        $result = pg_exec($db,$query);
        list($id)=pg_fetch_row($result,0);
        $events[$key]["ID"]=$id;
        $query = "INSERT INTO events_names (eventid,shortname,longname)
VALUES ($id,'$shortName','$name');";
        pg_exec($db,$query);
        $query = "INSERT INTO events_est_reg (eventid,estreg,curreg)
VALUES ($id,0,0);";
        pg_exec($db,$query);
        fillEventDances($db,$id,$name,$dances);
    }
}

function fillDbPackages($db,$fees,&$packages,$prices)
{
  $numFeeClasses = count($fees);
  $feeKeys = array_keys($fees);
  for ($i=0;$i<$numFeeClasses;$i++)
    {
      $name = $feeKeys[$i];
      $query = "INSERT INTO rules_age_level (name) values ('$name');";
      $result = pg_exec($db,$query);
    }
  $numPackages = count($packages);
  $packageKeys =array_keys($packages);
  for ($i=0;$i<$numPackages;$i++)
    {
      $name = $packageKeys[$i];
      $query = "INSERT INTO packages (name) values ('$name');";
      $result = pg_exec($db,$query);
      $oid = pg_getlastoid($result);
      $query = "SELECT packageid FROM packages WHERE oid=$oid;";
      $result = pg_exec($db,$query);
      list($pid) = pg_fetch_row($result,0);
      $packages[$name]["PID"]=$pid;
      $levelsPerDance    = $packages["$name"]["LevelsPerDance"];
      if (""==$levelsPerDance) $levelsPerDance="NULL";
      $levelsOverall     = $packages["$name"]["LevelsOverall"];
      if (""==$levelsOverall) $levelsOverall="NULL";
      $ignoreEvents      = $packages["$name"]["IgnoreEvents"];
      $usersIgnoreEvents= (count($ignoreEvents)>=1);
      if (!$usesIgnoreEvents) $usesIgnoreEvents="FALSE"; else
$usesIgnoreEvents="TRUE";
      $excludeLevels     = $packages["$name"]["ExcludeLevels"];
      $usersExcludeLevels= (count($excludeLevels)>=1);
      if (!$usesExcludeLevels) $usesExcludeLevels="FALSE"; else
$usesExcludeLevels="TRUE";
      $styleLevels       = $packages["$name"]["StyleLevels"];
      $usersStyleLevels = (count($styleLevels)>=1);
      if (!$usesStyleLevels) $usesStyleLevels="FALSE"; else
$usesStyleLevels="TRUE";
      $allowOnlyLevel    = $packages["$name"]["AllowOnlyLevel"];
      if (!$allowOnlyLevel) $allowOnlyLevel="FALSE"; else $allowOnlyLevel="TRUE";
      $whichLevelOnly    = $packages["$name"]["WhichLevelOnly"];
      if (""==$whichLevelOnly)
        $whichLevelOnly="NULL";
      else
        {
          $query = "SELECT levelid FROM levels WHERE name='$whichLevelOnly';";
          $levelResult = pg_exec($db,$query);
```

```
            list($whichLevelOnly) =pg_fetch_row($levelResult,0);
          }
        $breakWithAltRole = $packages["$name"]["BreakWithAltRole"];
        if (!$breakWithAltRole) $breakWithAltRole="FALSE";
        else $breakWithAltRole="TRUE";
        $query = "INSERT INTO comp_reg_rules (packageid,lvls_per_dance,
lvls_overall,usesignoreevents, usesexcludelevels,usesstylelevels,
allowonlylevel,whichlevelonly,breakwithaltrole) VALUES($pid,$levelsPerDance,
$levelsOverall,$usesIgnoreEvents,$usesExcludeLevels,$usesStyleLevels,
$allowOnlyLevel,$whichLevelOnly,$breakWithAltRole);";
        $result = pg_exec($db,$query);
        if ($usesIgnoreEvents)
          {
            $numEvents=count($ignoreEvents);
            for ($j=0;$j<$numEvents;$j++)
              {
                $eventName = $ignoreEvents[$j];
                $query = "SELECT eventid FROM events_names
WHERE shortname='$eventName';";
                $result = pg_exec($db,$query);
                list($eid) = pg_fetch_row($result,0);
                $query = "INSERT INTO comp_reg_rules_exclude
(packageid,eventid) VALUES ($pid,$eid);";
                $result = pg_exec($db,$query);
              }
          }
        if ($usesExcludeLevels)
          {
            $numLevels=count($excludeLevels);
            for ($j=0;$j<$numLevels;$j++)
              {
                $levelName = $ignoreEvents[$j]["Name"];
                $ignoreAlways = $ignoreEvents[$j]["Always"];
                $ignoreAdj    = $ignoreEvents[$j]["Adjacent"];
                $query = "SELECT levelid FROM levels WHERE name='$levelName';";
                $result = pg_exec($db,$query);
                list($lid) = pg_fetch_row($result,0);
                $query = "INSERT INTO comp_reg_rules_ignore
(packageid,level,ignorealways,ignoreadjacentto)
VALUES ($pid,$lid,$ignoreAlways,$ignoreAdj);";
                $result = pg_exec($db,$query);
              }
          }
        if ($usesStyleLevels)
          {
            $numStyles=count($styleLevels);
            for ($j=0;$j<$numStyles;$j++)
              {
                $styleName = $styleLevels[$j]["Name"];
                $numLevels    = $styleLevels[$j]["NumLevels"];
                $query = "SELECT styleid FROM styles WHERE name='$styleName';";
                $result = pg_exec($db,$query);
                list($sid) = pg_fetch_row($result,0);
                $query = "INSERT INTO comp_reg_rules_styles
(packageid,styleid,lvls_this_style) VALUES ($pid,$sid,$numLevels);";
                $result = pg_exec($db,$query);
              }
          }
      }
    $numPrices = count($prices);
    $priceKeys = array_keys($prices);
```

```
  for ($i=0;$i<$numPrices;$i++)
    {
      $name = $priceKeys[$i];
      $query = "SELECT packageid FROM packages WHERE name='$name';";
      $result = pg_exec($db,$query);
      list ($pid) = pg_fetch_row($result,0);
      $test = $prices[$name];
      if (is_array($test))
        {
          $catKeys =array_keys($test);
          $numCats = count($catKeys);
          for ($j=0;$j<$numCats;$j++)
            {
              $catName =$catKeys[$j];
              $cost = $prices[$name][$catKeys[$j]];
              $query = "INSERT INTO packages_cost (packageid,cost,category)
VALUES ($pid,$cost,'$catName');";
              $result = pg_exec($db,$query);
            }
        }
      else
        {
          $query = "SELECT name FROM rules_age_level;";
          $catResult = pg_exec($db,$query);
          $numCats = pg_numrows($result);
          for ($j=0;$j<$numCats;$j++)
            {
              list($cat) = pg_fetch_row($catResult,$j);
              $query = "INSERT INTO packages_cost (packageid,cost,category)
VALUES ($pid,$test,'$cat');";
              pg_exec($db,$query);
            }
        }
    }
}

function eventsBuildFile($db,$filename)
{
    $text =  "<?php\n";
    $text="#############################################################\n";
    $text .= "# This file is generated by PrepStep and is required  #\n";
    $text .= "# for proper operation of your SlidingDoor web site.  #\n";
    $text .= "# Editing this file by hand will likely result in     #\n";
    $text .= "# breaking your site.  You have been warned.          #\n";
    $text .= "#                                                     #\n";
    $text .= "# Generated By: parseEventsFile.php::eventsBuildFile  #\n";
    $text="#############################################################\n";
    $text .= "\n";
    $text .= '$includedEvents=array('."\n";
    $query = "SELECT eventid, shortname, longname FROM events_names;";
    $result = pg_exec($db,$query);
    $numEvents = pg_numrows($result);
    for ($i=0;$i<$numEvents;$i++)
    {
        if ($i!=0)
            $text .= ",\n";
        list($id,$shortName,$longName) = pg_fetch_row($result,$i);
        $text .= "\tarray($id,\"$shortName\",\"$longName\")";
    }
    $text .= ");\n";
    $text="#############################################################\n";
```

```
    $text .= "# THIS FILE IS COMPUTER GENERATED CODE DO NOT EDIT BY #\n";
    $text .= "# HAND.  See top comment for information on what      #\n";
    $text .= "# function produces it and what function includes it. #\n";
    $text.="##########################################################\n";
    $text .="?>\n";

    $fp = fopen($filename,"w");
    fwrite($fp,$text);
    fclose($fp);
}

function eventsStatsFile($db,$filename,$tree)
{
    $text =  "<?php\n";
    $text.="##########################################################\n";
    $text .= "# This file is generated by PrepStep and is required  #\n";
    $text .= "# for proper operation of your SlidingDoor web site.  #\n";
    $text .= "# Editing this file by hand will likely result in     #\n";
    $text .= "# breaking your site.  You have been warned.          #\n";
    $text .= "#                                                     #\n";
    $text .= "# Generated By: parseEventsFile.php::eventsStatsFile  #\n";
    $text.="##########################################################\n";
    $text .= "\n";
    $levels = array_keys($tree);
    $numLevels = count($levels);
    $navbar ="";
    $body="";
    for ($i=0;$i<$numLevels;$i++)
    {
        if ($i!=0)
            $navbar .= " | ";
        $styles = array_keys($tree[$levels[$i]]);
        $numStyles = count($styles);
        $levelName = $levels[$i];
        $navbar .= "<a href=\\\"#$levelName\\\">$levelName</a> ";
        $body .= "\$includedTemp .= \"<a name=\\\"$levelName\\\"><h3>
$levelName</h3></a>\\n\";\n";
        if ($numStyles!=1)
        {
            $navbar .= "(";
        }
        for ($j=0;$j<$numStyles;$j++)
        {
            $anchor = $levels[$i]."-".$styles[$j];
            $label = $stystyles[$j];
            if ($j!=0) $navbar .= ", ";
            if ($numStyles!=1)
            {
                $navbar.="<a href=\\\"#$anchor\\\">$label</a>";
                $body .= "\$includedTemp .= \"<a name=\\\"$anchor\\\">
<h4>$label</h4></a>\\n\";\n";
            }
            $events = $tree[$levels[$i]][$styles[$j]]["Events"];
            $numEvents = count($events);
            for ($k=0;$k<$numEvents;$k++)
            {
                $eventID = $events[$k]["ID"];
                $shortName = $events[$k]["Short"];
                $longName = $events[$k]["Long"];

                $body .= "\$query = \"SELECT COUNT(*) FROM
```

```
events_registration WHERE eventid=$eventID;\";\n";
                $body .= "\$result=\$this->query(\$query);\n";
                $body .= "list(\$num) = \$result->getRowAt(0);\n";
                $body .= "\$includedTemp .=
\"<a href=\\\"\$CIB_BASE_URL/stats/\$unixname/Events/$shortName.html\\\">\";\n";
                $body .= "\$includedTemp .= \"$longName</a>: \$num<br>\";\n";
            }
        }
        if ($numStyles!=1)
        {
            $navbar .= ")";
        }
    }
    $navbar .= "<br />";

    $navbar = '$includedTemp = "'.$navbar.'";'."\n";
    $text .= $navbar . $body;
    $text="#########################################################\n";
    $text .= "# THIS FILE IS COMPUTER GENERATED CODE DO NOT EDIT BY #\n";
    $text .= "# HAND.  See top comment for information on what      #\n";
    $text .= "# function produces it and what function includes it. #\n";
    $text="#########################################################\n";
    $text .= "?>\n";
    $fp = fopen($filename,"w");
    fwrite($fp,$text);
    fclose($fp);
}

function eventsParsedFile($db,$filename,$tree)
{
    $text =  "<?php\n";
    $text="#########################################################\n";
    $text .= "# This file is generated by PrepStep and is required  #\n";
    $text .= "# for proper operation of your SlidingDoor web site.  #\n";
    $text .= "# Editing this file by hand will likely result in     #\n";
    $text .= "# breaking your site.  You have been warned.          #\n";
    $text .= "#                                                     #\n";
    $text .= "# Generated By: parseEventsFile.php::eventsParsedFile #\n";
    $text .= "#                                                     #\n";
    $text .= "# This code is directly embedded inside the function  #\n";
    $text .= "# HTMLDisplay::displayFormEventsTable and has access  #\n";
    $text .= "# to its variables: table, dances, person, and couple #\n";
    $text="#########################################################\n";
    $text .= "\n";
    $text .= "if (\$couple!=0)\n";
    $text .= "\t\$events=\$couple->getDances();\n";
    $text .= "if (formValue(\"Events\")!=\"\")\n";
    $text .= "\t\$events=formValue(\"Events\");\n";
    $text .= "if (\$events==\"\") \$events=array();\n";
    # loopTest holds the code for figuring out which -ALL blocks should
    # be silvered out
    $loopText .= "\$numEvents=count(\$dances);\n";
    $loopText .= "for (\$i=0;\$i<\$numEvents;\$i++)\n";
    $loopText .= "{\n";
    $loopText .= "\t\$registered[\$dances[\$i]]=TRUE;\n";



    $levels = array_keys($tree);
    $numLevels = count($levels);
    for ($i=0;$i<$numLevels;$i++)
```

```
    {
        $styles = array_keys($tree[$levels[$i]]);
        $numStyles = count($styles);
        $levelVar = $levels[$i];
        $levelVarOrig = $levelVar;
        $levelVar = str_replace("/","_",$levelVar);
        $initText .= "\$$levelVar=FALSE;\n";
        $levelTest1Var=$levelVar."Test";
        $loopText .= "\t\$$levelTest1Var=FALSE;\n";
        $query = "SELECT abbreviation FROM abbreviations WHERE
fulltext='$levelVarOrig';";
        echo $query . "<br />";

        $result = pg_exec($db,$query);
        if (pg_numrows($result)>0)
          {
            list($levelAbbrev)=pg_fetch_row($result,0);
            $loopText .= "\tif(!$levelTest1Var)\n";
            $loopText .= "\t\t\$$levelTest1Var =
strpos(\".\".\$dances[\$i],\"$levelAbbrev\".\"-\");\n";
            $loopText .= "\tif(\$$levelTest1Var!=1) \$$levelTest1Var=0;\n";
          }
        else
          {
            $possMatches = explode("_",$levelVar);
            $numLookups = count($possMatches);
            for ($j=0;$j<$numLookups;$j++)
              {
                $lookupText = $possMatches[$j];
                $query = "SELECT abbreviation FROM abbreviations
WHERE fulltext='$lookupText';";
                $result = pg_exec($db,$query);
                list($levelAbbrev) = pg_fetch_row($result,0);
                $loopText .= "\tif(!$levelTest1Var)\n";
                $loopText .= "\t\t\$$levelTest1Var =
strpos(\".\".\$dances[\$i],\"$levelAbbrev\".\"-\");\n";
                $loopText .= "\tif(\$$levelTest1Var!=1) \$$levelTest1Var=0;\n";
              }
          }
        $levelTest2Var=$levelVar."Test2";
        $loopText .= "\tif (\$$levelTest1Var)\n";
        $loopText .= "\t\t\$$levelVar=TRUE;\n";
        for ($j=0;$j<$numStyles;$j++)
        {
            $style = $styles[$j];
            $label = $levelVar . $style;
            $initText .= "\$$label=FALSE;\n";

            $events = $tree[$levels[$i]][$styles[$j]]["Events"];
            $numEvents = count($events);
            $eventArray="";
            for ($k=0;$k<$numEvents;$k++)
            {
                if ($k!=0)
                    $eventArray .= ", ";
                $eventID = $events[$k]["ID"];
                $shortName = $events[$k]["Short"];
                $longName = $events[$k]["Long"];
                $eventArray .= "\"$shortName\"";

            }
```

```
            $loopText .= "\tif (\$$levelVar && !\$$label &&
in_array(\$dances[\$i],array($eventArray)))\n";
            $loopText .= "\t\t\$$label = TRUE;\n";
        }
    }
    # tableText  holds the actual table
    $loopText .= "}\n";
    $tableText .= "\$table .= \"<table width=\\\"100%\\\">\\n\";\n";
    for ($i=0;$i<$numLevels;$i++)
    {
        $styles = array_keys($tree[$levels[$i]]);
        $numStyles = count($styles);
        $levelVar = $levels[$i];
        $query = "SELECT * from levels where name='$levelVar';";
        $result = pg_exec($db,$query);
        if (pg_numrows($result))
          $levelLink =
"<a href=\\\"\$baseURL/PREPARSED/rules.html#$levelVar\\\"
target=\\\"_blank\\\">$levelVar</a>";
        else
          {
            $levelLink = "";
            $levelComponents = explode("/",$levelVar);
            foreach ($levelComponents as $aLevel)
              {
                if ($levelLink!="") $levelLink .= " / ";
                $levelLink .=
"<a href=\\\"\$baseURL/PREPARSED/rules.html#$aLevel\\\"
target=\\\"_blank\\\">$aLevel</a>";
              }
          }
        $levelLabel = str_replace("/","_",$levelVar);
        $tableText .= "\$table .= \"\\t<tr>\\n\";\n";
        $tableText .= "\$table .= \"\\t\\t<th \" . (\$$levelLabel ?
\"bgcolor=\\\"silver\\\" \" :\"\") . \"colspan=\\\"$numStyles\\\">
$levelLink - <input type=\\\"checkbox\\\" name=\\\"events[]\\\"
value=\\\"$levelLabel"."-ALL\\\">All</th></tr>\";\n";
        $tableText .= "\$table .= \"\\t<tr>\\n\";\n";
        $maxEvents=0;
        for ($j=0;$j<$numStyles;$j++)
        {
            $style = $styles[$j];
            $label = $levelsLabel . $style;
            $tableText .= "\$table .= \"\\t\\t<th \" . (\$$label ?
\"bgcolor=\\\"silver\\\" \" :\"\") . \"width=\\\"25%\\\"
colspan=\\\"1\\\">$style - <input type=\\\"checkbox\\\"
name=\\\"events[]\\\" value=\\\"$levelLabel-$style"."-ALL\\\">All</th>\";\n";
            $events = $tree[$levels[$i]][$styles[$j]]["Events"];
            $numEvents = count($events);
            if ($numEvents > $maxEvents) $maxEvents=$numEvents;
        }
        $tableText .= "\$table .= \"\\t</tr>\";\n";
        # looping over events then styles in order to build table properly
        # maintaining the same index mapping (k for events, j for styles)
        for ($k=0;$k<$maxEvents;$k++)
        {
            $tableText .= "\$table .= \"\\t<tr>\\n\";\n";
            for ($j=0;$j<$numStyles;$j++)
            {
                if (isset($tree[$levels[$i]][$styles[$j]]["Events"][$k]))
                {
```

```
                    $event = $tree[$levels[$i]][$styles[$j]]["Events"][$k];
                    $short = $event["Short"];
                    $eventID = $event["ID"];
                    $query = "SELECT dancename FROM events_dances
WHERE evcntid=$eventID ORDER BY danceorder;";
                    $result = pg_exec($db,$query);
                    $numDances = pg_numrows($result);
                    $eventName = "";
                    for ($m=0;$m<$numDances;$m++)
                    {
                        if ($m!=0)
                            $eventName .= "/ ";
                        list($danceName)=pg_fetch_row($result,$m);
                        $eventName.=$danceName;
                    }
                    $tableText .= "\$table .= \"\\t\\t<td \".
(\$registered[\"$short\"] ? \" bgcolor=\\\"silver\\\" \" : \"\") .
\"colspan=\\\"1\\\"><input type=\\\"checkbox\\\" name=\\\"events[]\\\"
value=\\\"$short\\\"\" . (in_array(\"$short\",\$events) ? \"
checked=\\\"true\\\"\" :\"\") . \"></td>\\n\";\n";
                }
                else
                    $tableText .= "\$table .= \"\\t\\t<td></td>\\n\";";
            }
            $tableText .= "\$table .= \"\\t</tr>\\n\";\n";
        }
    }
    $tableText .= "\$table .= \"</table>\\n\";\n";
    $text .= $initText . $loopText . $tableText;
    $text="#############################################################\n";
    $text .= "# THIS FILE IS COMPUTER GENERATED CODE DO NOT EDIT BY #\n";
    $text .= "# HAND.  See top comment for information on what      #\n";
    $text .= "# function produces it and what function includes it. #\n";
    $text="#############################################################\n";
    $text .=  "?>\n";
    $fp = fopen($filename,"w");
    fwrite($fp,$text);
    fclose($fp);
}

function getEventTree($db,$levelLabels)
{
  $query = "SELECT categoryid,level,style,preciseorder,levelorder
FROM events_categories JOIN levels ON (level=levels.name) JOIN styles
ON (style=styles.name) ORDER BY preciseorder, styleorder;";
  $result = pg_exec($db,$query);
  $numCats = pg_numrows($result);
  for ($i=0;$i<$numCats;$i++)
    {
      list($catID,$aLevel,$aStyle,$preciseOrder,$roundedOrder) =
pg_fetch_row($result,$i);
      $query = "SELECT eventid, longname, shortname FROM events NATURAL
JOIN events_names WHERE categoryid=$catID;";
      $catResult = pg_exec($db,$query);
      $numEvents = pg_numrows($catResult);
      $tree[$levelLabels[$aLevel]][$aStyle]["Label"]=$aLevel;
      $tree[$levelLabels[$aLevel]][$aStyle]["NumEvents"]=$numEvents;
      $tree[$levelLabels[$aLevel]][$aStyle]["Events"]=array();
      for ($j=0;$j<$numEvents;$j++)
        {
          list($eventID, $lName, $sName) = pg_fetch_row($catResult,$j);
```

```
            array_push($tree[$levelLabels[$aLevel]][$aStyle]["Events"],
                       array("ID"=>$eventID,"Short"=>$sName,
                             "Long"=>$lName));
          }
      }
#  echo "<PRE>";
#  echo print_r($tree);
#  echo "</PRE>";
  return $tree;
}

function couple__getDances($db,$filename)
{
  $text =  "<?php\n";
  $text="#########################################################\n";
  $text .= "# This file is generated by PrepStep and is required  #\n";
  $text .= "# for proper operation of your SlidingDoor web site.  #\n";
  $text .= "# Editing this file by hand will likely result in     #\n";
  $text .= "# breaking your site.  You have been warned.          #\n";
  $text .= "#                                                     #\n";
  $text .= "# Generated By:                                       #\n";
  $text .= "#    parseEventsFile.php::couple__getDances           #\n";
  $text .= "#                                                     #\n";
  $text .= "# This code is directly embedded inside the function  #\n";
  $text .= "# Couple::getDances and has access to its variables:  #\n";
  $text .= "# this, events                                        #\n";
  $text="#########################################################\n";
  $text .= "\n";
  $text .= "switch(\$style)\n";
  $text .= "{\n";

  $query = "SELECT styleid, name, type FROM styles;";
  $result = pg_exec($db,$query);
  $numStyles = pg_numrows($result);
  $comboStyles=array();
  for ($i=0;$i<$numStyles;$i++)
    {
      list($styleid,$name,$type) = pg_fetch_row($result,$i);
      $typeVar = $type."FamilyList";
      $styleVar = $name."StyleList";
      $query = "SELECT abbreviation FROM abbreviations WHERE fulltext ='$type';";
      if (!in_array("$type",$comboStyles))
        array_push($comboStyles,"$type");
      $styleResult = pg_exec($db,$query);
      list($styleAbbrev) = pg_fetch_array($styleResult,0);
      $query = "SELECT abbreviation FROM styles_dances JOIN abbreviations
ON (dancename=fulltext) WHERE styleid=$styleid;";
      $danceResult = pg_exec($db,$query);
      $numDances = pg_numrows($danceResult);
      for ($j=0;$j<$numDances;$j++)
        {
          if (""!=$$styleVar)
            $$styleVar .= ", ";
          if (""!=$$typeVar)
            $$typeVar .= ", ";
          list($danceAbbrev) = pg_fetch_row($danceResult,$j);
          $$styleVar .= "\"$danceAbbrev\"";
          $$typeVar .= "\"$danceAbbrev\"";
        }
      $temp = $$styleVar;
      if ($type!=$name)
```

```
        $text .= "case \"$name\" : \$result =
\$this->styleFilter(\$events,\"$styleAbbrev\",array($temp));\nbreak;\n";
    }
  $numStyles = count($comboStyles);
  for ($i=0;$i<$numStyles;$i++)
    {
      $typeVar = $comboStyles[$i]."FamilyList";
      $typeName = $comboStyles[$i];
      $query = "SELECT abbreviation FROM abbreviations WHERE
fulltext ='$typeName';";
      $styleResult = pg_exec($db,$query);
      list($styleAbbrev) = pg_fetch_array($styleResult,0);
      $temp = $$typeVar;
      $text .= "case \"$typeName\" : \$result =
\$this->styleFilter(\$events,\"$styleAbbrev\",array($temp));\nbreak;\n";
    }
  $text .= "\tdefault:\n\t\t\$result = \"\";\n";
  $text .= "}\n";
  $text="#############################################################\n";
  $text .= "# THIS FILE IS COMPUTER GENERATED CODE DO NOT EDIT BY #\n";
  $text .= "# HAND.  See top comment for information on what      #\n";
  $text .= "# function produces it and what function includes it. #\n";
  $text="#############################################################\n";
  $text .= "?>\n";
  $fp = fopen($filename,"w");
  fwrite($fp,$text);
  fclose($fp);
}

function abbreviationList($items)
{
  $text = "";
  $numItems = count($items);
  for ($i=0;$i<$numItems;$i++)
    {
      if ($i!=0) $text .= ", ";
      $text .= "\"".$items[$i]."\"";
    }
  return $text;
}

function couple__eventBefore($db,$levelBlock,$styleBlock,$danceBlock)
{
  $text =  "<?php\n";
  $text="#############################################################\n";
  $text .= "# This file is generated by PrepStep and is required  #\n";
  $text .= "# for proper operation of your SlidingDoor web site.  #\n";
  $text .= "# Editing this file by hand will likely result in     #\n";
  $text .= "# breaking your site.  You have been warned.          #\n";
  $text .= "#                                                     #\n";
  $text .= "# Generated By:                                       #\n";
  $text .= "#    parseEventsFile.php::couple__eventBefore         #\n";
  $text .= "#                                                     #\n";
  $text .= "# This code is directly embedded inside the function  #\n";
  $text .= "# Couple::eventBefore and has access to its variables:#\n";
  $text .= "# event1Lvl,event1Style,event1Dance                   #\n";
  $text="#############################################################\n";
  $text .= "\n";
  $levelText = $text;
  $styleText = $text;
  $danceText = $text;
```

```
  $levelText .= "switch(\$event2Lvl)\n";
  $levelText .= "{\n";
  $query = "SELECT abbreviation FROM abbreviations JOIN levels ON
(fulltext=name) ORDER BY levelorder;";
  $result = pg_exec($db,$query);
  $numLevels = pg_numrows($result);
  $levels = array();
  for ($i=0;$i<$numLevels;$i++)
    {
      list($abbrev) = pg_fetch_row($result,$i);
      array_push($levels,"$abbrev");
    }
  while (count($levels)>1)
    {
      $aLevel = array_shift($levels);
      $compareList = abbreviationList($levels);
      $levelText .= "case \"$aLevel\": if (in_array(\$event1Lvl,
array($compareList))) return TRUE; break;\n";
    }

  $levelText .= "\tdefault:\n\t\tbreak;\n}\nreturn FALSE;\n";
  $query = "SELECT MIN(styleorder) AS orderinfo, abbreviation FROM styles
JOIN abbreviations ON (type=fulltext) GROUP BY abbreviation ORDER BY orderinfo;";
  $result = pg_exec($db,$query);
  $numStyles = pg_numrows($result);
  $styles = array();
  $styleText .= "switch(\$event2Style)\n";
  $styleText .= "{\n";
  for ($i=0;$i<$numStyles;$i++)
    {
      list($dummy,$abbrev) = pg_fetch_row($result,$i);
      array_push($styles,"$abbrev");
    }
  while (count($styles)>1)
    {
      $aStyle = array_shift($styles);
      $compareList = abbreviationList($styles);
      $styleText .= "case \"$aStyle\": if (in_array(\$event1Style,
array($compareList))) return TRUE; break;\n";
    }
  $styleText .= "\tdefault:\n\t\tbreak;\n}\nreturn FALSE;\n";

  $query = "SELECT a1.abbreviation,a2.abbreviation FROM styles JOIN
abbreviations AS a1 ON (type=a1.fulltext) NATURAL JOIN styles_dances
JOIN abbreviations AS a2 ON (dancename=a2.fulltext)
ORDER BY a1.abbreviation,styleorder, danceorder;";
  $result = pg_exec($db,$query);
  $styleDances = array();
  $numDances = pg_numrows($result);
  for ($i=0;$i<$numDances;$i++)
    {
      list($style,$dance) = pg_fetch_row($result,$i);
      if (!in_array($style,array_keys($styleDances)))
        $styleDances[$style]=array();
      array_push($styleDances[$style],"$dance");
    }
  $danceText .= "switch(\$event2Style)\n";
  $danceText .= "{\n";
  $styleNames = array_keys($styleDances);
  while (count($styleDances))
```

```
    {
      $styleAbbrev = array_shift($styleNames);
      $dances = array_shift($styleDances);
      $danceText .= "case \"$styleAbbrev\" : switch (\$event2Dance)\n\t{\n";
      while (count($dances)>1)
        {
          $aDance = array_shift($dances);
          $compareList = abbreviationList($dances);
          $danceText .= "\t\tcase \"$aDance\": if (in_array(\$event1Dance,
array($compareList))) return TRUE; break;\n";
        }
      $danceText .= "\t\tdefault: break;\n\t} break;\n";
    }
  $danceText .= "default:\n\tbreak;\n";
  $danceText .= "}\n";
  $danceText .= "return TRUE;\n";
  $footer="############################################################\n";
  $footer .= "# THIS FILE IS COMPUTER GENERATED CODE DO NOT EDIT BY #\n";
  $footer .= "# HAND.  See top comment for information on what      #\n";
  $footer .= "# function produces it and what function includes it. #\n";
  $footer.="############################################################\n";


  $levelText .= $footer ."?>\n";
  $fp = fopen($levelBlock,"w");
  fwrite($fp,$levelText);
  fclose($fp);
  $styleText .= $footer. "?>\n";
  $fp = fopen($styleBlock,"w");
  fwrite($fp,$styleText);
  fclose($fp);
  $danceText .= $footer . "?>\n";
  $fp = fopen($danceBlock,"w");
  fwrite($fp,$danceText);
  fclose($fp);
}

function couple__setEvents($db,$eventLutFile, $eventSetFile,$tree)
{
  $text =  "<?php\n";
  $text.="############################################################\n";
  $text .= "# This file is generated by PrepStep and is required  #\n";
  $text .= "# for proper operation of your SlidingDoor web site.  #\n";
  $text .= "# Editing this file by hand will likely result in     #\n";
  $text .= "# breaking your site.  You have been warned.          #\n";
  $text .= "#                                                     #\n";
  $text .= "# Generated By:                                       #\n";
  $text .= "#     parseEventsFile.php::couple__setDances          #\n";
  $text .= "#                                                     #\n";
  $text .= "# This code is directly embedded inside the function  #\n";
  $text .= "# Couple::setEvents and has access to its variables:  #\n";
  $text .= "# temp,events                                         #\n";
  $text.="############################################################\n";
  $text .= "\n";
  $query = "SELECT eventid,shortname FROM events_names;";
  $result = pg_exec($db,$query);
  $numEvents = pg_numrows($result);
  $masterLutText = $text;
  $masterLutText .= "\$eventLUT=array();\n";
  for ($i=0;$i<$numEvents;$i++)
    {
```

```
      list ($eventID,$eventAbbrev) = pg_fetch_row($result,$i);
      $masterLutText .= "\$eventLUT[\"$eventAbbrev\"]=$eventID;\n";
    }
  $eventSetText = $text;
  $eventSetText .= "switch(\$events[\$i])\n{\n";
  $levels = array_keys($tree);
  $numLevels = count($levels);
  for ($i=0;$i<$numLevels;$i++)
    {
      $styles = array_keys($tree[$levels[$i]]);
      $numStyles = count($styles);
      $levelVar = $levels[$i];
      $levelVar = str_replace("/","_",$levelVar);
#      $query = "SELECT abbreviation FROM abbreviations WHERE
fulltext='$levelVar';";
#      $result = pg_exec($db,$query);
#      list($levelAbbrev) = pg_fetch_row($result,0);
      $levelAll = $levelVar."-ALL";
      $eventSetText .= "case \"$levelAll\" : \$fallthrough=TRUE;\n";
      for ($j=0;$j<$numStyles;$j++)
        {
          $style = $styles[$j];
          $styleAll = $levelVar."-".$style."-ALL";
          $eventSetText .= "case \"$styleAll\" :";
          $events = $tree[$levels[$i]][$styles[$j]]["Events"];
          $numEvents = count($events);
          for ($k=0;$k<$numEvents;$k++)
            {
              if ($k!=0)
                $eventSetText .= "\n\t";
              $shortName = $events[$k]["Short"];
              $eventSetText .= "array_push(\$temp,
\$eventLUT[\"$shortName\"]);";
            }
          $eventSetText .= "\n\tif (!\$fallthrough) break;\n";
        }
      $eventSetText .= "case \"{$levelVar}Catch\" : break;\n";
    }
  $eventSetText .= "default: array_push(\$temp,\$eventLUT[\$events[\$i]]);\n";
  $eventSetText .= "}\n";
  $footer="#############################################################\n";
  $footer .= "# THIS FILE IS COMPUTER GENERATED CODE DO NOT EDIT BY #\n";
  $footer .= "# HAND.  See top comment for information on what      #\n";
  $footer .= "# function produces it and what function includes it. #\n";
  $footer="#############################################################\n";


  $masterLutText .= $footer."?>\n";
  $fp = fopen($eventLutFile,"w");
  fwrite($fp,$masterLutText);
  fclose($fp);
  $eventSetText .=  $footer."?>\n";
  $fp = fopen($eventSetFile,"w");
  fwrite($fp,$eventSetText);
  fclose($fp);
}



function htmlDisplay__partnerTableIncludes($db,
                                           $headerFile, $rowFile)
```

```
{
  $text =  "<?php\n";
  $text="#########################################################\n";
  $text .= "# This file is generated by PrepStep and is required  #\n";
  $text .= "# for proper operation of your SlidingDoor web site.  #\n";
  $text .= "# Editing this file by hand will likely result in     #\n";
  $text .= "# breaking your site.  You have been warned.          #\n";
  $text .= "#                                                     #\n";
  $text .= "# Generated By:  parseEventsFile.php::                #\n";
  $text .= "#          htmlDisplay__partnerTableIncludes          #\n";
  $text .= "#                                                     #\n";
  $text .= "# This code is directly embedded inside the function  #\n";
  $text .= "# HTMLDisplay::partnerTable and has access to its     #\n";
  $text .= "# variables: this, tempCouple, table                  #\n";
  $text="#########################################################\n";
  $text .= "\n";

  $headerText = $text .
"\$table .= \"\\t<tr>\\n\\t\\t<th colspan=\\\"2\\\">Options</th><th>Partner</th>";
  $rowText = $text;
  $query = "SELECT name FROM styles ORDER BY styleorder;";
  $result = pg_exec($db,$query);
  $numStyles = pg_numrows($result);
  for ($i=0;$i<$numStyles;$i++)
    {
      list($style)=pg_fetch_row($result,$i);
      $headerText .= "<th>$style</th>";
      $varName = $style . "Dances";
      $rowText .=
"\$$varName = \$this->colorDances(\$tempCouple->getDances(\"$style\"));\n";
      $rowText .= "\$table .= \"\\t\\t<td>\$$varName</td>\\n\";\n";
    }
  $headerText .= "</tr>\\n\";\n";
  $footer="#########################################################\n";
  $footer .= "# THIS FILE IS COMPUTER GENERATED CODE DO NOT EDIT BY #\n";
  $footer .= "# HAND.  See top comment for information on what      #\n";
  $footer .= "# function produces it and what function includes it. #\n";
  $footer="#########################################################\n";

  $headerText .= $footer."?>\n";
  $rowText .= $footer ."?>\n";
  $fp = fopen($headerFile,"w");
  fwrite($fp,$headerText);
  fclose($fp);
  $fp = fopen($rowFile,"w");
  fwrite($fp,$rowText);
  fclose($fp);
}

function htmlDisplay__colorIncludes($db, $colorFile, $legendFile,$levels,$labelTable)
{
  $text =  "<?php\n";
  $text="#########################################################\n";
  $text .= "# This file is generated by PrepStep and is required  #\n";
  $text .= "# for proper operation of your SlidingDoor web site.  #\n";
  $text .= "# Editing this file by hand will likely result in     #\n";
  $text .= "# breaking your site.  You have been warned.          #\n";
  $text .= "#                                                     #\n";
  $text .= "# Generated By:  parseEventsFile.php::                #\n";
  $text .= "#          htmlDisplay__colorIncludes                 #\n";
  $text .= "#                                                     #\n";
```

```
  $colorText = $text;
  $legendText = $text;

  $colorText .= "# This code is directly embedded inside the function  #\n";
  $colorText .= "# HTMLDisplay::colorDances and has access to it s     #\n";
  $colorText .= "# variables:  color                                   #\n";
  $colorText .="#######################################################\n";
  $colorText .= "\n";
  $colorText .= "switch(\$level)\n";
  $colorText .= "{\n";
  $legendText .= "# This code is directly embedded inside the function  #\n";
  $legendText .= "# HTMLDisplay::colorLegend and has access to it s     #\n";
  $legendText .= "# variables: result                                  #\n";
  $legendText .="#######################################################\n";
  $legendText .= "\n";
  $numLevels = count($levels);
  $level_keys = array_keys($levels);
  $percent = floor(100/$numLevels);
  $legendLine=array();
  for ($i=0;$i<$numLevels;$i++)
    {
      $name =   $labelTable[$levels[$level_keys[$i]]["Name"]];
      $abbrev = $levels[$level_keys[$i]]["Abbrev"];
      $order =  $levels[$level_keys[$i]]["Order"];
      $Color =  $levels[$level_keys[$i]]["Color"];
      $color = substr($Color,0,1).strtolower(substr($Color,1,strlen($Color)));
      $colorLine["$order"] = "\tcase \"$abbrev\" : \$color = \"$Color\"; break;\n";
      $legendLineTest=
"<td width=\"$percent%\"><font color=\"$Color\">$name: $color</font></td>";
      if (!in_array($legendLineTest,$legendLine))
        $legendLine["$order"]=$legendLineTest;
    }
  echo "<pre>";
echo  print_r($colorLine);
 echo "</pre>";

  $order_keys = array_keys($colorLine);
  for ($i=0;$i<$numLevels;$i++)
    {
      $colorText .= $colorLine[$order_keys[$i]];
      $legendText .= "\$result .= '".$legendLine[$order_keys[$i]]."';\n";
    }
  $colorText .= "\tdefault:\n\t\t\$color=\"BLACK\"; break;\n";
  $footer="#############################################################\n";
  $footer .= "# THIS FILE IS COMPUTER GENERATED CODE DO NOT EDIT BY #\n";
  $footer .= "# HAND.  See top comment for information on what      #\n";
  $footer .= "# function produces it and what function includes it. #\n";
  $footer .="#############################################################\n";
  $colorText .= "}\n$footer?>\n";
  $legendText .= $footer."?>\n";
  $fp = fopen($colorFile,"w");
  fwrite($fp,$colorText);
  fclose($fp);
  $fp = fopen($legendFile,"w");
  fwrite($fp,$legendText);
  fclose($fp);
}

function rulesHTML($db, $htmlFile, $rules, $url)
{
```

```
  $bodyHead = "<html><head><title>Competition Rules</title></head>\n";
  $bodyHead .= "<body bgcolor=\"white\">\n";

  #:TODO: grab the compname from the user
  #:TODO: Create a pre-Constants file that when combined with the other two
  #:TODO: PS inputs generates the real Constants.inc, allowing PS to know
  #:TODO: the comname, set defaults, etc
  $bodyHead .= "<h1>General Notes on Level Rules</h1>";
  $bodyHead .= $rules;
  if ($url!="")
    $bodyHead .= "<p />Additional information on these rules may be found
at <a href=\"$url\">$url</a>\n";
  $query = "SELECT name, rule FROM levels JOIN rules_text ON (levelid=level)
WHERE markup in ('HTML','html','NONE','none','XHTML','xhtml',NULL) ORDER BY
preciseorder;";
  $result = pg_exec($db,$query);
  $numLevels = pg_numrows($result);
  $bodyNav ="";
  $bodyText ="";
  for ($i=0;$i<$numLevels;$i++)
    {
      list ($levelName, $levelDesc) = pg_fetch_row($result,$i);
      $levelDesc = stripslashes($levelDesc);
      if ($i!=0)
        $bodyNav .= " | ";
      $bodyNav .= " <a href=\"#$levelName\">$levelName</a>";
      $bodyText .= "<a name=\"$levelName\"><h2>$levelName</h2></a>\n";
      $bodyText .= $levelDesc;
    }
  $body .= $bodyHead . "<p />";

  $body .= $bodyNav . $bodyText;
  $body .= "</body></html>";
  $fp = fopen($htmlFile,"w");
  fwrite($fp,$body);
  fclose($fp);
}


function packagesHtml($db,$packageHtmlFile,$packageIncludeFile,
                      $feeIncludeFile,$packages,$feeCategories)
{

  $packageText .= "<?php\n";
  $packageText .= "\$packages = array(";
  $feeText .= "<?php\n";
  $feeText .= "\$ages = array (";
  $htmlText .= "<html>\n<head>\n<title>Defination</title>\n</head>\n";
  $htmlText .= "<body bgcolor=\"white\">\n";
  $numPackages = count($packages);
  if ($numPackages>0)
    $htmlText .= "<h1>Packages Offered</h1>\n";
  $packageNav ="";
  $packageBody = "";
  $packageKeys = array_keys($packages);
  for ($i=0;$i<$numPackages;$i++)
    {
      if ($i!=0) $packageText.=",\n";
      if ($i!=0) $packageNav .= " | ";
      $packageName = $packageKeys[$i];
      $description = $packages[$packageName]["Description"];
```

```
      $query = "SELECT packageid FROM packages WHERE name='$packageName';";
      $result = pg_exec($db,$query);
      list($pid)=pg_fetch_row($result,0);
      $query = "SELECT distinct cost FROM packages_cost WHERE
packageid=$pid ORDER BY cost;";
      $result= pg_exec($db,$query);
      $numCosts = pg_numrows($result);
      $costText="(";
      for ($j=0;$j<$numCosts;$j++)
        {
          if ($j!=0) $costText .= "/";
          list($cost)=pg_fetch_row($result,$j);
          $costText.="\\\$$cost";
        }
      $costText.=")";
      $packageText .= "\"<a href=\\\"\$baseURL/PREPARSED/packages.html#
$packageName\\\" target=\\\"_blank\\\">$packageName</a> ($costText)\"=>$pid";
      $packageNav .= "<a href=\\\"#$packageName\\\">$packageName</a>";
      $packageBody .= "\n<h3><a name=\\\"$packageName\\\">$packageName</a>
</h3>\n";
      $packageBody .= $description;
    }
  if ($packageNav!="")
    $htmlText .= "$packageNav<br />\n$packageBody\n";
  $numFeeCats = count($feeCategories);
  if ($numFeeCats>0)
    $htmlText .= "<h1>Fee Category Defination</h1>\n";
  $feeNav ="";
  $feeBody = "";
  $feeKeys = array_keys($feeCategories);
  for ($i=0;$i<$numFeeCats;$i++)
    {
      if ($i!=0) $feeText.=",\n";
      if ($i!=0) $feeNav .= " | ";
      $feeName = $feeKeys[$i];
      $description = $feeCategories[$feeName];
      $query = "SELECT index FROM rules_age_level WHERE name='$feeName';";
      $result = pg_exec($db,$query);
      list($cid)=pg_fetch_row($result,0);
      $feeText .= "\"<a href=\\\"\$baseURL/PREPARSED/packages.html#
$feeName\\\" target=\\\"_blank\\\">$feeName</a>\"=>$cid";
      $feeNav .= "<a href=\\\"#$feeName\\\">$feeName</a>";
      $feeBody .= "\n<h3><a name=\\\"$feeName\\\">$feeName</a></h3>\n";
      $feeBody .= $description;
    }
  if ($feeNav!="")
    $htmlText .= "$feeNav<br />\n$feeBody\n";


  $htmlText .= "</body></html>";
  $feeText .= ");\n?>\n";
  $packageText .= ");\n?>\n";

  $fp = fopen($packageIncludeFile,"w");
  fwrite($fp,$packageText);
  fclose($fp);
  $fp = fopen($feeIncludeFile,"w");
  fwrite($fp,$feeText);
  fclose($fp);
  $fp = fopen($packageHtmlFile,"w");
  fwrite($fp,$htmlText);
```

```
  fclose($fp);
}

echo "<html><head><title>PrepStep Prototype Functions</title></head>";
echo "<body bgcolor=\"white\">";

if ($buildArea=="") die ("Please edit this file to include a path to a
directory where the <b>web server</b> can <b>write</b> files.");

$db = pg_connect("dbname=$dbname user=$dbuser password=$dbpass");
echo "Established database connection...<br />";

include ("events.inc");
echo "Read events.inc file...<br />";
include("packages.inc");
echo "Read packages.inc... <br />";

purgeDatabase($db,TRUE);
echo "Purging database...<br />";

fillDbEvents($db,$levels,$dances,$styles,$events);
echo "Populated database, with event information...<br />";

fillDbPackages($db,$feeCategories, $packages, $packagePrices);
echo "Populated database, with fee/package information...<br />";

$labelTable = levelLabels($db);
$tree = getEventTree($db,$labelTable);
echo "Created common level labels for equivilent levels... <br />";

eventsBuildFile($db,"$buildArea/compDB__rebuildEvents__eventsList.inc");
echo "Wrote compDB__rebuildEvents__eventsList.inc...<br />";

eventsStatsFile($db,"$buildArea/compDB__rebuildEvents__eventsSummary.inc",$tree);
echo "Wrote compDB__rebuildEvents__eventsSummary.inc...<br />";

eventsParsedFile($db,"$buildArea/htmlDisplay__displayFormEventsTable.inc",$tree);
echo "Wrote htmlDisplay__displayFormEventsTable.inc...<br />";

couple__setEvents($db,
                  "$buildArea/couple__setEventsLut.inc",
                  "$buildArea/couple__setEventsSet.inc",
                  $tree);
echo "Wrote couple__setDancesLut...<br />";
echo "Wrote couple__setDancesSet...<br />";

couple__getDances($db,"$buildArea/couple__getDances.inc");
echo "Wrote couple__getDances...<br />";
couple__eventBefore($db,
                    "$buildArea/couple__eventBeforeLevel.inc",
                    "$buildArea/couple__eventBeforeStyle.inc",
                    "$buildArea/couple__eventBeforeDance.inc");
echo "Wrote couple__eventBeforeLevel...<br />";
echo "Wrote couple__eventBeforeStyle...<br />";
echo "Wrote couple__eventBeforeDance...<br />";

htmlDisplay__colorIncludes($db,"$buildArea/htmlDisplay__colorDances.inc",
                           "$buildArea/htmlDisplay__colorLegend.inc",
$levels,$labelTable);
echo "Wrote htmlDisplay__coupleDances... <br />";
echo "Wrote htmlDisplay__coupleLegend... <br />";
```

```
htmlDisplay__partnerTableIncludes($db,
                                  "$buildArea/htmlDisplay__partnerTable__header.inc",
                                  "$buildArea/htmlDisplay__partnerTable__row.inc");
echo "Wrote htmlDisplay__parterTable_header... <br />";
echo "Wrote htmlDisplay__parterTable_row... <br />";

rulesHtml($db,"$buildArea/rules.html",$levelRules,$levelRulesURL);
echo "Wrote rules.html...<br/>";

packagesHtml($db,"$buildArea/packages.html",
             "$buildArea/htmlDisplay__packages.inc",
             "$buildArea/htmlDisplay__feeCategory.inc",
             $packages,$feeCategories);
echo "Wrote htmlDisplay__packages...<br />";
echo "Wrote htmlDisplay__feeCategory...<br />";
echo "Wrote packages...<br />";
echo "You have to fee htmlDisplay__package,feeCategory to find the ID's_to
insert_use_for_SD_defaults__in_constants.inc._NOTE_even_if_you_have_a_single
package,_you_must_set_the_default.";

echo_"DONE<br_/></body></html>";
exit;
?>
```

setup/events.inc

```
<?php
#

$levelRules = "At this competition you may enter in <b>one</b> level per style
(Standard,Latin,Smooth,Rhythm). Events within a style may not span two levels.
You may span an arbitrary number of levels between styles provided you meet
the eligiblity guidleines included below. 'Placing_out' refers to having
accumulated 7 or more YCN points earned at any amateur competition.<p />
Please note that this competition uses different levels name and eligibility
guidleines for American vs International events, please read the rules for your
events carefully.";

$levelRulesURL = 'http://tufts.edu/ballroom/comp/rules.html';

$levels = array(array("Name"=>"Newcomer","Abbrev"=>"N",
                      "Order"=>1,"Color"=>"PURPLE",
                      "Description"=>"The Newcomer level contains events in
    both American and International Styles. Neither member of a couple
dancing Newcomer may have been dancing for more than 3 months."),
                array("Name"=>"Beginner","Abbrev"=>"B",
                      "Order"=>2.1,"Color"=>"BLUE",
                      "Description"=>"The Beginner level is used for American
    events only. Any couple where both members have been dancing for less than
    two years and have not placed out of Bronze in any event within the style
    is eligible to compete at this level."),
                array("Name"=>"Bronze","Abbrev"=>"Br",
                      "Order"=>2,"Color"=>"BLUE",
                      "Description"=>"The Bronze level is used for
    International events. Any couple where both members have been dancing less
    that two years and have not placed out of Bronze in any event within the
style is eligible to compete at this level"),
                array("Name"=>"Silver","Abbrev"=>"S",
                      "Order"=>3,"Color"=>"GREEN",
                      "Description"=>"The Silver level is used for
    International events only. Any couple where neither member has placed
    out of Silver may compete. There is no time restriction."),
                array("Name"=>"Advanced","Abbrev"=>"A",
                      "Order"=>3.5,"Color"=>"GREEN",
                      "Description"=>"The Advanced level is used for American
    events only. This is a combined Silver/Gold level. As such, any couple
    where neither member has placed out of <b>Gold</b> is eligible to compete.
      Placing in this event awards YCN points at the <b>Silver</b> level."),
                array("Name"=>"Gold","Abbrev"=>"G",
                      "Order"=>4,"Color"=>"GOLD",
                      "Description"=>"The Gold level is used for International
    events only. Any couple where neither member has placed out of Gold may
    compete."),
                array("Name"=>"Collegiate_Open","Abbrev"=>"C",
                      "Order"=>5,"Color"=>"ORANGE",
                      "Description"=>"The Collegiate Open level is open to
any couple where <strong>both</strong> members are <strong>undergraduates
</strong>. Eligible couples may also enter the unrestrited Open level, as an
exception to the 1 level per level rule stated above."),
                array("Name"=>"Open","Abbrev"=>"O",
                      "Order"=>6,"Color"=>"RED",
                      "Description"=>"The Open level is used for both
    International and American events. Any amateur is eligble is compete."));

$dances["Waltz"]="W";
```

```
$dances["Tango"]="T";
$dances["Viennese Waltz"]="Vw";
$dances["Foxtrot"]="F";
$dances["Quickstep"]="Q";
$dances["Rumba"]="R";
$dances["Cha Cha"]="C";
$dances["Samba"]="Sa";
$dances["Jive"]="J";
$dances["Paso Doble"]="P";
$dances["Swing"]="Sw";
$dances["Bolero"]="B";
$dances["Mambo"]="M";
$dances["Merengue"]="Me";
$dances["Hustle"]="H";
$dances["Night Club 2-Step"]="N";
$dances["Peabody"]="Pb";
$dances["Salsa"]="Ss";



$styles = array(array("Name"=>"Smooth","Family"=>"American",
                      "Abbreviation"=>"A","Dances"=>array("Waltz",
                                                          "Tango",
                                                          "Foxtrot",
                                                          "Viennese Waltz"),
                      "Order"=>3),
                array("Name"=>"Standard","Family"=>"International",
                      "Abbreviation"=>"I","Dances"=>array("Waltz",
                                                          "Tango",
                                                          "Viennese Waltz",
                                                          "Foxtrot",
                                                          "Quickstep"),
                      "Order"=>1),
                array("Name"=>"Latin","Family"=>"International",
                      "Abbreviation"=>"l","Dances"=>array("Cha Cha",
                                                          "Samba",
                                                          "Rumba",
                                                          "Paso Doble",
                                                          "Jive"),
                      "Order"=>2),
                array("Name"=>"Rhythm","Family"=>"American",
                      "Abbreviation"=>"A","Dances"=>array("Cha Cha",
                                                          "Rumba",
                                                          "Swing",
                                                          "Mambo",
                                                          "Bolero"),
                      "Order"=>4)#,
#               array("Name"=>"Nightclub","Family"=>"American",
#                     "Abbreviation"=>"A","Dances"=>array("Hustle",
#                                                         "Night Club 2-Step",
#                                                         "Merengue",
#                                                         "Salsa"),
#                     "Order"=>5)
                );
$events=array(
              array("Name"=>"Newcomer American Cha Cha","EventNum"=>1),
              array("Name"=>"Newcomer American Swing","EventNum"=>2),
              array("Name"=>"Newcomer American Foxtrot","EventNum"=>3),
              array("Name"=>"Newcomer American Tango","EventNum"=>4),

              array("Name"=>"Beginner American Cha Cha/Rumba","EventNum"=>5),
```

```
                array("Name"=>"Beginner American Swing","EventNum"=>6),
                array("Name"=>"Beginner American Waltz","EventNum"=>7),
                array("Name"=>"Beginner American Tango/Foxtrot","EventNum"=>8),

                array("Name"=>"Advanced American Cha Cha/Rumba/Swing","EventNum"=>9),
                array("Name"=>"Advanced American Bolero/Mambo","EventNum"=>10),
                array("Name"=>"Advanced American Waltz/Tango","EventNum"=>11),
                array("Name"=>"Advanced American Foxtrot/Viennese Waltz",
"EventNum"=>12),

                array("Name"=>"Open American Cha Cha/Rumba/Swing/Bolero/Mambo",
"EventNum"=>13),
                array("Name"=>"Open American Waltz/Tango/Foxtrot/Viennese Waltz",
"EventNum"=>14),

                array("Name"=>"Newcomer International Rumba",
"EventNum"=>15),
                array("Name"=>"Newcomer International Samba",
"EventNum"=>16),
                array("Name"=>"Newcomer International Waltz",
"EventNum"=>17),
                array("Name"=>"Newcomer International Quickstep",
"EventNum"=>18),

                array("Name"=>"Bronze International Cha Cha/Rumba",
"EventNum"=>19),
                array("Name"=>"Bronze International Samba",
"EventNum"=>20),
                array("Name"=>"Bronze International Waltz/Quickstep",
"EventNum"=>21),
                array("Name"=>"Bronze International Tango",
"EventNum"=>22),

                array("Name"=>"Silver International Cha Cha/Rumba",
"EventNum"=>23),
                array("Name"=>"Silver International Samba/Jive",
"EventNum"=>24),
                array("Name"=>"Silver International Waltz/Quickstep",
"EventNum"=>25),
                array("Name"=>"Silver International Foxtrot/Tango",
"EventNum"=>26),

                array("Name"=>"Gold International Cha Cha/Rumba/Samba/Jive",
"EventNum"=>27),
                array("Name"=>"Gold International Waltz/Tango/Foxtrot/Quickstep",
"EventNum"=>28),
                array(
"Name"=>"Collegiate_Open International Cha Cha/Rumba/Samba/Paso Doble/Jive",
"EventNum"=>29),
                array(
"Name"=>"Collegiate_Open International Waltz/Tango/Viennese Waltz/Foxtrot/Quickstep",
"EventNum"=>30),
                array(
"Name"=>"Open International Cha Cha/Rumba/Samba/Paso Doble/Jive",
"EventNum"=>31),
                array(
"Name"=>"Open International Waltz/Tango/Viennese Waltz/Foxtrot/Quickstep",
"EventNum"=>32));
```

?>

setup/packages.inc

```
<?
#$feeCategories["Student"]="Anyone attending school full-time, working
towards a degree is considered a student, regardless of age.  K-12 are
working towards a High School Degree";
#$feeCategories["Adult"]="Anyone not eligible for Student status is an Adult,
#regardless of age.";
$feeCategories["General"]="One price fits all";
$fee_default="General";


#$packages["Newcomer"]=array("Description"=>
#"This package is availible only to competitors competeing only in the newcomer
#events.  The beginner team match and any fun dances are also open to holders
#of this package.  The registration package will not allow you to select non-
#newcomer events if you choose this package.  This package is offered at the
#low price of \$10 to all competitors.",
#                            "LevelsPerDance"=>1,
#                            "LevelsOverall"=>"",
#                            "IgnoreEvents"=>array(),
#                            "ExcludeLevels"=>array(),
#                            "StyleLevels"=>array(),
#                            "AllowOnlyLevel"=>TRUE,
#                            "WhichLevelOnly"=>"Newcomer",
#                            "BreakWithAltRole"=>FALSE);
$packages["Regular Competitor"]=array("Description"=>
"This is the package used by all competitors at this competition.  Holders
of this package are allowed to register for upto one level per dance for
as many dances as they wish.  Competitors must still obey eligibity
(YCN/USABDA proficiency points.  Regular Competitors are eligible to compete
in any team matches or fun dances offered.  This package is NEED_COST.",

                                        "LevelsPerDance"=>1,
                                        "LevelsOverall"=>"",
                                        "IgnoreEvents"=>array(),
                                        "ExcludeLevels"=>array(),
                                        "StyleLevels"=>array(),
                                        "AllowOnlyLevel"=>FALSE,
                                        "WhichLevelOnly"=>"",
                                        "BreakWithAltRole"=>FALSE);
$package_default="Regular Competitor";

$packagePrices["Regular Competitor"]["General"]="0";
```

Bibliography

[1] Tim Converse and Joyce Park. *PHP 4 Bible.* IDG Books, 2000.

[2] Elizabeth Dew, 2000-2002. Meetings, discussions, email correspondence.

[3] Warren J. Dew, 2000-2002. Meetings, discussions, email correspondence.

[4] Rusty Russel (ed) and Daniel Quinlan (ed). *Filesystem Hierarchy Standard – Version 2.2 final.* Filesystem Hierarchy Standard Group, 2001. http://www.pathname.com/fhs/.

[5] Luis Argerich et al. *Professional PHP4.* Wrox, 2002.

[6] James R. Groff and Paul N. Weinberg. *SQL: The Complete Reference.* Osborne:McGraw-Hill, 1999.

[7] Ian Jackson and Christian Schwarz. *Debian Policy Manual.* Debian, 1998. http://www.debian.org/doc/debian-policy/.

[8] David Leung, 2000-2002. Meetings, discussions, email correspondence.

[9] Elizabeth Nugent, 2000-2002. Meetings, discussions, email correspondence.

[10] Thomas J. Nugent, 2000-2002. Meetings, discussions, email correspondence.

[11] Discussions on http://forums.devnetwork.net.

[12] USA DanceSport Rulebook, 2002. http://www.usabda.org/usabda/pdf/2002-2003_USA_DanceSport_Rulebook_jan02.pdf.

[13] John C. Worsley and Joshua D. Drake. *Practical PostgreSQL.* O'Reilly, 2002.