Web Services @ MIT

by

Sapna DevendraSingh Tyagi

Bachelor of Engineering in Civil Engineering, National Institute of Technology, Rourkela (2000)

Master of Science in Operations and Information Technology, WPI (2003)

Submitted to the Department of Civil and Environmental Engineering
in partial fulfillment of the requirements for the Degree of

MASTER OF ENGINEERING IN
CIVIL AND ENVIRONMENTAL ENGINEERING

AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
JUNE 2004

Signature of Author:..
Sapna DevendraSingh Tyagi
Department of Civil and Environmental Engineering
May 7, 2004

Certified by:............................. ....
John R. Williams
Associate Professor, Department of Civil and Environmental Engineering
Thesis Supervisor

Certified by:..
Alfred Essa
Director, Sloan Technology Services, Sloan School of Management
Thesis Reader

Accepted by:..
Heidi Nepf
Chairman, Department Committee of Graduate Studies

Web Services @ MIT

By
Sapna DevendraSingh Tyagi

Submitted to the Department of Civil and Environmental Engineering
On May 17th, 2004 in Partial Fulfillment for the Degree of
Master of Engineering in Civil and Environmental Engineering

Abstract

There are several useful web services developed at MIT by students, faculty and researchers. However, they are scattered all over MIT. Most people at MIT are unaware of the availability of these web services and hence they cannot leverage them.

The thesis provides a solution to this problem called Web Services @ MIT that tracks all the web services scattered over MIT at a single location, with a brief description of the service they provide. Web Services @ MIT is intended to facilitate exploitation of both centrally and locally developed web services in a highly distributed enterprise such as MIT. While there is a compelling need for such a service, no such service is currently available at MIT. Web Services @ MIT is a portal built on top of IBuySpy Portal Solution Kit that allows one to build intranet and Internet application using ASP.NET along with Microsoft .NET Framework.

Thesis Supervisor: John R. Williams
Title: Associate Professor, Department of Civil and Environmental Engineering

Thesis Reader, Alfred Essa
Title: Director, Sloan Technology Services, Sloan School of Management

Acknowledgements

I am very grateful to Professor John Williams and Al Essa for their constant guidance and invaluable advice on my thesis. My sincerest gratitude to them for extending enormous support during my illness.

Many thanks to Dr. Eric Adams for his unremitting support throughout my M.Eng program.

I would also like to express my gratitude to Andrew Grumet for his insightful comments and encouragement during my thesis. Thanks to Genevieve for writing a white paper to put up as a sample document on my website. Thanks to Tracy and everyone else in STS for a memorable work experience in the STS penthouse.

I would like to thank my M.Eng class-mates and project-mates for a wonderful learning experience. It was fun doing those all-nighters in the M.Eng lab, working on the thesis and projects.

I would also like to extend my heart-felt acknowledgements to Cynthia Stewart and Jeanette Marchocki for being extremely supportive and going out of their way to help me complete my thesis during my illness.

My acknowledgements cannot be complete without extending my gratitude to my parents and family for their blessings and encouragement and for always being there for me. Without their support I wouldn't be where I am today. Last but not the least, I want to thank all my friends and suite-mates, especially Aarthi, Richa and Karolina for standing by me both during my happy and difficult times and making my stay at Ashdown like at home away from home.

TABLE OF CONTENTS

List of Figures

CHAPTER 1: INTRODUCTION AND THESIS SUMMARY

This chapter describes the motivation for building (Web Services @ MIT) and summarizes each chapter of the thesis.

1.1 Introduction and Problem Definition

Application integration is a major technology issue facing businesses these days. Application integration provides tactical value to businesses by improving operational efficiency and reducing costs. It also provides strategic value to businesses by enabling better access to information, better decision-making, and tighter integration of business processes in areas such as supply chain management. Integrating heterogeneous systems within and across organizations is one of the "holy grails" of information technology and it is now widely accepted that web services (Refer to Section 2.3 for details about the technology) will be one of the key technologies in this solution space. Indeed, web services are rapidly becoming a core part of mainstream information technology architectures, particularly in the corporate world.

It is important to note, however, that currently most web services projects originate and stay within Information Technology organizations and, therefore, their application has been restricted to "centralized" models of information delivery and information exchange. It is our assertion and thesis that the promise of web services will not be fully realized until they move beyond central IT organizations and there are well established institutional mechanisms for both sharing and consuming "locally" developed web services.

Higher educational institutions, particularly research universities, by nature are highly distributed organizations where innovation in learning, research, and practice emerges typically from local departments, professional schools, research centers and laboratories, and even individual faculty and researchers.

Use of web services technology is gaining momentum in educational institutions as well but unlike in the corporate world there is more fervent, activity, and innovation at the "edges". Web services technology is becoming a part of MIT through course work, research labs like IESL, educational technology initiatives at MIT's Sloan School of Management, and large-scale projects such as ICampus and DSpace. Being a student in Information Technology and learning about web services through my course work and research, I came to realize that there are several useful web services at MIT, in many cases locally developed, by students, faculty and researchers. However, the MIT community cannot leverage them because they are unaware of their existence. Web Services @ MIT is intended to facilitate exploitation of both centrally and locally developed web services in a highly distributed enterprise such as MIT.

Web services can be described, advertised and discovered on the Web using an XML-based language like WSDL and listing them in an XML-based registry such as UDDI (Refer Section 2.4 for details about UDDI). UDDI is the industry standard for registering and discovering web services. But its heavy technology footprint makes it difficult to understand, maintain, and support. Moreover, it does not fit the MIT-type distributed enterprise where innovation occurs at the edges and not at the center.

A directory of web services must also provision for assessing the quality of the service provided. UDDI also has a lot of useless web services registered, making it "costly" in an information sense to find quality services. Finally, UDDI's heavy footprint makes it less amenable to creating a self-managed web site that operates on community-based principles for sharing and evaluating information and services. There is a compelling need, therefore, for a service at MIT that makes it easier for people to track, access and assess web services.

This motivated me to build a web site called Web Services @ MIT, which could serve as a "Web Services Registry" for MIT. Unlike UDDI, Web Services @ MIT has a smaller technology footprint, making it comparatively easier to support and maintain. From a user perspective, Web Services @ MIT is easy to understand and navigate. The technical architecture for the site also provides a robust foundation for creating an online community around each service and content area. Thus, in addition to rating each service, community tools would allow a richer feedback loop to the service provider in terms of comments as well as sharing of best practices between users.

Web Services @ MIT is a portal built on top of IBuySpy framework, where people from all over MIT could list/register their services at a single accessible location, with a brief description of the service they provide, maturity of the service as per the service provider and average ranking of the service given by the users. Web Services @ MIT has a potential to become a powerful online community service at MIT that would promote the use of web services in education improve the quality of student life at MIT.

Like any other new application, it will be a challenge for Web Services @ MIT to gain an initial recognition. To begin with, this process can be triggered with the following steps:

- People can be encouraged to list their web services, URLs on Web Services @ MIT through word-of-mouth and mass-emails, by personally contacting friends, colleagues and by encouraging them to get their web services up and running.
- People can be encouraged to update this information through reminders (List/de-list services)
- People can be encouraged to build applications on top of these services and write/publish white papers to share their experiences.

Currently, there already are a few people from the MIT Community, who are willing to list their web services on this site/system. There are two major issues that the system would face once it gains initial recognition. One is sustainability and other is security concern that would be raised by exposing the real web services to the public. These issues have been discussed in detail in Chapter 5.

The purpose of this system is not to compete with players like UDDI. This system does not cater to the needs of people/groups that are experts in web services or are involved with web services on a large scale. This system caters to the needs of people from MIT community, who are new to web services and want to learn more about consuming web services or people who simply wish to share their web services for the betterment of the MIT community, or people who want to use some existing web services and have the skills and understanding to build applications that would consume these services to cater to their specific needs.

This thesis focuses on building a simple and an easily available solution that would change the way how people think about using web services at MIT. It is not just about Technology – it is about technology associated with people, about how people can use a simple technology that can create a very positive impact and add a tremendous value to the MIT community provided it is successful. Once again, like any other software application, the surety of success for this application is also uncertain. However, something like this has never been tried before and I strongly believe that it is worth a try considering the value it would add if it proves to be as successful as expected.

1.3 Thesis Summary

After the introductory chapter, ***Chapter 2*** focuses on the technologies used in building the Web Services @ MIT Portal like ASP.NET and .NET Framework, ADO.NET and IBuySpy and the technologies relevant to the portal like Web Services and XML-based registries like UDDI. All the technologies herein have been discussed briefly and are meant to provide only an introduction to the system-specific implementation. It has been assumed that the readers will rely on external sources for additional information and in-depth understanding of these technologies. Some of these sources are listed in chapter 6 (References).

Chapter 3 discusses in detail the design aspects of the portal. This includes the data model using SQL Server, process model in order to understand the process flow on the site and User Interface diagram which explains the front-end user interface in detail.

The physical data model has been built using SQL Server. For building the data model of this system, a few existing tables of IBuySpy framework like Users, Roles, User Roles, ModuleDefinitions, Modules, ModuleSettings, Links, Tabs, Events, HtmlText, Documents, Portals have been customized. This data layer has been extended by creating additional tables like WebService, StatusDetails, MaturityDetails, WSUsage and WSRanking.

The process model consists of data flow diagrams that describe flow of data in various processes in the system. These include the processes like Register User, Log in User, Add/Approve Web Service, Rate a Web Service, Add Admin, Log in Admin and Edit/Delete Web Service, Add/Edit/Delete Documents.

The User Interface diagram displays and explains how the site would appear to the users. It also describes all the features available to the user at the front-end. The screens shots of user interface are included in Appendix A.

Chapter 4 focuses on the architecture of the portal frame work, features, existing modules and custom modules, roles, privileges and security.

The portal has been built on physical 2-Tier architecture, which is often used for distributed web applications and is also the existing architecture of the IBuySpy application. These 2-Tiers include the following layers:

A combination of Database Access Layer and Business Logic layer, which includes the database, stored procedures and the components that allow the application to talk to the database and also the components that contain the business logic of the application.

- Presentation Layer, which contains the ASP.NET web forms or the web application pages.

The application development is based on the code-behind concept, which separates the Logic (code) from UI (HTML).

For building the portal, some of the existing IBuySpy modules like Sign-in/Register, QuickLaunch, HTML/Text Document, Upcoming events, Image have been customized to meet the needs of the portal. In addition, new modules are being developed like List Web Services, Add/Edit/Delete Web Services and Search Web Services.

The portal supports two types of Roles: users and admins. The admins would be able to see the admin tab and would be able to approve, delete any web service and delete or change the role of any user. The users would not be able to see the admin tab but would be able to add, edit or delete only their own web services. The portal uses the authorization and authentication security features of the IBuySpy framework.

Chapter 5 discusses ways to address some security concerns like exposure of web services to unauthorized users and maintenance issues like keeping the system alive. The system would be low maintenance, simple, contain easy-to understand code using .NET and would just require a moderator's time and effort to keep the server up and running.

This chapter also includes an in-depth analysis of the future applications of this portal. It mainly focuses on analyzing the scalability of the portal that would increase the use of web services for educational purposes and for improving the quality of student life at MIT. It introduces some of the web services and applications completed, in progress or being considered at MIT like ShuttleTrack, DSpace, and Events Calender at MIT to give an idea of the future applications of this system at MIT.

It could also be extended to provide an environment where people could not only write/publish white papers related to web services but also share their experience of building applications that consume the web services listed on this portal through discussion forums.

Chapter 6 cites all the references used for writing this thesis and also lists additional resources about web services, technologies used in this thesis and applications that can be built to consume the web services listed on this portal.

APPENDIX A provides additional information to the readers by providing few screen-shots of the system and the work in progress.

CHAPTER 2: TECHNOLOGY

2.1 ASP.NET

ASP.NET is a programming framework built on the common language runtime that can be used on a server to build powerful Web applications. As compared to previous Web development models, ASP.NET offers several important advantages like enhanced performance, cleaner code, power and flexibility, improved deployment, reliability, simplicity, manageability, scalability and availability, customizability and extensibility and Security. ASP.NET includes features like compiled language support, an object model for the page, Server Controls, Web Services, xcopy deployment, new configuration features, new session state options, new caching, new authentication/authorization options, a new architecture (Refer Figure 2.1), and thus enables a new breed of web applications and provides better support for different browsers and devices.

ASP.NET technology runs simultaneously with the existing ASP infrastructure. Unlike traditional ASP, ASP.NET supports event-handling. There are different filename extensions for different type of web pages in ASP.NET that would enable IIS to call the appropriate ISAPI filter to handle processing. For example, a basic ASP.NET page, also known as a web form has an extension .aspx, whereas the ASP.NET page for web services uses the extension .ascx. The events get raised on the client and processed on the server. These events do not automatically cause a post back to the server, server controls have a built-in set of events like button click that automatically cause a post back to the server. Other events are cached until a button click even causes a post back to the server. Then, the other events as well as the button click event are processed on the server.

Most ASP pages have a combination of scripting code diffused with HTML elements. However, in ASP.NET, the server code (scripting code) can be separated from the presentation code within a single <SCRIPT RUNAT= "SERVER">....... </SCRIPT> block or placed within a code-behind page. This makes the code cleaner by avoiding the spaghetti code. ASP.NET also has controls called server controls and user controls. The server controls have tags like <asp:textbox> and the server-side code interacts with these controls, producing standard HTML that is sent to the web browser. User Controls are self-contained visual elements that can be placed on a web page any other web controls and are used for code-reuse in ASP.NET.

2.1.1 ASP.NET Architecture[1]

ASP.NET is managed by information stored in XML-format in a configuration file (Web.Config). The cache allows for improved performance of ASP.NET, as the most commonly requested pages would be served from the ASP.NET cache. State management services for ASP.NET are provided by the ASP.NET state service. The .NET Framework provides the Common Language Runtime (CLR), which compiles and manages the execution of ASP.NET code, and the class libraries, which offer pre-built programmatic functionality for Web Forms, XML support, and exception handling. ADO.NET provides ASP.NET with connections to databases. Figure 2.1 below displays the ASP.NET Architecture.

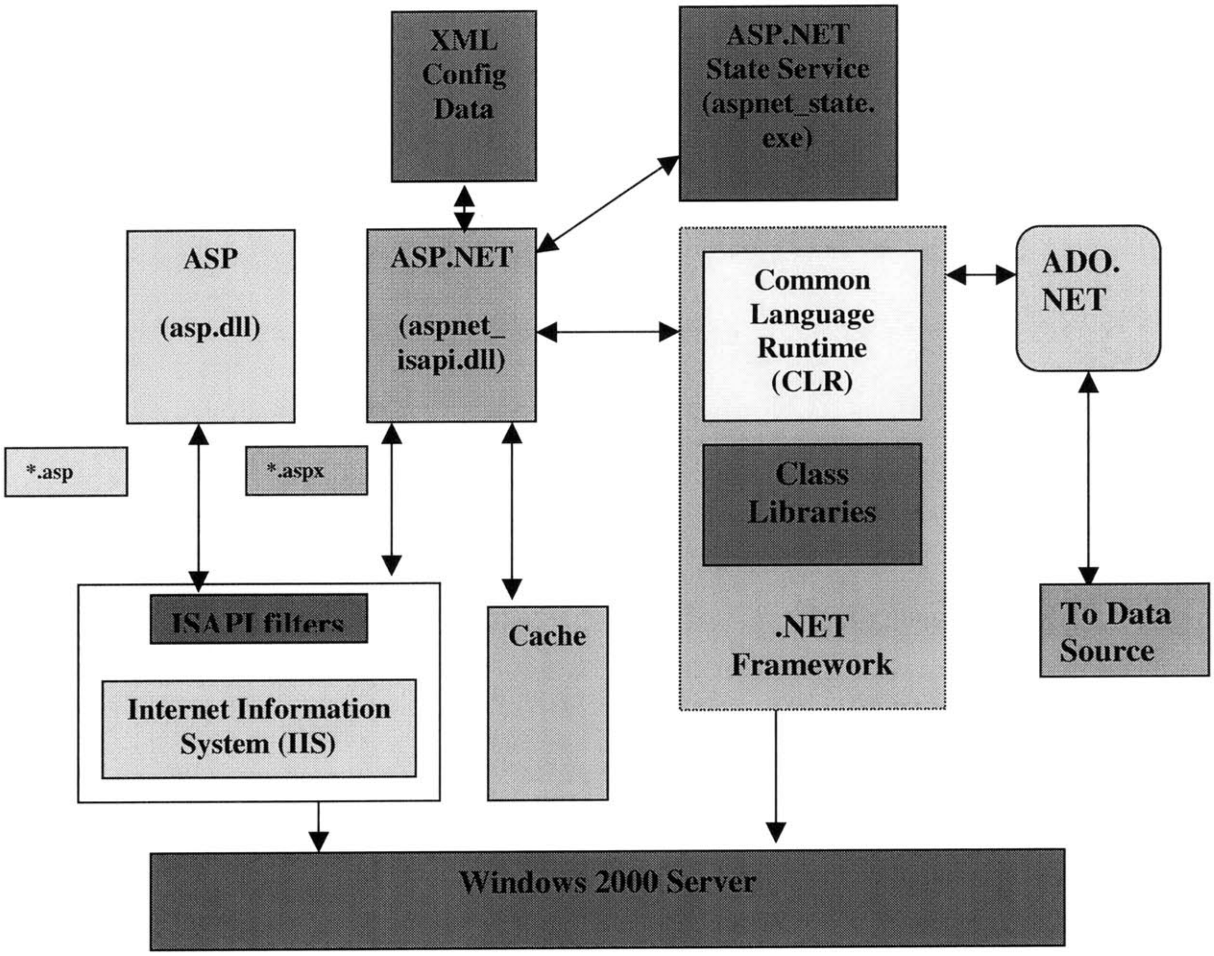

XML Config Data
ASP.NET State Service (aspnet_state.exe)
ASP (asp.dll)
ASP.NET (aspnet_isapi.dll)
Common Language Runtime (CLR)
Class Libraries
.NET Framework
ADO. NET
*.asp
*.aspx
ISAPI filters
Internet Information System (IIS)
Cache
To Data Source
Windows 2000 Server

Figure 2.1: ASP.NET Architecture

2.2 ADO.NET

ADO.NET is comprised of classes in System.Data namespace that encapsulate data access for distributed applications. However, rather than simply mapping the ADO object model to .NET to provide a managed interface to OLEDB and SQL Server, ADO.NET changes the way data is stored and marshaled within and between applications. This allows the applications to leverage on the scalability and flexibility of being able to distribute data across the Internet in a distributed and disconnected fashion.

2.3 Web Services[2]

A Web Service is programmable application logic accessible using standard Internet protocols. Web Services combine the best aspects of component-based development and the Web. Like components, Web Services represent black-box functionality that can be reused without worrying about how the service is implemented. Unlike current component technologies, Web Services are not accessed via object-model-specific protocols, such as DCOM, RMI, or IIOP. Instead, Web Services are accessed via ubiquitous Web protocols (i.e.: HTTP); data formats (i.e.: XML or Extensible Markup Language); and SOAP (Simple Object Access Protocol), an XML-based object invocation protocol. Web Services technology provides the framework in which to build "small modules which perform analytical tasks and are linked to other modules and data sets." Web Services make software functionality available over the Internet so that programs like PHP, ASP, JSP, JavaBeans, the COM object can make a request to a program running on another server (a Web Service) and use that program's response in a website, WAP service, or other application.

Web Services are "Libraries" providing data and services to other applications over Web through a consistent set of interfaces and protocols. In that sense, Web Services are a new breed of Web application. They are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web Services perform functions, which can be anything from simple requests to complicated business processes. Once a Web Service is deployed, other applications (and other Web Services) can discover and invoke the deployed service. Web Services is a term that is being used to define a set of technologies that exposes business functionality over the Web as a set of automated interfaces. These automated interfaces allow businesses to discover and bind to interfaces at run-time, supposedly minimizing the amount of static preparation that is needed by other integration technologies.

To summarize, a Web Service has 5 characteristics:

- Firstly, web services are *reusable software components.* Rather than requiring programmers to write one start-to-finish set of instructions after another, the component-based model allows developers to reuse the building blocks of code created by others to assemble and extend them in new ways.
- Secondly, these software components are *loosely coupled.* Traditional application design depends upon a tight interconnection of all subsidiary elements. Consequently, it is exceedingly difficult to extract one element and replace it with another. Loosely coupled systems, on the other hand, require a much simpler level of coordination and allow for more flexible reconfiguration.
- Thirdly, web services *semantically encapsulate discrete functionality.* A web service is a self-contained "applet" that performs a single task. The component describes its own inputs and outputs in a way that other software can determine what it does, how to invoke its functionality, and what result to expect in return.
- Fourthly, web services can be *accessed programmatically.* Unlike web sites and desktop applications, web services are not designed for direct human interaction, and they do not have a graphical user interface. Rather, web services operate at the code level; they are called by and exchange data with other software.

- Finally, web services are *distributed over the Internet.* Web services make use of existing, ubiquitous transport protocols like HTTP. They are thus able to leverage existing infrastructure and can comply with current corporate firewall policies.

2.3.1 Web Services Core Technologies

What is most important is that all Web Services strategies use a well-known and widely implemented Internet protocol for communication—HTTP—the foundation upon which all Web sites operate. The advantages of this arrangement are numerous. The most obvious is that most organizations already have a Web infrastructure in place, so implementing Web Services can be handled in a familiar way, and the wealth of Web software can be used to develop and run new Web Service-based applications. The other important aspect of Web Services is that they use XML for passing messages between computers, preserving the transparency that has made XML so popular and useful (although some implementations, most notably those promoted by Microsoft in their .NET framework, often still hide the actual message content (data) in a non-human-readable-format).

The second core component of web services is XML or Extensible Markup Language. XML is the most important technology since the advent of the Web. XML is really nothing by itself. It is simply a framework in which to write languages for data encoding and system-to-system messaging. What XML provides is a consistent language structure and a way of describing the language's content in the form of XML Schema. For example, the use of XML as an encoding and messaging language allows spreadsheet, database, CADD and GIS packages to communicate with each other. Because structure and content are described in XML syntax, the software industry has built powerful, reliable tools to read XML on every operating system and application in common use. It is also very important that XML languages are plain text, so that their content is transparent to humans, even in the absence of computer programs that can read and manipulate the XML. This has a profound effect on people's trust in the content and in the ability of the content to be used in almost all current and future computing environments.

A key component of web services is XML or Extensible Markup Language. XML is the most important technology since the advent of the Web. XML is really nothing by itself. It is simply a framework in which to write languages for data encoding and system-to-system messaging. What XML provides is a consistent language structure and a way of describing the language's content in the form of XML Schema. For example, the use of XML as an encoding and messaging language allows spreadsheet, database, CADD and GIS packages to communicate with each other. Because structure and content are described in XML syntax, the software industry has built powerful, reliable tools to read XML on every operating system and application in common use. It is also very important that XML languages are plain text, so that their content is transparent to humans, even in the absence of computer programs that can read and manipulate the XML. This has a profound effect on people's trust in the content and in the ability of the content to be used in almost all current and future computing environments.

However, XML by itself does not ensure effortless communication. The applications need standard formats and protocols that allow them to properly interpret the XML. Hence three XML-based technologies have emerged as the de facto standards for Web Services:[3]

- Simple Object Access Protocol (SOAP) defines a standard communications protocol for Web Services. SOAP provides a simple and consistent mechanism that allows one application to send an XML message to another application.
- Web Services Description Language (WSDL) defines a standard mechanism to describe a Web Service. A WSDL document describes what functionality a Web service offers, how it communicates and where it is accessible.
- Universal Description, Discovery and Integration (UDDI) provides a standard mechanism to register and discover Web Services. Users can query a UDDI registry based on company name, industry category, service type or other criteria. UDDI provides pointers to the WSDL document that describes a service and the access point of a service implementation.

Figure 2.2 shows how these technologies relate to one another. When a service provider wants to make the service available to service consumers, he describes the service using WSDL and registers the service in a UDDI registry. The UDDI registry will then maintain pointers to the WSDL description and to the service. When a service consumer wants to use a service, he queries the UDDI registry to find a service that matches his needs and obtains the WSDL description of the service, as well as the access point of the service. The service consumer uses the WSDL description to construct a SOAP message with which to communicate with the service. The core elements of UDDI are discussed in the next section.

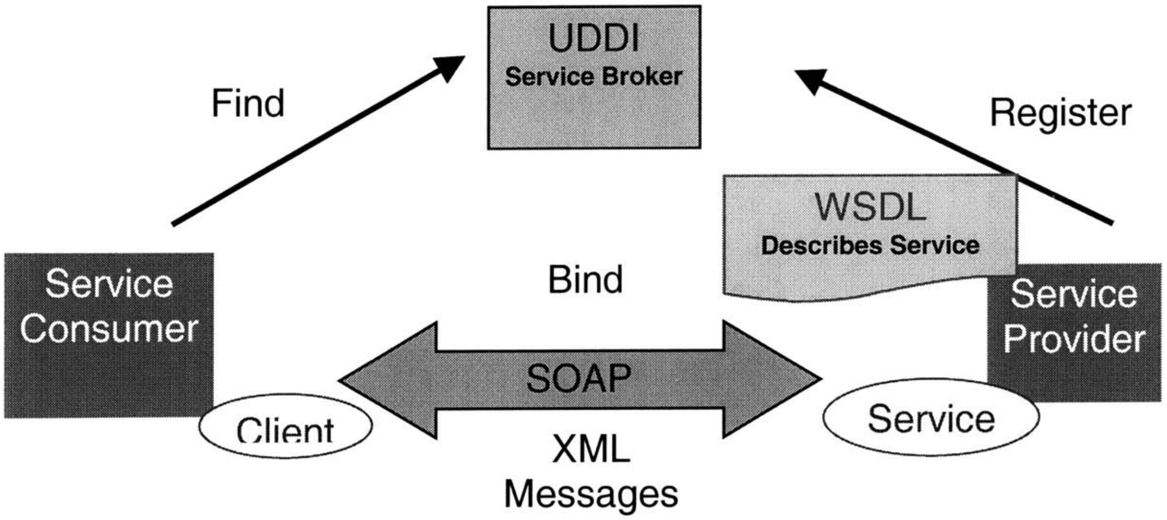


Figure 2.2: Core Web Services Technologies are SOAP, WSDL and UDDI

2.4 XML-Based Registries

Businesses can describe their Web services on the Web using an XML-based language like WSDL and listing them in an XML-based registry such as UDDI. A registry is a key component in Web services architecture, since it facilitates dynamic business-to-business interactions by allowing organizations to publish, discover, and utilize Web services.

2.4.1 UDDI[4],[5]

- UDDI is an XML Registry that allows businesses to register themselves and their services in different categories like yellow pages, contain the contact information of these businesses like white pages and contain technical details about the services offered by business entities like green pages.

The UDDI information model contains four core elements:

- businessEntity element that contains information like name, description and contacts of an organization and enables searches that can locate these organization categorically.
- businessService element that groups together related services offered by an organization.
- bindingTemplate element that contains information on how to invoke a web service. This information can be in the form of WSDL or plain-text.
- tModel element that has information about specifications for service like name, publishing organization and URL pointers to the actual specifications.

The APIs for UDDI are divided into two logical parts: the *Publish* API and the *Inquiry* API that describe the Simple Object Access Protocol (SOAP) messages that are used to publish and discover an entry in the registry.

- The Publish API provides methods for publishing and updating information contained in a UDDI registry. There are methods for saving and deleting the information available from UDDI information model. They require authorization for being invoked and are invoked through https.
- The Inquiry API provides methods for querying the registry.

2.4.2 XMethods[6]

XMethods is a site that lists publicly available web services and provides services that facilitate the development, deployment, and usage of web services and web service networks. These web services are generally contributed by third-party service providers. The site lists the publisher's name and service name as well as a brief description of the service.

Web Services @ MIT is similar to XMethods, however it caters specifically to the needs of the MIT Community.

2.5 IBuySpy[7],[8]

IBuySpy Developer Solution Kits are application samples built on the ASP.NET Technology. These samples serve as demo applications and demonstrate how to use the best practices in ASP.NET along with .NET Framework to develop Internet and Intranet applications. Currently the IBuySpy Solution kit has two sample applications – IBuySpy Store Solution Kit and IBuySpy Portal Solution Kit. Developers can use this framework as a starting point for their own applications and thus can leverage upon and learn to use a number of coding standards and best practices in ASP.NET, while understanding some of the complicated features of ASP.NET. Web Services @ MIT is being built on top of IBuySpy Portal Solution Kit. IBuySpy Store is an e-commerce shop containing shopping basket, account management and custom comments. IBuySpy Portal is a module-based, multi-tabbed, configurable portal. Both Portal and the Store Solution Kits are downloadable from the http://ibuyspy.com website and are available in four versions:

- SDK Version, C#
- SDK Version, Visual Basic.NET
- Visual Studio.NET Version, C#
- Visual Studio.NET Version, Visual Basic.NET

For installing these solution kits on a machine, the pre-requisites are Microsoft.NET Framework, SQL Server 2000 or Microsoft Data Engine 2000, Windows XP Professional or Windows 2000

(Professional, Server or Advanced Server) or Windows XP. The Portal solution kit also requires Microsoft Mobile Internet Toolkit. Every installation package contains complete source code in C# or Visual Basic.NET, Documentation, Source Code Viewer and HTML Application setup Package.

2.4.1 Key Features of IBuySpy[9]

IBuySpy has end to end samples that demonstrate how to build real-world ASP .NET applications, which provide a great starting point for building one's own applications. IBuySpy Solution reveals the simplicity of building applications using ASP.NET Technology. It uses ADO.NET that automatically creates and retains database connections and avoids the need to use COM+ in a distributed application. Instead of using Business Objects, IBuySpy uses optimized design where the .aspx pages communicate with the database through a set of database classes. Each table in the database that is used by the application is associated with a separate source code file, which includes an entity class and a stateless service provider class. User Interface functionality is handled by the User/Custom Controls. For example the menu navigation is organized by User controls and the same controls are used repetitively on multiple pages.

CHAPTER 3: DESIGN OF THE WEB SERVICES @ MIT PORTAL

3.1 Process Model[10]

A process model describes business processes or how a business system operates and illustrates the activities that people would do while using the system. Data flow diagramming is one of the techniques of process modeling, in which business processes and the data that passes through them are described diagrammatically. Many times data flow diagrams are a step further from use cases i.e. the information in a use case is used to draw the data flow diagrams.

3.1.1 Elements of Data Flow Diagram

Each Data Flow Diagram has four elements (processes, data flows, data stores and external entities), each of which is represented by a different graphical symbol. There are two styles of symbols, one is Gane and Sarson symbol and other is DeMarco and Yourdan Symbol. This thesis uses the Gane and Sarson symbol as described below:

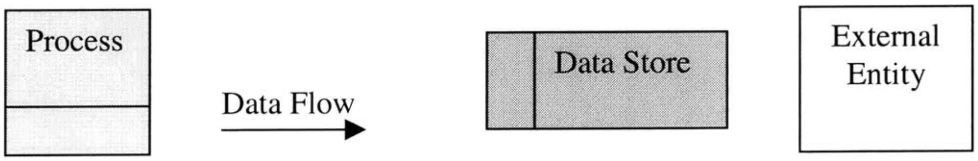


Figure 3.1: Gane and Sarson Symbol for Data Flow Diagrams

Process: A process is a function or activity performed for a business purpose.

Data Flow: A data flow is the information or data elements that connect processes. One end of data flow will always either come out of the process or go into a process.

Data Store: A data store is a collection of data is forms the starting point of the data model.

External Entity: An external entity is a person, organization or system that interacts with the new system but is external to the system.

This section describes the major data flow diagrams developed during the design of Web Services @ MIT system.

1. ***User Registration and Login Process:*** This data flow diagram (DFD) describes the registration and login of a new or existing user on Web Services @ MIT site. If the user is a new user, he would use the Register user process. Information like name, email ID and password form the data flow for this process of the data flow diagram. These data elements are stored in the data store called user/admin info. User is the external entity for this process. If the user is already registered or is an admin, he would use the Login User/Admin process. User name and password are the data flows for this process and are stored in and verified from the User info/admin info data store.

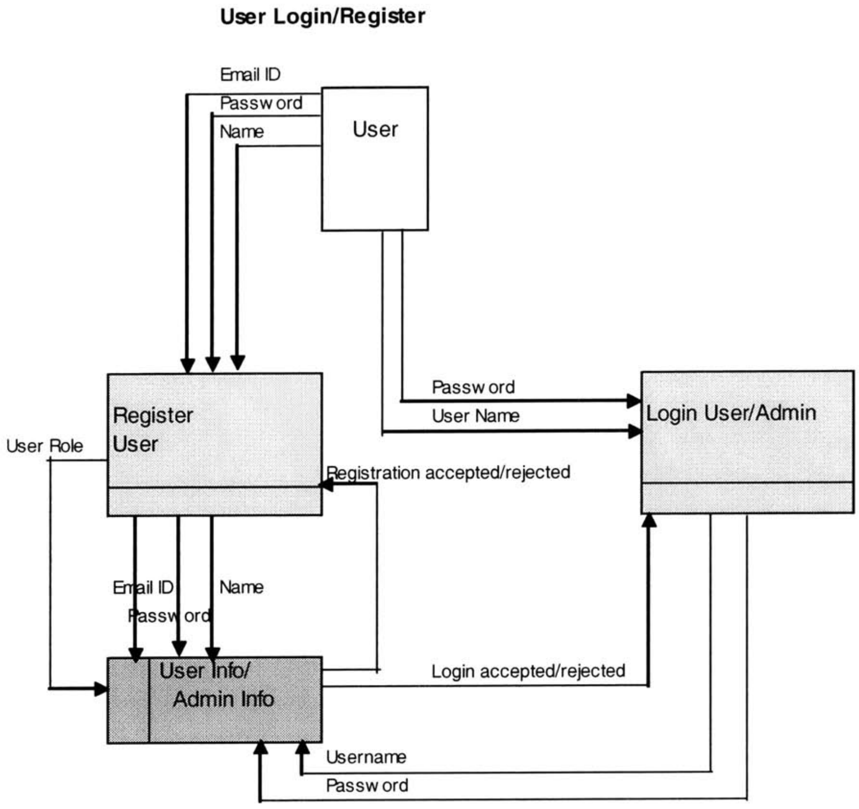


Figure 3.2: DFD for User Login/Registration

*2. **Process of adding and approving new web service:*** This data flow diagram describes the process of registering new web service by a user (service provider) and approval of the service by the user with an admin role. Web service URL, description, maturity, status are some of the input data flow elements and web service URL, description, maturity, owner, system date and time are the output data flow elements while registering a service. The data store is web service and external entity is user. For approval of the web service process, admin is the external entity, service approval is the input data flow and web service URL, description, system data & time, owner and maturity are the output data flows.

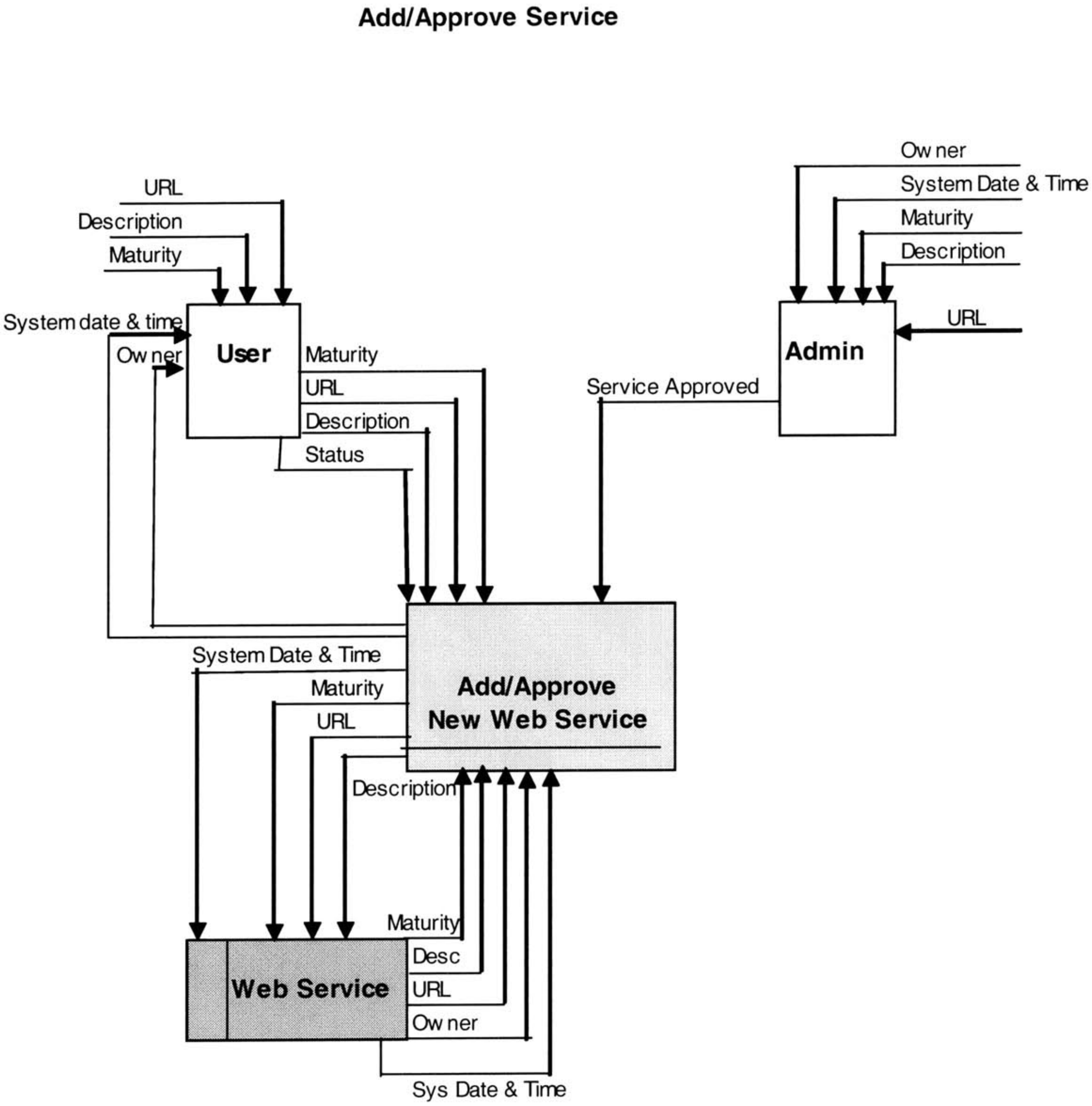


Figure 3.3: DFD for Adding/Approving a Web Service

3. Edit, delete web service: This data flow diagram describes the process of editing and deleting a web service. User is the external entity for the process of editing his own web service and admin is the external entity for the process of editing and deleting any web service. The input data flow elements are web service maturity, description, URL and delete request (in case of delete process) and the output data flow elements are owner, URL, description, maturity and deletion confirmation.

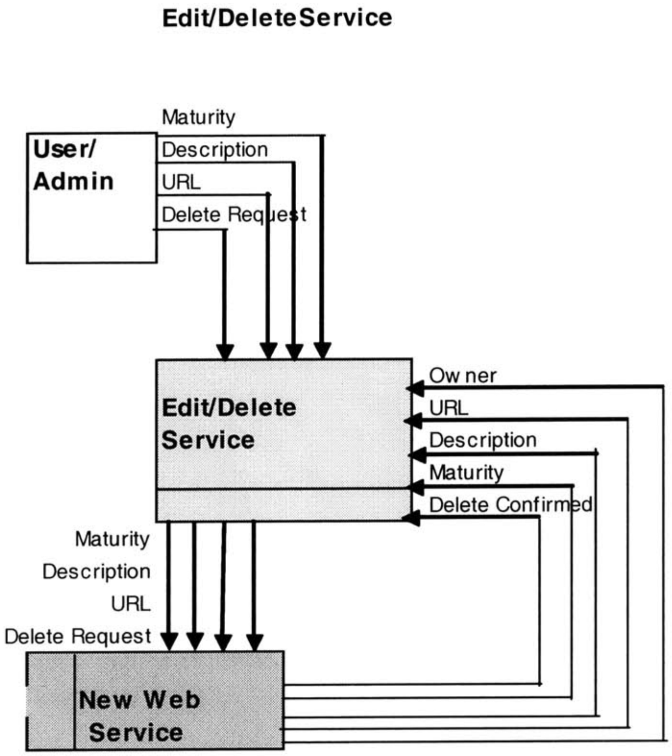


Figure 3.4: DFD for Editing/Deleting a Web Service

4. ***Process of adding, editing or deleting documents:*** This data flow diagram describes the process of adding, editing or deleting documents. Admin is the external entity and documents are the data store for the process. The input data flow elements are title, owner, description, file name URL, file content and add/delete/edit request. The output data flow elements are title, owner, file name URL, file content and add/delete/edit confirmation.

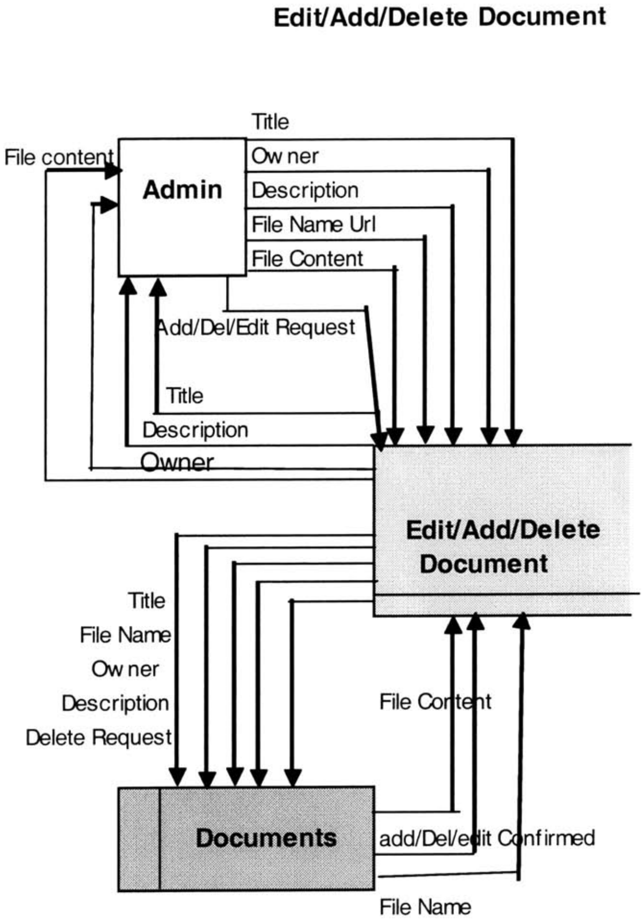


Figure 3.5: DFD for Adding/Editing/Deleting a Document

5. ***Process of rating the service:*** This data flow diagram describes the process of rating a web service. Both, user and admin are external entities for this process. The input data flow element is adding the rating (selecting the rating) and the output data flow element is average rating, which is a calculated value from the web services rating data store.

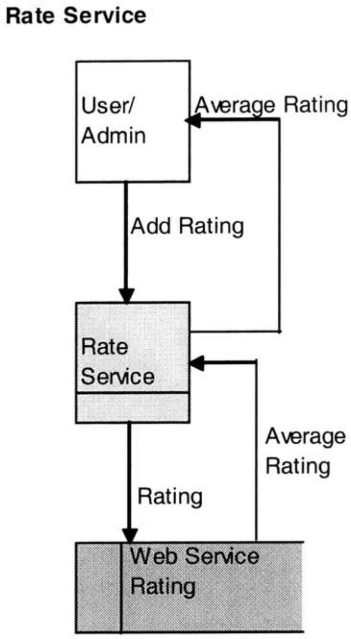


Figure 3.6: DFD for Rating a Web Service

6. Process of showing the web service usage: This data flow diagram describes the process of showing the web service usage. Both, user and admin are external entities for this process. The input data flow element is adding the usage (selecting the usage) and the output data flow element is average usage, which is a calculated value from the web services usage data store.

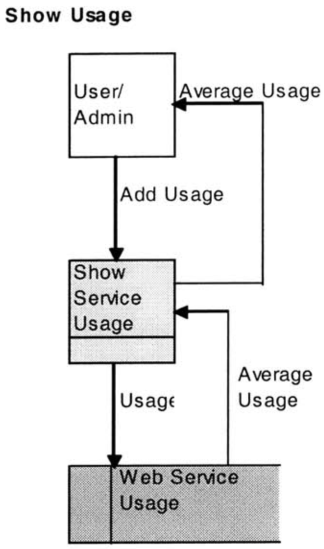


Figure 3.7: DFD for showing Web Service Usage

*7. **Process of Searching the Web Service:*** This data flow diagram describes the process of searching a web service from the web services listed on Web Services @ MIT. Both, user and admin are the external entities for this process. The input data flow elements can be either of web service name, owner, maturity, average rating and average usage and the output data flow elements are corresponding to the input data flow elements. The data stores are either of web services, web services rating and web services usage depending on the input data flow elements.

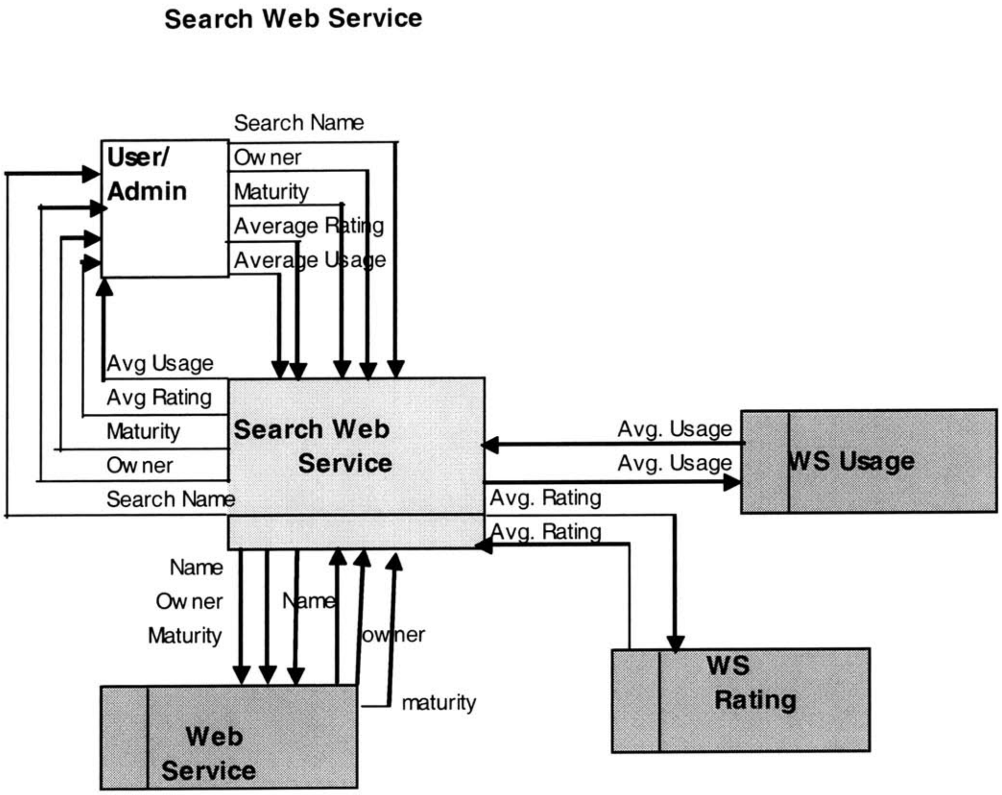


Figure 3.8: DFD for Searching a Web Service

8. Changing the role of a user: This data flow diagram describes the process of changing the role of a user. Admin is the only external entity in this process. Te input data flow elements are email id and change user role (user role that needs to be changed) and the output data flow elements are email id and new user role. The data store is user role.

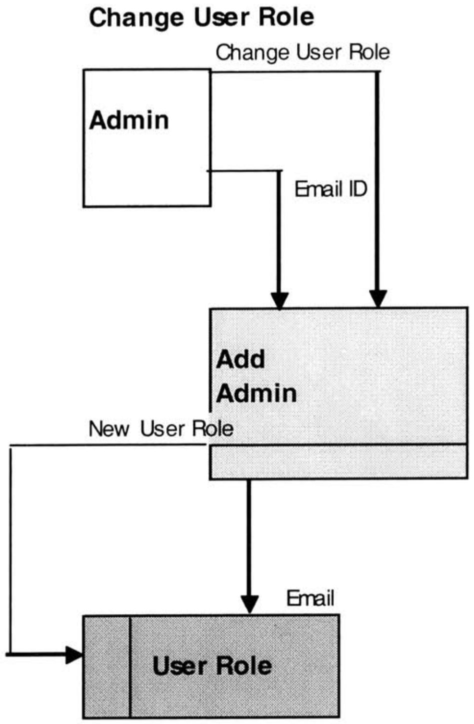


Figure 3.9: DFD for Changing the Role of a User

3.2 Data Model

This section provides an overview of the data model the database used in the Web Services @ MIT Portal. A data model describes the data used and created by a business system. One of the most popular data modeling techniques is entity-relationship diagram (ERD). There are 3 basic elements in a data model:

Entity: Building block of data model about which data is being collected.

Attribute: Information captured about the data model

Relationships: convey the association between entities

An intersection or composite entity is placed between two entities to capture information about relationships.

A SQL Server database is used for physically storing the data. The ERD or the Physical Database schema of the Web Services @ MIT portal is shown below:

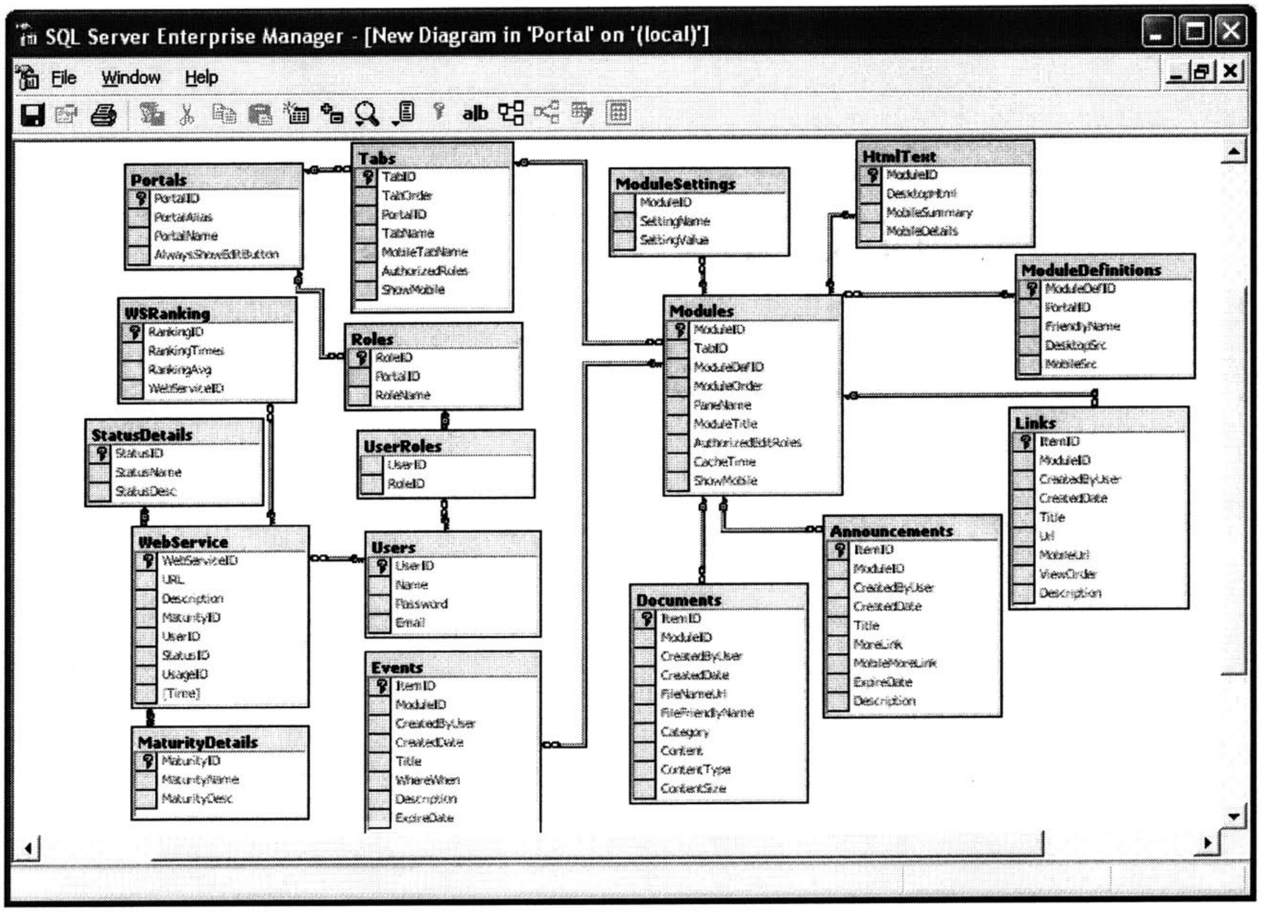


Figure 3.10: ERD for Web Services @ MIT Portal

Data Tables/Entities of the Web Services @ MIT Portal

The portal database includes the following tables from the IBuySpy database: Users, Roles, UserRoles, Portals, Tabs, Modules, ModuleDefinitions, ModuleSettings, Announcements, Documents, Events, Links, HtmlText. In addition to these there are additional tables like

WebService, MaturityDetails, StatusDetails, WSRankings, WSUsage which are an extension to the IBuySpy data model.

Users table contains the information like name, emailed, password for users and admins. The ***Roles*** table describes the number and types of roles assigned to users of the site. The ***User Roles*** table connects the user and roles table by associating a user with his role. The ***Portals*** table contains portal information like ID, Name, etc. The ***Tab*** table contains information about tabs and also has the corresponding portal id (since portal contains tabs) for each tab. The ***TabSettings*** table stores custom data for each Tab. The ***Modules*** table contains a list of module instances and the corresponding Tab ID (since tabs contain modules). The ***ModuleDefinitions*** table contains a list of all available modules. ***ModuleSettings*** table stores custom data for each Module. The ***ModuleDefinitions*** table contains a list of modules associated with a single portal. There are separate table for specific modules like ***Links, HtmlText, Announcements, Events, Documents***, which contain data about these specific modules. The ***WebService*** table contains the information like URL, owner, Description, Maturity, etc about the registered web services. The ***StatusDetails*** table contains the status of the registered web service. The ***MaturityDetails*** table contains the id, names and descriptions of the defined levels of web service maturity. The ***WSRankings*** and ***WSUsage*** tables contain the information about the rankings and usage of web services respectively.

For better maintenance and performance, all the database queries have been stored in stored procedures, which separate the database and the middle-tier access layer.

3.4 User Interface

User Interface Design is the process that defines how the system that is being developed would interact with the external entities like users (service providers, service consumers, admins, etc). A user interface has three main parts[7]: Navigation mechanism, input mechanism and output mechanism.

- ***Navigation mechanism*** is the way in which the user gives instructions to the system regarding what to do. The types of navigation control are languages, menus (drop-down, pop-up, tab, toolbar, image map, menu bar), messages, direct manipulation.
- ***Input mechanism*** is the way in which the system captures user information and preferences through various means. The types of inputs are text, numbers, selection boxes (check boxes, radio-buttons, on-screen list boxes, drop-down list boxes, combo boxes, and sliders. Input mechanism also needs validation checks like completeness check, format check, range check, check digit check, consistency check and database check in order to prevent invalid information from entering the system.
- ***Output mechanism*** is the way by which the system displays information that the users of the system are looking for (e.g.: reports, web pages).

Though these three mechanisms are conceptually different, they generally cannot be completely separated in a user interface display. Also, they have the following common design principles[7]:

- ***Layout*** refers to the areas on the screen that are repeated on every page or are used consistently for varied purposes.

- ***Content Awareness*** refers to the ease with which users can identify with their position with respect to the system and the corresponding information.
- ***Aesthetics*** refers to the pleasing appearance of the interface with respect to white spaces, fonts, colors, images, etc.
- ***User Experience*** refers to the usability or ease of use and learning curve for the user to learn the system. There is always a tradeoff between the ease of use and learning curve depending on the level of expertise of the user.
- ***Consistency*** refers to the predictability that the users can have about the system before performing a particular action on the system.
- ***Minimal User Effort*** refers to the ease with which the user can access the system. (e.g.: no of mouse clicks).

The figure below displays the initial user interface for Web Services @ MIT. The detailed user interfaces can be found in some of the actual screenshots of the application included in ***Appendix A***, included after chapter 6. These consist of screenshots for an admin as well as non-admin user and also the validation checks for input mechanism.

The Initial User Interface Diagram below displays how the navigation between the different pages of the site would take place. Three basic tabs – home, About the Portal and Documents would be visible to all users in order to encourage the visitors of the site to register and login to the site if they find the information on the publicly available tabs interesting. A new user would click on the register link button and would be directed to the register.aspx page after which he would be redirected to the home page. An existing user would enter his user name and password to login to the system and access the home page (webservices.aspx) with the web services listings tab. An admin user would also be able to see an admin tab in addition to the web services listings tab. For a non-admin user, the web services listings page would display the web services, owner, description, maturity, average rating and average usage of the approved web service and would also be able to select a rating and usage for any web service. In addition, he would be able to edit or delete his own web service. An admin user would be able to see all the registered web services and would be able to change the status of a newly registered web service to approved, if appropriate. In addition, he would be able to edit or delete all the web services registered on the site. Also, both admin and non-admin users can search for a web service by either of name, owner, maturity or average rating. The document tab would display the title, owner and description of the document. Admin user can also see link buttons for add new documents and edit documents. The admin user would be able to control the layout of the site, modules, tabs through the admin page/tab. In addition, he would be able to add/edit/delete roles and users as well as assign/change user roles through the admin tab.

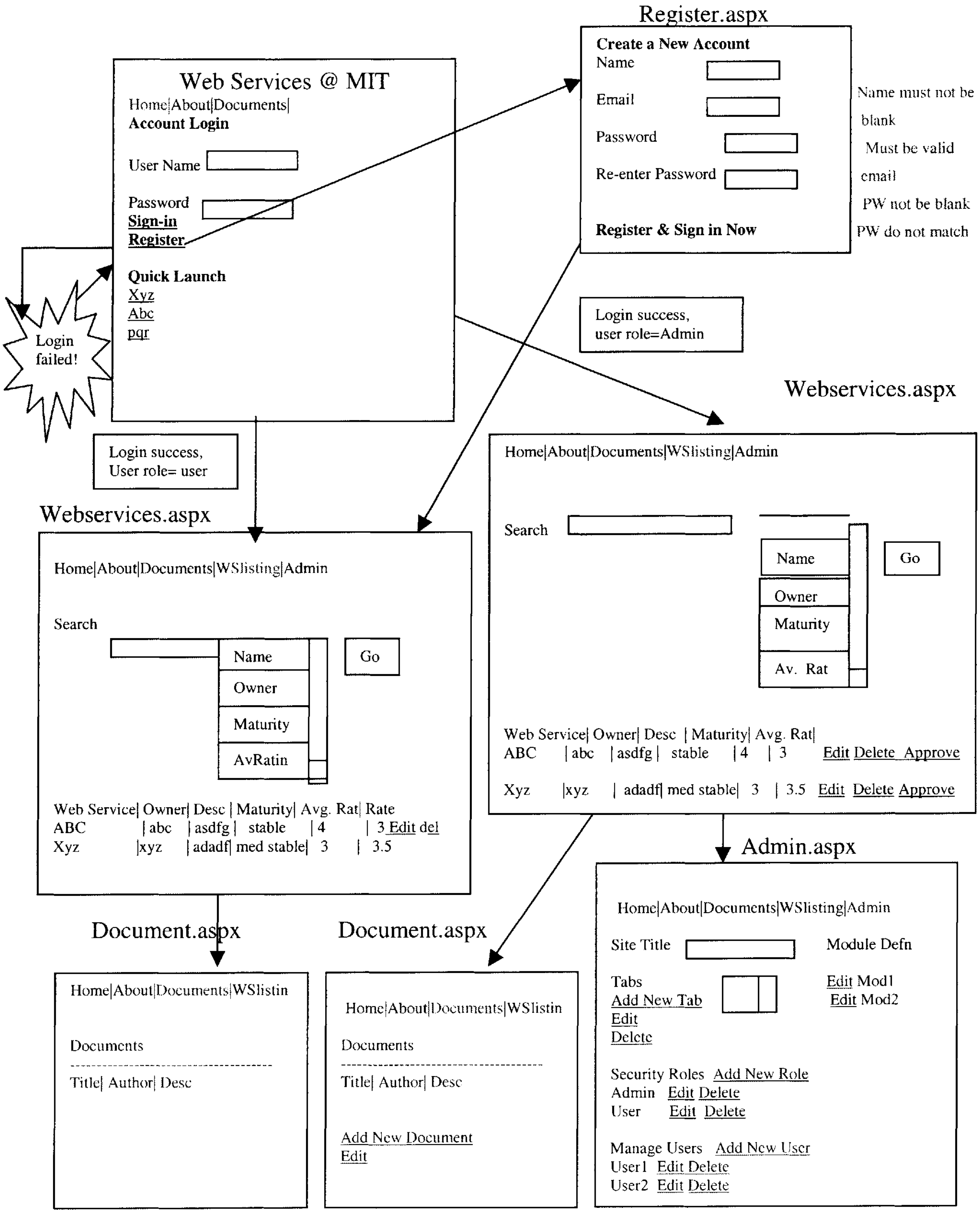


Figure 3.11: Initial User Interface Diagram

CHAPTER 4: PORTAL ARCHITECTURE AND FRAMEWORK

4.1 Web Services @ MIT Portal Architecture[8]

Most IBuySpy applications have a physical 2-tier architecture that combines the presentation tier and the middle tier on one server and the database on a second in order to get optimal performance.

The Web Services @ MIT Portal also uses two-tier application architecture: the data layer and the presentation layer. For simplicity, the business logic layer and the database interface components of the data access layer, that are separate in .NET architecture are combined into a single layer, the principal data source for the application being SQL Server database along with the stored procedures. The data access is provided through a Microsoft .NET assembly that talks to SQL Server via the stored procedures. Also, the portal framework is built through the use of a number of assemblies that handle the portal security and configuration. The presentation layer comprises of web forms and user controls that handle the display and management of the portal data for the users. Also, the logic and UI are separated from one another through a concept called code-behind, which separates the code by placing it in a different file. This makes things simpler for the developers by preventing spaghetti code and hence reducing the probability of bugs in the application.

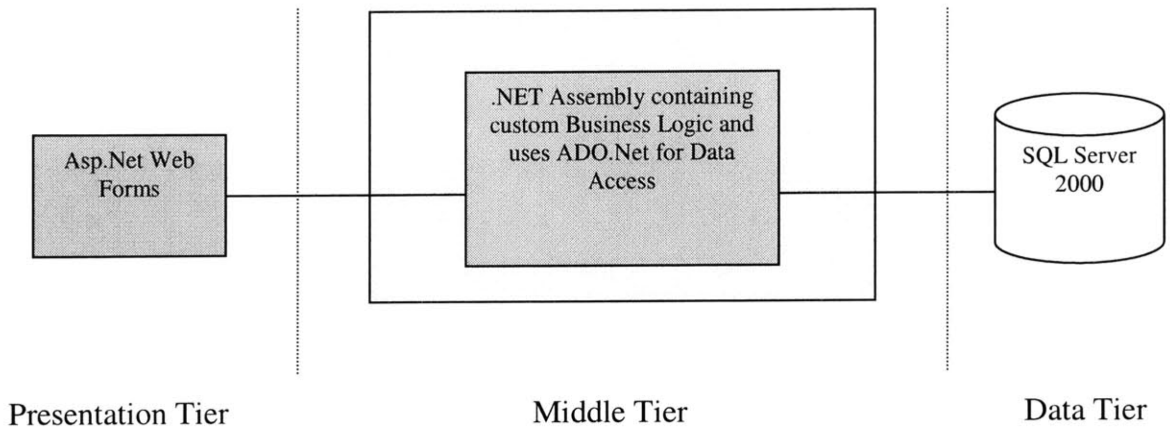

Asp.Net Web Forms
.NET Assembly containing custom Business Logic and uses ADO.Net for Data Access
SQL Server 2000
Presentation Tier
Middle Tier
Data Tier

Figure 4.1: Web Services @ MIT Application Architecture

4.2 Web Services @ MIT Portal Framework[8,9]

4.2.1 Framework Components

Based on IBuySpy framework, the Web Services @ MIT portal framework consists of the following for displaying and managing the data in the presentation layer. These include:

- Portal Settings
- Portal Tabs
- Portal modules
- Portal Security

Portal Settings include portal ID, portal name, the desktop tabs and module tabs collection, portal custom settings, active tab and its custom settings, access to portal and site properties, active theme and layout. PortalSettings class that is defined in the Configuration namespace component represents portal settings and with each web request of portal application, it is updated and placed into the 'Context' object. After being placed in the 'Context' object, portal settings can be accessed from any page of the application, component or any control by accessing the Context item with the name "PortalSettings". Current portal settings can be accessed by using the PortalSettings property defined on WebServices@MIT.UI.Page and WebServices@MIT.UI.PortalModuleControl

Portal Tabs is the part of the portal that comprises of the top-level navigation. In other words, portal tabs represent the pages of the portal. These are stored in the public field called DesktopTabs of the PortalSettings object. DesktopTabs are of the type ArrayList and can be accessed through the PortalSetting Context item. The top navigation of the tabs is handled in the DesktopPortalBanner.ascx user control. The "Tab" User Interface is dynamically constructed with the "Tab strip" at the top of page. Each tab can have up to 3 columns of "modules".

Portal Modules are user controls that provide the actual content of the Web Services @ MIT portal. These user controls inherit the PortalModuleControl base class that enables the modules to talk to the underlying portal framework. Web Services @ MIT uses some of the modules of IBuySpy framework by customizing them and also has some additional modules developed by extending the IBuySpy framework. These modules are described in the next section.

Portal Security is described separately with user Roles in section **4.3.**

Also the portal directory structure is as follows:

- /Web.Config
- /Global.asax
- *(3 .ASPX Pages)*

- *(2 .ASCX User Controls)*
- /Components/
- /DesktopModules/
- /Admin/
- /Images/
- /Bin/

4.2.2 Customizing the Portal

A few files are generally easy to configure to begin customization of the portal. These include the portal.css file that describes the color scheme of the portal. This file is a cascading style sheet used to define color, shading, orientation, etc of site but not the actual data. Thus it helps change the presentation of the html. Changing values here affects the overall style of the page but cannot change structure. Another such file is one is DesktopPortalBanner.ascx. This file is an ASP web control that is used to render the top part of the page like the title and is common across the entire site. It includes the hashed lines and the asp.net logo. The third file that can be the starting point for customization is DesktopDefault.ascx, which has the main content for desktop users.

4.3 Web Services @ MIT Portal Modules[8,9]

The Portal modules are implemented as ASP.NET User Controls. They are dynamically loaded and populated into portal based on portal configuration. ASP .NET user controls are .ascx files that can be thought of as "mini pages" . They can have nested server controls within them. They have fully support post back, validation, etc. These user controls can be used within a page either statically or dynamically with <WebServices@MIT:UserRatings runat=server /> and with

Page.LoadControl(String srcFile) method. The dynamic layout of the modules is managed by *DesktopDefault.aspx,* which is the .aspx page that dynamically loads and populates portal modules and supports browsers like IE, Netscape.

Portal Modules provide the users with functionality or the content on the portal tabs. These modules could be text, graphics, news, links or provide user with discussion list, contacts or similar data. Each portal tab can have several modules and these modules can be repeated on different tabs. The Web Services @ MIT portal uses the following modules of IBuySpy framework:

- ***Sign-in module:*** This module allows the user to log in to the portal. If the user is not already logged it, the portal framework displays this module on the portal homepage. If the user has already signed it, his authentication information (email) is displayed at the top right corner of the portal.
- ***Links module:*** This module renders a list of hyperlinks and includes an edit page, which allows authorized users to edit the Links data stored in the SQL Server database.
- ***QuickLinks module:*** This module compactly renders a list of useful links that can globally appear on several tabs on the portal. QuickLinks also has an edit page that is same as in Links module and allows authorized users to edit the data stored in the SQL Server database.
- ***Html/Text module:*** This module displays the information about the portal (introduction of the Web Services @ MIT portal on the home page) in an Html format and also allows the user to enter this information in an Html format.

- ***Events module:*** This modules contains a list of upcoming events related to topics of interest to the portal users. This module includes time, location and a brief description of the event. It also includes an edit page associated with every event, which allows authorized users to edit the Events data stored in the SQL Server database.
- ***Contacts module:*** This module displays contact information for a group of people. This has been used to display the contact of admin of the web services @ MIT Portal. This module also includes an edit page that allows authorized users (admin) to edit the Contacts data stored in the SQL Server database.
- ***Discussion module:*** This module includes a group of message threads on a specific topic. Discussion includes a Read/Reply Message page, which allows authorized users to reply to existing messages or add a new message thread. All users of the Web Services @ MIT Portal can communicate using this discussion forum and the data is stored in the SQL Server database.
- ***Documents module:*** This module renders a list of documents, including links to download the document. These documents can be white papers or documents related to web services and users' experience of building web services or applications that consume web services. Documents module also includes an edit page, which allows authorized users to edit the information regarding the document about the Documents (for example, a friendly title) stored in the SQL database. The actual document is stored in the file system and the URL is stored in the SQL database.
- ***Image module:*** This module render image using an HTML IMG tag and is used on the home page as well as several other tabs of the Web Services @ MIT portal. This module also displays the height and width attributes of images in order to customize

the modules. Image module also has an edit page which allows authorized users to edit the image source, height and width.

4.3.1 Custom Modules[9]

IBuySpy framework is specifically built to be extensible in that the entire architecture is built in moving the modules around within the framework of the site itself. Hence, one can easily implement own modules. The main extensibility point is in creating custom modules. These custom modules are then registered with the portal. The portal framework will add the newly created module to the list of available modules and it can then be added to the site. This process is described below by adding the web services module to the Web Services @ MIT Portal.

Extending the Data Layer

For building a WebServices Module, the first step is to create a table called Web Services in the portal database and create a many to one relationship of this table with the module table. The next step is to create stored procedures associated with the WebServices table. The following stored procedures are needed:

- AddWebServices
- DeleteWebServices
- GetWebServices
- GetSingleWebServices
- UpdateWebServices

After the stored procedures are created, a data access layer (DAL) component needs to be created to provide access to the WebServices procedures. A few of the methods that need to be defined for this are shown below:

- WebServicesPortal.WebServicesDB.GetWebService()
- WebServicesPortal.WebServicesDB.GetSingleWebService()
- WebServicesPortal.WebServicesDB.DeleteWebService()
- WebServicesPortal.WebServicesDB.AddWebService()
- WebServicesPortal.WebServicesDB.UpdateWebService()

Creating the User Control

The next step is to create user control for handling the WebService user interface. This user control would contain a DataGrid that will define four columns: Title, Description, Owner, and Status. A new user control called "WebServices.ascx" then needs to be added to DesktopModules and also a DataGrid control called "myDataGrid" needs to be added on the WebServices Control page.

The UI part of the control is given in the "WebServices.ascx" file below:

```
-------------------------------------------------------------------------------------------------------
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="WebServices.ascx.cs"
Inherits="WebServices.DesktopModules.WebServices"%>
<%@ Register TagPrefix="tra" Namespace="WebServices.UI.WebControls.Globalized"
Assembly="WebServices" %>
<asp:DataGrid id="myDataGrid" HeaderStyle-CssClass="Normal" HeaderStyle-Font-Bold="true"
ItemStyle-CssClass="Normal" AutoGenerateColumns="false" CellPadding="5" BorderWidth="0"
Width="100%" EnableViewState="false" runat="server">
	<Columns>
		<asp:TemplateColumn>
			<ItemTemplate>
				<tra:HyperLink id="editLink" TextKey="EDIT" Text="Edit"
ImageUrl="~/images/edit.gif" NavigateUrl='<%#
```

```
WebServices.HttpUrlBuilder.BuildUrl("~/DesktopModules/WebServicesEdit.aspx", "ItemID=" +
DataBinder.Eval(Container.DataItem,"ItemID") + "&Mid=" + ModuleId) %>' Visible="<%# IsEditable
%>" runat="server" />
                        </ItemTemplate>
                </asp:TemplateColumn>
                <tra:BoundColumn DataField="Title" TextKey="WEBSERVICE_TITLE"
HeaderText="Title" runat="server" />
                <tra:BoundColumn DataField="EstCompleteDate" TextKey="WEBSERVICE_DESC"
HeaderText="Description" runat="server" DataFormatString="{0:d}" />
                <tra:BoundColumn DataField="Status" TextKey="WEBSERVICE_STATUS"
HeaderText="Status" runat="server" />
<tra:BoundColumn DataField="Owner" TextKey="WEBSERVICE_OWNER" HeaderText="Owner"
runat="server" />
        </Columns>
</asp:DataGrid>
```

The code-behind part of the control is given below in the “WebServices.ascx.cs” file:

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using System.Data.SqlClient;
using WebServices.Admin;

namespace WebServices.DesktopModules
{
        public class WebServicesEdit : WebServices.UI.EditItemPage
        {
                protected WebServices.UI.WebControls.Globalized.Literal Literal1;
                protected WebServices.UI.WebControls.Globalized.Literal Literal2;
                protected System.Web.UI.WebControls.TextBox TitleField;
                protected WebServices.UI.WebControls.Globalized.RequiredFieldValidator Req1;
                protected WebServices.UI.WebControls.Globalized.Literal Literal3;
                protected System.Web.UI.WebControls.TextBox Description;
                protected System.Web.UI.WebControls.RequiredFieldValidator Req2;
                protected WebServices.UI.WebControls.Globalized.Literal Literal4;
                protected System.Web.UI.WebControls.TextBox StatusBox;
                protected WebServices.UI.WebControls.Globalized.RequiredFieldValidator Req3;
                protected System.Web.UI.WebControls.TextBox OwnerField;
                protected WebServices.UI.WebControls.Globalized.RequiredFieldValidator Req4;
                protected System.Web.UI.WebControls.Label CreatedDate;

                private void Page_Load(object sender, System.EventArgs e)
        {

            if (Page.IsPostBack == false)
            {
                //Item id is defined in base class
                if (ItemId > 0)
                {
                    //Obtain a single row of WebService information.
                    WebServicesDB WebServicesDB = new WebServicesDB();
                    SqlDataReader dr =
WebServicesDB.GetSingleWebServices(ItemId);

                    //Load the first row into the DataReader
                    dr.Read();
                    TitleField.Text = (String) dr["Title"];
                    EstCompleteDate.Text = ((DateTime)
dr["Description"]).ToShortDateString();
                    StatusBox.Text = (String) dr["Status"];
                    CreatedBy.Text = (String) dr["Owner"];
                    CreatedDate.Text = ((DateTime)
dr["CreatedDate"]).ToShortDateString();
                    dr.Close();
                }
                else
                {
                    //Provide defaults
                    CreateDate.Text =
DateTime.Now.AddDays(60).ToShortDateString();
                }
            }
        }


        protected override void OnUpdate(EventArgs e)
        {

            base.OnUpdate(e);
            if (Page.IsValid == true)
            {
                WebServicesDB WebServicesDB = new WebServicesDB();
                if (ItemId <= 0)
                    WebServicesDB.AddWebServices(ItemId, ModuleId,
Context.User.Identity.Name, DateTime.Now, TitleField.Text, Description.Text, OwnerField.Text,
StatusBox.Text);
                else
                    WebServicesDB.UpdateWebServices(ItemId, ModuleId,
Context.User.Identity.Name, DateTime.Now, TitleField.Text, Description.Text, OwnerField.Text,
StatusBox.Text);
                this.RedirectBackToReferringPage();
            }
        }
        /// called on UpdateDescription.Text, OwnerField.Text,
        override protected void OnDelete(EventArgs e)
        {
            // Calling base we check if the user has rights on deleting
            base.OnUpdate(e);

            if (ItemId > 0)
            {
                WebServicesDB WebServicesDB = new WebServicesDB();
                WebServicesDB.DeleteWebServices(ItemId);
            }

            // This method is provided by the base class
            this.RedirectBackToReferringPage();
        }

        #region Web Form Designer generated code

        /// Raises OnInitEvent

        protected override void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }


        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);

        }
        #endregion

    }
}
--------------------------------------------------------------------------------------------------------------------------
```

Inherit the WebServices Module Control Base Class

The user control then needs to inherit the WebServices Module Control Base Class that provides all the hooks that are required for the WebServices module to interact with the framework.

Add the Module Title

One of the user controls that is available to portal modules is the Title user control. This control will generate the appropriate HTML for the Module's title. The title is added in the OnInit procedure.

Add Support for an Edit Page

In order to allow users to edit and add additional WebServices, support for an edit page must be added. This support is added by defining EditUrl to the Module Title Control on a page that edits the content of your module directly, or defining AddUrl, on a page for adding items to the control (like in the case of WebServices Module). The naming convention starts with ModuleName and then by adding the function. e.g.WebServicesEdit

Create the Edit Page

Once the support for an edit page has been added using the Title User Control, the actual edit page needs to be created. In addition to the HTML, the following four methods need to be defined: Page_Load, OnUpdate, OnDelete, OnCancel

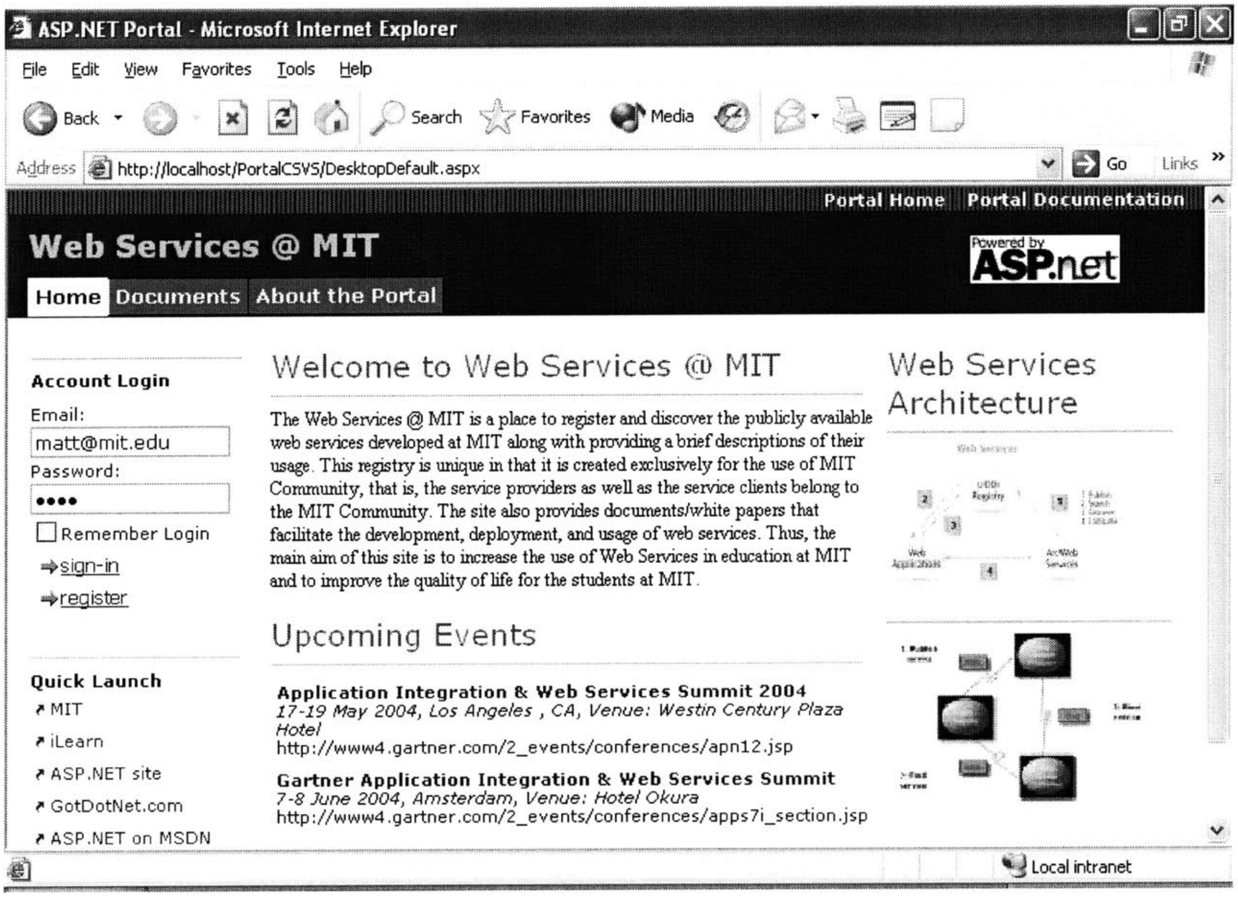
Adding the New WebServices Module to the Framework

The final step is to add the completed WebServices Portal Module to the portal framework by using the Module Definitions section on the Admin Page. This needs rebuilding the project before adding the module to the portal. This can be done by clicking on the Add New Module Type, which would bring up the page for adding new module type page and the module can then be added by using "Tab Name and Layout" section.

4.4 User Roles and Security[5,8,9]

4.4.1 User Roles

The Web Services @ MIT portal supports two types of Roles: Users and Admins. Any visitor of the site (all users) can see the Home, Documents and About the Portal Tabs without having to log in to the portal.



http://localhost/PortalCSVS/DesktopDefault.aspx

Web Services @ MIT

Account Login

Email: matt@mit.edu

Password: ••••

☐ Remember Login

⇒sign-in

⇒register

Quick Launch

- MIT
- iLearn
- ASP.NET site
- GotDotNet.com
- ASP.NET on MSDN

Welcome to Web Services @ MIT

The Web Services @ MIT is a place to register and discover the publicly available web services developed at MIT along with providing a brief descriptions of their usage. This registry is unique in that it is created exclusively for the use of MIT Community, that is, the service providers as well as the service clients belong to the MIT Community. The site also provides documents/white papers that facilitate the development, deployment, and usage of web services. Thus, the main aim of this site is to increase the use of Web Services in education at MIT and to improve the quality of life for the students at MIT.

Upcoming Events

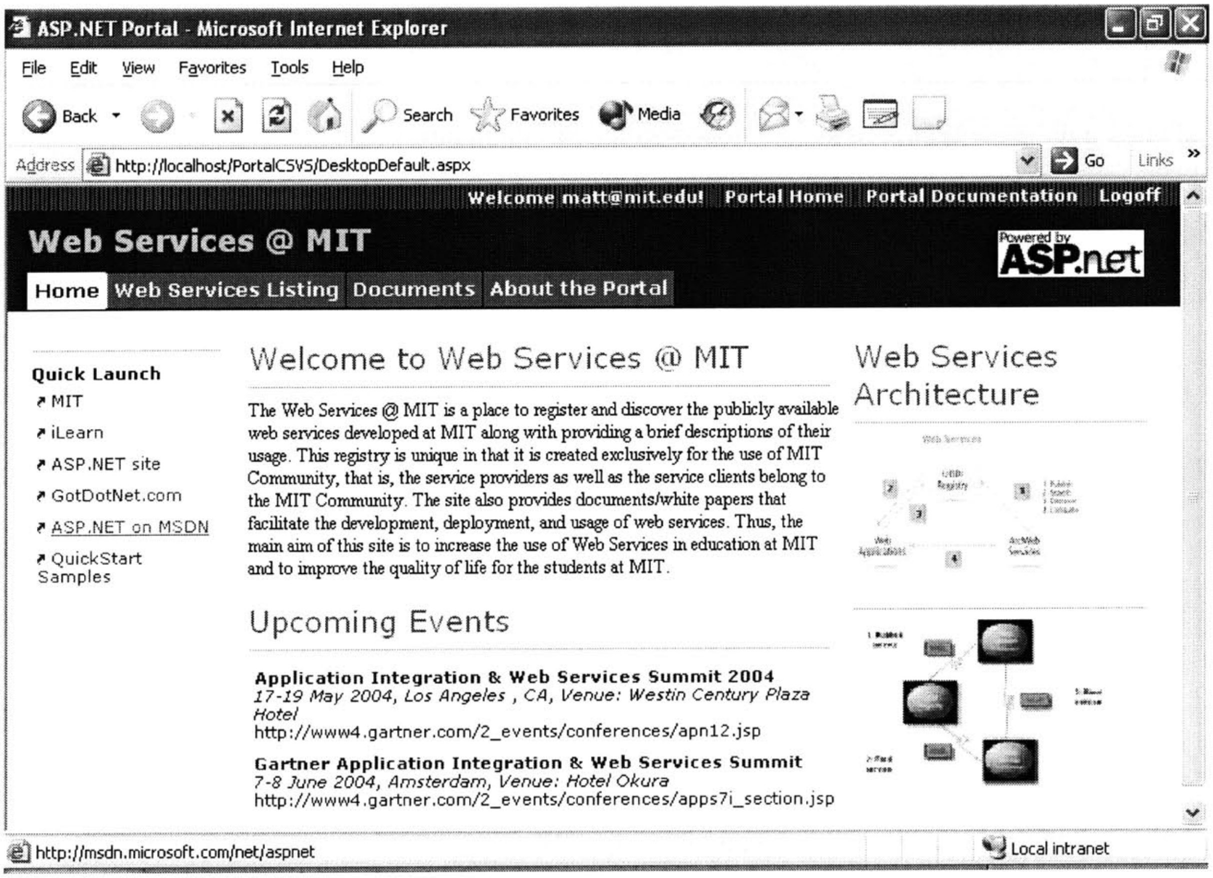
Application Integration & Web Services Summit 2004
17-19 May 2004, Los Angeles , CA, Venue: Westin Century Plaza Hotel
http://www4.gartner.com/2_events/conferences/apn12.jsp

Gartner Application Integration & Web Services Summit
7-8 June 2004, Amsterdam, Venue: Hotel Okura
http://www4.gartner.com/2_events/conferences/apps7i_section.jsp

Figure 4.2: Home Page of Web Services @ MIT Portal

User Role

Once a user logs in to the portal, he would be able to see the Web Services Listing Page in addition to Home, Documents and About the Portal pages as shown in the Figure 4.3 below. The users would not be able to see the admin tab but would be able to add, edit or delete only their own web services.


ASP.NET Portal - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/PortalCSVS/DesktopDefault.aspx
Go Links
Welcome matt@mit.edu! Portal Home Portal Documentation Logoff
Web Services @ MIT
Powered by ASP.net
Home Web Services Listing Documents About the Portal
Quick Launch
MIT
iLearn
ASP.NET site
GotDotNet.com
ASP.NET on MSDN
QuickStart Samples
Web Services Architecture
http://msdn.microsoft.com/net/aspnet
Local intranet

Welcome to Web Services @ MIT

The Web Services @ MIT is a place to register and discover the publicly available web services developed at MIT along with providing a brief descriptions of their usage. This registry is unique in that it is created exclusively for the use of MIT Community, that is, the service providers as well as the service clients belong to the MIT Community. The site also provides documents/white papers that facilitate the development, deployment, and usage of web services. Thus, the main aim of this site is to increase the use of Web Services in education at MIT and to improve the quality of life for the students at MIT.

Upcoming Events

Application Integration & Web Services Summit 2004
17-19 May 2004, Los Angeles , CA, Venue: Westin Century Plaza Hotel
http://www4.gartner.com/2_events/conferences/apn12.jsp

Gartner Application Integration & Web Services Summit
7-8 June 2004, Amsterdam, Venue: Hotel Okura
http://www4.gartner.com/2_events/conferences/apps7i_section.jsp

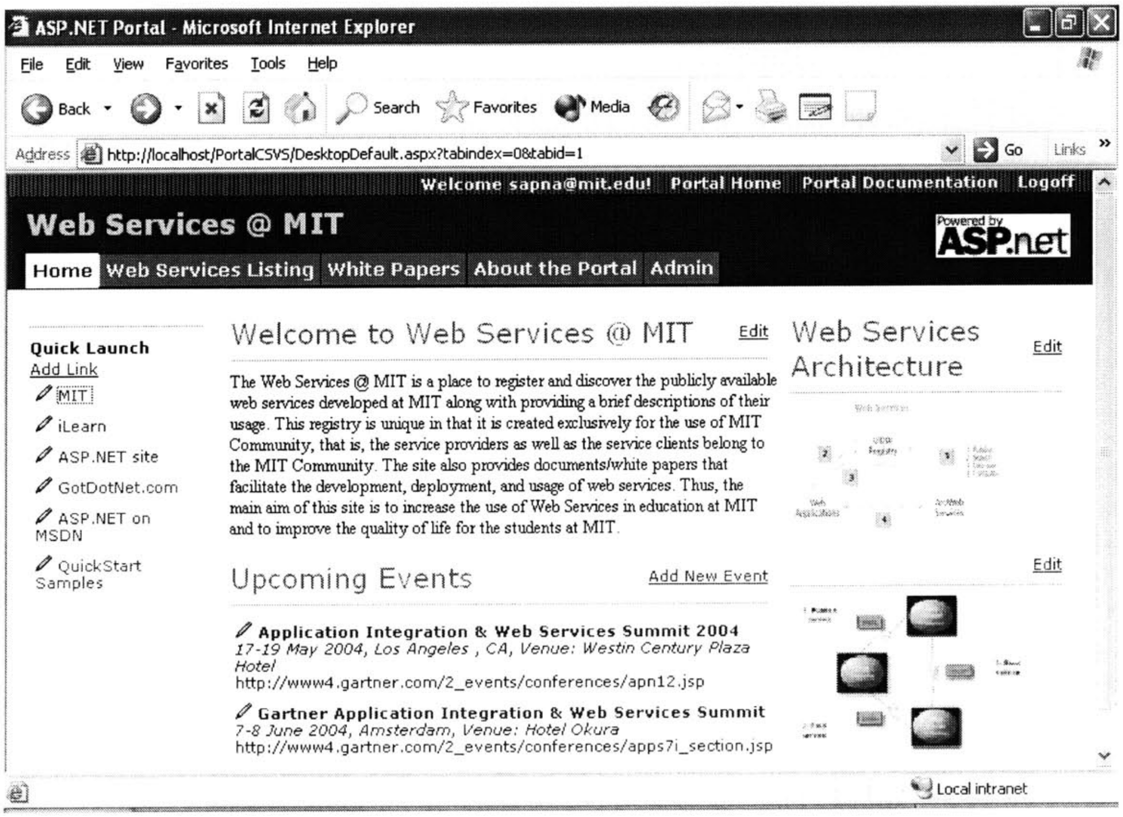
Figure 4.3: Homepage of Web Services @ MIT Portal after User Login

Admin Role

As shown in the Figure 4.3 below, a user who has admin privileges can also see the admin tab upon login, in addition to the Web Services Listing, Home, About the Portal and Documents pages. As shown in figure below, an admin user would be able to see the edit link button or edit icon (Pencil icon) next to each item or module on every page. The admin user can click on these and add, update and/or delete these items. Or example, he would be able to approve, update and delete any web service and would be able to add, delete or update any document (white paper).

Also, as shown in Figures 4.3 and 4.4 below, through the admin page, a user with admin privileges would also be able to see the admin tab through which he would be able to change the title of the site, add, update and delete modules, add new tabs or update/delete existing tabs, add new role or update/delete an existing role, change the role of a user, add new user or delete and existing user.

The admins would be able to see the admin tab and would be able to approve, delete any web service and delete or change the role of any user. Based on the IBuySpy framework, the admin can also add additional roles if required. The users would not be able to see the admin tab but would be able to add, edit or delete only their own web services. The portal uses the authorization and authentication security features of the IBuySpy framework.

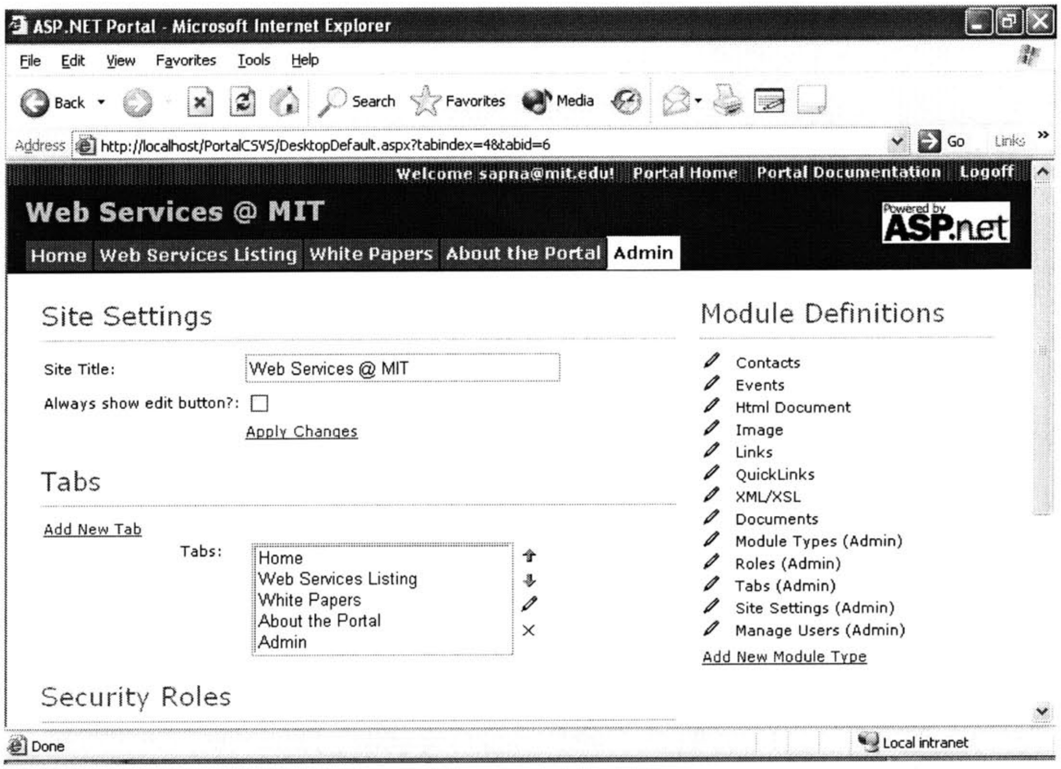

ASP.NET Portal - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/PortalCSVS/DesktopDefault.aspx?tabindex=0&tabid=1
Go Links
Welcome sapna@mit.edu! Portal Home Portal Documentation Logoff
Web Services @ MIT
Powered by ASP.net
Home Web Services Listing White Papers About the Portal Admin
Quick Launch
Add Link
MIT
iLearn
ASP.NET site
GotDotNet.com
ASP.NET on MSDN
QuickStart Samples
Welcome to Web Services @ MIT
Edit
Web Services Architecture
Edit
Upcoming Events
Add New Event
Edit
Local intranet

The Web Services @ MIT is a place to register and discover the publicly available web services developed at MIT along with providing a brief descriptions of their usage. This registry is unique in that it is created exclusively for the use of MIT Community, that is, the service providers as well as the service clients belong to the MIT Community. The site also provides documents/white papers that facilitate the development, deployment, and usage of web services. Thus, the main aim of this site is to increase the use of Web Services in education at MIT and to improve the quality of life for the students at MIT.

Application Integration & Web Services Summit 2004
17-19 May 2004, Los Angeles , CA, Venue: Westin Century Plaza Hotel
http://www4.gartner.com/2_events/conferences/apn12.jsp

Gartner Application Integration & Web Services Summit
7-8 June 2004, Amsterdam, Venue: Hotel Okura
http://www4.gartner.com/2_events/conferences/apps7i_section.jsp

Figure 4.4: Homepage of Web Services @ MIT Portal for Admin User


ASP.NET Portal - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/PortalCSVS/DesktopDefault.aspx?tabindex=4&tabid=6
Go Links
Welcome sapna@mit.edu! Portal Home Portal Documentation Logoff
Web Services @ MIT
Powered by ASP.net
Home Web Services Listing White Papers About the Portal Admin
Done
Local intranet

Site Settings

Site Title: Web Services @ MIT

Always show edit button?: ☐

Apply Changes

Tabs

Add New Tab

Tabs:
- Home
- Web Services Listing
- White Papers
- About the Portal
- Admin

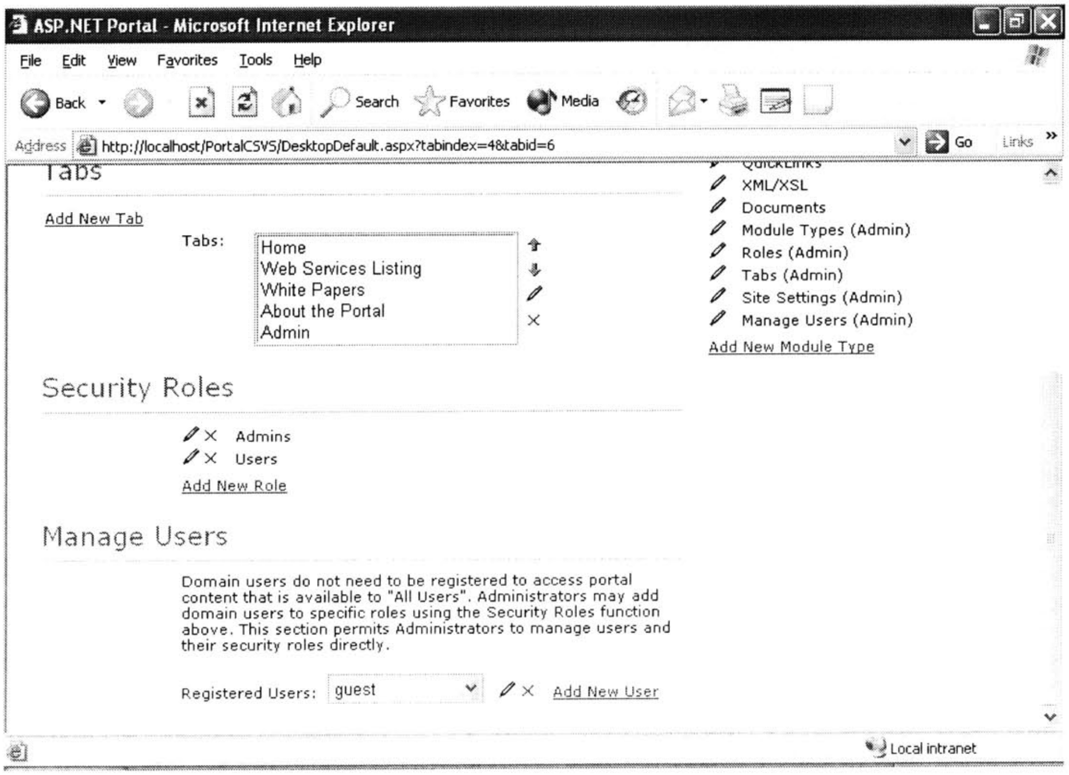
Security Roles

Module Definitions

- Contacts
- Events
- Html Document
- Image
- Links
- QuickLinks
- XML/XSL
- Documents
- Module Types (Admin)
- Roles (Admin)
- Tabs (Admin)
- Site Settings (Admin)
- Manage Users (Admin)

Add New Module Type

Figure 4.5: Admin Page for Web Services @ MIT Portal -a


ASP.NET Portal - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/PortalCSVS/DesktopDefault.aspx?tabindex=4&tabid=6
Go Links
Local intranet

Tabs

Add New Tab

Tabs:
- Home
- Web Services Listing
- White Papers
- About the Portal
- Admin

- XML/XSL
- Documents
- Module Types (Admin)
- Roles (Admin)
- Tabs (Admin)
- Site Settings (Admin)
- Manage Users (Admin)

Add New Module Type

Security Roles

- Admins
- Users

Add New Role

Manage Users

Domain users do not need to be registered to access portal content that is available to "All Users". Administrators may add domain users to specific roles using the Security Roles function above. This section permits Administrators to manage users and their security roles directly.

Registered Users: guest Add New User

Figure 4.6: Admin Page for Web Services @ MIT Portal -b

In addition to the content and layout of the portal, the users with Admin privileges are allowed to manage the security of the portal.

4.4.2 Security

The Web Services @ on MIT portal inherits the security features of the IBuySpy Framework. These include authentication and authorization.

Portal Authentication

Web Services @ MIT supports two modes of application authentication:

- Forms based authentication, in which usernames/passwords are stored in database
- Windows authentication that uses a domain/active directory with the NTLM challenge/response protocol.

The authentication mode is controlled by web.config configuration file. The User.Identity.Name property encapsulates user name access for users.

Portal Authorization

Authorization is role based security used to control whether or not user has access rights. All users are divided into 2 roles: Users and Admins. Role mappings are stored within database and can be accessed via UserDB.GetRoles(User as String). Current user's role mappings are set for request within Global.asax. Application_AuthenticateRequest event is used to apply role mappings for current user. Then the Context.User is set using the GenericPrincipal method. The User.IsInRole method can be used to verify whether the current user is in a specific role. Also, security.cs encapsulates DB lookup. User.IsInRole can be used to verify whether current user is in role. This works from all pages, controls, classes and can be checked as "If (User.IsInRole("Admin")) then .....".

CHAPTER 5: CONCLUSIONS/FUTURE WORK

5.1 Discussion

Addressing Security Issues

Publicizing a real working web service may raise security concerns. For example, many web services store data. If they are open to anyone and everyone, the hackers and malicious users can overload the system by adding dummy data. This is a critical issue. In order to ensure that the quality of a web service is not compromised and any service provider feels safe exposing his web service on this site, it becomes a necessity to prevent unauthorized users to access these web services.

To address this problem, Web Services @ MIT would allow a user to access web services only after he logs in to the site using his MIT email ID and password. This reduces the risk to a huge extent by preventing any user outside of the MIT Community to access the web services. A more secure possibility of addressing this problem is to authenticate the user using his MIT web certificate. Another option could be a machine where people could put up sample web services that would very closely imitate the real web services but not actually allow the users to store data. Though this option seems very safe for the service providers, it involves a trade-off with the level of satisfaction of the web service users.

Addressing Maintenance Issues

The Web Services @ MIT portal is built on top of IBuySpy Portal, which uses the best practices of ASP.NET and .NET Framework. Thus, any person with a decent knowledge of C#, ASP.NET and .NET Framework would be able to maintain the system. The main role of the administrator/moderator would be to delete the web services registered on the site if they become obsolete and to keep the server up and running. This would not require a lot of administrator's time and effort, especially if the moderator is interested in web services. Also, the portal has an option of assigning more that one administrator. Hence the responsibility of moderating the portal could also be shared by more than one individual. As mentioned earlier, once the portal gains initial recognition, there are possibilities of sustaining it by several groups/individuals: Sloan, I-Campus, any course 1 or course 6 student who is interested in web services or any member of the community who is really interested in the propagation of web services on the MIT campus.

5.2 Future Applications of Web Services @ MIT

MIT is strongly committed to the enhancement of e-learning and is investing huge resources to achieve this goal. Keeping this effort in view, Web Services @ MIT Portal could become a promising way to market online learning tools through the medium of web services and applications that consume web services. Several such efforts have already been initiated at MIT and a large portion of the MIT community is completely unaware of these initiatives. Propagating these initiatives would foster the environment for innovation. Web Services @ MIT is one way to publicize such applications and initiatives, which can encourage the people of MIT community to get more involved in such efforts.

One such successful initiative is ShuttleTrack Alerts, developed by a Course 6, M.Eng student. ShuttleTrack Alerts is an application that consumes the ShuttleTrack Web Service (http://shuttletrack.mit.edu/ShuttleTrackWebServices/ShuttleTrackWebServices.asmx), which provides position data for the safe ride vans to improve the quality of life of MIT's students. This web service was created based on the Shuttle Track application (http://shuttletrack.mit.edu).

ILearn ShuttleTrack Alerts application allows a user to subscribe to AOL instant messenger alerts and activate them for any MIT shuttle route and stop. Once the user chooses the shuttle and the shuttle stop that he wants to track, an AOL instant message will be sent to his AOL screen name N minutes before the shuttle would arrive at the stop. The user can make his preference for the shuttle, the stop, and the number (N) of minutes in the "Manage Alerts Page" on the ShuttleTrack Alerts website. (http://ilearn.mit.edu/shuttletrack). For instance, if the user wants to receive an AOL instant message when the Cambridge East Saferide is 10 minutes from 84 Massachusetts Ave., he just needs to enter his screen name, choose the Cambridge East shuttle, choose the 84 Massachusetts Ave. stop, and choose 10 min. for the notification time.

If this application and the white paper describing the development experience for the application is listed on Web Services @ MIT site, it would not only encourage other students to build innovative applications that would consume the web services listed or already registered on the portal but also build new web services to share with the MIT community.

Another such initiative in progress is the development of DSpace Web Service.

"DSpace[10] is an open source software platform that enables institutions to:

- capture and describe digital works using a submission workflow module

- distribute an institution's digital works over the web through a search and retrieval system
- preserve digital works over the long term"

Providing DSpace as a Web Service would not only help realize the vision of DSpace but also create opportunities where the people from MIT community could build customized desktop and web-based applications to access the library resources faster and more efficiently. Web Services @ MIT could play a role in this by communicating the existence of this Web Service to the MIT community through a medium that would be conducive to such development.

Events Calendar at MIT is another promising application that would be provided as a web service, which could be consumed to develop customized desktop and web applications that would be useful to the students and other members of the MIT Community.

Web Services @ MIT is not only a portal that would register and list the web services being developed by the people at MIT but also has a potential of serving as a strong online community within MIT where people could learn from each other and share their own experiences to enhance the use of technology for education at MIT and increase the quality of student life at MIT.

CHAPTER 6: REFERENCES

---

[1] Michael Cherry, "ASP.NET Improves Development of Web Applications". Dec 17, 2001. <www.directionsonmicrosoft.com>

[2] http://webservices.org/

[3] Web Services: A Practical Introduction. Systinet. 2003.

[4] http://www.microsoft.com/windowsserver2003/techinfo/overview/uddiguide.mspx

[5] Mahmoud, Qusay H.; "Registration and Discovery of Web Services Using JAXR with XML Registries such as UDDI and ebXML". June 2002

[6] www.xmethods.com

[7] http://www.asp.net

[8] http://www.ibuyspyportal.com

[9] http://www.ondotnet.com/pub/a/dotnet/2002/06/10/aspnet.html

[10] Dennis, Alan; Wixom, Barbara Haley; "System Analysis and Design". John Wiley & Sons, Inc. 2000.

[8] "IBuySpy Application Series: Overview"; Vertigo Software, Inc. January 2002.

[9] "IBuySpy Portal: Design and Implementation"; Vertigo Software, Inc. January 2002.

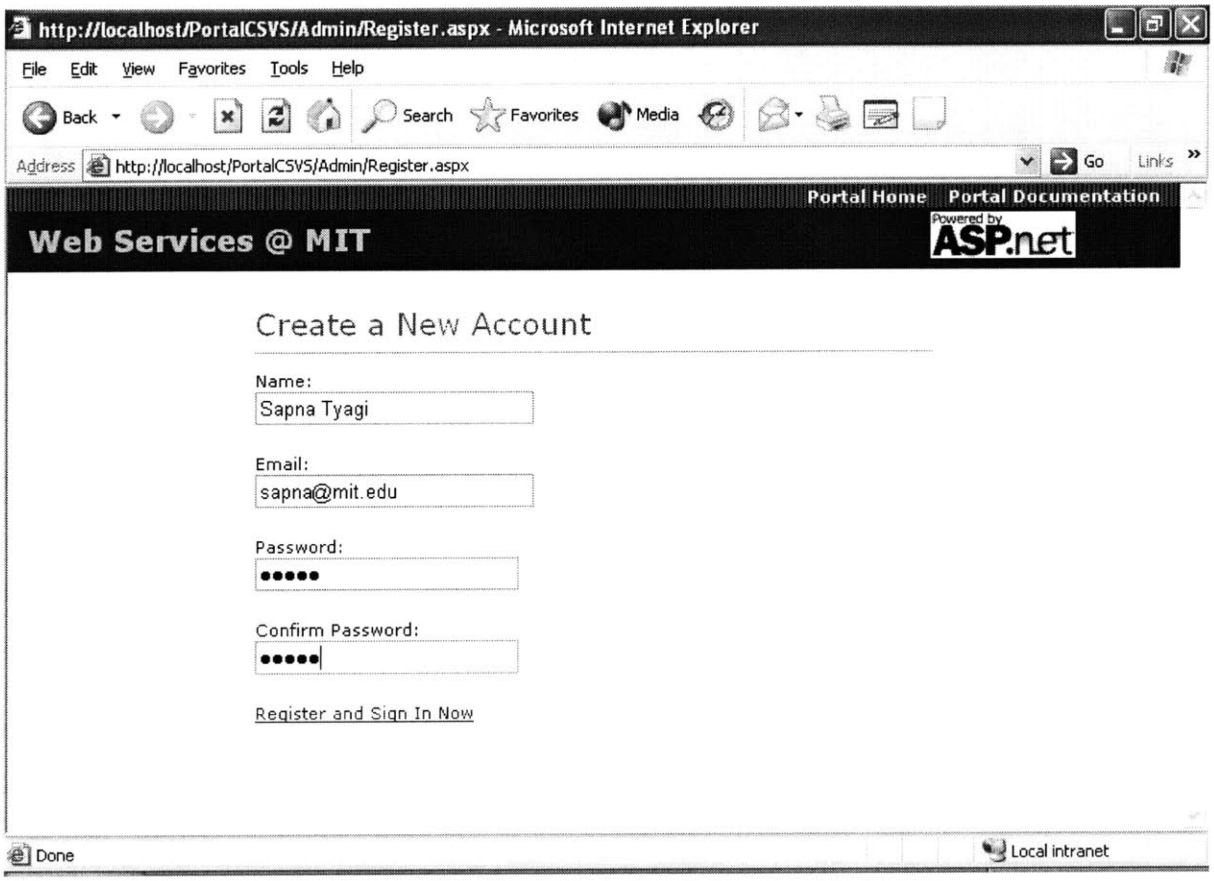
[10] http://libraries.mit.edu/dspace-mit/

Other References

1. http://www.gotdotnet.com
2. http://www.devhood.com
3. http://www.codproject.com
4. http://www.msdn.microsoft.com

APPENDIX A: USER INTERFACE – APPLICATION SCREENSHOTS

A.1 Admin user – sapna@mit.edu


http://localhost/PortalCSVS/Admin/Register.aspx - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/PortalCSVS/Admin/Register.aspx
Go Links
Portal Home Portal Documentation
Web Services @ MIT
Powered by ASP.net
Done
Local intranet

Create a New Account

Name:
Sapna Tyagi

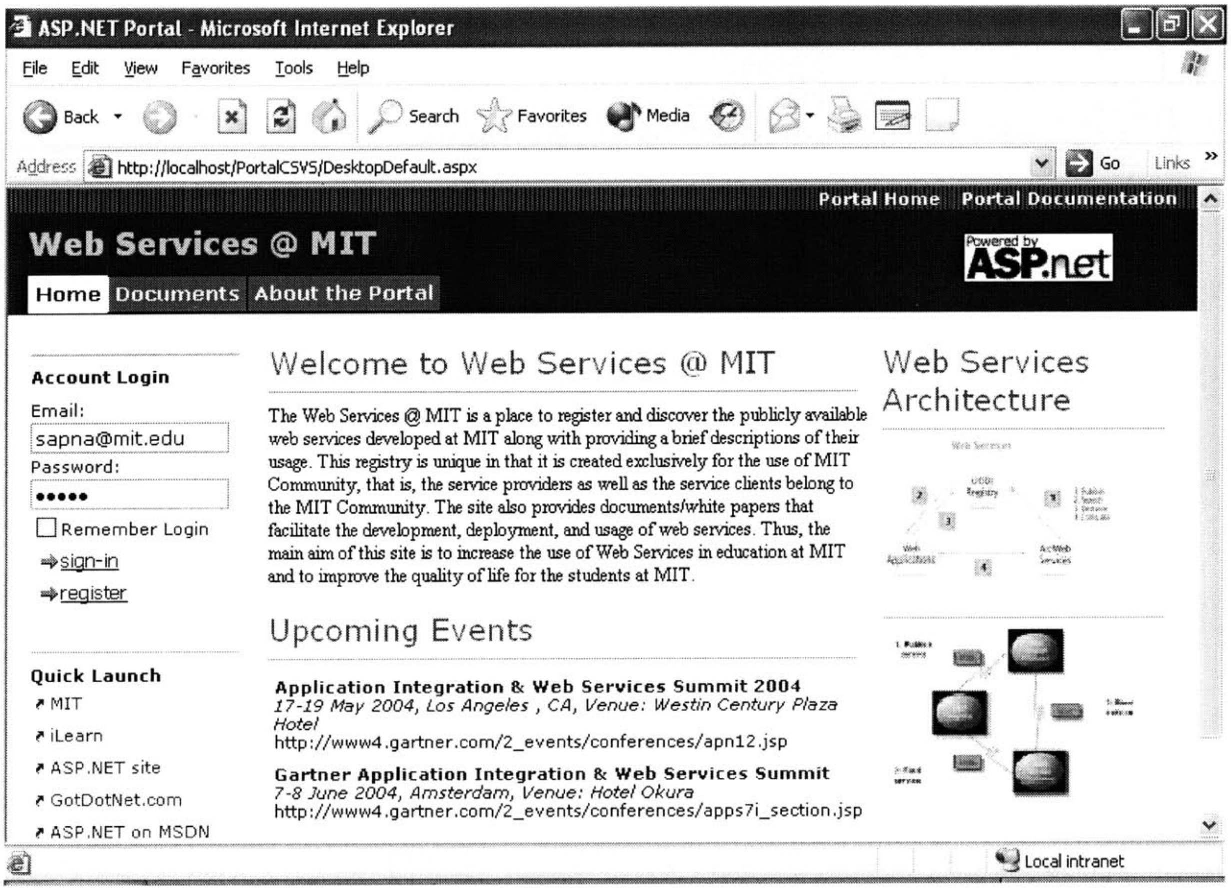
Email:
sapna@mit.edu

Password:
●●●●●

Confirm Password:
●●●●●

Register and Sign In Now

This screen shot describes the user interface for user registration.


ASP.NET Portal - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/PortalCSVS/DesktopDefault.aspx
Go Links
Portal Home Portal Documentation
Powered by ASP.net
Home Documents About the Portal
Local intranet

Web Services @ MIT

Account Login

Email: sapna@mit.edu

Password: •••••

☐ Remember Login

sign-in

register

Quick Launch
- MIT
- iLearn
- ASP.NET site
- GotDotNet.com
- ASP.NET on MSDN

Welcome to Web Services @ MIT

The Web Services @ MIT is a place to register and discover the publicly available web services developed at MIT along with providing a brief descriptions of their usage. This registry is unique in that it is created exclusively for the use of MIT Community, that is, the service providers as well as the service clients belong to the MIT Community. The site also provides documents/white papers that facilitate the development, deployment, and usage of web services. Thus, the main aim of this site is to increase the use of Web Services in education at MIT and to improve the quality of life for the students at MIT.

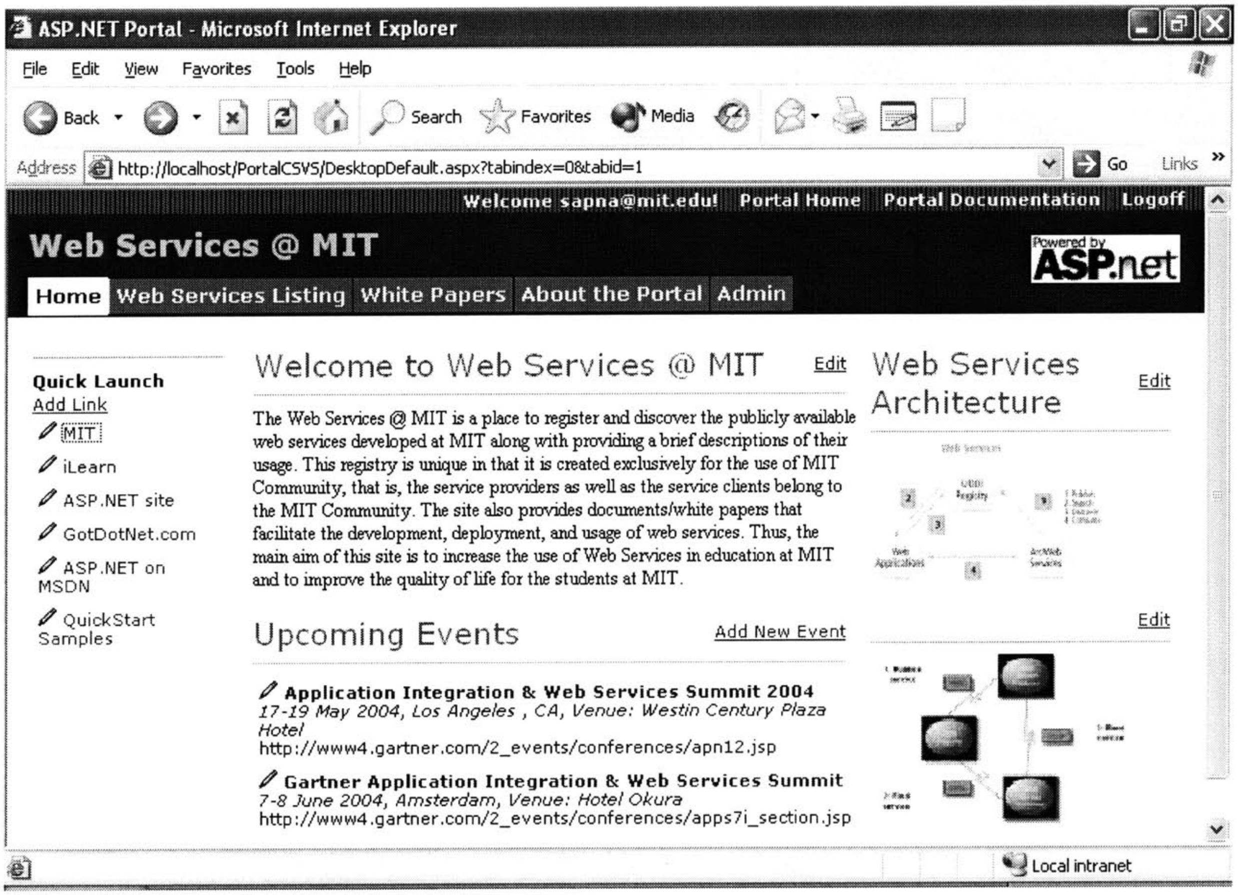
Upcoming Events

Application Integration & Web Services Summit 2004
17-19 May 2004, Los Angeles , CA, Venue: Westin Century Plaza Hotel
http://www4.gartner.com/2_events/conferences/apn12.jsp

Gartner Application Integration & Web Services Summit
7-8 June 2004, Amsterdam, Venue: Hotel Okura
http://www4.gartner.com/2_events/conferences/apps7i_section.jsp

Web Services Architecture

The above screen shot describes the input mechanism for user login and displays the navigation tabs before the admin logs in to the system. The screenshot below describes the same after the admin user logs in. It can be seen that after login, the admin user can see edit links for each module and can also see two additional tabs Web Services Listing and Admin.


ASP.NET Portal - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/PortalCSVS/DesktopDefault.aspx?tabindex=0&tabid=1
Go Links
Welcome sapna@mit.edu! Portal Home Portal Documentation Logoff
Web Services @ MIT
Powered by ASP.net
Home Web Services Listing White Papers About the Portal Admin
Quick Launch
Add Link
MIT
iLearn
ASP.NET site
GotDotNet.com
ASP.NET on MSDN
QuickStart Samples
Web Services Architecture
Edit
Edit
Edit
Add New Event
Local intranet

Welcome to Web Services @ MIT

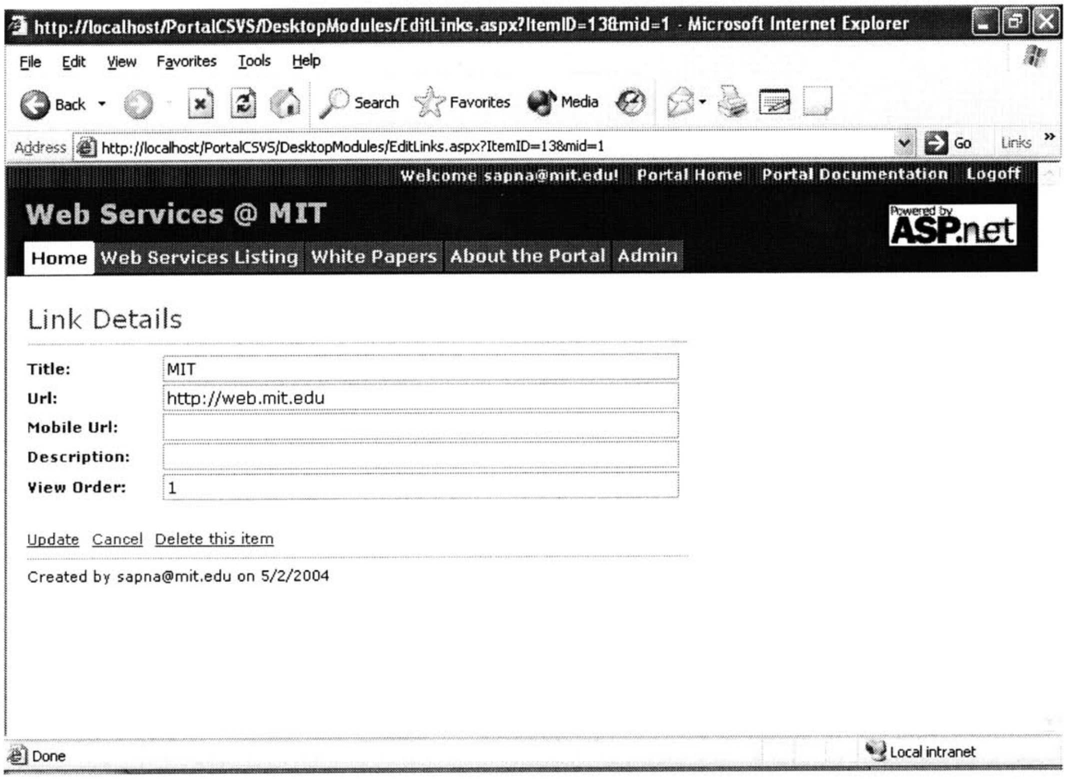
The Web Services @ MIT is a place to register and discover the publicly available web services developed at MIT along with providing a brief descriptions of their usage. This registry is unique in that it is created exclusively for the use of MIT Community, that is, the service providers as well as the service clients belong to the MIT Community. The site also provides documents/white papers that facilitate the development, deployment, and usage of web services. Thus, the main aim of this site is to increase the use of Web Services in education at MIT and to improve the quality of life for the students at MIT.

Upcoming Events

Application Integration & Web Services Summit 2004
17-19 May 2004, Los Angeles , CA, Venue: Westin Century Plaza Hotel
http://www4.gartner.com/2_events/conferences/apn12.jsp

Gartner Application Integration & Web Services Summit
7-8 June 2004, Amsterdam, Venue: Hotel Okura
http://www4.gartner.com/2_events/conferences/apps7i_section.jsp

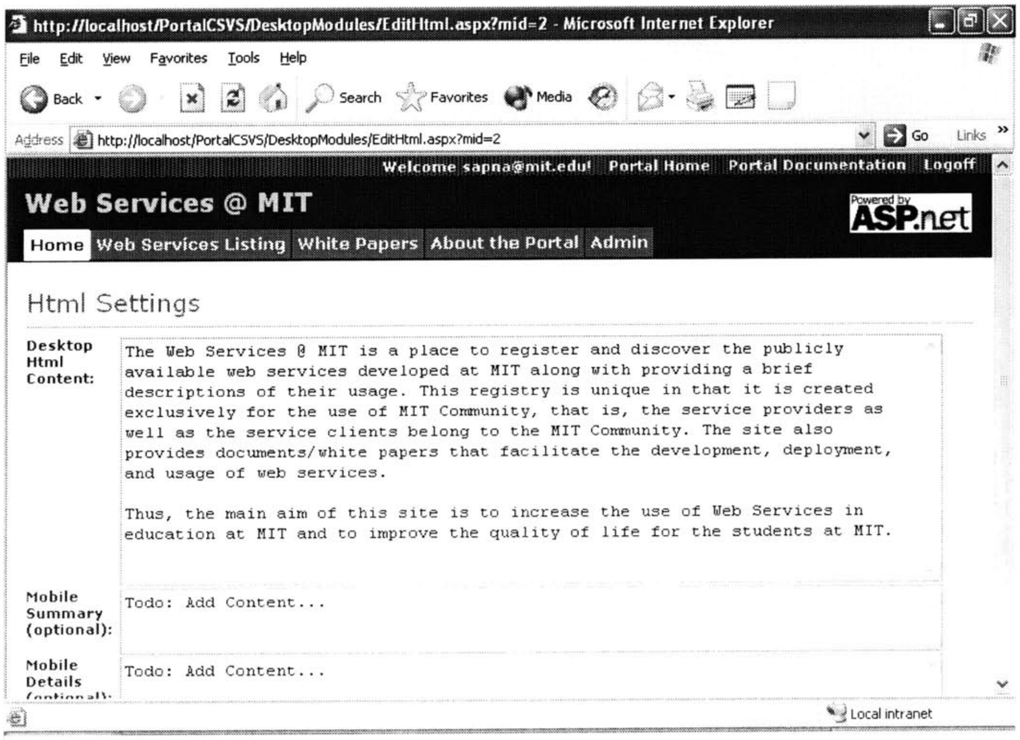
The next four screenshots display the input mechanism for Link, HtmlText, Events and Documents modules.


http://localhost/PortalCSVS/DesktopModules/EditLinks.aspx?ItemID=13&mid=1 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/PortalCSVS/DesktopModules/EditLinks.aspx?ItemID=13&mid=1
Go Links
Welcome sapna@mit.edu! Portal Home Portal Documentation Logoff
Web Services @ MIT
Powered by ASP.net
Home Web Services Listing White Papers About the Portal Admin
Update Cancel Delete this item
Done
Local intranet

Link Details

Title:	MIT
Url:	http://web.mit.edu
Mobile Url:	
Description:	
View Order:	1

Created by sapna@mit.edu on 5/2/2004


http://localhost/PortalCSVS/DesktopModules/EditHtml.aspx?mid=2 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/PortalCSVS/DesktopModules/EditHtml.aspx?mid=2
Go Links
Welcome sapna@mit.edu! Portal Home Portal Documentation Logoff
Web Services @ MIT
Powered by ASP.net
Home Web Services Listing White Papers About the Portal Admin
Local intranet

Html Settings

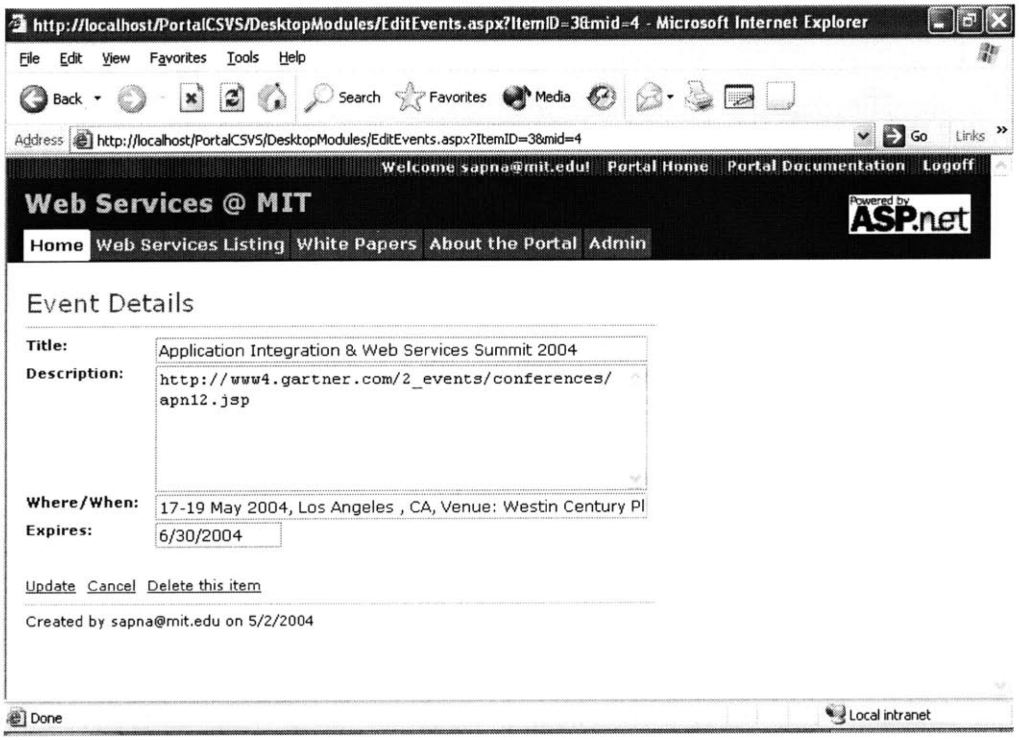
Desktop Html Content:

The Web Services @ MIT is a place to register and discover the publicly available web services developed at MIT along with providing a brief descriptions of their usage. This registry is unique in that it is created exclusively for the use of MIT Community, that is, the service providers as well as the service clients belong to the MIT Community. The site also provides documents/white papers that facilitate the development, deployment, and usage of web services.

Thus, the main aim of this site is to increase the use of Web Services in education at MIT and to improve the quality of life for the students at MIT.

Mobile Summary (optional): Todo: Add Content...

Mobile Details Todo: Add Content...


http://localhost/PortalCSVS/DesktopModules/EditEvents.aspx?ItemID=3&mid=4 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/PortalCSVS/DesktopModules/EditEvents.aspx?ItemID=3&mid=4
Go Links
Welcome sapna@mit.edu! Portal Home Portal Documentation Logoff
Web Services @ MIT
Powered by ASP.net
Home Web Services Listing White Papers About the Portal Admin
Update Cancel Delete this item
Done
Local intranet

Event Details

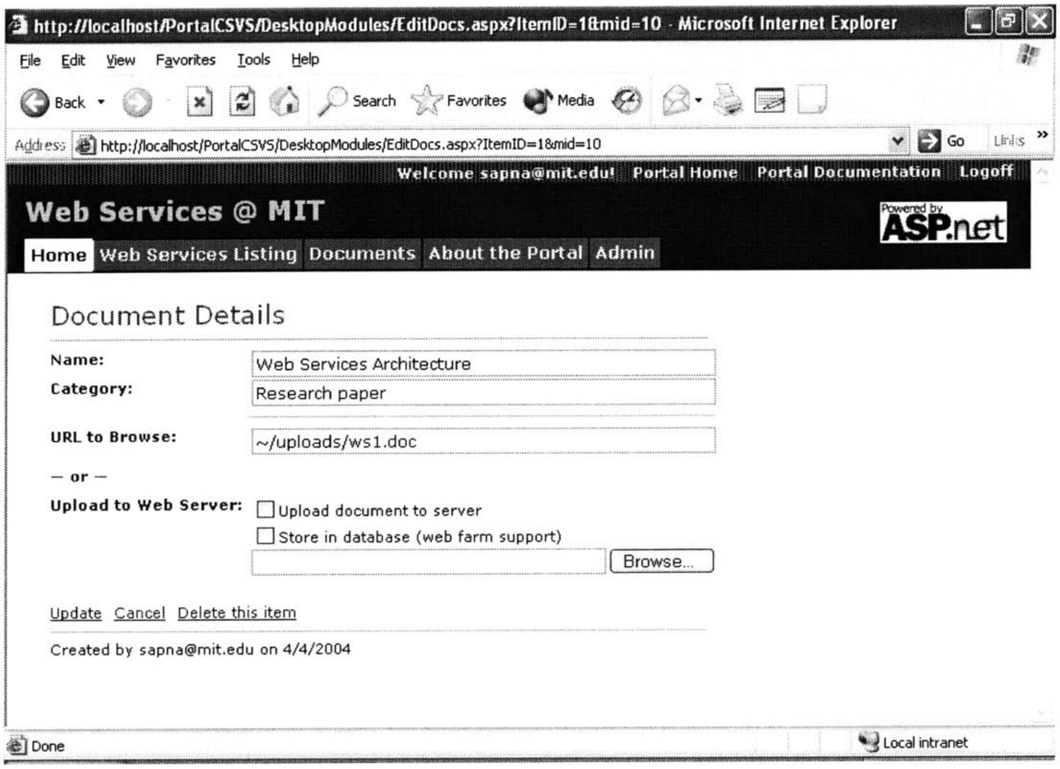
Title: Application Integration & Web Services Summit 2004

Description: http://www4.gartner.com/2_events/conferences/apn12.jsp

Where/When: 17-19 May 2004, Los Angeles , CA, Venue: Westin Century Pl

Expires: 6/30/2004

Created by sapna@mit.edu on 5/2/2004

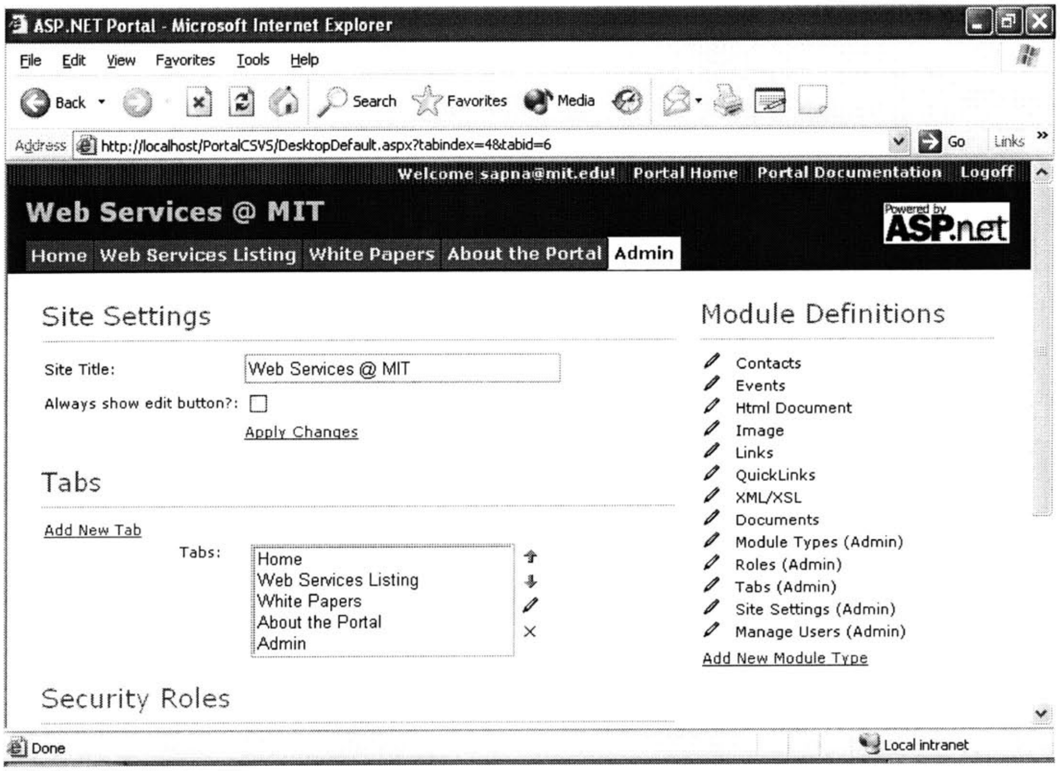

http://localhost/PortalCSVS/DesktopModules/EditDocs.aspx?ItemID=1&mid=10 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/PortalCSVS/DesktopModules/EditDocs.aspx?ItemID=1&mid=10
Go Links
Welcome sapna@mit.edu! Portal Home Portal Documentation Logoff
Web Services @ MIT
Powered by ASP.net
Home Web Services Listing Documents About the Portal Admin
Browse...
Update Cancel Delete this item
Done
Local intranet

Document Details

Name: Web Services Architecture
Category: Research paper
URL to Browse: ~/uploads/ws1.doc
— or —
Upload to Web Server: ☐ Upload document to server
☐ Store in database (web farm support)

Created by sapna@mit.edu on 4/4/2004

The next six screen shots display the interface through which the admin can change the site lay out, tab lay out, module lay out, assign roles to the users, create new roles as well as add and delete users.


ASP.NET Portal - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/PortalCSVS/DesktopDefault.aspx?tabindex=4&tabid=6
Go Links
Welcome sapna@mit.edu! Portal Home Portal Documentation Logoff
Powered by ASP.net
Home Web Services Listing White Papers About the Portal Admin
Done
Local intranet

Web Services @ MIT

Site Settings

Site Title: Web Services @ MIT

Always show edit button?: ☐

Apply Changes

Tabs

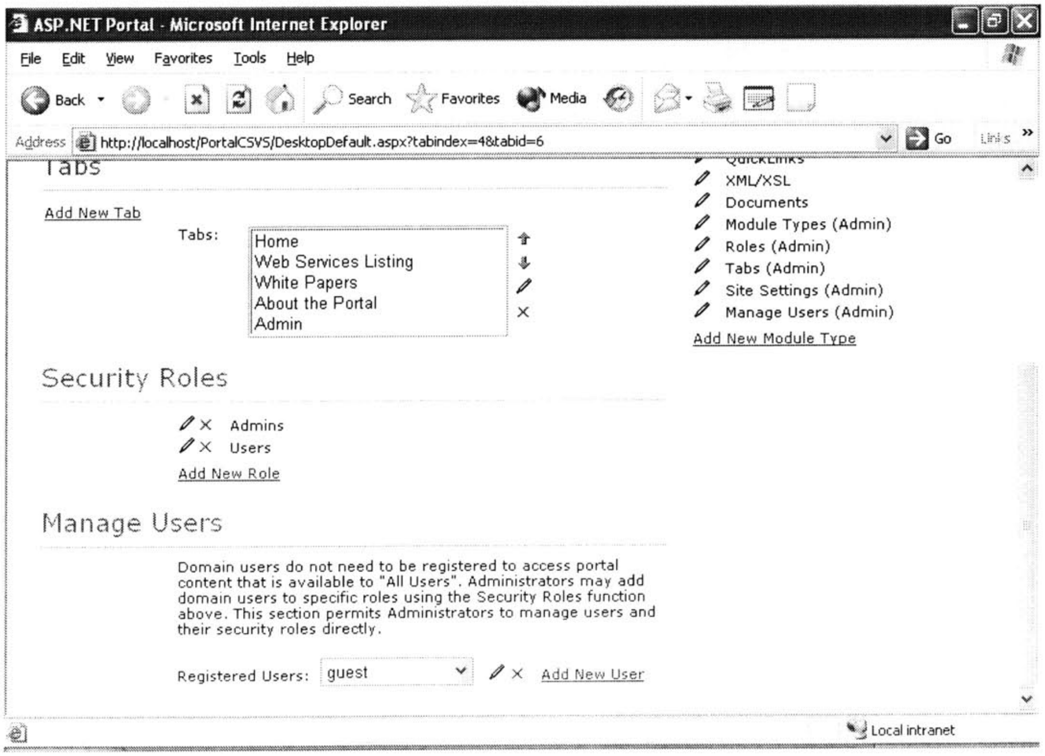
Add New Tab

Tabs:
- Home
- Web Services Listing
- White Papers
- About the Portal
- Admin

Security Roles

Module Definitions
- Contacts
- Events
- Html Document
- Image
- Links
- QuickLinks
- XML/XSL
- Documents
- Module Types (Admin)
- Roles (Admin)
- Tabs (Admin)
- Site Settings (Admin)
- Manage Users (Admin)

Add New Module Type


ASP.NET Portal - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/PortalCSVS/DesktopDefault.aspx?tabindex=4&tabid=6
Go Links
Local intranet

Tabs

Add New Tab

Tabs:
- Home
- Web Services Listing
- White Papers
- About the Portal
- Admin

Security Roles
- Admins
- Users

Add New Role

Manage Users

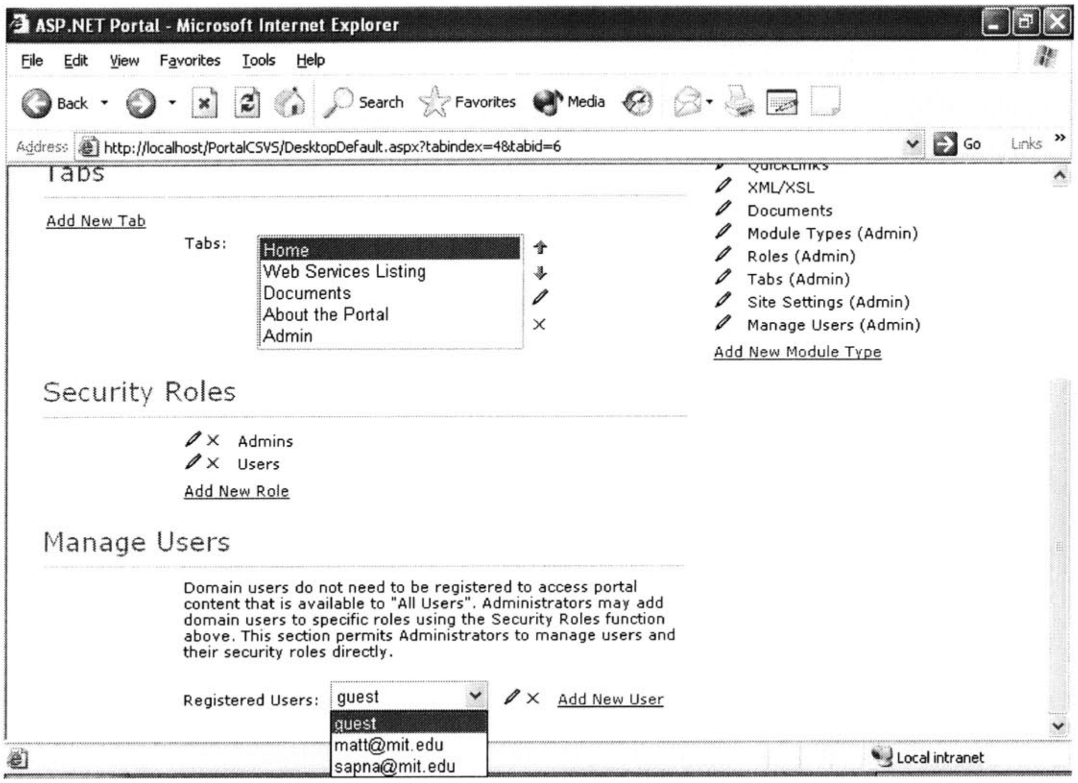
Domain users do not need to be registered to access portal content that is available to "All Users". Administrators may add domain users to specific roles using the Security Roles function above. This section permits Administrators to manage users and their security roles directly.

Registered Users: guest

Add New User

- XML/XSL
- Documents
- Module Types (Admin)
- Roles (Admin)
- Tabs (Admin)
- Site Settings (Admin)
- Manage Users (Admin)

Add New Module Type


ASP.NET Portal - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/PortalCSVS/DesktopDefault.aspx?tabindex=4&tabid=6
Go Links
Local intranet

Tabs

Add New Tab

Tabs:
Home
Web Services Listing
Documents
About the Portal
Admin

QuickLinks
XML/XSL
Documents
Module Types (Admin)
Roles (Admin)
Tabs (Admin)
Site Settings (Admin)
Manage Users (Admin)
Add New Module Type

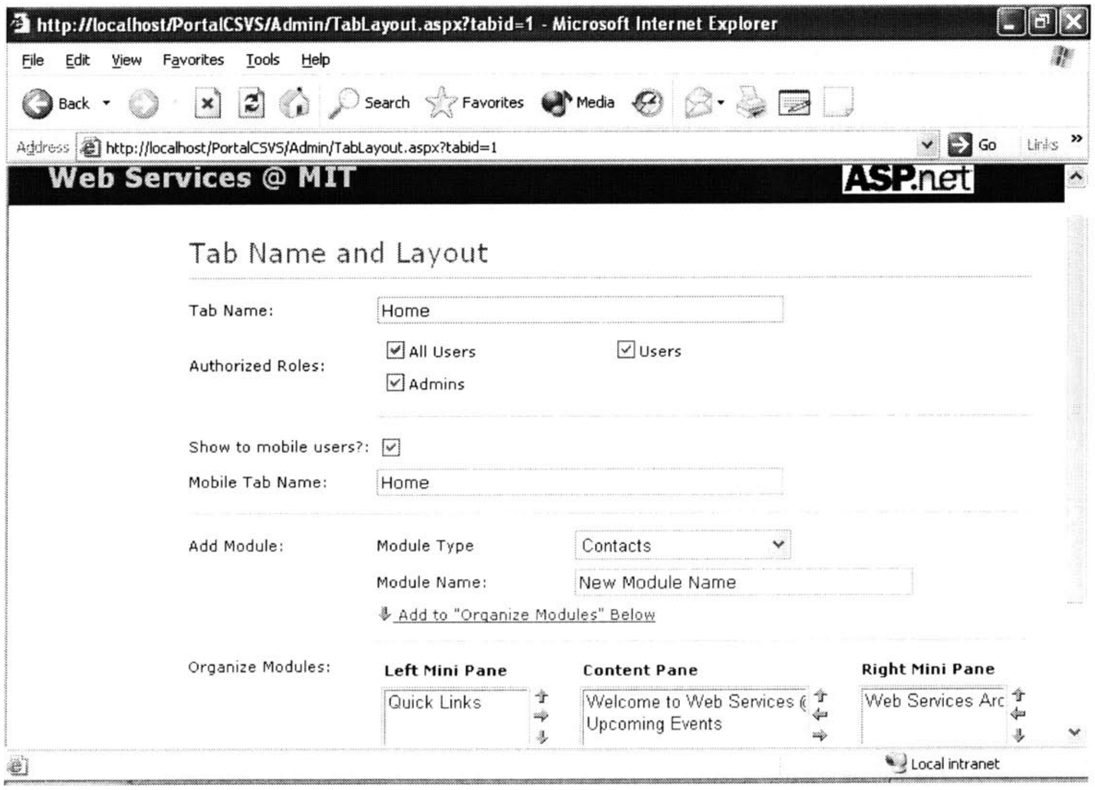
Security Roles

Admins
Users
Add New Role

Manage Users

Domain users do not need to be registered to access portal content that is available to "All Users". Administrators may add domain users to specific roles using the Security Roles function above. This section permits Administrators to manage users and their security roles directly.

Registered Users: guest
guest
matt@mit.edu
sapna@mit.edu
Add New User


http://localhost/PortalCSVS/Admin/TabLayout.aspx?tabid=1 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/PortalCSVS/Admin/TabLayout.aspx?tabid=1
Go Links
Web Services @ MIT
ASP.net
Local intranet

Tab Name and Layout

Tab Name: Home

Authorized Roles: ☑ All Users ☑ Users ☑ Admins

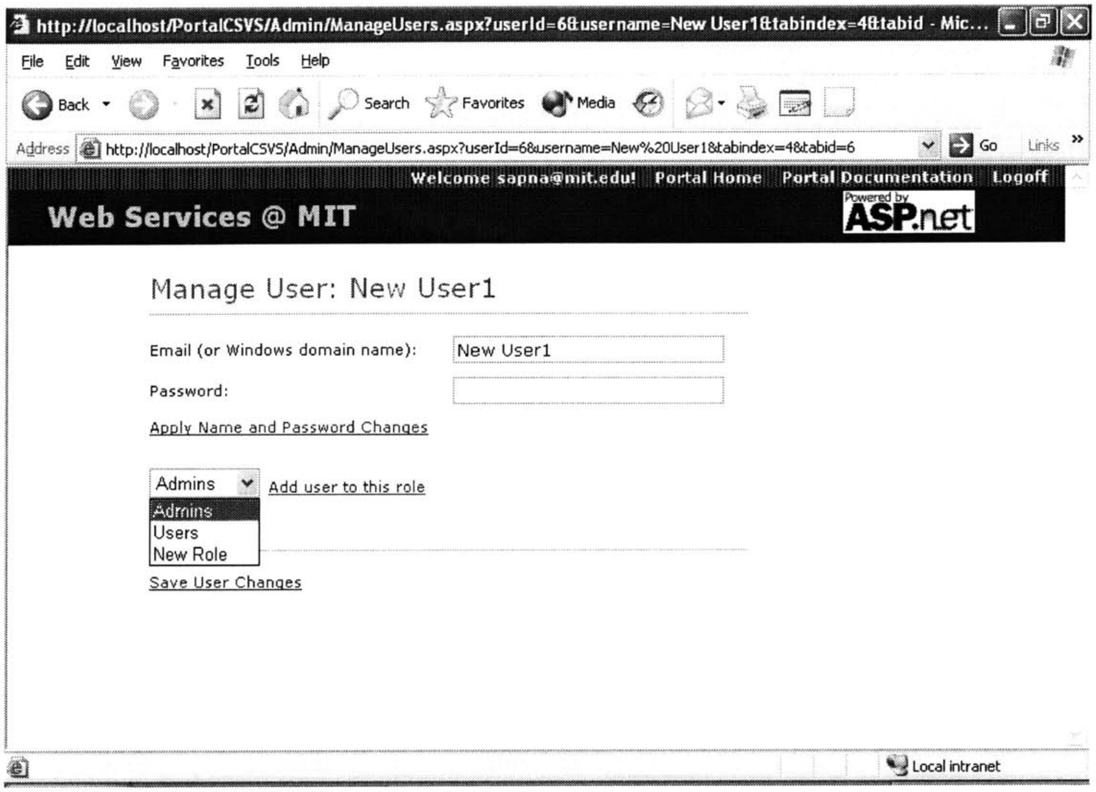
Show to mobile users?: ☑

Mobile Tab Name: Home

Add Module: Module Type Contacts
Module Name: New Module Name
Add to "Organize Modules" Below

Organize Modules:

Left Mini Pane	Content Pane	Right Mini Pane
Quick Links	Welcome to Web Services (Upcoming Events	Web Services Arc


File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Go
Links
Welcome sapna@mit.edu! Portal Home Portal Documentation Logoff
Web Services @ MIT
Powered by ASP.net
Apply Name and Password Changes
Add user to this role
Save User Changes
Local intranet

http://localhost/PortalCSVS/Admin/ManageUsers.aspx?userId=6&username=New User1&tabindex=4&tabid - Mic...

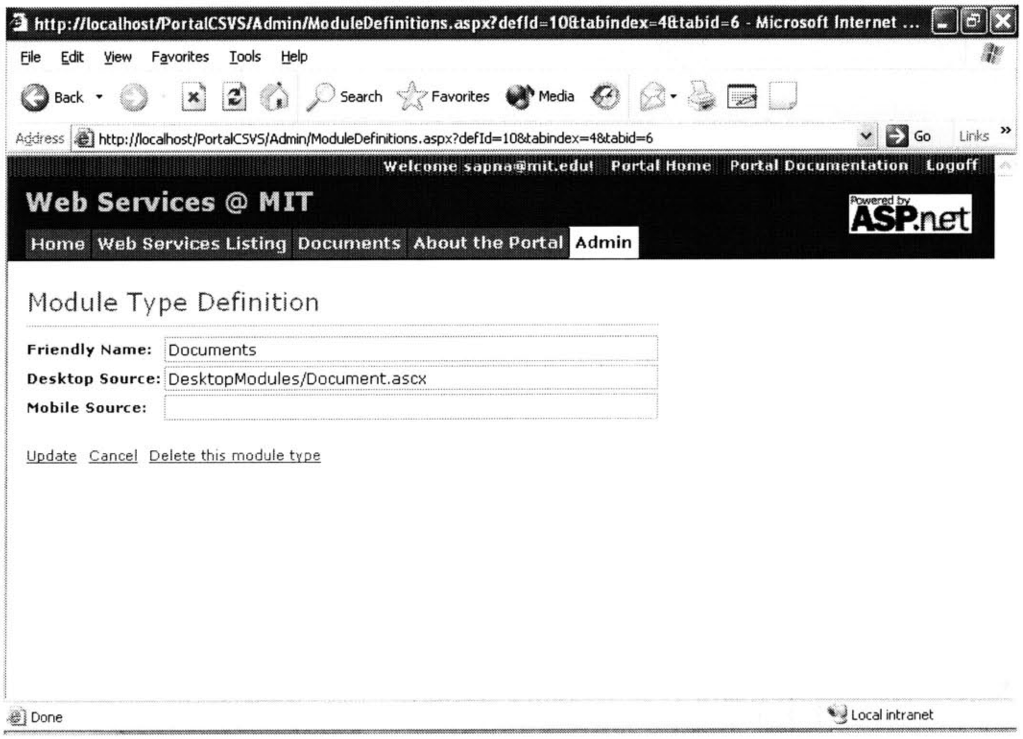
Address http://localhost/PortalCSVS/Admin/ManageUsers.aspx?userId=6&username=New%20User1&tabindex=4&tabid=6

Manage User: New User1

Email (or Windows domain name): New User1

Password:

Admins
- Admins
- Users
- New Role


File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Go
Links
Welcome sapna@mit.edu! Portal Home Portal Documentation Logoff
Web Services @ MIT
Powered by ASP.net
Home Web Services Listing Documents About the Portal Admin
Update Cancel Delete this module type
Done
Local intranet

http://localhost/PortalCSVS/Admin/ModuleDefinitions.aspx?defId=10&tabindex=4&tabid=6 - Microsoft Internet ...

Address http://localhost/PortalCSVS/Admin/ModuleDefinitions.aspx?defId=10&tabindex=4&tabid=6

Module Type Definition

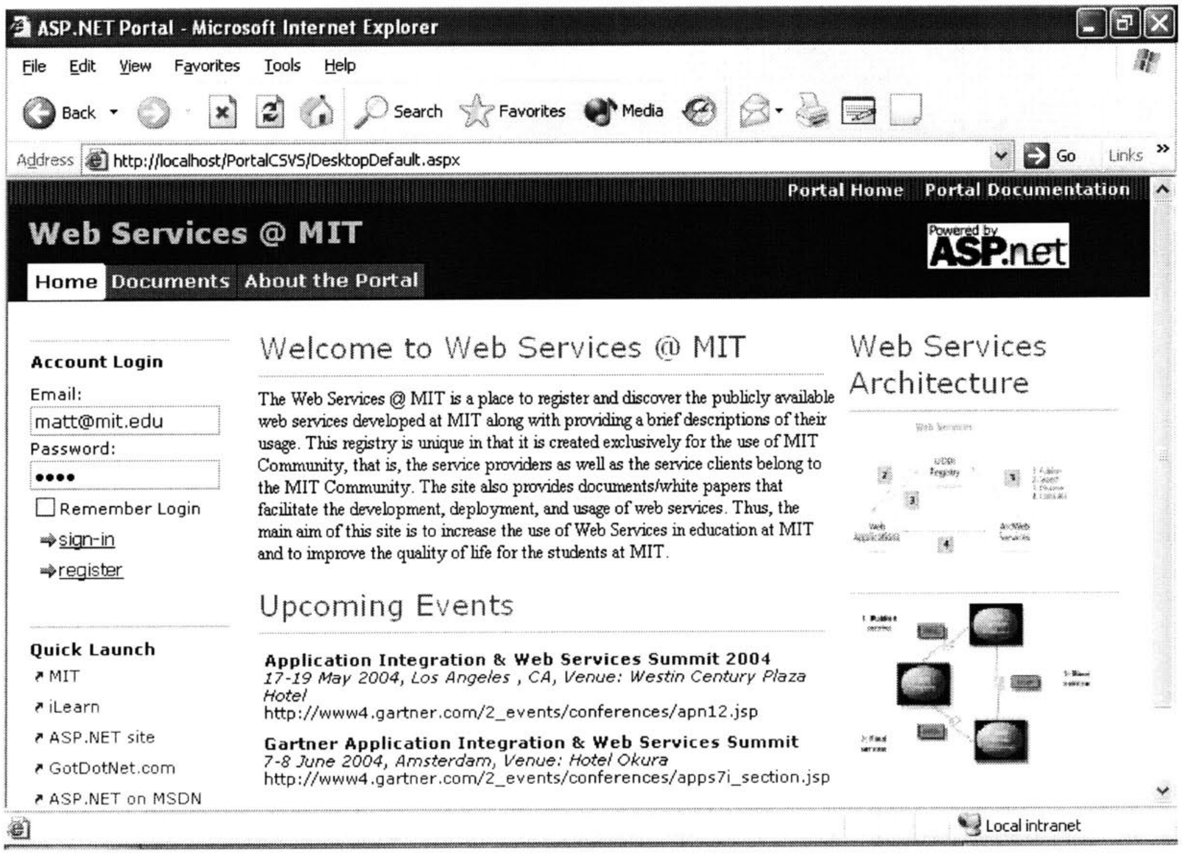
Friendly Name: Documents

Desktop Source: DesktopModules/Document.ascx

Mobile Source:

A.2 Non-Admin User – matt@mit.edu

The next two screen shots display the interface for .a non-admin user before and after login. It can be observed in the second screen shot that the admin tab is not visible to the non-admin user.



Email: matt@mit.edu

Password: ●●●●

☐ Remember Login

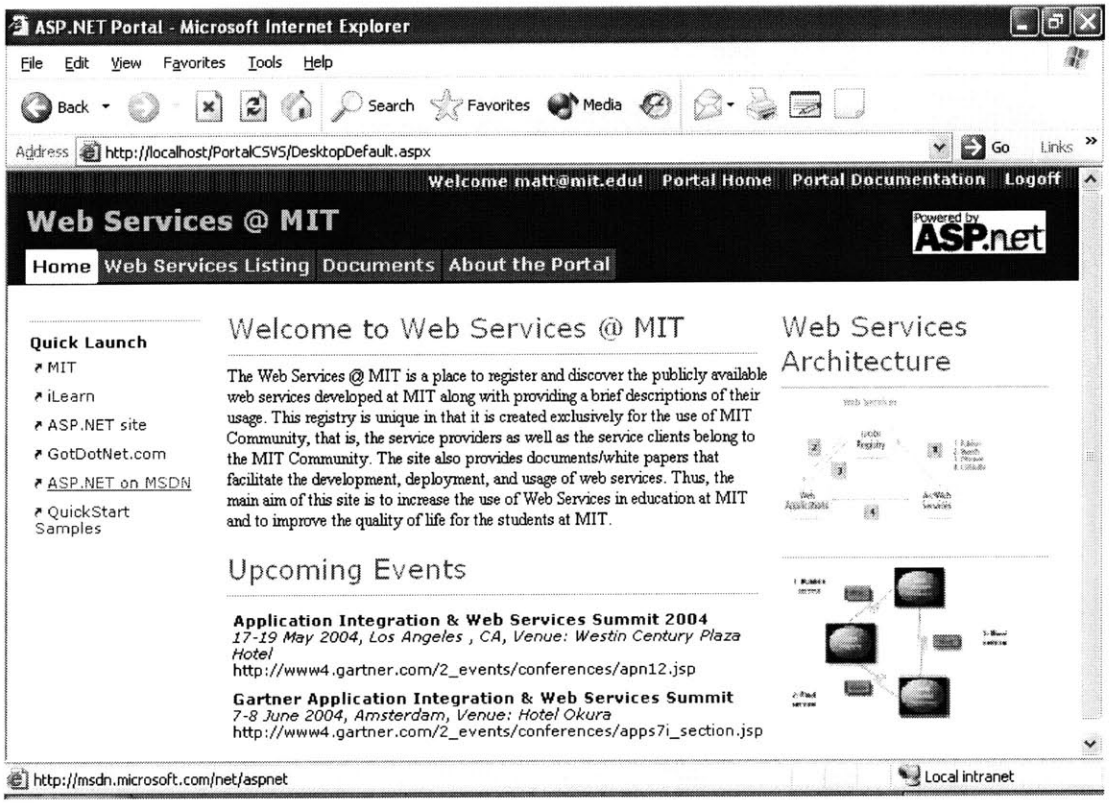
Welcome to Web Services @ MIT

The Web Services @ MIT is a place to register and discover the publicly available web services developed at MIT along with providing a brief descriptions of their usage. This registry is unique in that it is created exclusively for the use of MIT Community, that is, the service providers as well as the service clients belong to the MIT Community. The site also provides documents/white papers that facilitate the development, deployment, and usage of web services. Thus, the main aim of this site is to increase the use of Web Services in education at MIT and to improve the quality of life for the students at MIT.

Upcoming Events

Application Integration & Web Services Summit 2004
17-19 May 2004, Los Angeles , CA, Venue: Westin Century Plaza Hotel
http://www4.gartner.com/2_events/conferences/apn12.jsp

Gartner Application Integration & Web Services Summit
7-8 June 2004, Amsterdam, Venue: Hotel Okura
http://www4.gartner.com/2_events/conferences/apps7i_section.jsp


ASP.NET Portal - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/PortalCSVS/DesktopDefault.aspx
Go Links
Welcome matt@mit.edu! Portal Home Portal Documentation Logoff
Web Services @ MIT
Powered by ASP.net
Home Web Services Listing Documents About the Portal
Quick Launch
MIT
iLearn
ASP.NET site
GotDotNet.com
ASP.NET on MSDN
QuickStart Samples
Web Services Architecture
http://msdn.microsoft.com/net/aspnet
Local intranet

Welcome to Web Services @ MIT

The Web Services @ MIT is a place to register and discover the publicly available web services developed at MIT along with providing a brief descriptions of their usage. This registry is unique in that it is created exclusively for the use of MIT Community, that is, the service providers as well as the service clients belong to the MIT Community. The site also provides documents/white papers that facilitate the development, deployment, and usage of web services. Thus, the main aim of this site is to increase the use of Web Services in education at MIT and to improve the quality of life for the students at MIT.

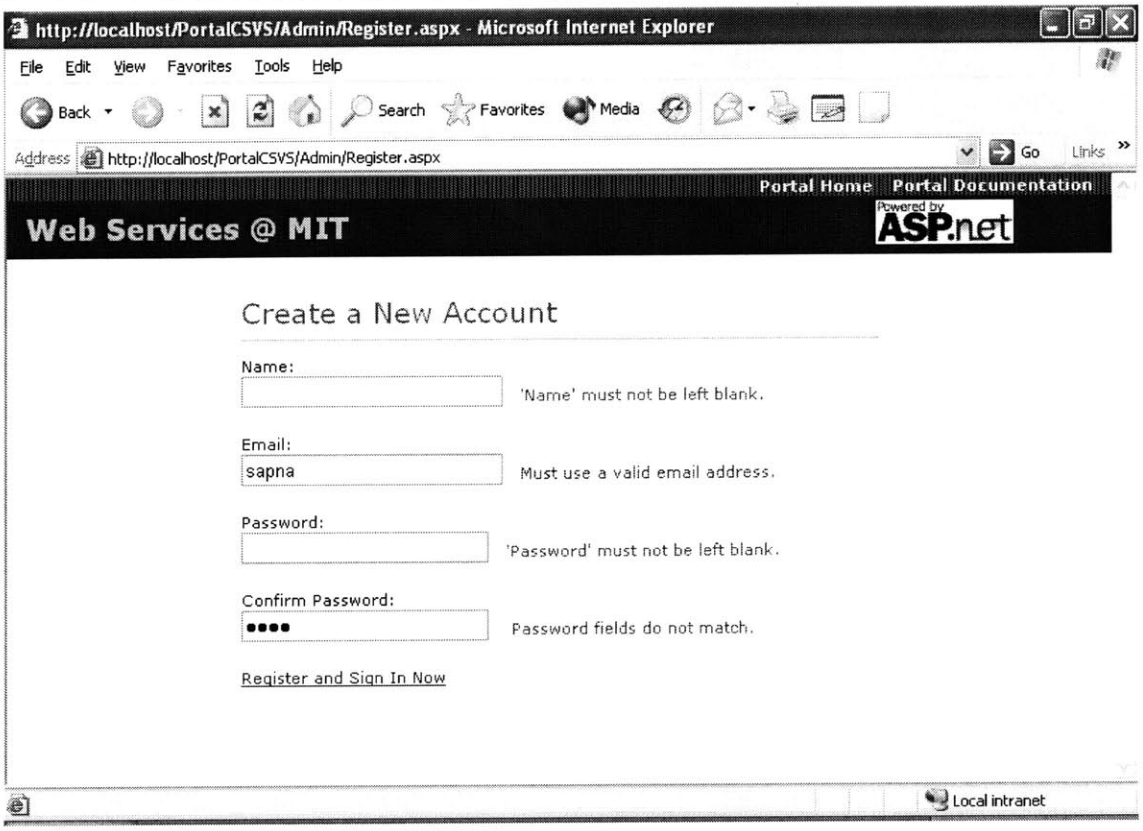
Upcoming Events

Application Integration & Web Services Summit 2004
17-19 May 2004, Los Angeles , CA, Venue: Westin Century Plaza Hotel
http://www4.gartner.com/2_events/conferences/apn12.jsp

Gartner Application Integration & Web Services Summit
7-8 June 2004, Amsterdam, Venue: Hotel Okura
http://www4.gartner.com/2_events/conferences/apps7i_section.jsp

A.3 Input validations

The next five screen shot display the input validations used to ensure the accuracy of user information in the system.


http://localhost/PortalCSVS/Admin/Register.aspx - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address
http://localhost/PortalCSVS/Admin/Register.aspx
Go
Links
Portal Home
Portal Documentation
Powered by ASP.net
Local intranet

Web Services @ MIT

Create a New Account

Name:

'Name' must not be left blank.

Email: sapna

Must use a valid email address.

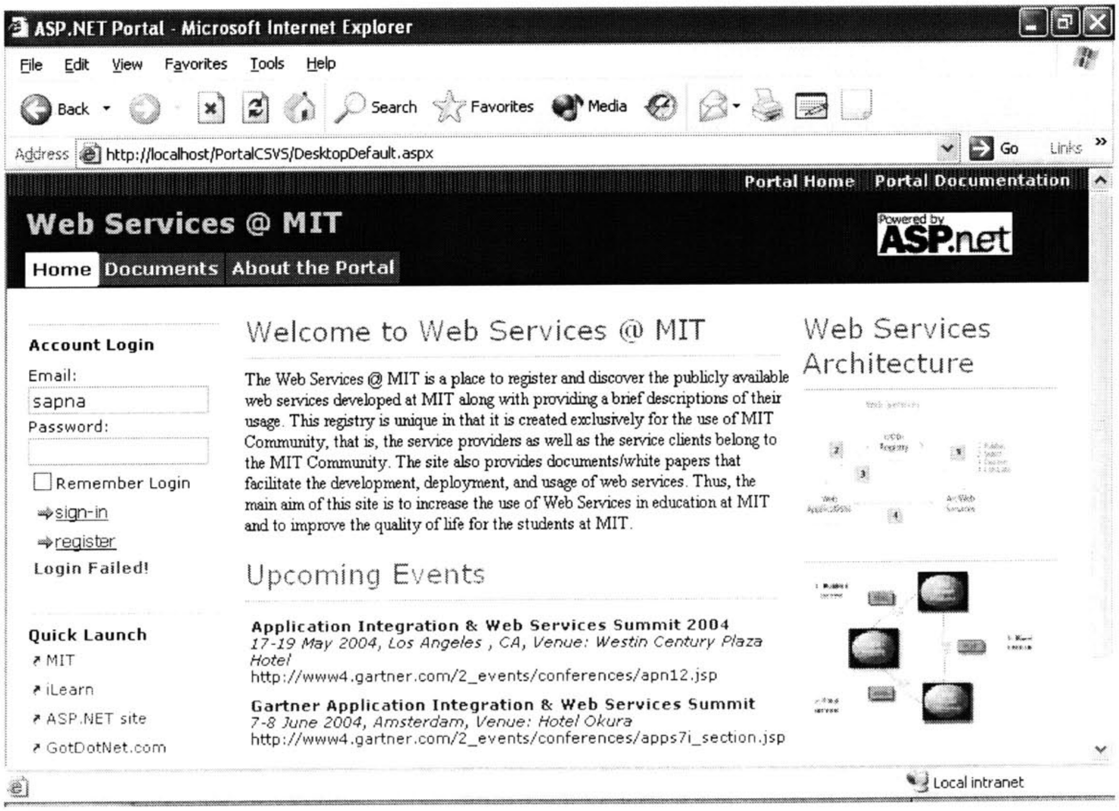
Password:

'Password' must not be left blank.

Confirm Password: ●●●●

Password fields do not match.

Register and Sign In Now



Web Services @ MIT

Account Login

Email: sapna

Password:

☐ Remember Login

sign-in

register

Login Failed!

Quick Launch
- MIT
- iLearn
- ASP.NET site
- GotDotNet.com

Welcome to Web Services @ MIT

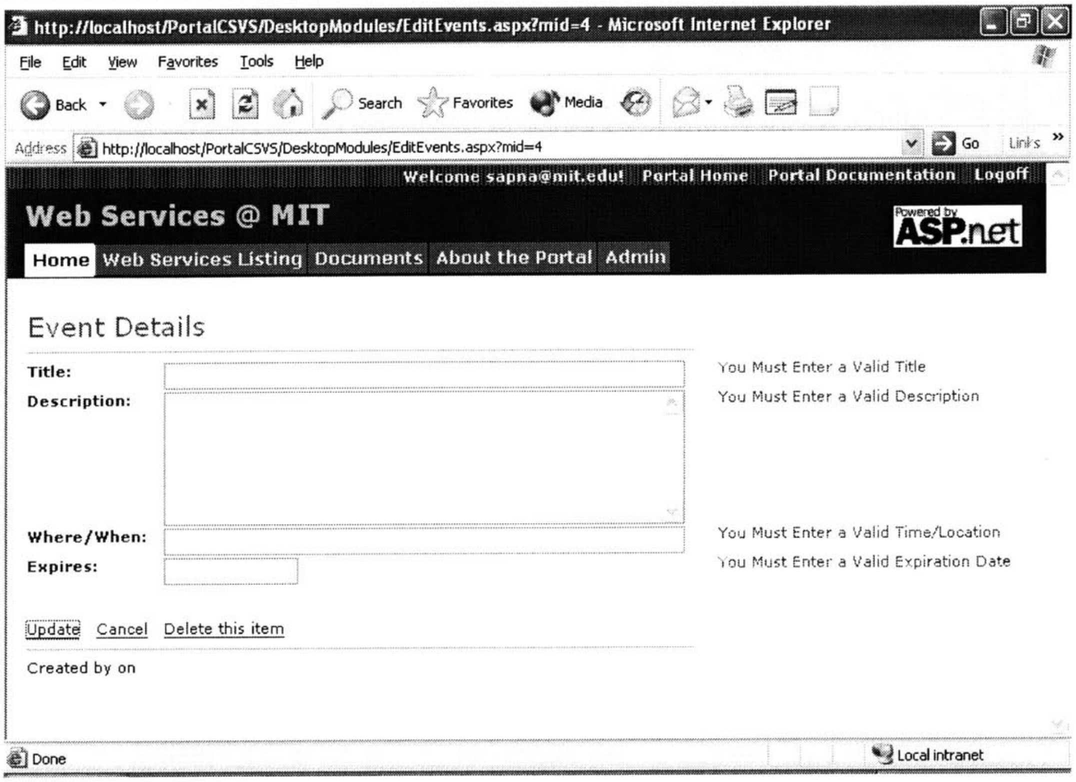
The Web Services @ MIT is a place to register and discover the publicly available web services developed at MIT along with providing a brief descriptions of their usage. This registry is unique in that it is created exclusively for the use of MIT Community, that is, the service providers as well as the service clients belong to the MIT Community. The site also provides documents/white papers that facilitate the development, deployment, and usage of web services. Thus, the main aim of this site is to increase the use of Web Services in education at MIT and to improve the quality of life for the students at MIT.

Upcoming Events

Application Integration & Web Services Summit 2004
17-19 May 2004, Los Angeles , CA, Venue: Westin Century Plaza Hotel
http://www4.gartner.com/2_events/conferences/apn12.jsp

Gartner Application Integration & Web Services Summit
7-8 June 2004, Amsterdam, Venue: Hotel Okura
http://www4.gartner.com/2_events/conferences/apps7i_section.jsp

Web Services Architecture


http://localhost/PortalCSVS/DesktopModules/EditEvents.aspx?mid=4 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/PortalCSVS/DesktopModules/EditEvents.aspx?mid=4
Go Links
Welcome sapna@mit.edu! Portal Home Portal Documentation Logoff
Web Services @ MIT
Powered by ASP.net
Home Web Services Listing Documents About the Portal Admin
Done
Local intranet

Event Details

Title: You Must Enter a Valid Title

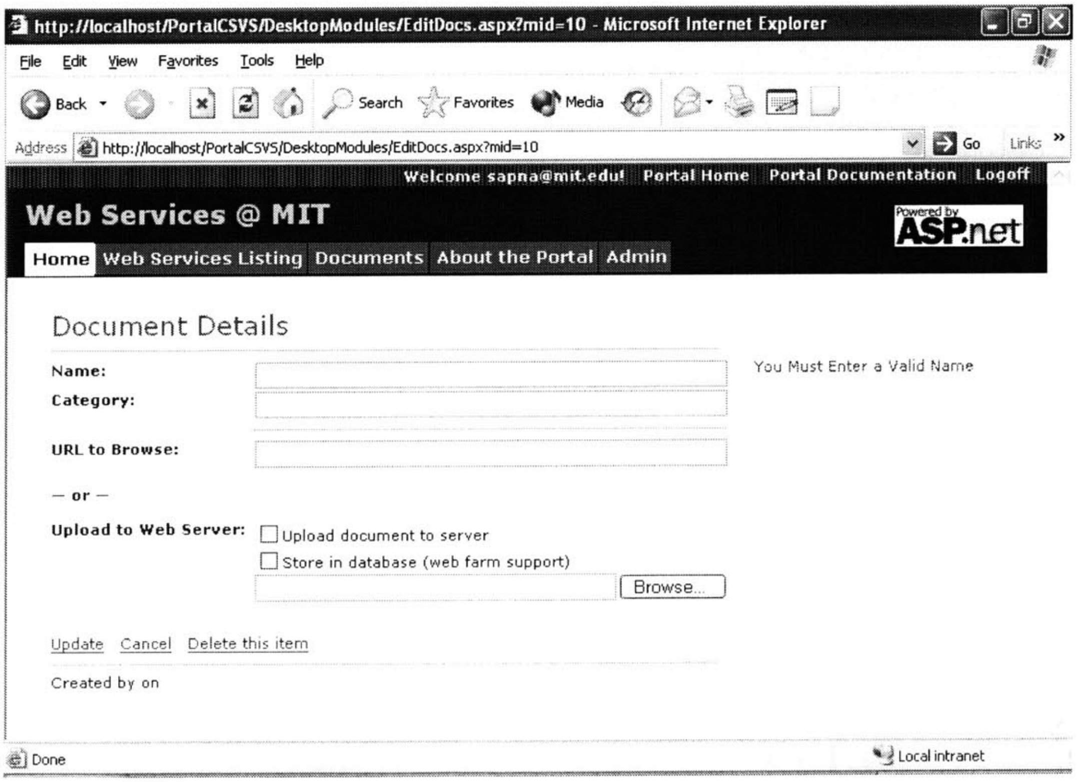
Description: You Must Enter a Valid Description

Where/When: You Must Enter a Valid Time/Location

Expires: You Must Enter a Valid Expiration Date

Update Cancel Delete this item

Created by on


http://localhost/PortalCSVS/DesktopModules/EditDocs.aspx?mid=10 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/PortalCSVS/DesktopModules/EditDocs.aspx?mid=10
Go Links
Welcome sapna@mit.edu! Portal Home Portal Documentation Logoff
Web Services @ MIT
Powered by ASP.net
Home Web Services Listing Documents About the Portal Admin
Done
Local intranet

Document Details

Name: You Must Enter a Valid Name

Category:

URL to Browse:

— or —

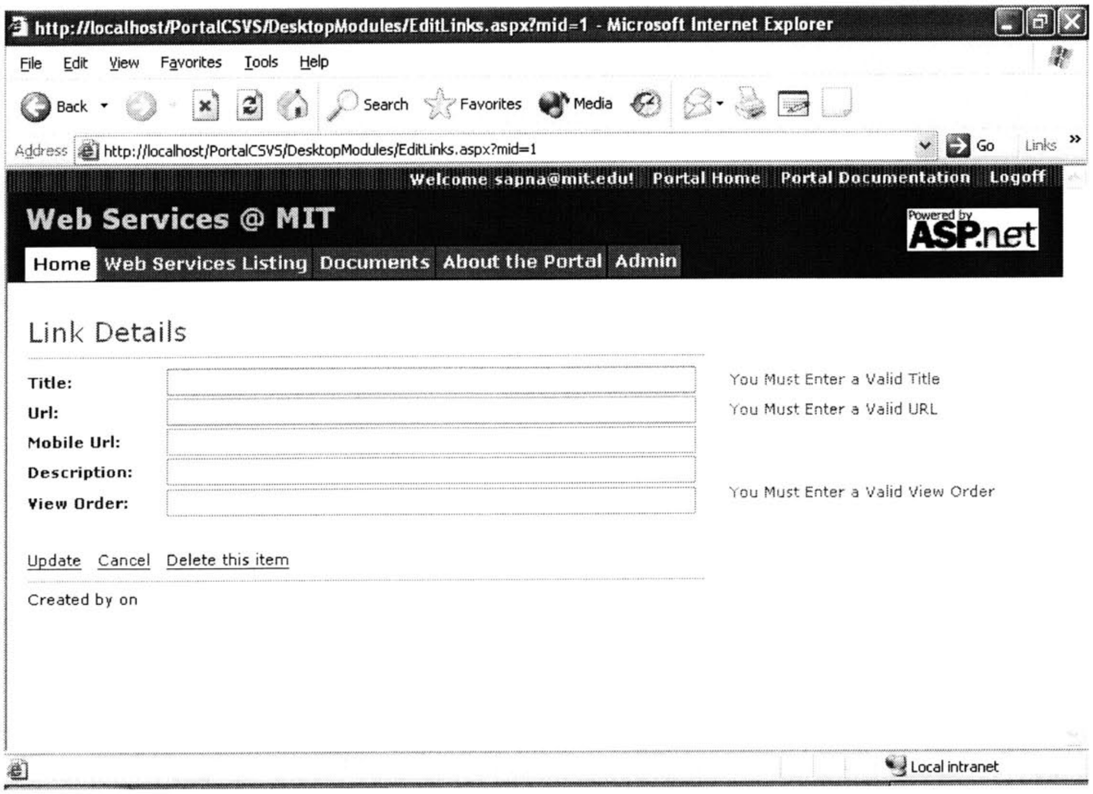
Upload to Web Server: ☐ Upload document to server

☐ Store in database (web farm support)

Browse...

Update Cancel Delete this item

Created by on


http://localhost/PortalCSVS/DesktopModules/EditLinks.aspx?mid=1 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/PortalCSVS/DesktopModules/EditLinks.aspx?mid=1
Go Links
Welcome sapna@mit.edu! Portal Home Portal Documentation Logoff
Web Services @ MIT
Powered by ASP.net
Home Web Services Listing Documents About the Portal Admin
Local intranet

Link Details

Title:		You Must Enter a Valid Title
Url:		You Must Enter a Valid URL
Mobile Url:		
Description:		
View Order:		You Must Enter a Valid View Order

Update Cancel Delete this item

Created by on