1.264 Lecture 10

Data and the Web

Role of Web server in supply chain, engineering collaboration, etc.

- Web server and its pages are a communications channel between databases
  - Or, if you prefer, a protocol converter between databases
- We cover the human-readable elements of the Web, HTML, and use them in the homework because they are visual and intuitive
  - In real life, we will use XML documents instead of HTML pages: they are machine-readable
  - And we will exchange XML documents between databases, with as little human intervention as possible
    - Much is still needed because there isn't complete trust among trading partners, engineering collaborators, etc.
    - We'll have more to say about this under 'Security'
- In real life, we'd almost always use database queries in SQL to fill in 'forms' and send them to the next database, not human input
  - But we'll do forms for starters here

HTML forms

- Used as front ends to Web server programs
- Forms are user interface controls to collect data from user and transmit it to the server application program
  - Control is a 'child window' within a 'parent window' or page
- Forms are most common user interface method
  - Forms are placed on Web pages that can also have other elements
  - JavaScript (totally different than Java language!) can be used with HTML forms to enhance their operation
  - Java applets can also provide a richer user interface, are more complex and have security features. Not used much.
  - All of these run on the browser (on the client machine) and are user interface components
  - Note the asymmetry:
    - Data goes from the client to the server and is stored there
    - Data sent from the server, if any, is not stored on the client. Why?
    - "Normalization" of a sort: don't store the same data in more than one place or it will get out of sync

How HTML forms transmit data

- Forms allow a series of controls to be placed on the page
  - Controls are text boxes, dropdowns, radio buttons, check boxes…
  - Each control has a name and a value
- Form data is sent when user presses 'Submit' button. Three options:
  - Data is sent to URL with HTTP POST command as string of form:
    - `Name1=Value1&Name2=Value2&…NameN=ValueN`
    - POST data is sent after the blank line after HTTP headers, as a string
  - Data can alternatively be sent to URL with HTTP GET command, appended to end of GET string after a ?:
    - `GET/Index.html?Name1=Value1&Name2=Value2&NameN=ValueN …`
  - SOAP is an extension to HTTP to allow XML documents to be sent by POST (or GET) commands instead of plain text strings
    - We'll cover XML later but it allows validation and business rules
- Form's 'Submit' button is associated with the URL of a server program that will process the input data
  - Server programs have methods (function points!) to extract the data from the string and use it in the program
    - See why we counted Web pages and estimated function points?

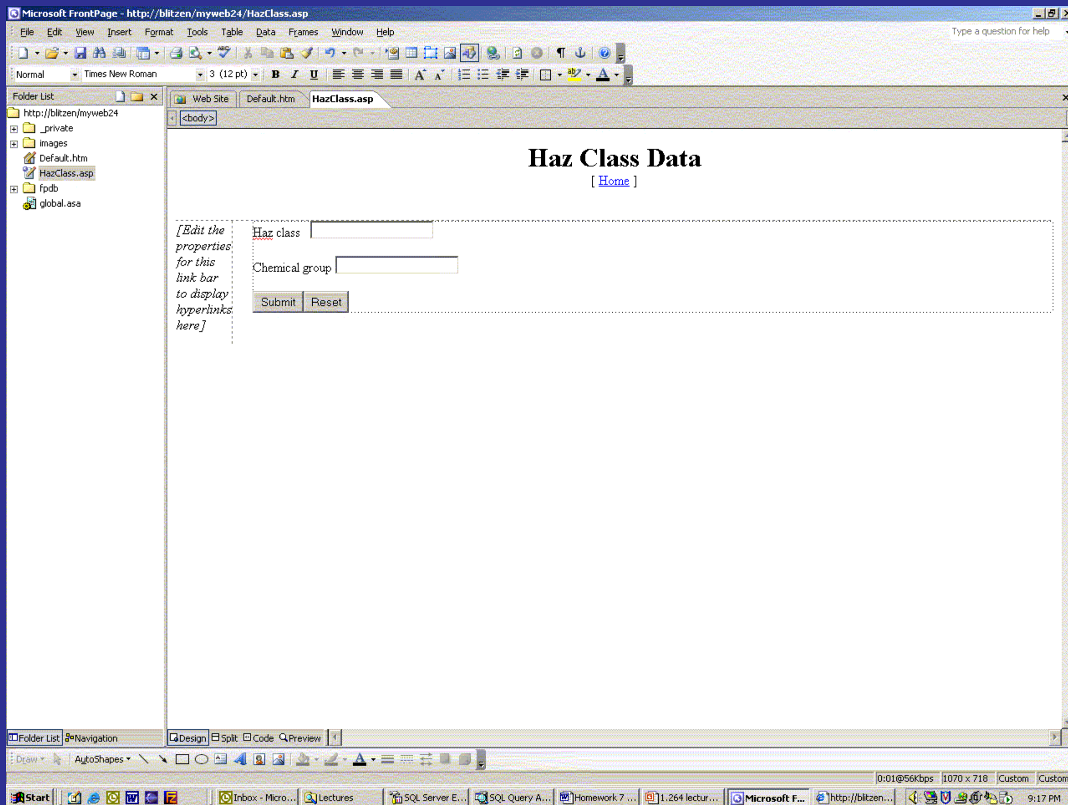
HTML 4.0 tags for forms

Tag	Type	Definition
<FORM>		Start a form
<INPUT TYPE=	text	Single line of text entry
	password	Single line password entry
	file	File to upload, with ‘Browse’ button
	checkbox	Checkbox
	radio	Radio button (option box)
	image	Image acting as button
	hidden	track user, store predefined input, etc.
	submit	Submit button for form
	reset	Button to restore default values
<SELECT>		List box or combo box
<OPTION>		Item in scrolling list or popup menu
<TEXTAREA>		Start multiple-line text entry field

HTML forms demo

- Front Page: File->New Site
  - Must create site on machine running a Web server:
    - course server, or
    - your computer, running MS Internet Information Server (IIS)
- Connect to SQL Server database on database server:
  - In FrontPage: Tools->Site Settings->Database
  - Name: Database1
  - Choose: Network connection to db server
  - Type of driver: SQL Server
  - Server name: arlington6
  - Database: MIT1264HW2004 [or whatever you called it]
  - Advanced: Username:…, password: ... Use your group’s.
  - Verify connection; should be ok.
- Build form on new page, HazClass.asp, for HazClass
  - Create new page, drag below home page (Default.htm), open it
  - Insert-> Form -> Textbox
  - Insert -> other components on form (see demo)
- Save and then test in browser

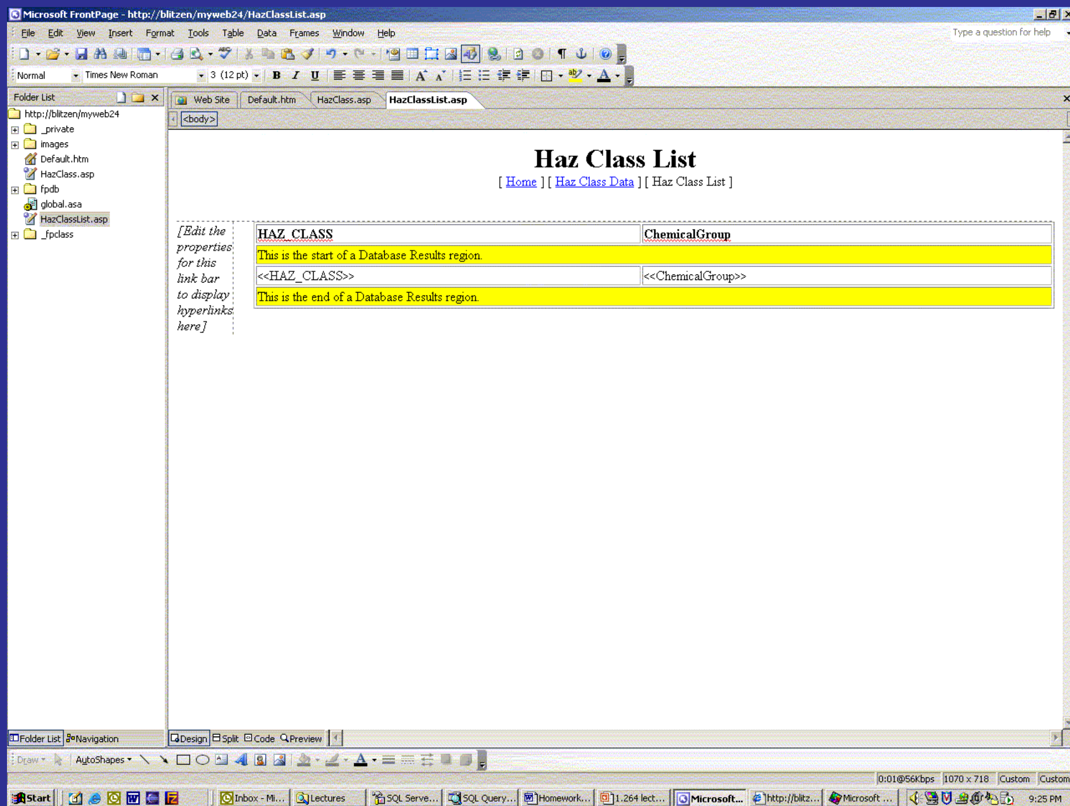
Demo



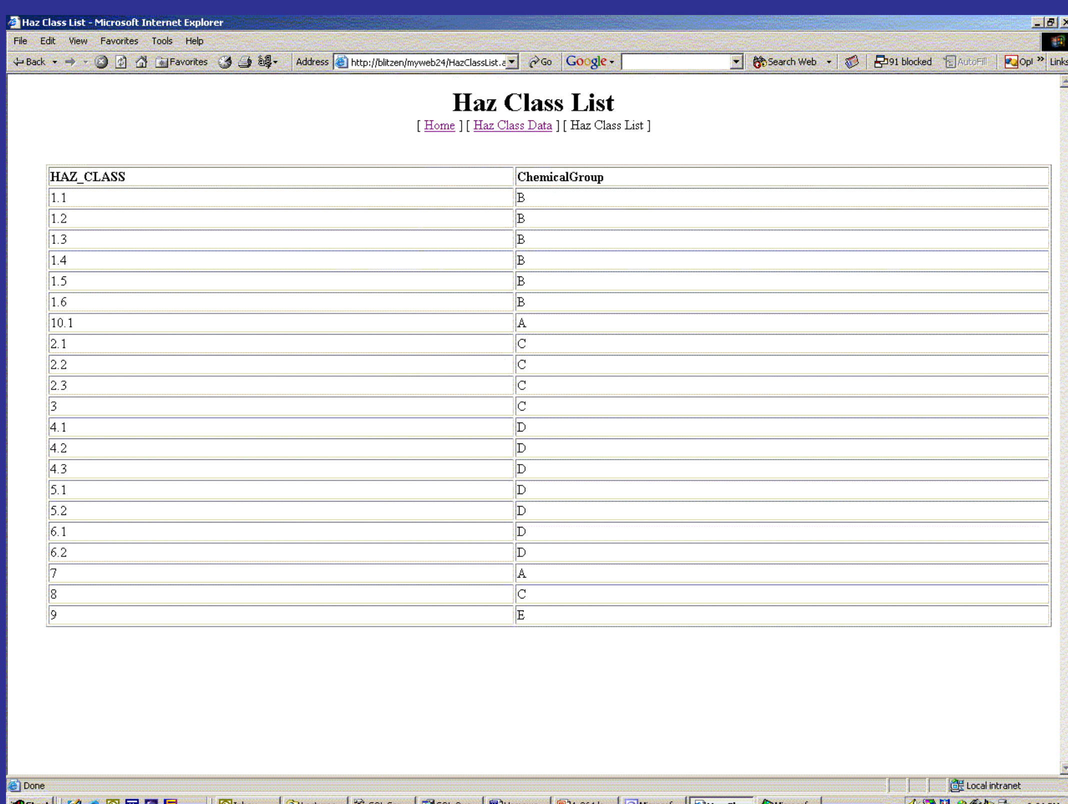
Demo, cont.

- We can add haz class 10.1 with a valid group
  - But not with an invalid group. Why?
- We can’t add haz class 1.1. Why?
- The database enforces integrity and data correctness, <u>not</u> the Web application.
  - Remember that, in supply chain or collaborative engineering or transportation operations, many applications will hit the same database
  - If only one application is wrong, it will hose all the data
- We’ll create a simple data query Web page to see what data we’ve added
  - FrontPage: create new page,
  - Drag it under home page, rename it HazClassList.asp
  - Insert->Database-> Result. Follow wizard
  - Save and test

Demo…


Microsoft FrontPage - http://blitzen/myweb24/HazClassList.asp
Folder List
http://blitzen/myweb24
_private
images
Default.htm
HazClass.asp
fpdb
global.asa
HazClassList.asp
_fpclass
Web Site
Default.htm
HazClass.asp
HazClassList.asp
Haz Class List
[Home] [Haz Class Data] [Haz Class List]
[Edit the properties for this link bar to display hyperlinks here]
HAZ_CLASS
ChemicalGroup
This is the start of a Database Results region.
<<HAZ_CLASS>>
<<ChemicalGroup>>
This is the end of a Database Results region.
Design
Split
Code
Preview
9:25 PM

Demo…


Haz Class List - Microsoft Internet Explorer
http://blitzen/myweb24/HazClassList.asp
Haz Class List
[Home] [Haz Class Data] [Haz Class List]
HAZ_CLASS
ChemicalGroup
1.1
B
1.2
B
1.3
B
1.4
B
1.5
B
1.6
B
10.1
A
2.1
C
2.2
C
2.3
C
3
C
4.1
D
4.2
D
4.3
D
5.1
D
5.2
D
6.1
D
6.2
D
7
A
8
C
9
E
Done
Local intranet
9:26 PM

Data transmittal across Web

- These forms are sending user entered data
  - Is it an HTTP POST or HTTP GET?
- What do you think the URL is?
  - What machine? What application on the machine?
- How is the form data being sent?
  - Text string? XML?
  - Which do you think would be better? Why?
- Would it be nice if the data were sent the same way, regardless of whether it was user-entered or came from a database on the ‘client’ machine?
  - (The client is the machine sending the data; it may only be a client for the purposes of this transaction. In others it may be a server, receiving data, in a supply chain/engineering collaboration.)
- We next turn to dynamic data handling on the client

Script basics (Java Script, VB Script)

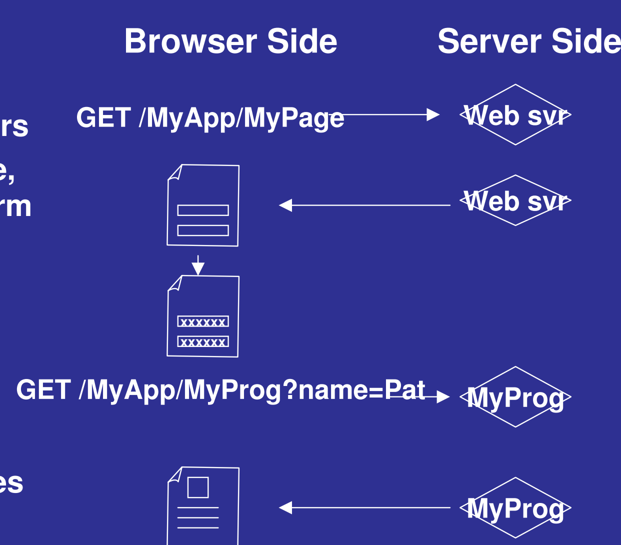
- Script is program run on client (or server) in response to browser request, usually to assist in data validation or data entry
  - Early scripts (Common Gateway Interface, or CGI) run on server only
  - Perl, C are typical languages in the past (and present) for server CGI
  - Modern scripts (JavaScript, VBScript) run on client or server
- Characteristics of <u>client</u> script languages (JavaScript, VBScript)
  - These client scripts are loosely termed ‘Dynamic HTML’ or DHTML
  - Microsoft and Netscape/Java DHTMLs are different and incompatible, though both support JavaScript
    - MS JScript essentially identical to JavaScript; VBScript differs

Motivation for scripts

- If sophisticated programs (Java, etc.) execute on server only, there are some problems:
  – Send and receive html from browser, which is treated as dumb terminal
  – Not very good interactively: long round-trip delay in displaying and getting forms with any internal logic or operations
  – Economic issue (still unsolved): idle clients and expensive, loaded servers. Grid computing is now being explored.
- Solution <u>was</u> thought to be to put small programs on client machine to execute locally
  – Forms with built-in checking, animations, popup window prompts are used to assist in data entry and validation
- It hasn't been a good solution at all. Why?
  – Security issues with loading unknown scripts on browser
  – JavaScript capabilities are limited for data validation because it has no access to the database (it runs on the client)
- Better solutions, covered in next lectures:
    - Validation on the server, against data tables
    - XML, XSLT for transmitting well-formed, valid data

Browser-server interaction

- Step 1: Browser requests page without any parameters
- Step 2: Server fetches page, and returns empty input form
- Step 3: User fills out input form and presses 'submit' button
- Step 4: Browser packages form into query string and sends to server
- Step 5: Program synthesizes a response document and returns it



Every <u>dynamic</u> validation, fill-in requires server intervention.
- If later choices depend on earlier choices, etc.
- JavaScript was felt to be solution, now XML, XSLT are
- Pages can be XML or HTML

JavaScript (or VBScript) on browser

- Can read and write parts of their own documents and other documents the browser has open
- Limited access to browser resources (e.g., history list, back and forward buttons)
- Can read and set contents of fill-out forms, open new windows, create new documents (access to local folders!)
- Library of math and text manipulation functions
- Limited network communications, graphics and window creation
- Must use Java, not JavaScript, if you need to go beyond the bounds of an html document or a simple feature, such as:
  - Animation, simulation, network communications, modules
- JavaScript code can't be kept private; it's in the html sent to the client
  - Can disclose data, business logic, etc. to clients

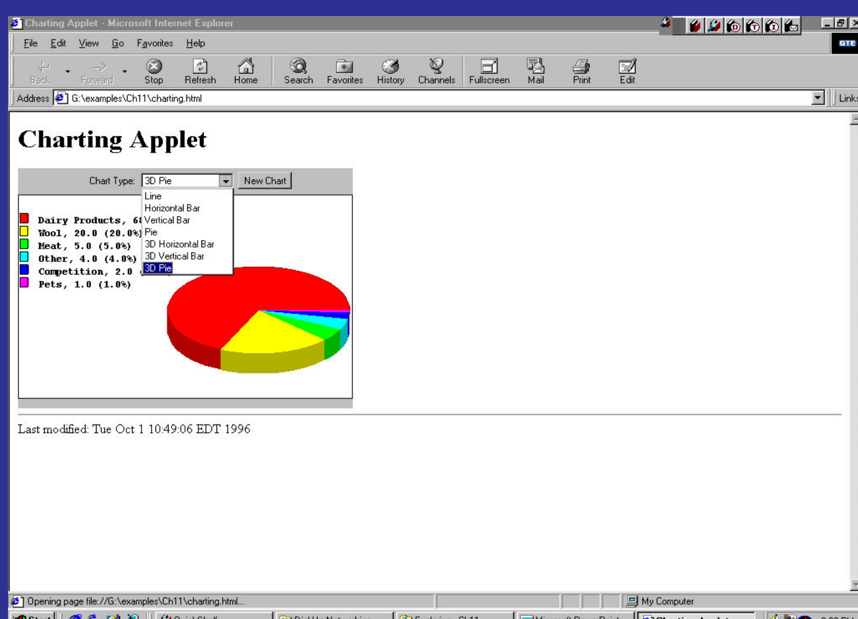
Common JavaScript applications

- Validating fill-out form before submitting it
  - Routine runs to check for required fields and valid formats
  - If not correct, display alert box and refuse to submit form
- Creating pick lists for text fields using popup menus
- Create navigation bars in frames
  - Allow menu in one frame to control display in another frame
- Create 'forward' and 'backward' buttons using history list
- These are simple enough that XML and XSLT can provide these features without downloading 'active' content to the browser from a server. Conclusions:
  - Nothing in supply chain or engineering collaboration or other areas seems to require active content on the client
  - Active content is a large security risk
  - With broadband networks (vs. dialup), turnaround time for server to regenerate Web pages is acceptably low
  - Much Web traffic is machine-machine now, and increasing, so response time is less critical
  - So: We create a limited set of built-in capabilities for what we need without downloading active content: XML and its kin

Java applets (on browser)

- Java-enabled browsers can download and execute Java applets (small Java programs)
- Java applets are much more powerful than JavaScript
  - Open own windows
  - Create buttons
  - Create animations on the fly, 3D model views, calculators, etc.
  - Open network connections to home server, etc.
- Libraries for networks, graphics, multimedia, controls (widgets)
- Platform-independent: UNIX, Windows, Mac, …
- Security features (much better than scripts or ActiveX)
- However, applets are still very limited because they run on the browser:
  - No direct database access, little access to client hardware, less access to browser internal objects than JavaScript (due to security concerns)

Java applet (runs on client, not server)



Typical current Java applet examples

- Page decorations
  - Marquees, banners, etc.
- Animation (set of .gif files that are looped through)
- Multimedia buttons
- Image maps (client side)
- Graphics (charts, graphs)
- Slide shows
- Real applications implemented as applets are rare…
  - JavaScript can now pass parameters to applets
  - Signed applets have more permissions than anonymous applets
- Similar to JavaScript (or VB Script), this has not been very useful yet
  - None of the applet examples above is a compelling feature for supply chain or engineering
  - There is some security risk from applets as active content
  - Again, the important things can be done with XML…

Summary

- HTML forms are used to transmit data from human users to computers
  - POST or GET are used, with data in strings
  - Not validated or structured
- Scripts (JavaScript) are used for dynamic data entry
  - Security risks of active content and data disclosure
  - No access to database on server for sophisticated checks
- Active content (DHTML) hasn't been needed so far
- What we do want is:
  - Data transfer to servers, from other computers or humans in a consistent way
  - Validation and correctness of data
  - Ability to have good security
- XML, rather than active content, is the current solution
  - Uniform, validated format for human and machine data