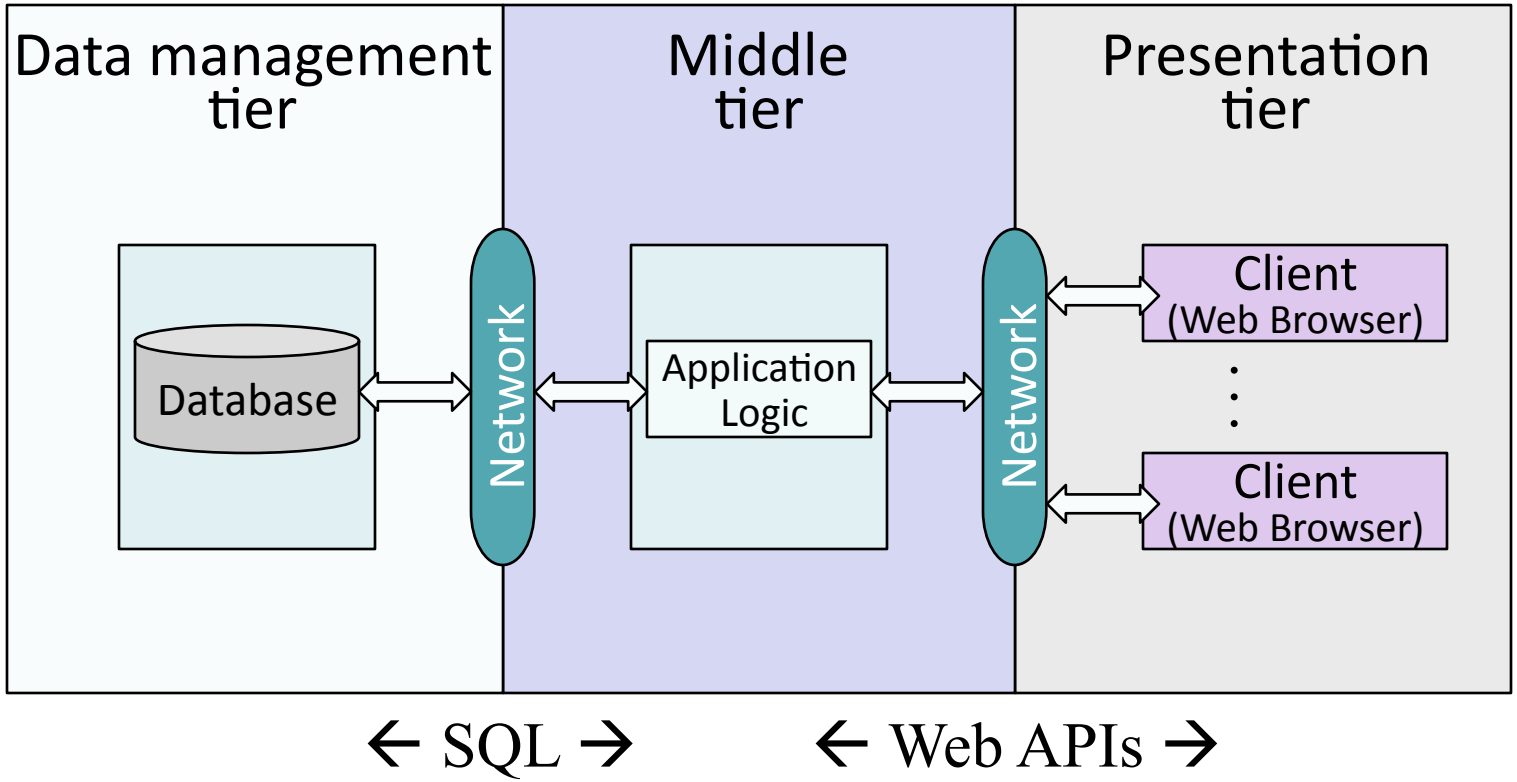
Databases and Internet Applications

Part 2

Chapter 7.5-7.9

Had there been no horses.

Review: Three-Tier Architecture



Lecture Overview

- Internet Concepts
- Web data formats
  - HTML, XML, DTDs
- Introduction to three-tier architectures
- The presentation layer
  - HTML forms; HTTP Get and POST, URL encoding; Javascript; Stylesheets. XSLT
- The middle tier
  - CGI, application servers, Servlets, JavaServerPages, passing arguments, maintaining state (cookies)

Overview of the Presentation Tier

- Recall: Functionality of the presentation tier
  - Primary user interface
  - Needs to adapt to different display devices (PC, PDA, smart phone, voice access?)
  - Simple functionality, such as field validity checking
- We will cover:
  - HTML Forms: How to pass data to the middle tier
  - JavaScript: Programmable functionality at the presentation tier (e.g., form checking, simple animations)
  - Style sheets: Present the same webpage with customized formatting for clients with different capabilities.

HTML Forms

- Common way to *communicate data* from client to middle tier
- General format of a *form*:

```
<FORM  ACTION="page.jsp"  METHOD="GET" NAME="LoginForm">
…
</FORM>
```

- Inside an HTML form,
  - We have form elements that allow the user to enter information in a form
  - We can use any HTML tags

Attributes of Form Tag

```
<FORM  ACTION="page.jsp"  METHOD="GET" NAME="LoginForm">
…
</FORM>
```

- ACTION: Specifies URI of the page to which the form contents are submitted (if absent, URI of current page is used)

  → the page provides logic for processing input from the form.

- METHOD: Specifies HTTP GET or POST method for form submission (GET arguments are part of the URI, POST arguments are part of the request)

- NAME: Name of the form; can be used in client-side scripts to refer to the form (more on this later)

Form Element - Example 1

- INPUT is the most used form tag:

```
<FORM>
First name:
<INPUT TYPE="text" NAME="firstname">
<br>
Last name:
<INPUT TYPE="text" NAME="lastname">
</FORM>
```

- Display in a browser:

First name:

Last name:

20 characters by default

Form Element - Example 2

- Using radio buttons:

```
<FORM>
<INPUT TYPE="radio" NAME="status" VALUE="U" checked />
   Undergraduate<br>
<INPUT TYPE="radio" NAME="status" VALUE="G" />
   Graduate
</FORM>
```

- Display in a browser:

  - ◯ Undergraduate
  - ◯ Graduate

Form Element - Example 3

- Using Submit botton:

```
<FORM ACTION="phone.jsp" METHOD="GET" NAME="Phone">
1 Phone number:
2 <INPUT TYPE="text" NAME="phonenum">
3 <INPUT TYPE="submit" VALUE="Submit">
</FORM>
```

- Display in a browser:

Phone number: [ ] Submit

- When "Submit" button is clicked, the content of the form is sent to the page called phone.jsp on the server
- The page named action needs to be a program, script, or page that will process the user input

INPUT - General Format

<INPUT TYPE="text" NAME="username" VALUE="Joe">

text: a text input field

password: a text field where the entered characters are displayed as stars

reset: a button that resets all input fields

submit: a button that sends the values in the form to the server

- Specifies the symbolic name for this element
- Used to identify the associated input when sent to the middle tier

- Specifies default value for text or password fields
- For submit or reset buttons, it sets the label of the button

Three Types of User Input

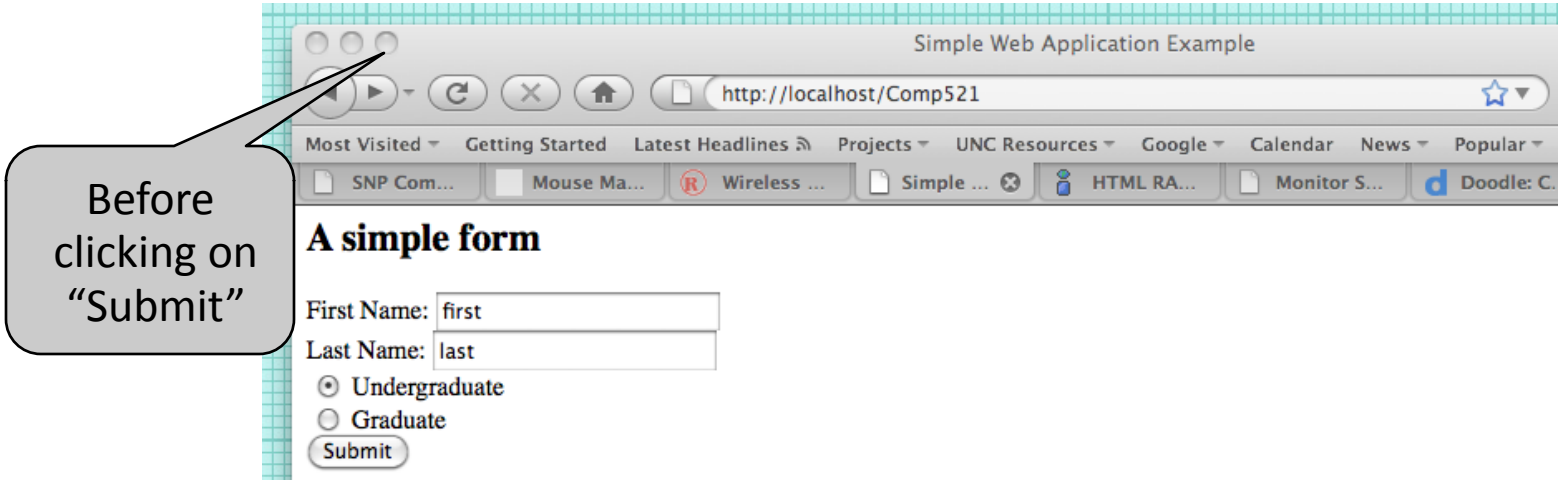
- INPUT tag
  - *button*, *checkbox*, *file*, *hidden*, *image*, in addition to password, radio, reset, submit, text
- TEXTAREA tag
  - A multi-line box for text entry with ROWS, and COLS attributes to set the size
- SELECT tag
  - Choose one of multiple choices via a drop-down list or some other GUI element
  - Block enclosing OPTION tags; returns “value” of selected one

Passing Arguments

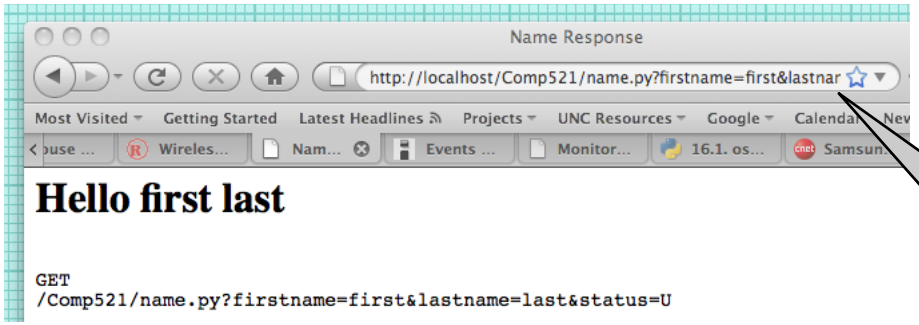
Two methods for submitting HTML Form data to the Web server:

- GET:
  - The contents of the form are assembled into a query URI (i.e., Middle tier receives header that actually contains data from the form)
  - The form contents are directly visible to the user as the constructed URI
  - The users can bookmark the page with the constructed URI
- POST:
  - The contents of the form are encoded as in the GET method, but the they are sent in a separate data block
  - The form contents are sent inside the *HTTP request message body* and are not visible to the user

GET

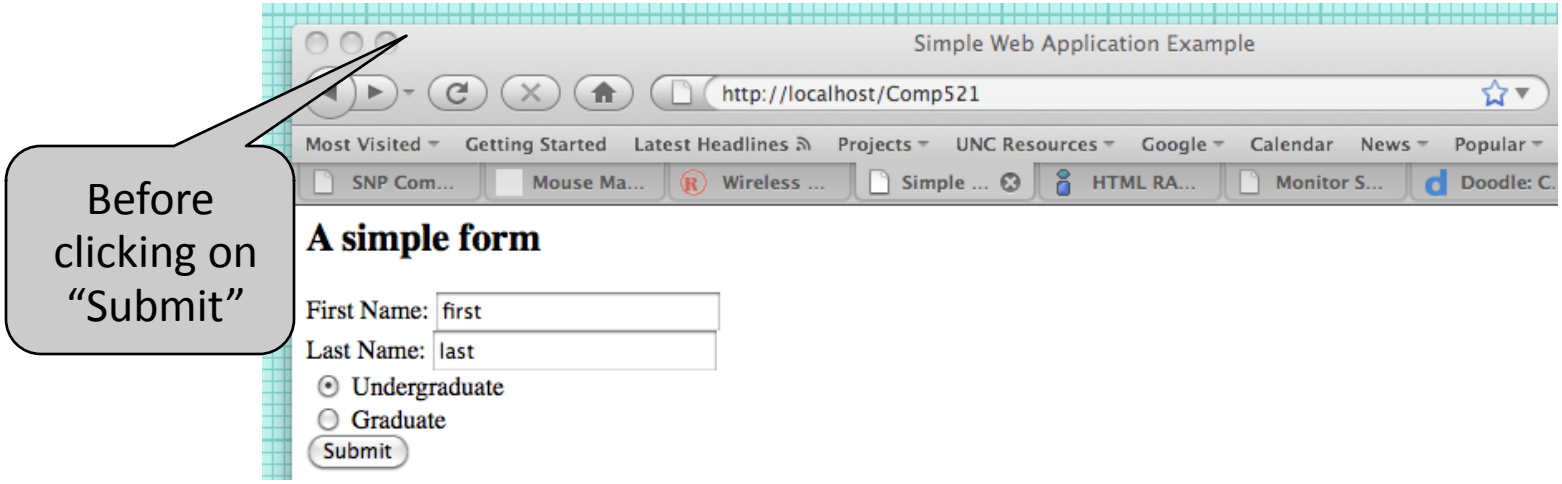


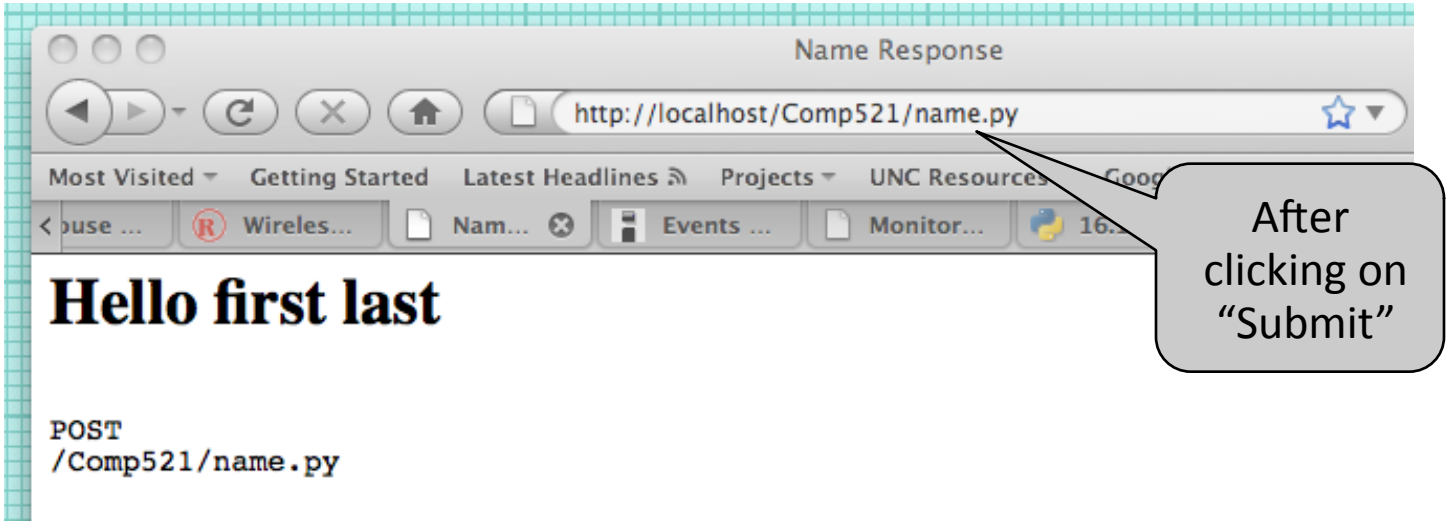
Before clicking on "Submit"



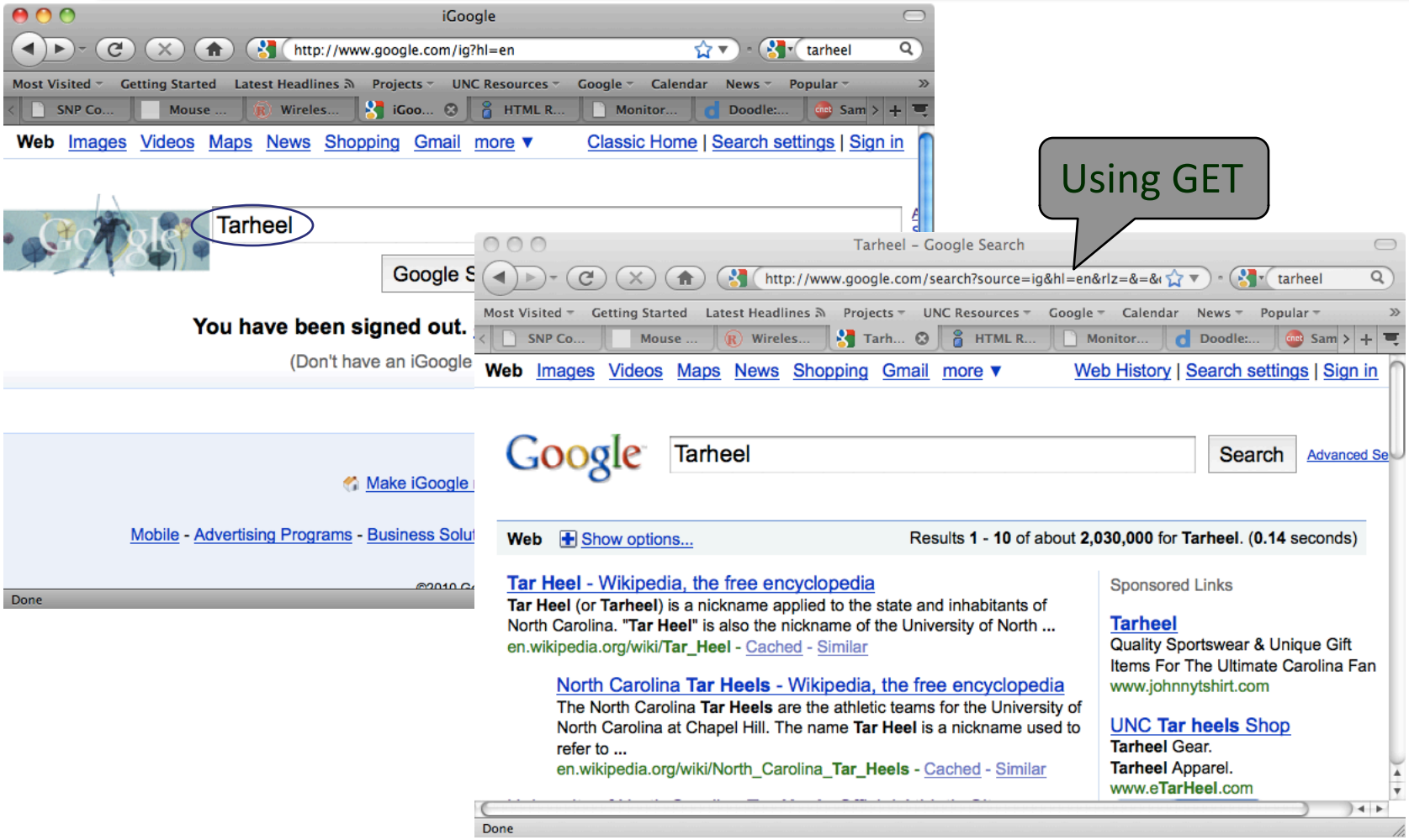
After clicking on "Submit"

POST


Simple Web Application Example
http://localhost/Comp521
Most Visited
Getting Started
Latest Headlines
Projects
UNC Resources
Google
Calendar
News
Popular
SNP Com...
Mouse Ma...
Wireless ...
Simple ...
HTML RA...
Monitor S...
Doodle: C.
A simple form
First Name: first
Last Name: last
Undergraduate
Graduate
Submit
Before clicking on "Submit"


Name Response
http://localhost/Comp521/name.py
Most Visited
Getting Started
Latest Headlines
Projects
UNC Resources
Wireles...
Nam...
Events ...
Monitor...
Hello first last
POST
/Comp521/name.py
After clicking on "Submit"

Example: Google Search



Encoded URI

Only shown for GET method

action?name1=value1&name2=value2&name3=value3

The action is the URI specified in the ACTION attribute of the FORM tag

'name=value' pairs are the user inputs from the INPUT fields in the form

Example:

page.py?username=John+Doe&password=secret

'+' represents a space character

HTTP GET: Encoding Form Fields

- Form fields can contain general ASCII characters that cannot appear in an URI (e.g., space character)

Name: Jane Doe

Cannot have a space in an URI

- A special encoding convention converts such field values into "URI-compatible" characters:

HTTP GET: Encoding Form Fields

- Overview of the encoding convention :
  1. Converts all spaces to the "+" character
  2. Glue (name,value)-pairs from the form INPUT tags together with "&" to form the URI

page.py?coursename=Database+Systems&CourseID=Comp521

First INPUT

Second INPUT

  3. Convert all "special" characters to %xyz, were xyz is the ASCII code of the character. Special characters include &, =, +, %, etc.

Clientside Scripting

- JavaScript is an interpreted scripting language
  - A JavaScript is a program added to a Web page.
  - It runs at the client tier to add functionality to the presentation
- Sample applications:
  - Browser Detection: Detect browser type and load browser-specific page.
  - Form validation: Validate form input fields (e.g., an email address contains '@'), or if all required fields have input data.
  - Browser control: Open new windows (e.g., pop-up ads)

JavaScript: SCRIPT Tags

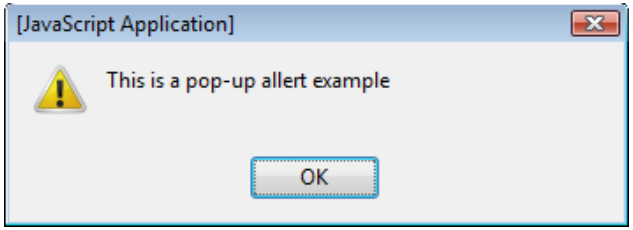
- A JavaScript is usually embedded inside the HTML with the <SCRIPT> … </SCRIPT> tag.
- <SCRIPT> tag has several attributes:
  - LANGUAGE: indicates language of the script (such as JavaScript)
  - SRC: Specifies the external file with the script code
- <u>Example</u>:

```
<SCRIPT LANGUAGE="JavaScript" SRC="validate.js">
  </SCRIPT>
```

JavaScript without SRC Attribute

- If <SCRIPT> tag does not have an SRC attribute, then the JavaScript code is enclosed within the tag of the HTML file.
- Example: Create a pop-up box with a message

```
<SCRIPT LANGUAGE="JavaScript">
<!-- alert("This is a pop-up alert example")
//-->
</SCRIPT>
```



- Two different commenting styles
  - <!-- marks the start of an HTML comment
  - // marks the start of a JavaScript comment
  - <!-- indicates the following JavaScript code should be ignored by the HTML processor
  - // comment is used to comment out the HTML "-- >" comment as it is interpreted otherwise

Two Different Commenting Styles

- "//..." is used for single-line comments
- "/* ... */" is used for multi-line comments

JavaScript Basics

- Variables:
  - numbers, boolean values, strings, …
  - Variables do not have a fixed type, but implicitly have the type of the data to which they have been assigned
- Assignments: =, +=, …
- Comparison operators: <,>,…
- Boolean operators: && for logical AND, || for logical OR, ! for negation
- Statements
  - if (condition) {statements;} else {statements;}
  - Loops: for-loop, do-while, and while-loop
- Functions with return values

  function funcname(arg1, …, argk) {statements;}

A Complete Example

```
<script language="JavaScript">
<!--
function strip(strval) {
   while (strval.charAt(0) == ' ') {
      strval = strval.substr(1);
   }
   while ((strval.length > 0) && (strval.charAt(strval.length - 1) == ' ')) {
      strval = strval.substr(0, strval.length - 1);
   }
   return strval;
}

function validate() {
   var first = strip(document.NameForm.firstname.value);
   var last = strip(document.NameForm.lastname.value);
   if ((first == "") || (last == "")) {
      alert('A First and Last name are required');
      return false;
    } else {
      document.NameForm.firstname.value = first
      document.NameForm.lastname.value = last
      return true;
    }
}
//-->
</script>
```

"document" is an implicitly defined variable referring to the current HTML page

Current HTML Page

```
function validate()
  …
  … = document.NameForm
  …
```

This slide

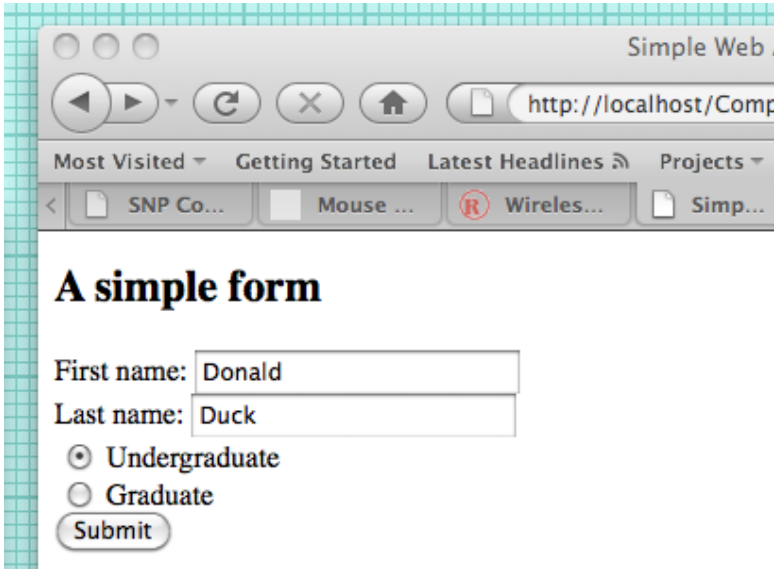
```
<FORM NAME="NameForm"
 …
 …
</FORM>
```

Next slide

A Complete Example – Cont'd

```
<h2>A simple form</h2>
<form action="name.py"
    onsubmit="return validate()"
    method="get"
    name="NameForm">
First name: <input type="text" name="firstname"><br>
Last name: <input type="text" name="lastname"><br>
<input type="radio"
    name=status value="U" checked>
    Undergraduate<br>
<input type="radio"
    name=status value="G">
    Graduate<br>
<input type="submit"
    value="Submit">
</form>
```

The form contents are submitted to server if function returns true



Event Handlers

```
<FORM NAME = "LoginForm" METHOD="POST"
ACTION="TableOfContents.jsp"
onSubmit="return testLoginEmpty()">
```

- An event handler is a function that is called if an event happens on an object in a webpage
- onSubmit() is an event handler, which is called if the submit button is pressed
- If the event handler returns true, then the form contents are submitted to the server
- Others OnFocus(), OnBlur(), onKeyDown(), etc..

Style Sheets

- We need different ways of displaying the same information to clients with different displays
  - Using different font sizes or colors to provide better contrast on a black-and-white screen
  - Rearranging objects on the page to accommodate small screens
  - Highlighting different information to focus on some important part of the page
- A style sheet is a method to adapt the same document contents to different presentation formats
  - It tells a Web browser how to translate the data into a presentation that is suitable for the client's display

Style Sheets

- Idea: Separate display from contents, and adapt display to different presentation formats
- Two aspects:
  - Document transformations to decide what parts of the document to display and in what order
  - Document rendering to decide how each part of the document is displayed
- Two stylesheet languages
  - Cascading style sheets (CSS): For HTML documents
  - Extensible stylesheet language (XSL): For XML documents

CSS: Cascading Style Sheets

- Styles are normally stored in style sheets, which are files that contain style definitions. They define how to display HTML documents.
- Many HTML documents (e.g., all in a website) can refer to the same CSS
  - Can change format of a website by changing a single file (i.e., separation of content from presentation)
- Example: Include the following line into an HTML file to link to the external CSS style sheet.

```
<LINK REL="style sheet" TYPE="text/css" HREF="books.css"/>
```

Relationship between current document and linked document

The type of the linked document

Location of the linked document

CSS: Example

<LINK REL="style sheet" TYPE="text/css" HREF="books.css"/>

The books.css file:

```
BODY {BACKGROUND-COLOR: yellow}
H1 {FONT-SIZE: 36pt}
 {COLOR: blue}
MARGIN-LEFT: 50px;  COLOR: red}
```

Same effect as:
<BODY BACKGROUND -COLOR="YELLOW">

- Each line consists of three parts: selector {property: value}
  - Selector: Tag whose format is defined
  - Property: Tag's attribute whose value is set
  - Value: value of the attribute
- Multiple properties for the same selector are separated by semicolons

XSL

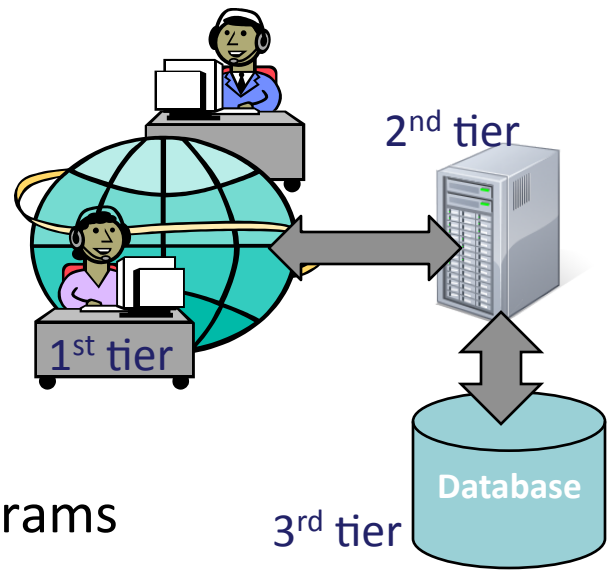
- Language for expressing style sheets
  - More at: http://www.w3.org/Style/XSL/
- Three components
  - XSLT: XSL Transformation language
    - Can transform one document to another
    - More at http://www.w3.org/TR/xslt
  - XPath: XML Path Language
    - Selects parts of an XML document
    - More at http://www.w3.org/TR/xpath
  - XSL Formatting Objects
    - Formats the output of an XSL transformation
    - More at http://www.w3.org/TR/xsl/

Lecture Overview

- Internet Concepts
- Web data formats
  - HTML, XML, DTDs
- Introduction to three-tier architectures
- The presentation layer
  - HTML forms; HTTP Get and POST, URL encoding; Javascript; Stylesheets. XSLT
- The middle tier
  - CGI, application servers, Servlets, JavaServerPages, passing arguments, maintaining state (cookies)

Review of the Middle Tier

- Recall: Functionality of the middle tier
  - Encodes business logic
  - Connects to database system(s)
  - Accepts input from the presentation tier
  - Generates output for the presentation tier

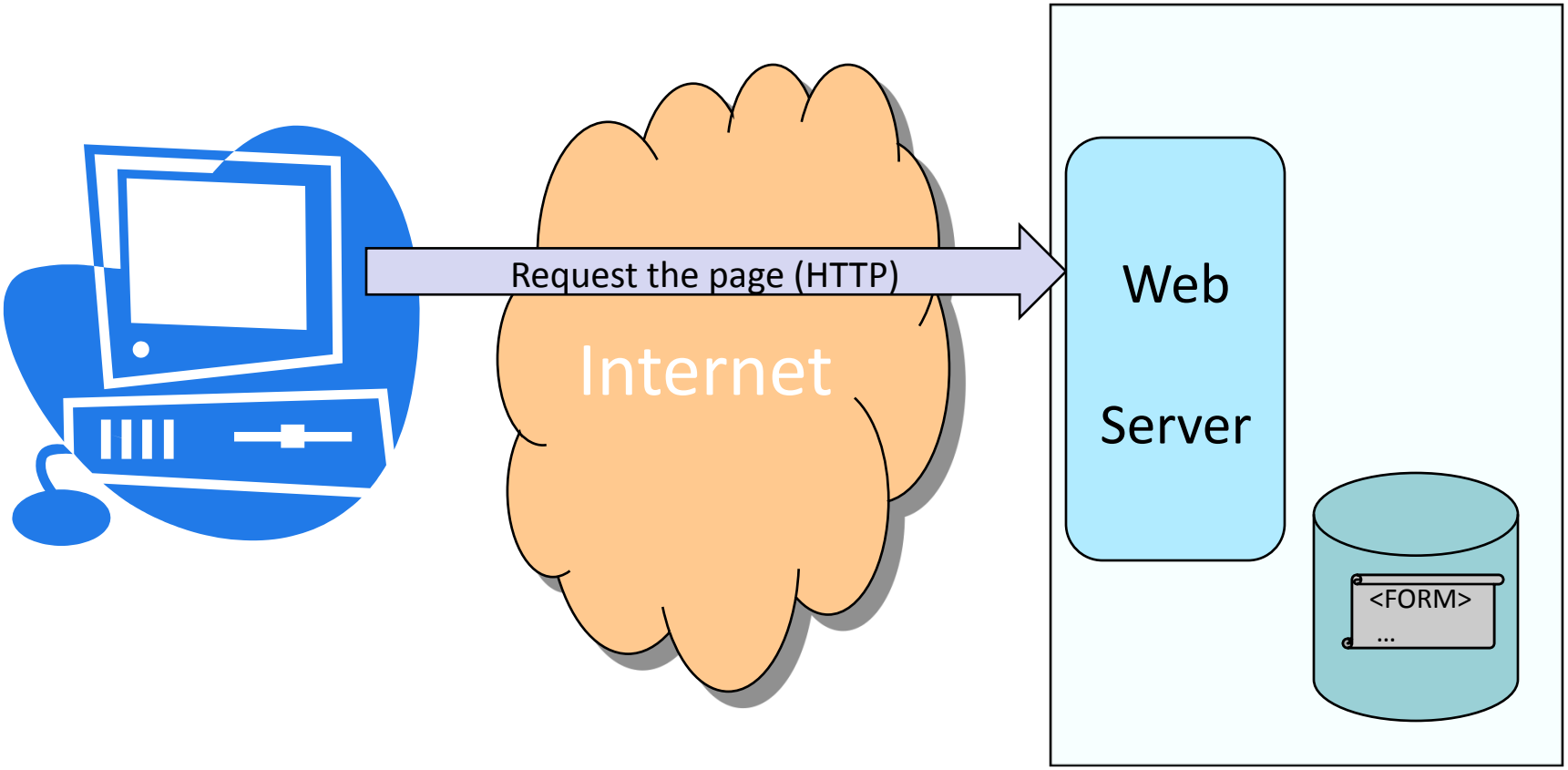


- We will cover
  - **CGI**: Protocol for passing arguments to programs running at the middle tier
  - **Application servers**: Runtime environment at the middle tier
  - **Servlets**: Java programs at the middle tier
  - **JavaServerPages**: Java scripts at the middle tier
  - Maintaining state: How to maintain state at the middle tier. Main focus: **Cookies**.

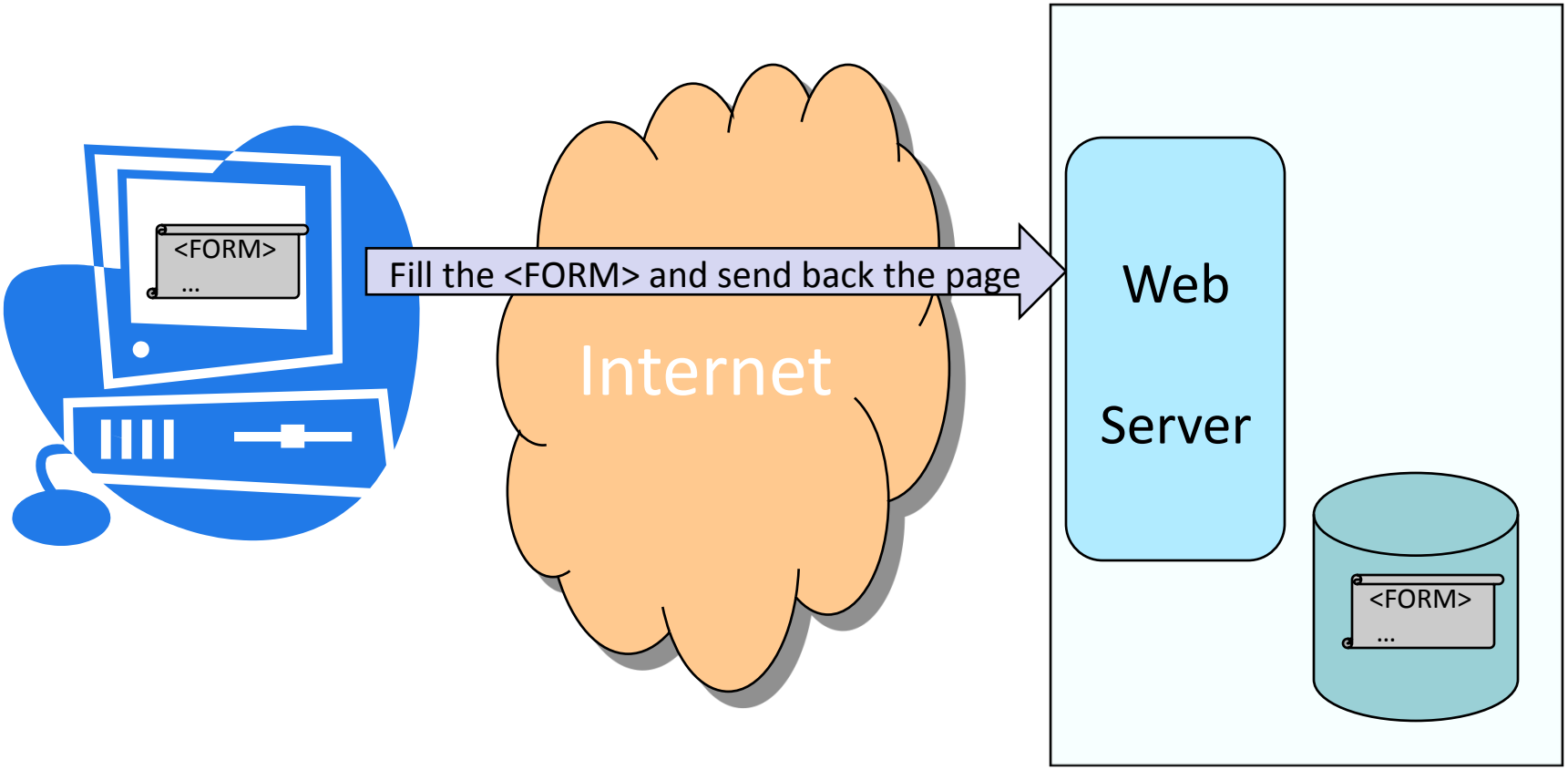
CGI: Common Gateway Interface

- Goal: Transmit arguments from HTML forms to application programs running at the middle tier
- Details of the actual CGI protocol unimportant → libraries implement high-level interfaces (enable application programs to get arguments from the HTML form)
- Disadvantages:
  - The application program is invoked as a new process at every invocation (remedy: FastCGI)
  - No resource sharing between application programs (e.g., database connections)
  - Remedy: Application servers

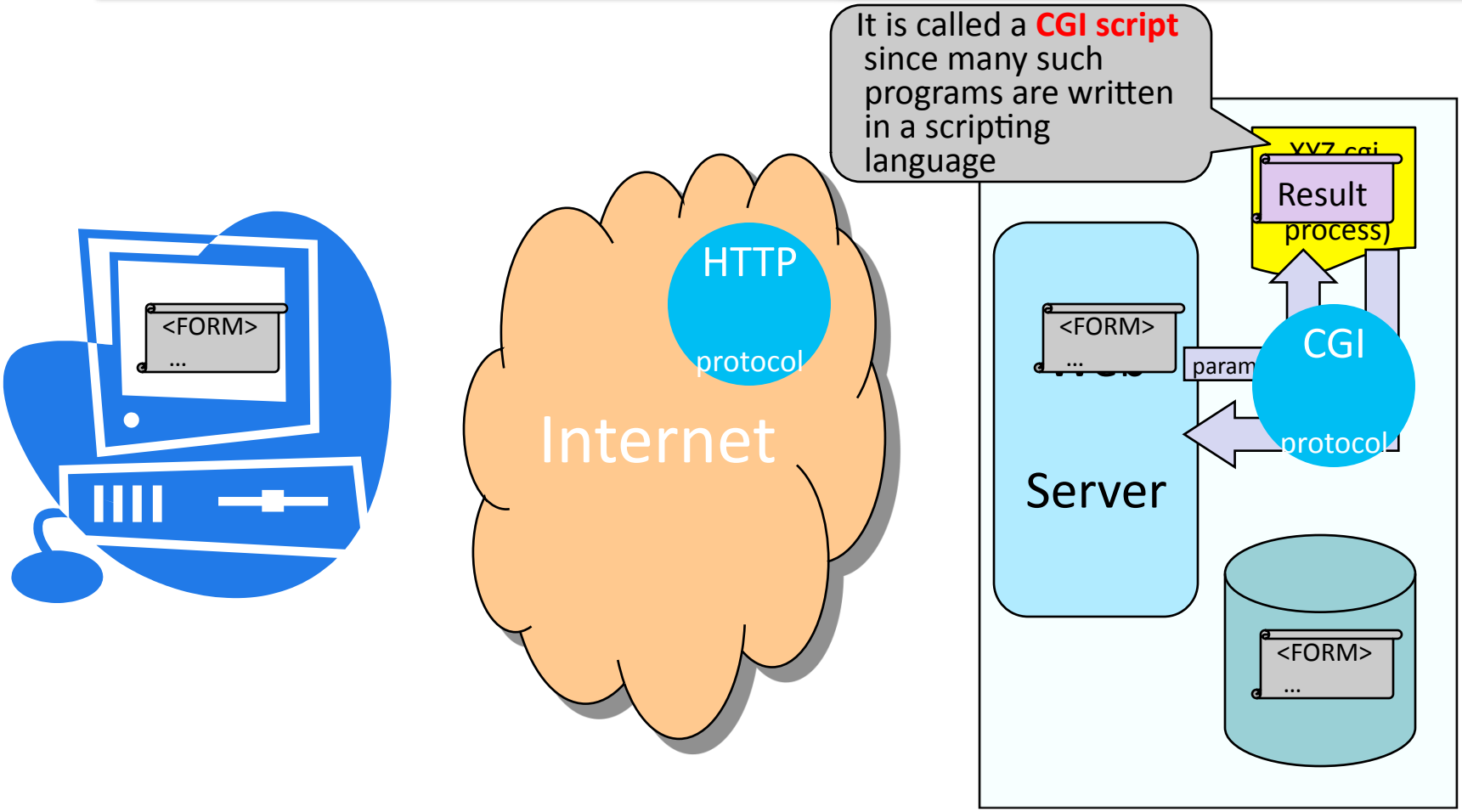
CGI Illustration: Request the Page



CGI Illustration: Send Back the page



CGI Illustration: Process <FORM>



Review: HTTP Responses

The HTTP response message has three parts:

- status line, e.g., "HTTP/1.1 200 OK" (The request succeeded and the object is in the body of the message.)
- several header lines

  Date: Mon, 04 Mar 2002 12:00:00 GMT

  Server: Apache/1.3.0 (Linux)

  Last-Modified: Mon, 01 Mar 2002 09:23:24 GMT

  Content-Length: 1024

  Content-Type: text/html
- body of the message (which contains the requested object)

CGI: Example

- Python can be used for CGI scripting
- Provides libraries supporting high-level interfaces to the CGI protocol

Example Python code:

```
#!/usr/bin/python
import cgi
import cgitb    # provides nice debugging info
cgitb.enable()
import os
pageStart = """
<html><head><title>Name Response</title></head>
<body><h1>Hello %s %s</h1><pre>"""
pageEnd = """</pre></body></html>"""

if __name__ == '__main__':
    form = cgi.FieldStorage()
    first = form["firstname"].value
    last = form["lastname"].value
    print "Content-type: text/html\n\n"
    print pageStart % (first, last)
    print os.environ['REQUEST_METHOD']
    print os.environ['REQUEST_URI']
    print pageEnd
```

Include the standard library CGI

Arguments from CGI

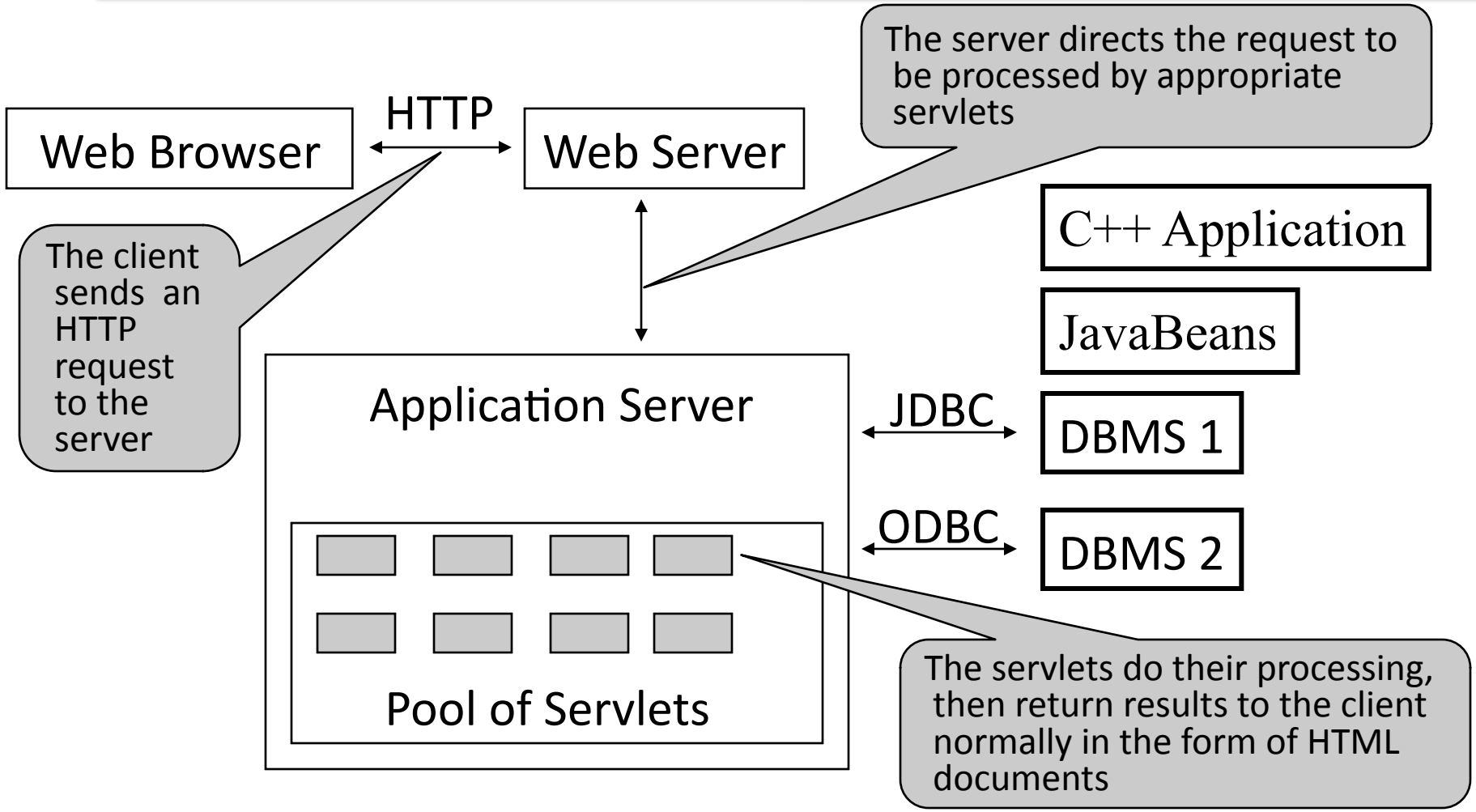
Extract arguments from the HTML form

Dynamically construct the Webpage

Application Servers

- The application program is invoked using the CGI protocol. Each page request results in the creation of a new process → Do not scale well to a large number of simultaneous requests.
- Avoid the process creation overhead:
  - Application server maintains a pool of threads or processes and uses them to execute requests
  - Application server also provides other functionality
    - Enable access to heterogeneous data sources (e.g., by providing JDBC drivers.)
    - Provide APIs for session management

Application Server: Process Structure



What is a Servlet ?

- Servlets are Java's answer to CGI programming
- Servlet is a Java class used to extend the capabilities of servers that host applications
- In most cases, servlets extend the specific HttpServlet class for Web servers that communicate with clients via HTTP
- HttpServlet class provides methods such as
  - doGet (for HTTP GET) and doPost (for HTTP POST) to receive arguments from HTML forms, and
  - sending output back to the client via HTTP

A Servlet Template

This simple servlet just outputs two words "Hello World"

```
import java.io.*;
import java.servlet.*;
import java.servlet.http.*;

public class ServetTemplate extends HttpServlet {
    public void doGet(HTTPServletRequest  request,
                                HTTPServletResponse  response)
    throws SerletExpection, IOException {
          PrintWriter out=response.getWriter();
       out.println("Hello World");     // sends content to browser
    }
}
```

request object is used to read HTML form data

response object is used to specify: (1) the response status code, and (2) headers of the HTTP response

Use 'out' to compose content that is returned to the client

Servlets: A Complete Example

```
public class ReadUserName extends HttpServlet {
    public void doGet(    HttpServletRequest request,
                          HttpSevletResponse response)
        throws ServletException, IOException {
        reponse.setContentType("text/html");
        PrintWriter out=response.getWriter();
        out.println("<HTML><BODY>\n <UL> \n" +
                "<LI>" + request.getParameter("userid") + "\n" +
                "<LI>" + request.getParameter("password") + "\n" +
                "<UL>\n<BODY></HTML>");
    }
    public void doPost(   HttpServletRequest request,
                          HttpSevletResponse response)
        throws ServletException, IOException {
        doGet(request,response);
    }
}
```

We extend the servlet to dynamically generate HTML

Servlet Life Cycle

- Servlet container is the intermediary between the Web server and the servlets in the container
- When a request arrives from the Web server:
  - If an instance of the servlet does not exist, the container
    - Loads the servlet class
    - Creates an instance of the servlet class
    - Initializes the servlet instance (i.e., places it into service)
  - Container calls service() method to allow the servlet to respond to the request. Two objects are passed:
    - The HttpServletRequest object contains the client's HTTP request information, and
    - The HttpServletResponse encapsulates the servlet's response
- Servlet container calls the destroy method before removing a servlet from service (e.g., to free memory)

Java Server Pages (JSP)

- Servlets
  - Generate HTML by writing it to the "PrintWriter" object
  - Code first, webpage second
- JavaServer Pages
  - Written in HTML, Servlet-like code embedded in the HTML
  - Webpage first, code second
  - They are usually compiled into a Servlet

JavaServer Pages

- Change the file extensions to ".jsp" instead of ".html"
- Embed Java expressions in JSP pages by putting them between <%= and %>

  The time is now <%= date %>
- Embed block of Java code (called scriplet) between <% and %>
- There are a number of useful predefined objects for scriplet:
  - The variable out can be used to generate HTML

    <% java.util.Date date = new java.util.Date(); %>
    The time is now
    <% out.println(String.valueOf(date)); %>
  - request is another useful variable

    request.getParameter("username"): returns value of the requested parameter

JavaServer Pages: Example

```
<html>
<head><title>Welcome to B&N</title></head>
<body>
   <h1>Welcome back!</h1>
   <% String name="NewUser";
      if (request.getParameter("username") != null) {
            name=request.getParameter("username");
      }
   %>
   You are logged on as user <%=name%>
   <p>
</body>
</html>
```

- Placing java code between <% and %>
- The block of code is known as scriptlet
- This makes it possible to generate dynamic HTML pages

Maintaining State

HTTP is stateless

- Advantages
  - Easy to use: don't need anything
  - Great for static-information applications
  - Requires no extra memory space
- Disadvantages
  - No record of previous requests means
    - No shopping baskets
    - No user logins
    - No custom or dynamic content
    - Security is more difficult to implement

Application State

- Server-side state
  - Information is stored in a database, or in the application layer's local memory
- Client-side state
  - Information is stored on the client's computer in the form of a cookie
- Hidden state
  - Information is hidden within dynamically created web pages

Server-Side State

Many types of Server side state:

1. Store information in a database
   - Data will be safe in the database
   - BUT: requires a database access to query or update the information
2. Use application layer's local memory
   - Can map the user's IP address to some state
   - BUT: this information is volatile and takes up lots of server main memory

Server-Side State

- Should use Server-side state maintenance for information that needs to persist
  - Old customer orders
  - "Click trails" of a user's movement through a site
  - Permanent choices a user makes

Client-side State: Cookies

- Cookies are textual information a Web server sends to a browser, and that browser returns unchanged when sending HTTP requests to the Web server later
  - Can be disabled by the client.
  - Are wrongfully perceived as "dangerous," and therefore will scare away potential site visitors if asked to enable cookies
    - Cookies are never interpreted or executed → cannot be used to insert virus
    - Browser generally accept 20 cookies per site and 300 cookies total → cannot be used to fill up someone's disk or launch other denial of service attacks
- A cookie is a collection of (Name, Value) pairs

Client State: Cookies

- Advantages
  - Easy to use in Java Servlets / JSP
  - Provide a simple way to persist non-essential data on the client (in the browser cache) even after the browser is closed
- Disadvantages
  - Limit of 4 KB of information (not bad for most applications)
  - Users can (and often will) disable them
- Should use cookies to store interactive state
  - The current user's login information
  - The current shopping basket
  - Any non-permanent choices the user has made

Creating A Cookie

```
Cookie myCookie =
  new Cookie("username", "jeffd");
response.addCookie(myCookie);
```

Create a new cookie object with the specified (name, value) pair

Add the cookie to the response header, i.e., send cookie to client

You can create a cookie at any time

Cookie – Another Example

```
Cookie myCookie = new Cookie("username", "jeffd");
Cookie.setDomain(www.bookstore.com);
                                    // Web site that receives this cookie
Cookie.setSecure(false):            // no SSL required
Cookie.setMaxAge(60*60*24*31)       // one month lifetime
response.addCookie(myCookie);       // add cookie to
                                       response object
```

Cookie - How it works

- We create a cookie through the Java or Python Cookie class in the middle tier application code
- The cookie is added to the **response** object within the java servlet to be sent to the client
- Once a cookie is received, the client's Web browser appends it to all HTTP requests it sends to this site, until the cookie expires

Reading Cookies from the Client

Look for the cookie with name 'username'

```
Cookie[]   cookies = request.getCookies();    // returns an array of cookies
String   theUser;
for(int i=0; i<cookies.length; i++) {
  Cookie   cookie = cookies[i];
  if(cookie.getName().equals("username")) theUser = cookie.getValue();
}
// at this point theUser == "username"
```

Hidden State

- Often users will disable cookies
- You can "hide" data in two places:
  - Hidden fields within a form
  - Using the path information
- Requires no "storage" of information because the state information is passed inside of each web page

Hidden State: Hidden Fields

- Declare hidden fields within a form:
  - <input type='hidden' name='user' value='username'/>
- Users will not see this information (unless they view the HTML source)
- Typically used when we have variables we want to pass from one form to another without making the user to re-type the information over and over again
  - We can use this feature to maintain state, e.g., remember the user in order to update the shopping cart

Hidden State:

Using Extra Path Information

- The middle tier can embed an identifier, as extra path information, within a document's URL
- As a user traverse through the site, the dynamically generated html pages can pass the identifier from document to document
- Thus, we can track the documents requested by the user

Hidden State:

Using Extra Path Information

- Path information is stored in the URL request:

  http://server.com/index.htm?user=jeffd

- Can separate 'fields' with an & character:

  index.htm?user=jeffd&preference=pepsi

- There are mechanisms to parse this field in Java. Check out the

  javax.servlet.http.HttpUtils parserQueryString

  method.

Multiple state methods

Typically all methods of state maintenance are used:

- User logs in and this information is stored in a cookie
- User issues a query which is stored in the path information
- User places an item in a shopping basket cookie
- User purchases items and credit-card information is stored/retrieved from a database
- User leaves a click-stream which is kept in a log on the web server (which can later be analyzed)

Summary

We covered:

- Internet Concepts (URIs, HTTP)
- Web data formats
  - HTML, XML, DTDs
- Three-tier architectures
- The presentation layer
  - HTML forms, Javascript, Stylesheets.
- The middle tier
  - CGI, application servers, Servlets, JavaServer Pages, maintaining state (cookies)