JSP Lecture 0

HTML

Original Slides from MIT AITI 2004

Markup Language

- **H**yper-**T**ext **M**arkup **L**anguage
- Markup Languages are not compiled
- ML is processed by the client (eg. web browser)

- Text is processed using *tags* and *attributes*

Basic HTML Example

```
<html>
<head>
  <title>Title Bar</title>
</head>
<body>
<h1>Header</h1>
Regular text.
<br>
More Text
<hr>
Even More Text
</body>
</html>
```

<html> Tag

```
<html>
<head>
  <title>TitleBar</title>
</head>
<body>
<h1>Header</h1>
Regular text.
<br>
More Text
<hr>
Even More Text
</body>
</html>
```

- <html> - signifies the start of an HTML document, should always be the first and last tag on the page

Basic HTML Tags

```
<html>
<head>
  <title>TitleBar</title>
</head>
<body>
<h1>Header</h1>
Regular text.
<br>
More Text
<hr>
Even More Text
</body>
</html>
```

- <head> - marks the section of the page that will contain basic header information
- <title> - text will be shown at the top of the window bar
- <body> - text in this area will be displayed inside the main part of the browser window

Basic HTML Tags II

```
<html>
<head>
  <title>TitleBar</title>
</head>
<body>
<h1>Header</h1>
Regular text.
<br>
More Text
<hr>
Even More Text
</body>
</html>
```

- <h1> - <h4> - header tags which make the text larger and bold; there is an automatic <br> after this
- <br> - no end tag; same as a carriage return (ENTER)
- <hr> - no end tag; puts a horizontal rule (line) on the page

Attributes

- HTML tags can have properties
- Properties are defined by attributes
- Each tag may have one or more attributes.
- Attributes give greater power to tags by expanding their capabilities

```
<html>
<head>
  <title>TitleBar</title>
</head>
<body bgcolor="green">
Regular text.
<a href =
"http://www.yahoo.com">
 This is a link.</a>
<font face="Arial">Text in
Arial font</font>
</body>
</html>
```

Basic Tags & Attributes

```
<html>
<head>
  <title>TitleBar</title>
</head>
<body bgcolor="green">
Regular text.
<a href =
  "http://www.yahoo.com">
 This is a link.</a>
<font face="Arial">Text in
  Arial font</font>
</body>
</html>
```

- <a> - anchor tag; used for links; main attribute is "href" which defines the location of where the link will go
- <font> - font tag; used to define a particular font or style of font to display on the page; attributes used most often: "face", "color", "size"

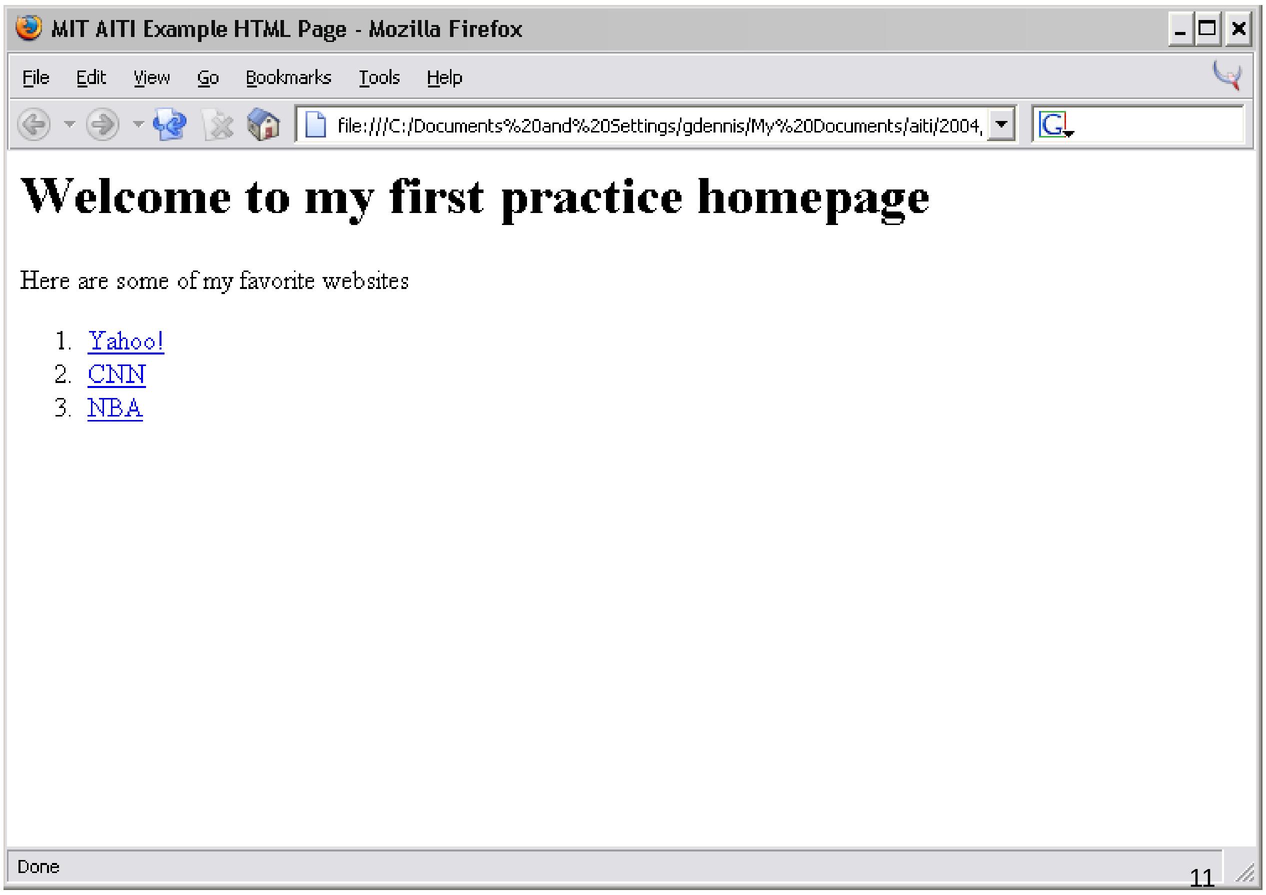
More Basic Tags

- <i> - italics
- <b> - bold
- <u> - underline
- <img> - image tag; used to place photos, images or graphics within a page; attributes used are "src" and "border"
- <p> - paragraph tag; used to separate paragraphs by a break
- <ul> - unordered list tag; signifies the start of an unordered list of items
- <ol> - same as the unordered list tag, but items are numbered (ordered)
- <li> - used within the <ul> or <ol> tags, this signifies a list item

HTML Example I

```
<html>
<head>
  <title>MIT AITI Example HTML Page</title>
</head>

<body bgcolor="white">
<h1>Welcome to my first practice homepage</h1>
Here are some of my favorite websites
  <ol>
  <li><a
  href="http://www.yahoo.com">Yahoo!</a></li>
  <li><a href="http://www.cnn.com">CNN</a></li>
  <li><a href="http://www.nba.com">NBA</a></li>
  </ol>
</body>
</html>
```


MIT AITI Example HTML Page - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

file:///C:/Documents%20and%20Settings/gdennis/My%20Documents/aiti/2004,

Welcome to my first practice homepage

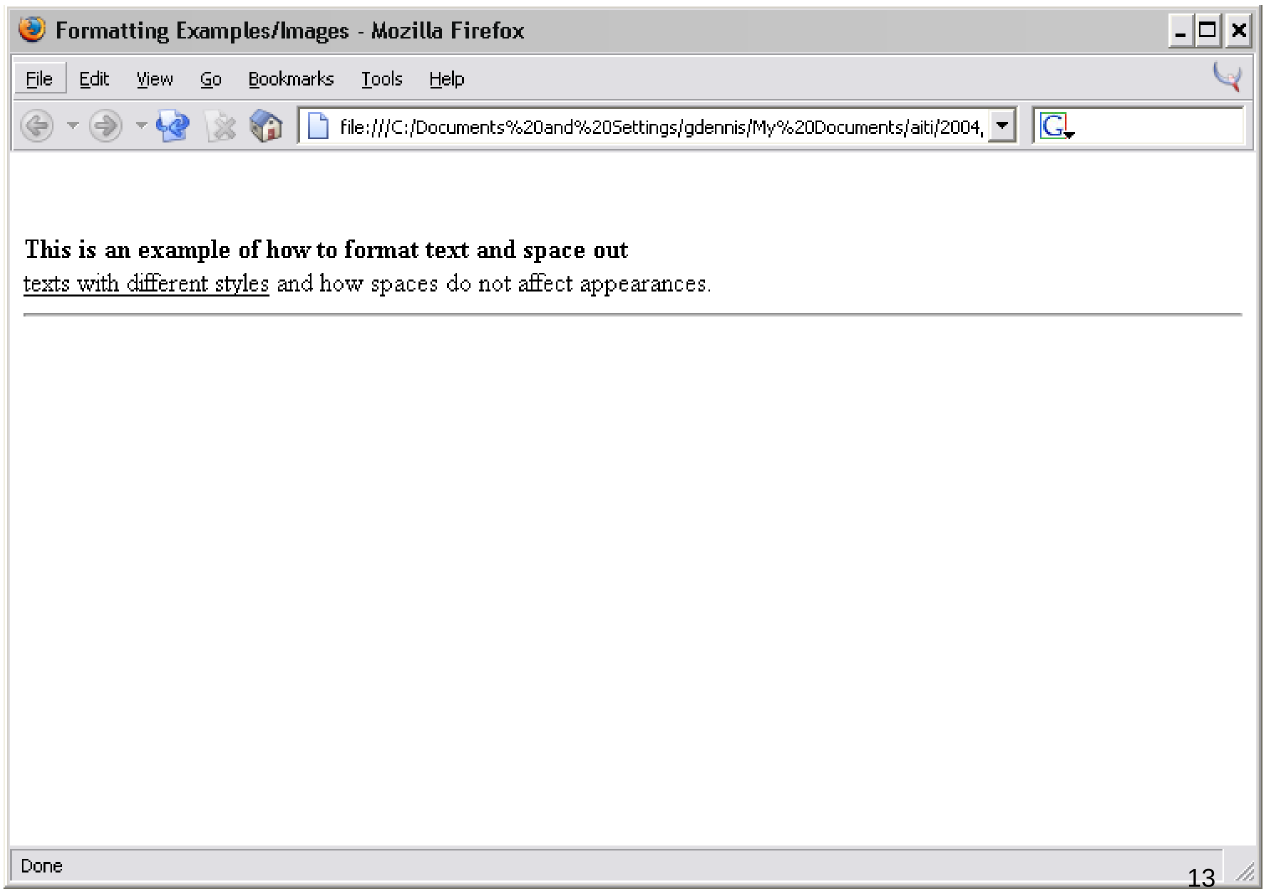
Here are some of my favorite websites

1. Yahoo!
2. CNN
3. NBA

Done

HTML Example II

```
<html>
<head>
  <title>Formatting Examples/Images</title>
</head>
<body bgcolor="white">
<br><br>
<b>This is an example of how to
  format text and space out</b>
<br>
<u>texts with different styles</u> and how
  spaces do not affect appearances.
<hr>
</body>
</html>
```


Formatting Examples/Images - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
file:///C:/Documents%20and%20Settings/gdennis/My%20Documents/aiti/2004,
Done

This is an example of how to format text and space out
texts with different styles and how spaces do not affect appearances.

HTML Tables

- Tables provide a way to format the way information is displayed on pages
- Tables are just a series of tags which define rows and columns, as well as properties of the table through attributes
- Tables are important since they can change the layout of a webpage

Table Tags

- <table> - basic table tag; signifies the start and end of a table
- <tr> - table row tag; signifies the start of a row; <tr> tags are always found within <table> tags; in HTML, rows are always defined before columns
- <td> - table down tags; signifies start of columns; <td> tags are always found within <tr> tags

Table Example 1

```
<html>
<head><title>Table Example 1</title></head>
<body bgcolor="ffffff">
<h2>Here's a very simple table (2 x 2)</h2>
<table border=1 cellpadding=3 cellspacing=2
  width=300>
  <tr>
      <td width=150>First name:</td>
      <td width=150>Greg</td>
  </tr>
  <tr>
      <td width=150>Last name:</td>
      <td width=150>Dennis</td>
  </tr>
</table>
</body>
</html>
```

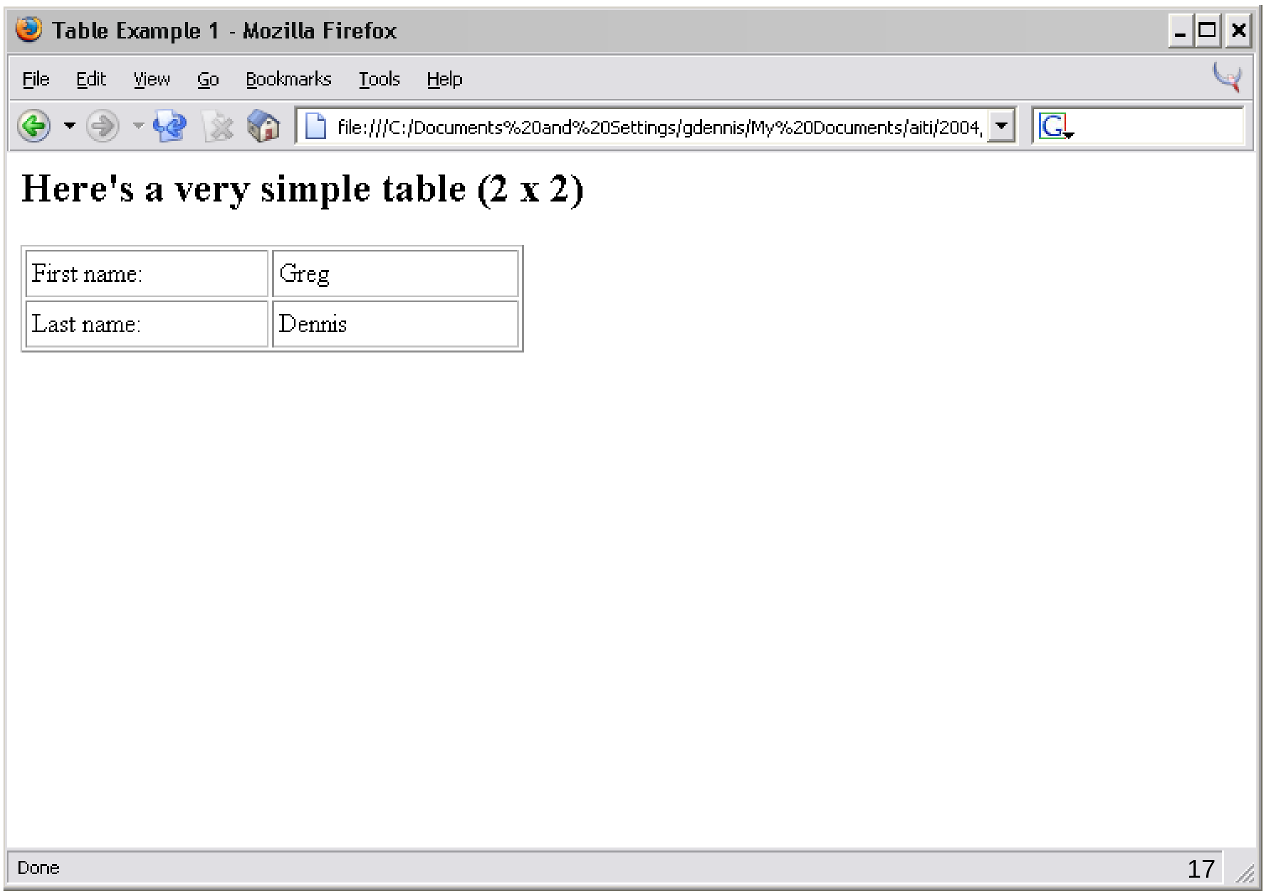


Table Example 1 - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
file:///C:/Documents%20and%20Settings/gdennis/My%20Documents/aiti/2004,
Done

Here's a very simple table (2 x 2)

First name:	Greg
Last name:	Dennis

Table Example 2

```
<html>
<head><title>Table Example 2</title></head>
<body bgcolor="ffffff">
<h2>Here's a slightly more complicated table</h2>
<table border=1 cellpadding=3 cellspacing=2
       width=400 height=350 bgcolor="99CCFF">
  <tr>
    <td colspan=2 width=400 height=50>
      Student List</td>
  </tr>
  <tr>
    <td width=200 height=100>Last name:</td>
    <td width=200 valign="top">Dennis</td>
  </tr>
  <tr>
    <td width=200 height=100>Last name:</td>
    <td width=200 align="right">Kim</td>
  </tr>
  <tr bgcolor="3366CC">
    <td width=200 height=100>Last name:</td>
    <td width=200 bgcolor="33CCCC">Grant</td>
  </tr>
</table>
</body>
</html>
```

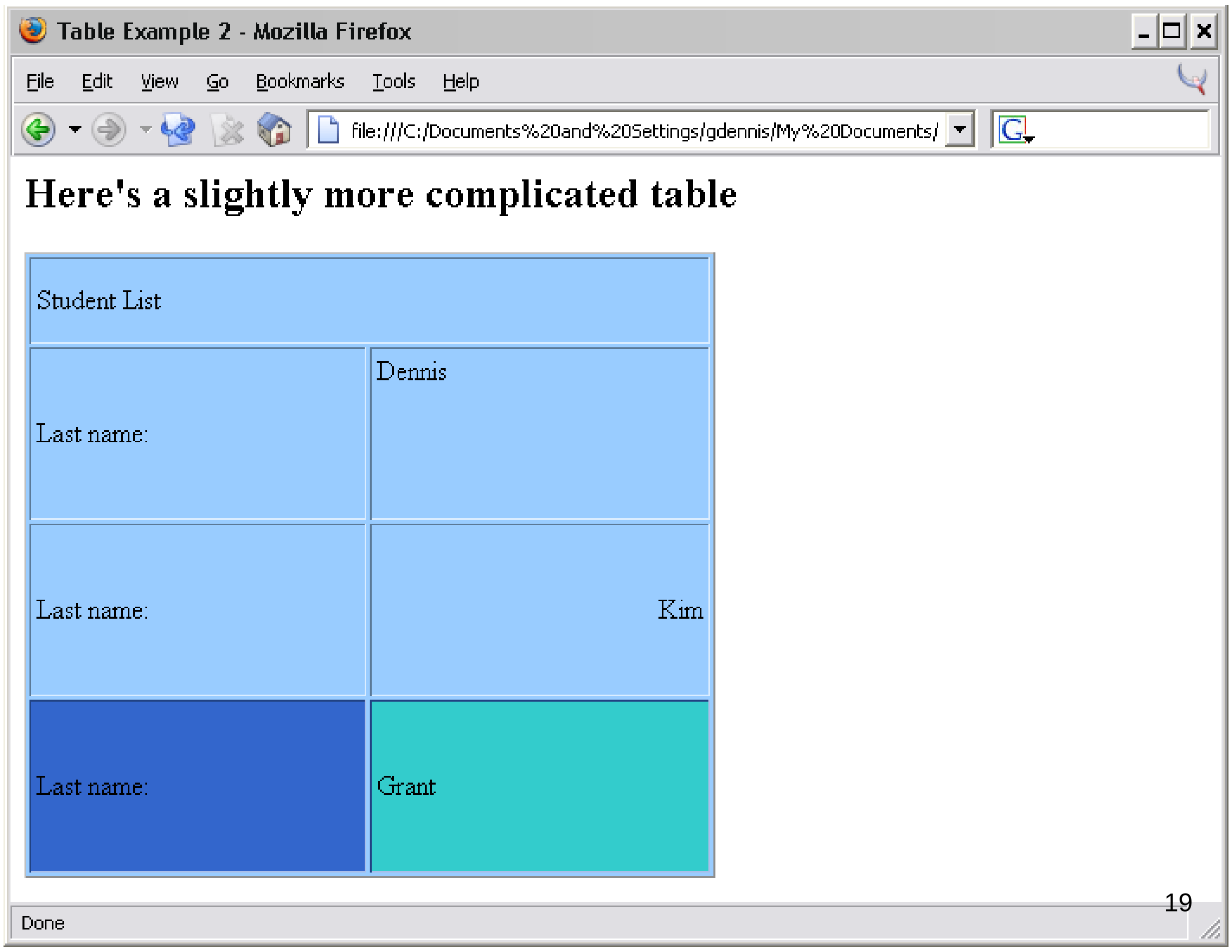


Table Example 2 - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
file:///C:/Documents%20and%20Settings/gdennis/My%20Documents/
Done

Here's a slightly more complicated table

Student List	
Last name:	Dennis
Last name:	Kim
Last name:	Grant

Important <table> Attributes

- align – aligns the table to the *left, right,* or *center*
- bgcolor – specifies a background color for the entire table
- border – specifies a width (in pixels) of the border around the table and its cells
- cellpadding – sets the amount of space (in pixels) between the cell border and its contents
- cellspacing – sets the amounts of space (in pixels) between table cells
- height – specifies the height of the entire table (pixels or percentage)
- width – specifies the width of the entire table (pixels or percentage)

Important <tr> Attributes

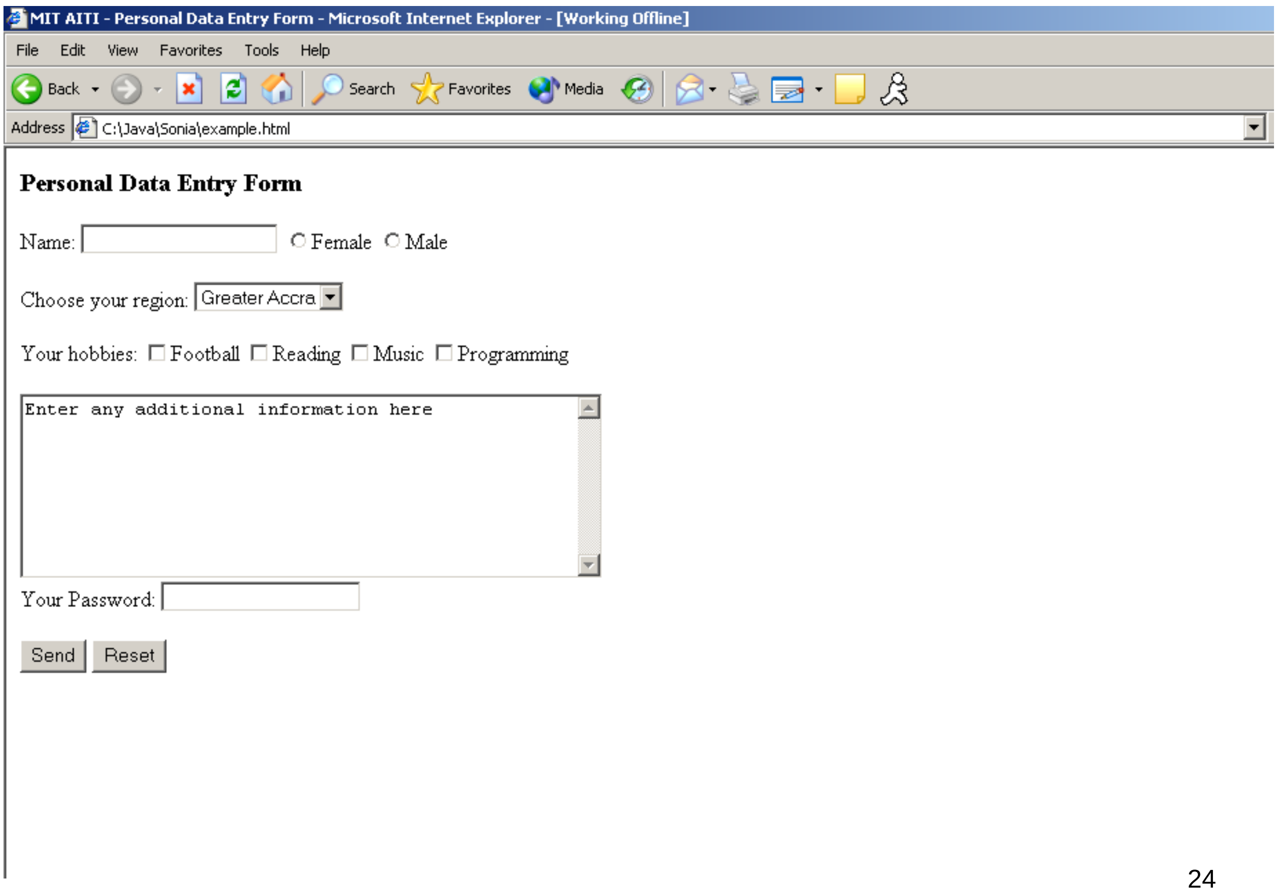
- align – aligns the row to the *left, right,* or *center*
- bgcolor – specifies a background color for the entire row (overrides the table's bgcolor)
- valign – specifies the vertical alignment of the text within the cell or row to *top, middle,* or *bottom*

Important <td> attributes

- align – aligns the cell to the *left, right,* or *center*
- bgcolor – specifies a background color for the cell (overrides table or row color)
- colspan – specifies the number of columns a cell should span
- height – specifies the height of the cell in pixels or percentage (relative to table)
- rowspan – specifies the number of rows spanned by a current cell
- valign – specifies the vertical alignment of the text within the cell to *top, middle,* or *bottom*
- width – specifies the width of the cell in pixels or percentage (relative to table)

HTML Forms

- Use forms to get information from users
- Have interacted with web forms anytime you have typed words, selected buttons or clicked checkboxes
- Learn how to create the "front end" of a form, which is the look and feel of the form, using HTML


MIT AITI - Personal Data Entry Form - Microsoft Internet Explorer - [Working Offline]
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\Java\Sonia\example.html

Personal Data Entry Form

Name: ○ Female ○ Male

Choose your region: Greater Accra

Your hobbies: ☐ Football ☐ Reading ☐ Music ☐ Programming

```
Enter any additional information here
```

Your Password:

Send Reset

```
<form action="process_data.jsp" method="POST">
 Name: <input type="text" name="name">
 <input type="radio" name="gender" value="female">Female
 <input type="radio" name="gender" value="male">Male
 <br><br>Choose your region:
 <select name="region">
     <option>Ashanti</option>  <option>Brong Ahafo</option>
     <option>Central</option>  <option>Eastern</option>
     <option selected>Greater Accra</option>
     <option>Northern</option> <option>Upper</option>
     <option>Volta</option>    <option>Western</option>
 </select><br><br>
 Your hobbies:
 <input type="checkbox" name="hobby" value="fball">Football
 <input type="checkbox" name="hobby" value="read">Reading
 <input type="checkbox" name="hobby" value="music">Music
 <input type="checkbox" name="hobby" value="java">Java<br>
 <textarea name="info" cols=50 rows=8>More Info</textarea>
 Your Password: <input type="password" name="pwd"><br><br>
 <input type="submit" value="Send"> <input type="reset">
 <input type="hidden" name="id" value="497">
</form>
```

Form Tags and Attributes

- <form> - indicates the beginning and end of a form; there can be multiple forms in one page but they cannot be nested and must never overlap
  - action – a URL which will process the form when it is submitted
  - method – get or post; get adds the information at the end of the URL, post adds the information in the HTML header
- <input type=checkbox> - this creates a checkbox;
  - checked: when added, the checkbox will be checked by default
  - name: assigns a name to the checkbox to be passed to the form processing page
  - value: specifies a value that will be passed; if not specified, "on" will be used

Form Tags and Attributes …

- <input type=radio> - creates a radio button; when various radio buttons share the same name only one can be selected
  - checked: select the button as default
  - name: assigns a name to the button
  - value: value passed to processing page
- <input type=submit> - creates a submit button that sends the information in a form
  - value: specifies text to appear on button
- <input type=reset>
  - creates a reset button that clears the contents of an entire form
  - value: specifies text to appear on button

Form Tags and Attributes …

- <input type=hidden> - creates a hidden element that is not displayed
  - name: name of hidden input
  - value – same as checkbox
- <input type=text> - creates a text input element
  - maxlength: max # of characters
  - name: name of textbox passed to processing page
  - size: size of the textbox
  - value: value passed to the processing page
- <input type=password> - creates a text input element with the text rendered so that it hides the characters (usually with *'s)
  - maxlength: maximum # of characters allowed
  - name: same as above
  - size: specifies the size of the text entry box
  - value: same as above

Form Tags and Attributes …

- <select> - defines a multiple choice menu or scrolling list; contains <option> tags
  - multiple: allows the user to select more than one option
  - name: name of drop down
  - size: same as above
- <option> - defines an option within a select element
  - selected: makes this item selected initially
  - value: value of menu option
- <textarea> - creates a multiline entry; the text within the tag will be displayed when the form is displayed
  - cols: specifies the visible width of the field in # of characters
  - name: name of text area
  - rows: specifies height
  - wrap: off/virtual/physical; sets the word wrap for the textarea

JSP – Lecture 1

JSP Basics

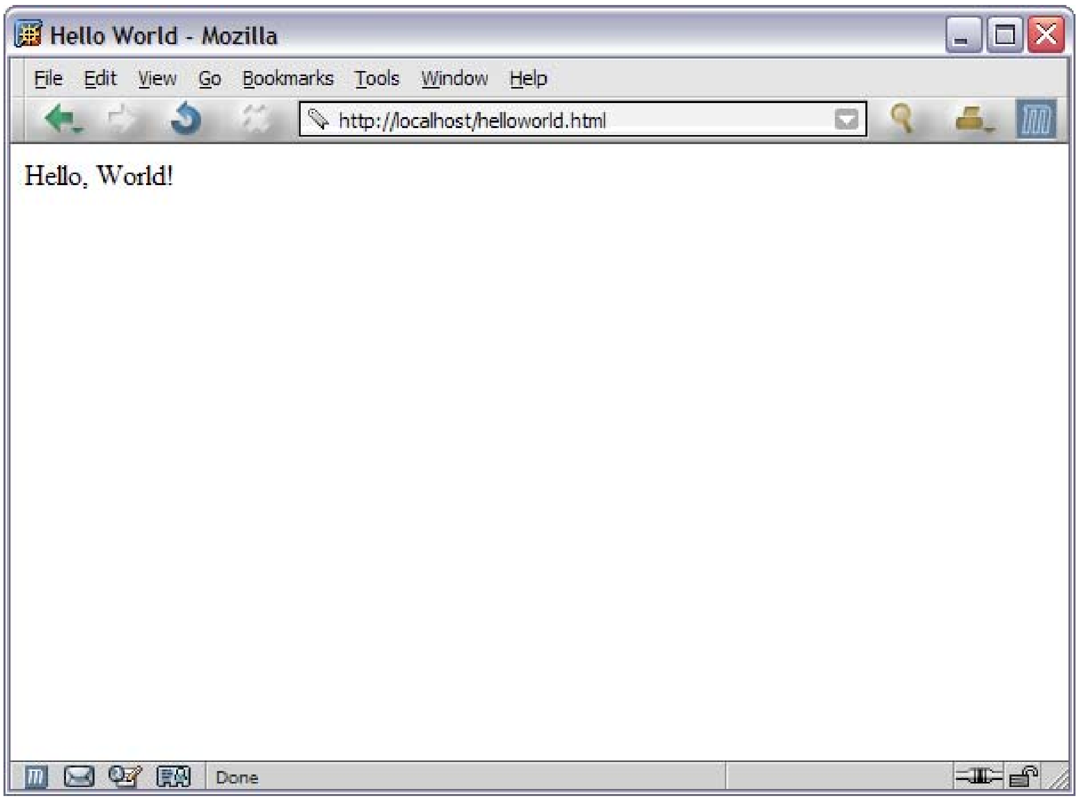
Original Slides from MIT AITI 2004

HTML Review

- `helloworld.html`

```
<HTML>
<HEAD>
<TITLE>Hello World</TITLE>
</HEAD>
<BODY>
      Hello, World!
</BODY>
</HTML>
```

Hello World Snapshot


Hello World - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://localhost/helloworld.html
Done

Hello, World!

HTML is Static

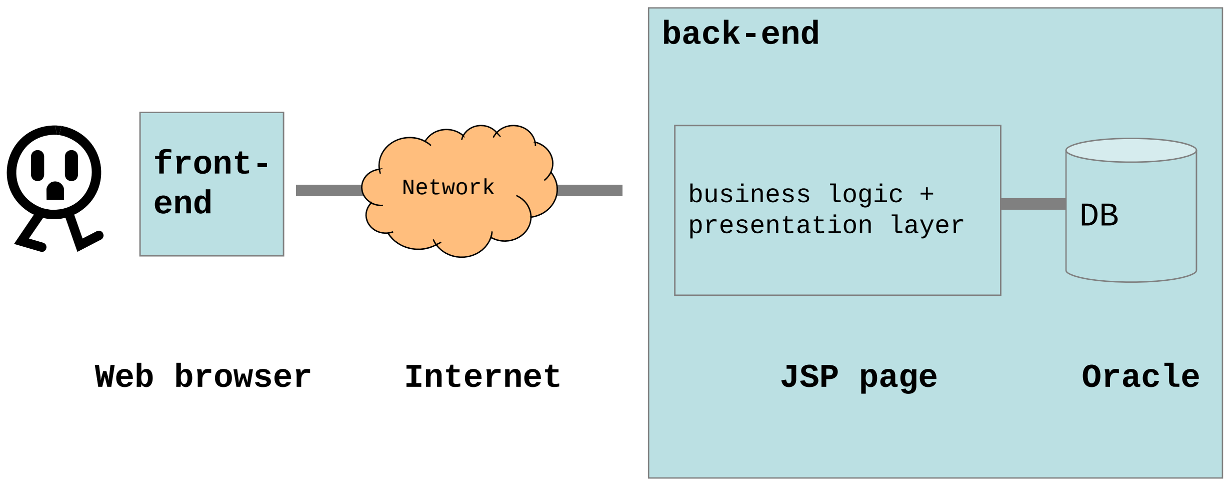
- HTML page shows the same thing every time you load it in your browser
- But you may want the content to change
  - Show latest weather, news, scores, etc . . .
  - Disallow certain people for logging in
  - Remember user's preferences for future

JSP to the Rescue!

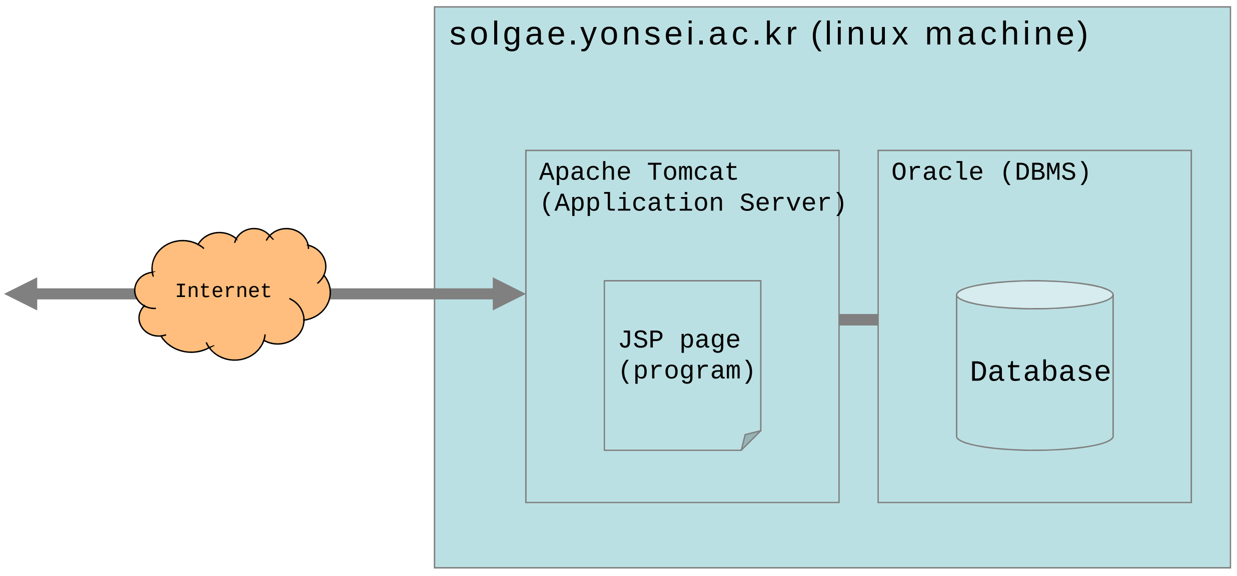
- JSP = Java Server Pages
- Combines Java and HTML to create dynamic (changing) Web pages
- Similar technologies:

  ASP, PHP, Perl, Cold Fusion, etc.
- But JSP is the only one in Java!

Architecture



Back-End Architecture


solgae.yonsei.ac.kr (linux machine)
Apache Tomcat (Application Server)
Oracle (DBMS)
Internet
JSP page (program)
Database

More Details: Servlets & JSP

- The purpose of a servlet is to create a Web page in response to a client request
- Servlets are written in Java, with a little HTML mixed in
  - The HTML is enclosed in `out.println( )` statements
- JSP (Java Server Pages) is an alternate way of creating servlets
  - JSP is written as ordinary HTML, with a little Java mixed in
  - The Java is enclosed in special tags, such as `<% ... %>`
  - The HTML is known as the template text
- JSP files must have the extension `.jsp`
  - JSP is *translated* into a Java servlet, which is then *compiled*
  - Servlets are run in the usual way
  - The browser or other client sees only the resultant HTML, as usual
- Tomcat knows how to handle servlets and JSP pages

More Details: How JSP works

- When Tomcat needs to use a JSP page, it:
  - Translates the JSP into a Java servlet
  - Compiles the servlet
  - Creates one instance of the JSP servlet
  - Executes the servlet as normal Java code
  - Hence, when you are writing JSP, you are writing “higher-level” Java code
- Each call to the JSP servlet is executed in a new Thread
  - Since there is only one JSP object, you have to use synchronization if you use any instance variables of the servlet
- Bottom line: JSP is just a convenient way of writing Java code!

Your First JSP Page

- `helloworld.jsp`

```
<HTML>
<HEAD>
<TITLE>Hello World</TITLE>
</HEAD>
<BODY>
       Hello, World!
</BODY>
</HTML>
```

- `Every legal HTML page is a legal JSP page`

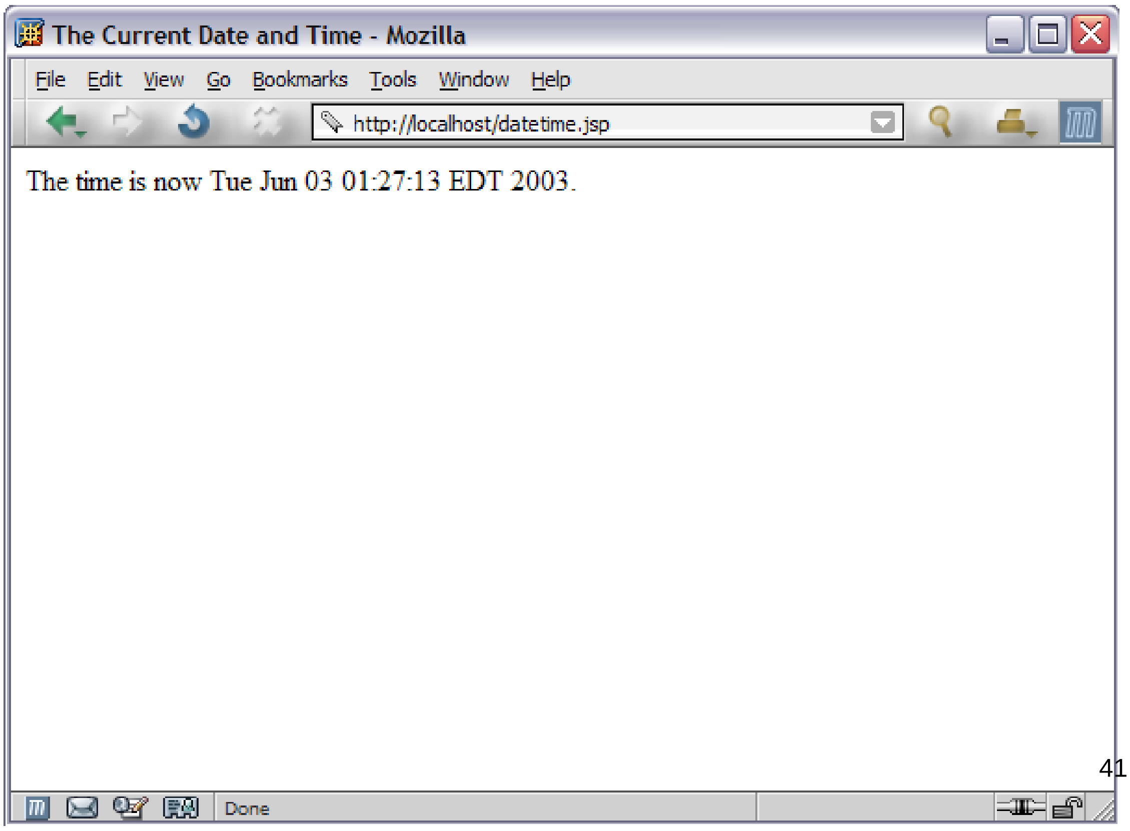
JSP Expressions

- JSP Page to show the time of day:

```
<HTML>
<HEAD>
<TITLE>The Current Date and Time</TITLE>
</HEAD>
<BODY>
     The time is now <%=new java.util.Date()%>.
</BODY>
</HTML>
```

- Any Java expression inside <%=. . . %> tags will be printed to HTML

Date and Time Snapshot


The Current Date and Time - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://localhost/datetime.jsp
Done

The time is now Tue Jun 03 01:27:13 EDT 2003.

JSP Expressions 2

- Other examples:
  - `<%="<B>" + new java.util.Date() + "</B>"%>`
  - `<%= "Hello, World!" %>`
  - `<%= i %>, for some integer variable i`
- Prints to HTML like System.out.println does:
  - for numbers, prints the number
  - for booleans, prints "true" or "false"
  - for null, prints "null"
  - for non-null objects, prints `Object.toString()`

JSP Scriptlets

- Another page to show the current time:

```
<%
    java.util.Date now = new java.util.Date();
%>
<HTML>
<BODY>
    The time is now <%= now %>
</BODY>
</HTML>
```

- Java code in `<% . . . %>` tags executed.

JSP Scriptlets 2

- We can intersperse code and HTML

```
<BODY>
<%
    if (Math.random() > 0.5) {
    %>
        Hello, World
        <%
    } else {
        %>
        Goodbye, World
        <%
    }
%>
</BODY>
```

JSP Scriptlets 3

- Alternatively . . .

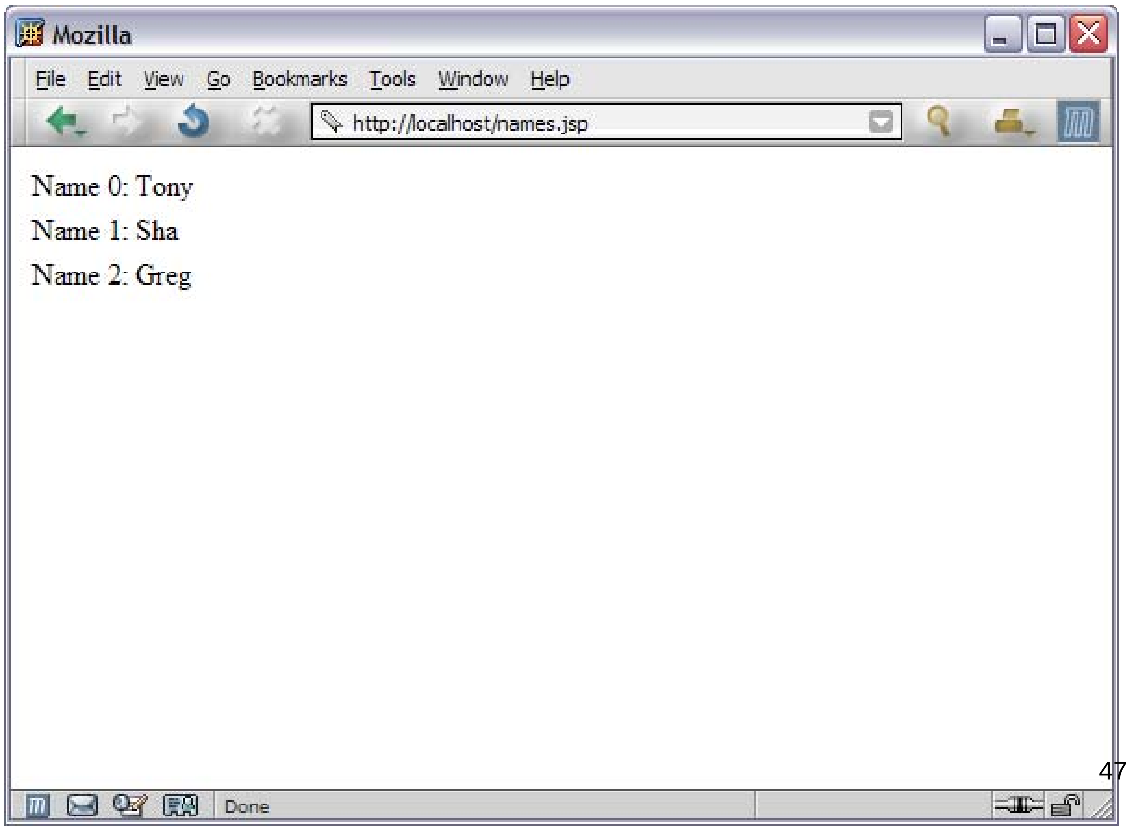
```
<BODY>
<%
    String greeting;
    if (Math.random() > 0.5) {
         greeting = "Hello, World";
    } else {
         greeting = "Goodbye, World"
    }
%>
<%= greeting %>
</BODY>
```

JSP Scriptlets 4

- Example of JSP with iteration

```
<TABLE>
<%
  String[] names = {"Tony", "Sha", "Greg"};
  for (int i = 0; i < names.length; i++) {
    %>
    <tr><td>Name <%=i%>:</td>
        <td><%=names[i]%></td></tr>
    <%
  }
%>
</TABLE>
```

Iteration Snapshot


Mozilla
File Edit View Go Bookmarks Tools Window Help
http://localhost/names.jsp
Done

```
Name 0: Tony
Name 1: Sha
Name 2: Greg
```

JSP Declarations

- Declare methods and variables that are reused every time the page is loaded.

```
<%!
     int n = 2;
     int addn(int i) {
         return i + n;
     }
%>
<%= addn(5) %>
```

- Q: What does this print to the screen?

JSP Declarations 2

- Q: Are these two equivalent?

```
<% double randomNum =
        Math.random(); %>
<%= randomNum %>
```

```
<%! double randomNum =
        Math.random(); %>
<%= randomNum %>
```

- A: No! While the left prints out a new random number each time, the right prints out the same one. Declarations declare variables that are reused on every load.

Page Directive

- How do we avoid writing out "java.util.Date"?
- In Java, we would write

```
import java.util.Date;
```

- In JSP, we use a page *directive*:

```
<%@ page import="java.util.Date" %>
```

- We will learn other directives in this class.
  All use the `<%@ . . . %>` tags.

Using the Page Directive

```
<%@page import="java.util.Date" %>

<%
    Date now = new Date();
%>

<HTML>
<BODY>
    The time is now <%= now %>
</BODY>
</HTML>
```

Quick JSP Quiz

- Which print out the current date on each load?

```
<%
    Date d = new Date();
%>
<%= d %>
```

```
<%!
    Date d = new Date();
%>
<%= d %>
```

```
<%!
    Date d = new Date();
    Date getDate() {
        return d;
    }
%>
<%= getDate() %>
```

```
<%!
    Date getDate() {
        return new Date();
    }
%>
<%= getDate() %>
```

JSP Review

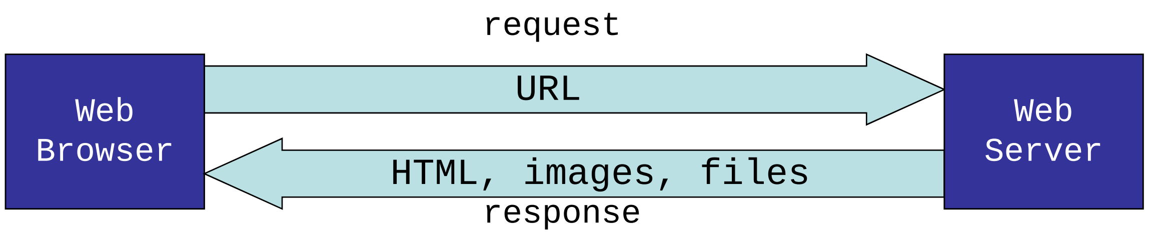
- Expressions `<%= . . . %>`
  - Prints a Java Expression to HTML
  - Example: `<%= new Date() %>`
- Scriptlets `<% . . . %>`
  - Executes Java code block
  - Example: `<% Date now = new Date() %>`
- Declarations `<%! . . . %>`
  - Declare global methods and variables
  - Example: `<%! Date getDate() {return new Date();} %>`
- Page Directive `<%@ page import = . . . %>`
  - Imports a Java class or classes
  - Example `<%@ page import="java.util.Date" %>`

JSP – Lecture 2

Get and Post Requests

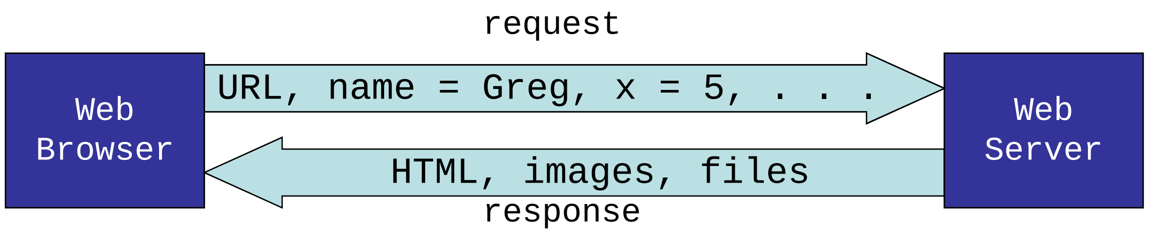
Original Slides from MIT AITI 2004

Request-Response Cycle



- When enter an address (URL) into the address bar of a web browser or click on a link, we generate a **request** for a file
- The request is routed to a Web server, which sends back a **response**, usually an HTML page

Not Interactive :-(



- So far request just includes a URL
- We still cannot
  - Login
  - Enter search queries
  - Make online purchases
- We can also send arguments in the request!

Get Method

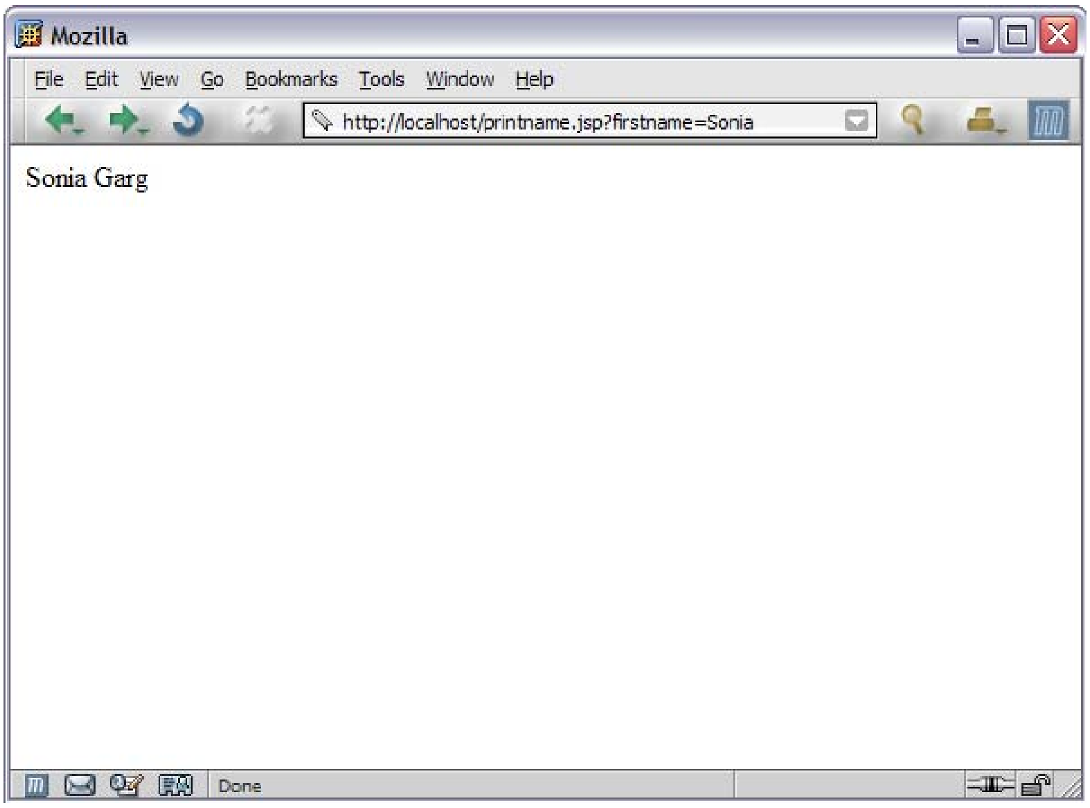
- Send parameters in the URL

`http://www.domain.com/printname.jsp?firstname=Sonia`

- **`printname.jsp`**

```
<%
  String first = request.getParameter("firstname");
  if (first.equals("Sonia")) {
    %> Sonia Garg <%
  } else if(first.equals("Greg") {
    %> Greg Dennis <%
  } else if (first.equals("Eric") {
    %> Eric Mibuari <%
  }
%>
```

URL Parameter Snapshot



http://localhost/printname.jsp?firstname=Sonia

Sonia Garg

URL Encoding

- Send parameters to a page via a URL
  ```
  page.jsp?param1=value1&param2=value2&...
  ```
- Get the valueX of paramX with:
  ```
  String valueX = request.getParameter("paramX");
  ```
- Request Object
  - *Implicit Object* = it is never declared
  - Every JSP page automatically has it
  - Contains the parameters passed to the page

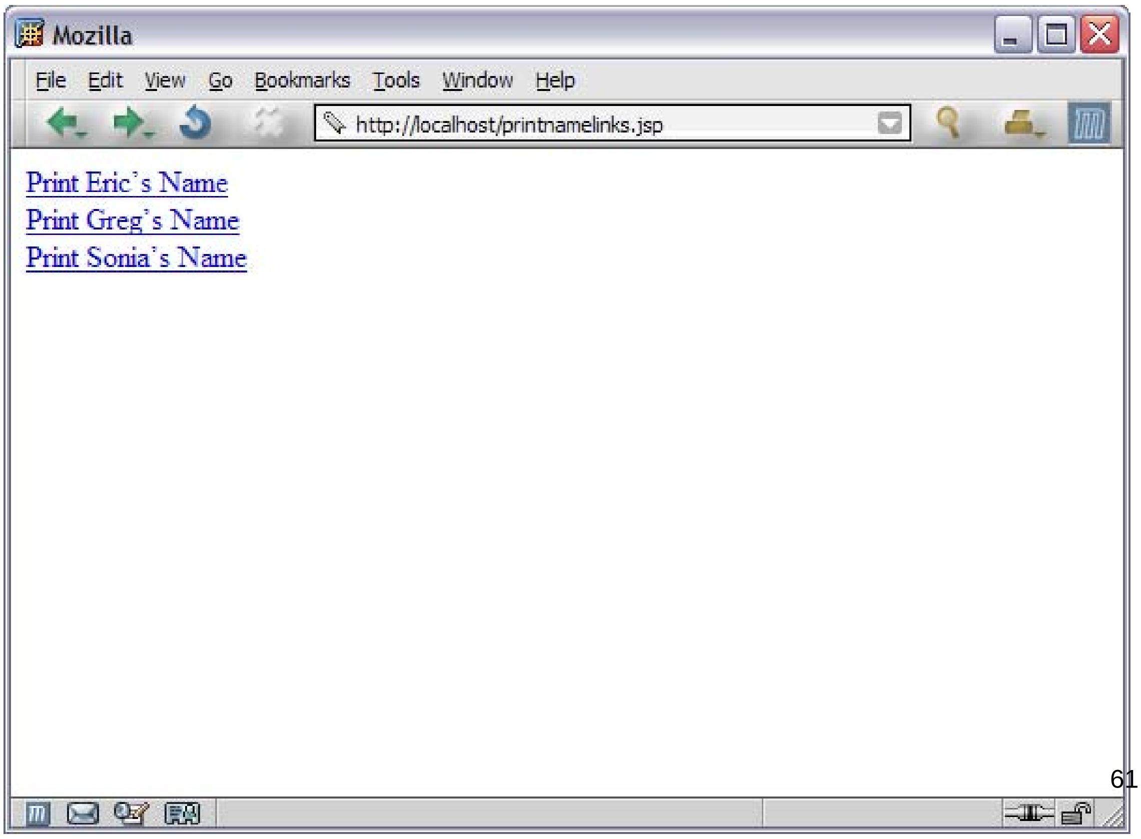
URL Encoding Example

- **printnamelinks.jsp:**

```
<%!
  String[] names = {"Eric", "Greg", "Sonia"};
%>

<%
  for(int i = 0; i < names.length; i++) {
    %>
    <a href="printname.jsp?firstname=<%=names[i]%>">
      Print <%=names[i]%>'s Name</a>
    <br>
    <%
  }
%>
```

URL Encoding Snapshot


Mozilla
File Edit View Go Bookmarks Tools Window Help
http://localhost/printnamelinks.jsp

Print Eric's Name
Print Greg's Name
Print Sonia's Name

HTML Forms

- Send parameters via forms

```
<FORM ACTION="printname.jsp" METHOD="get">
<INPUT TYPE="text" NAME="firstname" SIZE=40>
<INPUT TYPE="submit" value="Print Name">
</FORM>
```

- Access these with the request object

```
<%
  String name = request.getParameter("firstname");
  . . .
%>
```

Post Method

- Post attaches parameter values to request
- `login.html`

```
<FORM ACTION="checklogin.jsp" METHOD="post">
<INPUT TYPE="password" NAME="passwd" SIZE=20>
<INPUT TYPE="submit" value="Login">
</FORM>
```

- `checklogin.jsp`

```
<%
 String password = request.getParameter("passwd");
 if (password.equals("somePassword")) {
   %> Correct password <%
 else {
   %> Incorrect password <%
 }
%>
```

Get versus Post

	GET	POST
parameters	encoded in URL	attached to request
data limit	URL length	no limit
bookmark	yes	no
reload warning	no	yes

- Default to POST
- Use GET if
  - No passwords
  - Short data
  - Request has no side effects

Form Data Validation

- What if user leaves a text field blank?
- What if user types a letter instead of a number?
- What if user types a negative number for their age?

- Need to validate data typed into a form!

How to Validate Form Data

1. Get the String value of the parameter
2. Check if string is empty
   – Error if must provide non-empty value
3. Convert it to proper datatype
   – Error if value not in incorrect format
4. Check if value in proper range
   – Error if illegal value for parameter

Data Validation Example

- **`print_age.jsp`**

```
<%
  String ageStr = request.getParameter("age");
   if (ageStr.length() == 0) {
       %> Error: must provide an age <%
  } else {
    try {
      int age = Integer.parseInt(ageStr);
      if (age <= 0) {
        %> Error: age must be positive <%
      } else {
        %> Your age is <%=age%> <%
      }
    } catch (NumberFormatException e) {
      %> Error: age must be a valid integer <%
    }
  }
%>
```

Quick Get and Post Quiz

- Should we use Get or Post for . . .
  - Login with password — POST
  - Entering search query — GET
  - Adding a message to a bulletin board — POST
  - Making a purchase online — POST
- Describe process for validating a parameter value is a score on a test, e.g. "92.5"

1. value is not empty
2. value converts to a double with `Double.parseDouble`
3. double value is between 0 and 100 inclusive

Get and Post Review

- Get puts parameters and values in the URL
  `page.jsp?param1=value1&param2=value2&...`
  – Send with a link or in a form with `method="get"`
- Post attaches parameters to request
  – Send in a form with `method="post"`
- Get value of parameter through request object
  `String valueX = request.getParameter("paramX");`
- Must validate manually entered form data

JSP – Lecture 3

Database Connection

Databases and Web Application

- Variety of databases available for use by web applications

- Typically will use relational database with support for Structured Query Language

- Examples of common databases used:
SQL server, MySQL, Oracle, Access

Accessing a database from JSP

- Need to identify and connect to the database to be used with the JSP page:

  1) **Global datasource:** Can specifying a default datasource in a Tomcat configuration file for the application called the *web.xml* file. The datasource will automatically be made available to the JSP if done this way – Good approach for larger applications.

  OR

  2) **Direct from JSP:** by specifying the database details directly within the JSP page. Use instead of (1) all the time OR just to override the default data source specified in (1)

Accessing a database directly from JSP page

Will use 2) for development purposes.
(Useful for smaller applications)

Using option 2):

- Can use java code(via scriptlets) OR
- JSTL <SQL> tags to access databases

1. Using Java Codes

- Load a driver
- Connect to the database
- Create a `Statement`
- Execute the `Statement`
- Process the `ResultSet`

Loading the Driver

```
<%@ page import="java.sql.*"%>

 Class.forName( "oracle.jdbc.driver.O
 racleDriver");
```

- This specified the database driver to load
- This driver can then be used in subsequent calls to `DriverManager.getConnection()`

Creating the Connection

```
Connection con =
  DriverManager.getConnection(
  "jdbc:oracle:thin:@localhost:1521:orcl",
  “system", “your_passwd");
```

- Specifies the database URL, the user name and the password
- This URL is same to the URL which we have used in the JDBC examples
- Also possible to have web urls, with different drivers

Creating a Statement

```
Statement statement = con.createStatement();
```

- A Statement is used to execute SQL calls on the database
- Common methods are:

```
ResultSet rs = statement.executeQuery(String query);
int nRows = statement.executeUpdate(String update);
```

Using the ResultSet

- Iterating over the ResultSet
- Extracting fields from each tuple

```
ResultSet rs = stmt.executeQuery(select);
while ( rs.next() ) {
    out.println(rs.getString(“name") );
}
```

- Here we used getString() – can also get other types

Do It Yourself !!

```
<HTML><BODY>
<%@ page import="java.sql.*"%>
<%
   String DB_URL = "jdbc:oracle:thin:@localhost:1521:orcl";
   try{
      Class.forName("oracle.jdbc.driver.OracleDriver");
      Connection con = DriverManager.getConnection(DB_URL, “system", "your_passwd");

      Statement stmt = con.createStatement();
      ResultSet rs = stmt.executeQuery("SELECT id, name FROM instructor");
      while (rs.next()) {
         int id = rs.getInt("id");
         String name = rs.getString("name");
         out.println(id + " : " + name + "<BR>");
      }
        rs.close();
        stmt.close();
        con.close();
    } catch (Exception e){
      out.println("ERROR");
      e.printStackTrace();
   }
%>
</BODY></HTML>
```

Another Example

```
<body>

<%
    try {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        System.out.println("loaded class");
        Connection con =
                DriverManager.getConnection(
                        "jdbc:odbc:artshop", "", "");
%>
  <table>

  <tr><th>Product</th><th>Price</th></tr>
```

Another Example (cont'd)

```
 <%    // now for each row of the table
        Statement s = con.createStatement();
        ResultSet rs = s.executeQuery("Select * from Product");

        while(rs.next()) {
%>
   <tr> <td> <%= rs.getString("Title") %> </td>
   <td> <%= rs.getString("Price") %>  </td></tr>
<%
        }
%>

  </table>
<%
    } catch (Exception e) {
        out.println(e);
    }
%>
</body>
```

Generalising DB Code

- Not good to put SQL directly in your JSP code
- Generally better to define DB access in an interface
- This makes the JSP code
  - Neater
  - Easier to understand
  - Easier to debug

Example Atomic Transaction

- Following outline example uses a JDBC connection to make an atomic transaction
- It assumes that the Strings *update1 and update2* have been set up appropriately
- And that DB constraints have been placed on the value of an account (e.g. not allowed to be negative)
- Note that `con` is of type `java.sql.Connection`

JDBC Transaction

```
try {
  con.setAutoCommit( false );
  statement = con.createStatement();
  statement.executeUpdate( update1 );
  statement.executeUpdate( update2 );
  // to get here, both must have worked
  con.commit();
}
catch(SQLException e) { // something wrong!
  con.rollback();
}
```