

Agenda:

- Basic HTML concepts
- HTML Forms and Servlets
- Tomcat Common Bean Utility library

A crash course in HTML

HTML basics

- *HyperText Markup Language (HTML)* is used to provide the user interface for web applications.
- To write and edit HTML code and JSPs, you can use a general text editor like NotePad, a text editor that's specifically designed for working with HTML, or an *Integrated Development Environment*, or *IDE*, that's designed for developing web applications.
- An *HTML document* is used to define each *HTML page* that's displayed by a web browser.
- Within an HTML document, *HTML tags* define how the page will look when it is displayed. Each of these HTML tags is coded within a set of brackets (<>).
- HTML tags aren't case sensitive.
- To make your code easier to read, you can use spaces, indentation, and blank lines.

Basic HTML tags

Tag	Description
<code><!doctype ... ></code>	Identifies the type of HTML document. This tag is often inserted automatically by the HTML editor.
<code><html> </html></code>	Marks the beginning and end of an HTML document.
<code><head> </head></code>	Marks the beginning and end of the Head section of the HTML document.
<code><title> </title></code>	Marks the title that is displayed in the title bar of the browser.
<code><body> </body></code>	Marks the beginning and end of the Body section of the HTML document.

Basic HTML tags (cont.)

Tag	Description
<code><h1> </h1></code>	Tells the browser to use the default format for a heading-1 paragraph.
<code><h2> </h2></code>	Tells the browser to use the default format for a heading-2 paragraph.
<code><p> </p></code>	Tells the browser to use the default format for a standard paragraph.
<code>
</code>	Inserts a line break.
<code> </code>	Marks text as bold.
<code><i> </i></code>	Marks text as italic.
<code><u> </u></code>	Marks text as underlined.
<code><!-- --></code>	Defines a comment that is ignored by the browser.

Anchor tags...

With URLs that are relative to the current directory

```
<a href="join.html">The Email List application 1</a><br>
<a href="email/join.html">
    The Email List application 2</a><br>
```

With relative URLs that navigate up the directory structure

```
<a href=". . /">Go back one directory level</a><br>
<a href=". . . /">Go back two directory levels</a><br>
```

With URLs that are relative to the webapps directory

```
<a href="/">Go to the default root directory for the web
server</a><br>
<a href="/musicStore">Go to the root directory of the
musicStore app</a>
```

With absolute URLs

```
<a href=
    "http://www.murach.com/email">An Internet address</a>
<a href="http://64.71.179.86/email">An IP address</a>
```

The Anchor tag

Tag	Description
<code><a> </code>	Defines a link to another URL. When the user clicks on the text that's displayed by the tag, the browser requests the page that is identified by the Href attribute of the tag.

One attribute of the Anchor tag

Attribute	Description
<code>href</code>	Specifies the URL for the link.

The HTML code for a table

```
<p>Here is the information that you entered:</p>

<table cellspacing="5" cellpadding="5" border="1">
  <tr>
    <td align="right">First name:</td>
    <td>John</td>
  </tr>
  <tr>
    <td align="right">Last name:</td>
    <td>Smith</td>
  </tr>
  <tr>
    <td align="right">Email address:</td>
    <td>jsmith@hotmail.com</td>
  </tr>
</table>
```

The table displayed in a browser

Here is the information that you entered:

First name:	John
Last name:	Smith
Email address:	jsmith@hotmail.com

The tags for working with tables

Tag	Description
<code><table> </table></code>	Marks the start and end of the table.
<code><tr> </tr></code>	Marks the start and end of each row.
<code><td> </td></code>	Marks the start and end of each data cell within a row.

A form displayed in a browser before the user enters data

Here's a form that contains two text boxes and a button:

First name:

Last name:

A screenshot of a web browser window. Inside the window, there is a form with a light gray background. At the top left, the text "Here's a form that contains two text boxes and a button:" is displayed. Below this, there are two horizontal text input fields. The first field has the placeholder "First name:" to its left. The second field has the placeholder "Last name:" to its left. To the right of the second field is a rectangular button with the word "Submit" centered on it. The browser window has a dark gray border and a vertical scroll bar on the right side.

Description

- A *form* contains one or more *controls* such as text boxes, buttons, check boxes, and list boxes.

Web Site for HTML tags Demos

HTML Basic - Mozilla Firefox

File Edit View History Bookmarks Tools Help

HTML Basic

www.w3schools.com/html/html_basic.asp

w3schools.com

Search w3schools.com: Select Language | Google Custom Search

HOME HTML CSS JAVASCRIPT JQUERY XML ASP.NET PHP SQL MORE... REFERENCES EXAMPLES FORUM ABOUT

SQL Server Query Tool www.Confio.com/SQL-Server... Improve SQL Server Performance 65%, Save on Costs, Free Trial Download!

Join Google Get your business online for FREE—with help from Google. Free website includes domain name and hosting for one year.

REGISTER TODAY! Google

SHARE THIS PAGE

HTML Basic

HTML HOME

HTML Introduction

HTML Editors

HTML Basic

HTML Elements

HTML Attributes

HTML Headings

HTML Paragraphs

HTML Formatting

HTML Links

HTML Head

HTML CSS

HTML Images

HTML Tables

HTML Lists

HTML Blocks

HTML Layout

HTML Forms

HTML Iframes

HTML Colors

HTML Colornames

HTML Colorvalues

HTML Basic - 4 Examples

« Previous Next Chapter »

Don't worry if the examples use tags you have not learned. You will learn about them in the next chapters.

HTML Headings

HTML headings are defined with the <h1> to <h6> tags.

Example

```
<h1>This is a heading</h1>
<h2>This is a heading</h2>
<h3>This is a heading</h3>
```

Try it yourself »

HTML Paragraphs

HTML paragraphs are defined with the <p> tag.

Example

WEB HOSTING

Best Web Hosting eUK Web Hosting UK Reseller Hosting Domain, Hosting & Email

WEB BUILDING

XML Editor - Free Trial! FREE Website BUILDER FREE Website Creator Best Website Templates

STATISTICS

Browser Statistics OS Statistics Display Statistics

Nissan LEAF®

MOUSE OVER TO EXPLORE

Secure Search McAfee

Web Site for HTML tags Demos

The screenshot shows a Mozilla Firefox browser window with the title "Tryit Editor v1.8 - Mozilla Firefox". The address bar displays "www.w3schools.com/html/tryit.asp?filename=tryhtml_headers". The main content area is divided into two sections: "Source Code" on the left and "Result" on the right.

Source Code:

```
<!DOCTYPE html>
<html>
<body>

<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
<h5>This is heading 5</h5>
<h6>This is heading 6</h6>

</body>
</html>
```

Result:

This is heading 1
This is heading 2
This is heading 3
This is heading 4
This is heading 5
This is heading 6

At the bottom of the editor, there is a message: "Edit the code above and click "Submit Code" to see the result."

Web Site for HTML tags Demos

HTML Tables - Mozilla Firefox

File Edit View History Bookmarks Tools Help

HTML Tables

www.w3schools.com/html/html_tables.asp

Secure Search

HTML Object

- HTML Audio
- HTML Video
- HTML YouTube

HTML Examples

- HTML Examples
- HTML Quiz
- HTML5 Quiz
- HTML Certificate
- HTML5 Certificate

HTML References

- HTML Tag List
- HTML Attributes
- HTML Events
- HTML Canvas
- HTML Audio/Video
- HTML Doctypes
- HTML Colormames
- HTML Colorpicker
- HTML Colormixer
- HTML Character Sets
- HTML ASCII
- HTML ISO-8859-1
- HTML Symbols
- HTML URL Encode
- HTML Lang Codes
- HTTP Messages
- HTTP Methods
- Keyboard Shortcuts

HTML Table Headers

Header information in a table are defined with the `<th>` tag.

All major browsers display the text in the `<th>` element as bold and centered.

```
<table border="1">
<tr>
<th>Header 1</th>
<th>Header 2</th>
</tr>
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

How the HTML code above looks in your browser:

Header 1	Header 2
row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

More Examples

[Tables without borders](#)
How to create tables without borders.

[Table headers](#)
How to create table headers.

[Table with a caption](#)
How to add a caption to a table.

[Table cells that span more than one row/column](#)
How to define table cells that span more than one row or one column.

Secure Search

McAfee

Web Site for HTML tags Demos

Screenshot of a web browser window showing a Tryit Editor v1.8 interface for demonstrating HTML tables.

The browser title bar reads "Tryit Editor v1.8 - Mozilla Firefox". The tabs are "HTML Tables" and "Tryit Editor v1.8". The address bar shows "www.w3schools.com/html/tryit.asp?filename=tryhtml_tables".

Source Code:

```
<!DOCTYPE html>
<html>
<body>

<p>
Each table starts with a table tag.
Each table row starts with a tr tag.
Each table data starts with a td tag.
</p>

<h4>One column:</h4>
<table border="1">
<tr>
<td>100</td>
</tr>
</table>

<h4>One row and three columns:</h4>
<table border="1">
<tr>
<td>100</td>
<td>200</td>
<td>300</td>
</tr>
</table>

<h4>Two rows and three columns:</h4>
<table border="1">
<tr>
<td>100</td>
<td>200</td>
<td>300</td>
</tr>
<tr>
<td>400</td>
<td>500</td>
<td>600</td>
</tr>
</table>
```

Submit Code >

Result:

W3Schools.com - Try it yourself

Each table starts with a table tag. Each table row starts with a tr tag. Each table data starts with a td tag.

One column:

100

One row and three columns:

100	200	300
-----	-----	-----

Two rows and three columns:

100	200	300
400	500	600

Waiting for pubads.g.doubleclick.net... **Submit Code** to see the result.

Web Site for HTML tags Demos

Tryit Editor v1.8 - Mozilla Firefox

File Edit View History Bookmarks Tools Help

HTML Tables Tryit Editor v1.8

www.w3schools.com/html/tryit.asp?filename=tryhtml_tables

SQL Server Query Tool
www.Confio.com/SQL-Server
Identify Bottlenecks & Wait
Types. DBA Performance
Tools. Free Trial.

\$1 A WEEK for 12 WEEKS THE WALL STREET JOURNAL SUBSCRIBE NOW!

Source Code:

```
<!DOCTYPE html>
<html>
<body>

<p>
Hello World ... A. Bader
</p>

<p>
Each table starts with a table tag.
Each table row starts with a tr tag.
Each table data starts with a td tag.
</p>

<h4>One column:</h4>
<table border="1">
<tr>
<td>100</td>
</tr>
</table>

<h4>One row and three columns:</h4>
<table border="1">
<tr>
<td>100</td>
<td>200</td>
<td>300</td>
</tr>
</table>

<h4>Two rows and three columns:</h4>
<table border="1">
<tr>
<td>100</td>
<td>200</td>
<td>300</td>
</tr>
<tr>
<td>400</td>
<td>500</td>
<td>600</td>
</tr>
</table>
```

Submit Code >

Result:

W3Schools.com - Try it yourself

Each table starts with a table tag. Each table row starts with a tr tag. Each table data starts with a td tag.

One column:

100

One row and three columns:

100 200 300

Two rows and three columns:

100	200	300
400	500	600

Edit the code above and click "Submit Code" to see the result.

Secure Search McAfee

HTML forms

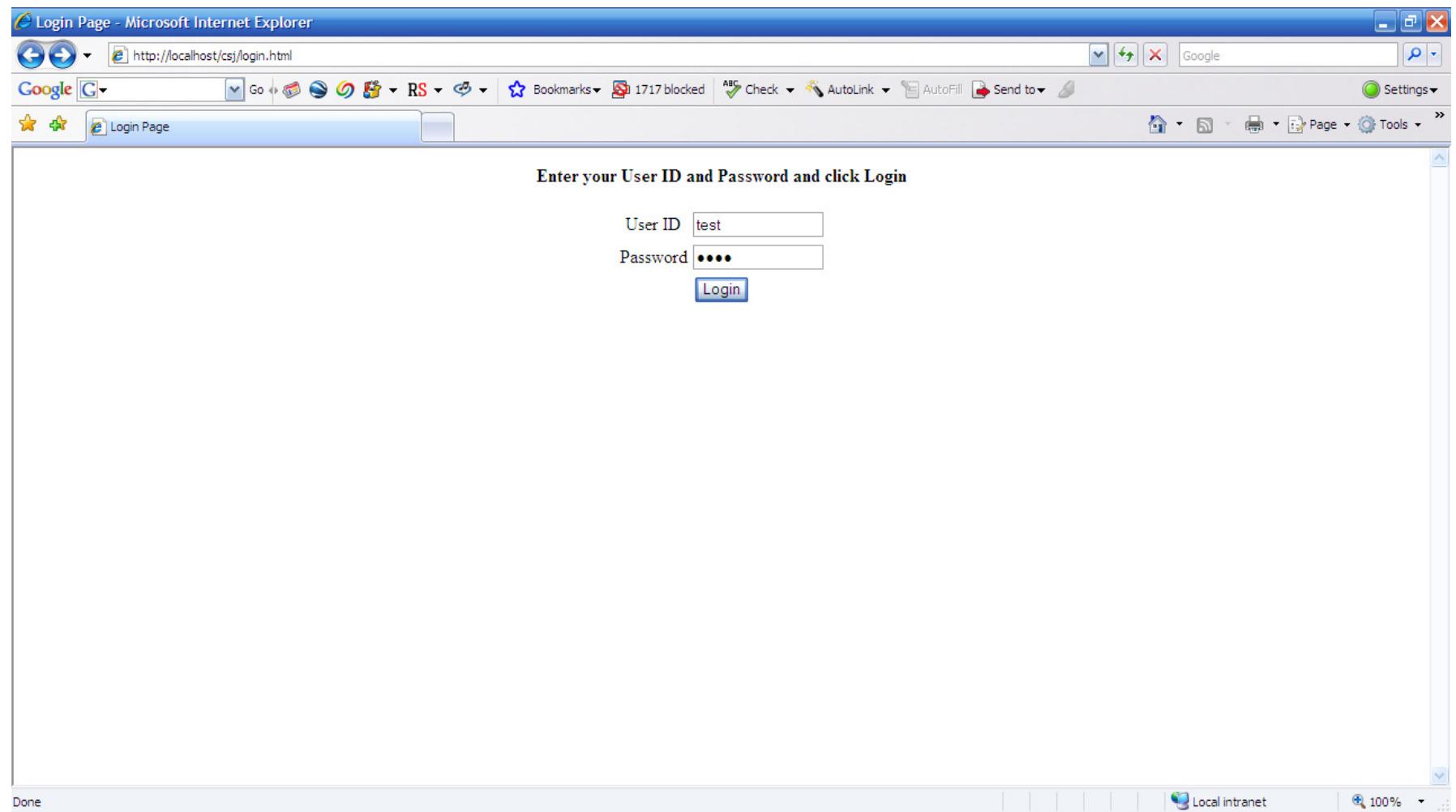
- An HTML form is identified in HTML source by the `<form>` tag

```
<form method=m action=uri>
    ... more html, including fields
</form>
```
- The **method** is the HTTP request method to use when submitting the data (POST or GET)
- The **action** attribute is the server resource that will handle the request. This points to the servlet URL that we will write to handle this form.

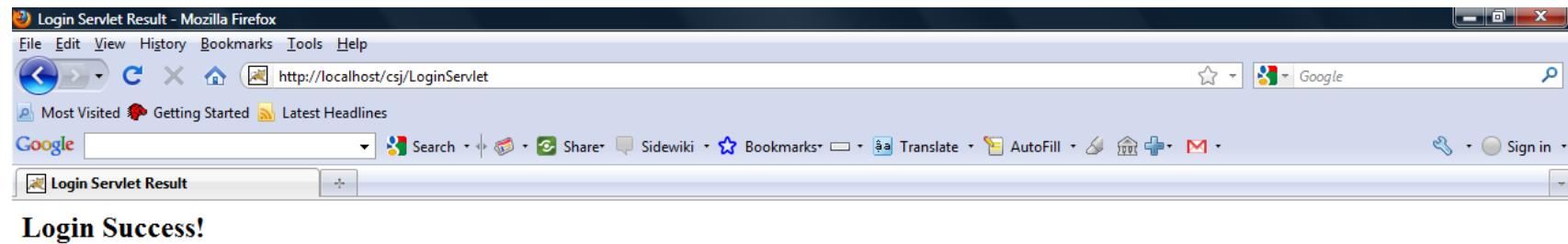
GET vs. POST

- The form data is submitted as part of the HTTP request as a sequence of *URL-encoded* name-value pairs.
- The GET method encodes the name-value pairs onto the request URL (i.e. Visible in the Location of the browser)
- The POST method encodes the name-value pairs in the message body of the request. (invisible to the user, but not encrypted)
- GET is the default method

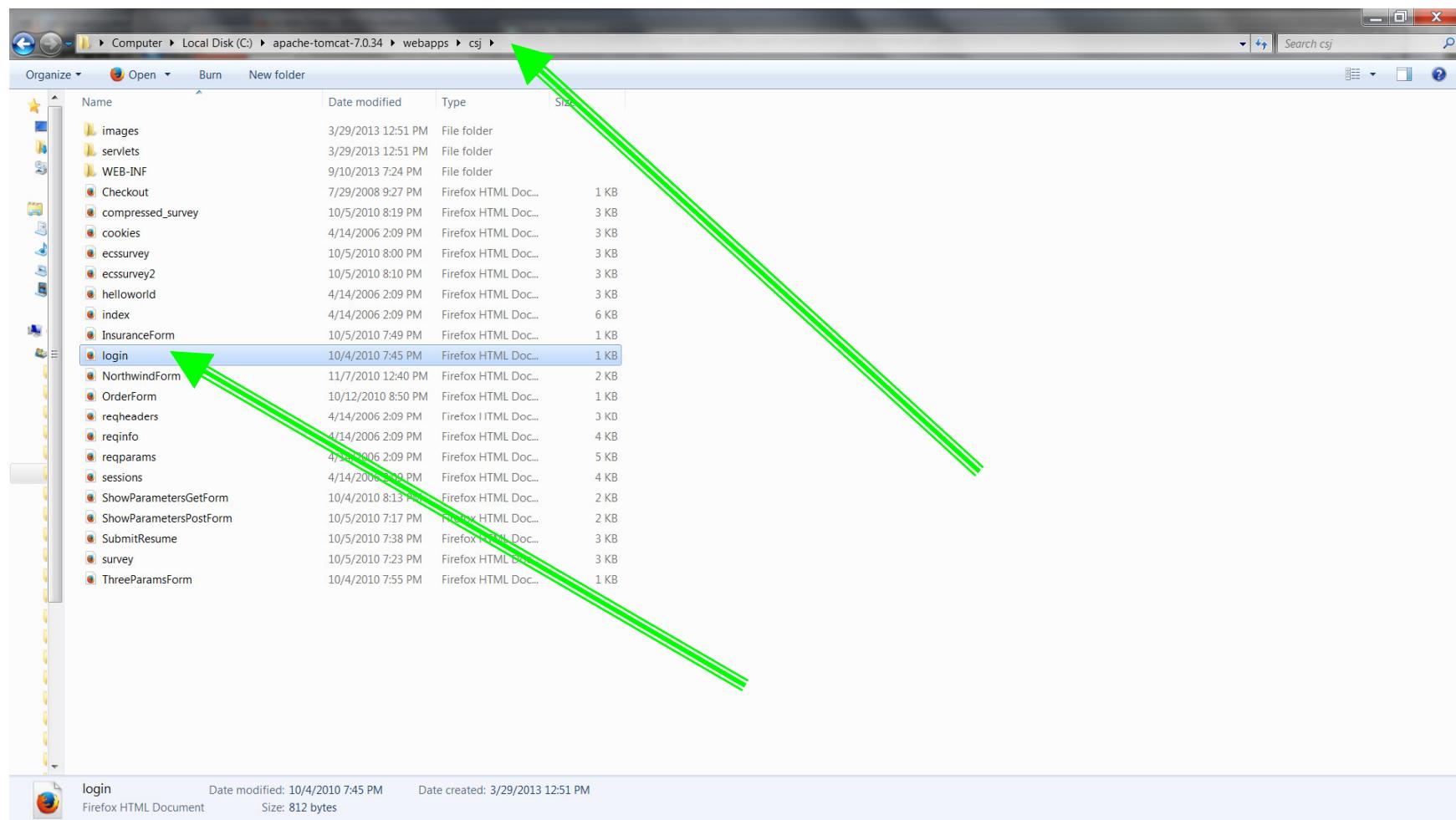
A Login HTML form



A Login HTML form



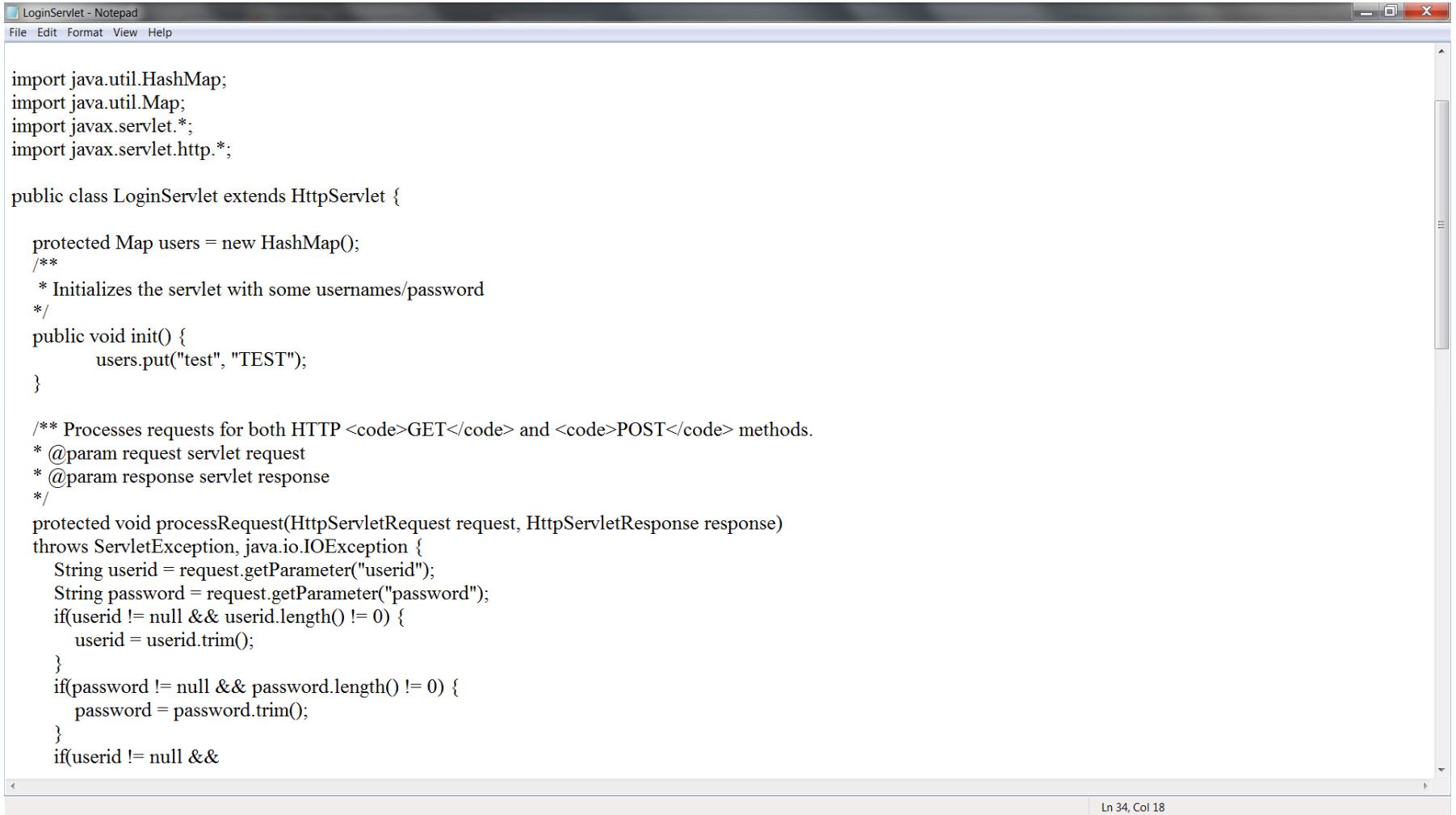
A Login HTML form



login.html

```
<html>
  <head><title>Login Page</title></head>
  <body>
    <form method="post" action="/csj/LoginServlet">
      <h4>Enter your User ID and Password and click Login</h4>
      <table cellpadding='2' cellspacing='1'>
        <tr><td>User ID</td>
          <td><input type="TEXT" size="15"
          name="userid"></input></td></tr>
        <tr><td>Password</td>
          <td><input type="PASSWORD" size="15"
          name="password"/></td></tr>
        <tr><td colspan='2'>
          <center><input type="SUBMIT" value="Login" /></center>
        </td></tr>
      </table>
    </form>
  </body>
</html>
```

LoginServlet.java



The screenshot shows a Windows Notepad window with the title "LoginServlet - Notepad". The window contains Java code for a servlet named LoginServlet. The code includes imports for HashMap, Map, and various javax.servlet and javax.servlet.http packages. It features a protected Map named users initialized with a single entry ("test", "TEST"). The init() method adds this entry. The processRequest() method handles both GET and POST requests, extracting userid and password parameters from the request, trimming them, and checking if they are null or empty. The code ends with a closing brace for the if(userid != null && password.length() != 0) block.

```
import java.util.HashMap;
import java.util.Map;
import javax.servlet.*;
import javax.servlet.http.*;

public class LoginServlet extends HttpServlet {

    protected Map users = new HashMap();
    /**
     * Initializes the servlet with some usernames/password
     */
    public void init() {
        users.put("test", "TEST");
    }

    /** Processes requests for both HTTP <code>GET</code> and <code>POST</code> methods.
     * @param request servlet request
     * @param response servlet response
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, java.io.IOException {
        String userid = request.getParameter("userid");
        String password = request.getParameter("password");
        if(userid != null && userid.length() != 0) {
            userid = userid.trim();
        }
        if(password != null && password.length() != 0) {
            password = password.trim();
        }
        if(userid != null &&
```

Request Parameters

- A servlet can access the data that is in the form via Request Parameters

String **getParameter**(String name)

- Returns the value of a Request Parameter as a string, or null if the parameter doesn't exist
- Works the same for GET or POST
- The name is case sensitive
- Does the hard work of parsing the QUERY_STRING environment variable for you.

HTML Forms: Radio Buttons

Company Type Private Public Government

- Only one option can be selected at a time

Company Type

```
<input type="radio" name="co_type"  
value="private" />Private &nbsp;  
<input type="radio" name="co_type"  
value="public" checked="true" />Public &nbsp;  
<input type="radio" name="co_type"  
value="government" />Government
```

HTML Forms: Text Input

First Name

- Allows arbitrary data to be entered
- The amount can be limited, but doesn't prevent the client from submitting more than the size in the request

First Name

```
<input type="text" name="first" size="20"/>
```

HTML Forms: Combo Box

State

- Allows single or multiple selection via the multiple option.
- Without multiple, it displays a Combo Box
- With multiple, it displays a Select Box

```
<select name="state">
    <option value="IL">Illinois</option>
    <option value="OH">Ohio</option>
    <option value="IN">Indiana</option>
    <option value="WI">Wisconsin</option>
    <option value="MI">Michigan</option>
</select>
```

HTML Forms: CheckBoxes

Languages Used Java C++ C Visual Basic C# Perl

- Like the Radio buttons, but allows more than one to be selected

```
<input type="checkbox" name="java">Java ;
<input type="checkbox" name="cpp">C++ ;
<input type="checkbox" name="c">C ;
<input type="checkbox" name="vb">Visual Basic ;
<input type="checkbox" name="csharp">C# ;
<input type="checkbox" name="perl">Perl ;
```

HTML Forms: TextArea

- Allows for more text to be entered than text inputs, over multiple rows.

Comments

```
<textarea name="comments" rows="2" cols="40">  
</textarea>
```

HTML Forms: Password Fields

- Just like a text input, but will display a '*' instead of the character typed.
- WARNING! – the value of the field is not encrypted. The data is just not displayed to the user. To encrypt your data, use SSL (Secure Sockets Layer).

More ServletRequest methods

Enumeration **getParameterNames ()**

- Returns an Enumeration of the parameter names in this request

String[] **getParameterValues (String name)**

- Returns a String array of values for a given Parameter name. This method is needed if you have parameters with multiple results, like a Select input.
- Returns null if the Parameter doesn't exist

Retrieving all form data

```
Enumeration parameters = request.getParameterNames();
if(parameters.hasMoreElements()) {
    out.println("<tr><th>Parameter Name</th><th>Parameter" +
                "Value</th></tr>");
}
while(parameters.hasMoreElements()) {
    String parameter = (String)parameters.nextElement();
    // get the parameter values
    String[] values = request.getParameterValues(parameter);
    if(values != null) {
        for(int i = 0; i < values.length; i++) {
            out.println("<tr><td><b><fontcolor=\"blue\">" +
                        parameter + "</font></b></td>" +
                        "<td>" + values[i] + "</td></tr>");
        }
    }
}
```

A Login Servlet

```
import java.util.HashMap;
import java.util.Map;
import javax.servlet.*;
import javax.servlet.http.*;
public class LoginServlet extends HttpServlet {
    protected Map users = new HashMap();
    /**
     * Initializes the servlet with some usernames/passwords
     */
    public void init() {
        users.put("test", "TEST");
    }
}
```

A Login Servlet (cont.)

- Since we want to handle both GET and POST requests, just use the same boilerplate code for `doGet()` and `doPost()`, calling the same `processRequest()` method.

```
protected void doGet (
    HttpServletRequest request,
    HttpServletResponse response)
throws ServletException, java.io.IOException {
    processRequest(request, response);
}
```

A Login Servlet (cont.)

```
protected void processRequest(
    HttpServletRequest request,
    HttpServletResponse response)
    throws ServletException, java.io.IOException {
    String userid = request.getParameter("userid");
    String password = request.getParameter("password");
    if(userid != null && userid.length() != 0) {
        userid = userid.trim();
    }
    if(password != null && password.length() != 0) {
        password = password.trim();
    }
}
```

A Login Servlet (cont.)

- processRequest() continued...

```
if(userid != null && password != null) {  
    String realpassword = (String)users.get(userid);  
    if(realpassword != null &&  
        realpassword.equals(password)) {  
        showPage(response, "Login Success!");  
    } else {  
        showPage(response, "Login Failure! Username or"  
            + "password is incorrect");  
    }  
} else {  
    showPage(response, "Login Failure! You must supply a"  
        + "username and password");  
}  
}
```

A Login Servlet (cont.)

- Showing the result

```
protected void showPage(HttpServletRequest response, String
message)
throws ServletException, java.io.IOException {
response.setContentType("text/html");
java.io.PrintWriter out = response.getWriter();
out.println("<html>");
out.println("<head>");
out.println("<title>Login Servlet Result</title>");
out.println("</head>");
out.println("<body>");
out.println("<h2>" + message + "</h2>"); out.println("</body>");
out.println("</html>");
out.close();
}
```

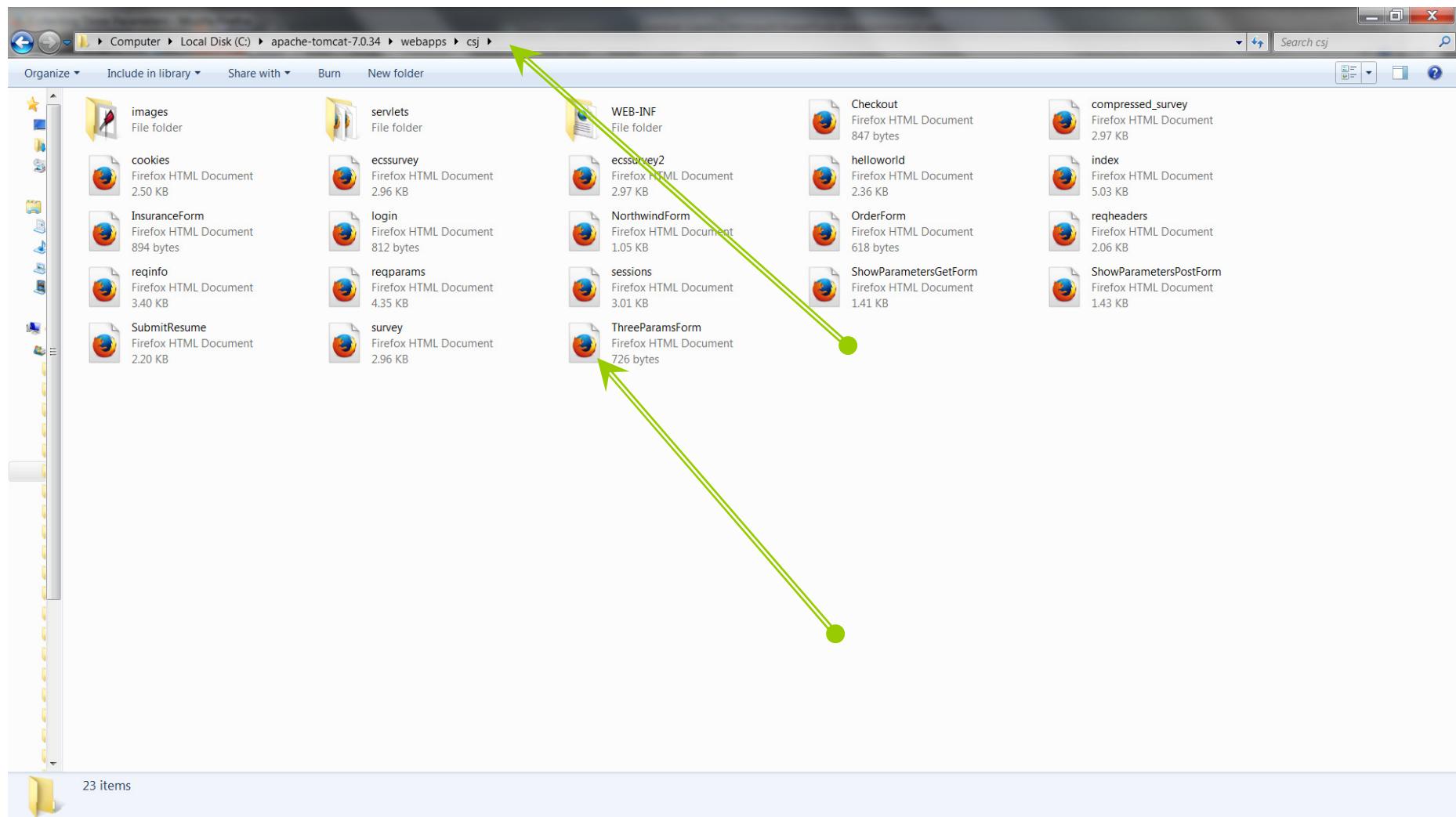
Deploy the Login Servlet

- Compile the code
- Move the `LoginServlet.class` file to
`<app_home>/WEB-INF/classes/`
- Move the `login.html` file to `<app_home>`
- Point your browser to:
 - `http://localhost/csj/login.html`
- Try to log in!

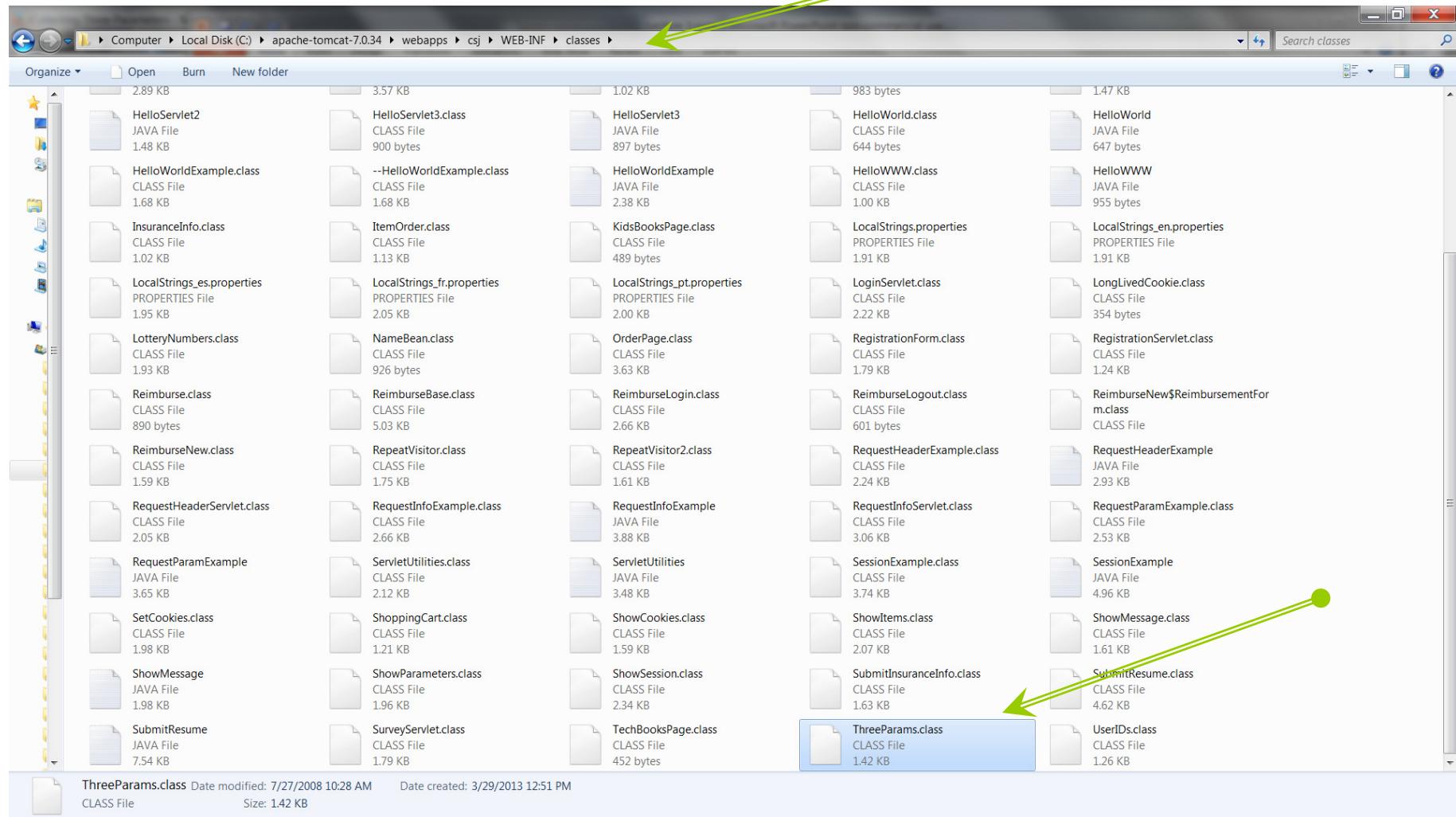
ThreeParams Servlet

- How to read 3 parameters passed from html page?
 - ThreeParams.java
 - ThreeParamsForm.html
 - <http://localhost/csj/ThreeParamsForm.htm>

ThreeParams Servlet



ThreeParams Servlet



ThreeParams Servlet

Collecting Three Parameters - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Minimize

http://localhost/csj/ThreeParamsForm.htm

Most Visited Getting Started Latest Headlines

Google Search Share Sidewiki Bookmarks Translate AutoFill

Sign in

Collecting Three Parameters

Collecting Three Parameters

First Parameter:

Second Parameter:

Third Parameter:

Done

ThreeParams Servlet

The screenshot shows a Mozilla Firefox browser window. The title bar reads "Reading Three Request Parameters - Mozilla Firefox". The address bar contains the URL "http://localhost/csj/ThreeParams?param1=Chicago¶m2=Pizza¶m3=Style". The main content area displays the heading "Reading Three Request Parameters" followed by a bulleted list of request parameters:

- param1: Chicago
- param2: Pizza
- param3: Style

At the bottom left of the browser window, there is a "Done" button.

<http://localhost/csj>ShowParametersGetForm.htm>

- How to read N parameters passed from html page?

- <http://localhost/csj>ShowParametersGetForm.htm>
- ShowParameters.java

<http://localhost/csj>ShowParametersGetForm.htm>

- We will look at two examples
 - Using Get
 - Using Post
- When using Get in the form ...
 - Watch how the parameters are passed next to the URL

<http://localhost/csj>ShowParametersGetForm.htm>

A Sample FORM using GET - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost/csj>ShowParametersGetForm.htm

Google

Most Visited Getting Started Latest Headlines

Sign in

A Sample FORM using GET

Item Number: 123

Description: car

Price Each: \$22000

First Name: Tom

Last Name: Rinka

Middle Initial: J

Adams St.
Chicago, IL

Shipping Address:

Credit Card:

Visa

MasterCard

American Express

Discover

Java SmartCard

Credit Card Number:

Repeat Credit Card Number:

Submit Order

Done

<http://localhost/csj>ShowParametersGetForm.htm>

The screenshot shows a Mozilla Firefox browser window with the title "Reading All Request Parameters". The address bar contains the URL "http://localhost/csj>ShowParameters?itemNum=123&description=car&price=%2422000&firstName=Tom&lastName=Rinka&initial=J&address=Adams+St.+Chicago,+IL". The main content area displays a table titled "Reading All Request Parameters" with the following data:

Parameter Name	Parameter Value(s)
cardNum	• 0987654321 • 0987654321
cardType	Visa
price	\$22000
initial	J
itemNum	123
address	Adams St. Chicago, IL
firstName	Tom
description	car
lastName	Rinka

At the bottom left of the browser window, there is a "Done" button.

<http://localhost/csj>ShowParametersPostForm.htm>

- When using Post in the form ...
 - Watch how the parameters are passed separate from the URL
 - Note that doPost called doGet to avoid repeating the same code

<http://localhost/csj>ShowParametersPostForm.htm>

A Sample FORM using POST - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost/csj>ShowParametersPostForm.htm

Most Visited Getting Started Latest Headlines

Google Search Share Bookmarks Translate AutoFill

A Sample FORM using POST

A Sample FORM using POST

Item Number:

Description:

Price Each:

First Name:

Last Name:

Middle Initial:

Shipping Address:

Credit Card:

Visa

MasterCard

American Express

Discover

Java SmartCard

Credit Card Number:

Repeat Credit Card Number:

Done

<http://localhost/csj>ShowParametersPostForm.htm>

The screenshot shows a Mozilla Firefox browser window with the title bar "Reading All Request Parameters - Mozilla Firefox". The address bar contains the URL "http://localhost/csj>ShowParameters". The main content area displays a table titled "Reading All Request Parameters" with the following data:

Parameter Name	Parameter Value(s)
cardNum	• 0987654321 • 0987654321
cardType	MasterCard
price	\$22000
initial	J
itemNum	123
address	Chicago, IL
description	car
firstName	Tom
lastName	Rinka

At the bottom left of the browser window, there is a "Done" button.

A more advanced Form, A Survey

<http://localhost/csj/survey.html>

Survey Submission - Mozilla Firefox

File Edit View Bookmarks Tools Help

Survey Submission

localhost/csj/survey.html Secure Search

Software Engineering Survey

Name Information

First Name

Middle Initial

Last Name

Email

Address Information

Address

City

State Illinois ZIP

Company Information

Company Name

Company Type Private Public Government

Languages Used Java C++ C Visual Basic C# Perl

Comments

Secure Search McAfee

After submitting the form...

Survey Submission - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost/csj/survey.html

Most Visited Getting Started Latest Headlines

Google Search Share Sidewiki Bookmarks Translate AutoFill

Survey Submission

Software Engineering Survey

Name Information

First Name: Tom

Middle Initial: J

Last Name: Rinka

Email: tjr@company999.com

Address Information

Address: Adams St

City: chicago

State: Illinois ZIP 60616

Company Information

Company Name: Tech

Company Type: Private Public Government

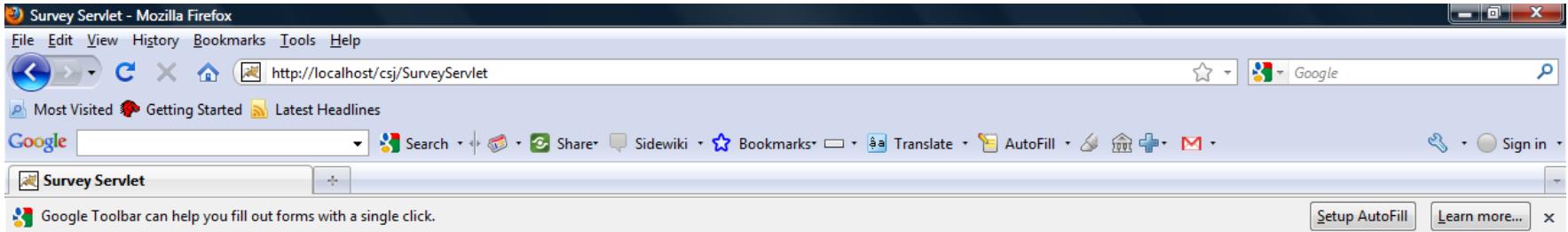
Languages Used: Java C++ C Visual Basic C# Perl

Comments: Learning ~~Servleets~~

Submit

Done

After submitting the form...



The screenshot shows a Mozilla Firefox browser window with the title "Survey Servlet - Mozilla Firefox". The address bar displays the URL "http://localhost/csj/SurveyServlet". The main content area shows a table of submitted parameters:

Parameter Name	Parameter Value
state	IL
zip	60616
java	on
co_type	public
city	chicago
first	Tom
comments	Learning Servlets
company	Tech
email	tjr@company999.com
cpp	on
page_name	login
csharp	on
address	Adams St.
last	Rinka
middle	J

At the bottom of the browser window, there is a "Done" button.

Using Default Values When Parameters are missing or Malformed

- What the servlet should do when dealing with bad input?
 1. Use Default Values
 2. Redisplay the form for the user to type again

Example: Resume Submission

- <http://localhost/csj/SubmitResume.htm>
- SubmitResume.java Servlet

In case you never used DL, DD, DT HTML tags ...
visit this website to get demo

The screenshot shows a Mozilla Firefox window displaying the w3schools.com website. The title bar reads "HTML dl tag - Mozilla Firefox". The main content area shows the "HTML <dl> Tag" page. On the left, there's a sidebar with "HTML Reference" links including "HTML by Alphabet", "HTML by Function", "HTML Global Attributes", "HTML Events", "HTML Canvas", "HTML Audio/Video", "HTML Doctypes", "HTML Colornames", "HTML Colorpicker", "HTML Colormixer", "HTML Character Sets", "HTML ASCII", "HTML ISO-8859-1", "HTML Symbols", "HTML URL Encode", "HTML Lang Codes", "HTTP Messages", "HTTP Methods", and "Keyboard Shortcuts". Below this is a "HTML Tags" section. The main content area has a heading "HTML <dl> Tag" and a sub-section "Example" with the following code:

```
<dl>
  <dt>Coffee</dt>
  <dd>Black hot drink</dd>
  <dt>Milk</dt>
  <dd>White cold drink</dd>
</dl>
```

Below the code is a green button labeled "Try it yourself »". To the right of the main content are several sidebar boxes: "SHARE THIS PAGE" with social sharing icons, "WEB HOSTING" with links to Best Web Hosting, eUK Web Hosting, UK Reseller Hosting, and Domain, Hosting & Email; "WEB BUILDING" with links to XML Editor - Free Trial!, FREE Website BUILDER, FREE Website Creator, and Best Website Templates; and "STATISTICS" with links to Browser Statistics, OS Statistics, and Display Statistics. At the bottom of the page, it says "Waiting for tags.mathtag.com...". The browser status bar at the bottom right shows "Secure Search" and a McAfee logo.

In case you never used DL, DD, DT HTML tags ... visit this website to get demo

The screenshot shows a Mozilla Firefox window with the title "Tryit Editor v1.8 - Mozilla Firefox". The address bar displays "www.w3schools.com/tags/tryit.asp?filename=tryhtml_dd_test". The main content area shows the following HTML code:

```
<!DOCTYPE html>
<html>
<body>

<dl>
  <dt>Coffee</dt>
  <dd>Black hot drink</dd>
  <dt>Milk</dt>
  <dd>White cold drink</dd>
</dl>

</body>
</html>
```

Below the code, a message from Google Play Music reads: "Unlimited Music Made Easy! play.google.com/music Discover and play millions of songs with Google Play Music All Access. Google". A "Ad muted. Undo" message with a link to "ads settings" is also present.

The "Source Code:" panel contains the provided HTML code. The "Result:" panel shows the output:

Coffee	Black hot drink
Milk	White cold drink

At the bottom, a note says "Edit the code above and click "Submit Code" to see the result."

<http://localhost/csj/SubmitResume.htm>

Free Resume Posting - Mozilla Firefox
File Edit View History Bookmarks Tools Help
C X http://localhost/csj/SubmitResume.htm Google
Most Visited Getting Started Latest Headlines
Google Search Share Sidewiki Bookmarks Translate AutoFill
Free Resume Posting not-computer-jobs.com

To use our *free* resume-posting service, simply fill out the brief summary of your skills below. Use "Preview" to check the results, then press "Submit" once it is ready. Your mini-resume will appear online within 24 hours.

First, give some general information about the look of your resume:

Heading font: default
Heading text size: 32
Body font: default
Body text size: 18
Foreground color: BLACK
Background color: WHITE

Next, give some general information about yourself:

Name: Joe Rao
Current or most recent title: SE
Email address: jrao@seproject.com
Programming Languages: Java, C#

Finally, enter a brief summary of your skills and experience: (use <P> to separate paragraphs. Other HTML markup is also permitted.)
Building web based  application

[Preview] [Submit]

Done

Once you Click Preview

The screenshot shows a Mozilla Firefox browser window with the title bar "Resume for Joe Rao - Mozilla Firefox". The address bar contains the URL "http://localhost/csj/SubmitResume". The page content displays a resume for "Joe Rao" with the title "SE" and email "jrao@seproject.com". Below this, under the heading "Programming Languages", there is a bulleted list: "• Java" and "• C#".

Joe Rao
SE
jrao@seproject.com

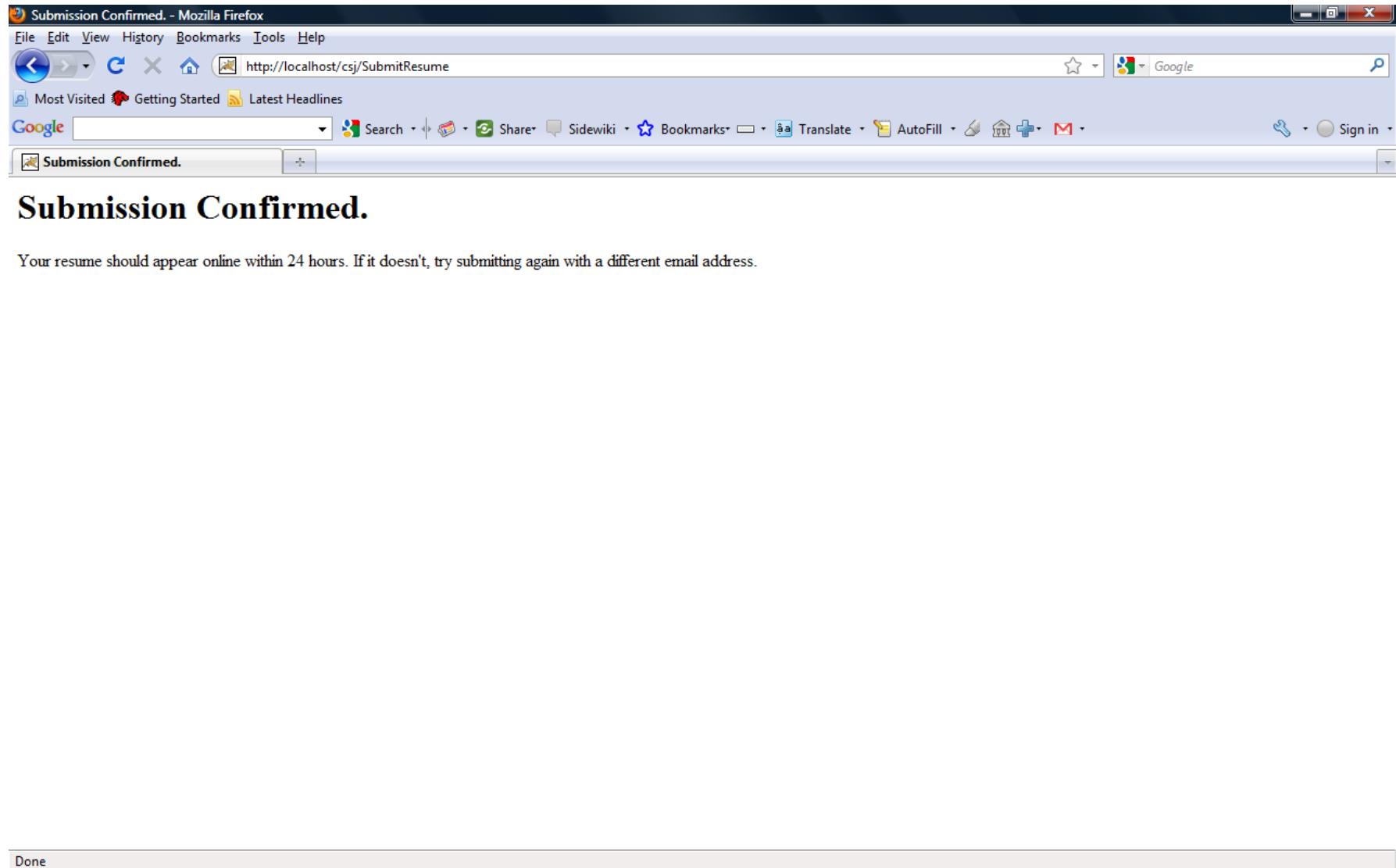
Programming Languages

- Java
- C#

Skills and Experience

Building web based OO application

Once you Click Submit

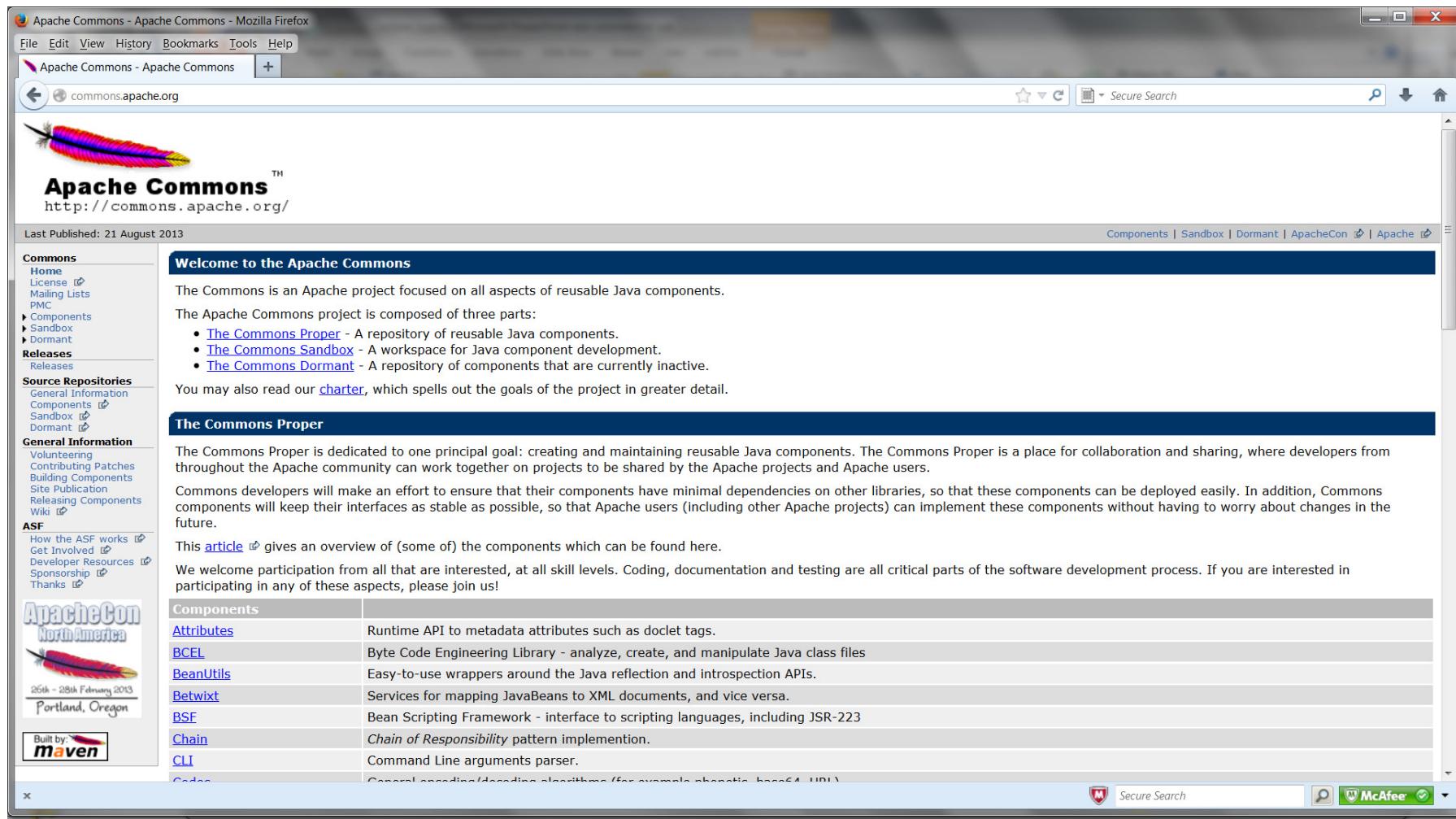


The screenshot shows a Mozilla Firefox browser window with the title bar "Submission Confirmed. - Mozilla Firefox". The address bar displays the URL "http://localhost/csj/SubmitResume". The main content area of the browser shows the text "Submission Confirmed." in large, bold, black font. Below this, a smaller text message reads: "Your resume should appear online within 24 hours. If it doesn't, try submitting again with a different email address." At the bottom left of the browser window, there is a "Done" button.

Using BeanUtils to access Request Parameters

- How to automatically populate a bean based on the request parameters?
 - Apache/tomcat has common packages that could be downloaded and automatically populate a bean according to incoming request parameters
- You need to download the following jars from <http://commons.apache.org/>
 - [BeanUtils](#)
 - [Collections](#)
 - [Logging](#)

Using BeanUtils to access Request Parameters



The screenshot shows a Mozilla Firefox browser window displaying the Apache Commons homepage. The address bar shows 'commons.apache.org'. The page features a large feather logo and the text 'Apache Commons™' with the URL 'http://commons.apache.org/'. A sidebar on the left contains links for Commons Home, License, Mailing Lists, PMC, Components, Sandbox, Dormant, Releases, and Source Repositories. Below these are sections for General Information, Volunteering, Contributing Patches, Building Components, Site Publication, Releasing Components, and Wiki. The main content area has a dark blue header 'Welcome to the Apache Commons'. It explains that the Commons is an Apache project focused on reusable Java components and is composed of three parts: The Commons Proper, The Commons Sandbox, and The Commons Dormant. It also links to the Charter for more details. A section titled 'The Commons Proper' discusses its goal of creating and maintaining reusable Java components for collaboration and sharing. It mentions minimal dependencies, easy deployment, and stable interfaces. An article link provides an overview of the components. The footer contains a table titled 'Components' listing various sub-projects like Attributes, BCEL, BeanUtils, Betwixt, BSF, Chain, CLI, and Codec.

Last Published: 21 August 2013

Components | Sandbox | Dormant | ApacheCon | Apache

Welcome to the Apache Commons

The Commons is an Apache project focused on all aspects of reusable Java components.

The Apache Commons project is composed of three parts:

- [The Commons Proper](#) - A repository of reusable Java components.
- [The Commons Sandbox](#) - A workspace for Java component development.
- [The Commons Dormant](#) - A repository of components that are currently inactive.

You may also read our [charter](#), which spells out the goals of the project in greater detail.

The Commons Proper

The Commons Proper is dedicated to one principal goal: creating and maintaining reusable Java components. The Commons Proper is a place for collaboration and sharing, where developers from throughout the Apache community can work together on projects to be shared by the Apache projects and Apache users.

Commons developers will make an effort to ensure that their components have minimal dependencies on other libraries, so that these components can be deployed easily. In addition, Commons components will keep their interfaces as stable as possible, so that Apache users (including other Apache projects) can implement these components without having to worry about changes in the future.

This [article](#) gives an overview of (some of) the components which can be found here.

We welcome participation from all that are interested, at all skill levels. Coding, documentation and testing are all critical parts of the software development process. If you are interested in participating in any of these aspects, please join us!

Components	
Attributes	Runtime API to metadata attributes such as doclet tags.
BCEL	Byte Code Engineering Library - analyze, create, and manipulate Java class files
BeanUtils	Easy-to-use wrappers around the Java reflection and introspection APIs.
Betwixt	Services for mapping JavaBeans to XML documents, and vice versa.
BSF	Bean Scripting Framework - interface to scripting languages, including JSR-223
Chain	<i>Chain of Responsibility</i> pattern implementation.
CLI	Command Line arguments parser.
Codec	General encoding/decoding algorithms (for example <code>phoenix</code> , <code>base64</code> , <code>URL</code>)

Using BeanUtils to access Request Parameters

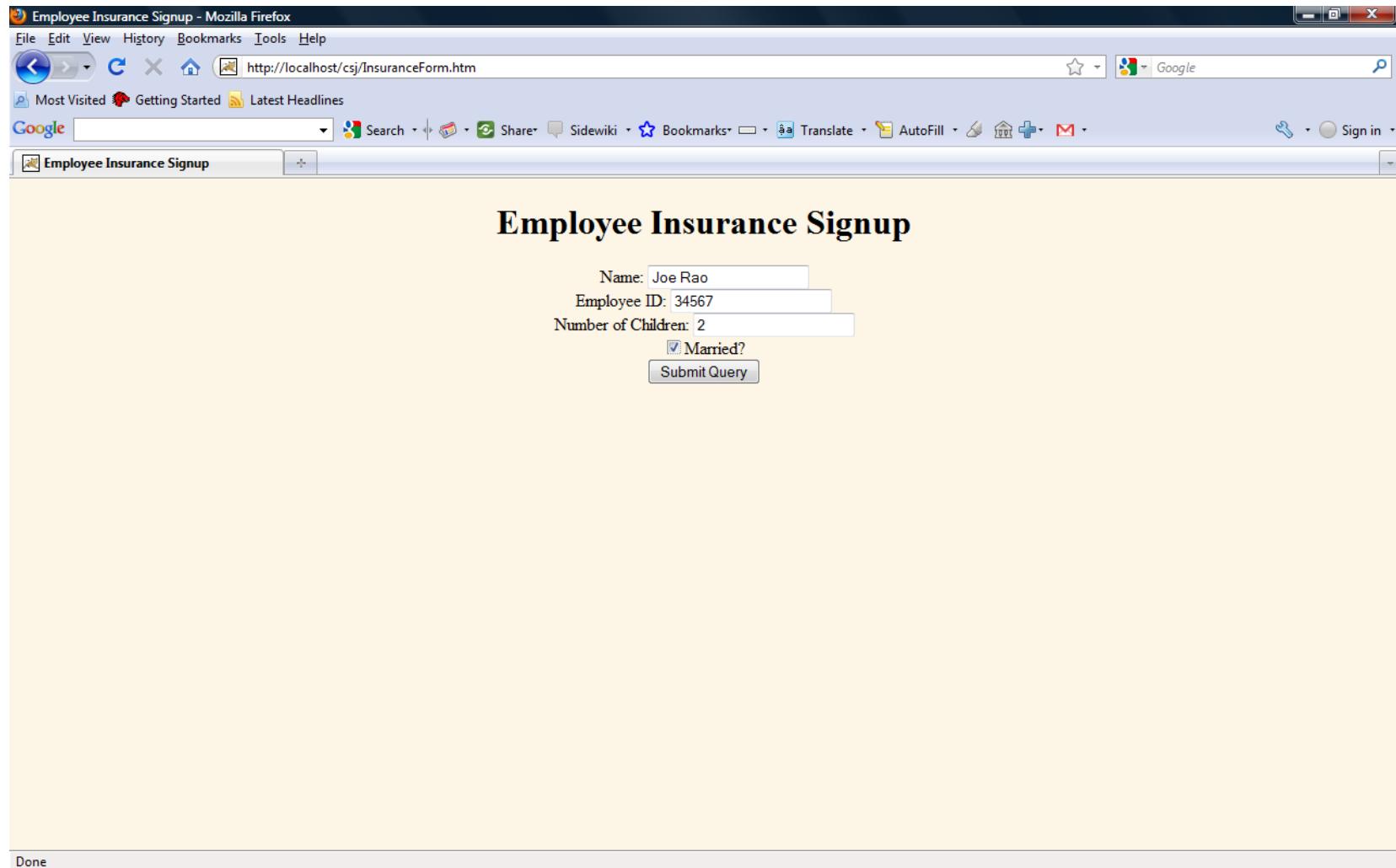
- Once you unzip these files you need to copy the following jar files to:
 - C:\apache-tomcat-7.0.34\lib
 - commons-beanutils-1.8.3
 - commons-beanutils-bean-collections-1.8.3
 - commons-beanutils-core-1.8.3
 - commons-collections-3.2.1
 - commons-logging-1.1.1

Example: BeanUtils to access Request Parameters

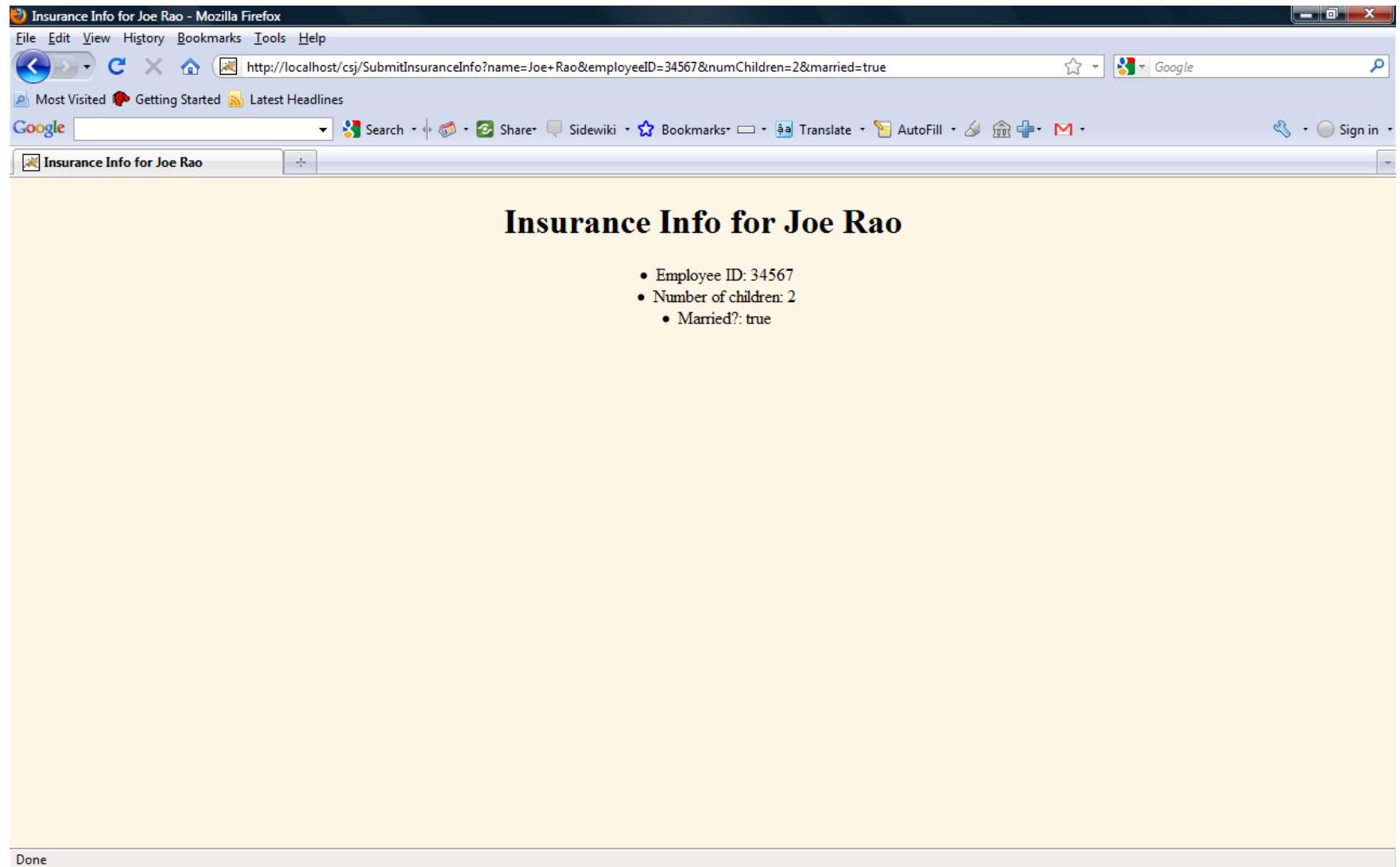
- The following example has the following files:
 - InsuranceForm.htm
 - BeanUtilities.java
 - Has a method BeanUtilities.populateBean that lets you fill in The required information in a single method call.
 - BeanUtilities.populateBean calls BeanUtils.populate from tomcat/apache Common package
 - InsuranceInfo.java (This is your simple Bean that will be populated BeanUtils.populate)
 - SubmitInsuranceInfo.java

Using BeanUtils to access Request Parameters

<http://localhost/csj/InsuranceForm.htm>



Using BeanUtils to access Request Parameters



HTTP Request Object

GET / HTTP/1.1

Accept: image/gif, image/x-bitmap, image/jpeg, */*

Accept-Language: en-us

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0;
Q312461)

Host: localhost

Connection: Keep-Alive

HTTP Request Information

- The class `ServletRequest` contains methods that provide information about the current request
 - `int getContentType()`
 - The length, in bytes, of the request body. Returns -1 if length isn't known.
 - `String getProtocol()`
 - The name and version of the protocol the client uses. In the form: *protocol* / *majorVersion* . *minorVersion*
e. g., `HTTP/ 1.1`.

HTTP Request Information (cont.)

- `String getRemoteAddr ()`
 - The IP address of the client.
- `String getRemoteHost ()`
 - The fully qualified host name of the client.
- `boolean isSecure ()`
 - Whether this request was made using a secure channel, such as `https` .

HTTP Request Information (cont.)

- In class `HttpServletRequest`
 - `String getAuthType()`
 - The name of the authentication scheme used, e. g., **BASIC** or **SSL** or null
 - `String getContextPath()`
 - The portion of the request URI that indicates the context of the request.
 - `String getMethod()`
 - The name of the HTTP request method e. g., **GET**, **POST**, or **PUT**.

HTTP Request Information (cont.)

- `String getPathInfo()`
 - Any extra path information associated with the URL the client sent.
- `String getPathTranslated()`
 - Any extra path information after the servlet name but before the query string, and translates it to a real path.
- `String getQueryString()`
 - The query string that is appended to the request URL after the path.

HTTP Request Information (cont.)

- `String getRemoteUser ()`
 - The login of the user making this request, if the user has been authenticated, or `null` .
- `String getRequestURL ()`
 - The part of this request's URL from the protocol name up to the query string in the first line of the HTTP request.
- `String getServletPath ()`
 - The part of this request's URL that calls the servlet.

Request Information Servlet

```
protected void processRequest(HttpServletRequest request, HttpServletResponse
    response)
    throws ServletException, java.io.IOException {
    String [][] requestInfo = {
        {"Content Length", String.valueOf(request.getContentLength()) },
        {"Content Type", request.getContentType() },
        {"Method", request.getMethod() },
        {"Authorization Type", request.getAuthType() },
        {"Remote User", request.getRemoteUser() },
        {"Remote Address", request.getRemoteAddr() },
        {"Scheme", request.getScheme() },
        {"Is Secure", String.valueOf(request.isSecure()) },
        {"Protocol", request.getProtocol() },
        {"Context Path", request.getContextPath() },
        {"Path Info", request.getPathInfo() },
        {"Path Translated", request.getPathTranslated() },
        {"Query String", request.getQueryString() },
        {"Servlet Path", request.getServletPath() },
        {"Request URI", request.getRequestURI() }
    };
}
```

Request Headers

- Most of the above Request information is actually stored in a Request Header.
- Each header is one line of the HTTP request, in a **key:value** format.

Common Request Headers

- **User-Agent**
 - Identifies the browser type and version, e. g., Mozilla/ 4.72 [en] (X11; U; Linux 2.2.14- 5.0 i686)
- **Host**
 - Indicates the host given in the request URL, e. g.,
 - Required in HTTP 1.1
- **Accept**
 - Indicates MIME types browser can handle, e. g., image/gif, image/jpeg, image/png, */*
- **Accept-Encoding**
 - Indicates encodings browser can handle, e. g., gzip or compress

Common Request Headers (cont.)

- **Connection**
 - `keep-alive` : browser can handle persistent connection.
- **Authorization**
 - User identification for password-protected pages.
- **Cookie**
 - Cookies previously sent to the client by the same server.
- **If-Modified-Since**
 - Send the page only if it has been changed after specified date.
- **Referer**
 - URL of the referring Web page.

Request Headers API

- `String getHeader(String name)`
 - Returns the value of the specified request header as a `String`.
- `Enumeration getHeaderNames()`
 - Returns an enumeration of all the header names this request contains.
- `Enumeration getHeaders(String name)`
 - Returns all the values of the specified request header as an `Enumeration` of `String` objects.

Retrieving Request Headers

```
Enumeration headers = request.getHeaderNames();
while(headers.hasMoreElements()) {
    String headerName = (String)headers.nextElement();
    String value = request.getHeader(headerName);
    ...
}
```