

JSPs

Lecture 8

cs193i – Internet Technologies
Summer 2004
Stanford University

Administrative Stuff

- HW#3 due today
- HW#4 due August 11
- All local SCPD students must come to campus for final exam

Why JSPs?

- Goal: Create dynamic web content (HTML, XML, ...) for a Web Application
- Goal: Make it easier/cleaner to mix static HTML parts with dynamic Java servlet code
- JSP specification ver. 2.0
Java Servlet specification ver. 2.4

JSP/ASP/PHP vs CGI/Servlets

- CGI & Servlets -- Mostly Code with some HTML via print & out.println
- JSP/ASP/PHP -- Mostly HTML, with code snippets thrown in
 - No explicit recompile
 - Great for small problems
 - Easier to program
 - Not for large computations

What is JSP?

- Mostly HTML page, with extension .jsp
- Include JSP tags to enable dynamic content creation
- Translation: JSP → Servlet class
- Compiled at Request time
(first request, a little slow)
- Execution: Request → JSP Servlet's service method

What is a JSP?



```
<html>
  <body>
    <jsp:useBean.../>
    <jsp:getProperty.../>
    <jsp:getProperty.../>
  </body>
</html>
```

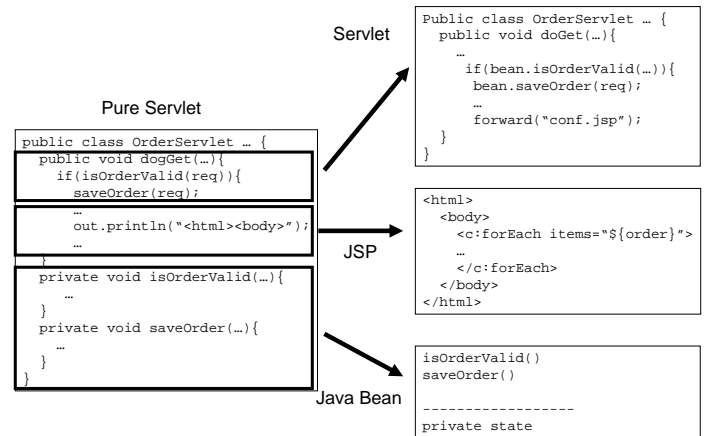
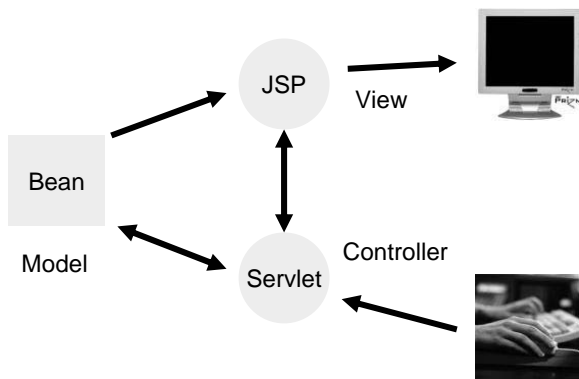
Advantages

- Code -- Computation
- HTML -- Presentation
- Separation of Roles
 - Developers
 - Content Authors/Graphic Designers/Web Masters
 - Supposed to be cheaper... but not really...

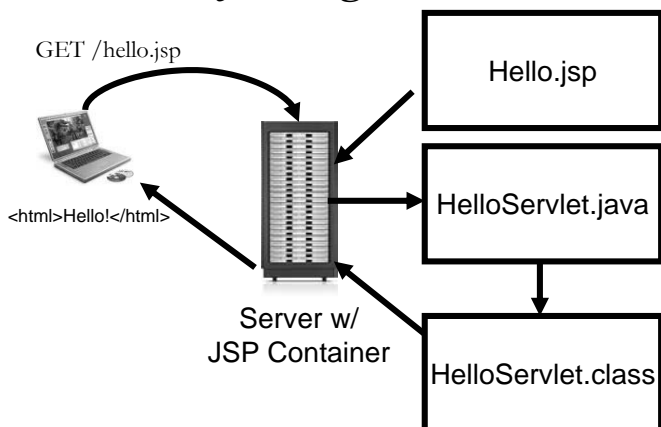
Model-View-Controller

- A Design Pattern
- Controller -- receives user interface input, updates data model
- Model -- represents state of the world (e.g. shopping cart)
- View -- looks at model and generates an appropriate user interface to present the data and allow for further input

Model-View-Controller



JSP Big Picture



A JSP File

```

<%@ page language="java" contentType="text/html" %> -- JSP element
<html> -- template text
<body bgcolor="white"> -- template text

<jsp:useBean -- JSP element
id="userInfo"
class="com.ora.jsp.beans.userInfo.UserInfoBean">
<jsp:setProperty name="userInfo" property="*" />
</jsp:useBean>

The following information was saved: -- template text
<ul> -- template text
<li>User Name: -- JSP element

<jsp:getProperty name="userInfo" -- JSP element
property="userName" />

<li>Email Address: -- template text

<jsp:getProperty name="userInfo" -- JSP element
property="emailAddr" />

</ul> -- template text
</body> -- template text
</html> -- template text
    
```

<%@ Directive %>

```
<%@ page contentType="text/html" %>
<%@ taglib prefix="c" uri=http://java.sun.com/jstl/core %>
<html>
  <head>
    <title>JSP is Easy</title>
  </head>
  <body bgcolor="white">
    <h1>JSP is as easy as ...</h1>
    <%-- Calculate the sum of 1 + 2 + 3 dynamically --%>
    1 + 2 + 3 = <c:out value="\${1 + 2 + 3}" />
  </body>
</html>
```

<%-- JSPComment -->

```
<%@ page contentType="text/html" %>
<%@ taglib prefix="c" uri=http://java.sun.com/jstl/core %>
<html>
  <head>
    <title>JSP is Easy</title>
  </head>
  <body bgcolor="white">
    <h1>JSP is as easy as ...</h1>
    <%-- Calculate the sum of 1 + 2 + 3 dynamically --%>
    1 + 2 + 3 = <c:out value="\${1 + 2 + 3}" />
  </body>
</html>
```

Template Text

```
<%@ page contentType="text/html" %>
<%@ taglib prefix="c" uri=http://java.sun.com/jstl/core %>
<html>
  <head>
    <title>JSP is Easy</title>
  </head>
  <body bgcolor="white">
    <h1>JSP is as easy as ...</h1>
    <%-- Calculate the sum of 1 + 2 + 3 dynamically --%>
    1 + 2 + 3 = <c:out value="\${1 + 2 + 3}" />
  </body>
</html>
```

<%= expr >

- Java expression whose output is spliced into HTML
- <%= userInfo.getUserName() %>
<%= 1 + 1 %>
<%= new java.util.Date() %>
- Translated to
out.println(userInfo.getUserName()) ...

WARNING: Old School JSP! The new way is introduced after break.... you may use either, but Old School JSP is used sparingly nowadays

<% code %>

- Scriptlet: Add a whole block of code to JSP...
passed through to JSP's service method

```
<%@ page language="java" contentType="text/html" %>
<%@ page import="java.util.*" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
<%
  // Create an ArrayList with test data
  ArrayList list = new ArrayList( );
  Map author1 = new HashMap( );
  author1.put("name", "John Irving");
  author1.put("id", new Integer(1));
  list.add(author1);
  Map author2 = new HashMap( );
  author2.put("name", "William Gibson");
  author2.put("id", new Integer(2));
  list.add(author2);
  Map author3 = new HashMap( );
  author3.put("name", "Douglas Adams");
  author3.put("id", new Integer(3));
  list.add(author3);
  pageContext.setAttribute("authors", list);
%>
<html>
  <head>
    <title>Search result: Authors</title>
  </head>
  <body bgcolor="white">
    Here are all authors matching your search criteria:
    <table>
      <th>Name</th>
      <th>Id</th>
      <c:forEach items="\${authors}" var="current">
```

```

<%@ page language="java" contentType="text/html" %>
<html>
<head>
<title>Browser Check</title>
</head>
<body bgcolor="white">

<%
String userAgent = request.getHeader("User-Agent");
if (userAgent.indexOf("MSIE") != -1) {
%>
You're using Internet Explorer.
<% } else if (userAgent.indexOf("Mozilla") != -1) { %>
You're probably using Netscape.
<% } else { %>
You're using a browser I don't know about.
<% } %>
</body>
</html>

```

<%! decl >

- Turned into an instance variable for the servlet
- What did I tell you about instance variables & multithreaded servlets?
- Serious Race Conditions Here!

```

<%@ page language="java" contentType="text/html" %>
<%!
int globalCounter = 0;
%>
<html>
<head>
<title>A page with a counter</title>
</head>
<body bgcolor="white">
This page has been visited: <%= ++globalCounter %> times.
<p>
<%
int localCounter = 0;
%>
This counter never increases its value: <%= ++localCounter %>
</body>
</html>

```

Declarations have serious multithreading issues!

```

<%@ page language="java" contentType="text/html" %>
<%!
int globalCounter = 0;
%>
<html>
<head>
<title>A page with a counter</title>
</head>
<body bgcolor="white">
This page has been visited: <%= ++globalCounter %> times.
<p>
<%
int localCounter = 0;
%>
This counter never increases its value: <%= ++localCounter %>
</body>
</html>

```

Not saved between requests...!

```

<%@ page language="java" contentType="text/html" %>
<%@ page import="java.util.Date" %>
<%!
int globalCounter = 0;
java.util.Date startDate;

public void jspInit() {
startDate = new java.util.Date();
}

public void jspDestroy() {
ServletContext context = getServletConfig().getServletContext();
context.log("test.jsp was visited " + globalCounter +
" times between " + startDate + " and " + (new Date()));
}
%>
<html>
<head>
<title>A page with a counter</title>
</head>
<body bgcolor="white">
This page has been visited: <%= ++globalCounter %> times
since <%= startDate %>.
</body>
</html>

```

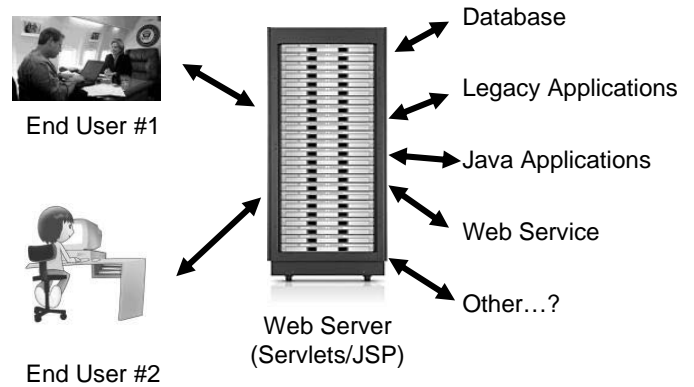
No real need anymore, since we don't use instance variables!

<jsp:include ...>

- <jsp:include page="trailer.html" flush="true" />

Five Minute Break

Big Picture – Web Apps

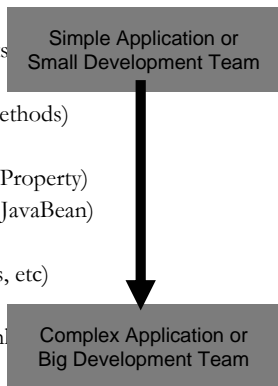


Invoking Dynamic Code (from JSPs)

- Call Java Code Directly (Expressions, Declarations, Scriptlets)
- Call Java Code Indirectly (Separate Utility Classes, JSP calls methods)
- Use Beans (jsp:useBean, jsp:getProperty, jsp:setProperty)
- Use MVC architecture (servlet, JSP, JavaBean)
- Use JSP expression Language (shorthand to access bean properties, etc)
- Use custom tags (Develop tag handler classes; use xml-like custom tags)

Invoking Dynamic Code (from JSPs)

- Call Java Code Directly (Expressions, Declarations, Scriptlets)
- Call Java Code Indirectly (Separate Utility Classes, JSP calls methods)
- Use Beans (jsp:useBean, jsp:getProperty, jsp:setProperty)
- Use MVC architecture (servlet, JSP, JavaBean)
- Use JSP expression Language (shorthand to access bean properties, etc)
- Use custom tags (Develop tag handler classes; use xml-like custom tags)



Servlets and JSPs

- Core Servlets and JavaServer Pages, 2nd Edition, Volumes 1 & 2. Marty Hall & Larry Brown

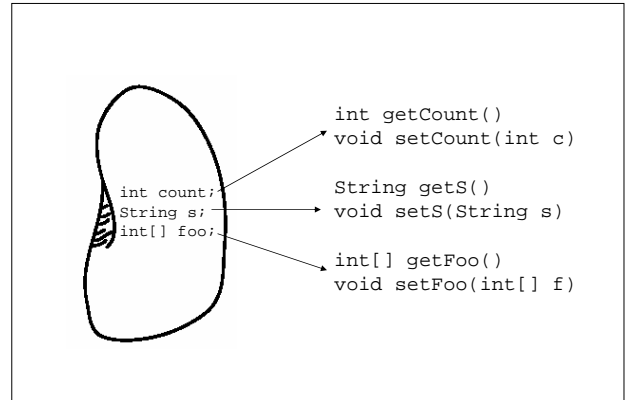
Java Beans

- Purpose: Store Data
- Simple Object, requires no argument constructor
- Properties accessible via get & set methods

Java Beans

- For a "foo" property, a java bean will respond to
 - Type getFoo()
 - void setFoo(Type foo)
- For a boolean "bar" property, a java bean will respond to
 - boolean isBar()
 - void setBar(boolean bar)

Java Bean



```
// MagicBean.java
/* A simple bean that contains a single
 * "magic" string.
 */
public class MagicBean {
    private String magic;
    public MagicBean(String string) {
        magic = string;
    }
    public MagicBean() {
        magic = "Woo Hoo"; // default magic string
    }
    public String getMagic() {
        return(magic);
    }
    public void setMagic(String magic) {
        this.magic = magic;
    }
}
```

Java Beans

- `<jsp:useBean id="myBean" class="com.foo.MyBean" scope="request"/>`
- `<jsp:getProperty name="myBean" property="lastChanged" />`
- `<jsp:setProperty name="myBean" property="lastChanged" value="<%= new Date()%>" />`
- Example
 - `<jsp:usebean id="bean" class="MagicBean" />`
 - `<jsp:getProperty name="bean" property="magic" />`

```
<!-- bean.jsp -->
<hr>
<h3>Bean JSP</h3>

<p>Have all sorts of elaborate, tasteful HTML ("presentation") surrounding the
data we pull off the bean.

<p>Behold -- I bring forth the magic property from the Magic Bean...

<!-- bring in the bean under the name "bean" -->
<jsp:usebean id="bean" class="MagicBean" />

<table border=1>
<tr>
<td bgcolor=green><font size=+2>Woo</font> Hoo</td>
<td bgcolor=pink>
<font size=+3>
<td bgcolor=pink>
<font size=+3>
    <!-- the following effectively does bean.getMagic() -->
    <jsp:getProperty name="bean" property="magic" />
</font>
</td>
<td bgcolor=yellow>Woo <font size=+2>Hoo</font></td>
</tr>
</table>

<!-- pull in content from another page at request time with a relative URL ref
to another page -->
<jsp:include page="trailer.html" flush="true" />
```

```

public class HelloBean extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<html>");
        out.println("<head>");
        String title = "Hello Bean";
        out.println("<title>" + title + "</title>");
        out.println("</head>");
        out.println("<body bgcolor=white>");
        out.println("<h1>" + title + "</h1>");
        out.println("<p>Let's see what Mr. JSP has to
            contribute...");
        request.setAttribute("foo", "Binky");
        MagicBean bean = new MagicBean("Peanut butter sandwiches!");
        request.setAttribute("bean", bean);
        RequestDispatcher rd = getServletContext().getRequestDispatcher("/bean.jsp");
        rd.include(request, response);
        rd.include(request, response);
        out.println("<hr>");
        out.println("</body>");
        out.println("</html>");
    }
    // Override doPost() -- just have it call doGet()
    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException
    {
        doGet(request, response);
    }
}

```

JSP Tags

- Found in JSP Pages
- Look like HTML Tags
 - <blah attr=val>Foo</blah>
- Single Tag Format
 - <blah attr=val/>

<%-- JSPComment --%>

```

<%@ page contentType="text/html" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
<html>
  <head>
    <title>JSP is Easy</title>
  </head>
  <body bgcolor="white">
    <h1>JSP is as easy as ...</h1>
    <%-- Calculate the sum of 1 + 2 + 3 dynamically --%>
    1 + 2 + 3 = <c:out value="${1 + 2 + 3}" />
  </body>
</html>

```

<%@ Directive %>

```

<%@ page contentType="text/html" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
<html>
  <head>
    <title>JSP is Easy</title>
  </head>
  <body bgcolor="white">
    <h1>JSP is as easy as ...</h1>
    <%-- Calculate the sum of 1 + 2 + 3 dynamically --%>
    1 + 2 + 3 = <c:out value="${1 + 2 + 3}" />
  </body>
</html>

```

Page Directive

- <%@ page import="package.class" %>
- <%@ page import="java.util.*" %>
- <%@ page contentType="text/html" %>
 - <% response.setContentType("text/html"); %>

Include Directive

- <%@ include file="Relative URL">
- Included at Translation time
- May contain JSP code such as response header settings, field definitions, etc... that affect the main page

<jsp:include ...>

- Include Files at Request Time
- <jsp:include page="news/Item1.html"/>
- page attribute must point to a page that is HTML or a page that produces HTML (via JSP, CGI, etc)...

Scripting Element Tags

- <%= expr %>
- <%! decl %>
- <% code %>

Action Elements

- Standard Actions
- JSTL (tag library) Actions
- Custom Actions

Standard Actions

- <jsp:useBean>
Makes a JavaBeans component available in a page
- <jsp:getProperty>
Gets a property value from a JavaBeans component and adds it to the response
- <jsp:setProperty>
Set a JavaBeans property value
- <jsp:include>
Includes the response from a servlet or JSP page during the request processing phase

Standard Actions

- <jsp:forward>
Forwards the processing of a request to servlet or JSP page
- <jsp:param>
Adds a parameter value to a request handed off to another servlet or JSP page using <jsp:include> or <jsp: forward>
- <jsp:plugin>
Generates HTML that contains the appropriate client browser-dependent elements (OBJECT or EMBED) needed to execute an applet with the Java Plug-in software

Custom Actions (Tag Libraries)

- Can Define Your own!
- Description
 - Define
 - Install
 - Declare
 - Use
- Details in JavaServer Pages 2nd ed found on Safari Techbooks

JSTL Tags

```
<%@ taglib prefix="c"
uri="http://java.sun.com/jstl/core" %>
...
<c:out value="${1 + 2 + 3}" />
```

JSP Standard Tag Library

- Built on custom tag infrastructure

```
<%@ page contentType="text/html" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
<html>
<head>
  <title>JSP is Easy</title>
</head>
<body bgcolor="white">
  <h1>JSP is as easy as ...</h1>
  1 + 2 + 3 = <c:out value="${1 + 2 + 3}" />
</body>
</html>
```

JSTL Control Tags

```
<%@ page contentType="text/html" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>

<c:if test="${2>0}">
  It's true that (2>0)!
</c:if>

<c:forEach items="${paramValues.food}" var="current">
  <c:out value="${current}" />&nbsp;&nbsp;&nbsp;
</c:forEach>
```

Expression Language Motivation

- Limitations of MVC
 - `<%= ... %>`, `<% ... %>`, `<%! ... %>`
 - `jsp:useBean`, `jsp:getProperty`
 - verbose
 - clumsy scripting & expression elements to do more complicated Java things
- With Expression Language
 - `${expression}`
 - Short and readable and fluid, like JavaScript

Advantages of Expression Language (EL)

- Simple & Concise
- Flexible (use in conjunction with tag libraries & custom tags)
- Robust against Error

Basic Arithmetic

- `${1.2 + 2.3} => 3.5` `<%= 1.2 + 2.3 %>`
- `${3/0} => Infinity`
- `\${1} => ${1}`
- `${10 mod 4} => 2`

Basic Comparisons

- `#{4.0 >= 3} => true`
- `#{4.0 ge 3} => true` Not in Java
- `#{100.0 == 100} => true`
- `#{(10*10) ne 100} => false` Not in Java
- `#{'hip' > 'hit'} => false` Not in Java
- `#{'a' < 'b'} => true` Not in Java

Implicit Objects

- `#{param.foo} => booyah`
- `#{param["foo"]} => booyah`
- `\#{param["foo"]} => #{param["foo"]}`
- `#{header["host"]} => localhost:8080`
- `#{header["accept"]} => /*/*`
- `#{header["user-agent"]} => Mozilla/5.0
(Macintosh; U; PPC Mac OS X; en-us)
AppleWebKit/124 (KHTML, like Gecko) Safari/125`

Functions

- Implemented by Tag Libraries (Tag Libraries, implemented as Static Methods)
- `#{param["foo"]}`
=> JSP 2.0
- `#{my:reverse(param["foo"])}`
=> 0.2 PSJ
- `#{my:reverse(my:reverse(param["foo"]))}`
=> JSP 2.0
- `#{my:countVowels(param["foo"])}`
=> 0

Conditionals

```
<TD ALIGN="RIGHT" BGCOLOR="{(oranges.total < 0) ? "RED" : "WHITE"}">
```

```
package coreservlets;

public class SalesBean {
    private double q1, q2, q3, q4;

    public SalesBean(double q1Sales, double q2Sales, double q3Sales, double
        q4Sales) {
        ...
    }

    public double getQ1() {return(q1);}

    public double getQ2() {return(q2);}
    ...
}
}
```

```
package coreservlets;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Conditionals extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        SalesBean apples = new SalesBean(150.25, -75.25, 22.25, -33.57);
        SalesBean oranges = new SalesBean(-220.25, -49.57, 138.25, 12.25);

        request.setAttribute("apples", apples);
        request.setAttribute("oranges", oranges);
        RequestDispatcher dispatcher =
            request.getRequestDispatcher("/el/conditionals.jsp");
        dispatcher.forward(request, response);
    }
}
```

```

package coreservlets;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Conditionals extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        SalesBean apples = new SalesBean(150.25, -75.25, 22.25, -33.57);
        SalesBean oranges = new SalesBean(-220.25, -49.57, 138.25, 12.25);

        request.setAttribute("apples", apples);
        request.setAttribute("oranges", oranges);
        RequestDispatcher dispatcher =
            request.getRequestDispatcher("/el/conditionals.jsp");
        dispatcher.forward(request, response);
    }
}

```

```

...
<TD ALIGN="RIGHT" BGCOLOR="${(oranges.total < 0) ? "RED" : "WHITE"}">
<TD ALIGN="RIGHT" BGCOLOR="${(apples.q1 < 0) ? "RED" : "WHITE"}">
...

```

EL Examples

- `<input name="firstName" value="${customer.firstName}">`
- `<c:out value="${order.amount + 5}"/>`
- `${order.amount + 5}`
- `{order['amount'] + 5}`