Introduction to Java Server Pages

Jeff Offutt & Ye Wu

http://www.ise.gmu.edu/~offutt/

SWE 432

Design and Implementation of Software for the Web

From servlets lecture ….

Enabling Technologies - Plug-ins Scripted Pages

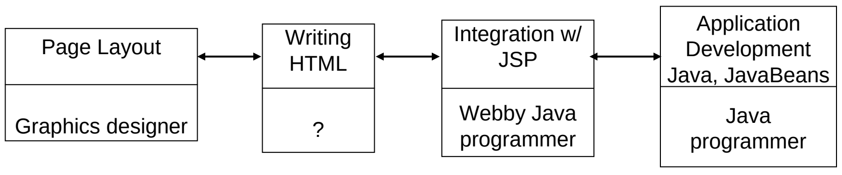
- Scripted pages look like HTML pages that happen to process business logic
- Execution is server-side, not client-side (like JavaScripts)
- They are HTML pages that access software on the server to get and process data
- Common scripted pages:
  - Allaire's Cold Fusion
  - Microsoft's Active Server Pages (ASP)
  - Java Server Pages (JSP)
- JSPs are compiled and run as servlets (very clean and efficient)
- Scripted pages are generally easier to develop, deploy, and modify

Java Server Pages (JSP)

- Java Scripts provide client-side execution ability
  - Interpreted
  - Cumbersome and error prone
  - Non-portable
- Java Servlets provide server-side execution
  - Compiled
  - Portable
  - Robust
  - Not integrated with HTML – Java creates HTML
  - Mixes static (HTML) with dynamic (business logic)

Java Server Pages (2)

- JSPs turn servlets "*inside-out*":
  - Instead of HTML in Java …
  - Java in HTML
- JSPs are translated to servlets, compiled, then executed
- This encourages separation of tasks:

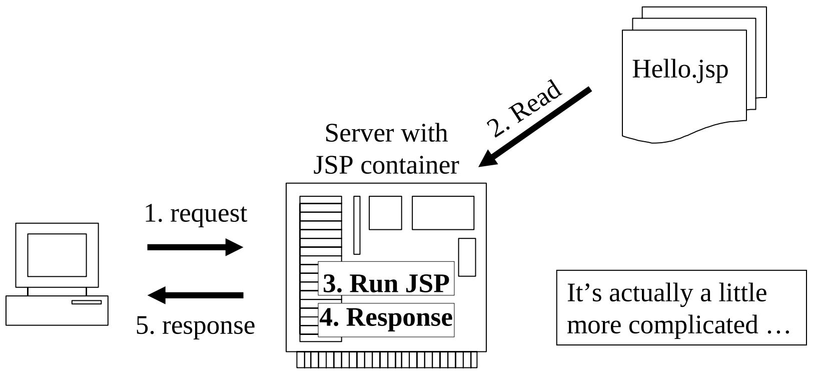


First Look at JSP Code

```
<%@page  import =  "java.util.Date"%>
<HTML>
<BODY>
<CENTER>
  <H1>Java Server Page example</H1>
  The current time is <%= new Date() %>
</CENTER>
</ BODY >
</ HTML >
```

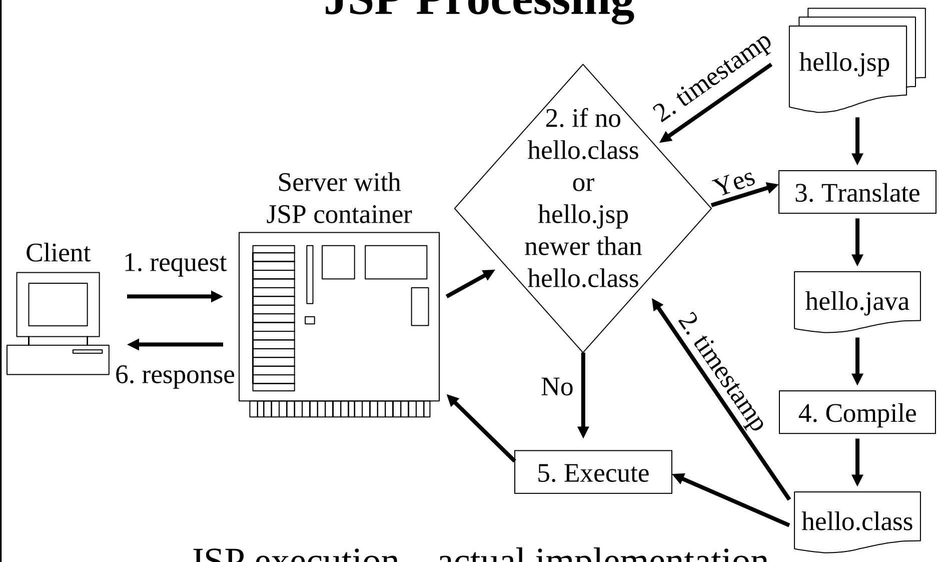
http://apps-inst.ite.gmu.edu:8080/offutt/jsp/date.jsp

JSP Processing – Simple View


Hello.jsp
2. Read
Server with JSP container
1. request
5. response
3. Run JSP
4. Response
It's actually a little more complicated …

JSP execution – mental model of JSP developer

JSP Processing



JSP execution – actual implementation

JSP Servlet Methods

- jspInit()
- jspDestroy()
- _jspService()

Can be defined in JSP declarations
Called when servlet starts and ends

– Servlet service()
Includes all Java from JSP

JSP Example

http://ise.gmu.edu/~offutt/classes/432/Examples/JSP/

Just do one, save the others

JSP Elements

JSP syntax: <%*X* … %> // *X* is one of the following:

1. @ Directive: Global information for page
   Language, import statements, etc.
2. Scripting Elements: Java code
   - ! Declarations: Class level variables and methods
   - (blank) Scriptlets: A block of Java code
     Can make external calls
   - = Expressions: Values to be printed
3. Actions: To modify runtime behavior

1) JSP Directives

Messages sent to the JSP container

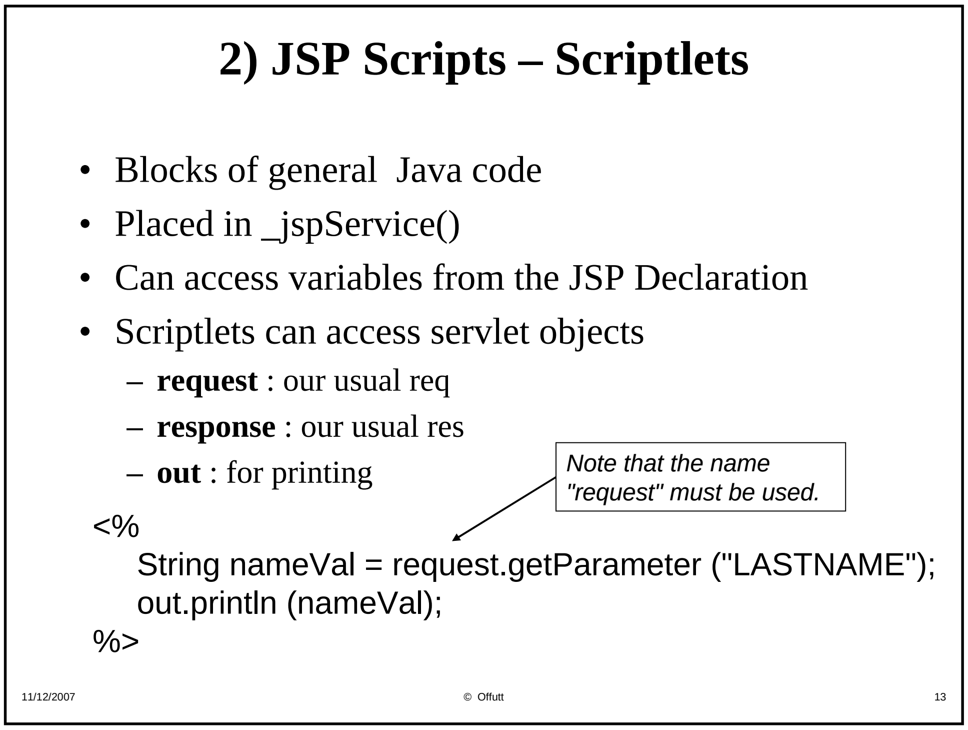
- <%@ page attribute=value … %>
  - Page attributes are listed in book
  - You will usually use the defaults
- <%@ include <filename> %>
  - File inserted into the JSP inline before JSP is compiled
- <%@ taglib uri="tagLibURI" prefix="tagPrefix" %>

2) JSP Scripts – Declarations

Java code to define class-level variables and methods

```
<%! int Sum = 0;
    private void AddToCount (int X)
    {   // To be called from a scriptlet
        Sum = Sum + X;
    }
%>
```

jspInit() and *jspDestroy()* can also be defined here to initialize and clean up state



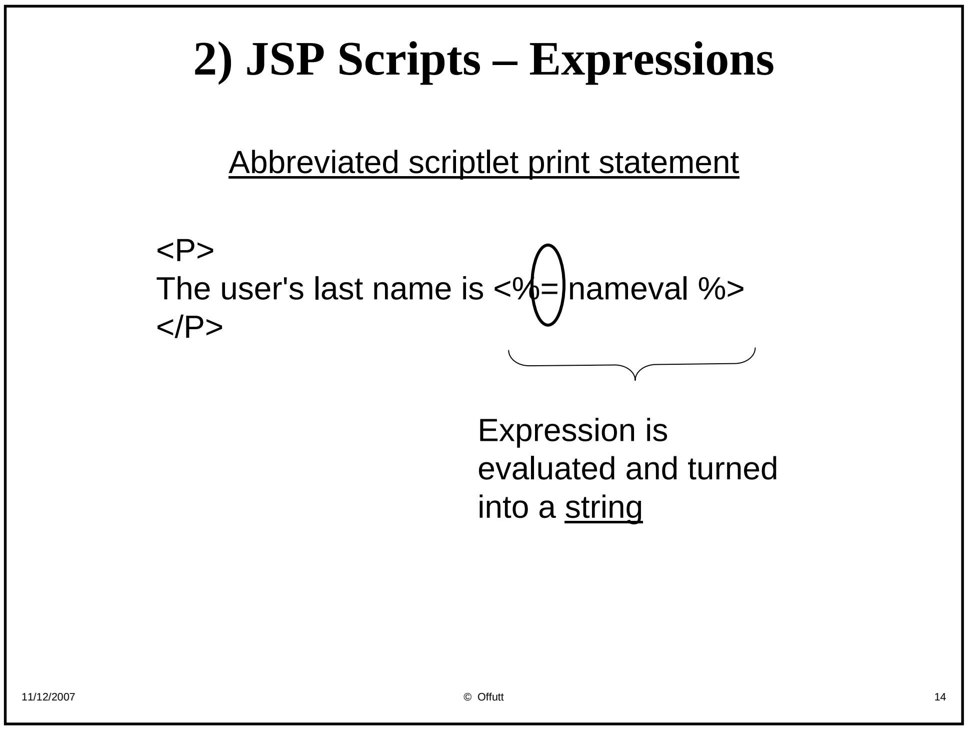
2) JSP Scripts – Scriptlets

- Blocks of general Java code
- Placed in _jspService()
- Can access variables from the JSP Declaration
- Scriptlets can access servlet objects
  - **request** : our usual req
  - **response** : our usual res
  - **out** : for printing

Note that the name "request" must be used.

```
<%
    String nameVal = request.getParameter ("LASTNAME");
    out.println (nameVal);
%>
```

11/12/2007 © Offutt 13



2) JSP Scripts – Expressions

Abbreviated scriptlet print statement

```
<P>
The user's last name is <%= nameval %>
</P>
```

Expression is evaluated and turned into a string

11/12/2007 © Offutt 14

3) JSP Actions

- Tags to change the behavior of the JSP
- <jsp:include page="myjsp.jsp" flush="true" />
  - myjsp.jsp is compiled
  - myjsp.jsp is executed
  - output from myjsp is included in the current JSP
- Action types:
  - <jsp: useBean>
  - <jsp: setProperty >
  - <jsp: getProperty >
  - <jsp: param >
  - <jsp: include >
  - <jsp: forward >
  - <jsp: plugin >

3) JSP Actions – Java Beans

- A Java Bean is a Java class with 3 characteristics:
  1. public class
  2. public constructor with no arguments
  3. public get and set methods
- Property: A special, simple data object (that is, variable)
  - getName () … <jsp:getProperty>
  - setName (String name) … <jsp:setProperty>
  - Note that a bean is not a Java language feature, but a design convention (pattern)

3) JSP Actions – Java Beans

- useBean causes a JavaBean object to be instantiated
- useBean gives a name to the new object (id=)
- useBean defines the scope
- useBean declares the location (bean details)

3) JSP Actions – Java Bean Example

- Syntax for using a bean:

```
<%@ page import="color.*" %>
<jsp:usebean id="LetterColor" scope="page"
      class="color.AlphabetCode" />
```

Converts to Java import statement, Java 4 requires all imports to be packages

ID name to use for object (AlphabetCode LetterColor = new …)

- Note that scope="application" allows Beans to be shared among different servlets – DON'T USE IT! That can lead to interactions among each other.

3) JSP Actions – Properties

- setProperty gives a value to a property in a bean
  - <jsp:setProperty name="langBean" property="language" value="Java"/>

    Equivalent to the call: langBean.setLanguage ("Java");
  - <jsp:setProperty name="langBean" property="*" />

    Sets all of the properties with values from HTML FORM

- getProperty retrieves the value of a property
  - <jsp:getProperty name="langBean" property="language"/>

    Equivalent to the call: langBean.getLanguage();

3) JSP Actions – Java Bean Summary

- Using Java Beans allows for more separation between the HTML and Java
- The Beans / Property pattern provides a very convenient standard for implementing standard Java classes
- JSP's useBean uses Java reflection to translate property names ("language") to method calls that are assumed to exist ("setLanguage()" and "getLanguage()")

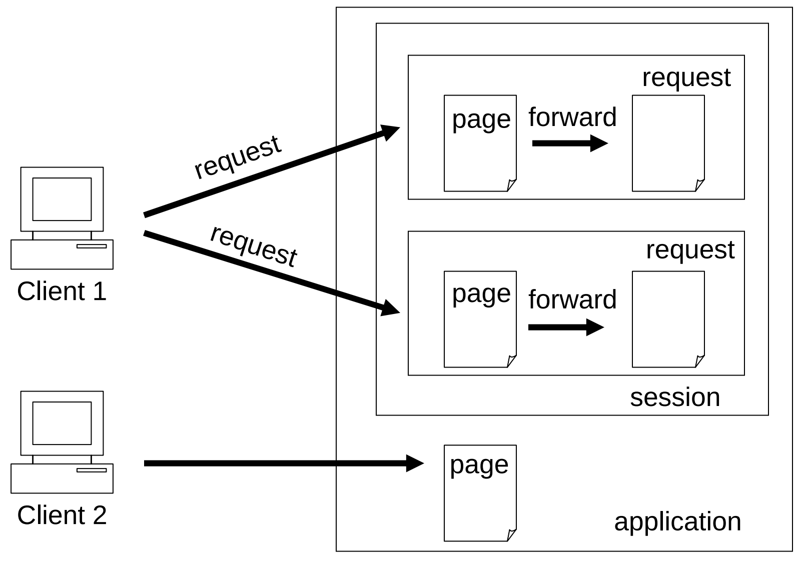
3) JSP Actions – Forwarding

- jsp:Forward allows a request to be forwarded to another JSP
- <jsp:forward page="anotherPage.jsp" />
  - When this statement is reached, execution will "jump" to the JSP anotherPage.jsp
  - Use as a front-end when we need to decide which JSP to execute based on some input data

Sharing Data Between JSP Pages, Requests, and Users

- JSPs provide different scopes for sharing data objects
  - Page : Within the same web page
  - Request: Within the same request
  - Session: Within all requests from the same session
  - Application: Within all sessions for one servlet context

Sharing Data Objects



Sharing Data Between JSP Pages, Requests, and Users (1)

- Using JSP Scriptlet
  - getParameter(); // retrieves client form data
  - request.getAttribute(), request.setAttribute();
  - session.getAttribute(), session.setAttribute();
  - context.getAttribute(), context.setAttribute();

For example:

<% session.setAttribute ("ID",request.getParameter ("ID")); %>

Predefined variable

Sharing Data Between JSP Pages, Requests, and Users (2)

- The previous approach makes the code clumsy
- Alternative approach – JavaBean:
  Use the scope attribute in the <jsp:useBean> action

```
<jsp:useBean id = “languageBean” scope=“session” class =
    “lang.LanguageBean”>
<jsp:getProperty name=“languageBean” property=“name”>
```

Sharing Sessions and Application Data

- Sharing sessions
  <jsp:useBean … scope=“session”…>

- Sharing application data
  <jsp:useBean … scope=“application”…>

Installing JSPs on Our Server

It turns out to be subtle to get JSPs to interface with Java Beans

- A JSP is converted to a Java servlet, which is then compiled by the servlet engine
- Therefore the bean has to be in a directory that is in the Java CLASSPATH of the servlet engine
- On our server, the Java servlet engine CLASSPATH includes the directory where we put servlets:

  /apps/tomcat/swe432/WEB-INF/classes/
- We all have write permissions there

Deploying JSPs on Our Server

1. Import the bean into JOUseBean.jsp:

   <%@ page import=“offutt.JOBean" %>
2. Copy the bean's .class file into the classes directory:

   cp JOBean.class /apps/tomcat/swe432/WEB-INF/classes/offutt/
3. Copy your JSP file into the JSP directory:

   cp JOUseBean.jsp /apps/tomcat/swe432/jsp/
4. Now you can run your JSP from your browser by entering the URL:

   http://apps-swe432.ite.gmu.edu:8080/swe432/jsp/JOUseBean.jsp

JSP & Java Bean Examples

http://ise.gmu.edu/~offutt/classes/432/Examples/JSP/