

Scripted Page Web Application Development (Java Server Pages)

Jeff Offutt

<http://www.cs.gmu.edu/~offutt/>

SWE 432

**Design and Implementation of
Software for the Web**

Web Applications

review

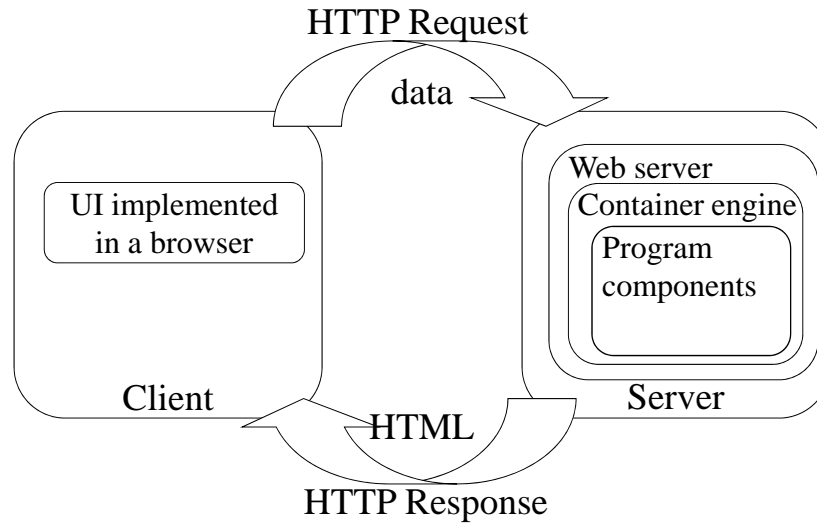
- A web application uses enabling technologies to
 1. make web site contents dynamic
 2. allow users of the system to implement business logic on the server
- Web applications allow users to affect state on the server

A web application is a program deployed on the web

An enabling technology is a mechanism that makes web pages interactive and responsive to user inputs

Server Side Processing

review



17 November 2011

© Offutt, 2011

3

What We Will Study

review

We will learn about these concepts in the context of several technologies

- PHP ✓
- Ajax ✓
- Servlets ✓
- JSPs

17 November 2011

© Offutt, 2011

4

Enabling Technologies - Plug-ins review *Scripted Pages*

- Scripted pages look like HTML pages that happen to process business logic
- Execution is on the server, not on the client
 - unlike JavaScripts
- They have HTML with program statements that get and process data
- JSPs are compiled and run as servlets
 - very clean and efficient
- PHP scripts are interpreted within the server

Enabling Technologies - Plug-ins review *Scripted Pages*

- Common scripted pages:
 - Adobe's ColdFusion
 - Microsoft's Active Server Pages (ASP)
 - PHP
 - Java Server Pages (JSP)
- Scripted pages are generally easy to develop and deploy
- They mix logic with HTML, so can be difficult to read and maintain
- Not as effective for heavy-duty engineering

Java Server Pages (JSP)

new ...

- 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 that creates HTML”

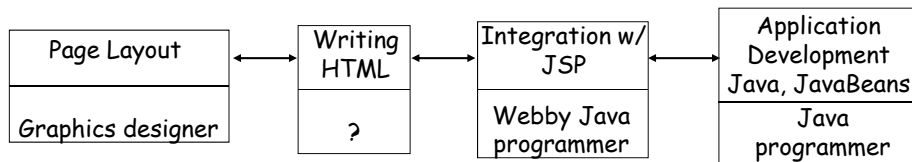
17 November 2011

© Offutt, 2011

7

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:



11/17/2011

© Offutt

8

First Look at a JSP

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

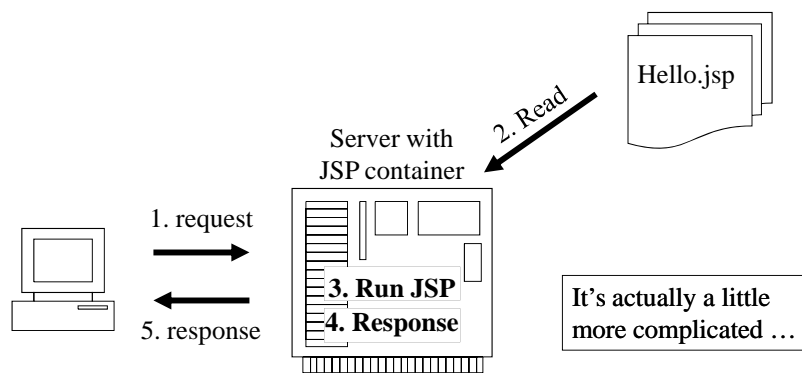
<http://cs.gmu.edu:8080/offutt/jsp/432/date.jsp>

11/17/2011

© Offutt

9

JSP Processing – Simple View

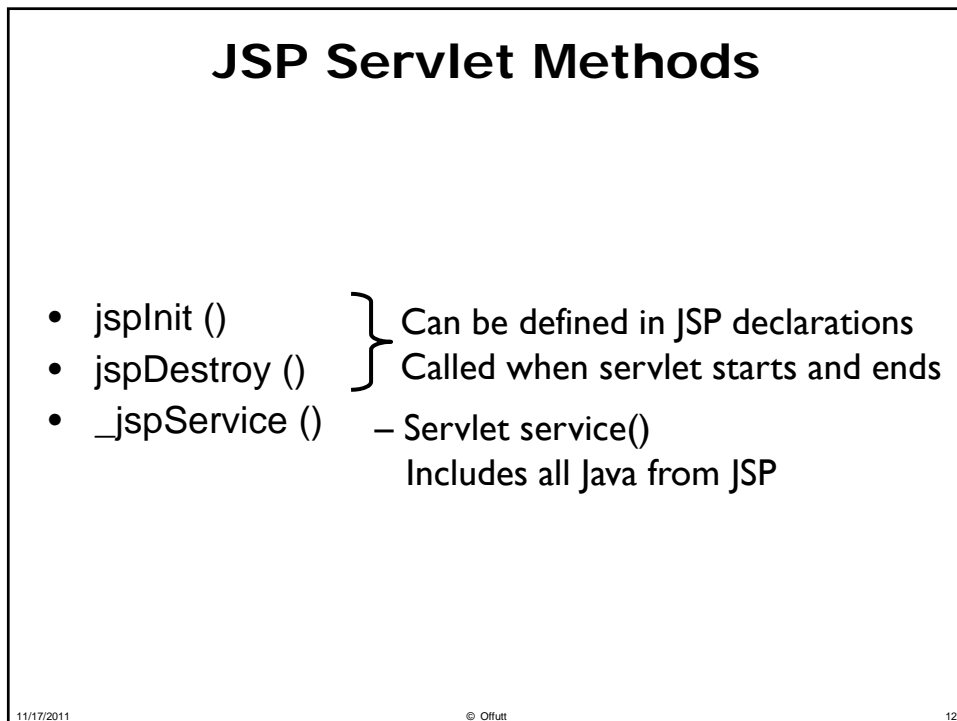
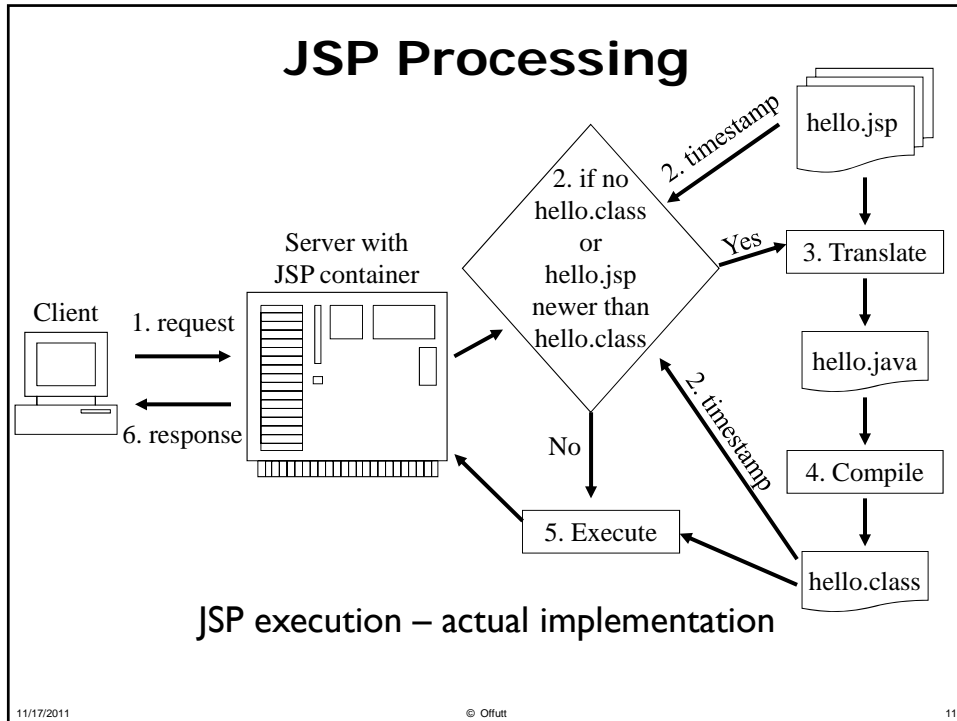


JSP execution – mental model of JSP developer

11/17/2011

© Offutt

10



JSP Example – Counter

```
<BODY>
  <!-- Set global information for the page -->
  <%@ page language="java" %>
  <!-- Declare the variable -->
  <%! int count = 0; %>
  <!-- Scriptlet - Java code -->
  <% for (int i = 0; i < 10; i++) {
    count = count+1;
  %>
  <BR/>
  The counter value is: <%= count %>
  <% } %>
</BODY>
```

11/17/2011

© Offutt

13

JSP Example

<http://cs.gmu.edu/~offutt/classes/432/examples/jsp/>

Just do one, save the others

11/17/2011

© Offutt

14

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

11/17/2011

© Offutt

15

(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" %>`

We won't use these in this class

11/17/2011

© Offutt

16

(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

11/17/2011

© Offutt

17

(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** : `HttpServletRequest` object (our usual req)
 - **response** : `HttpServletResponse` object (our usual res)
 - **out** : for printing

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

Note that the name "request" must be used

11/17/2011

© Offutt

18

(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

11/17/2011

© Offutt

19

(3) JSP Actions

- Tags to change the behavior of the JSP
- Action types:
 - <jsp: include>
 - <jsp: useBean>
 - <jsp: setProperty>
 - <jsp: getProperty>
 - <jsp: forward>
 - <jsp: param>
 - <jsp: plugin>

11/17/2011

© Offutt

20

(3) JSP Actions – Include

- `<jsp:include>` can be used to include either a static or dynamic resource
- Static : A static file is loaded inline into the JSP before translation and compiling
 - The same content is included every time
 - `<jsp:include page="copyright.html" />`
- Dynamic : A web software component is run and the results are included as part of the response
 - A dynamic include can result in different content each time
 - `<jsp:include page="myjsp.jsp" flush="true" />`
 - `myjsp.jsp` is compiled
 - `myjsp.jsp` is executed
 - Output from `myjsp` is included in the current JSP
 - Current output is flushed before `myjsp` is included

11/17/2011

© Offutt

21

(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 (called *getters* and *setters*)
- 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*)

11/17/2011

© Offutt

22

(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)

11/17/2011

© Offutt

23

(3) JSP Actions – Java Bean Example

- Syntax for using a bean:

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

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

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

Name of class

JSPs offer several useful scopes for variables ...

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

11/17/2011

© Offutt

24

(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();`
- Case of property name is very important
 - Property must begin with a lower case letter ("language")
 - Getters and setters must have the property name start with a capital letter (`setLanguage()`, `getLanguage()`)

11/17/2011

© Offutt

25

(3) JSP Actions–Java Bean Summary

- Using Java Beans increases separation between the HTML and Java
- The Beans / Property pattern provides a convenient standard for implementing standard Java classes
- JSP's `useBean` uses Java reflection to translate property names (for example, "language") to method calls that are assumed to exist ("`setLanguage()`" and "`getLanguage()`")
- The bean does not have to have an object with the name of the property, as long as it has a getter or setter

11/17/2011

© Offutt

26

(3) JSP Actions – Forwarding

- `jsp:Forward` sends a request to another JSP on the same server
 - Similar to a method call, but no return
- `<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
 - Use to authenticate users (see student info system example)

11/17/2011

© Offutt

27

Deploying JSPs on Apps Cluster

It takes effort to get JSPs to interface with Java Beans

- A JSP is translated 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 webapps server, the Java servlet engine CLASSPATH includes the directory where we put servlets:
`/data/tomcat/swe432/WEB-INF/classes/`
- Put bean .class files into your “package” directory

11/17/2011

© Offutt

28

Deploying JSPs on Hermes

1. Import the bean into your JSP :
`<%@ page import="username.*" %>`
 2. Copy the bean's .class file into the directory
`/apps/tomcat/swe432/WEB-INF/classes/ :`
`cp useBean.class /apps/tomcat/swe432/WEB-INF/classes/username/`
 3. Copy your JSP file into the directory that we set up for JSPs :
`cp useBean.jsp /apps/tomcat/swe432/jsp/`
 4. Now you can run your JSP from your browser by entering the URL:
`http://apps-swe432.vse.gmu.edu:8080/swe432/jsp/useBean.jsp`
- Look for the Java versions of your JSPs in:
`/usr/share/tomcat5/work/Catalina/apps-swe432.ite.gmu.edu/swe432/org/apache/jsp/jsp/useBean_jsp.java`

11/17/2011

© Offutt

29

JSP & Java Bean Examples

<http://cs.gmu.edu/~offutt/classes/432/examples/jsp/>

11/17/2011

© Offutt

30