Session 10

Introduction to Java Server Pages (JSP)

1

Reading & Reference

- Reading
  - Head First - Chapter 7 (pages 281-317)
- Reference
  - JSP syntax reference - java.sun.com/products/jsp/syntax/2.0/syntaxref20.html
  - JSP syntax card java.sun.com/products/jsp/syntax/2.0/card20.pdf
  - JSP Documentation www.oracle.com/technetwork/java/javaee/jsp/index.html
  - JSP 2.1 specification download.oracle.com/otn-pub/jcp/jsp-2.1-fr-eval-spec-oth-JSpec/jsp-2_1-fr-spec.pdf

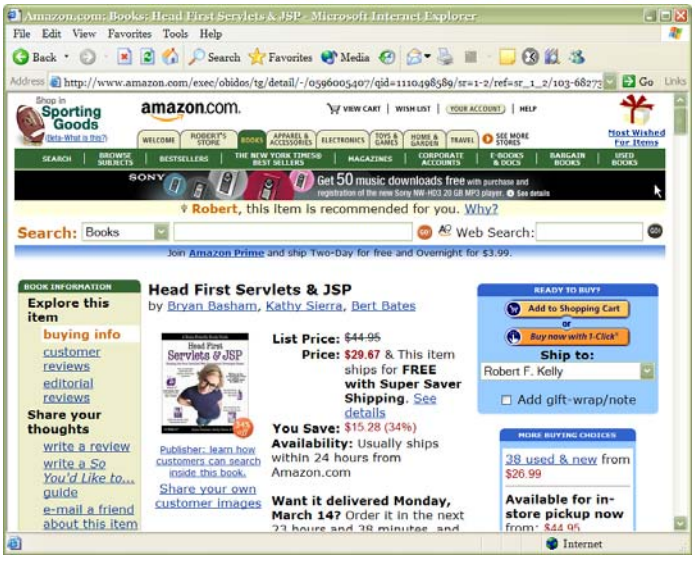
2

Lecture Objectives

- Recognize that a JSP is much easier to write (compared with a servlet) when you are creating a view
- Understand how a JSP is translated into a servlet (i.e., where your JSP code winds up in the generated servlet)
- Understand the difference between JSP (compile-time) directives and run-time actions
- Know how to obtain visibility to shared objects through JSP predefined variables

3

Why Servlets are Tough to Code



4

JavaServer Page (JSP)

- Used to rapidly create dynamically-generated Web pages
- Separates web presentation from Web page content (and allows for differing programming skills to work on a project)
- A JSP is:
  - A text-based document (filename extension of .jsp) that processes a request and constructs a response
  - Translated into a servlet

5

Servlet Vs. JSP

- JSP (helloworld.jsp)

```
<!DOCTYPE HTML PUBLIC
"- //W3C//DTD HTML 4.0
   Transitional//EN">
<html>
<head>
  <title>Hello WWW</title>
</head>
<body>
  <h1>Hello WWW</h1>
</body>
</html>
```

Which one (servlet or JSP) is easier to read - and write?

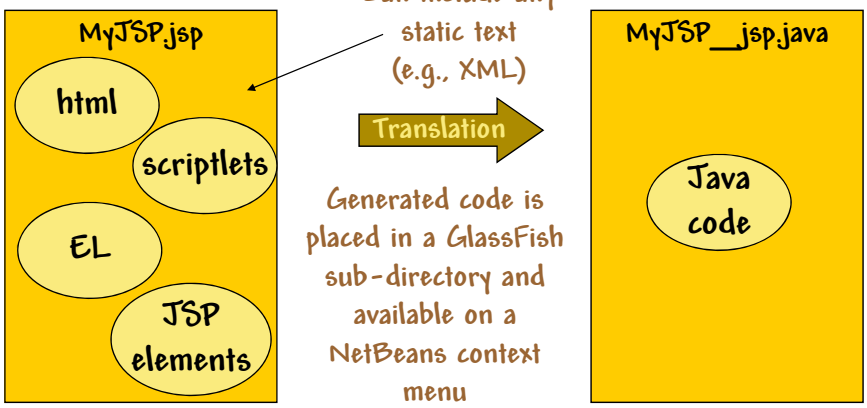
- Servlet (HelloWorld.java)

```
public class HelloWorld
   extends HttpServlet {
  public void doGet(
   HttpServletRequest request,
   HttpServletResponse response)
   throws ServletException,
  IOException {
    response.setContentType(
        "text/html");
  PrintWriter out =
        response.getWriter();
  String docType =
      "<!DOCTYPE HTML PUBLIC \"-
      //W3C//DTD HTML 4.0 " +
       "Transitional//EN\">\n";
  out.println(docType);
  out.println("<html>");
  out.println("<head><title>
        Hello WWW</title></head>");
  out.println("<body>");
  out.println(
        "<h1>Hello WWW</h1>");
  out.println("</body></html>")}}
```


6

JSP Translation

- The Web container translates the JSP into the equivalent servlet (and compiles it into a servlet class)



7

HelloWorld.jsp

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  <head><title>Hello WWW</title></head>
  <body>
    <h1>Hello WWW</h1>
  </body>
</html>
```

Doesn't this look like an html file?

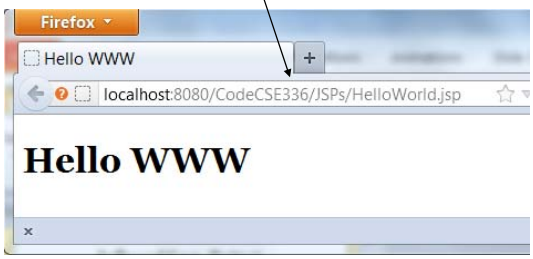
8

helloworld.jsp Output

```
<?xml version="1.0" encoding="utf-8"?> <!DOCTYPE
html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">
<html>
  <head><title>Hello WWW</title></head>
  <body>
    <h1>Hello WWW</h1>
  </body>
</html>
```

You can place the JSPs in the root or in any sub-directory (e.g., JSPs)

Notice the URL



9

Generated HelloWorld Servlet ...

```
package org.apache.jsp;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;

public final class HelloWorld_jsp extends
   org.apache.jasper.runtime.HttpJspBase
     implements
   org.apache.jasper.runtime.JspSourceDependent {

 private static final JspFactory _jspxFactory =
  JspFactory.getDefaultFactory();

  private static java.util.List<String>
  _jspx_dependants;

  private org.glassfish.jsp.api.ResourceInjector
  _jspx_resourceInjector;

  public java.util.List<String> getDependants() {
    return _jspx_dependants;
  }
```

10

... Generated HelloWorld Servlet

```
public void _jspService(HttpServletRequest request,
 HttpServletResponse response)
      throws java.io.IOException, ServletException {

  PageContext pageContext = null;
  HttpSession session = null;
  ServletContext application = null;
  ServletConfig config = null;
  JspWriter out = null;
  Object page = this;
  JspWriter _jspx_out = null;
  PageContext _jspx_page_context = null;
```

Note the predefined JSP variables

When you use the identifier "session", it refers to this variable in the generated servlet

11

... Generated HelloWorld Servlet ...

```
    try {
       response.setContentType("text/html;charset=UTF-8");
      response.setHeader("X-Powered-By", "JSP/2.2");
pageContext = _jspxFactory.getPageContext(this,
       request, response, null, true, 8192, true);
      _jspx_page_context = pageContext;
      application = pageContext.getServletContext();
      config = pageContext.getServletConfig();
      session = pageContext.getSession();
      out = pageContext.getOut();
      _jspx_out = out;
  _jspx_resourceInjector =
  (org.glassfish.jsp.api.ResourceInjector)
  application.getAttribute("com.sun.appserv.jsp.resource.
  injector");
```

12

... Generated HelloWorld Servlet ...

```
      out.write("\r\n");
      out.write("\r\n");

      out.write("<?xml version=\"1.0\" encoding=\"utf-
    8\"?>\r\n");
      out.write("<!DOCTYPE html PUBLIC \"-//W3C//DTD
    XHTML 1.0 Strict//EN\"\r\n");
      out.write("
    \"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
    strict.dtd\">\r\n");      out.write("<html>\n");
      out.write("  <head><title>Hello
    WWW</title></head>\n");
      out.write("  <body>\n");
      out.write("    <h1>Hello WWW</h1>\n");
      out.write("  </body>\n");
      out.write("</html>\n");
    }
...
```

13

... Generated HelloWorld Servlet

```
catch (Throwable t) {
      if (!(t instanceof SkipPageException)){
        out = _jspx_out;
        if (out != null && out.getBufferSize() != 0)
          out.clearBuffer();
        if (_jspx_page_context != null)
    _jspx_page_context.handlePageException(t);
      }
    } finally {
      if (_jspxFactory != null)
    _jspxFactory.releasePageContext(_jspx_page_context);
    }
  }
}
```

14

JSP Translation

- The Web container will translate your static text (e.g., html) into the corresponding servlet statements

Note: the Web container is sometimes referred to as the servlet/JSP container

JSP

```
<body>
<h1>Hello WWW</h1>
</body>
</html>
```

Translation

Servlet

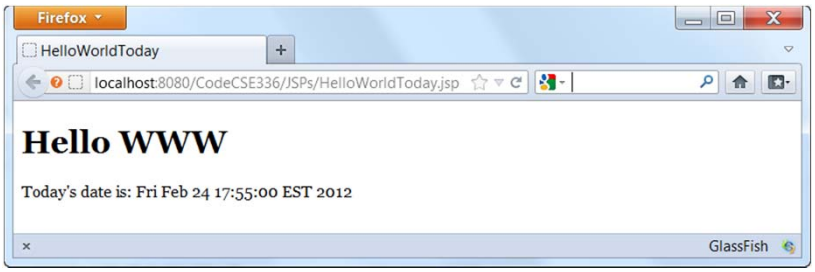
```
out.write("<body>\n");
out.write("<h1>Hello
       WWW</h1>\n");
out.write("</body>\n");
out.write("</html>\n");
```

- You can insert code (Java, EL, etc.) into your JSP - it is directly translated into the corresponding part of your servlet

15

Example - HelloWorldToday

- The JSP will display today's date



16

HelloWorldToday.jsp

```
<html>
<head>
<title>HelloWorldToday</title>
</head>
<body>
<h1>Hello WWW</h1>
<p>Today's date is:
<%= new java.util.Date() %>
</p>
</body>
</html>
```

Note the syntax of the JSP expression
<%= ... %>

This JSP expression translates to
`out.print( new java.util.Date());`
the translated statement is placed into the newly created servlet

17

JSP Expressions

- Used to insert values directly into the output
- Form: `<%= Java Expression %>`
- Expression is evaluated, converted to a string and inserted into the page
- Evaluation is performed at run time
- Can use predefined variables:
  - request - the HttpServletRequest object
  - response - the HttpServletResponse object
  - session - the HttpSession associated with the request
  - out - the PrintWriter object (actually a buffered version)

We cover this lightly since you will learn a better language for JSP expressions

18

Use of Class libraries

```
<html>
<head>
<title>HelloWorldToday</title>
</head>
<body>
<h1>Hello WWW</h1>
<p>Today's date is:
<%= new java.util.Date() %>
</p>
</body>
</html>
```

Note that we did not need an import statement since the JSP expression contained a fully qualified reference to the Date class

- If we add an import statement, its position in the translated file would not be near the expression
- Instead, we use a JSP directive

19

helloworldimport.jsp

```
<html>
<head>
<title>HelloWorldImport</title>
</head>
<body>
<h1>Hello WWW</h1>
<%@ page import="java.util.*" %>
<p>Today's date is:
<%= new Date() %>
</p>
</body>
</html>
```

This JSP directive results in an import statement being placed in the translated servlet class - at the appropriate location

Note the syntax of the JSP directive

This directive is of type page

JSP directives are performed at JSP translation time

20

Example - HelloWorldPath

- The JSP will display the path to the JSP, which it obtains from the request object

24

helloworldpath.jsp

```
<html>
<head>
<title>Hello World Path</title>
</head>
<body>
<h1>Hello WWW</h1>
<p>The path to this JSP is:
<%= request.getRequestURI()%>
</p>
</body>
</html>
```

Note the object named request. In the translated servlet, this is defined as:

```
public void _jspService(
  HttpServletRequest request,
  HttpServletResponse response)
```

Notice that there is no semicolon - because the content of the expression is really a parameter to an out.print

This JSP expression translates to

```
out.print( request.getRequestURI());
```

25

JSP Directives

- JSP tags that provide the JSP container with special instructions used to process the page
- Used to:
  - Import Java classes and packages
  - Include the contents of other files
  - Specify content type, error page, page encoding, etc.
- Syntax

```
<%@ directive_type attr1="value1" attr2="value2" %>
```

- Example

```
<%@ page import="hw.*" %>
```

- Directive types
  - include, page, and taglib

☹ The <%@ tells you this is a JSP directive

A page directive with an import attribute

26

Page Directive

- Includes page specific processing instructions that are used by the JSP container when the JSP is translated into the corresponding servlet
- Can be placed anywhere within the document
- Page directives specify:
  - Import classes
  - Parent class of the servlet
  - MIME content of the response
- Specified through the page directive attributes

```
<%@ page autoflush="true">
```

attribute name

attribute value

27

Page Directive Attributes

- import - specify the packages imported by the servlet
- contentType - equivalent to the use of the response.setContentType method in a scriptlet
- isThreadSafe - controls whether the resulting servlet will use the SingleThreadModel interface
- session - controls whether the resulting servlet participates in Http sessions (default is true)
- buffer - specifies the size of the buffer used by the `out` variable
- autoflush - specifies that the buffer is flushed when full
- errorPage/isErrorPage - specifies an error handling JSP
- extends - parent class from which the JSP servlet extends

28

Include Directive

- Inserts the content of another file into the main JSP file (before the JSP is translated into a servlet)
- Very useful for menus, headers, or footers that are repeated on many pages

You will use an include directive (in a different form) in your projects to include html for the header, footer, and sidebar

- Syntax

```
<%@ include file="includes/footer.jsp %>
```

Note: Included html should not include head or body tags (or any tag that would interfere with the HTML in the JSP file)

29

JSP Page Contents

- HTML (or XML)
- JSP constructs
  - Directives - control the overall structure of the servlet (page, include, and taglib directives)
  - Scripting elements
    - Expressions - inserted into servlet output

      `<%= expression %>`
    - Scriptlets - inserted into servlet code

      `<% code %>`
    - Declarations - inserted into body of servlet class

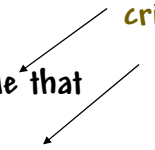
      `<%! code %>`
  - Actions - control behavior of the JSP engine

You can think of a JSP as an HTML page with escapes to insert dynamic data

30

Servlets vs. JSP

- Servlet is Java code that generates HTML output by writing to an output writer
- JSP is an HTML like document with special tags that cause data to be obtained / calculated and inserted into the HTML
- A servlet is better as a Web module that does not directly generate html
- JSP is better when it is a pure view (i.e., does not control the response)



These are the critical system design principles

31

XML Syntax for Directives

- You can use an alternate XML-compatible syntax for directives
- Example: the equivalent of

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

is

😐 Not so ugly

```
<jsp:directive.page import="java.util.*" />
```

Check the Sun JSP syntax card for XML syntax equivalents

32

Scripting Elements

- JSP tags that allow code to be embedded in a JSP page
- The code contained in these JSP scripting elements is inserted into the corresponding location of the JSP servlet
- JSP scripting elements include the following:
  - Expressions - single line of code
  - Scriptlets - blocks of code
  - Declarations - class level declarations (e.g., new instance variables/methods)
- Not considered to be a good programming practice - use servlets, beans, and custom tags for data and control
- Use of Expression Language (EL) is much better JSP practice - we cover this in the next session

33

JSP Scriptlets

- You can insert arbitrary code into the JSP
- Form: <% Java Code %>
- Standard access to automatically defined variables
- Scriptlets are sometimes used to:
  - Set Http response headers and status codes
  - Update a database
  - Provide conditional code and loop

Scriptlets might be included in a CSE336 exam to demonstrate an understanding of JSP operation

JSP Declarations

JSP declaration

```
<html>
<%! Date d = new Date(); %>
<head>
<title> <%= d %> </title>
</head><body>
<h1>Hello WWW</h1>
<%@ page import="java.util.*" %>
<p>Today's date is:
<%= d %>
</p></body></html>
```

- In the Hello World example, if we wanted to reuse the date, we would need to declare a Date variable (and import the library)

JSP Declarations

- Define methods or instance variables that get inserted into the main body of the servlet class (scriptlet declarations are inserted into the _jspService method)
- Form: <%! Java Code %>
- Example:

```
<%! int count=0; %>
```

Used for static variables, instance variables and methods

Note that this is inserted into the servlet class, not in the current location of the service method

36

Predefined Variables

- Also referred to as implicit objects, and includes
  - request - HttpRequest object
  - response - HttpResponse object
  - out - PrintWriter object
  - session - note that sessions are created automatically
  - application - the ServletContext object that can be used to store persistent data (using setAttribute and getAttribute methods)
  - config - servletConfig object
  - pageContext
  - page - not typically used by JSP authors

Be careful with the name inconsistency

37

JSP Actions

- A JSP action is executed when a JSP page is requested (i.e., run time)

Actually, this is executed when the translated servlet is called

- Categories
  - Standard (useBean, getProperty, setProperty, include, forward, etc.)
  - JSTL - if/else, loops, XML access
  - Custom - your own custom tag

38

Exception Handling

- A JSP can generate an exception
- The exception can be handled by another JSP named as the errorPage
- errorPage has access to the exception object

39

Inserting Files in JSP Documents

- Files can be included at translation time or at request time
- Translation time - Use the include directive to include a file (JSP or HTML/XML) in the main JSP document - file update requires a "recompile"

```
<%@ include file="Relative URL" %>
```

- Request time - use the `jsp:include` action to include a file

40

What Does the Container Do With Your JSP?

- Looks at directives to determine if anything should be done at translation
- Creates an HttpServlet subclass
- Writes import statements into the servlet (if there is a page directive with an import attribute)
- Writes JSP declaration code
- Builds the service method
- Merges HTML, scriptlets, and expressions into the service method

41

JSP Summary

- JSPs are text-based documents that contain
  - Static template data (e.g., html, xml)
  - JSP elements (for constructing dynamic content) denoted by <% ... %>
- JSPs access dynamic data through objects that
  - Are provided with the environment (e.g., Session object)
  - You create (e.g., Java bean)
- JSPs can employ an alternate XML-based JSP syntax
- JSPs encapsulate the design view of a page (separate from code for dynamic actions, often contained in java beans and custom tags

42

Have You Satisfied the Lecture Objectives

- Recognize that a JSP is much easier to write (compared with a servlet) when you are creating a view
- Understand how a JSP is translated into a servlet (i.e., where your JSP code winds up in the generated servlet)
- Understand the difference between JSP (compile-time) directives and run-time actions
- Know how to obtain visibility to shared objects through JSP predefined variables

43

HW 4a (Partial)

- Get started on your project form JSP
- Redirect from your servlet to the Registration JSP
- Populate one or two fields using JSP Expressions (you will replace these once you learn EL)
- Have your assignment running, so you can just place the EL statements into your JSP once we cover EL in class

44