Fundamentals of Java Enterprise Components

Session: 6

JavaServer Pages

Objectives

- Describe JavaServer Pages (JSP)
- Compare it with other technologies
- Create JavaServer Pages
- Deploy JavaServer Pages in an application
- Understand how the servlets and JSPs interact
- Describe expression language used by JavaServer Pages

Introduction

- JSPs are used to develop dynamic Web pages.
- Web pages are interactive with the ability to modify the content based on different parameters set for the Web page.
- JSP comprises a tag library which contains various mark up elements.
- JSP allows creating custom tags for the application.
- The JSP tags invoke the servlets on the server which perform the required processing and return the response to the client Web page.
- JSPs are used to create presentation logic of the application.

Need for JSP

Separates presentation logic from the business logic thus modularizing the application development process.

Platform independent and compatible with any Web server.

Following are technologies which are similar to JSPs:

- Active Server Pages (ASP) – Built to be compatible with .NET framework and IIS server
- Hypertext Preprocessor(PHP) - Open source server side technology
- JavaScript – Client side technology to generate dynamic Web pages

Uses and Benefits of JSP

Uses of JSP

- Separation of presentation logic from actual processing.
- Enables embedding Java code into the Web page through JSP scriptlets.
- JavaServerPages Tag Library(JSTL) has number of tags defined to be embedded in the HTML tags.

Benefits of JSP

- Allows delegating of tasks.
- Supports loosely coupled architecture of the application.
- Introduces well defined modules in the application.
- Allows enhancing the user interface independent of the underlying servlets.

JSPs in Web Applications

In a Web application, JSPs are stored in the Web pages directory.

On request, a JSP is converted to a servlet and executed in the container.

The execution parameters of JSP can be manipulated through JSP page directives.

JSP configuration information is specified in the deployment descriptor web.xml through a <jsp-config> tag.

Authoring JSP Pages

A JavaServer Page has three types of constructs:

- **Scripting elements** allow embedding Java code in the JSP page. This code later becomes part of the resultant servlet.
- **Directives** are those constructs which control overall structure of the servlet.
- **Actions** are those constructs which are used to control the behavior of the JSP engine or allow the developer to make use of already existing components in the application.

JSP Scripting elements

- Scripting elements are enclosed between <%...%> symbols. They are used to write Java code in JSP.
- Following are the three types of scripting elements used to embed Java code in JSP:
  - Expressions
  - Declarations
  - Scriptlets

JSP Expressions 1-3

Expressions are used to insert values to the output.

Following is the syntax of expressions:

- <%= Expression %>

Following code shows the usage of expressions. Here, the code has two expressions:

- <%= new java.util.Date()%> which retrieves the value of date
- <%= request.getRemoteAddr()%> which retrieves the value of the IP address of the host

```
<%@page contentType="text/html"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type"
content="text/html; charset=UTF-8">
        <title>Expressions Demo</title>
    </head>
    <body>
        <h4>Date: " <%= new
java.util.Date()%>"</h4>
        <p> Your IP address : "<%=
request.getRemoteAddr()%>"</p>
    </body>
</html>
```

JSP Expressions 2-3

JSP consists of many predefined objects. Following are some of the frequently used predefined objects. These predefined objects are also known as implicit objects.

- **request** – refers to the request received.
- **response** – refers to the response generated by the JSP.
- **session** – is an instance of HTTPSession which is currently active.
- **out** – object used to write the output of JSP, which is usually PrintWriter object.
- **application** – It is a ServletContext object which refers to the context of the servlet.
- **config** – refers to the configuration of the application which is a ServletConfig object.
- **pageContext** – refers to the environment of the Web page, which is a PageContext object.
- **page** – refers to the current JSP.

JSP Expressions 3-3

Attributes are properties associated with the JSP tags. These attributes can be represented as expressions also.

Following are the elements which allow expressions as their attribute values:

- **< jsp:set property …>**: This element has name and value attributes which can be set through expressions as follows:
  - <jsp:setProperty name="entry" property="itemID" value='<%=request.getParameter("itemID") %>' />
- **<jsp:include >**: This element allows you to import files or pages into the current JSP.
- **<jsp:forward >**: This element links the servlets and the JSPs. When the JSP is executed, the request is forwarded to the appropriate servlet.
- **<jsp:param>**: This element is included in the JSP or forwarded. If additional parameters need to be passed, then this element is useful.

JSP Declarations

Declaration statements are required to declare variables.

Syntax:

<%! declaration %>

The XML equivalent for declaration is:

<jsp:declaration>
code fragment
</jsp:declaration>

The given code demonstrates the usage of JSP declarations.

```
<HTML>
<HEAD>
<TITLE>JSP
Declarations</TITLE>
</HEAD>
<BODY>
<H1>JSP Declarations</H1>
<%! private int accessCount
= 0; %>
<H2>Accesses to page since
server reboot:
<%= ++accessCount %></H2>
</BODY>
</HTML>
```

JSP Scriptlets 1-4

- Scriptlets are used to embed Java code into JSPs. Scriptlets are declared using the following syntax:

  Syntax:

  `<%.....code fragment…..%>`

- Scriptlets can be used to perform tasks such as setting the headers of an HTTP response, setting the HTTP status codes and so on.
- Following demonstrates usage of scriptlets:

```
<html>
<head>
  <title>Order form</title>
</head>
<body>
  <h3>Choose your products:</h3>
   <form method="get">
 <input type="checkbox" name="product" value="iPod">iPod
```

JSP Scriptlets 2-4

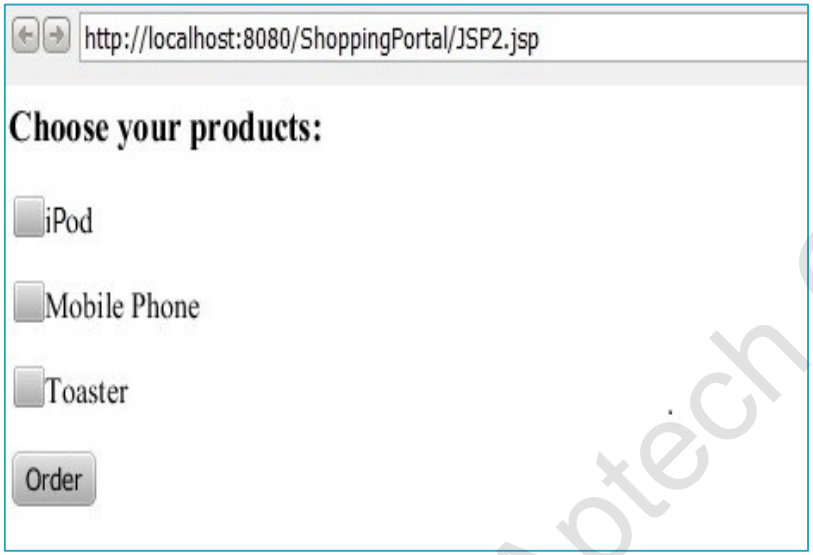
```
<p></p>
    <input type="checkbox" name="product" value="Mobile
Phone">Mobile Phone
    <p></p>
    <input type="checkbox" name="product" value="Toaster">Toaster
    <p></p>
    <input type="submit" value="Order">
  </form>

  <%
  String[] products = request.getParameterValues("product");
  if (products != null) {
  %>
 <h3>You have selected <%= products.length%>  product(s):</h3>
```

JSP Scriptlets 3-4

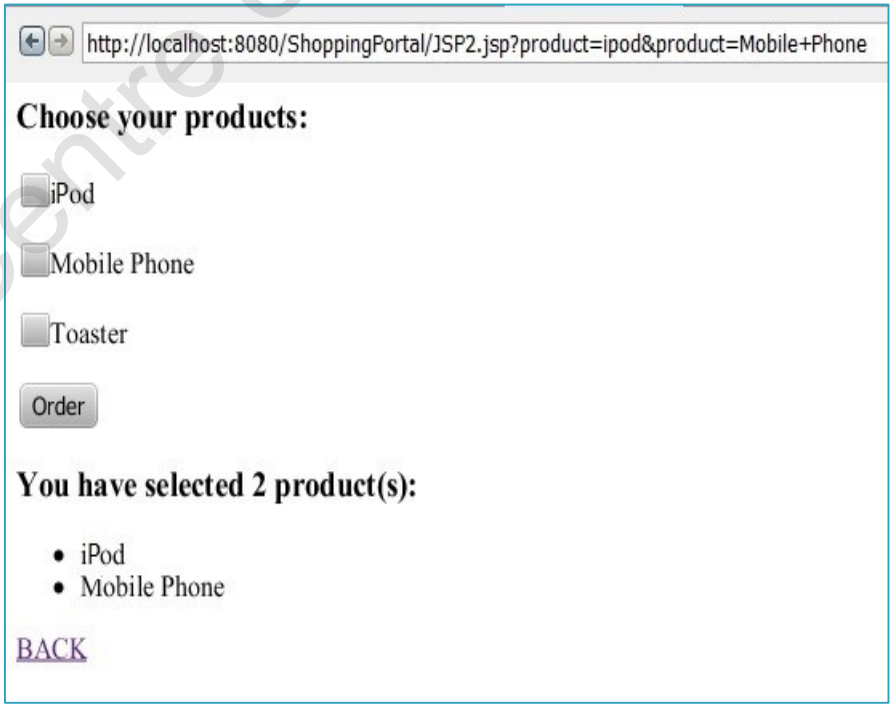
```
<ul>
 <%
      for (int i = 0; i < products.length; ++i) {
  %>
        <li><%= products[i] %></li>
  <%
      }
  %>
    </ul>
    <a href="<%= request.getRequestURI() %>">BACK</a>
  <%
  }
  %>
</body>
</html>
```

JSP Scriptlets 4-4

Given figure shows the output of the JSP code:



After selecting the products, the user clicks the **Order** button and the output will be as shown in following figure:



JSP Directives

A JSP directive defines the structure of the servlet which is generated as a result of the JSP page.

- **Syntax**:
- `<%@directive attribute = "value" %>`

A directive defining multiple attributes can be defined as follows:

- `<%@directive attribute1 = "value" attribute2 = "value" ….%>`

JSP supports following three types of directives:

- @page
- @include
- @taglib

@page Directive 1-2

Used to define page level attributes in the JSP. Following are the attributes which can be defined through the page directive:

- **import** – specifies packages that need to be imported to the JSP.
- **contentType** – used to set the HTTPResponse header content type field, the value is any one of the MIME types.
- **isThreadSafe** – used to specify whether the JSP implements a single thread model or not. The value of this attribute can be true/false.
- **session** - specifies whether the current JSP page participates in HTTP sessions or not. It assumes true/false values.
- **buffer** - specifies if the current response requires some buffer to be allocated on the client side.
- **autoflush** – specifies whether the output buffer should be flushed after the response is received. It assumes true/false values.

@page Directive 2-2

Following are some more attributes which can be defined through the page directive:

- **extends** – specifies the classes from which the current page can be inherited.
- **info** – takes a string value which can be returned through getServletInfo() method.
- **errorPage** – specifies the URL of the page which can be returned in case of error.
- **isErrorPage** – assumes a true/false value, specifies whether the current page is an error page or not.
- **language** – specifies the underlying scripting language of the JSP.

@include and @taglib Directives

@include directive

- Used to include files to the current JSP.
- Following is the usage of the @include directive:
  - <%@include = “/404.html” %>

@taglib directive

- JSP allows users to create their own tag libraries.
- The tag library comprises a tag library descriptor and tag handlers.
- The tag handlers are responsible for the implementation of the tag definitions and the tag library descriptor defines the mapping of the tags to appropriate handlers. The tag handlers are generally Java classes.
- These user-defined tag libraries can be included in the JSP files through the @taglib directive.

Thread Safe JSPs

For JSPs, single threaded programs can be executed by implementing the interface SingleThreadModel.

When SingleThreadModel interface is implemented, no two threads will simultaneously access the service () method of the servlet.

Thread safety can also be implemented though the @page directive where the entire page can be declared thread safe through the attribute isThreadSafe.

Processing Data Received from Servlet in JSPs 1-6

The standard way of handling data received from the user interface is through Java Beans.

Bean class has variables equal to the number of input elements in the form.

Each input element has a getter and setter method for handling the values of the variables.

Once the interface between the servlet and JSP is set, the servlet can perform the required operations on the input values read through the form.

Processing Data Received from Servlet in JSPs 2-6

Following is a JSP code where the values in the input elements are bound to a bean:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <jsp:useBean id="probean" scope="session"
class="Product.ProductsBean" />
        <jsp:setProperty name="probean" property="proName"
value="Galaxy"/>
        <jsp:setProperty name="probean" property="description"
value="Touchphone"/>
```

Processing Data Received from Servlet in JSPs 3-6

```
 <p>Product Name:
         <jsp:getProperty name="probean"
property="proName"/>
        </p>
        <p>Product description:
         <jsp:getProperty name="probean"
property="description"/>
        </p>
    </body>
</html>
```

Processing Data Received from Servlet in JSPs 4-6

The JSP uses a bean Product.ProductsBeans. The following demonstrates a bean class:

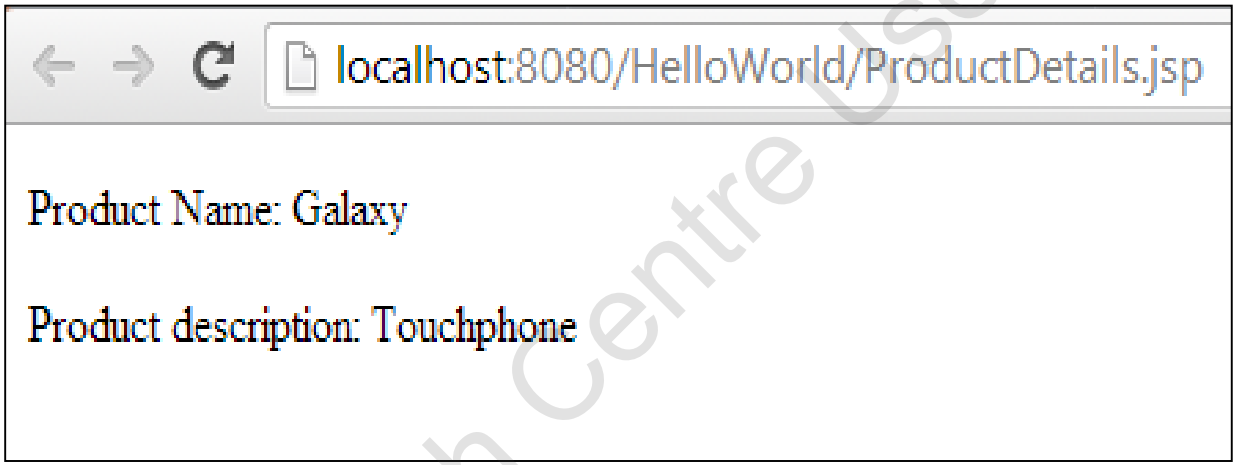
```
package Student;
public class ProductsBean
{
   private String proName;
   private String description;
   //private int age = 0;
   public ProductsBean() {
   }
   public String getProName(){
      return proName;
   }
   public String getDescription(){
      return description;
   …}
```

Processing Data Received from Servlet in JSPs 5-6

```
public int getAge(){
   return (int)age;
}
   public void setProName(String firstName){
      this.proName = firstName;
   }
   public void setDescription(String lastName){
      this.description = lastName;
   }
   public void setAge(Integer age){
      this.age = age;
   }
}
```

Processing Data Received from Servlet in JSPs 6-6

Following shows the output of the code:



JSP Standard Tag Library (JSTL)

JSTL is a repository of tags which are used to create JSPs. JSTL has tags for implementing various tasks which can be classified as follows:

- Core tags
- XML tags
- Formatting tags
- SQL tags
- JSTL functions

These tags support functions such as iterations, implement conditions, access databases, and so on.

JSP allows for custom defined tags, where user can define a set of tags.

The tag library can be included with @taglib directive.

Core Tags

All the core tags are prefixed with the letter ‘c’.

These tags support features such as variable support, flow control, URL management, and so on.

The URL for the core tag library is as follows:

- http://java.sun.com/jsp/jstl/core

Core tags perform the following categories of functions:

- Variable support
- Flow Control
- URL management
- Miscellaneous

XML Tags

All the XML tags are prefixed with ‘x’.

There are three categories of XML tags - core, flow control, and transformation.

The URL for XML tags is: http://java.sun.com/jsp/jstl/xml

Core tags provide the basic function of parsing and interpreting XML tags. The core XML tags are out, parse, and set.

Flow control tags are used to control the interpretation order of XML tags. The tags used for this are choose…when…otherwise, forEach, and if.

Transformation tags perform functions such as styling of the document. The tags used for transformation are transform…..param.

Formatting Tags and SQL Tags

Formatting tags

- Formatting tags are used for formatting the XML documents based on local parameters such as local time and so on.
- The prefix for formatting tags is 'fmt' and the formatting tags library is located at: http://java.sun.com/jsp/jstl/fmt
- The tags in this category are formatNumber, formatDate, parseDate, parseNumber, setTimeZone, and timeZone.

SQL tags

- SQL tags are used for database operations.
- All the SQL tags are prefixed with 'sql'.
- The URL for SQL tags is: http://java.sun.com/jsp/jstl/sql
- The functions of the SQL tag library are setting up the data source and performing the SQL operations.

JSTL Functions 1-4

- There are a set of predefined functions in JSP which are used over a group of objects.
- They are prefixed with 'fn'.
- The URL for the location of the JSTL functions is:
  http://java.sun.com/jsp/jstl/functions
- Following demonstrates the usage of commonly used JSTL tags.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib prefix="c"
uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fmt"
uri="http://java.sun.com/jsp/jstl/fmt" %>
<!DOCTYPE html>
<html>
    <head>
```

JSTL Functions 2-4

```
 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
 </head>
    <body>

        <P><b>Using forEach</b></p>
<c:forEach var="i" begin="101" end="105">
   Item <c:out value="${i}"/><p>
</c:forEach>
   <p><b>Using formatDate </b></p>
   <c:set var="now" value="<%=new java.util.Date()%>" />
   <p>Formatted Date: <fmt:formatDate type="date" value="${now}" /></p>


   <p><b>Using forTokens</b></p>
   <c:forTokens items="John,Mary,Rosy" delims="," var="name">
   <c:out value="${name}"/><p>
</c:forTokens>
```

JSTL Functions 3-4

```
        <p><b>Using choose</b></p>
  <c:set var="price" scope="session" value="${200*2}"/>
<p>Product price is : <c:out value="${price}"/></p>
        <c:choose>
    <c:when test="${price <= 0}">
        The product is for free.
    </c:when>
    <c:when test="${price < 1000}">
        Price is nominal.
    </c:when>
    <c:otherwise>
        Price is very high.
    </c:otherwise>
        </c:choose>

   <p><b>Using import</b></p>
<c:import var="data" url="http://www.tutorialspoint.com"/>
<c:out value="${data}"/>
    </body>
</html>
```

JSTL Functions 4-4

Following is the output of the code:

Custom Tags 1-5

JSP allows creation of custom tags, where the developer can define a set of tags.

The function of custom tags is defined through a tag handler.

The custom tags can be inserted directly into a JSP in the same manner as built-in tags.

A Tag Library Descriptor (TLD) file is an XML document containing information about each user defined tag.

Custom Tags 2-5

Following provides the definition of a Tag Library Descriptor(TLD):

```
<?xml version="1.0" encoding="UTF-8"?>
<taglib version="2.1" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-jsptaglibrary_2_1.xsd">
  <tlib-version>1.0</tlib-version>
  <short-name>hellotld</short-name>
  <uri>/WEB-INF/tlds/HelloTLD</uri>
  <tag>
    <name>HelloTag</name>
    <tag-class>pkg.HelloTag</tag-class>
    <body-content>empty</body-content>
  </tag>
</taglib>
```

Custom Tags 3-5

Following code displays the Java file defining the function of the custom tag:

```
import javax.servlet.jsp.tagext.*;
import javax.servlet.jsp.*;
import java.io.*;
public class HelloTag extends SimpleTagSupport {
  public void doTag() throws JspException,
IOException {
    JspWriter out = getJspContext().getOut();
    out.println("Hello! This is my first Custom
Tag.");
  }
}
```

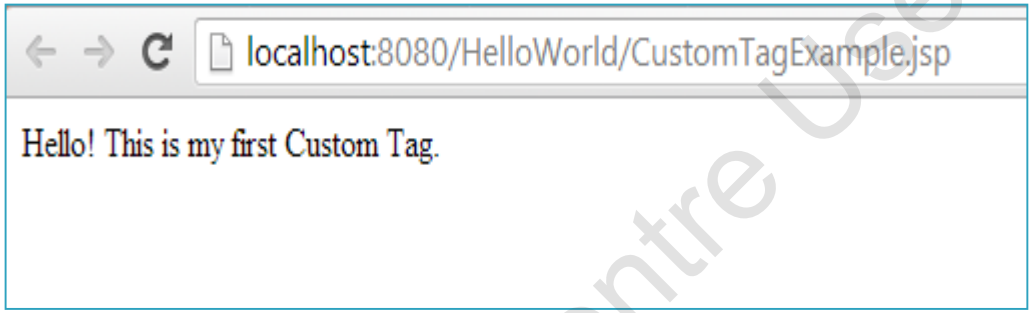
Custom Tags 4-5

Following demonstrates the usage of custom tag:

```
<%@ taglib prefix="ex" uri="WEB-INF/tlds/HelloTLD.tld"%>
<!DOCTYPE html>
<html>
    <head>
        <meta   http-equiv="Content-Type"   content="text/html;
charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>

        <ex:HelloTag />

    </body>
</html>
```

Custom Tags 5-5

Following displays the JSP generated by using a custom tag:



Unified Expression Language 1-2

- Expressions are used in JSPs to retrieve values in the scriptlets, setting values to the attributes, parameters, and so on.
- Expressions are also used in JSFs.
- Unified Expression language is used to make the expressions of both JSF and JSP compatible with each other.
- Following are the advantages of having a unified expression language which is compatible with both JSP and JSF:
  - It allows deferred evaluation of expressions.
  - It supports expressions which can set values for the variables and also invoke methods.
  - JSTL iteration tags can be used along with deferred evaluation of expressions.

Unified Expression Language 2-2

- There are two types of expressions supported by EL:
  - Method expressions
  - Value expressions
- All the expressions are resolved through a generic ELResolver class.
- The ELResolver identifies the expressions and evaluates the Java bean components. For instance, the following expression returns the context path of the page:
  - `${pageContext.request.contextPath}`

Summary

- JavaServer Pages is a server side technology used to develop dynamic Web pages.
- JSPs work along with servlets to handle client request and responses. The client requests and responses are HTTP requests and responses.
- Using JSPs is advantageous over other similar technologies used on Internet due to platform independence and portability of Java.
- At runtime, the JSPs are converted into servlets and executed.
- JSPs have three types of elements – scriptlets, expressions, and directives.
- JSP has a repository of tags to perform various functions known as JSP Standard Tag Library (JSTL).
- Unified Expression Language (UEL) is used to evaluate JSP expressions such that they are compatible with JSFs.