Java Server Pages

Code Example: Using scripting elements

The next example code consists on two JSP pages namely first.jsp and second.jsp. The user will enter two numbers on the first.jsp and after pressing the *calculate sum* button, able to see the sum of entered numbers on second.jsp

first.jsp

This page only displays the two text fields to enter numbers along with a button.

```
<html>
	<body>
		<h2>Enter two numbers to see their sum</h1>
		<!—the form values will be posted to second.jsp -->
		<form name = "myForm" action="second.jsp" >
			<h3> First Number </h3>
			<input type="text" name="num1" />
			<h3> Second Number </h3>
			<input type="text" name="num2" />
			<br/><br/>
			<input type="submit" value="Calculate Sum" />
		</form>
	</body>
</html>
```

second.jsp

This page retrieves the values posted by first.jsp. After converting the numbers into integers, displays their sum.

```
<html>
	<body>
		<!-- JSP to sum two numbers -->
		<%-- Declaration--%>
		<%!
			// declaring a variable to store sum
			int res;
			// method helps in calculating the sum
			public int sum(int op1, int op2) {
				return op1 + op2;
			}
		%>

		<%-- Scripltet--%>
		<%
			String op1 = request.getParameter("num1");
			String op2 = request.getParameter("num2");
			int firstNum = Integer.parseInt(op1);
			int secondNum = Integer.parseInt(op2);
			// calling method sum(), declared above in declartion tag
			res = sum(firstNum, secondNum);
		%>
		<%-- expression used to display sum --%>
		<h3>Sum is: <%=res%> </h3>
	</body>
</html>
```

Writing JSP scripting Elements in XML

Now days, the preferred way for composing a JSP pages is using XML. Although writing JSP pages in old style is still heavily used as we had shown you in the last example.
Equivalent XML tags for writing scripting elements are given below:

- **Comments:** No equivalent tag is defined
- **Declaration:** <jsp:declartion> </jsp:declaration>
- **Expression:** <jsp:expression> </jsp:expression>
- **Scriptlet:** <jsp:scriptlet> </jsp:scriptlet>

It's important to note that every opening tag also have a closing tag too.

The second.jsp of last example is given below in XML style.

```
<?xml version="1.0" encoding="UTF-8"?>

<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0">
        <!-- to change the content type or response encoding change the following line -->

<jsp:directive.page contentType="text/xml;charset=UTF-8"/>

<!-- any content can be specified here, e.g.: -->

<jsp:element name="text">
        <jsp:body>
                <jsp:declaration>
                        int res;
                        public int sum(int op1, int op2) {
                                return op1 + op2;
                        }
                </jsp:declaration>

                <jsp:scriptlet>
                        String op1 = request.getParameter("num1");
                        String op2 = request.getParameter("num2");
                        int firstNum = Integer.parseInt(op1);
                        int secondNum = Integer.parseInt(op2);
                        res = sum(firstNum, secondNum);
                </jsp:scriptlet>

        <jsp:text> Sum is: </jsp:text>
                <jsp:expression> res </jsp:expression>
        </jsp:body>
</jsp:element>
</jsp:root>
```

JavaServer Pages

We have started JSP journey in the last handout and thoroughly discussed the JSP scripting elements. JSP directive elements and implicit objects will be discussed in this handout. Let's review JSP journey again to find out what part we have already covered.

 Directive Elements
– Provides global control of JSP ……..…….. **<%@ %>**
 Scripting Elements
– JSP comments …….. **<%-- --%>**
– declarations …….. **<%! %>**
• Used to declare instance variables & methods
– expressions …….. **<%= %>**
• A java code fragment which returns String
– scriptlets …….. **<% %>**
• Blocks of java code
 Action Elements
– Special JSP tags ……..…….. **<jsp: ….. />**

We start our discussion from implicit objects. Let's find out what these are?

Implicit Objects

To simplify code in JSP expressions and scriptlets, you are supplied with eight automatically defined variables, sometimes called *implicit objects*. The three most important variables are request, response & out. Details of these are given below:

– request
This variable is of type HttpServletRequest, associated with the request. It gives you access to the request parameters, the request type (e.g. GET or POST), and the incoming HTTP request headers (e.g. cookies etc).

– response
This variable is of type HttpServletResponse, associated with the response to client. By using it, you can set HTTP status codes, content type and response headers etc.

– out
This is the object of JspWriter used to send output to the client.

Code Example: Use of Implicit Objects

The following example constitutes of 4 JSP pages. These are index.jsp, controller.jsp, web.jsp and java.jsp. The user will select either the option of *"java"* or *"web"* from index.jsp, displayed in the form of radio buttons and submits the request to controller.jsp. Based on the selection made by the user, controller.jsp will redirect the user to respective pages (web.jsp or java.jsp).

The code of these entire pages is given below.

index.jsp

```
<html>
<body>
	<h2>Select the page you want to visit</h2>
	<form name="myForm" action="controller.jsp" >
	<h3>
	<input type="radio" name = "page" value="web"/>
	Web Design & Develoment
	</h3>
	<br>
	<h3>
	<input type="radio" name = "page" value="java"/>
	Java
	</h3>

	<br>
	<input type="submit" value="Submit" />
	</form>
</body>
</html>
```

controller.jsp

```
<html>
<body>
	<!-- scriptlet -->
	<%
	// reading parameter “page”, name of radio button using
	// implicit object request
	String pageName = request.getParameter("page");
	// deciding which page to move on based on “page” value
	// redirecting user by using response implicit object
	if (pageName.equals("web")) {
		response.sendRedirect("web.jsp");
	} else if (pageName.equals("java") ) {
		response.sendRedirect("java.jsp");
	}
	%>
</body>
</html>
```

web.jsp

```
<html>
<body>
	// use of out implicit object, to generate HTML
	<%
	out.println( "<h2>" +
	"Welcome to Web Design & Development Page" +
	"</h2>"
	);
	%>
</body>
</html>
```

java.jsp

```
<html>
<body>
	// use of out implicit object, to generate HTML
	<%
	out.println( "<h2>" +
```

```
            "Welcome to Java Page" +
            "</h2>"
            );
            %>
</body>
</html>
```

The details of remaining 5 implicit objects are given below:

– **session**
This variable is of type HttpSession, used to work with session object.

– **application**
This variable is of type ServletContext. Allows to store values in key-value pair form that are shared by all servlets in same web application/

– **config**
This variable is of type ServletConfig. Represents the JSP configuration options e.g. init-parameters etc.

– **pageContext**
This variable is of type javax.servlet.jsp.PageContext, to give a single point of access to many of the page attributes. This object is used to stores the object values associated with this object.

– **exception**
This variable is of type java.lang.Throwable. Represents the exception that is passed to JSP error page.

– **page**
This variable is of type java.lang.Object. It is synonym for this.

JSP Directives

JSP directives are used to convey special processing information about the page to JSP container. It affects the overall structure of the servlet that results from the JSP page. It enables programmer to:
– Specify page settings
– To Include content from other resources
– To specify custom-tag libraries

Format

<%@ directive {attribute=”val”}* %>
In JSP, there are three types of directives: page, include & taglib. The formats of using these are:
– **page:** <%@ **page** {attribute=”val”}* %>
– **include:** <%@ **include** {attribute=”val”}* %>
– **taglib:** <%@ **taglib** {attribute=”val”}* %>

JSP page Directive

Give high level information about servlet that will result from JSP page. It can be used anywhere in the document. It can control

– Which classes are imported
– What class the servlet extends
– What MIME type is generated
– How multithreading is handled
– If the participates in session
– Which page handles unexpected errors etc.

The lists of attributes that can be used with page directive are:

• **language** = "java"
• **extends** = "package.class"
• **import** = "package.*,package.class,…"
• **session** = "**true** | false"
• **Info** = "text"
• **contentType** = "mimeType"
• **isThreadSafe** = "**true** | false"
• **errorPage** = "relativeURL"
• **isErrorPage** = "true | **false**"

Some example uses are:

– To import package like java.util <%@page import="java.util.*" info="using util package" %>
– To declare this page as an error page
<%@ page isErrorPage = "true" %>
– To generate the excel spread sheet
<%@ page contentType = "application/vnd.ms-excel" %>

JSP include Directive

Lets you include (reuse) navigation bars, tables and other elements in JSP page. You can include files at

– Translation Time (by using include directive)
– Request Time (by using Action elements, discussed in next handouts)

Format

<%@include file="relativeURL"%>

Purpose

To include a file in a JSP document at the time document is translated into a servlet. It may contain JSP code that affects the main page such as response page header settings etc.

Example Code: using include directive

This example contains three JSP pages. These are index.jsp, header.jsp & footer.jsp. The header.jsp will display the text of *"web design and development"* along with current date. The footer.jsp will display only *"Lahore University of Management Sciences"*. The outputs of both these pages will be included in index.jsp by using JSP include directive.

header.jsp

```
<% @page import="java.util.*"%>
<html>
<body>
        <h3> Web Desing & Development </h3>
        <h3><%=new Date()%></h3>
</body>
</html>
```

footer.jsp

```
<html>
<body>
        <h3> Lahore University of Management Science </h3>
</body>
</html>
```

index.jsp

```
<html>
<body>
        // includes the output of header.jsp
        <% @include file="header.jsp" %>

        <TABLE BORDER=1>
                <TR><TH></TH><TH>Apples<TH>Oranges
                <TR><TH>First Quarter<TD>2307<TD>4706
                <TR><TH>Second Quarter<TD>2982<TD>5104
                <TR><TH>Third Quarter<TD>3011<TD>5220
                <TR><TH>Fourth Quarter<TD>3055<TD>5287
        </TABLE>
        // includes the output of footer.jsp
        <% @include file="footer.jsp" %>
</body>
</html>
```

Example Code: setting content type to generate excel spread sheet

In this example, index.jsp is modified to generate excel spread sheet of the last example. The change is shown in bold face.

index.jsp

```
// setting content type to generate excel sheet using page directive
<% @page contentType="application/vnd.ms-excel" %>
<html>
<body>
        // includes the output of header.jsp
        <%@include file="header.jsp" %>
        <TABLE BORDER=1>
                <TR><TH></TH><TH>Apples<TH>Oranges
                <TR><TH>First Quarter<TD>2307<TD>4706
                <TR><TH>Second Quarter<TD>2982<TD>5104
                <TR><TH>Third Quarter<TD>3011<TD>5220
                <TR><TH>Fourth Quarter<TD>3055<TD>5287
        </TABLE>
        // includes the output of footer.jsp
        <%@include file="footer.jsp" %>
</body>
</html>
```

References:

- Java A Lab Course by Umair Javed
- Core Servlets and JSP by Marty Hall