UNIT 4-JSP

JSP

Overview

Jsp

ava Server Pages (JSP) is

a technology for developing web pages that support dynamic content which

helps developers insert java code in HTML pages by making use of special JSP tags, most of which start with <%

and end with %>.

A JavaServer Pages component is a type of Java servlet

that is designed to fulfill the role of a user interface for a

Java web application. Web developers write JSPs as text files that combine HTML or XHTML code, XML

elements, and embedded JSP actions and commands.

Using JSP, you can collect input from users t

hrough web page forms, present records from a database or another

source, and create web pages dynamically.

JSP tags can be used for a variety of purposes, such as retrieving information from a database or registering user
preferences, accessing JavaBeans
components, passing control between pages and sharing information
between requests, pages etc.

Why Use JSP?

JavaServer Pages often serve the same purpose as programs implemented using the Common Gateway
Interface (CGI). But JSP offer several advantages in
comparison with the CGI.

•Performance is significantly better because JSP allows embedding Dynamic Elements in HTML Pages itself instead of having a separate CGI files.

•JSP are always compiled before it's processed by the server unlike CGI/Perl which requires the server to load an interpreter and the target script each time the page is requested.

•JavaServer Pages are built on top of the Java Servlets API, so like Servlets, JSP also has access to all the powerful Enterprise Java APIs, including JDBC, JNDI, EJB, JAXP etc.

•JSP pages can be used in combination with servlets that handle the business logic, the model supported by Java servlet template engines.

Finally, JSP is an integral part of J2EE, a complete platform for enterprise class applications. This mean
s that JSP can play a part in the simplest applications to the most complex and demanding.C

Advantages of JSP:

Following is the list of other advantages of using JSP over other technologies:

•vs. Active Server Pages (ASP):The advantages of JSP are twofold. First, the dynamic part is written in Java, not Visual Basic or other MS specific language, so it is more powerful and easier to use. Second, it is portable to other operating systems and non-Microsoft Web servers.

•vs. Pure Servlets:It is more convenient to write (and to modify!) regular HTML than to have plenty of println statements that generate the HTML.

•vs. Server-Side Includes (SSI):SSI is really only intended for simple inclusions, not for "real" programs that use form data, make database connections, and the like.

•vs. JavaScript:JavaScript can generate HTML dynamically on the client but can hardly interact with the web server to perform complex tasks like database access and image processing etc.

•vs. Static HTML:Regular HTML, of course, cannot contain dynamic information

JSP Declarations:

A declaration declares one or more variables or methods that you can use in Java code later in the JSP file. You must declare the variable or method before you use it in the JSP file.

Following is the syntax of JSP Declarations:

<%!declaration;[declaration;]+...%>

You can write XML equivalent of the above syntax as follows:

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

Following is the simple example for JSP Declarations:

<%!int i =0;%>
<%!int a,b,c;%>
<%!Circle a =newCircle(2.0);%>
<\%Represents static <% literal.%\>
Represents static %>
literal.\'A single quote in an attribute that uses single quotes.\"A double quote in an attribute that uses double quotes.

JSP Directives:

A JSP directive affects the overall structure of the servlet class. It usually has the following form:

<%@directive attribute="value"%>

There are three types of directive tag:

Directive
Description
<%@ page ... %>
Defines page
-
dependent attributes, such as scripting language, error page, and buffering requirements.

<%@ include ... %>

Includes a file during the translation phase.

<%@ tag lib ... %>

Declares a tag library, containing custom actions, used in the page

The <jsp:forward> Action

The forward action terminates the action of the current page and forwards the request to another resource such as a static page, another JSP page, or a Java Servlet.

The simple syntax of this action is as follows:

<jsp:forward page="Relative URL"/>

Following is the list of required attributes associated with forward action:

Attribute	Description
page	Should consist of a relative URL of another resource such as a static page, another JSP page, or a Java Servlet

Example:

Let us reuse following two files (a) date.jps and (b) main.jsp as follows:

Following is the content of date.jsp file:

```
<p>
Today's date:
<%=(new java.util.Date()).toLocaleString()%>
</p>
```

Here is the content of main.jsp file:

```
<html>
<head>
<title>
The include Action Example
</title>
</head>
<body>
<center>
```

<h2>
The include action Example
</h2>
<jsp:forward page="date.jsp"/>

</body>
</html>

Now let us keep all these files in root directory and try to access main.jsp. This would display r esult something like as below. Here it discarded content from main page and displayed content from forwarded page only.

Today's date: 12-Sep-2010 14:54:22

The <jsp:plugin> Action

The plugin action is used to insert Java components into a JSP page. It dete rmines the type of browser and inserts the <object> or <embed> tags as needed.If the needed plugin is not present, it downloads the plugin and then executes the Java component. The Java component can be either an Applet or a JavaBean.

The plugin action has several attributes that correspond to common HTML tags used to format Java components.

The <param> element can also be used to send parameters to the Applet or Bean.

Following is the typical syntax of using plugin action:

<jsp:plugin type="applet"code base="dirname"code="MyApplet.class" width="60",height="80">
<jsp:paramname="fontcolor" value="red"/>
<jsp:param name="background"value="black"/>
<jsp:fallback>
Unable to initialize Java Plugin
</jsp:fallback>
</jsp:plugin>

You can try this action using some applet if you are interested. A new element, the <fallback> element, can be
used to specify an error string to be sent to the user in case the component fails.

The <jsp:element> Action

The <jsp:attribute> Action

The <jsp:body> Action

The <jsp:element>, lt;jsp:attribute> and <jsp:body> actions are used to define XML elements dynamically. The word dynamically is important, because it means that the XML elements can be generated at request time rather than statically at compile time.

Following is a simple example to define XML elements dynamically:

```
<%@page language ="java" contentType="text/html"%>
<html xmlns=http://www.w3c.org/1999/xhtml xmlns:jsp="http://java.sun.com/JSP/Page">
<head><title>
Genera
te XML Element
</title></head>
<body>
<jsp:element name="xmlElement">
<jsp:attribute name="xmlElementAttr">
Value for the attribute
</jsp:attribute>
<jsp:body>
Body for XML element
</jsp:body>
</jsp:element>
</body>
</html>
```

This would produce followin
g HTML code at run time:

```
<html xmlns=http://www.w3c.org/1999/xhtml xmlns:jsp="http://java.sun.com/JSP/Page">
<head><title>
Generate XML Element
</title></head>
<body>
<xmlElement xmlElementAttr= "Value for the attribute">
Body for XML element
</xmlElement>
</body>
```

</html>

The <jsp:text> Action

The <jsp:text> action can be used to write template text in JSP pages and documents. Following is the simple

syntax for this action:

<jsp:text>

Template data

</jsp:text>

The body fo the template cannot contain other elements; it can only contain text and EL expressions (Note: EL expressions are explained in subsequent chapter). Note that in XML files, you cannot use expressions such as ${whatever > 0}, because the greater than signs are illegal. Instead, use the gt form, such as ${whatever gt 0} or an alternative is to embed the value in a CDATA section.