Web Component Development with Servlet & JSP Technologies (EE 6)

Module-7: Developing JSP Pages

Team Emertxe

ORACLE
Certified Expert
Java EE 6 Web Component Developer

EMERTXE

Objectives

Upon completion of this module, you should be able to:

- Describe JSP page technology
- Write JSP code using scripting elements
- Write JSP code using the page directive
- Write JSP code using standard tags
- Write JSP code using the EL
- Configure the JSP page environment in the web.xml file

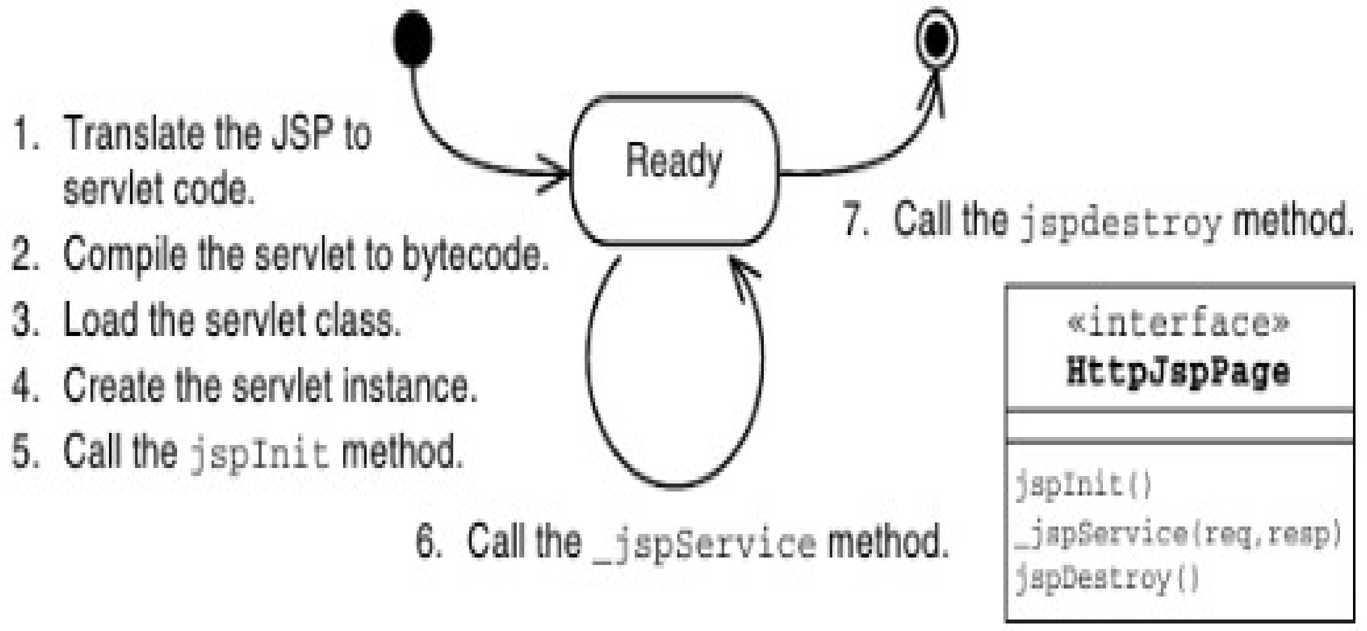
Relevance

Discussion – The following questions are relevant to understanding JSP technology:

- What problems exist in generating an HTML response in a servlet?
- How do template page technologies (and JSP technology in particular) solve these problems?

How a JSP Page Is Processed

A JSP page is essentially source code, and must be converted into a servlet before it can service requests. Figure shows the steps of JSP page processing.

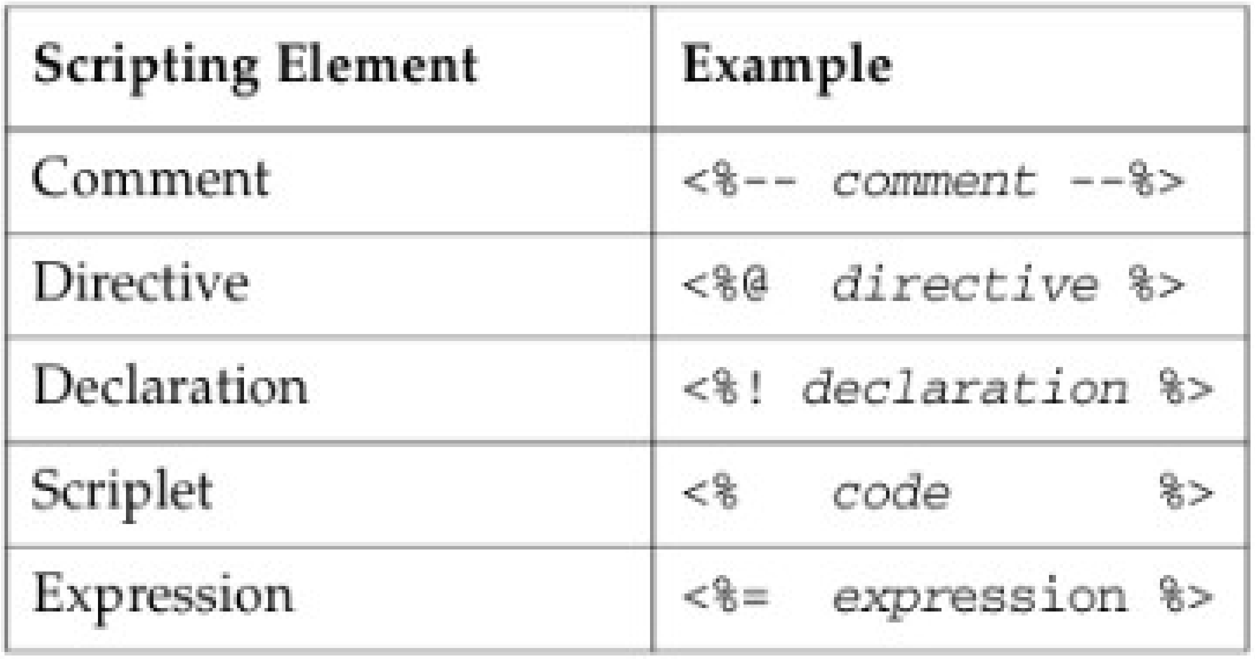

1. Translate the JSP to servlet code.
2. Compile the servlet to bytecode.
3. Load the servlet class.
4. Create the servlet instance.
5. Call the jspInit method.
Ready
6. Call the _jspService method.
7. Call the jspdestroy method.
«interface»
HttpJspPage
jspInit()
_jspService(req,resp)
jspDestroy()

Writing JSP Scripting Elements

JSP scripting elements are embedded with the <% %> tags and are processed by the JSP engine during translation of the JSP page. Any other text in the JSP page is considered part of the response and is copied verbatim to the HTTP response stream that is sent to the web browser.

```
<html>
<%-- scripting element --%>
</html>
```

Writing JSP Scripting Elements



Scripting Element	Example
Comment	`<%-- comment --%>`
Directive	`<%@ directive %>`
Declaration	`<%! declaration %>`
Scriplet	`<% code %>`
Expression	`<%= expression %>`

Comments

HTML Comment:

```
<!-- This is an HTML comment. It will show up in the response. -->
```

JSP Comment:

```
<%-- This is a JSP comment. It will only be seen in the JSP m code.
It will not show up in either the servlet code or the response.--%>
```

Java Comment:

```
<%
/* This is java comment and it will only be seen in servlet.It will not show
in response. */
%>
```

Directive Tag

A directive tag provides information that affects the overall translation of the JSP page. The syntax for a directive tag is as follows:

```
<%@ DirectiveName [attr=”value”]* %>
```

Three types of directives are used in JSP pages: page, include, and taglib. The following are some examples:

```
<%@ page session=”false” %>
```

```
<%@ include file=”incl/copyright.html” %>
```

The page and include directives are described later in this module. The taglib directive is described in Module 8, “Developing JSP Pages Using Custom Tags.”

Declaration Tag

A declaration tag lets you include members in the JSP servlet class, either attributes or methods. The syntax for a declaration tag is as follows:

```
<%! JavaClassDeclaration %>
```

Scriptlet Tag

A scriptlet tag lets the JSP page developer include arbitrary Java technology code in the _jspService method. The syntax for a scriptlet tag is as follows:

```
<% JavaCode %>
```

Expression Tag

An expression tag holds a Java language expression that is evaluated during an HTTP request. The result of the expression is included in the HTTP response stream. The syntax for an expression tag is as follows:

```
<%= JavaExpression %>
```

Implicit Variables

The JSP engine gives you access to the following implicit variables, also called implicit objects, in scriptlet and expression tags. These variables represent commonly used objects for servlets that JSP page developers might need to use. For example, you can retrieve HTML form parameter data by using the request variable, which represents the HttpServletRequest object.

Implicit Variables

Request
Response
Out
Session
Application
Config
PageContext
Page
Exception

Using the page Directive

You use the page directive to modify the overall translation of the JSP page. For example, you can declare that the servlet code generated from a JSP page requires the use of the Date class:

```
<%@ page import=”java.util.Date” %>
```

Including JSP Page Segments

There are two standard approaches to including presentation segments in your JSP pages:

- The include directive
- The jsp:include standard action

Using the include Directive

The include directive lets you include a segment in the text of the main JSP page at translation time. The syntax of this JSP technology directive is as follows:

```
<%@ include file=”segmentURL” %>
```

Using Standard Tags

- The JSP specification provides standard tags for use within your JSP pages. Because these tags are required in the specification, every JSP container must support them. This ensures that any applications using the standard tags will work in any compliant JSP container.

- Standard tags begin with the jsp: prefix. You use these tags to reduce or eliminate scriptlet code within your JSP page. However, the standard tags only provide limited functionality. Using the JSTL and EL reduces the need for the standard tags.

JavaBeans Components

A JavaBeans component is a Java technology class with at minimum the following features:

- Properties defined with accessors and mutators (get and set methods).
- A no-argument constructor.
- No public instance variables.
- The class implements the java.io.Serializable interface.

The useBean Tag

If you want to interact with a JavaBeans component using the standard tags in a JSP page, you must first declare the bean. You do this by using the useBean standard tag.

The syntax of the useBean tag is as follows:

```
<jsp:useBean id=”beanName”
scope=”page | request | session | application”
class=”className” />
```

The id attribute specifies the attribute name of the bean. Where the bean is stored is specified by the scope attribute. Scope defaults to page if not specified. The class attribute specifies the fully qualified class name.

The setProperty Tag

- You use the setProperty tag to store data in the JavaBeans instance. The syntax of the setProperty tag is as follows:

```
<jsp:setProperty name="beanName" property_expression/>
```

- The property attribute specifies the property within the bean that will be set. For example, to set the email property in the customer bean, you can use the following:

```
<jsp:setProperty name=”cust” property=”email” />
```

The getProperty Tag

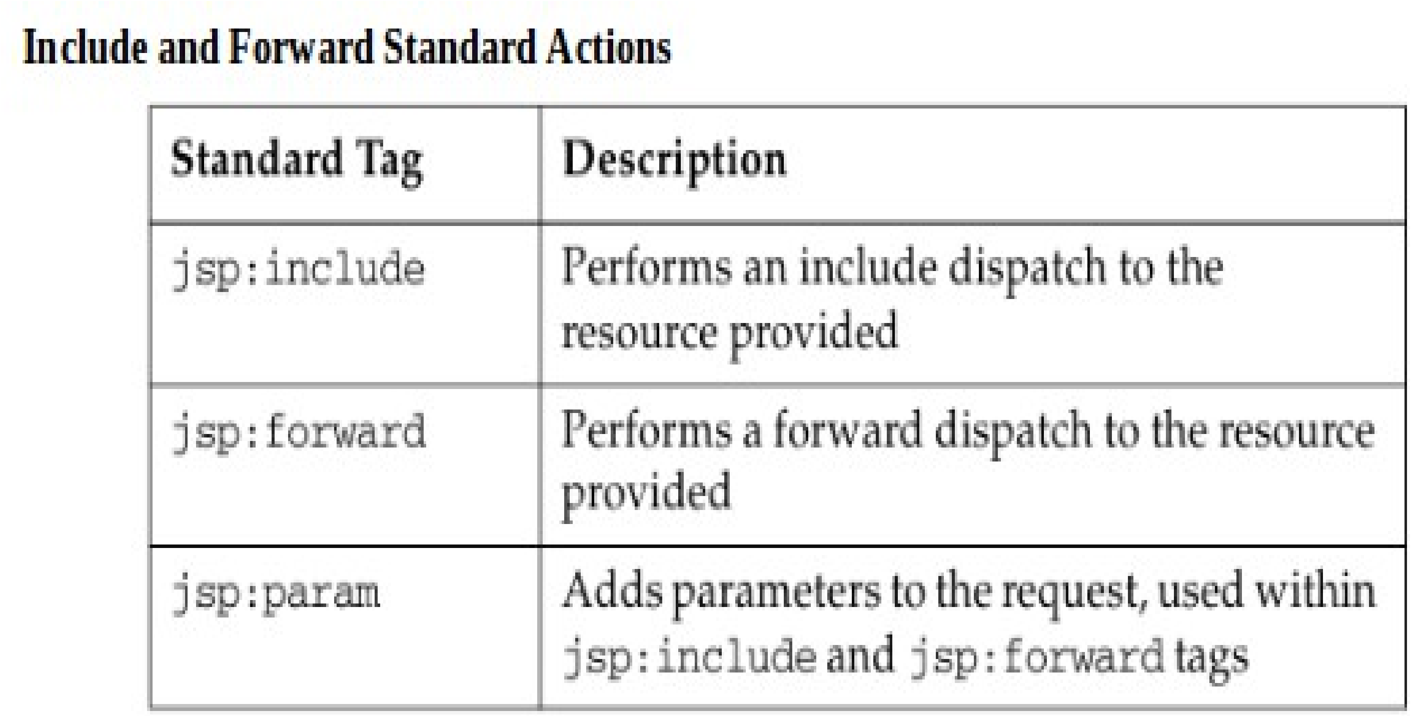
The getProperty tag is used to retrieve a property from a JavaBeans instance and display it in the output stream. The syntax of the getProperty tag is as follows:

```
<jsp:getProperty name=”beanName”
property=”propertyName” />
```

The name attribute specifies the name of the JavaBeans instance and the property attribute specifies the property used for the get method. Given a getProperty usage of the following:

```
<jsp:getProperty name=”cust” property=”email” />
```

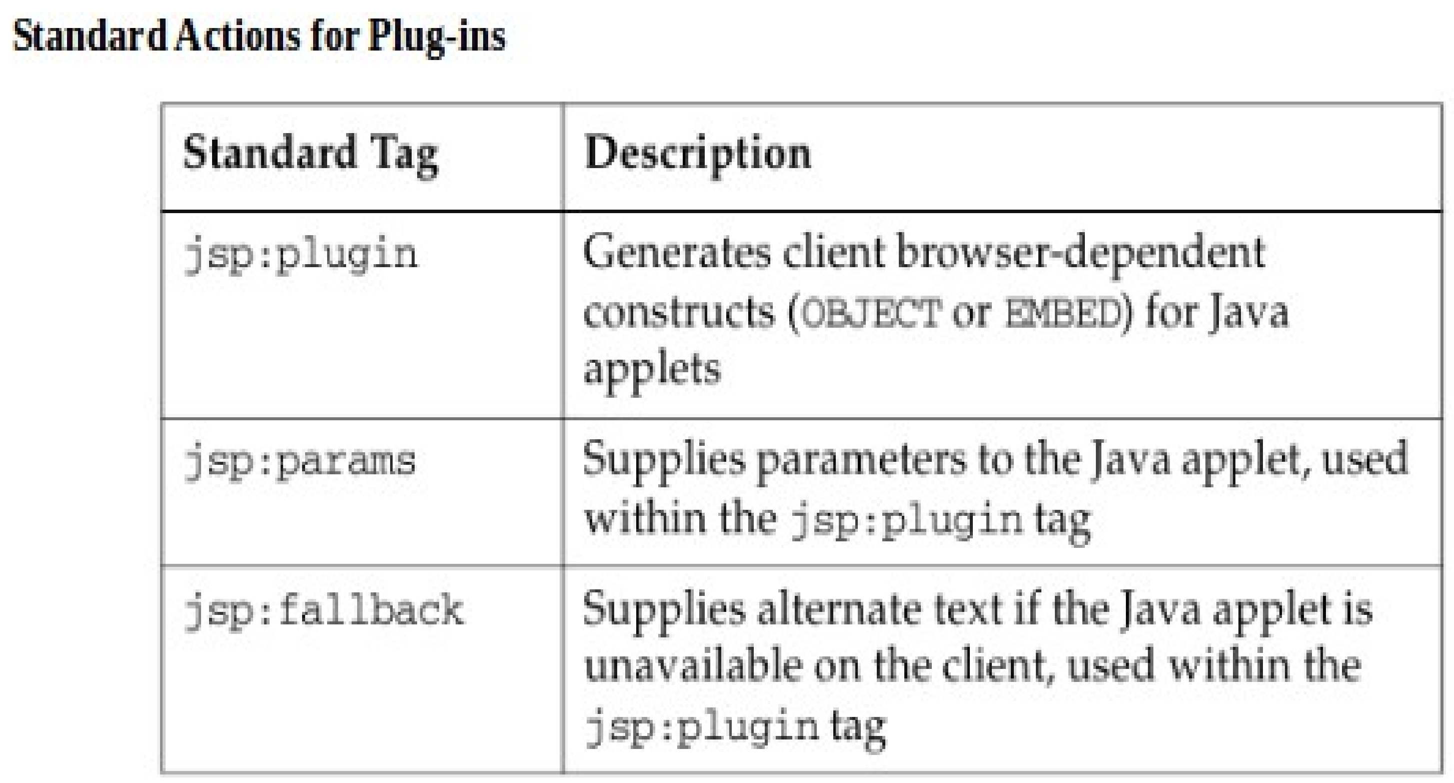
Other Standard Tags



Include and Forward Standard Actions

Standard Tag	Description
`jsp:include`	Performs an include dispatch to the resource provided
`jsp:forward`	Performs a forward dispatch to the resource provided
`jsp:param`	Adds parameters to the request, used within `jsp:include` and `jsp:forward` tags

Other Standard Tags



Standard Actions for Plug-ins

Standard Tag	Description
`jsp:plugin`	Generates client browser-dependent constructs (`OBJECT` or `EMBED`) for Java applets
`jsp:params`	Supplies parameters to the Java applet, used within the `jsp:plugin` tag
`jsp:fallback`	Supplies alternate text if the Java applet is unavailable on the client, used within the `jsp:plugin` tag

Using the jsp:include Standard Action

- The jsp:include standard action lets you include a segment in the text of the main JSP page at runtime. The syntax of this JSP technology action tag is as follows:

  ```
  <jsp:include page=”segmentURL” />
  ```

- You might have noticed that the include directive uses an attribute called file and the jsp:include standard action uses an attribute called page. These attributes represent the same concept: The URL to the segment file to be included.

- The jsp:include action is equivalent to using the include method on a RequestDispatcher object. For example:

  ```
  RequestDispatcher segment = request.getRequestDispatcher(page);
  segment.include(request, response);
  ```

Using the jsp:param Standard Action

Sometimes, you might want to alter a presentation segment at runtime.

The jsp:include action lets you pass additional parameters to the presentation segment at runtime using the jsp:param standard action. The syntax of this JSP technology action tag and the relationship to the jsp:include action is:

```
<jsp:include page=”segmentURL”>
<jsp:param name=”paramName” value=”paramValue”/>
</jsp:include>
```

ΣMERTXE

Using the jsp:param Standard Action

```
<!-- START of banner -->
<jsp:include page=”/WEB-INF/view/common/banner.jsp”>
<jsp:param name=”subTitle” value=”Registration” />
</jsp:include>
<!-- END of banner -->
```

Stay connected

About us: Emertxe is India's one of the top IT finishing schools & self learning kits provider. Our primary focus is on Embedded with diversification focus on Java, Oracle and Android areas

Emertxe Information Technologies,
No-1, 9th Cross, 5th Main,
Jayamahal Extension,
Bangalore, Karnataka 560046
T: +91 80 6562 9666
E: training@emertxe.com

https://www.facebook.com/Emertxe https://twitter.com/EmertxeTweet https://www.slideshare.net/EmertxeSlides

Thank You