Chapter 14

SECTION III: JAVA SERVER PAGES

Java Server Pages Document

A JSP page is very similar to an HTML or a XML page. Every JSP page holds text marked with a variety of tags. A standard JSP page is not a valid XML page however there is a separate JSP tag syntax, which allows using JSP tags within XML documents.

The JSP specification supports **JSP pages** as well as **JSP documents** i.e. two basic styles of delimiting its scripting elements. This means a JSP page can now use either **traditional JSP style syntax** or **XML style JSP syntax** within its source file.

JSP pages use the **traditional** or **shorthand** syntax, whereas JSP documents are completely **XML-compliant**. JSP documents are also referred to as JSP pages using XML syntax.

HINT

☺ As a JSP source file cannot **inter-mix** JSP syntax and XML syntax.

Following are the advantages of using a JSP Document:

- JSP documents can be easily verified as well-formed XML/HTML
- JSP documents can be validated against an XML Schema
- JSP documents can be readily written and parsed using standard XML tools
- JSP documents can be readily written and presented in alternate forms using XSLT [e**X**tensible **S**tylesheet **L**anguage **T**ransformations]
- JSP uses XML-compliant include and forward actions as well as custom tags. This makes the entire document XML-compliant, which in turn leads to increased code consistency
- JSP documents require slightly more developer discipline than JSP pages. This makes the code spec more readable and maintainable, especially for those to whom JSP is new

Converting a JSP page into the JSP document is very simple and straightforward.

Example: [In a regular JSP Page]

```
<%@ page language="java" session="true" %>
<%!
   public java.util.Date PrintDate()
   {
         return(new java.util.Date());
   }
   int Counter;
%>
<HTML>
   <HEAD>
         <TITLE>Displays Current Date</TITLE>
   <HEAD>
   <BODY>
         The current date is: <%= PrintDate()%><BR>
         This page is visited
         <%
               Counter++;
               out.print(Counter);
         %>
         times.
   </BODY>
```

For the most part, the conversion consists of converting expressions, scriptlets, directives and declarations to their XML style JSP syntax.

The following table describes the normal JSP page syntax and the XML style JSP syntax:

Components		**JSP Page Syntax**	**JSP Document Syntax**
Comments		<%-- . . . --%>	<!-- . . . -->
Directives	Page	<%@ page . . . %>	<jsp:directive.page . . . />
	Include	<%@ include . . . %>	<jsp:directive.include . . . />
	Taglib	<%@ taglib . . . %>	<jsp:root> element is annotated with namespace information
Declarations		<%! . . . %>	<jsp:declaration> . . . </jsp:declaration>
Scriptlets		<% . . . %>	<jsp:scriptlet> . . . </jsp:scriptlet>
Expressions		<%= . . . %>	<jsp:expression> . . . </jsp:expression>

HINT

JSP documents do not have an XML-compliant version of JSP comments. To overcome this issue, client-side [HTML-/XML-style] comments or embedded Java comments can be used in scriptlets. The Java comments in scriptlets require placing Java code directly in the JSP document.

Example: [In an XML style JSP syntax]

```
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0">
<jsp:directive.page language="java" session="true" />
<jsp:declaration>
   public java.util.Date PrintDate()
   {
         return(new java.util.Date());
   }
   int Counter;
</jsp:declaration>
<HTML>
   <HEAD>
         <TITLE>Displays Current Date</TITLE>
   <HEAD>
   <BODY>
         The current date is: <jsp:expression> PrintDate()</jsp:expression><BR>
         This page is visited
         <jsp:scriptlet>
               Counter++;
               out.print(Counter);
         </jsp:scriptlet>
         times.
   </BODY>
</jsp:root>
```

Directives

In a regular JSP page, directives are characterized using the @ character within the <% %> tags.

In an XML based JSP page, directives are characterized as follows:

```
<jsp:directive.<DirectiveName> attributes="<Value>" />
```

Following are the directives available in JSP:

- The **page** directive
- The **include** directive
- The **taglib** directive

The page Directive

Page directives can be placed anywhere in the document, but are often placed at the very top:

```
<jsp:directive.page language="java" contentType="text/html" import="java.util.*" />
<HTML>
   <HEAD>
        <TITLE>JSP page Directive Example</TITLE>
   </HEAD>
   <BODY>
        Hello World!
   </BODY>
</HTML>
```

Syntax:

```
<jsp:directive.page attributes="<Value>" />
```

Following are the attributes associated with the page directive:

- **language:** Defines which scripting language the file uses

  Syntax:

  ```
  <jsp:directive.page language="java" />
  ```

- **extends:** Holds the name of the Java language class that the Servlet extends

  Syntax:

  ```
  <jsp:directive.page extends="javax.servlet.http.HttpServlet" />
  ```

- **import:** Imports all the available classes from the specified Java package into the current JSP page

  Syntax:

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

- **session:** Allows the page to participate in an HTTP session

  Syntax:

  ```
  <jsp:directive.page session="true" />
  ```

- **buffer:** Holds the data stream size through which the application will pass data to the browser

  Syntax:

  ```
  <jsp:directive.page buffer="8kb" />
  ```

- **autoFlash:** Flushes output buffer when full

  Syntax:

  ```
  <jsp:directive.page autoFlash="true" />
  ```

- **isThreadSafe:** States whether JSP engine can pass multiple requests simultaneously to the page

  Syntax:

  ```
  <jsp:directive.page isThreadSafe="true" />
  ```

- **info:** Is used by the developer to add information for a page i.e. it is used by the developer to add details such as author, version, copyright, date and so on

  Syntax:

  ```
  <jsp:directive.page info="<Message>" />
  ```

- **errorPage:** A page or context relative URI path to a JSP page, Servlet or a static page to redirect to in case an exception is thrown by code spec in the current page

  Syntax:

  ```
  <jsp:directive.page errorPage="<pathToErrorPage>" />
  ```

- **isErrorPage:** Indicates whether or not the current page can act as the error page for another JSP page

  Syntax:

  ```
  <jsp:directive.page isErrorPage="false" />
  ```

- **contentType:** Sets the mime type and character set of the JSP page

  Syntax:

  ```
  <jsp:directive.page contentType="text/html; charset=Shift_JIS" />
  ```

- **pageEncoding:** Is the page source character encoding i.e. it is the character set of the current page

Syntax:

```
<jsp:directive.page pageEncoding="ISO-8859-1" />
```

The include Directive

This directive allows including files at the time the JSP page is translated into a Servlet. The included files are usually used for navigation, tables, headers and footers that are common to multiple pages.

Syntax:

```
<jsp:directive.include file="<Filename/Relative URL>" />
```

The URL specified is usually interpreted relative to the JSP page that refers to it. With relative URLs in general, the system can be asked to interpret the URL relative to the home directory of the Web server by beginning the URL with a forward slash.

Example:

```
<HTML>
  <HEAD>
      <TITLE>JSP include Directive Example</TITLE>
  </HEAD>
  <BODY>
      <jsp:directive.include file="/navigate.html " />
  </BODY>
</HTML>
```

Following is the only attribute associated with the include directive:

- **file:** A page-relative or context-relative URI path to the file that will be included at the current position in the file

  Syntax:

  ```
  <jsp:directive.include file="blogs.html" />
  ```

Example:
The contents of the **include.jspx** file are:

```
<HTML>
  <HEAD>
      <TITLE>JSP include directive Test</TITLE>
  </HEAD>
  <BODY>
      <FONT COLOR="Red">
          Current date and time: <jsp:directive.include file="date.jsp" />
      </FONT>
  </BODY>
</HTML>
```

The contents of the **date.jspx** file are:

```
<jsp:directive.page import="java.util.*" />
<jsp:expression> (new java.util.Date()).toLocaleString() </jsp:expression>
```

Explanation:
In the above example, **include.jspx** is the main JSP document. In include.jspx, **include** directive is used to include the **date.jspx** file. Here, an include directive is used to insert a JSP file. The JSP document recompiles, in case the included file changes.
In date.jspx, **page** directive is used to import the required package. The next line of the date.jspx file uses an expression statement to print the current date and time.
When the main file i.e. the include.jspx file is run, the output sent to the browser is as follows:

```
The current date and time are Aug 28, 2006 3:09:27 PM
```

The taglib Directive

A taglib is a collection of Custom tags that can be used by the page. In other words, this directive allows the page to use **Custom tags** inside the JSP page.

Syntax:

```
<jsp:directive.taglib uri="<TagLibraryURI>" prefix="<TagPrefix>" />
```

Following are the attributes associated with the taglib directive:

- **uri:** A Uniform Resource Identifier [URI] that identifies the tag library descriptor, which is used to uniquely name the set of custom tags and inform the server what to do with the specified tags

  Syntax:

  ```
  <jsp:directive.taglib uri="http://www.sharanamshah.com/tags" />
  ```

- **prefix:** Defines the prefix string in <prefix>:<tagname> pair. This is used to define the custom tag. It is the prefix, which is used in the action element names for all actions in the library

  Syntax:

  ```
  <jsp:directive.taglib prefix="mytag" />
  ```

Scripting Elements

Scripting elements are used to include scripting code spec, which is the usual Java code spec, within a JSP page

Following are the type of Scripting Elements available in JSP:

- Declarations
- Scriptlets
- Expressions

Declarations

A declaration is a Java code block that holds code spec for class-wide variables and methods declarations in the generated class file.

Syntax:

```
<jsp:declaration>
    Java variable and method declaration(s)
</jsp:declaration>
```

Example:

```
<jsp:declaration>
    int num = 0;
</jsp:declaration>
```

Methods can be declared as follows:

```
<jsp:declaration>
    public void countNum()
    {
        int num = 10;
    }
</jsp:declaration>
```

Scriptlets

A Scriptlet is a block of Java code spec that is executed at run time. A Scriptlet can produce output passed using an output stream back to the client.

Syntax:

```
<jsp:scriptlet> Scriptlet Code Spec </jsp:scriptlet>
```

There can be just about any valid Java code within a scriptlet. For example, the following displays a string Hello World using H1, H2 and H3 HTML elements, combining the use of expressions and scriptlets:

```
<jsp:scriptlet>
    for (int num=1; num<=3; num++)
    {
</jsp:scriptlet>
    <H<%=num%>>Hello World!</H<%=num%>>
<jsp:scriptlet>
    }
</jsp:scriptlet>
```

Since Java statements are included in the scriplets, the following code spec is a valid scriptlet:

```
<jsp:scriptlet>
    int num=0;
    /* Some code */
</jsp:scriptlet>
```

The above code looks similar to that of declaration element, but scriptlets are different from declarations in the following ways:

- Methods can be defined only by Declarations and not by Scriptlets
- Variables declared in a Scriptlet are **local** to a method in the JSP page implementation class i.e. they are visible only within their defining code block. Variables declared in a Declaration are instance variables of the JSP page implementation class i.e. these variables are visible to all other code statements or methods in the page

Expressions

An **expression** is a shorthand notation for a scriptlet that outputs a value to a response stream bound to a client. JSP expression element is explicitly used for output generation.

Syntax:

```
<jsp:expression> Expressions to be evaluated </jsp:expression>
```

Example:

```
Date: <jsp:expression> new java.util.Date() </jsp:expression>
```

Explanation:
The above example display the current date and time.

Output

```
Date: Tue Aug 29 12:25:59 GMT+05:30 2006
```

A Sample JSP Document

The following describes how the earlier example is written using XML syntax in a JSP Document. This means all the directives and the scripting elements used in the earlier example is now written in the XML syntax thus making it a JSP Document.

A JSP document must be identified to the web server so that the server interprets it as an XML document. Following are the methods to achieve the same:

1. Include a <**jsp:root**> element in the JSP document. This method is backward-compatible with JSP 1.2
2. Use a Java Servlet Specification version 2.4 or above web.xml file and name the JSP document using .jspx extension
3. In the application's web.xml file, set the **<is-xml>** element of the **<jsp-property-group>** element to **true**

The first method: Include a <jsp:root> element in the JSP document. This method is backward-compatible with JSP 1.2

This example therefore requires creating the following files:

- AuthorMasterForm.jsp: The User Interface
- header.jsp: The menu bar [To demonstrate the include directive]
- AuthorMaster.jsp: The file that processes the data captured by d/e form contained in AuthorMasterForm.jsp

Perform the following steps:

Create a JSP **document** named AuthorMasterForm.jsp with the following code spec. This file after it is created is placed under the **jsp** directory of the same application named **MyWebApplication** that was deployed in the *Chapter 13: Getting Started With Java Server Pages*.

```
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0">
   <HTML>
      <HEAD>
         <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=UTF-8" />
         <TITLE>Author Information</TITLE>
```

```
</HEAD>
<BODY BGCOLOR="pink">
   <jsp:directive.include file="header.jsp" />
<!-- Initializing a form object, which will submit data
captured on the form. -->
   <FORM ACTION="/MyWebApplication/MyJSP/AuthorMasterJSPDocument"
         METHOD="post" NAME="FrmAuthMast">
   <!-- Parent Table begins. -->
      <TABLE ALIGN="center" BGCOLOR="PINK" BORDER="0"
            CELLPADDING="0" CELLSPACING="0" NAME="tblOuter"
            WIDTH="50%">
         <TR HEIGHT="300" VALIGN="top">
            <TD ALIGN="center" BORDER="1" COLSPAN="10"><BR />
               <TABLE ALIGN="center" BORDER="1"
                     BORDERCOLOR="SKYBLUE" CELLPADDING="2"
                     CELLSPACING="0" NAME="tblFirstChild"
                     WIDTH="100%">
               <!-- Inserting table row to display the form
               title -->
                  <TR BGCOLOR="BLACK">
                     <TD ALIGN="left" COLSPAN="2">
                     <!-- Formatting the font of title -->
                        <FONT SIZE="2" COLOR="white"><B>Author
                              Master</B></FONT>
                     </TD>
                  </TR>
               <!-- Inserting table row that holds table
               data label and text box of Author Name -->
                  <TR>
                     <TD ALIGN="right" WIDTH="25%">Author Name:</TD>
                     <TD ALIGN="left">
                        <INPUT MAXLENGTH="35"
                              NAME="txtAuthorName" TYPE="text"
                              SIZE="25" />
                     </TD>
                  </TR>
               <!-- Inserting table row that holds table
               data label and text box of Degree -->
                  <TR>
                     <TD ALIGN="right">Degree:</TD>
                     <TD ALIGN="left">
                        <INPUT MAXLENGTH="255" NAME="txtDegree"
                              TYPE="text" SIZE="42" />
                     </TD>
                  </TR>
               <!-- Inserting table row that holds table
               data label and text box of Speciality -->
```

```
                    <TR>
                        <TD ALIGN="right">Speciality:</TD>
                        <TD ALIGN="left">
                            <INPUT MAXLENGTH="255" NAME="txtSpeciality"
                                TYPE="text" SIZE="42" />
                        </TD>
                    </TR>
                    <!-- Inserting table row that holds table
                    data label and text box of Date of birth -->
                    <TR>
                        <TD ALIGN="right">Date Of Birth:</TD>
                        <TD ALIGN="left">
                            <INPUT MAXLENGTH="255" NAME="txtDOB"
                                TYPE="text" SIZE="42" />
                        </TD>
                    </TR>
                    <!-- Inserting table row that holds inputs
                    for Save button -->
                    <TR>
                        <TD COLSPAN="2" ALIGN="RIGHT">
                            <INPUT NAME="cmdSubmit" TYPE="submit"
                                VALUE="Save" />
                        </TD>
                    </TR>
                </TABLE>
            </TD>
        </TR>
    </TABLE>
    </FORM>
  </BODY>
</HTML>
</jsp:root>
```

Explanation:
This is a basic HTML form contained within a JSP file. The most significant parts of the code spec are:

1. The **<jsp:root>** element:

```
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0"> . . . </jsp:root>
```

The <jsp:root> element represents the root element of a JSP document. The <jsp:root> element gives a signal to the JSP engine that the current page is a JSP document and the tags used in this page are **not the valid syntax** for an element in the **JSP page**.

The <jsp:root> element is also responsible for specifying appropriate namespaces available to the document such as the standard namespace, which defines the JSP specifications and other tag libraries, which define custom tag actions. This means tag libraries are included in the JSP document using the xmlns attribute of the root element instead of using the taglib directive.

For example, in this case xmlns**:jsp**="http://java.sun.com/JSP/Page" is used to make available the JSP tag library.

The <jsp:root> element has two attributes viz**. xmlns** and **version** attributes.

The **xmlns** attribute is optional. This attribute helps associate the jsp prefix with elements defined as JSP standard actions.

The **version** attribute is the required attribute and is used to specify the JSP specification version that the JSP document is using, in this case the version being **2.0**.

Prior to JSP 2.0, a JSP document always required a **<jsp:root>** element. This was just to inform the server about the type of JSP page it was. However, after the introduction of JSP 2.0, this limitation was lifted by defining following ways to identify a file as a JSP Document:

- The request path matches the URL pattern in the web.xml's JSP property group declaration with an **<is-xml>** element set to true
- The request path extension is .jspx, unless this extension matches the URL pattern for a JSP property group declaration with an **<is-xml>** element set to false. In short, .jspx is the default extension for JSP Documents, but it can be explicitly disabled by a property group declaration
- The request path extension is either .jsp

  OR

  Request path matches a URL pattern for a JSP property group declaration

  AND

  The root element in the file is <jsp:root>

HINT

Even though the **<jsp:root>** element is not compulsory, it is still useful to identify a page as a JSP document to the JSP server without having to add any configuration attributes to the deployment descriptor [web.xml] or name the document with a .jsp**x** extension.

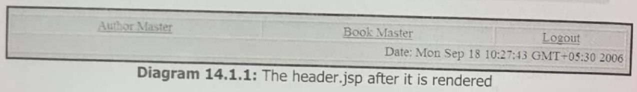
It is also possible to write a JSP Document as a regular HTML file that includes JSP elements [for dynamic content], for instance, without having to place all content within a <jsp:root> element.

The .html can be used as an extension for such files, if a JSP property group declaration is created:

```
...
<jsp-config>
    <jsp-property-group>
        <url-pattern>*.html</url-pattern>
        <is-xml>true</is-xml>
    </jsp-property-group>
</jsp-config>
...
```

2. A header is included, which, when rendered / executed will be displayed as shown in diagram 14.1.1. The header code spec is held inside the header.jsp file

```
<jsp:directive.include file="header.jsp" />
```



Author Master | Book Master | Logout
Date: Mon Sep 18 10:27:43 GMT+05:30 2006

Diagram 14.1.1: The header.jsp after it is rendered

The <jsp:directive.include> element is used to include the text contained in another file i.e. header.jsp into the including JSP document i.e. the AuthorMasterForm.jsp file.

HINT

The XML view of a JSP document does not contain the **<jsp:directive.include>** element, but the included file is expanded in place. This is done to simply validate.

In addition to including JSP documents in other JSP documents, JSP pages written in standard syntax can also be included in other JSP documents and JSP documents can also be included in other JSP pages written in standard syntax. The server detects the page included and parses it as either a standard syntax JSP page or a JSP document and then places it into the XML view for validation.

Create a JSP document named header.jsp with the following code spec. This file after it is created is placed under a new directory named **jsp** in the same application named **MyWebApplication** that was deployed in the *Chapter 13: Getting Started With Java Server Pages*:

```
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0">
```

```
    <jsp:directive.page language="java" import="java.util.*" />
    <TABLE BORDER="1" ALIGN="center" WIDTH="100%">
        <TR>
            <TD ALIGN="center"><A HREF="AuthorMasterForm.jsp">Author Master</A>
            </TD>
            <TD ALIGN="center"><A HREF="BookMaster.html">Book Master</A></TD>
            <TD ALIGN="center"><A HREF="javascript:alert('Goodbye');">Logout</A></TD>
        </TR>
        <TR>
            <TD COLSPAN="3" ALIGN="right">
                Date: <jsp:expression> new java.util.Date() </jsp:expression>
            </TD>
        </TR>
    </TABLE>
</jsp:root>
```

Explanation:
The most significant parts of the code specs are:

1. The page directive in the JSP Document:

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

The <jsp:directive.page> element must be a child of the root element i.e. it must appear at the beginning of the JSP document but inside the <jsp:root> element. The <jsp:directive.page> element defines a number of page-dependent properties and passes those values to the JSP engine.
The value of **language** attribute is **java**, which is the language used by the current JSP document i.e. header.jsp. The **page** directive will import the **java.util.*** API into the current JSP document i.e. header.jsp through the **import** attribute.

2. A menu is formed with the following items:

- **AuthorMaster:** Is a Link to the AuthorMasterForm.jsp file
- **BookMaster:** Is a Link to the BookMaster.html file [created earlier in *Chapter 8: Working With Servlets*]
- **Logout:** Displays a message "Goodbye"

3. JSP Document Expression:

```
Date: <jsp:expression> new java.util.Date() </jsp:expression>
```

The <jsp:expression> element is used to describe complete expressions that get evaluated at runtime. The <jsp:expression> element does not have any attributes and the body of the element is the expression. This expression is used to display the current date and time.

Create a JSP document named AuthorMaster.jsp with the following code spec. This is the file that processes the data captured by the Author Master Form [AuthorMasterForm.jsp]. This file after it is created is placed under a new directory named **jsp** in the same application named **MyWebApplication** that was deployed in the *Chapter 13: Getting Started With Java Server Pages*:

```
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0">
   <jsp:directive.page language="java" contentType="text/html" />
   <HTML>
        <HEAD>
             <TITLE>Author Information</TITLE>
        </HEAD>
        <BODY>
             <jsp:declaration>
                  String AuthorName;
                  String Degree;
                  String Speciality;
                  String DOB;
             </jsp:declaration>
             <jsp:scriptlet>
                  AuthorName = request.getParameter("txtAuthorName");
                  Degree = request.getParameter("txtDegree");
                  Speciality = request.getParameter("txtSpeciality");
                  DOB = request.getParameter("txtDOB");
             </jsp:scriptlet>
             <H1>This is Author Information</H1>
             Author Name: <jsp:expression>AuthorName</jsp:expression><BR />
             Degree: <jsp:expression>Degree</jsp:expression><BR />
             Speciality: <jsp:expression>Speciality</jsp:expression><BR />
             Date of Birth: <jsp:expression>DOB</jsp:expression>
        </BODY>
   </HTML>
</jsp:root>
```

Explanation:
Most of the file's content is plain HTML but it also has, interspersed within it, special JSP [XML Style] tags.

```
<jsp:directive.page language="java" contentType="text/html" />
```

This is a page directive with **language** and **contentType** attributes set.

```
<jsp:declaration>
     String AuthorName;
     String Degree;
     String Speciality;
     String DOB;
</jsp:declaration>
```

The <jsp:declaration> element is used to declare scripting-level constructs that are available to all other scripting elements. This element does not have any attributes and the body of the element is the declaration(s).

In the above example, four variables named AuthorName, Degree, Speciality and DOB are declared which is available throughout the JSP document, to other declarations, expressions or code blocks.

```
<jsp:scriptlet>
     AuthorName = request.getParameter("txtAuthorName");
     Degree = request.getParameter("txtDegree");
     Speciality = request.getParameter("txtSpeciality");
     DOB = request.getParameter("txtDOB");
</jsp:scriptlet>
```

The <jsp:scriptlet> element is used to insert fragments of code. This element does not have any attributes and the body of the element is the program fragment.

In the above example, a scriptlet assigns the data captured by the Author Master form to the four variables [declared earlier]. The data captured is available in the POST array and accessed using **request.getParameter()**.

```
Author Name: <jsp:expression>AuthorName</jsp:expression>
Degree: <jsp:expression>Degree</jsp:expression>
Speciality: <jsp:expression>Speciality</jsp:expression>
Date of Birth: <jsp:expression>DOB</jsp:expression>
```

To display the captured data which is held inside memory variables **<jsp:expression> . . . </jsp:expression>** is used. In side the expression tag of the scripting element the variable declared is accessed.

Once the files are ready and placed appropriately the deployment can begin.

To do so **web.xml** file will undergo some changes as follows:

1. Append the following lines [marked in dark shade] in the web.xml file:

```
<?xml version="1.0"?>
<web-app version="2.5" 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-app_2_5.xsd">
  <servlet>
       <servlet-name>BookInformation</servlet-name>
       <display-name>BookInformation</display-name>
       <servlet-class>BookInformation</servlet-class>
  </servlet>
```

```
    <servlet>
        <servlet-name>AuthorMasterJSPDocument</servlet-name>
        <display-name>AuthorMasterJSPDocument</display-name>
        <jsp-file>AuthorMasterDoc.jsp</jsp-file>
    </servlet>
    <servlet-mapping>
        <servlet-name>BookInformation</servlet-name>
        <url-pattern>/MyServlets/BookInformation</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>AuthorMasterJSPDocument</servlet-name>
        <url-pattern>/MyJSP/AuthorMasterJSPDocument</url-pattern>
    </servlet-mapping>
</web-app>
```

Explanation:

```
<jsp-file>AuthorMasterDoc.jsp</jsp-file>
```

In the above code spec, <jsp-file> element holds the path to a JSP file within the web application. Since in this case, the AuthorMaster.jsp file is placed in the root of the web application after it is deployed no path is mentioned.

1. Now build the web application using the **asant** utility
2. Next, deploy the web application
3. Run the deployed web application which now holds the AuthorMasterForm.jsp, AuthorMaster.jsp and header.jsp in addition to the previous examples using the URL http://localhost:8080/MyWebApplication/AuthorMasterForm.jsp as shown in diagram 14.1.2

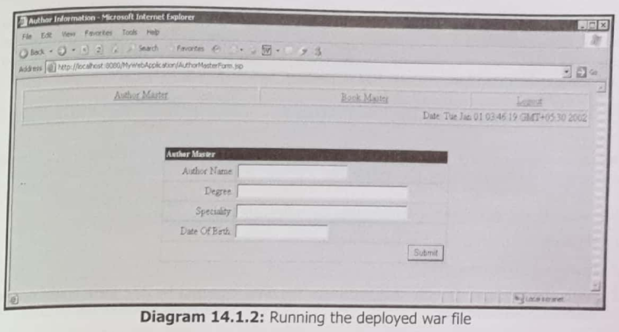
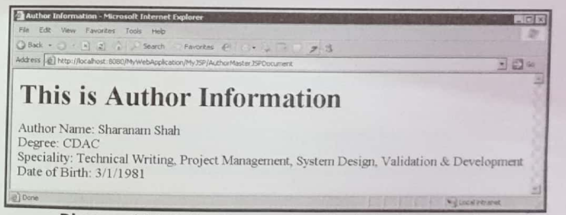

Author Information - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/MyWebApplication/AuthorMasterForm.jsp
Author Master
Book Master
Logout
Date: Tue Jan 01 03:46:19 GMT+05:30 2002
Author Master
Author Name
Degree
Speciality
Date Of Birth
Submit
Local intranet

Diagram 14.1.2: Running the deployed war file

REMINDER

The AuthorMasterForm.jsp file when executed as shown in diagram 14.1.2 also displays the header.jsp page, which holds the menu bar as defined earlier.

Enter the desired data and click Submit. The AuthorMasterForm.jsp file sends the captured data [using the post method] to the AuthorMaster.jsp file for further processing. The AuthorMaster.jsp file displays the data captured as shown in diagram 14.1.3.


Author Information - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/MyWebApplication/MyJSP/AuthorMasterJSPDocument
Done
Local intranet

This is Author Information

Author Name: Sharanam Shah
Degree: CDAC
Speciality: Technical Writing, Project Management, System Design, Validation & Development
Date of Birth: 3/1/1981

Diagram 14.1.3: The Response from the AuthorMaster.jsp file

Since the control is passed on to the JSP file, the URL changes to http://localhost:8080/MyWebApplication/MyJSP/AuthorMasterJSPDocument. This is the same URL that was specified in the ACTION attribute of the FORM element of the AuthorMasterForm.jsp file:

```
<FORM ACTION="/MyWebApplication/MyJSP/AuthorMasterJSPDocument"
          METHOD="post" NAME="FrmAuthMast">
```

The second method: Use a Java Servlet Specification version 2.4 or above web.xml file and name the JSP document using .jspx extension

This example will therefore require creating the following files:

- AuthorMasterForm.jsp**x**: The User Interface
- header.jsp**x**: The menu bar
- AuthorMaster.jsp**x**: The file that processes the data captured by d/e form contained in AuthorMasterForm.jspx

Perform the following steps to use the second method:

1. Remove the <jsp:root> opening and closing elements from the following files:
   - AuthorMasterForm.jsp
   - header.jsp
   - AuthorMaster.jsp
2. Rename the above mentioned files using .jspx extension
3. Modify the AuthorMasterForm.jspx as follows [modifications marked in **bold**]:

```
<HTML xmlns:jsp="http://java.sun.com/JSP/Page">
  <HEAD>
      <jsp:directive.page language="java" contentType="text/html" />
      <TITLE>Author Information</TITLE>
  </HEAD>
  <BODY BGCOLOR="pink">
      <jsp:directive.include file="headerDoc.jspx" />
. . .
. . .
```

4. Modify the AuthorMaster.jspx as follows [modifications marked in **bold**]:

```
<HTML xmlns:jsp="http://java.sun.com/JSP/Page">
  <HEAD>
      <jsp:directive.page language="java" contentType="text/html" />
      <TITLE>Author Information</TITLE>
```

```
  </HEAD>
. . .
. . .
```

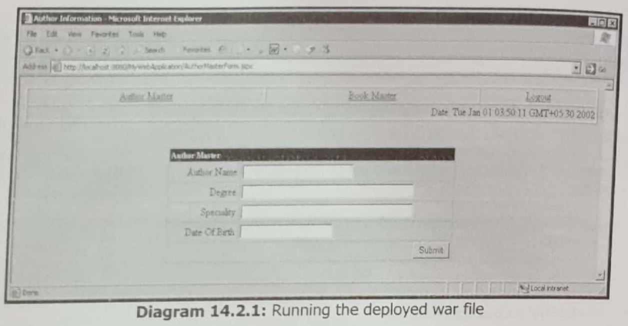
5. Modify the header.jspx as follows [modifications marked in **bold**]:

```
<TABLE BORDER="1" ALIGN="center" WIDTH="100%"
          xmlns:jsp="http://java.sun.com/JSP/Page">
  <jsp:directive.page language="java" contentType="text/html" import="java.util.*" />
  <TR>
      <TD ALIGN="center">
          <A HREF="AuthorMasterFormDoc.jspx">Author Master</A>
      </TD>
. . .
. . .
```

6. Modify the web.xml as follows:

```
<?xml version="1.0"?>
<web-app version="2.5" 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-app_2_5.xsd">
  <servlet>
. . .
. . .
  <servlet>
      <servlet-name>AuthorMasterJSPDocument</servlet-name>
      <display-name>AuthorMasterJSPDocument</display-name>
      <jsp-file>AuthorMaster.jspx</jsp-file>
  </servlet>
. . .
. . .
```

7. Now build the web application using the **asant** utility
8. Next, deploy the web application
9. Run the deployed web application which now holds the AuthorMasterForm.jsp**x**, AuthorMaster.jsp**x** and header.jsp**x** in addition to the previous examples using the URL http://localhost:8080/MyWebApplication/AuthorMasterForm.jsp**x** as shown in diagram 14.2.1


Author Information - Microsoft Internet Explorer

http://localhost:8080/MyWebApplication/AuthorMasterForm.jspx

Author Master | Book Master | Logout

Date: Tue Jan 01 03:50:11 GMT+05:30 2002

Author Master

Author Name

Degree

Speciality

Date Of Birth

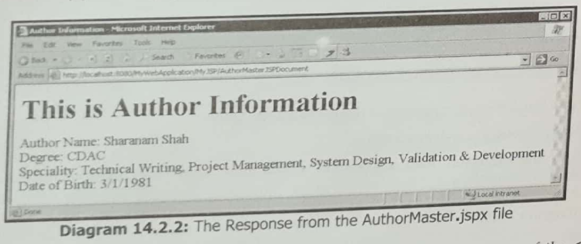
Submit

Diagram 14.2.1: Running the deployed war file

REMINDER

The AuthorMasterForm.jsp**x** file when executed as shown in diagram 14.2.1 also displays the header.jsp**x** page, which holds the menu bar as defined earlier.

Enter the desired data and click Submit. The AuthorMasterForm.jsp**x** file sends the captured data [using the post method] to the AuthorMaster.jsp**x** file for further processing. The AuthorMaster.jsp**x** file displays the data captured as shown in diagram 14.2.2.


Author Information - Microsoft Internet Explorer

http://localhost:8080/MyWebApplication/MyJSP/AuthorMasterJSPDocument

This is Author Information

Author Name: Sharanam Shah
Degree: CDAC
Speciality: Technical Writing, Project Management, System Design, Validation & Development
Date of Birth: 3/1/1981

Diagram 14.2.2: The Response from the AuthorMaster.jspx file

The third method: In the application's web.xml file, set the **<is-xml>** element of the **<jsp-property-group>** element to **true**.

This example therefore requires creating the following files:

- AuthorMasterForm.jsp: The User Interface
- header.jsp: The menu bar [To demonstrate the include directive]
- AuthorMaster.jsp: The file that processes the data captured by d/e form contained in AuthorMasterForm.jsp

Perform the following steps to use the third method:

1. Remove the <jsp:root> opening and closing elements from the following files:
   - AuthorMasterForm.jsp
   - header.jsp
   - AuthorMaster.jsp
2. Rename the above mentioned files using .jsp extension
3. Modify the AuthorMasterForm.jsp as follows [modifications marked in **bold**]:

```
<jsp:directive.page language="java" contentType="text/html" />
<HTML>
   <HEAD>
         <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=UTF-8" />
         <TITLE>Author Information</TITLE>
   </HEAD>
   <BODY BGCOLOR="pink">
         <jsp:directive.include file="header.jsp" />
         <FORM ACTION="AuthorMaster.jsp" METHOD="post" NAME="FrmAuthMast">
. . .
. . .
```

4. Modify the AuthorMaster.jsp as follows [modifications marked in **bold**]:

```
<jsp:directive.page language="java" contentType="text/html" />
<HTML>
   <HEAD>
         <TITLE>Author Information</TITLE>
   <HEAD>
. . .
. . .
```

5. Modify the header.jsp as follows [modifications marked in **bold**]:

```
<jsp:directive.page language="java" contentType="text/html" import="java.util.*" />
<TABLE BORDER="1" ALIGN="center" WIDTH="100%">
   <TR>
         <TD ALIGN="center">
               <A HREF="AuthorMasterForm.jsp">Author Master</A>
```

```
        </TD>
...
...
```

6. Modify the web.xml as follows:

```
<?xml version="1.0"?>
<web-app version="2.5" 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-app_2_5.xsd">
   <jsp-config>
        <jsp-property-group>
             <is-xml>true</is-xml>
        </jsp-property-group>
   </jsp-config>
   <servlet>
        <servlet-name>BookInformation</servlet-name>
   <servlet>
        <servlet-name>BookInformation</servlet-name>
        <display-name>BookInformation</display-name>
        <servlet-class>BookInformation</servlet-class>
   </servlet>
   <servlet-mapping>
        <servlet-name>BookInformation</servlet-name>
        <url-pattern>/MyServlets/BookInformation</url-pattern>
   </servlet-mapping>
</web-app>
```

7. Now build the web application using the **asant** utility
8. Next, deploy the web application
9. Run the deployed web application which now holds the AuthorMasterForm.jsp, AuthorMaster.jsp and header.jsp in addition to the previous examples using the URL http://localhost:8080/MyWebApplication/AuthorMasterForm.jsp as shown in diagram 14.3.1

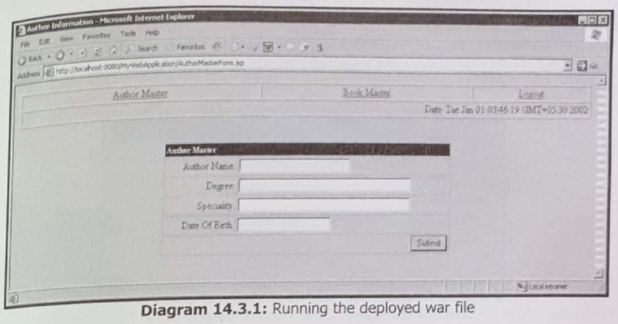
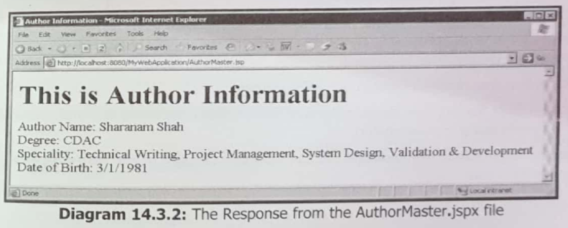

Author Information - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/MyWebApplication/AuthorMasterForm.jsp
Author Master
Book Master
Logout
Date Tue Jan 01 03:46:19 GMT+05:30 2002
Author Master
Author Name
Degree
Speciality
Date Of Birth
Submit
Local intranet

Diagram 14.3.1: Running the deployed war file

REMINDER

The AuthorMasterForm.jsp file when executed as shown in diagram 14.3.1 also displays the header.jsp page, which holds the menu bar as defined earlier.

Enter the desired data and click Submit. The AuthorMasterForm.jsp file sends the captured data [using the post method] to the AuthorMaster.jsp file for further processing. The AuthorMaster.jsp file displays the data captured as shown in diagram 14.3.2.


Author Information - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/MyWebApplication/AuthorMaster.jsp
Done
Local intranet

This is Author Information

Author Name: Sharanam Shah
Degree: CDAC
Speciality: Technical Writing, Project Management, System Design, Validation & Development
Date of Birth: 3/1/1981

Diagram 14.3.2: The Response from the AuthorMaster.jspx file

REMINDER

Action Elements are discussed in Chapter 15: Action Elements Of JSP.

Hands On Exercises

Create a user interface page for capturing Book information. This form will accept data from the user and display the same upon page submission i.e. when the user enters the required data and clicks **Save**.

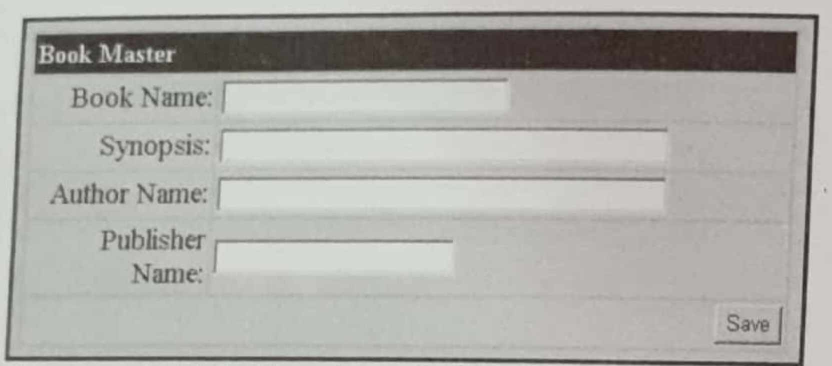
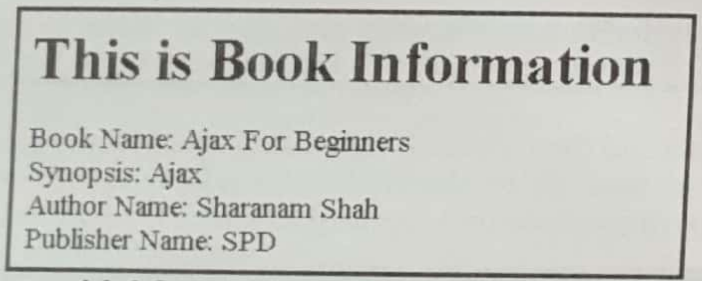


Diagram 14.4.1: Form Layout of the BookMaster.html file

When the data captured is submitted, the control passes to a **JSP document** called **BookMaster**. This document displays the data captured by the HTML form.



This is Book Information

Book Name: Ajax For Beginners
Synopsis: Ajax
Author Name: Sharanam Shah
Publisher Name: SPD

Diagram 14.4.2: Data displayed by the BookMaster.jsp file

Perform the above exercise using all the three methods of the JSP Document.
Following are the methods to be used:

1. Include a **<jsp:root>** element in the JSP document
2. Use a Java Servlet Specification version 2.4 or above web.xml file and name the JSP document using .jspx extension
3. In the application's web.xml file, set the **<is-xml>** element of the **<jsp-property-group>** element to **true**