Module 2 : Java server pages (JSP)

**1. What is JSP and why do we need it?**

JSP stands for JavaServer Pages. JSP is java server side technology to create dynamic web pages. JSP is extension of Servlet technology to help developers create dynamic pages with HTML like syntax.

It is a presentation layer technology independent of platform. It comes with SUN’s J2EE platforms. They are like HTML pages but with Java code pieces embedded in them. They are saved with a .jsp extension. They are compiled using JSP compiler in the background and generate a Servlet from the page.

JSP is a standard extension of Java and is defined on top of Servlet extensions. Its goal is to simplify management and creation of dynamic web pages. It is platform-independent, secure, and it makes use of Java as a server side scripting language.

**2. What does dynamic web page?**

The web pages that are generated based on user’s response and may be different for each user are called dynamic web pages unlike the static web pages that are same for every user no matter how they interact with the application.

**3. What does Server side technology means?**

There are basically two types of technologies: client-side and Server-side. Client-side means that the action takes place on the user’s (the client’s) computer. Server-side means that the action takes place on a web server (the place where you have stored all your JSP pages).

DR. RAJESHREE KHANDE 1 | 17

Module 2 : Java server pages (JSP)

**4. How does a servlet differ from the JSP?**

The JSP and Servlets are both web components and functionally similar. In fact, a JSP gets transformed into the servlet at run-time. The main variation between the two is that JSPs are

useful for operations that are HTML-intensive whereas the Servlets are best-suited for tasks that are Java-intensive. The below example can elaborate the difference clearly.

**Example:** A web application which follows the MVC model separates the components into three groups.

∙ **Model-** Represents the modules that contain application data e.g. Java beans. ∙ **Views-** It is the part of the application that displays information. e.g. dynamically generated HTML, JSP handles this part very well.

∙ **Controllers-** Its job is to work as a carrier between the model and the view. The controller places the user entered information into the model and forwards it to the next view. Servlets fulfill this purpose perfectly.

**5. What are the advantages of JSP over Servlet?**

JSP is a server-side construct to simplify the dynamic generation of the HTML content. One of its advantages is that JSP is document-centric. While the Servlets purely behave like the programs. A JSP Page can consist of Java code fragments that can instantiate and invokes the methods of Java classes. All of this occur inside an HTML template file which is mainly used to produce dynamic content. However, some of the JSP functionality can be achieved at the client-end using JavaScript. The power of JSP is that it is server-based and provides a framework for Web application.

DR. RAJESHREE KHANDE 2 | 17

Module 2 : Java server pages (JSP)

**Advantages of JSP.**

∙ JSP represents an HTML page embed with Java code.

∙ JSP is cross-platform technology.

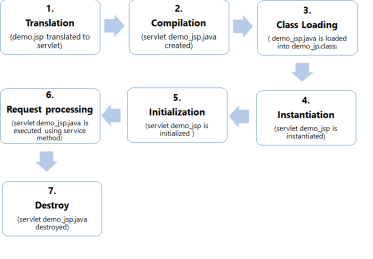
∙ JSP can create database-driven Web applications.

∙ JSP enables Server-side programming abilities.

**6. What are the JSP lifecycle phases?**

When first time a JSP page is requested, the necessary servlet code is generated and loaded into the servlet container. Now until the JSP page gets rendered fully, the servlet code handles the requests coming from the browser. Whenever there is a change in the JSP page, the JSP compiler regenerates the servlet code.

If you will observe the JSP page code, it looks like as HTML and doesn’t resemble the Java classes. JSP container takes care of the JSP page translation and creates the servlet class that the web application uses. Let’s now dig into the different JSP lifecycle phases:

DR. RAJESHREE KHANDE 3 | 17

Module 2 : Java server pages (JSP)

1. **Translation** – JSP container verifies the JSP page code and parses it to generate the servlet source code. For your note, if the JSP page name is <Login.jsp>, the generated servlet class name would be like Login\_jsp, and the Java file name would be Login\_jsp.java.

2. **Compilation** – JSP container compiles the JSP page and creates a class file in this phase. 3. **Class Loading** – At this stage, the container reads the JSP class into application memory.

4. **Instantiation** – This phase guides the container to execute the no-args constructor of the generated JSP class and loads it into memory to instantiate it.

5. **Initialization** – Container calls the JSP class init method and initializes the servlet config with the parameters specified in the deployment descriptor. This phase makes sure the JSP is fit to respond to the client requests.

6. **Request Processing** – This is the phase where the JSP page creates threads to processes a request. Every new thread leads to the creation of ServletRequest and ServletResponse object followed by a call to the JSP service method.

7. **Destroy** – This is the terminal phase of JSP life cycle where the JSP class gets unloaded from the memory. This stage comes in the picture when the application is undeployed, or the server is shut down.

**7. What are JSP lifecycle methods?**

∙ The JSP container executes the **jspInit()** method the first time it initializes the JSP.

∙ Whenever the container receives a request, it invokes **the jspService()** method, processes the request and generates a response.

∙ The container calls the **jspDestroy()** method for cleaning up the memory allocated for the JSP.

DR. RAJESHREE KHANDE 4 | 17

Module 2 : Java server pages (JSP)

**8. Explain implicit objects in JSP?**

Implicit objects in JSP are pure Java objects. JSP Container makes these objects available to the developers on each JSP page. There is no need to declare or instantiate these objects explicitly by the JSP developer. JSP has a set of standard variables to access these objects. Hence, they are called implicit objects. Below is an exclusive list of the implicit objects that the JSP offers:

| **Implicit**  **Object** | **Description** |
| --- | --- |
| **request** | The **HttpServletRequest** object associated with the request. |
| **response** | The **HttpServletRequest** object associated with the response that is sent back to the browser. |
| **out** | The **JspWriter** object associated with the output stream of the response. |
| **session** | The **HttpSession** object associated with the session for the given user of request. |
| **application** | The **ServletContext** object for the web application. |
| **config** | The **ServletConfig** object associated with the servlet for current JSP page. |
| **pageContext** | The **PageContext** object that encapsulates the enviroment of a single request for this current JSP page |
| **page** | The **page** variable is equivalent to **this** variable of Java programming language. |
| **exception** | The **exception** object represents the **Throwable** object that was thrown by some other JSP page. |

DR. RAJESHREE KHANDE 5 | 17

Module 2 : Java server pages (JSP)

**9. What is Scriptlet, Expression and Declaration in JSP?**

**a. Scriptlet Tag :**

• A scriptlet tag is used to execute Java source code in JSP. • Syntax :

**<% java source code %>**

Example :

<html>

<head>

<title>Hello World</title>

</head>

<body>

<% out.print("Welcome to JSP "); %>

<% int num1=10;

int num2=40;

int num3 = num1+num2;

out.println("Scriplet Number is " +num3);

%>

</body>

</html>

**b. JSP Expression Tag** :

• Expression tag is used to print the java language expression. • To execute java language expression in JSP file expression tag is used. • In expression tag, you don’t need to write out.print() to print .java • Syntax

**<% = java expression %>**

DR. RAJESHREE KHANDE 6 | 17

Module 2 : Java server pages (JSP)

• Example :

<html>

<body>

<% out.println("The expression number is "); %>

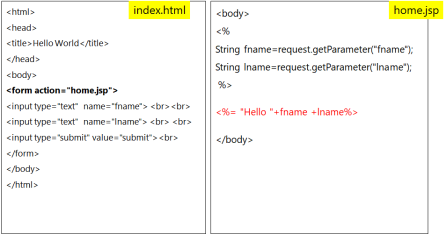
<% int num1=10; int num2=10; int num3 = 20; %>

<%= num1\*num2+num3 %>

</body>

</html>

• Example 2:



**c. Declaration Tag:**

∙ Used *to* ***declare variable methods and classes***

∙ If we declare a variable or method inside declaration tag it means that the declaration is made inside the servlet class but outside the service method.

∙ So it doesn’t get memory at each request.

DR. RAJESHREE KHANDE 7 | 17

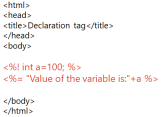
Module 2 : Java server pages (JSP)

∙ We can declare a static member, an instance variable (can declare a number or string) and methods inside the declaration tag.

∙ **Syntax :**

**<%! Scripting-language-declaration %>**

∙ Example:



**10. What is the logic behind a scriptlet in JSP?**

A scriptlet holds the executable Java code which runs whenever the JSP gets loaded. The scriptlet passes its code to the service() method while the JSP is getting compiled to a servlet. So all the scriptlet variables and methods become local to the service() method. A scriptlet is coded between the <% and %> tags and the container call it while processing the request.

**11. What do you understand of the JSP directives?**

∙ JSP directives are the instructions for the JSP container to control the processing of the whole page.

∙ They add the ability to set global values such as a class declaration, method definition, output data type, etc.

∙ They don’t send any output to the client.

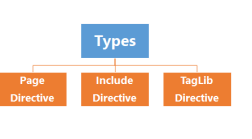
∙ All directives should get enclosed within <%@ %> tag.

∙ e.g. page and include directive, etc.

DR. RAJESHREE KHANDE 8 | 17

Module 2 : Java server pages (JSP)

∙ There are three types



| **Directive** | **Description** |
| --- | --- |
| **<%@ page ... %>** | defines page dependent properties such as language, session, errorPage etc. |
| **<%@ include ... %>** | defines file to be included. |
| **<%@ taglib ... %>** | declares tag library used in the page |

**12. What do you understand of the page directive?**

∙ It notifies the JSP container of the headers (facilities) that the page receives from the environment.

∙ Usually, the page directive remains at the top of a JSP page.

∙ A JSP page can have any number of page directives if the attribute – value pair is unique.

∙ syntax

**<%@ page attribute=”value”>**

∙ e.g.:

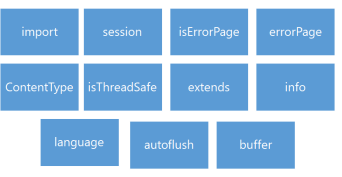
<%@ include file="header.jsp" %>

DR. RAJESHREE KHANDE 9 | 17

Module 2 : Java server pages (JSP)

**13. Explain the different attributes of a page directive?**

There are about 13-attributes available for a page directive. Some of the important ones are as follows:



| **S.No.** | **Attribute & Purpose** |
| --- | --- |
| 1 | **buffer**  Specifies a buffering model for the output stream. |
| 2 | **autoFlush**  Controls the behavior of the servlet output buffer. |
| 3 | **contentType**  Defines the character encoding scheme. |
| 4 | **errorPage** |

DR. RAJESHREE KHANDE 10 | 17

Module 2 : Java server pages (JSP)

|  | Defines the URL of another JSP that reports on Java unchecked runtime exceptions. |
| --- | --- |
| 5 | **isErrorPage**  Indicates if this JSP page is a URL specified by another JSP page's errorPage attribute. |
| 6 | **extends**  Specifies a superclass that the generated servlet must extend. |
| 7 | **import**  Specifies a list of packages or classes for use in the JSP as the Java import statement does for Java classes. |
| 8 | **info**  Defines a string that can be accessed with the servlet's **getServletInfo()** method. |
| 9 | **isThreadSafe**  Defines the threading model for the generated servlet. |
| 10 | **language**  Defines the programming language used in the JSP page. |
| 11 | **session**  Specifies whether or not the JSP page participates in HTTP sessions. |
| 12 | **isELIgnored**  Specifies whether or not the EL expression within the JSP page will be ignored. |
| 13 | **isScriptingEnabled**  Determines if the scripting elements are allowed for use. |

DR. RAJESHREE KHANDE 11 | 17

Module 2 : Java server pages (JSP)

**14. What is the include directive?**

Its purpose is to attach the static resources to the current JSP page during the translation process. The include directive syntax is as follows:

<%@ include file = “File-Name” %>

∙ The include directive statically embeds the contents of a resource into the current JSP.

∙ It allows a user to reuse the code without duplicating it and inserts the data of the target file during JSP translation.

∙ This directive has only one attribute called <file> that specifies the name of the file to include.

**15. What is the significance of the JSP standard actions and what is their purpose?**

∙ The standard actions in JSP not only control the runtime behavior of theJSP page, but they do affect the response posted back to the client-side.

∙ We use them for the following purpose:

∙ Include a file at the request time,

∙ Locate or instantiate a JavaBean,

∙ Forward a request to a new page, or

∙ Generate a browser-specific code, etc.

∙ Some of them you can see in the below example.

e.g.: include, param, useBean, etc.

**16. What are the standard actions available in JSP?**

DR. RAJESHREE KHANDE 12 | 17

Module 2 : Java server pages (JSP)

∙ **<jsp: include>**: Used to specify a response from the servlet or a JSP page into the current page.

∙ **<jsp: forward>**: Used to send a reply from the servlet/JSP page to another page.

∙ **<jsp: useBean>**: Allows a JavaBean to get accessible from a page and initializes the bean.

∙ **<jsp: setProperty>**: Used to set the JavaBean properties.

∙ **<jsp: getProperty>**: Fetches the property value from a JavaBean component and appends it to the response.

∙ **<jsp: param>**: Used with actions like <jsp:forward> and <jsp:, or plugin> to append a parameter to the request.

∙ **<jsp: plugin>**: Specifies whether to add a Java applet or a JavaBean to the current JSP page.

**17. Define the scopes available with the <jsp: useBean>?**

∙ **Page Scope:** Tells the bean object is available for the entire JSP page without any external access.

∙ **Request Scope:** Signifies the object can link with a particular request and persist till the time request lasts.

∙ **Session Scope:** States that the bean object is available throughout the session. ∙ **Application Scope:** Specifies the bean object is available throughout the entire Web application discarding any external access.

**18. What is the purpose of the <jsp: forward> action?**

∙ The <jsp: forward> standard action forwards a response from a servlet or a JSP page to another page.

∙ The execution of the current page gets stopped, and control shifts to the forwarded page.

DR. RAJESHREE KHANDE 13 | 17

Module 2 : Java server pages (JSP)

∙ The <jsp: forward> standard action uses the below synatx.

<jsp:forward page=”/targetPageTemplate” />

Here, targetPage could either be a JSP/HTML page, or a servlet within the same context.

∙ If anything gets written to the output stream but not buffered before the <jsp: forward>, it’ll result in an IllegalStateException.

Note : Before you use <jsp:forward> or <jsp:include> in a page, make sure the buffering is on. Though, the buffer is enabled by default.

**19. What is the purpose of the <jsp: include> standard action?**

The <jsp: include> standard action instructs the JSP page to include a static/dynamic resource at run-time. Unlike the include directive, the include action is best suited for the resources that undergo frequent changes. The resources you wish to add must be in the same context. The syntax of the <jsp: include> standard action is as follows:

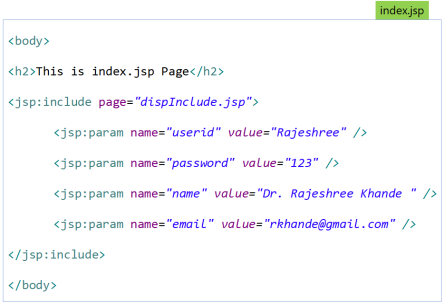
<jsp:include page="targetPage" flush="true"/>



DR. RAJESHREE KHANDE 14 | 17

Module 2 : Java server pages (JSP)

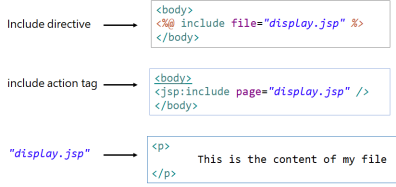
Fig Show the working of Include tag

DR. RAJESHREE KHANDE 15 | 17

Module 2 : Java server pages (JSP)

**20. Explain the main differences between an include action and the include directive?**

• **Include directive** and **include action tag** both are used for including a file to the current JSP page.

We will get the same output, Then what is difference?

| **include directive** | **include tag** |
| --- | --- |
| • Includes the file at translation time, where the JSP gets converted into the equivalent servlet | • Include action includes the file at runtime. |
| • If the included file is changed but not the JSP file which is including it then the changes will reflect not reflect | • If the included file is changed but not the JSP file which is including it then **the changes will reflect only when we use include action tag.** |
| • We can not pass the parameters to the included page | • We can also pass the parameters to the included page by using param action tag |

DR. RAJESHREE KHANDE 16 | 17

Module 2 : Java server pages (JSP)

| • Syntax  <%@ include file="file\_name" %> | • Syntax  <jsp:include page="file\_name" /> |
| --- | --- |

**21. Explain the jsp:setProperty action.**

It is used to give values to properties of beans that have been referenced beforehand. <jsp:useBean id=”ABC”.../> …

<jsp:setProperty name=”ABC” property=”myProperty”...

jsp:setproperty is executed even if a new bean is instantiated or existing bean is found.

By adding </jsp.useBean> at the end of the code, the condition for execution is inverted i.e. It is not executed if existing object was found and only if a new object was instantiated.

DR. RAJESHREE KHANDE 17 | 17