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

JSP is a serverside technology to make content generation a simple appear.The advantage of JSP is that they are document-centric. Servlets, on the other hand, look and act like programs. A Java Server Page can contain Java program fragments that instantiate and execute Java classes, but these occur inside an HTML template file and are primarily used to generate dynamic content. Some of the JSP functionality can be achieved on the client, using JavaScript. The power of JSP is that it is server-based and provides a framework for Web application development.

**2.What is the life-cycle of JSP?**

When a request is mapped to a JSP page for the first time, it translates the JSP page into a servlet class and compiles the class. It is this servlet that services the client requests.

A JSP page has seven phases in its lifecycle, as listed below in the sequence of occurrence:

* Translation.
* Compilation.
* Loading the class.
* Instantiating the class.
* jspInit() invocation.
* \_jspService() invocation.
* jspDestroy() invocation.

**3.What is the jspInit() method?**

The jspInit() method of the javax.servlet.jsp.JspPage interface is similar to the init() method of servlets. This method is invoked by the container only once when a JSP page is initialized. It can be overridden by a page author to initialize resources such as database and network connections, and to allow a JSP page to read persistent configuration data.

**4.What is the \_jspService() method?**

The \_jspService() method of the javax.servlet.jsp.HttpJspPage interface is invoked every time a new request comes to a JSP page. This method takes the HttpServletRequest and HttpServletResponse objects as its arguments. A page author cannot override this method, as its implementation is provided by the container.

**5.What is the jspDestroy() method?**

The jspDestroy() method of the javax.servlet.jsp.JspPage interface is invoked by the container when a JSP page is about to be destroyed. This method is similar to the destroy() method of servlets. It can be overridden by a page author to perform any cleanup operation such as closing a database connection.

**6.What JSP lifecycle methods can I override?**

You cannot override the \_jspService() method within a JSP page. You can however, override the jspInit() and jspDestroy() methods within a JSP page. jspInit() can be useful for allocating resources like database connections, network connections, and so forth for the JSP page. It is good programming practice to free any allocated resources within jspDestroy().

**7.How can I override the jspInit() and jspDestroy() methods within a JSP page?**

The jspInit() and jspDestroy() methods are each executed just once during the lifecycle of a JSP page and are typically declared as JSP declarations:

01.<%!

02.public void jspInit() {

03.. . .

04.}

05.%>

06.<%!

07.public void jspDestroy() {

08.. . .

09.}

10.%>

**8.What are implicit objects in JSP?**

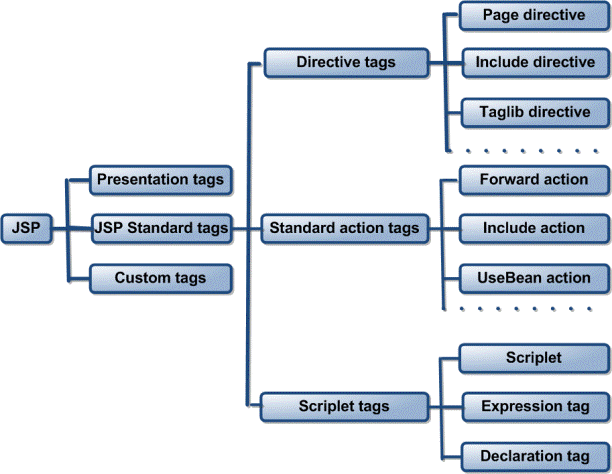
Implicit objects in JSP are the Java objects that the JSP Container makes available to developers in each page. These objects need not be declared or instantiated by the JSP author. They are automatically instantiated by the container and are accessed using standard variables; hence, they are called implicit objects.The implicit objects available in JSP are as follows:

* request
* response
* pageContext
* session
* application
* out
* config
* page
* exception

The implicit objects are parsed by the container and inserted into the generated servlet code. They are available only within the jspService method and not in any declaration.

**9.What are the different types of JSP tags?**

The different types of JSP tags are as follows:



**10.What are JSP directives?**

* JSP directives are messages for the JSP engine. i.e., JSP directives serve as a message from a JSP page to the JSP container and control the processing of the entire page.
* They are used to set global values such as a class declaration, method implementation, output content type, etc.
* They do not produce any output to the client.
* Directives are always enclosed within <%@ ….. %> tag.
* Ex: page directive, include directive, etc.

**11.What is page directive?**

* A page directive is to inform the JSP engine about the headers or facilities that page should get from the environment.
* Typically, the page directive is found at the top of almost all of our JSP pages.
* There can be any number of page directives within a JSP page (although the attribute – value pair must be unique).
* The syntax of the include directive is:

1.<%@ page attribute="value">

* **Example:**

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

**12.What are the attributes of page directive?**

There are thirteen attributes defined for a page directive of which the important attributes are as follows:

* **import:**It specifies the packages that are to be imported.
* **session:**It specifies whether a session data is available to the JSP page.
* **contentType:**It allows a user to set the content-type for a page.
* **isELIgnored:**It specifies whether the EL expressions are ignored when a JSP is translated to a servlet.

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

There are thirteen attributes defined for a page directive of which the important attributes are as follows:

* The include directive is used to statically insert the contents of a resource into the current JSP.
* This enables a user to reuse the code without duplicating it, and includes the contents of the specified file at the translation time.
* The syntax of the include directive is as follows:

1.<%@ include file = "FileName" %>

* This directive has only one attribute called file that specifies the name of the file to be included.

**14. What are the JSP standard actions?**

* The JSP standard actions affect the overall runtime behavior of a JSP page and also the response sent back to the client.
* They can be used to include a file at the request time, to find or instantiate a JavaBean, to forward a request to a new page, to generate a browser-specific code, etc.
* **Ex:** include, forward, useBean,etc. object.

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

The standard actions available in JSP are as follows:

* <jsp:include>: It includes a response from a servlet or a JSP page into the current page. It differs from an include directive in that it includes a resource at request processing time, whereas the include directive includes a resource at translation time.
* **<jsp:forward>**: It forwards a response from a servlet or a JSP page to another page.
* **<jsp:useBean>**: It makes a JavaBean available to a page and instantiates the bean.
* **<jsp:setProperty>**: It sets the properties for a JavaBean.
* **<jsp:getProperty>**: It gets the value of a property from a JavaBean component and adds it to the response.
* **<jsp:param>**: It is used in conjunction with <jsp:forward>, <jsp:, or plugin>; to add a parameter to a request. These parameters are provided using the name-value pairs.
* **<jsp:plugin>**: It is used to include a Java applet or a JavaBean in the current JSP page.

**16.What is the <jsp:useBean> standard action?**

The <jsp:useBean> standard action is used to locate an existing JavaBean or to create a JavaBean if it does not exist. It has attributes to identify the object instance, to specify the lifetime of the bean, and to specify the fully qualified classpath and type.

**17.What are the scopes available in <jsp:useBean>?**

The scopes available in <jsp:useBean> are as follows:

* **page scope**: It specifies that the object will be available for the entire JSP page but not outside the page.
* **request scope:** It specifies that the object will be associated with a particular request and exist as long as the request exists.
* **application scope:** It specifies that the object will be available throughout the entire Web application but not outside the application.
* **session scope:** It specifies that the object will be available throughout the session with a particular client.

**18.What is the <jsp:forward> standard 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 is stopped and control is transferred to the forwarded page.
* The syntax of the <jsp:forward> standard action is:

1.<jsp:forward page="/targetPage"></jsp:forward>

Here, targetPage can be a JSP page, an HTML page, or a servlet within the same context.

* If anything is written to the output stream that is not buffered before **<jsp:forward>**, an IllegalStateException will be thrown.

**Note**: Whenever we intend to use <jsp:forward> or <jsp:include> in a page, buffering should be enabled. By default buffer is enabled.

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

The **<jsp:include>** standard action enables the current JSP page to include a static or a dynamic resource at runtime. In contrast to the include directive, the include action is used for resources that change frequently. The resource to be included must be in the same context.The syntax of the **<jsp:include>** standard action is as follows:

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

Here, targetPage is the page to be included in the current JSP.

**20.What is the difference between include directive and include action?**

|  |  |
| --- | --- |
| **Include directive** | **Include action** |
| The include directive, includes the content of the specified file during the translation phase–when the page is converted to a servlet. | The include action, includes the response generated by executing the specified page (a JSP page or a servlet) during the request processing phase–when the page is requested by a user. |
| The include directive is used to statically insert the contents of a resource into the current JSP. | The include standard action enables the current JSP page to include a static or a dynamic resource at runtime. |
| Use the include directive if the file changes rarely. It’s the fastest mechanism. | Use the include action only for content that changes often, and if which page to include cannot be decided until the main page is requested. |

**21. Differentiate between pageContext.include and jsp:include ?**

The <jsp:include> standard action and the pageContext.include() method are both used to include resources at runtime. However, thepageContext.include() method always flushes the output of the current page before including the other components, whereas<jsp:include> flushes the output of the current page only if the value of flush is explicitly set to true as follows:

1.<jsp:include page="/index.jsp" flush="true"> </jsp:include>

**22. What is the jsp:setProperty action?**

You use jsp:setProperty to give values to properties of beans that have been referenced earlier. You can do this in two contexts. First, you can use jsp:setProperty after, but outside of, a jsp:useBean element, as below:

1.<jsp:usebean id="myName" ...="">

2.<jsp:setproperty name="myName" property="myProperty" ...=""></jsp:setproperty>

3.</jsp:usebean>

In this case, the jsp:setProperty is executed regardless of whether a new bean was instantiated or an existing bean was found.

A second context in which jsp:setProperty can appear is inside the body of a jsp:useBean element, as below:

1.<jsp:usebean id="myName" ...="">

2....

3.<jsp:setproperty name="myName" property="someProperty" ...=""></jsp:setproperty>

4.

5.</jsp:usebean>

Here, the jsp:setProperty is executed only if a new object was instantiated, not if an existing one was found.

**23. What is the jsp:getProperty action?**

The <jsp:getProperty> action is used to access the properties of a bean that was set using the <jsp:getProperty> action. The container converts the property to a String as follows:

* If it is an object, it uses the toString() method to convert it to a String.
* If it is a primitive, it converts it directly to a String using the valueOf() method of the corresponding Wrapper class.
* The syntax of the <jsp:getProperty> method is:

1.<jsp:getproperty name="Name" property="Property"> </jsp:getproperty>

Here, name is the id of the bean from which the property was set. The property attribute is the property to get. A user must create or locate a bean using the <jsp:useBean> action before using the <jsp:getProperty> action.

**24. What is the <jsp:param> standard action?**

The <jsp:param> standard action is used with <jsp:include> or <jsp:forward> to pass parameter names and values to the target resource. The syntax of the <jsp:param> standard action is as follows:

1.<jsp:param name="paramName" value="paramValue"></jsp:param>

**25. What is the jsp:plugin action ?**

This action lets you insert the browser-specific OBJECT or EMBED element needed to specify that the browser run an applet using the Java plugin.

**26. What are scripting elements?**

JSP scripting elements let you insert Java code into the servlet that will be generated from the current JSP page. There are three forms:

* Expressions of the form <%= expression %> that are evaluated and inserted into the output,
* Scriptlets of the form <% code %> that are inserted into the servlet's service method,
* Declarations of the form <%! code %> that are inserted into the body of the servlet class, outside of any existing methods.

**27. What is a scriptlet?**

A scriptlet contains Java code that is executed every time a JSP is invoked. When a JSP is translated to a servlet, the scriptlet code goes into the service() method. Hence, methods and variables written in scriptlets are local to the service() method. A scriptlet is written between the <% and %> tags and is executed by the container at request processing time.

**28. What are JSP declarations?**

As the name implies, JSP declarations are used to declare class variables and methods in a JSP page. They are initialized when the class is initialized. Anything defined in a declaration is available for the whole JSP page. A declaration block is enclosed between the <%! and %> tags. A declaration is not included in the service() method when a JSP is translated to a servlet.

**29.What is a JSP expression?**

A JSP expression is used to write an output without using the out.print statement. It can be said as a shorthand representation for scriptlets. An expression is written between the <%= and %> tags. It is not required to end the expression with a semicolon, as it implicitly adds a semicolon to all the expressions within the expression tags.

**30.How is scripting disabled?**

Scripting is disabled by setting the scripting-invalid element of the deployment descriptor to true. It is a subelement of jsp-property-group. Its valid values are true and false. The syntax for disabling scripting is as follows:

1.<jsp-property-group>

2.<url-pattern>\*.jsp</url-pattern>

3.<scripting-invalid>true</scripting-invalid>

4.</jsp-property-group>