JSP Interview Questions and Answers

### Interview Questions Answers on JSP

**Question 1: Explain include Directive and include Action of JSP**

Ans:  This is a very *popular interview question on JSP*, which has been asked from a long time and still asked in the various interview. This question is good to test some fundamental concept like translation of JSP and difference between translation time and run time kind of concept.

Syntax for include Directive is **<%@ include file="filename" %>** which means we are including some file to our JSP Page when we use include directive contents of included file will be added to calling JSP page at translation time means when the calling JSP is converted to servlet ,all the contents are added to that page .one important thing is that any JSP page is compiled if we make any changes to that particular page but if we have changed the included file or JSP page the main calling JSP page will not execute again so the output will not be according to our expectation, this one is the main disadvantage of using the include directive that why it is mostly use to add static  resources, like Header and footer .

Syntax for include action is **<jsp:include page=”relativeURL” />** it’s a runtime procedure means the result of the JSP page which is mentioned in relative URL is appended  to calling JSP at runtime on their response object at the location where we have used this tag

So any changes made to included page is being effected every time, this is the main advantage of this action but only relative URL we can use here ,because request and response object is passed between calling JSP and included JSP.

**Question** **2: Difference Between** **include Directive and include Action of JSP**

This JSP interview question is a continuation of the earlier question I just made it a separate one to write an answer in clear tabular format.

|  |  |
| --- | --- |
| **Include Directive** | **Include Action** |
| include directive is processed at the translation time | Include action is processed at the run time. |
| include directive can use relative or absolute path | Include action always use relative path |
| Include directive can only include contents of resource it will not process the dynamic resource | Include action process the dynamic resource and result will be added to calling JSP |
| We cannot pass any other parameter | Here we can pass other parameter also using JSP:param |
| We cannot  pass any request or response object to calling Jsp to included file or JSP or vice versa | In this case it’s possible. |

**Question 3: Is it possible for one JSP to extend another java class if yes how?**

**Ans:** Yes it is possible we can extends another JSP using this <%@ include page extends="classname" %> it’s a perfectly correct because when JSP is converted to servlet its implements **javax.servlet.jsp.HttpJspPage** interface, so for Jsp page it’s possible to extend another java class . This question can be tricky if you don’t know some basic fact , though it's not advisable to write java code in Jsp instead it's better to use expression language and tag library.

**Question 4: What is < jsp:usebean >tag why it is used.**

Ans: This was very popular JSP interview question during early 2000, it has lost some of its shine but still asked now and then on interviews.

**JSP Syntax**

<jsp:useBean   
        id="beanInstName"   
        scope="page | request | session | application"   
        m  
            class="package.class"    type="package.class"   
  
           </jsp:useBean>

This tag is used to create an instance of java bean, first of all, it tries to find out the bean if bean instance already exists assign stores a reference to it in the variable. If we specified the type, gives the Bean that type. Otherwise instantiates it from the class we specify, storing a reference to it in the new variable.so **jsp:usebean** is a simple way to create a java bean.

**Example:**

<jsp:useBean id="stock" scope="request" class="market.Stock" />

<jsp:setProperty name="bid" property="price" value="0.0" />

a <jsp:useBean> element contains a <jsp:setProperty> element that sets property values in the Bean, we have <jsp:getProperty>element also to get the value from the bean.

**Explanation of Attribute**

 id="beanInstanceName"

A variable that identifies the Bean in the scope we specify. If the Bean has already been created by another <jsp:useBean> element, the value of id must match the value of id used in the original <jsp:useBean> element.

scope="**page** | request | session | application"

The scope in which the Bean exists and the variable named in id is available. The default value is page. The meanings of the different scopes are shown below:

* page – we can use the Bean within the JSP page with the <jsp:useBean> element
* request – we can use the Bean from any JSP page processing the same request, until a JSP page sends a response to the client or forwards the request to another file.
* Session – we can use the Bean from any JSP page in the same session as the JSP page that created the Bean. The Bean exists across the entire session, and any page that participates in the session can use it.
* Application – we can use the Bean from any JSP page in the same application as the JSP page that created the Bean. The Bean exists across an entire JSP application, and any page in the application can use the Bean.

Class="*package.class*"

Instantiates a Bean from a class, using the new keyword and the class constructor. The class must not be abstract and must have a public, no-argument constructor.

type="*package.class*"

If the Bean already exists in the scope, gives the Bean a data type other than the class from which it was instantiated. If you use type without class or beanName, no Bean is instantiated.

**Question 5: How can one Jsp Communicate with Java file.**

Ans: we have import tag <%@ page import="market.stock.\*” %> like this we can import all the java file to our Jsp and use them as a regular class another way is  servlet can send  the instance of the java class to our  Jsp and we can retrieve that object from the request obj and use it in our page.

**Question 6: what are the implicit Object?**

Ans: This is a fact based *interview question* what it checks is how much coding you do in JSP if you are doing it frequently you definitely know them. Implicit object is the object that is created by web container provides to a developer to access them in their program using JavaBeans and Servlets. These objects are called implicit objects because they are automatically instantiated. They are by default available in JSP page.

They are request, response, pageContext, session, and application, out, config, page, and exception.

**Question 7: In JSP page how can we handle runtime exception?**

Ans: This is another popular JSP interview question which has asked to check how candidate used to handle Error and Exception in JSP. We can use the error Page attribute of the page directive to have uncaught run-time exceptions automatically forwarded to an error processing page.

Example: <%@ page errorPage="error.jsp" %>

It will redirect the browser to the JSP page error.jsp if an uncaught exception is encountered during request processing. Within error.jsp, will have to indicate that it is an error-processing page, using the directive: <%@ page isErrorPage="true" %>

**Question 8: Why is \_jspService() method starting with a '\_' while other life cycle methods do not?**

Ans: main JSP life cycle method are jspInit() jspDestroy() and \_jspService(), by default whatever content we write in our Jsp page will go inside the \_jspService() method by the container if again will try to override this method JSP compiler will give error but we can override other two life cycle method as we have implementing this two in Jsp so making this difference container use \_ in jspService() method and shows that we can’t override this method.

**Question 9: How can you pass information form one Jsp to included Jsp:**

Ans: This *JSP interview question* is little tricky and fact based. Using < Jsp: param> tag we can pass parameter from main Jsp to included Jsp page

Example:

<jsp:include page="newbid.jsp" flush="true">  
<jsp:param name="price" value="123.7"/>  
<jsp:param name="quantity" value="4"/>

**Question 10: what is the need of tag library?**

Ans tag library is a collection of custom tags. Custom actions helps recurring tasks will be handled more easily they can be reused across more than one application and increase productivity. JSP tag libraries are used by Web application designers who can focus on presentation issues rather than being concerned with how to access databases and other enterprise services. Some of the popular tag libraries are Apache display tag library and String tag library. You can also check my post on display [tag library example on spring](http://javarevisited.blogspot.com/2011/09/displaytag-examples-tutorial-jsp-struts.html).

# JSP Interview Questions and Answers

**Q) What is JSP?**  
A) JSP stands for Java Server Pages, it is a server side technology which is used for creating dynamic web pages. It is the extension of servlets.

Based on the above response, the interviewer may ask below follow-up questions:

**What does dynamic web page means here?** 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.

**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).

**What is Servlet?** [Refer this](https://beginnersbook.com/2013/05/servlet-tutorial/).

**Q) What are JSP life cycle phases?**  
A) A JSP page goes through the below phases:  
1) **Compilation**: In this phase the JSP code gets converted into the equivalent servlet code.  
2) **Initialization**: The converted JSP code gets loaded into the memory. jspInit() method gets called in this phase.  
3) **Execution**: \_jspService() method gets called in this phase. In this step a response is generated for the user based on the request made by them.  
4) **Destroy**: jspDestroy() method gets called in this phase to unload the JSP from the memory. This is also known as cleanup step.

**Q) What all JSP lifecycle methods can you override in your JSP application?**  
You can only override jspInit() and jspDestroy(), you cannot override the \_jspService() method within a JSP page. By overriding jspInit() method you can initialize things like database connections, network connections etc. Whatever you initialize in jspInit() method can be freed up (released) in jspDestroy() method.

**Q) How many implicit objects you have in JSP, name them?**  
The objects that can directly be used on any JSP page without the need of being declared first are known as implicit objects. In JSP we have total 9 implicit objects, they are as follows:  
1) out  
2) request  
3) response  
4) session  
5) config  
6) exception  
7) page  
8) pageContext  
9) application

**Q) What is the difference between include directive and include action tag?**  
A) Refer this article: [include directive vs. include action tag](https://beginnersbook.com/2013/12/difference-between-include-directive-and-include-tag-in-jsp/).

**Q) What is the purpose of scriptlets in JSP? What’s the syntax of it?**  
A) A scriptlet is used for including java code in a JSP page.  
Syntax:   
<% Java Code %>

**Q) What is JSP declaration tag?**  
A) A JSP declaration tag is used for declaring variables and methods so that you can use them later on a JSP page based on the requirement.  
Syntax:   
<%! Declare variables /Methods %>

Refer [this article](https://beginnersbook.com/2013/11/jsp-declaration-tag/) for more detail on JSP declaration tag.

**Q) What all directives available in JSP?**  
A) There are three types of directives available in JSP  
1) **Page directive**: This directive is used for setting up the attributes of a JSP page. Refer article for complete page directive tutorial.  
Syntax:   
<%@ page attribute=”value” %>

2) **Include directive**: Include a JSP file into another JSP file during the translation phase of JSP life cycle. Refer [include directive tutorial](https://beginnersbook.com/2013/11/jsp-include-directive/) for more detail.  
Syntax:   
<%@ include attribute=”value” %>

3) **Taglib directive**: This directive is basically used for custom tags. Read more about [custom tags here](https://beginnersbook.com/2014/01/jsp-custom-tags-with-example-jsp-tutorial/).

**Q) How to handle an exception in JSP?**  
A) Refer this article: [Exception handling in JSP](https://beginnersbook.com/2013/11/jsp-exception-handling/)

**Q) What is expression language in JSP?**  
A) Refer this article: [EL in JSP](https://beginnersbook.com/2013/11/jsp-expression-language-el/)

**Q) How do you disable a session on a particular JSP page?**  
A) By using the session attribute of page directive, we can disable the session on a particular JSP page. This is how we can do it: <%@ page session=”false”>  
by default the session attribute is set to true.

**Q) How do you add a comment on a JSP page?**  
A) This is how we can do it: <%– JSP comment –%>

**Q) Is it possible to import a package in a JSP page?**  
A) Yes, we can import a package using import attribute of page directive.

**Q) Explain JSTL?**  
A) Refer [JSTL tutorial](https://beginnersbook.com/jsp-jstl-tutorial-jstl-functions-and-core-tags/).

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

JSP is a server side 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:

<%!

public void jspInit() {

. . .

}

%>

<%!

public void jspDestroy() {

. . .

}

%>

**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:

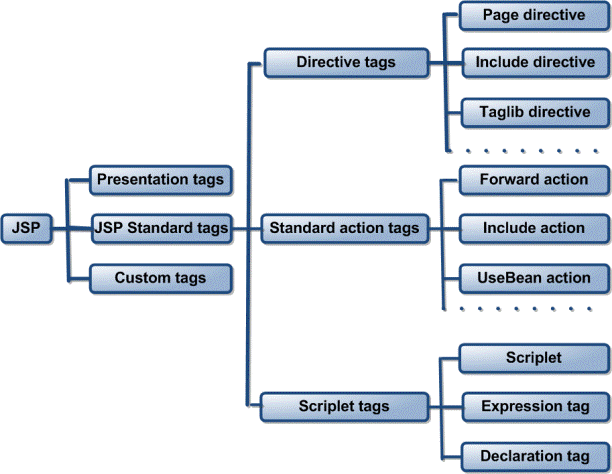
* request
* response
* pageContext
* session
* application
* out
* config
* page
* exception

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.

[Check more about implicit objects](http://developersbook.com/jsp/jsp-implicit-objects.php)

**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: <%@ page attribute="value">
* Example :< %@ 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:  
  <%@ 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 :    
  <jsp:forward page="/targetPage" />  
  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:   
<jsp:include page="targetPage" flush="true"/>   
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, the pageContext.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:

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

**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:

<jsp:useBean id="myName" ... />

...

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

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:

<jsp:useBean id="myName" ... >

...

<jsp:setProperty name="myName"

property="someProperty" ... />

</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: <jsp:getProperty name="Name" property="Property" />

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

**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:

1. **Expressions** of the form <%= expression %> that are evaluated and inserted into the output,
2. **Scriptlets** of the form <% code %> that are inserted into the servlet's service method,
3. **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 sub element of jsp-property-group. Its valid values are true and false. The syntax for disabling scripting is as follows:

<jsp-property-group>

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

<scripting-invalid>true</scripting-invalid>

</jsp-property-group>