Top 50 JSP Interview Questions And Answers For Java Developers

Interview Questions Java Interview Updated: October 20, 2016 Harsh S.
j2ee interview questions, list of jsp interview questions

Greetings readers, we've prepared the list of 50 most frequently asked JSP interview questions that can help you greatly in getting a lead during any Java/J2EE interview.

The idea we thought was to wear the shoes of the interviewer and come up with the straight JSP interview questions to ask from our readers.

JSP is an integral part of any Java EE web application, and we've covered the basics of JSP, JSP scripting, JSP custom tags, etc. in this post. If you are interested in evaluating your existing JSP/J2EE knowledge, then try your hands on this excellent JSP quiz we recently published.

However, if you wish to refer an online tutorial on the subject, we recommend peeking into Oracle's documentation. Let's now dig into the well of the most technical JSP interview questions and answers.

JSP Interview Questions And Answers For Java Developers.

The primary level of JSP Interview Questions.

1- What do you understand of the JSP technology and why is it used?

Java Server Pages (JSP) is a cross-platform technology residing on presentation layer and is the part of the SUN's J2EE platform. JSPs are like standard HTML web pages with Java code embedded inside. JSP files has the *.jsp file-extension. J2EE platform packages a JSP compiler to produce a Servlet class from the JSP page.

JSP technology enables the web developers to add Java code and some predefined actions into the static content. Java server pages get converted into Java servlet by the JSP compiler.

2- 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.

3- 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.

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.

4- 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:

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

5- 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.

6- Which of the JSP life cycle methods you can override?

You can't override the **_jspService()** Method within a JSP page. However, there are two approaches you can take to override the **jspInit()** and **jspDestroy()** methods within the JSP page.

jspInit() Can be quite useful for reserving resources like DB connections, N/W connections, and so forth for the JSP page. It is deemed as a good programming practice to free any allocated resources in the **jspDestroy()** method.

7- What do you have to say about the implicit objects in JSP?

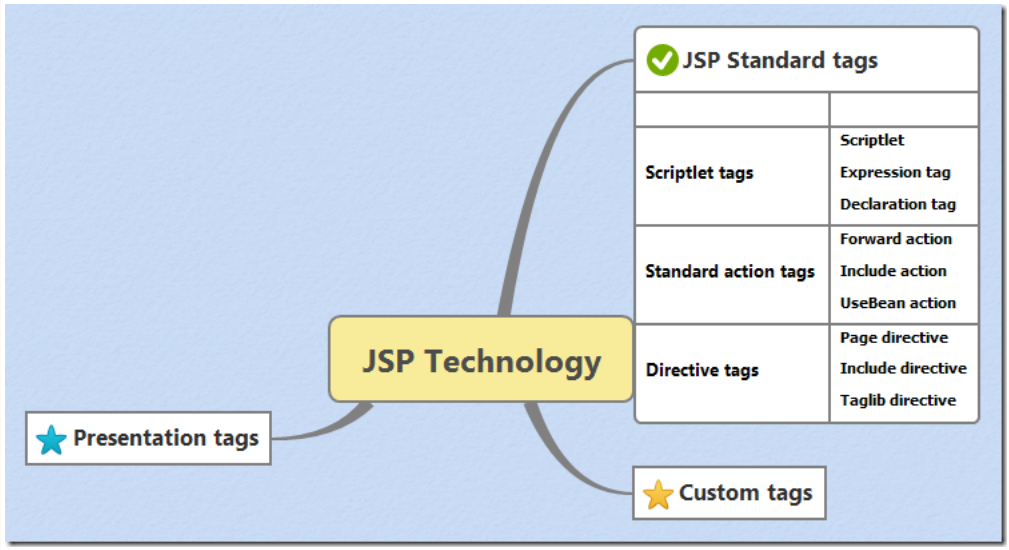
Implicit objects in JSP are pure Java objects. JSP Container makes these objects available to the developers on each JSP page. There is no need to declare or instantiate these objects explicitly by the JSP developer. JSP has a set of standard variables to access these objects. Hence, they are called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

- request,
- response,
- pageContext,

- session,
- application,
- out,
- config,
- page, and
- exception.

8- What are the different types of JSP tags?

The various types of JSP tags are as follows:



Top 50 JSP Interview Questions and Answers for Developer

9- What do you know about the JSP Standard Tag Library, please supplement with some quick example?

JSTL Library is more feature-rich than the JSP EL or Action elements because it allows us to loop through a collection or escape HTML tags to show them as text in response.

Example:

JSTL is one of the constituents of the J2EE API and available for the most servlet containers. But you can use it only after downloading the JSTL jars related to your servlet container. Locating them in the sample projects is easy. Once you've them, add these libraries to the project's **WEB-INF/lib** directory. You must note that every container has a jar specifically built for it, and you should consider this fact while adding the jar. For example in Tomcat, add the **<jstl.jar>** and **<standard.jar>** jar files to the project build path.

10- How many types of tags exist in the JSTL library?

Looking at the JSTL functions, there are five types of JSTL tags that we should know.

- **Core Tags –** Enable support for loops, conditional logic, exception handling, the ability of forwarding/redirecting response, etc.
- **Formatting/Localization Tags –** add features like the manipulation of date/number format and enabling i18n support with the help of locales and resource packages.
- **JSTL Functions Tags –** JSTL has a set of features that allow String handling features like concatenation, split operations, etc.
- **XML Tags –** XML tags enable reading/writing of XML documents. Some of the key features are XML parsing, transforming XML data and evaluation of XPath expressions.
- **SQL Tags –** JSTL brings some unique tags that cater the integration of JSP pages with databases like MySql, Oracle, etc.

Intermediate level of JSP Interview Questions.

11- What is a JSP Custom Tag and what are the components does it have?

It's a user-defined JSP element which can do operations that the standard tags like Action Tags or JSTL tags are not able to perform.

You can create JSP Custom Tags with the below items:

- JSP Custom Tag Handler,
- Creating Tag Library Descriptor (TLD) File, and
- Deployment Descriptor Configuration (DDC) for TLD.

You can use the custom tag library in JSP page using the <taglib> directive.

12- When should you use the JSP Custom Tag?

Let's consider the following illustration to understand the use of custom tags.

Example: Display a number value formatted using commas and spaces. This feature is extremely useful for long numbers. You can use the below custom tag for representing this complex number:

```
<customtag:formatNumber number="99999.99" format="#,###.##"/>
```

Depending on the number format, the JSP page should display the value as 99,999.99.

You may need to write a custom JSP tag to get the desired behavior if the JSTL doesn't have a built-in feature to do so.

13- Why is it not mandatory to configure the JSP standard tags in the web.xml?

There is no need to add the JSP standard tags to the web.xml file because the META-INF directory corresponding to the JSTL jar files contains the TLD files.

While loading the web application the container detects the TLD files present in the META-INF directory of the JAR file. It then automatically configures them and makes available for the JSP pages. You just have to add them to the JSP page using <taglib> directive.

14- How to deal with exceptions caused by a JSP service method?

The best approach for managing the exception on a JSP page is by defining the error page using the page directive.

You can quickly add a JSP error page by following the below steps:

1- Set the page directive attribute **<isErrorPage>** value to true.

2- Access the exception's implicit object in the JSP.

3- Use it to send the formatted error message to the client.

Refer the example given below.

```
<error-page>
<error-code>404</error-code>
<location>/error.jsp</location>
</error-page>

<error-page>
```

```
<error-page>
<exception-type>java.lang.Throwable</exception-type>
<location>/error.jsp</location>
</error-page>
```

15- How to deal with exceptions using the JSTL library?

JSTL has core tags like the **<c:catch>** and **<c:if>** to handle exception inside the JSP service method. The <c:catch> tag grabs the exception and saves it to a exception variable which you can process later using the **<c:if>** condition tag.

Refer the code fragment given below.

```
<c:catch var =”ex”>
<% int result = 99/0;%>
</c:catch>

<c:if test = “${ex ne null}”>
<p>Following exception occured : ${ex} <br />
Error Message: ${ex.message}</p>
</c:if>
```

We’ve used the JSP EL in the **<c:if>** condition of the above example.

16- What are the different JSP scripting elements?

There are three types of scripting language elements:

- Declarations,
- Scriptlets, and
- Expressions.

17- 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.

18- What is a JSP declaration?

JSP Declarations help to declare the class variables and methods in a JSP page. They get initialized along with the initialization of the class. Everything in a <declaration> is accessible to the whole JSP page. You can encircle a declaration block within the **<%! And %>** tags.

19- What is a JSP expression?

A JSP expression helps to write an output without using the < **out**.**print** statement> . You can see it as a shorthand description for the scriptlets. And as usual, you delimit an expression using the **<%= and %>** tags.

Ending the expression with a semicolon is not mandatory, as it gets automatically added to the <expressions> within the expression tags.

20- 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.

21- 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.
- The include directive syntax is: <%@ page attribute="value">

e.g.: <%@ include file="header.jsp" %>

22- 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:

- **<import>**: Signifies the packages that get queued for import.
- **<session>**: Specifies the session data available to the JSP page.
- **<contentType>:** Allows a user to update the content-type for a page.
- **<isELIgnored>**: Specifies if an EL expression gets ignored during the translation of the JSP to a servlet.

23- 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.

24- 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.

25- What are the standard actions available in JSP?

Please refer the below list:

- **<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.

26- What is the purpose of the <jsp: useBean> action?

The <jsp: useBean> standard action allows to identify an existing JavaBean or to create the one if it doesn't exist.

It contains attributes to classify the object instance, to specify the lifetime of the bean, as well as the fully-qualified classpath and type.

27- 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.

28- 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.
- 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.

29- 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"/>
```

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

30- Explain the main differences between an include action and the include directive?

- The *include* directive adds the target content during the translation phase while the page is getting compiled to a servlet.
  On the contrary, the *include* action adds the response generated after executing the given page (a JSP/a servlet) and during the request processing phase when the user queries the page from his browser.
- The include directive statically inserts the resource content into the current JSP whereas the include action enables the current JSP page to attach a static/dynamic resource at run-time.
- Include directive works the best if the file rarely changes whereas the include action is best suited to the content that changes quite often.

Advanced level of JSP Interview Questions.

31- What are the different scripting elements available in JSP?

JSP scripting elements let you embed the Java code into the servlet. Three forms of scripting elements are available in the JSP.

- **Expression** syntax is <%= Expression %>. They get evaluated and inserted into the output.
- **Scriptlet** syntax is <% Code %>. They get inserted into the servlet's service method.
- A **Declaration** follows the syntax <%! Code %>. They get attached to the body of the servlet class, outside of any existing methods.

32- What is the right way to disable scripting?

You can disable the scripting by changing the <scripting-invalid> element of the deployment descriptor (DD) to true. It is a child element of the <jsp-property-group> tag. You can only set it to a boolean value. Please check out the below sample to disable scripting.

```
<jsp-property-group>
<url-pattern>*.jsp</url-pattern>
<scripting-invalid>true</scripting-invalid>
</jsp-property-group>
```

33- What is the right way to declare a method in a JSP page?

You can declare a method in a JSP page as a declaration.

Later you can call this method from any other methods of the JSP expressions or scriptlets.

For your note, there is no direct access allowed to the JSP implicit objects (request, response, and session, etc.). But you can pass the JSP variables as parameters to the method which is declared.

34- Is it possible from a JSP page to process HTML form content?

Yes. You can fetch the data from the input elements of the form by using the request's implicit object. It lies either as a scriptlet or an expression. Next, there is no need to implement any HTTP methods like goGet() or doPost() in the JSP page.

35- How to prevent direct access to JSP pages from the client browser?

We can utilize the WEB-INF directory to place the JSP pages as a web application can't access it directly from the customer's browser. But in this case, we will have to configure it in deployment descriptor just like Servlets. Please refer the below example to view the sample configuration of the web.xml file.

```
<servlet>
<servlet-name>Test</servlet-name>
<jsp-file>/WEB-INF/test.jsp</jsp-file>
<init-param>
<param-name>test</param-name>
<param-value>Test Value</param-value>
</init-param>
</servlet>

<servlet-mapping>
<servlet-name>Test</servlet-name>
<url-pattern>/Test.do</url-pattern>
</servlet-mapping>
```

36- Is JSP technology extensible?

YES. JSP technology is indeed extensible as it allows to create custom actions or tags for a large no. of use cases. JSTL libraries enable this functionality in a JSP page.

37- Which of the JSP tags would you use to print the following text on a JSP page?
**
 Add a new line after every JSP interview questions**

Use the <c: out> escapeXml attribute to escape the HTML to show it as text in the browser. Check out the below code to achieve this.

```
<c:out value="<br> Add a new line after every JSP interview questions" escapeXml="true"></c:out>
```

38- How to prevent the page errors from printing in a JSP page?

Setup an "ErrorPage" attribute of the PAGE directory to the name of the error page in the JSP page template. And then use the following setting in the error page JSP.

```
isErrorpage="TRUE"
```

With the above attribute, you can stop the errors from getting displayed.

39- What is the best way to implement a thread-safe JSP page?

Apply the SingleThreadModel Interface in the JSP page.

You can get this done by adding the directive given in the below example.

```
<%@page isThreadSafe="false" %>
```

40- How to restrict the caching of the JSP/servlet output on the client side?

Use the relevant HTTP header attributes to block the caching of JSP page output by the browser.

41- List out the major difference between a cookie and a JSP session?

Sessions always preserved on the server-side while the cookies have their way on the client-end.

42- What is the correct way to delete cookies in a JSP page?

JSP gives you two ways to remove the cookies.

1- Set the age of the cookie to zero by calling the setMaxAge() method.

2- Set a timer in the header using the response. setHeader(expires { Specify the time } attribute). It will get the cookies removed after the defined time.

```
<%
Cookie rmCookie = new Cookie("webCookie", null);
rmCookie.setMaxAge(0);
rmCookie.setPath("/");
response.addCookie(rmCookie);
%>
```

43- How to forbid a JSP page from automatically creating a session?

```
<%@ page session="false">
```

44- How to embed the Java code in the HTML pages.

Please refer the below code example which clearly answers the question.

```
< % @ page language = " java " %>
```

```
< % @ page language = " java " %>
<HTML>
<HEAD><TITLE>JSP Interview Questions Page</TITLE></HEAD>
<BODY>
<%
PrintWriter print = request . get Writer ( ) ;
print . println ( " Welcome to the List of JSP Interview Questions!!!" );
%>
</BODY>
</HTML>
```

45- What is the way to pass data form one jsp to a jsp included in the page?

You can use the <Jsp: param> tag to pass parameter from main jsp to the included jsp page.

```
<jsp:include page="employee.jsp" flush="true">
<jsp:param name="empname" value="TechBeamers"/>
<jsp:param name="empsalary" value="99999″/>
```

46- What do you meant by the output comment in the JSP context?

An output comment is the part of the generated HTML of the JSP page. On the client-side, you can see the comment from the page source in the Web browser.

JSP output comment syntax.

```
<!– comment [ <%= expression %> ] –>
```

e.g.

```
<!– Output comment is a part of the HTML shown to client. –>
```

Following text would appear in the page source in the web browser.

```
<!– Output comment is a part of the HTML shown to client. — January 24, 2004 –>
```

47- What do you meant by the Hidden Comment in the JSP context?

Hidden comment is the one that doesn't appear in the generated HTML of the JSP page. The JSP engine ignores it completely and discards any code if lying inside the hidden comment.

It is useful in the cases where you don't want to show or "comment out" the sections in your JSP pages.

Using the closing –%> symbol in the hidden comment isn't permissible. Though you can use it by escaping it after typing –%\>.

JSP Hidden comment syntax. <%– comment –%>

e.g.

```
<%@ page language="java" %>
<html>
<head><title>Check Hidden-Comment Example</title></head>
<body>
<%– This text won't be visible in the HTML of this JSP page –%>
</body>
</html>
```

48- What is the need of using the HttpServlet Init method for memory intensive operations that are executed once?

As you would know that the loading of a servlet instance results in the invocation of the servlet init() method. It is by design the ideal location to run any expensive operations that require one-time execution.

The another reason is that the init() method is thread-safe and can safely cache all the instance variables

of the servlet, which get read-only in the service method of the servlet.

49- Is it not advisable to create HttpSessions in a JSP, if yes then why?

JSP files generate the HttpSessions by default. This behavior is perfectly alright as per the J2EE design guidelines. But, if you don't use the HttpSession in the JSP files then you can reduce the performance overhead by employing the JSP page directive as given below:

```
<%@ page session="false"%>
```

50- What would you do for printing the stack trace in a JSP page?

Stack trace helps to analyze a problem better while debugging the JSP code. It can guide a web developer to identify the method which produced the exception and isolate the source of that method.

However, there is no easy way to log the stacktrace in a JSP page. You need to use the PrintWriter class to achieve this. Please follow the below code snippet that illustrates how to print a stack trace from a JSP error page:

```
<%@ page isErrorPage="true" %>
<%
out.println("      ");
PrintWriter pWriter = response.getWriter();
exception.printStackTrace(pWriter);
out.println(" ");
%>
```

Final Review Of JSP Interview Questions.

Since JSP technology is critical for web development perspective, the interviewer looks for candidates who have a decent level of knowledge of this subject. We've observed from our experience that you won't struck in an interview just after reading the lengthy tutorials or having worked on live projects. There remain many areas that you may not have touched, and the interviewer could ask questions on that.

That's why we built this ladder with top 50 JSP interview questions. So that you can use it to take a step farther in your career. If you want to praise our efforts, then please do spread the word by sharing it on social media.

All the Best,

TechBeamers.

RELATED

Latest Selenium TestNG Interview Questions And Answers – 2017

July 16, 2016

15 C# Questions On For, While And If Else Statements

November 21, 2016

ABOUT THE AUTHOR

HARSH S. EMAIL AUTHOR

Seasoned IT specialist, thinker, creator and coffee lover. Years of experience in programming languages like C, C++, C#, Objective-C, PHP, HTML/JavaScript, Angular JS, Java/J2EE, Python, Selenium. Excelled in using technologies like CI/CD with Jenkins, AWS, VMWare, EXSI, Virtual Box and a no. of Unix/Windows/Android platforms.