**JSP**

**Q1. Differentiate between Include Directive and Include Action.**

Ans: The Difference between both can be explained as follows:

|  |  |
| --- | --- |
| Include Directive | Include Action |
| Includes content at page translation time | Includes content at page request time |
| Preferred in Static Pages | Preferred in Dynamic Pages |
| Includes Original content of the page | Does not include Original content of the page |
| Cannot invoke include() method | Can invoke include() method |
| Page size can be increased in the run-time | Page size is fixed |
|  |  |

**Q2. What is JSP?**

Ans: JSP is an abbreviation for Java Servlet Page. It is a Server-Side Programming Language used to create dynamic web-pages in the form of HTML. The JSP page is implicitly converted into a servlet and it enables some additional features such as Expression Language, Custom Tags, and many more.

Q3. How can you include the results of another page?

Ans: The results of another page can be included by using the following methods:

* Include Directive
* Include Action

**Q3. What are the Literals used in JSP?**

Ans: The Literals used in JSP are as follows:

1. Null
2. Boolean
3. String
4. Integer
5. Float

**Q4. List down the major differences between the JSP Custom Tags and Java Beans.**

**Ans:** The Major Differences between **JSP Custom Tags** and **Java Beans** are as follows:

|  |  |
| --- | --- |
| Custom Tags | Java Beans |
| Custom Tags can manipulate JSP content | Java Beans cannot manipulate JSP content |
| Executing complex operations is simple | Executing complex operations is difficult |
| Custom Tags are hard to set up | Java Beans are simple to set up |
| Custom Tags are available only in JSP 1.1 | Java Beans are used in all JSP 1.x versions |

**Q5. Can a Constructor be used in place of init() method to initialize a servlet?**

Ans: Yes, We can use a constructor in place of init() method. But it is not preferred because init() was developed because earlier Java versions could not invoke constructors with arguments dynamically. So they could not assign a servletConfig. However, servlet containers still call an only no-arg constructor. So there is no access to servletContext or servletConfig.

**Q6. Explain Client-Side and Server-Side Validation.**

Ans: The Client-Side validation is done using JavaScript. The validation takes place within the browser. Javascript is used to submit the data in the form when the validation is successful. Validation errors do not require any extra network trip because the form cannot be submitted if there are any errors.

Similar kind of data validation is carried out in the Server-Side after submission of the form. In if the validation fails, then, the extra network trip is required to resend the form to the client to refill the form with the correct data.

**Q7. Explain Translation Phase.**

Ans: During the Translation Phase, the JSP engine translates and compiles a JSP file into a servlet. This servlet moves to the execution phase where all the requests and responses are handled. They are compiled for the first time. They are not accessed unless they are manually compiled. The manual/explicit compilation is useful for long and convoluted programs.

**Q8. Mention the advantages of JSP over Pure Servlets?**

Ans: Some of the Major Advantages of JSP over Pure Servlets are as discussed below:

1. It is more convenient to write and modify normal HTML than to have plenty of println statements that generate the HTML.
2. Embedding of Java code in HTML pages.
3. Platform independence.
4. Creation of database-driven Web applications.
5. Server-side programming capabilities.

**Q11. How does JSP work?**

The JSP container has a special servlet called the page compiler. All HTTP requests with URLs that match the .jsp file extension are forwarded to this page compiler by the configuration of the servlet container. The servlet container is turned into a JSP container with this page compiler. When a .jsp page is first called, the page compiler parses and compiles the .jsp page into a servlet class. The JSP servlet class is loaded into memory on the successful compilation. For the subsequent calls, the servlet class for that .jsp page is already in memory. Hence, the page compiler servlet will always compare the timestamp of the JSP servlet with the JSP page. If the .jsp page is more current, recompilation is necessary. With this process, once deployed, JSP pages only go through the time-consuming compilation process once.

**Q12. What is the use of JSP?**

Earlier, Common Gateway Interface (CGI) was the only tool for developing dynamic web content and was not very efficient. The web server has to create a new operating system process, load an interpreter and a script, execute the script, and then tear it all down again, for every request that comes in. This is taxing for the server and doesn’t scale well when the number of traffic increases.

Alternatives such as ISAPI from Microsoft, and Java Servlets from Sun Microsystems, offer better performance and scalability. However, they generate web pages by embedding HTML directly in programming language code. JavaServer Pages (JSP) changes all of that.

**Q13. What are some of the advantages of using JSP?**

1) Better performance and quality as JSP is a specification and not a product.

2) JSP pages can be used in combination with servlets.

3) JSP is an integral part of J2EE, a complete platform for Enterprise-class applications.

4) JSP supports both scripting and element-based dynamic content.

**Q14) What are the JSP implicit objects?**

JSP provides nine implicit objects by default. They are as follows:

|  |  |  |
| --- | --- | --- |
| Object | Type | Use in JSP |
| Out | JspWriter | Out is used for writing any data to the buffer, JspWriter object is used in JSP page |
| Request | HttpServletRequest | Request is created by each JSP request by web container, request information contains parameters, header information, server name, remote address, server port, content type and character encoding |
| Response | HttpServletResponse | Response is used to add or update response as redirect response to another resource or send an error in case redirect does not work. |
| Config | ServletConfig | Config object like ServletConfig used in order to get initialization parameters from web.xml |
| Application | ServletContext | ServletContext is an application instance works as config object as well get, set or remove attribute from application scope. |
| Session | HttpSession | Session object is used for get, set or remove attribute or get session information. |
| PageContext | PageContext | PageContext object is used to get, set or remove attribute from page, request, session and application objects. |
| Page | Object | Page are assigned to reference of auto generated servlet class. |
| Exception | Throwable | Exception are objects from java.lang.Throwable class, used to print exception in error page. |

**Q16. What are features of JSP that make it an essential web-based technology?**

Answer: The features are listed below:

1. Create interactive websites.
2. Makes it easy-to-read user input data and display server response.
3. Use of implicit objects into the web page directly.
4. Use of Java code into HTML pages through JSP.
5. Makes database connectivity simple.
6. Visitors can be tracked using Session, Application, or Cookies on the JSP page.
7. Easy to learn and code.

**Q17. What are the differences between forward and sendRedirect?**

**Answer:** **Differences are explained below:**

|  |  |
| --- | --- |
| Forward() | sendRedirect() |
| Redirection of pages are handled at server end and are not visible to client in forward() method. | Redirection of pages are handled at client end and is visible to client, in sendRedirect() method. |
| forward() is faster than sendRedirect(). | sendRedirect() is slightly slower than forward(), as it needs two browser requests. |
| Original URL remains unaffected in forward() | A browser understands that for sendRedirect(), it is making new request, hence original URL changes. |
| Original URL and requests gets repeated when resulting page in browser is reloaded. | Original request or resultant URL is not repeated when browser reloads resultant URL. |

**Q18. Can we extend another Java class in JSP?**

Answer: Yes, developer can extend another JSP using <%@ include page extends =”classname” %>.

javax.servlet.jsp.HttpJspPage interface implements when the JSP page is converted to the Servlet page. Hence, it is possible to extend another Java class for the JSP page. It is advisable to use expression language or tag library instead of java code in JSP page.

**Q19 What are Cookies, their types, and how each type differs from the other?**

Answer: Cookies are text files that are created in the client machine and store details of web searches or sites explored using the browser, date, and time of visit, along with IP address. The presence of cookies helps websites identify clients that return to their sites.

Cookies are of various types such as Authentication, Tracking, and Session cookies.

Session cookies are short-lived until the user explores a particular website, once he leaves the site, session cookies are deleted by browsers.

Tracking cookies as the name suggests keeps a record of how often a particular website has been visited.

Authentication cookies validate a genuineness of a valid user that revisits the website.

**Q20. What are the JSP lifecycle phases?**

If you will look into JSP page code, it looks like HTML and doesn’t look anything like java classes. Actually JSP container takes care of translating the JSP pages and create the servlet class that is used in web application. JSP lifecycle phases are:

1. Translation – JSP container checks the JSP page code and parse it to generate the servlet source code. For example in Tomcat you will find generated servlet class files at TOMCAT/work/Catalina/localhost/WEBAPP/org/apache/jsp directory. If the JSP page name is home.jsp, usually the generated servlet class name is home\_jsp and file name is home\_jsp.java
2. BCompilation – JSP container compiles the jsp class source code and produce class file in this phase.
3. Class Loading – Container loads the class into memory in this phase.
4. Instantiation – Container invokes the no-args constructor of generated class to load it into memory and instantiate it.
5. Initialization – Container invokes the init method of JSP class object and initializes the servlet config with init params configured in deployment descriptor. After this phase, JSP is ready to handle client requests. Usually from translation to initialization of JSP happens when first request for JSP comes but we can configure it to be loaded and initialized at the time of deployment like servlets using load-on-startup element.
6. Request Processing – This is the longest lifecycle of JSP page and JSP page processes the client requests. The processing is multi-threaded and similar to servlets and for every request a new thread is spawned and ServletRequest and ServletResponse object is created and JSP service method is invoked.
7. Destroy – This is the last phase of JSP lifecycle where JSP class is unloaded from memory. Usually it happens when application is undeployed or the server is shut down.

**Q22.Which JSP lifecycle methods can be overridden?**

We can override jspInit() and jspDestroy() methods using JSP declaration scripting element. We should override jspInit() methods to create common resources that we would like to use in JSP service method and override jspDestroy() method to release the common resources.