JSP

1. List down the JSP page life cycle events in order?
    a. JSP page life cycle events are:
        i. JSP page translation,
        ii. JSP page compilation,
        iii. load class,
        iv. create instance,
        v. call the **jspInit** method,
        vi. call the **_jspService** method, and
        vii. call the **jspDestroy** method.
2. Describe the JSP Translation Phase?
    a. Different stages of the transalation phase are;
        i. **Initial input JSP**
        ii. Syntax checking
        iii. Code Generation
        iv. **Generated Servlet Source Code** (o/p of previous stage and input to the next)
        v. Compilation
        vi. **Compiled Servlet class (** final o/p**)**
3. Give the number of output files generated during a JSP translation phase? Do we have access to both?
    a. There are two outputs within the translation phase.
        i. The first is an interim output: a Java source file for a servlet.
        ii. The second is the compiled class file from the servlet source.
    b. The class file is retained for future use, and most JSP containers give you an option whereby you can retain the servlet source for debugging purposes.
4. When will the translation phase for a JSP occur?
    a. The translation occurs only when necessary, at some point before a JSP page has to serve its first request and doesn't have to happen again—unless the JSP source code is updated and the page redeployed.

b. A JSP container has discretion regarding when the translation occurs. It can occur on demand: as late as the first time a user requests a JSP page. At the other extreme, JSP pages can be translated on installation into a JSP container. The later process is often referred to as JSP pre-compilation.

5. Is the servlet generated during JSP translation phase same as any other servlet?
    a. The servlet created isn't just any servlet—it has some special characteristics. In particular:
        i. The servlet (or one of its superclasses) must implement the javax.servlet.jsp.HttpJspPage interface, or . . .
        ii. For the tiny minority of non-HTTP, specialist JSP containers, the servlet (or one of its superclasses) must implement the javax.servlet.jsp.JspPage interface.
    b. Because JSP servlets have to obey some special rules, a vendor will typically have a specialized JSP base servlet—perhaps extending HttpServlet or GenericServlet, and extended by all generated JSP servlets. In Tomcat, this is called org.apache.jasper.runtime.HttpJspBase.
6. Should we override _jspService method??
    a. Yes. The JSP spec is flexible enough to accommodate any request /response protocol you wish to implement. For that only you need to override _jspService method.
7. If a page fails to translate, an HTTP request for the page should give rise to a ……. status code communicated back in the HTTP response.
    a. 500 (server error)
8. Describe the JSP Request /Execution Phase?
    a. The servlet class is **loaded**, and any static initialization occurs.
    b. An **instance** of the servlet comes into being through normal construction processes.
    c. **jspInit()** is called. (Actually the servlet's init(ServletConfig config) method is called by the servlet container which must call the jspInit() method.) jspInit() is called once, before any requests for an instance of the JSP page are serviced.

d. **_jspService()** method is called. (When requests are made, the container calls the generated servlet's service(request, response) method, which in turn is obliged to call the method _jspService(), passing on the request and response parameters.)
e. **jspDestroy()** method is called. (Ultimately, when a JSP container decide to discard your generated servlet instance, the servlet's destroy() method is called — which, because it's a JSP page servlet, must call the jspDestroy() method.) The jspDestroy() is called before the instance of the JSP servlet is nullified and garbage collected, giving you the opportunity to reclaim resources in a controlled fashion.

9. Can we override servlet lifecycle methods from the JSP code?
    a. No. We should not override servlet lifecycle methods from the JSP code.
10. I have a jsp page in the context root. I have also registered the same JSP page in web.xml. How many instances of that JSP will be created?
    a. Two.
11. What will be the number of instances created in below situations?
        i. I have a jsp inside context root. I have a <servlet> element configured in web.xml for that file, but no servlet mapping.
        ii. I have a jsp inside context root. I have a <servlet> element configured in web.xml for that file, and in the only one servlet mapping I have three url patterns.
        iii. I have a jsp inside WEB-INF. I have a <servlet> element configured in web.xml for that file, and in the only one servlet mapping I have three url patterns.
        iv. I have a jsp inside context root. I have a <servlet> element configured in web.xml for that file, and two servlet mapping with a url pattern inside (URL patterns are different).
        v. I have a jsp inside context root. I have a <servlet> element configured in web.xml for that file, and two servlet mappings with the same url pattern inside.
    a. Answers:
        i. 1

ii. 2 – One for direct url access and 1 for servlet mapping. It doesn't matter the number of url pattern, but number of servlet elements for that.
iii. 2 – One for url access (we can use request dispatcher from another servlet or jsp) and 1 for servlet mapping. It doesn't matter the number of url pattern, but number of servlet elements for that.
iv. 2
v. 2

12. How can you suppress direct access to a JSP? Users have to go only through a registered name and a servlet mapping.
    a. If you want to suppress direct access to a JSP locate the JSP page under WEB-INF and configure it in the web.xml file.
13. Can we override the servlet life cycle methods from a JSP?
    a. No.
14. Which of the JSP life cycle methods you can override and which all you cannot from our JSP page?
    a. We can override **jspInit**() and **jspDestroy**().
    b. However we should not override **_jspService**() from our JSP page. This method represents your page source in Java code form—it's up to the container's page generators to implementation this method for each individual JSP.
15. In Tomcat, the generated servlet Java source and compiled class during JSP translation will, by default, be kept in which directory?
    a. <Tomcat-Installation-Directory>/work/Catalina/localhost/<context-directory>/org/apache/jsp
16. The ……. attribute of <script> tag specifies the scripting language of the element's contents and overrides the default scripting language.
    a. Type
17. Describe the <scripting-invalid> element of <jsp-property-group>?
    a. <scripting-invalid> of <jsp-property-group> when set to true causes a translation time error if JSP scriptlets, expressions, or declarations are used. The default (if this element is omitted) is false.

18. Can we create constructor for a JSP file?
    a. The servlet class corresponding to the JSP page is created by the container and we don't have details to create the constructor.

JSP Elements

19. What is Template text?
    a. It's any HTML or XML (or indeed any type of content at all) that you care to include in your JSP page. Template text is sent unchanged to response output.
20. A JSP page divides into ……. and ……..
    a. template text and elements
21. List down the type of elements?
    a. Directive - Most often use these to communicate global information to your page.
    b. Scripting - To incorporate dynamic information or execute presentation logic.
    c. Action - These use XML-style tags for the inclusion of dynamic data.
22. List down the two forms of scripting in a JSP?
    a. EL
    b. Traditional
23. List down the two forms of action elements?
    a. Standard
    b. Custom
24. In JSP page source, if it's not template text, it must be……..
    a. an element
25. List down the traditional language-based scripting elements in a JSP?
    a. Expressions (<%= … %>)
    b. Scriptlets (<% … %>
    c. Declarations (<%! … %>
    d. Comments (<%-- … %>
26. What are expressions in JSP?
    a. Expressions use the result of evaluating a piece of Java code directly in the page output. An expression must begin with <%= and conclude with %>. Since the evaluated value is passed as the parameter to the print method of a JspWriter

object, we should not end the statement with a semicolon. We can use a function, but it should return something. **Example: <%**= new Date() **%>.**

27. Give the resulting Java statement when this JSP expression is generated into servlet code <%= new java.util.Date() %>?
    a. When this is generated into servlet code, the resulting Java statement will probably look like this:
        i. out.print(new java.util.Date());
28. What will happen if we provide a function that return void inside an expression?
    a. JSP page will fail the translation stage. The Java used will be employed as parameter code inside a method call, which will not accept a void.
29. What are scriptlets in JSP?
    a. Scriptlets allow you to include an extended series of Java statements inside the _jspService() method that are executed on every request to the page. This time, the Java statements are incorporated "as is," so you must terminate each with a semicolon. A scriptlet begins with a <% and ends with a %>.
    b. **Example: <%** System.out.println("Normal Java Code"); **%>**
30. Can you mix template text with scriptlets?
    a. Yes. We can mix both template text and scriptlets. All code within scriptlets will go directly to the _jspService() method and the template text (such as <tr><td><font color="red">) is incorporated as out.write() statements in the _jspService() method, and expressions (such as <%= planets[i] %>) as out.print() statements.
    b. The template text, expressions, and scriptlets are all translated and generated into _jspService() in order of their appearance in the JSP page source.
31. Can you declare a local variable in one scriptlet and use that in another scriptlet or expression?
    a. Yes, as long as both are in same page. This is because both of them will be added in order to the _jspService() method.
32. What are declarations?
    a. If you want to place code in the generated servlet outside of the _jspService() method such as declaring another method, you can use declarations. It begins with

a **<%!** Marker and end with **%>**. You can place in your declaration any code that can legally appear in a servlet source file: instance methods, static methods, static initialization code, static or instance data members, inner classes.

b. Example:

i. <%! public void jspInit() {

ii. System.out.println("TRIAL INITIALIZED");

iii. } %>

33. Which of the traditional scripting elements – Expressions, Scriptlets, Declaration – has access to JSP implicit variables?

a. Expressions and scriptlets as they will be added to _jspService() method. Expressions doesn't have access to implicit objects.

34. For the four scripting elements —expressions, scriptlets, declarations, and comments— which all will next inside each other?

a. None. For the four scripting elements — expressions, scriptlets, declarations, and comments— none will nest inside each other or inside themselves. You can't have an expression inside a scriptlet, or a scriptlet inside a declaration, or a scriptlet in a scriptlet— or any other combination.

35. What are comments?

a. Whatever you give inside the comment element are ignored by the compiler. Starts with <%-- and end with --%>

b. If you want comments sent within the web page output, you can use regular HTML comment syntax: Open the comment with <!-- and close with -->.

36. Can an HTML comment contain scriptlets?

a. An HTML comment is treated exactly like other HTML template text. So it's perfectly acceptable to include JSP scriptlets, expressions, expression language, or any other legal JavaServer Page syntax within the comment—and it will be processed at translation time. However you cannot have scriptlet or any other traditional scripting element inside a JSP comment.

37. When will an initialization block and static initialization block executed for a JSP?

a. For every instance of a JSP page, an initialization block and static initialization block will be executed.

Directives

38. Give an example for the page directive?
    a. **<%@ page** import="java.util.*" language="Java"**%>**
39. Where all can a page directive occur on a JSP page?
    a. You can include a page directive anywhere in your JSP page source: beginning, middle, or end.
40. Which all of the statements below are valid page directive syntaxes?
        i. <%@page import="java.util.*"%>
        ii. <%@ page import = "java.util.*"%>
        iii. <%@ page import='java.util.*' %>
    a. All of them are valid.
41. Can we have more than one page directive in a page?
    a. Yes.
42. List down the important attributes for the page directive?
    a. **import** - to create import statements in the generated servlet source produced by the JSP container's translation phase. We can specify comma-separated list of classnames or packages as value of the import statement. Import attribute can be specified more than once—either across separate page directives that contain import once or even (silly as it is) using the import attribute more than once in the same page directive.
        i. **Ex**: <%@ page import="java.util.StringTokenizer" %>
    b. **session** - used to determine whether an HttpSession object is available within your JSP page source through an implicit object session. The default value is true.
        i. **Ex:** <%@ page session="true" %>
    c. **contentType** – used to set the content type. Same as the java code response.setContentType("image/gif");
        i. **Ex**: <%@ page contentType="image/gif" %>
    d. **isELIgnored** - you can switch off EL evaluation and have the text be treated as template text. The JSP 2.0 sets a default of isELIgnored="true" for backward compatibility. However if the deployment descriptor web.xml is at the version level of 2.4, the default of isELIgnored is set to "false".

i. **Ex**: <%@ page isELIgnored="true" %>

e. **language -** Denotes the scripting language. “Java” is the only value supported.
   i. Ex: <%@ page language=“Java” %>

f. **buffer, autoFlush** - You can use these attributes to control whether or not you have a buffer (you can specify a size in kilobytes), and how this buffer is flushed.
   i. Ex: <%@ page buffer=“none” autoFlush=“true” %>

g. **isThreadSafe** –
   i. Is set with the **true** which enables for accessing multiple request and give multiple responses simultaneously by generating multiple threads for your JSP application and developer is responsible for implementing thread safety.
   ii. Is set to **false** means container is responsible for thread safety, it makes the servlet to implement the SingleThreadModel with this every client request will have their own instances of the servlet hence making the JSP thread safe.
   iii. **Ex**: <%@ page isThreadSafe=“true” %>

h. **errorPage, isErrorPage -** Use errorPage to set a URL pointing to another JSP within the same web application. Should an exception occur, your users will be forwarded to this other JSP. The page you forward to must have “isErrorPage” set to true.
   i. **Ex:** <%@ page errorPage=“errorPage.jsp” %>
   ii. <%@ page isErrorPage=“true” %>

43. The ……. page attribute determines the MIME type of the response sent back.
    a. contentType
44. You make use of which attribute of page directive to override the base servlet that your container provides when generating servlets?
    a. extends
    b. **Ex:** <%@ page extends=“com.osborne.webcert.MyBaseJSPServlet” %>
45. The attribute of page directive to publish information about your JSP page, accessible through the getServletInfo() method?
    a. info

46. List few packages that are already available in the generated servlet source produced by the JSP container's translation phase?
    a. java.lang
    b. javax.servlet
    c. javax.servlet.http
    d. javax.ervlet.jsp
47. Is there any advantage for disabling session object in a page using the page directive?
    a. If your JSP page genuinely doesn't need access to the session, there's a small performance gain to be made with this. We can eliminate the time spent on creating or obtaining an HttpSession object.
48. Are the below statements same?
        i. <%@ page session="true" %>
        ii. <%@ page session="TRUE" %>
    b. Yes. Unlike boolean literals, however, these values are case insensitive.
49. Why is it a good practice to use page directive <%@ page contentType="image/gif" %> rather than setting it from the code as <% response.setContentType ("image/gif"); %>?
    a. When we use the page directive the line of code below which is response.setContentType ("image/gif"); will be available in the generated servlet source produced by the JSP container's translation phase.
    b. When we use response.setContentType ("image/gif"); code, in the generated code we will have two lines as:
        i. response.setContentType("text/html");
        ii. // A few other lines …
        iii. response.setContentType("image/gif");

       The first occurrence comes about because even if you omit the page directive for attribute contentType, the JSP container is bound to set a default MIME-type of "text /html." Because no content has been committed, this doesn't matter—the setting of "image/gif " comes later, so it takes precedence. However it is better to avoid this using the page directive.
50. Give the syntax of include directive?
    a. <%@ include **file**="stubs/header.html" %>

b. Here file is the only mandatory parameter.

51. Will the container reperform translation if either the including or included file is updated?
    a. Most containers will reperform translation if either the including or included fi le is updated. However, this is not mandated by the JSP specifi cation, even though it states a preference for that behavior.
52. What is an include directive used for?
    a. The purpose of include is to merge the contents of one JSP source file into another (the one doing the including). This happens at the point where the including JSP page source goes through translation. A JSP source file and the pages it includes through the include directive are collectively referred to as a "translation unit."
53. What are the types of files you can include using an include directive?
    a. The file type can be anything whose contents make sense to the JSP translation process once included. It can be JSP or HTML or XML documents, or document fragments.
54. How can you use the include directive to include a file from another context?
    a. You cannot include a file from another context using the include directive.
55. What is the major difference between include directive and <jsp:include> standard action?
    a. The key difference between <jsp:include> and the include directive is that <jsp:include> executes afresh with every new request to the including JavaServer Page. The include directive happens at translation time.
56. The ……. directive makes custom actions available in the JSP page by referencing a tag library.
    a. taglib
57. Give an example usage of taglib directive?
    a. <%@ taglib prefix="mytags" uri="http://www.abc.com/taglibs/mytags" %>

Implicit Objects

58. What are implicit objects?
    a. They are local variables declared at the outset of the _jspService() method. Because they have standardized names, they are available for use within your expressions and scriptlets.
59. List down the implicit objects with their types?
    a. Implicit objects are:
        i. **request**
            1. javax.servlet.http.HttpServletRequest interface
        ii. **response**
            1. javax.servlet.http.HttpServletResponse interface
        iii. **application**
            1. javax.servlet.ServletContext interface
        iv. **config**
            1. javax.servlet.ServletConfig interface
        v. **session**
            1. javax.servlet.http.HttpSession interface
        vi. **out**
            1. javax.servlet.jsp.JspWriter abstract class
        vii. **pageContext**
            1. javax.servlet.jsp.PageContext abstract class
        viii. **page**
            1. java.lang.Object class
        ix. **exception**
            1. java.lang.Throwable class
60. How can you get the HTTP method used by the request for the JSP from the request object?
    a. request.getMethod()
61. Using an implicit object display the name of your web application (as defined in the <display> element of the deployment descriptor, web.xml)?
    a. application.getServletContextName()

62. How can you end a session using an implicit object?
    a. session.invalidate()
63. List two cases where you won't get a session implicit object?
    a. You can make the session implicit object unavailable using the following directive:
        i. <%@ page session="false" %>
    b. In the JSP containers that don't operate with the HTTP protocol.
64. An implicit object which is not of much use without casting it?
    a. page
65. The javax.servlet.jsp.PageContext abstract class inherits from the ……. class.
    a. java.servlet.jsp.JspContext
66. Which implicit object provides a mechanism for handling exceptions on a page, and forwarding to another page?
    a. pageContext
67. The ……. object knows about all the other implicit objects, and it has methods to get hold of them.
    a. pageContext
68. The page scope is associated with which implicit object?
    a. pageContext
69. List down the page context attribute methods?
    a. **Object getAttribute(String name, int scope)** – To get hold of the attributes. The allowed scopes are defined as public, final, static constants on the PageContext class:
        i. PageContext.**PAGE_SCOPE**
        ii. PageContext.**REQUEST_SCOPE**
        iii. PageContext.**SESSION_SCOPE**
        iv. PageContext.**APPLICATION_SCOPE**
    b. **void setAttribute(String name, Object value, int scope) –** To set an attribute. If you pass in a null Object for the value, this has the same effect as calling the equivalent removeAttribute() method.

c. **void removeAttribute (String name, int scope) -** Removes the named attribute in the given scope. Has no effect if no attribute of this name exists in the given scope.

d. **Object findAttribute(String name) -** You merely pass in an attribute name, and the method works through the scopes in order: page, request, session, and application. The fi rst attribute found is returned.

e. **get, set, remove Attribute()** without the scope parameter – specifically target page scope.

70. The ……. implicit variable represents a writer associated with the response for your JSP page.

    a. out

71. How is out implicit object different from a PrintWriter object we get from the response?

    a. The out is a javax.servlet.jsp.JspWriter. The main thing that JspWriter gives you is buffering: The content in your page (template text, expressions, whatever) isn't—by default—committed to the response straightaway. There are things you can't do once output has been committed, such as setting response headers. You can control the buffering through page directives. PrintWriter is unbuffered, so the response output is sent to the page directly.

72. Can we import packages like this:

    i. <%! import java.util.*; %>

    a. No. code inside a declaration will be placed outside all methods, but inside the class. However an import statement should be used outside the class after the package statement.

73. The ……. method of Throwable returns the detail message string of this throwable.

    a. getMessage()