Chapter - 02 - Web Applications

1. Is it necessary to have a directory called WEB-INF in your web application?
    a. Having a directory called WEB-INF is a strong recommendation, but not compulsory.
2. If we have /WEB-INF/classes and /WEB-INF/lib directories, classes inside which directory will be loaded first?
    a. Java classes in the /WEB-INF/classes directory should be loaded before classes resident in JAR files in the /WEB-INF/lib directory.
3. Which is the correct variant - WEB-INF, Web-Inf, WEB_INF, web-inf?
    a. WEB-INF. Other variants don't count.
4. ………. Is the home directory for the web.xml file?
    a. WEB-INF
5. List the standard directories inside WEN-INF folder?
    a. **/WEB-INF/classes**—for classes that exist as separate Java classes (not packaged within JAR files).
    b. **/WEB-INF/lib**—for JAR files. These can contain anything at all.
    c. **/ WEB-INF** itself is the home for web.xml.
    d. **/WEB-INF/tags** - for TLDs and tag files.
6. Can we define our own directories (or directory structure) under WEB-INF?
    a. Yes.
7. List down the elements within root element web-app, which are mandatory?
    a. None.
8. List down the order of Top-level elements inside the root element web-app?
    a. Top-level elements can now appear in any order from servlet version 2.4.
9. Does the order of elements matter inside the top level elements?
    a. Yes.
10. Is it necessary to have a deployment descriptor (web.xml) file?
    a. A web application is NOT required to contain a web.xml if it does NOT contain any Servlet, Filter, or Listener components. In other words an application containing only static files or JSP pages does not require a web.xml to be present.
11. Can I have a web application with servlets, but without web.xml file?

   a. We can have servlets. They will compile into class files also. However we won't be able to call it unless we declare them inside web.xml file.

12. Can I have a web application with servlets, but without web.xml file?
    a. Yes. Many web servers have the capability of "serving servlets by name."
13. What is a context root?
    a. The context root identifies home directory for the web application. It is part of the URL that uniquely identifies a particular web application running on a server. Within a server context root must be unique. Though it is often same as the home directory name, which is not compulsory.
14. What is the root element of a web application?
    a. It is <web-app> </web-app>
15. List down the top-level elements within the root element <web-app> of a web.xml file?
    a. They are:
        1. <description>
        2. <display-name>
        3. <icon>
        4. <distributable>
        5. <context-param>
        6. <filter>
        7. <filter-mapping>
        8. <listener>
        9. <servlet>
        10. <servlet-mapping>
        11. <session-config>
        12. <mime-mapping>
        13. <welcome-file-list>
        14. <error-page>
        15. <jsp-config>
        16. <security-constraint>
        17. <login-config>
        18. <security-role>

19. <env-entry>
20. **<ejb-ref>**
21. <ejb-local-ref>
22. <service-ref>
23. **<resource-ref>**
24. **<resource-env-ref>**
25. <message-destination-ref>
26. <locale-encoding-mapping-list>

16. List down the sub elements of the **<servlet> tag**? Also give the number of times each of them can occur.
    a. <description> (*0 or many*)
    b. <display-name> (*0 or many*)
    c. <Icon> (*0 or many*)
    d. <servlet-name> (***Always 1***)
    e. <servlet-class> or <jsp-file> (***Always 1***)
    f. <init-param> (*0 or many*)
    g. <load-on-startup> (*0 or 1*)
    h. <run-as> (*0 or 1*)
    i. <security-role-ref> (*0 or many*)
17. Who will validate the order within any top level element, say <servlet> element?
    a. The deployment descriptor schema validates that.
18. Describe the <servlet-name> sub element of the <servlet> element?
    a. This subelement defines a logical name for the servlet. The name must be unique within the web application. However it can be different from the actual java class name.
19. Give two reasons for naming a serlvet?
    a. Servlets are normally a protected resource, kept in the WEB-INF/classes directory. The servlet name is part of the mechanism by which controlled access to servlets is allowed.
    b. A logical, unique name for the servlet is less cumbersome than always referring to a servlet, say, by its fully qualified Java class name.

20. How can you access the servlet name within a servlet?
    a. We can use the getServletName() method, which is defined in the ServletConfig interface.
21. Briefly describe the HttpServlet class?
    a. Provides an abstract class to be subclassed to create an HTTP servlet suitable for a Web site. A subclass of HttpServlet must override at least one method, usually one of these:
        i. doGet, if the servlet supports HTTP GET requests
        ii. doPost, for HTTP POST requests
        iii. doPut, for HTTP PUT requests
        iv. doDelete, for HTTP DELETE requests
        v. init and destroy, to manage resources that are held for the life of the servlet
        vi. getServletInfo, which the servlet uses to provide information about itself
22. Give the parents (classes extended and interfaces implemented) of the HttpServlet class?
    a. **HttpServlet** extends **GenericServlet** and implements Serializable. Through extending **GenericServlet**, it also implements **Servlet** and **ServletConfig** interfaces indirectly.
23. Can we override the service method of the HttpServlet class? What does the service method do?
    a. Yes. However there's almost no reason to override the service method. service handles standard HTTP requests by dispatching them to the handler methods for each HTTP request type.
24. Which of the following methods we can use inside an HttpServlet is/are derived from ServletConfig interface?
        i. getServletContext()
        ii. getInitParameter(String name)
        iii. getServletName()
    a. All three.
25. Give the use of <servlet-class> sub element of the <servlet> element? Give an example usage.

   a. This subelement defines the fully qualified name of a Java servlet class.
   b. Example:
      i. <servlet>
      ii.   <servlet-name>First</servlet-name>
      iii.   <servlet-class>example.FirstSample</servlet-class>
      iv. </servlet>

26. Can we define a servlet against more than one name?
   a. Yes.
27. Is there any use of defining a servlet against more than one name?
   a. It ensures a separate running instance of the servlet.
   b. We can supply different set of initialization parameters to the servlet.
28. How do you access a JSP file inside the WEB-INF folder?
   a. We cannot directly access the JSPs. However we can use the deployment descriptor to make a url pattern converted to that JSP. First we use the <jsp-file> tag inside <servlet> element to provide the JSP file name and then map the url-pattern to it using the <servlet-mapping> element.
   b. Example:
      i. <**servlet**>
      ii. <**servlet-name**>ConcealedJSP</servlet-name>
      iii. <**jsp-file**>/WEB-INF/secure/concealed.jsp</jsp-file>
      iv. </servlet>
      v. <**servlet-mapping**>
      vi. <**servlet-name**>ConcealedJSP</servlet-name>
      vii. <**url-pattern**>/allIsRevealed</url-pattern>
      viii. </servlet-mapping>
29. How can you prevent the direct access to a JSP file? What will be the alternate access method?
   a. We can place the JSP inside WEB-INF folder.
   b. We can use the deployment descriptor to make a url pattern converted to that JSP. First we use the <jsp-file> tag inside <servlet> element to provide the JSP file name and then map the url-pattern to it using the <servlet-mapping> element.

30. Describe the <init-param> sub element of <servlet> element?
    a. The <init-param> provide a means to pass parameters to a servlet through the Deployment Descriptor. It can occur 0 to many times.
31. List down the sub elements of the <init-param> element?
    a. <description></description> (*0 or many*)
    b. <param-name></param-name> (***Always 1***)
    c. <param-value></param-value> (***Always 1***)
32. Give an element in addition to <servlet> element which can contain the <init-param> element?
    a. <filter>
33. List down the functions along with their parent class for retrieving the init param information from the <init-param> sub element of <servlet> element?
    a. Methods are:
        i. ServletConfig.**getInitParameterNames**() returns an enumeration of all the parameter names available to the servlet.
        ii. ServletConfig.**getInitParameter**(String paramName) return a parameter value.
    b. Both the methods are implemented in the GenericServlet abstract class.
34. What will happen if we call getInitParameterNames().nextElement() inside a servlet when no init param elements are configured?
    a. We will get java.util.NoSuchElementException.
35. What will happen if we call getInitParameterNames().nextElement() inside a servlet when an empty <param-name/> and <param-value/> elements are only present inside the <init-param> element?
    a. An empty string will be returned.
36. How can you get the name of your servlet within your servlet code?
    a. We can call this.getServletName() defined in the ServletConfig interface.
37. Describe the use of <servlet-mapping> element?
    a. <servlet-mapping> element is basically used for getting at a servlet or jsp as a url resource. It has two sub elements <servlet-name> and <url-pattern>. A servlet or jsp defined in a corresponding servlet tag with same <servlet-name> will be

executed when we use a file name according to <url-pattern>. We can have more <url-pattern> elements and specify more patterns for the same servlet.

38. Can servers set up implicit mappings outside web.xml?
    a. Servers may have some implicit mappings setup outside of web.xml. If we specify the same, it will override the existing one.
39. If there is one url patten matching against more than one servlet in web.xml, will it throw any error? If no error, which servlet it will select?
    a. When tried with tomcat 6, the second servlet entry was actually selected.
40. Describe the different servlet mapping strategies for <url-pattern>?
    a. The matching procedure has four simple rules.
        i. First, the container prefers an exact path match over a wildcard path match.
        ii. Second, the container prefers to match the longest pattern.
            1. If we have <url-pattern>/Catalog/*</url-pattern> and <url-pattern>/Catalog/search/*</url-pattern>, then /Catalog/search/ always matches the second mapping (/Catalog/search/*).
        iii. Third, the container prefers path matches over filetype matches.
        iv. Finally, the pattern <url-pattern>/</url-pattern> always matches any request that no other pattern matches
41. Are the URL patterns case sensitive?
    a. Yes
42. What all syntaxes are allowed for URL mapping using </url-pattern?
    a. In the Web application deployment descriptor, the following syntax is used to define mappings:
        i. A string beginning with a '/' character and ending with a '/*' suffix is used for path mapping.
        ii. A string beginning with a '*.' prefix is used as an extension mapping.
        iii. A string containing only the '/' character indicates the "default" servlet of the application. In this case the servlet path is the request URI minus the context path and the path info is null.

iv. All other strings are used for exact matches only and should start with a forward slash.

43. Describe the use of the <description>, <display-name> and <icon> tags of the <servlet> element? List down the api methods for retrieving that info?
    a. Optionally, you can enter descriptive text for your servlet for the benefit of administrators. The <display-name> is also meant for use in web container administrative user interfaces in order to provide a short descriptive name, less expansive than the description text. There is no API in the servlet packages to retrieve info related to both the tags. You can include as many descriptions or diplay names as you want mainly to accommodate multiple languages.
    b. <icon> tag entirely optional, and you can have many occurrences. Within the icon element you can embed a <small-icon> and a <large-icon> element (one, both, or neither). The element describes a path to an image file that might be used by your web container administrative GUI to display next to your servlet.
44. Describe the use <load-on-startup> element? What will happen if it is not present?
    a. By defining the <load-on-startup> sub element of <servlet>, the servlet is loaded at the point when the web container starts. A servlet with a lower number will be loaded before a servlet with a higher number. If the numbers are the same there are no guarantees on which servlet starts first. If the number is negative, it’s as if the <load-on-startup> element wasn’t there at all. If a <jsp-file> is specified rather than a <servlet-class>, a <load-onstartup> setting ensures that the JSP is pre-compiled (turned into a servlet), then loaded as any other servlet would be.
    b. A web container’s usual practice is to load a servlet at the point where it is fi rst accessed.
45. What is the use of <welcome-file-list> element?
    a. The <welcome-file-list> provides a list of one or more pages to return, when the user types a path that identifies a directory and that path is not there in servlet mapping in the deployment descriptor. The welcome file list must describe a partial URL with no leading or trailing slashes and should be defined inside the <wecome-file> sub element. The process of finding the resource when a request URL doesn’t immediately match a specific resource is as follows:

i. Trailing slash is appended to the url(if not already present).
ii. Each file in the welcome file list is appended to the url and searched for a specific match.
iii. If the resource is present relative to the directory, it is returned.

b. Example:
i. <welcome-file-list>
ii. <welcome-file>../Trial.jsp</welcome-file>
iii. <welcome-file>/STrial1.jsp</welcome-file>
iv. <welcome-file>default.htm</welcome-file>
v. </welcome-file-list>

46. Once the container receives a request, what all things are done before it goes for <welcome-file-list> element?
   a. When the container receives a request,
      i. It first looks for a servlet mapping in the deployment descriptor. If a match is there corresponding servlet is executed.
      ii. Then it looks for an exact match. So if we have an URL pattern as /Trial.jsp and if a request come the servlet mapping will get precedence even if there is a file called Trial.jsp.
      iii. If there is no servlet mapping and exact match then it will go for <welcome-file-list> element.
47. The deployment descriptor file is validated against a schema file or DTD?
   a. From J2ee 1.4 it is validated using a schema file rather than a DTD. Up to and including J2EE 1.3 and version 2.3 of the servlet standard, web.xml was validated against a DTD.
48. Describe the use of the <error-page> element in the deployment descriptor?
   a. The <error-page> element in the deployment descriptor gives you customized control over the web page displayed to the user in the event of requests going wrong. The <error-page> element associates custom error pages with HTTP status codes or exception types.
   b. Sub elements are:

i. **<error-code> or <exception-type>** - <error-code> tag specifies the error code and <exception-type> can specify the fully qualified exception types. We cannot have both **<error-code>** and **<exception-type>** together.

ii. In all cases, **<location>** element contains the resource that needs to be displayed. The resource specified in the location should start with a forward slash to denote a path described from the context root.

c. Example 1:

i. <web-abb>

ii. <error-page>

iii. **<error-code>**404</error-code>

iv. **<location>**/customErrorPage.html</location>

v. </error-page>

vi. </web-app>

d. Example 2:

i. <web-abb>

ii. <error-page>

iii. **<exception-type>**javax.servlet.ServletException</exception-type>

iv. **<location>**/customErrorPage.html</location>

v. </error-page>

vi. </web-app>

49. Is this <error-page> in web.xml valid?

i. <error-page>

ii. <exception-type>java.lang.Error</exception-type>

iii. <location>/error.html</location>

iv. </error-page>

b. No

50. Describe the **<env-entry>** element?

a. **<env-entry>** is a top level element under <web-app> element. Sub elements of <env-entry> are:

i. **<description>**

ii. **<env-entry-name>** - contains the name of a Deployment Component's environment entry. The name is a JNDI name relative to the java:comp/env context. The name must be unique within a Deployment Component. The uniqueness constraints must be defined within the declared context. Example: minAmount.

iii. **<env-entry-type>** - contains the Java language type of the environment entry. If no injection target is specified, the type is required. Enumerated Values possible are: java.lang.**Boolean**, java.lang.**Byte**, java.lang.**Character**, java.lang.**String**, java.lang.**Short**, java.lang.**Integer**, java.lang.**Long**, java.lang.**Float**, java.lang.**Double**.

iv. **<env-entry-value>** - designates the value of a Deployment Component's environment entry. The value must be a String that is valid for the constructor of the specified type that takes a single String parameter, or for java.lang.Character, a single character.

v. **<injection-target>** - Contains <injection-target-class> and <injection-target-name>.

vi. **<mapped-name>** - A product specific name that this resource should be mapped to. The name of this resource, as defined by the resource's name element or defaulted, is a name that is local to the application component using the resource. (It's a name in the JNDI java:comp/env namespace.) This mapped name is often a global JNDI name, but may be a name of any form. Application servers are not required to support any particular form or type of mapped name, nor the ability to use mapped names. The mapped name is product-dependent and often installation-dependent.

51. What is MIME?

    a. MIME stands for multipurpose internet mail extension. MIME is an industry standard for describing the media types. This could be anything like plain text, images or movies.

52. Describe the <mime-mapping> element in the DD?

    a. The <mime-mapping> element serves to associate file extensions with officially recognized file types.

   b. Example:
      i. <web-app>
      ii. <mime-mapping>
      iii. <extension>xmlhjk</extension>
      iv. <mime-type>text/xml</mime-type>
      v. </mime-mapping>
      vi. </web-app>

53. Give the importance of WAR files in web applications?
    a. WAR stands for web archive. They are files with .war extension used to store the complete web application in a compressed file. The whole contents of the context directory are included in a WAR file without the context directory itself and can be deployed in any context paths.
54. What is the use of META-INF directory of a war file?
    a. A WAR file must contain a META-INF directory, containing a file called MANIFEST.MF. MANIFEST.MF lists dependencies on common code JAR files stored outside of the web application context (but available through a web server's own mechanisms).
55. What are the possible locations of MANIFEST.MF?
    a. META-INF directory of a war
    b. META-INF directory of a jar
56. Is it necessary to have META-INF folder for a war file?
    a. No. Only if needed.
57. Can we access files inside META-INF directory?
    a. When META-INF is present in your web application, the same access rules apply as for WEB-INF. Server-side code is welcome to access resource files in the META-INF directory. However, the web container should reject any attempt at client-side access with the regular "page not found" HTTP error code of 404.
58. What are the files/directories in the META-INF directory recognized and interpreted by the Java 2 Platform to configure applications, extensions, class loaders and services?
    a. MANIFEST.MF
       i. The manifest file that is used to define extension and package related data.
    b. INDEX.LIST

        i. This file is generated by the new "-i" option of the jar tool, which contains location information for packages defined in an application or extension. It is part of the JarIndex implementation and used by class loaders to speed up their class loading process.

    c. x.SF
        i. The signature file for the JAR file. 'x' stands for the base file name.
    d. x.DSA
        i. The signature block file associated with the signature file with the same base file name. This file stores the digital signature of the corresponding signature file.
    e. services/
        i. This directory stores all the service provider configuration files.

59. Compare war file vs jar file? Are they same?
    a. Although a WAR file can be produced in the same way as a JAR file, and has the same underlying file format, it is different.
        i. The most obvious difference is the file extension naming convention: .jar for Java ARchive, and .war for Web (Application) ARchive.
        ii. JARs are packaged in a particular way to make it easy for a running JVM to find and load Java class files. WARs are packaged for a different purpose: to make it as easy as possible for a web container to deploy an application.

60. How can you create a war file using a jar command?
    a. We can go to the context directory of the web application and use the below command:
        i. jar cvf0 mywarfile.war *.*
    b. This creates a WAR file called mywarfile.war in the context directory, containing all the files in the context directory and any subdirectories. The following parameters can be used along with WAR:
        i. **c** - Creates the WAR file.
        ii. **v** - Verbose: outputs messages on the command line telling you about every file added.

    iii. **f** - A WAR file name will be specified (can be omitted).
    iv. **0** - Don’t compress the file (you would usually omit this).

61. How can you extract a war file using a jar command?
    a. We can use the jar command as:
        i. jar xvf mywebapp.war
    b. Different parameters are as follows:
        i. **x** - Extracts the contents of the WAR fi le.
        ii. **v** - Verbose: outputs messages on the command line telling you about every file extracted.
        iii. **f** - A WAR file name will be specified.
        iv. mywarfile.war - The name of the WAR file to be created. It must follow straight after the “xvf ” group (separated by a space).
62. Write the command to display the contents of a war file mywar.war?
    a. Jar tf mywar.war
63. For web applications running under the same server to be distinguished from one another, the ………. must be unique.
    a. context root
64. Briefly describe the **resource-ref** element**?** List down its major sub elements.
    a. The optional **resource-ref** element of web.xml defines a reference lookup name to an external resource. This allows the servlet code to look up a resource by a "virtual" name that is mapped to the actual location at deployment time. Its most sub elements start with <res as:
        i. <res-ref-name>
        ii. <res-type>
        iii. <res-auth>
        iv. <res-sharing-scope>
65. What is the use of ejb-local-ref element?
    a. The ejb-local-ref element of web.xml is used for the declaration of a reference to an enterprise bean's local home.
66. If an exception matches against more than one **<exception-type>** of **<error-page>** element what will happen?

a. When an exception matches against more than one **<exception-type>** of **<error-page>** the more specific one will be selected and corresponding page in the **<location>** element will be sent back.

67. More Notes

a. There's no requirement that the context name match the home directory name, though this is often the case.
b. Any files directly within the context root are meant to be available to users of your web application.
c. All unpacked classes and resources in the /WEB-INF/classes directory, plus classes and resources in JAR files under the /WEB-INF/lib directory are included in classpath and made visible to the containing web application.
d. A web application class loader may NOT override any classes in the java.* and javax.* namespaces.
e. Resources in the WAR class directory or in any of the JAR files within the library directory may be accessed using the getResource() function.