Tag Libraries

1. What are tag libraries?
    a. We can consider a tag library as an abstract collection of possible related actions. A tag library is made available to the JSP page or document using the taglib directive. The URI in the taglib directive need not point to anything existing; it is just used for the tag library resolution.
2. **TLD** stands for…….
    a. tag library descriptor
3. How do you reference the tld file in your JSP page while using a custom tag or JSTL?
    a. A tag has a **name** and a **prefix** (e.g <mytags:round).
    b. The **prefix** (here mytags) must match the value of a **prefix attribute** in a **taglib directive** in the JSP page source: **<%@ taglib prefix="mytags" uri="http://www.anyURL.com/taglibs/mytags" %>.** The **uri** should match an entry in the deployment descriptor web.xml which will tell where to find the tld file.
4. What all values are not allowed in the prefix attribute of a taglib directive?
    a. In a taglib directive, empty prefixes are illegal. If you are developing or using custom tags, you cannot use the tag prefixes jsp, jspx, java, javax, servlet, sun, and sunw, as they are reserved by Sun Microsystems.
5. How the container will find the correct tag handler for a custom tag used in a JSP page?
    a. The prefix of the tag must match the value of the prefix attribute in a taglib directive in the JSP page source like: **<%@ taglib prefix="mytags" uri="http://www.anyURL.com/taglibs/mytags" %>** for a tag usage <mytags:round. The uri should match an entry in the deployment descriptor web.xml which will in turn tell where to find the tld file.
        i. In web.xml file we use the **<taglib-uri>** and **<taglib-location>** subelements of the **<taglib>** element to configure the matching taglib uri and corresponding tld location respectively. **<taglib-uri>—**body value matches the uri quoted in the taglib directive. **<taglib-location>,** gives the actual location in the web application where the TLD file is located. The

<taglib> is a subelement of **<jsp-config>** which sits directly under the root element <web-app>.

b. The entry in the deployment descriptor web.xml, looks like this:

i. **<web-app>**
ii. …
iii. **<jsp-config>**
iv. **<taglib>**
v. **<taglib-uri>**http://www.anyURL.com/taglibs/mytags</taglib-uri>
vi. **<taglib-location>/**WEB-INF/tags/mytags.tld</taglib-location>
vii. </taglib>
viii. </jsp-config>
ix. …
x. </web-app>

6. What are the child elements of <jsp-config> element?
   a. Child elements <jsp-config> are:
      i. <taglib> and
      ii. <jsp-property-group>.
   b. Both of them can have 0 or 1 occurrences.
7. List down the child elements of <taglib> elemenet?
   a. <taglib-uri>
   b. <taglib-location>
8. List down the child elements of <jsp-property-group> element?
   a. <description>
   b. <praram-name>
   c. <icon>
   d. <url-pattern>
   e. <el-ignored>
   f. <page-encoding>
   g. <scripting-invalid>
   h. <is-xml>
   i. <include-prelude>

j. <include-coda>

9. Explain the below code snippet from web.xml file?
    i. <jsp-config>
    ii. <jsp-property-group>
    iii. <url-pattern>/*</url-pattern>
    iv. <el-ignored>true</el-ignored>
    v. <scripting-invalid>true</scripting-invalid>
    vi. </jsp-property-group>
    vii. </jsp-config>
    a. The **<url-pattern>** element works exactly as for servlet declarations. So /* here indicates that all resources in the web application are affected by the settings shown.
    b. **<el-ignored>** when set to true causes expression language to be unevaluated—so treated as template text. The default is false.
    c. **<scripting-invalid>** when set to true causes a translation time error if JSP scriptlets, expressions, or declarations are used. The default is false.
10. List down the important tags inside a tld file?
    a. Mandatory tags are:
        i. **<taglib>,** which is the root element. This must now always contain the attribute/value pair version="2.0" to reference JSP specification level 2.0.
        ii. **<tlib-version>,** is the first element under <taglib>. This denotes the tag library version number which you impose for your own versions of the tag library.
        iii. <short-name>
    b. Optional tags for a tag handler class are:
        i. **<tag>** , which has following sub elements:
            1. **<name>** - name for the tag as used on the jsp page. The name of a tag must remain unique within any tags, EL functions or tag files defined in the tag library.
            2. **<tag-class>** - The fully qualified name of the class implementing the tag functionality.

3. **<body-content>** dictates what can go in the body of the tag. There are four valid values:
    a. **empty**—the body of the tag must be empty.
    b. **tagdependent**—the body of the tag contains something that isn't regular JSP source. The tag handler code works out what to do with it. A typical use is to put an SQL statement in the body.
    c. **scriptless**—the body of the tag does contain regular JSP source, but nothing involving scripting language. So Java language syntax like scriptlets or expressions is forbidden. EL is fine and will be evaluated.
    d. **JSP**—the body of the tag contains any kind of regular JSP source.
4. Attributes if present are represented by **<attribute>** elements. The <attribute> element has below subelements:
    a. **<name>—**a unique name for the attribute. The unique rule applies only to all the attributes within a single tag.
    b. **<required>—**if set to "true," the attribute must appear in the opening tag. If "false," the attribute is optional. The tag handler class can supply a default value.
    c. **<rtexprvalue>—**if set to "true," a run-time (EL) expression value is permitted for the attribute's value setting. A constant value is always valid. If set to "false," and we use run-time expressions a translation error results.
5. For tag files, we have <tag-file> instead of <tag> with subelements <name> and <path>.

11. Is it necessary to have the explicit mapping to the tld files always in the deployment descriptor file?
    a. No, nothing is needed in the deployment descriptor for "implicit" mapping entries for tag libraries. For that:

i. First, you can place files with a .tld extension directly in the / WEB-INF directory or one of its subdirectories.
ii. Alternatively, you can package .tld files in a JAR file that is placed in /WEB-INF/lib. The .tld files within the JAR file must have a path that begins /META-INF. Also, the .tld files must contain the optional <uri> element, which must match the uri of the taglib directive or (if using JSP documents) the namespace value.

b. JSP container searches subdirectories of /WEB-INF and JARs in /WEB-INF/lib for a TLD file with a matching URI.

12. Give the xml syntax equivalent for the tag directive: <%@ taglib prefix="mytags" uri="http://www.anyurl.com/taglibs/mytags" %>?
    a. The tag directive doesn't have a direct equivalent in JSP document syntax—you have to use namespaces instead:
        i. <html xmlns:mytags=http://www.anyurl.com/taglibs/mytags xmlns:jsp="http://java.sun.com/JSP/Page">
13. Why do you not have to place a taglib directive when you use standard actions in a jsp page? Is there any situation where you have to explicitly place them?
    a. The tag library is just understood to be there and available, with a jsp: prefix and hence you do not have to place a taglib directive when you use standard actions in a jsp page.
    b. However, when you use JSP document syntax, taglib directives disappear in favor of XML namespace definitions. In that case, even standard actions must be explicitly referenced with a namespace—in exactly the same way as your own custom tag libraries.
14. Can we have Sun or JAVA for your own custom tag library declarations?
    a. There are certain "reserved prefixes" that you can't use for your own custom tag library declarations, whether as directives or namespaces: jsp, jspx, java, javax, servlet, sun, and sunw. They are case sensitive. So Sun or JAVA are not reserved and hence are legal.

15. The best way to set up mappings from your JSP pages to the TLDs lodged in your web application is through the **web.xml** entries even though there are alternative ways to do so without using deployment descriptor. Why?
    a. This makes the tag library mapping explicit and obvious to anyone reading the deployment descriptor.