EL Functions

1. List down the stages involved in the process of developing and using an EL function?
    a. Writing a Java class containing the method underpinning the EL function
    b. Defining the function within a tag library definition (TLD) file
    c. Providing details of where to find the TLD file in the deployment descriptor, web.xml
    d. Referencing the TLD file in your JSP page source and using the EL functions from it
2. What are the major criteria for writing an EL function?
    a. The only requirement of a method that acts as an EL function is that it should be declared (1) public and (2) static. This means that any existing public static method in any class is already a candidate for EL function-hood. Most of the functions in java.lang.Math, for example, can be made available as EL functions without having to write a line of code. However you will have to do the remaining steps like defining function within TLD, providing TLD details in web.xml and referencing the TLD file from JSP.
3. Is it legal to return nothing (void) from an EL function?
    a. There's nothing wrong with it—provided you state that void is returned from the function signature in the TLD:
        i. <function-signature>void voidFunction(int)</function-signature>
4. Why would you want an EL function that doesn't return anything in a JSP page?
    a. You might need to call a function that causes some effect in your application (sets up some attributes, perhaps) but shouldn't or mustn't return anything to the JSP page.
5. In the process of developing and using an EL function, how would you define the function in the TLD file?
    a. You use the function element to do so. The subelements of <function> are as follows:
        i. <description> This is an optional description for the function.
        ii. <name> This is the name of the function as it will be used within EL expressions. This doesn't have to match the Java method name.

iii. <function-class> This is the fully qualified name of the Java class containing the implementation of the function.

iv. <function-signature> This is the signature of the function with the following rules:

1. Qualifiers are omitted as the method must be always public and static.
2. A return type must be supplied (or void)—just like Java. If the function returns an object type (rather than a primitive), the fully qualified name of the class must be given. This applies even to classes in java.lang (java.lang.String, not just String).
3. The method name follows, and this has to match the method name in the Java class. Just as in Java, a pair of parentheses follows the method name—to enclose the parameters.
4. Parameters must be listed in order, following the order within the Java method. However, only the types are provided—the parameters don't have an accompanying name in the TLD signature. Primitives are listed as in Java, and—as for the return type—objects must have their fully qualified name.

6. Can you mix EL functions and custom tags in the same TLD?
   a. You can mix EL functions and custom tags in the same TLD. The only rule is on names: Each must be unique across all functions, custom tags, and other top-level elements within the TLD.
7. In the process of developing and using an EL function, how would you reference the TLD in the JSP file and use it?
   a. For referencing the TLD file, You can either use the taglib directive (for JSP syntax) or reference the taglib in a namespace (JSP document XML syntax).
      i. <%@ taglib prefix="mytags" uri="http://www.xyz.com/tags/mytags" %>
      ii. <html xmlns:mytags=" http://www.xyz.com/tags/mytags " xmlns:jsp="..">
   b. However, using the function is different. You use EL rather than tag syntax.
      i. ${mytags:round(9.21 / 3, 2)}