Descriptive questions on JSP, Servlet, EL, JSTL, JSF

1: What do you understand by JSP Actions?
Answer: JSP actions are XML tags that direct the server to use existing components or control the behavior of the JSP engine. JSP Actions consist of a typical (XML-based) prefix of "jsp" followed by a colon, followed by the action name followed by one or more attribute parameters.
There are six JSP Actions:
<jsp:include/>
<jsp:forward/>
<jsp:plugin/>
<jsp:usebean/>
<jsp:setProperty/>
<jsp:getProperty/>

2: What is the difference between <jsp:include page = ... > and <%@ include file = ... >?.
Answer: Both the tag includes the information from one page in another. The differences are as follows:
<jsp:include page = ... >: This is like a function call from one jsp to another jsp. It is executed (the included page is executed and the generated html content is included in the content of calling jsp) each time the client page is accessed by the client. This approach is useful to for modularizing the web application. If the included file changed then the new content will be included in the output.

<%@ include file = ... >: In this case the content of the included file is textually embedded in the page that have <%@ include file=".."> directive. In this case in the included file changes, the changed content will not included in the output. This approach is used when the code from one jsp file required to include in multiple jsp files.

The <%@include file="abc.jsp"%> directive acts like C "#include", pulling in the text of the included file and compiling it as if it were part of the including file. The included file can be any type (including HTML or text).

The <jsp:include page="abc.jsp"> tag compiles the file as a separate JSP file, and embeds a call to it in the compiled JSP.

There's a huge difference. As has been mentioned, `<%@ include` is a static include, `<jsp:include`is a dynamic include. Think of it as a difference between a macro and a function call (if you are familiar with those terms). Another way of putting it, a static include is exactly the same thing as copy-pasting the exact content of the included file (the "code") at the location of the `<%@ include` statement (which is exactly what the JSP compiler will do.
A dynamic include will make a *request* (using the request dispatcher) that will *execute* the indicated page and then include the output from the page in the output of the calling page, in place of the`<jsp:include` statement.

The big difference here is that with a dynamic include, the included page will execute in it's own pageContext. And since it's a request, you can send parameters to the page the same way you can send parameters along with any other request. A static include, on the other hand, is just a piece of code that will execute inside the context of the calling page. If you statically include the same file more than once, the code in that file will exist in multiple locations on the calling page so something like

```
<%
int i = 0;
%>
```

would generate a compiler error (since the same variable can't be declared more than once).

3: Identify the advantages of JSP over Servlet.

a) Embedding of Java code in HTML pages
b) Platform independence
c) Creation of database-driven Web applications
d) Server-side programming capabilities

4: What are all the different scope values for the <jsp:useBean> tag?
Answer:<jsp:useBean> tag is used to use any java object in the jsp page. Here are the scope values for <jsp:useBean> tag:
a) page
b) request
c) session and
d) application

5: What is JSP Scriptlet?
Answer: JSP Scriptlet is jsp tag which is used to enclose java code in the JSP pages. Scriptlets begins with **<%** tag and ends with **%>** tag. Java code written inside scriptlet executes every time the JSP is invoked.
Example:

```
 <%
 //java codes
  String userName=null;
  userName=request.getParameter("userName");
  %>
```

6: How you will handle the runtime exception in your jsp page?
Answer: The **errorPage** attribute of the page directive can be used to catch run-time exceptions automatically and then forwarded to an error processing page.
For example:
<%@ page errorPage="customerror.jsp" %>
above code forwards the request to "customerror.jsp" page if an uncaught exception is

encountered during request processing. Within "customerror.jsp", you must indicate that it is an error-processing page, via the directive: <%@ page isErrorPage="true" %>.

7. What are the implicit objects in JSP?
Implicit objects are created by the web container and contain information related to a particular request, page, or application. They are request, response, pageContext, session, application, out, config, page and exception.

8. Why we use Servlets?
Ans: A Servlet is a server-side component that is capable of dynamically processing requests and constructing response in a protocol independent manner.
Servlet are best used in situations where a great deal of programmatic control is required, Such as decision making, database querying or accessing other enterprise resource.
Servlet technology has been an extremely popular choice for building dynamic web applications such as e-commerce sites, online banking, and news portals.

9. Difference between GET and POST in Java Servlets?
In GET your entire form submission can be encapsulated in one URL, like a hyperlink. query length is limited to 260 characters, not secure, faster, quick and easy.
In POST Your name/value pairs inside the body of the HTTP request, which makes for a cleaner URL and imposes no size limitations on the form's output. It is used to send a chunk of data to the server to be processed, more versatile, most secure.

10. What are the advantages of JSF?
Ans: The JSF specification lists the following ways that JSF helps web-application developers to create user
interfaces (UIs):
- Makes it easy to construct a UI from a set of reusable UI components
- Simplifies migration of application data to and from the UI
- Helps manage UI state across server requests
- Provides a simple model for wiring client-generated events to server-side application code
- Allows custom UI components to be easily built and reused

11. Write the syntax of EL expression? Why we use them?
Ans: No matter where the EL is used, it's always invoked in a consistent manner, via the construct ${expr} or
#{expr}, where expr is the EL expression that is wished to have evaluated. The syntax of ${expr} and #{expr} are equivalent and can be used interchangeably. However, when used with some other Java Platform, Enterprise EditionAPI, the other API may enforce restrictions on the use of ${expr} and #{expr}. Specifically, when used with JSP pages, the two forms cannot be used interchangeably. Within a JSP page, ${expr} is used for expressions that are evaluated immediately, whereas #{expr} is used for expressions for which evaluation is deferred.

12. What are tags in JSF?
JSF application typically uses JSP pages to represent views. JSF provides useful special tags to enhance these views. Each tag gives rise to an associated component. JSF (Sun Implementation) provides 43 tags in two standard JSF tag libraries: 1. JSF Core Tags Library 2. JSF Html Tags Library Even a very simple page uses tags from both libraries. These tags can be used adding the following lines of code at the head of the page. <%@ taglib uri="http://java.sun.com/jsf/core " prefix="f" %> (For Core Tags) <%@ taglib uri="http://java.sun.com/jsf/html " prefix="h" %> (For Html Tags)

13. What is JSF life cycle and its phases?
T he series of steps followed by an application is called its life cycle. A JSF application typically follows six steps in its life.
1. Restore view phase
2. Apply request values phase
3. Process validations phase
4. Update model values phase
5. Invoke application phase
6. Render response phase

14. What is JavaServer Pages Standard Tag Library (JSTL)?
A tag library that encapsulates core functionality common to many JSP applications. JSTL has support for common, structural tasks such as iteration and conditionals, tags for manipulating XML documents, internationalization and locale-specific formatting tags, SQL tags, and functions.

15. What is deployment descriptor?
An XML file provided with each module and J2EE application that describes how they should be deployed. The deployment descriptor directs a deployment tool to deploy a module or application with specific container options and describes specific configuration requirements that a deployer must resolve.

16. What is Tag Library Descriptor (TLD)?
Ans: To use a custom tag library, the web container needs to be made aware of specific information about the library itself. A special file called a tag library descriptor (TLD) is used for this purpose. The TLD file contains essential information about each of the custom actions or tags that are included inside the tag library, such as which attributes are permitted by which tags, whether the tags accept body content, and so on.

17. Write the Expanded Directory Format? Pg-12
The Java Servlet specification defines a special structure that all web applications must follow so that servlet/JSP containers know exactly where to find the resources that compose the web application. Most containers allow web applications to be deployed in one of the following two forms:

• **Expanded directory format**: The web application in its predefined structure is simply copied into the container's deployment directory.

• **Web ARchive file (WAR)**

18. What is Taglib? Write the syntax of taglib?

The JavaServer Pages API allows us to define custom JSP tags that look like HTML or XML tags and a tag library is a set of user-defined tags that implement custom behavior. The **taglib** directive declares that your JSP page uses a set of custom tags, identifies the location of the library, and provides a means for identifying the custom tags in JSP page.

The general syntax :

```
  <%@ taglib uri="uri" prefix="prefixOfTag" >
```

Example:

<%@ taglib prefix="f" uri="http://java.sun.com/jsf/core"%>

19. How do you configure a Servlet?

Several list of configuration for servlet in web.xml (deployment descriptor):

1. Configuring and Mapping a Servlet
2. Servlet Init Parameters
3. Servlet Load-on-Startup
4. Context Parameters

Most important is to Configuring and Mapping a Servlet

```
<web-app>

  <servlet>
    <servlet-name>controlServlet</servlet-name>
    <servlet-class>com.jenkov.butterfly.ControlServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>controlServlet</servlet-name>
    <url-pattern>*.html</url-pattern>
  </servlet-mapping>
</web-app>
```

First to configure the servlet. This is done using the `<servlet>` element. Here you give the servlet a name, and writes the class name of the servlet.

Second, map the servlet to a URL or URL pattern. This is done in the `<servlet-mapping>` element. In the above example, all URL's ending in `.html` are sent to the servlet.

Other possible servlet URL mappings are:

```
/myServlet

/myServlet.do

/myServlet*
```

20. Write the default value of EL expression. Pg-98

Default values are type-correct values that are assigned to a subexpression when there is a problem, and errors are exceptions to be thrown (and then handled by the standard JSP error-handling process). An example of such a default value is 'infinity'. This value is

assigned to an expression that results in a divide by zero. For example, the following piece of EL will display infinity rather than causing an error:
${2/0}
The equivalent Java expression would throw an ArithmeticException.

21. What is the type of request /response handily by JSF? Pg-185

Several kinds of request/response cycles can occur in a JSF-enabled application. We are concerned with these three request/response pairs:
• Non-JSF request generates JSF response
• JSF request generates JSF response
• JSF request generates non-JSF response
Of course, you can also have a non-JSF request that generates a non-JSF response. Because this does not involve JSF in any way, the JSF life cycle does not apply.

22. Write the name of JSF libraries?

• **Six JSF JARs:** commons-beanutils.jar, commons-collections.jar, commons-digester.jar, commons-logging.jar, jsf-api.jar, and jsf-impl.jar
• **Two JSTL JARs:** jstl.jar and standard.jar

23. What is the job of faces-config.xml class?

faces-config.xml can contain a lot of information about a web application. Main two things are identify the flow of control and identify the JavaBean used by the application.

24. Write the event handling process in JSF?

Java Server Faces technology supports three kinds of events:

- action events
- value-change events
- data-model events.

Different ways of handling events in JSF

- Implement a method in a backing bean to handle the event and refer that method with a JSF EL expression from the appropriate attribute of the component
- Implement an event listener to handle the event and registers the listener on a component by nesting a listener tag inside the UI component tag.

25. What do you mean by MVC in JSF?

In the big architectural picture, JSF code is the V:
M - Business domain/Service layer (e.g. EJB/JPA/DAO)
V - Your JSF code
C - FacesServlet
In the smaller developer picture, the architectural **V** is in turn dividable as follows:
M - Entity
V - JSP/XHTML page
C - Managed bean