Descriptive question on JSP

1. **What do you understand by JSP Actions?**

   JSP actions are 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:forward page = ... > and response.sendRedirect(url)?**

   - The element forwards the request object containing the client request information from one JSP file to another file. The target file can be an HTML file, another JSP file, or a servlet.
   - sendRedirect sends HTTP temporary redirect response to the browser, and browser creates a new request to go the redirected page. The response.sendRedirect also kills the session variables.

3. **Identify the advantages of JSP over Servlet? -: 1**

JSP can contain HTML, JavaScript, XML and Java Code whereas Servlets can contain only Java Code, making JSPs more flexible and powerful than Servlets.
However, Servlets have their own place in a J2EE application and cannot be ignored altogether. They have their strengths too which cannot be overseen.

4. **What are all the different scope values for the <jsp:useBean> tag?**

< 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?-: 2**

JSP Scriptlets is a term used to refer to pieces of Java code that can be embedded in a JSP PAge. Scriptlets begins with <% tag and ends with %> tag. Java code written inside scriptlet executes every time the JSP is invoked.

6. **What you will handle the runtime exception in your jsp page?-: 3**

The errorPage attribute of the page directive can be used to catch run-time exceptions automatically and then forwarded to an error processing page. You can define the error page to which you want the request forwarded to, in case of an exception, in each JSP Page.

7. What are the implicit objects in JSP?-: 4

Implicit objects are the objects available to the JSP page. These objects are created by Web container and contain information related to a particular request, page, or application. The JSP implicit objects are:
application,config,exception,out,page,pageContext,request,response and session

8. Why we use Servlets?

Servlets are used to process the client requests.
* A Servlet can handle multiple requests concurrently and be used to develop high performance of system
* A Servlet can be used to load balance among serveral servers, as Servlet can easily forward request.

9. Write the syntax of EL expression? Why we use them?

-Syntax of EL Expression:
EL expressions are always within the braces { ..... } and prefixed with the $ or, # sign as:

<jsp:useBean id="bean" class="class"/>
${bean.name}
-Expression language use to easily access application data stored in JavaBeans components. For example, the JSP expression language allows a page author to access a bean using simple syntax such as `${name}` for a simple variable or `${name.foo.bar}` for a nested property.

10. What is Java Server Pages Standard Tag Library (JSTL)?

Java Server Pages Standard Tag Library (JSTL) encapsulates as simple tags the core functionality common to many Web applications. It also provides a framework for integrating existing custom tags with JSTL tags.
It is a collection of four tag library. They are –
(a)Core (b) Internationalization (118n) and formatting
(c) Relational database access (d) XML processing

11. What is deployment descriptor?-:8

The deployment descriptor(web.xml) is an xml file that contains the basic and most important information that is required to deploy a web application (Servlet)
Without this, the web server would not know, which requests to entertain/consider as requests to access this servlet.

12. What is Tag Library Descriptor (TLD)?

A tag library descriptor is an XML document that contains information about a library as a whole and about each tag contained in the library. TLDs are used by a web container to validate the tags and by JSP page development tools. A TLD must begin with a root `taglib` element.
The syntax for the taglib directive is as follows:
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

13. Difference between GET and POST in Java Servlet?-:5

Get sends information from the browser to the Servlet as contents appended to the query string in the URL while Post uses hidden variables
* Because of the above point, post is a lot safer than get
* Get has a size limitation - i.e., we can send only approximately 1 Kb of data while Post can send a significantly higher amount of data
* Get is the most common type of sending data from a browser to a servlet, while Post is gaining popularity because of its size and safety which is better than the get.

14. Write the Expanded Directory Format?

The web application in its predefined structure is simply copied into the container's deployment directory.

Root-web-context

META-INF

Context.xml

WEB-INF

Web.xml

Lib-all libraries

Classes-all package and classes

Index.jsp and other jsp pages

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

A tag library (commonly known as a taglib) is simply a collection of one or more custom tags that are generally related in some way. For example the JSP 2.0 specification introduced a standard tag library known as the JSTL.

For example, suppose the **custlib** tag library contains a tag called **hello**. Your JSP file as follows:

```
<%@ taglib
uri="http://www.example.com/custlib"
prefix="mytag" %>
<html>
<body><mytag:hello/></body></html>
```

16. How do you configure a Servlet?

First you 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, you 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.

```
<servlet>
<servlet-name>controlServlet</servlet-name>
 <servlet-class>org.idb.j2ee.ControlServlet</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>controlServlet</servlet-name>
<url-pattern>*.html</url-pattern>
</servlet-mapping>
```

17. **Write the default value of EL expression.**

Default values are type-correct values that are assigned to a sub-expression when there is a problem and errors are exceptions to be thrown. An example of such a default value is ‘infinity’. This value is assigned to an expression that results in a divide by zero. Exp: ${2/0}

18.What are the advantages of JSF?-:6

Advantages and Disadvantages – JSF

JSF is becoming more popular framework for user interface layer development, many architects and companies assuming that Struts is becoming outdated and JSF is catching up the market. I am not sure whether it is true at this point of time. However I would like to express my critic on the **advantages and disadvantages of JSF**.

also read:

- Introduction to JSF

- JSF Interview Questions
- Request Processing Lifecycle phases in JSF

Update (25-Sep-2013): In the recent trend, many companies started using the Java Script frameworks like jQuery, DOJO, etc. Still there are clear advantages on using the JSF, but the technology trend is moving in different direction after the evolution of Rich Internet Applications (RIA) concept.

Advantages of JSF

- Big vendors (Oracle, IBM, JBoss, etc) backing JSF implementation like EJB. Can expect good level of support and quality components from these vendors.
- By design and concept it allows to create reusable components. That will help to improve productivity and consistency.
- Many quality and ready to use components are available from Apache, Richfaces, Infragistics, Oracle, etc.
- The concept of action and action listener for button invocation is good.
- Has very good support for EL expression that improves the user interface code readability.
- The concept the validator and converter is excellent. Unlike struts JSF keeps the validation logic very close to the component declaration.
- JavaScript code are embedded as part of the component; this keep less confusion for developers and more re-usability on JavaScript code.
- With JSF 2.0 release, there is great looking third party libraries are released. The popular ones are PrimeFaces, Openfaces, etc.

Disadvantages of JSF

- Steep learning curve is one of the main dis-advantage of the JSF. However, if you are familiar with HTML and CSS concepts, it is going to be a cake walk for you

- There is no benchmarking report or promise from Sun Microsystems about the performance of JSF framework. By seeing their concept I believe it is not suitable for high performance application.

- The specification doesn't consider bookmarking facility.

- Hardly a very few examples available for developing dynamic pages including new component and removing a component from a page based on business rule.

- Every button or link clicked results in a form post. That's just wrong why can't I have true links like the web is supposed to? Form submission for page navigation make complex coding for simple requirement like Cancel button. Read here to know the work around.

- Datatable component requires same data from bean on restore view phase. If the data retrieved from database, this will have impact on performance. Click here to know more about this issue.

- There is no tight coupling between managed bean and phase listener. This is a major drawback of JSF which makes JSF phase listener feature unusable.

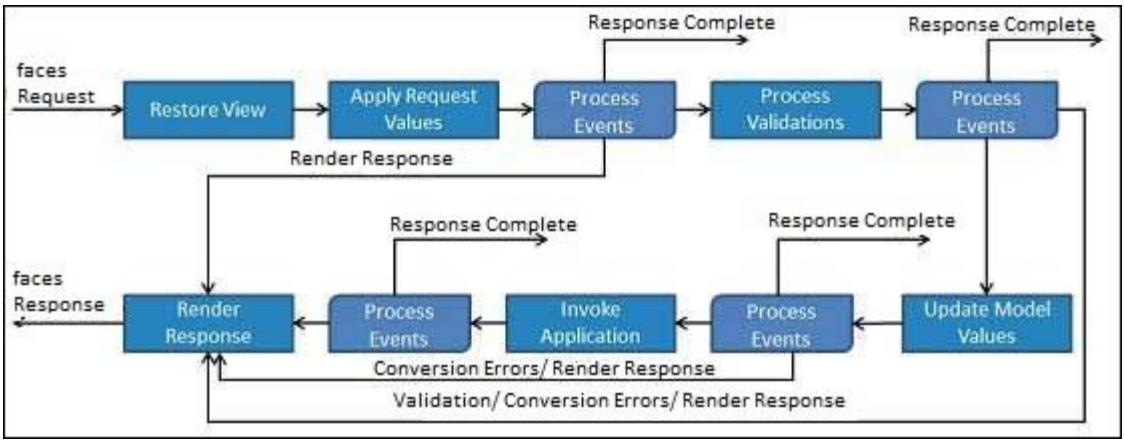
- Default error message is not good. Need to customize the default error message.

- Not Scalable. It uses session object to store the component state across the request. In server farm environment it is too costly to replicate the session data.

19.What is JSF life cycle and its phases?-:7

JSF application life cycle consists of six phases which are as follows −

- Restore view phase
- Apply request values phase; process events
- Process validations phase; process events
- Update model values phase; process events
- Invoke application phase; process events
- Render response phase



The six phases show the order in which JSF processes a form. The list shows the phases in their likely order of execution with event processing at each phase.

Phase 1: Restore view

JSF begins the restore view phase as soon as a link or a button is clicked and JSF receives a request.

During this phase, JSF builds the view, wires event handlers and validators to UI components and saves the view in the FacesContext instance. The FacesContext instance will now contain all the information required to process a request.

Phase 2: Apply request values

After the component tree is created/restored, each component in the component tree uses the decode method to extract its new value from the request parameters. Component stores this value. If the conversion fails, an error message is generated and queued on FacesContext. This message will be displayed during the render response phase, along with any validation errors.

If any decode methods event listeners called renderResponse on the current FacesContext instance, the JSF moves to the render response phase.

Phase 3: Process validation

During this phase, JSF processes all validators registered on the component tree. It examines the component attribute rules for the validation and compares these rules to the local value stored for the component.

If the local value is invalid, JSF adds an error message to the FacesContext instance, and the life cycle advances to the render response phase and displays the same page again with the error message.

Phase 4: Update model values

After the JSF checks that the data is valid, it walks over the component tree and sets the corresponding server-side object properties to the components' local values. JSF will update the bean properties corresponding to the input component's value attribute.

If any updateModels methods called renderResponse on the current FacesContext instance, JSF moves to the render response phase.

Phase 5: Invoke application

During this phase, JSF handles any application-level events, such as submitting a form/linking to another page.

Phase 6: Render response

During this phase, JSF asks container/application server to render the page if the application is using JSP pages. For initial request, the components represented on the page will be added to the component tree as JSP container executes the page. If this is not an initial request, the component tree is already built so components need not be added again. In either case, the components will render themselves as the JSP container/Application server traverses the tags in the page.

After the content of the view is rendered, the response state is saved so that subsequent requests can access it and it is available to the restore view phase.

20.What do you mean by MVC in JSF?-:9

Part of the reason why it's often not entirely clear in JSF and many other web frameworks which parts of it correspond to which part of MVC, is that the MVC pattern was originally devised for desktop applications.

In a desktop application, the nodes M, V and C are a maximum connected graph, meaning each part can communicate with every other part. E.g. if the model changes, it can push this change to the view. This is particularly visible in case there are multiple representations of the view in a desktop application. Change one, and see the other update in real-time.

Due to the client/server and request/response nature of web applications, classic MVC doesn't map 1:1 to most web frameworks.

Specifically, in JSF the mapping is as follows:

- **Model** - The Services/DAOs plus the entities they produce and consume. The entry point to this is the managed bean, but in Java EE (of which JSF is a part) these artifacts are typically implemented by EJB and JPA respectively.
- **View** - The UI components and their composition into a full page. This is fully in the domain of JSF and implemented by JSF `UIComponent`s and Facelets respectively.
- **Controller** - The traffic cop that handles commands and incoming data from the user, routes this to the right parts and selects a view for display. In JSF one doesn't write this controller, but it's already provided by the framework (it's the `FacesServlet`).

Especially the last part is frequently not well understood: In JSF you don't implement a controller. Consequently, a backing bean or any other kind of managed bean is **NOT** the controller.

The first part (the model) is also not always clearly understood. Business logic may be implemented by EJB and JPA, but from the point of view of JSF everything that is referenced by a value binding is the model. This is also where the name of one of the JSF life-cycle phases comes from: `Update Model`. In this phase JSF pushes data from the UI components into the model. In that sense, (JSF) managed beans are thus the model.

Although JSF itself doesn't explicitly define the concept, there is an often recurring and specific usage of managed beans called the **backing bean**.

For JSF a backing bean is still the model, but practically it's a plumbing element that sits in the middle of the Model, View and Controller. Because it performs some tasks that may be seen as some controller tasks, this is often mistaken to be the controller. But, as explained before

this is not correct. It can also perform some model tasks and occasionally do some view logic as well.

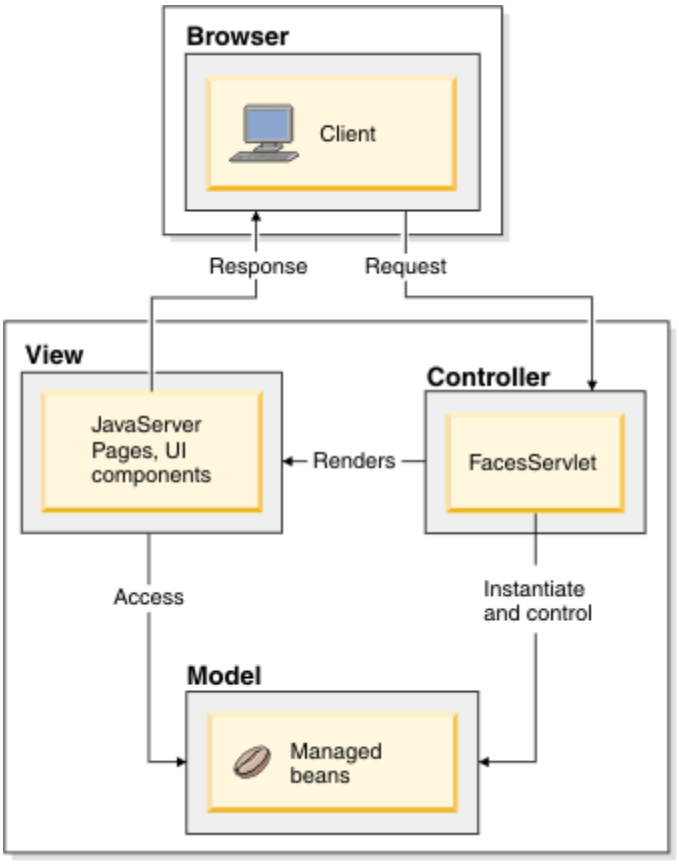
MVC

Applications built with JSF are intended to follow the model-view-controller (MVC) architectural pattern. The JSF framework implements the Model-View-Controller (MVC) architecture ensuring that applications are well designed and easier to maintain..

According to the MVX pattern, a software component should be separated into layers along the following lines:

Model	Encapsulates the information (data) and the methods to operate on that information (business logic). Managed beans define the model of a JSF application. These Java™ beans typically interface with reusable business logic components or external systems, such as a mainframe or database.
View	Presents the model. JSPs make up the view of a JSF web application. These JSPs are built using predefined and custom-made UI components and by connecting these components to the model.
Controller	Processes user events and drives model and view updates. The Faces Servlet, which handles the request processing lifecycle defined by JSF, drives the application flow.

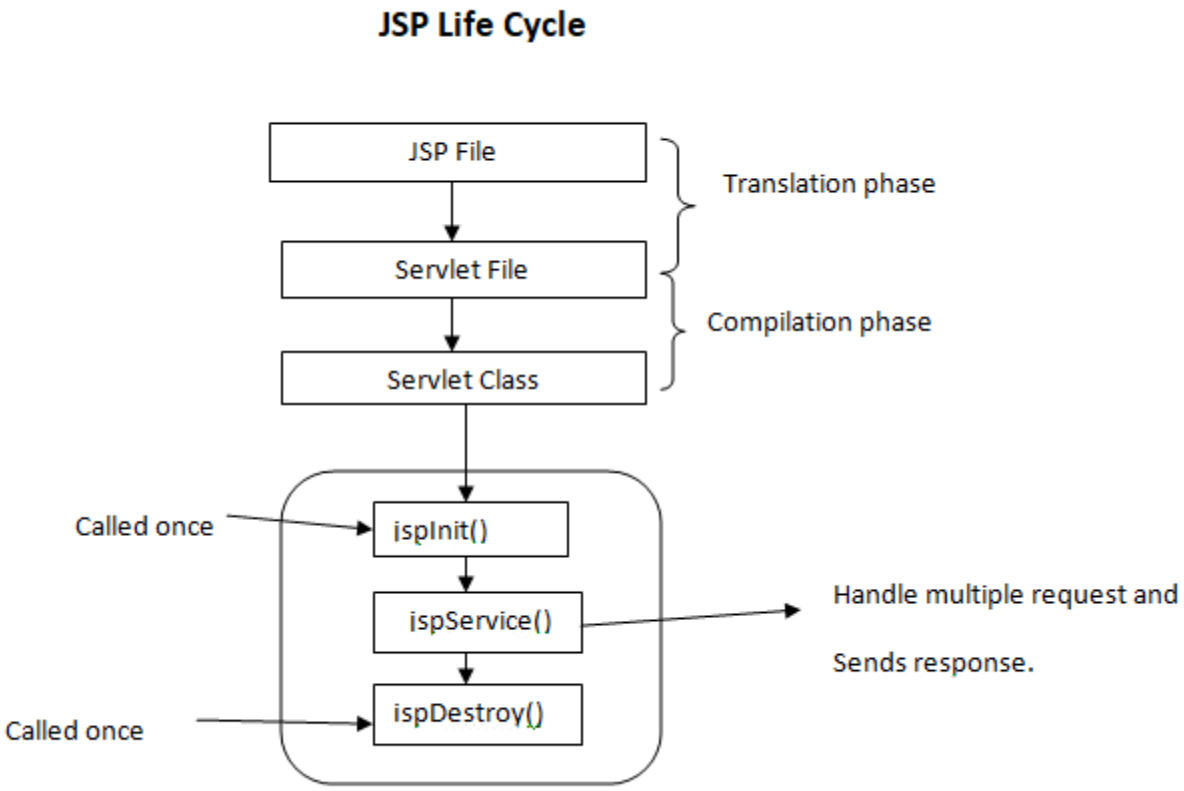
JSF enables Java programmers to focus on backend data encapsulation development that is then integrated with the UI. It enables Web page creators to create UI by assembling prebuilt JSF components that already contain the necessary logic.



21.What is JSP life cycle?

Life cycle of JSP

A Java Server Page life cycle is defined as the process started with its creation which later translated to a servlet and afterward servlet lifecycle comes into play. This is how the process goes on until its destruction.



Following steps are involved in JSP life cycle:

- Translation of JSP page to Servlet
- Compilation of JSP page(Compilation of JSP into test.java)
- Classloading (test.java to test.class)
- Instantiation(Object of the generated Servlet is created)
- Initialization(jspInit() method is invoked by the container)
- Request processing(_jspService()is invoked by the container)
- JSP Cleanup (jspDestroy() method is invoked by the container)

We can override jspInit(), jspDestroy() but we can't override _jspService() method.

Translation of JSP page to Servlet :

This is the first step of JSP life cycle. This translation phase deals with Syntactic correctness of JSP. Here test.jsp file is transllated to test.java.

Compilation of JSP page :

Here the generated java servlet file (test.java) is compiled to a class file (test.class).

Classloading :

Servlet class which has been loaded from JSP source is now loaded into container.

Instantiation :

Here instance of the class is generated. The container manages one or more instance by providing response to requests.

Initialization :

jspInit() method is called only once during the life cycle immediately after the generation of Servlet instance from JSP.

Request processing :

_jspService() method is used to serve the raised requests by JSP.It takes request and response object as parameters.This method cannot be overridden.

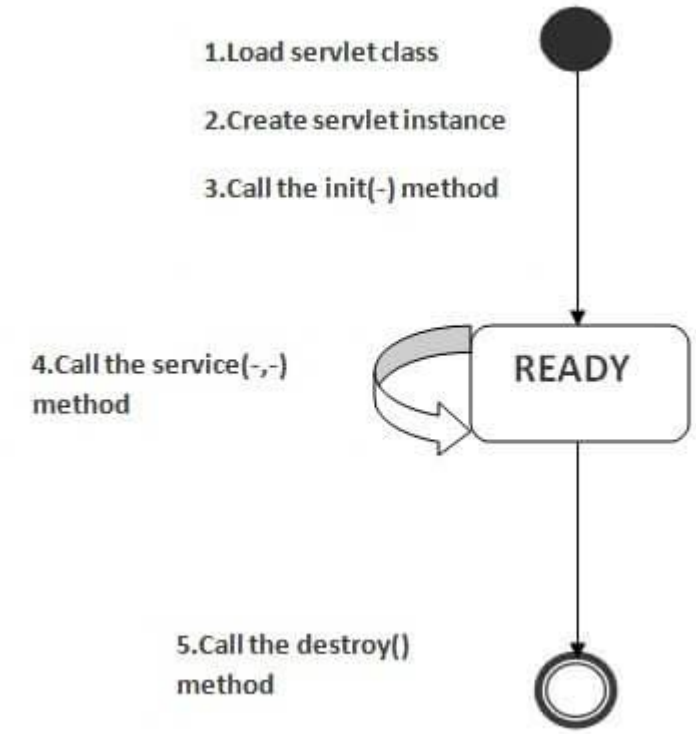
JSP Cleanup :

In order to remove the JSP from use by the container or to destroy method for servlets jspDestroy()method is used. This method is called once, if you need to perform any cleanup task like closing open files, releasing database connections jspDestroy() can be overridden.

22.What is Servlet life cycle?

The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:

1. Servlet class is loaded.
2. Servlet instance is created.
3. init method is invoked.
4. service method is invoked.
5. destroy method is invoked.



As displayed in the above diagram, there are three states of a servlet: new, ready and end. The servlet is in new state if servlet instance is created. After invoking the init() method, Servlet comes in the ready state. In the ready state, servlet performs all the tasks. When the web container invokes the destroy() method, it shifts to the end state.

---

1) Servlet class is loaded

The classloader is responsible to load the servlet class. The servlet class is loaded when the first request for the servlet is received by the web container.

---

2) Servlet instance is created

The web container creates the instance of a servlet after loading the servlet class. The servlet instance is created only once in the servlet life cycle.

---

3) init method is invoked

The web container calls the init method only once after creating the servlet instance. The init method is used to initialize the servlet. It is the life cycle method of the javax.servlet.Servlet interface. Syntax of the init method is given below:

```
1. public void init(ServletConfig config) throws ServletException
```

---

4) service method is invoked

The web container calls the service method each time when request for the servlet is received. If servlet is not initialized, it follows the first three steps as described above then calls the service method. If servlet is initialized, it calls the service method. Notice that servlet is initialized only once. The syntax of the service method of the Servlet interface is given below:

```
1. public void service(ServletRequest request, ServletResponse respo
   nse)
```

2. throws ServletException, IOException

---

5) destroy method is invoked

The web container calls the destroy method before removing the servlet instance from the service. It gives the servlet an opportunity to clean up any resource for example memory, thread etc. The syntax of the destroy method of the Servlet interface is given below:

1. public void destroy()