**1. What is JSF (or JavaServer Faces)?**

A server side user interface component framework for Java™ technology-based web applications.JavaServer Faces (JSF) is an industry standard and a framework for building component-based user interfaces for web applications.   
  
JSF contains an API for representing UI components and managing their state; handling events, server-side validation, and data conversion; defining page navigation; supporting internationalization and accessibility; and providing extensibility for all these features.

New to JSF ? Check [**JavaServer Faces (JSF) tutorial**](http://www.developersbook.com/jsf/jsf-tutorial/jsf-tutorial.php)

**2. What are the advantages of JSF?**

The major benefits of JavaServer Faces technology are:

* JavaServer Faces architecture makes it easy for the developers to use. In JavaServer Faces technology, user interfaces can be created easily with its built-in UI component library, which handles most of the complexities of user interface management.
* Offers a clean separation between behavior and presentation.
* Provides a rich architecture for managing component state, processing component data, validating user input, and handling events.
* Robust event handling mechanism.
* Events easily tied to server-side code.
* Render kit support for different clients
* Component-level control over statefulness
* Highly 'pluggable' - components, view handler, etc
* JSF also supports internationalization and accessibility
* Offers multiple, standardized vendor implementations

**3. What are differences between struts and JSF?**

In a nutshell, Faces has the following advantages over Struts:

* Eliminated the need for a Form Bean
* Eliminated the need for a DTO Class
* Allows the use of the same POJO on all Tiers because of the Backing Bean

**The primary advantages of Struts as compared to JavaServer Faces technology are as follows:**

* Because Struts is a web application framework, it has a more sophisticated controller architecture than does JavaServer Faces technology. It is more sophisticated partly because the application developer can access the controller by creating an Action object that can integrate with the controller, whereas JavaServer Faces technology does not allow access to the controller. In addition, the Struts controller can do things like access control on each Action based on user roles. This functionality is not provided by JavaServer Faces technology.
* Struts includes a powerful layout management framework, called Tiles, which allows you to create templates that you can reuse across multiple pages, thus enabling you to establish an overall look-and-feel for an application.
* The Struts validation framework includes a larger set of standard validators, which automatically generate both server-side and client-side validation code based on a set of rules in a configuration file. You can also create custom validators and easily include them in your application by adding definitions of them in your configuration file.

**The greatest advantage that JavaServer Faces technology has over Struts is its flexible, extensible UI component model, which includes:**

* A standard component API for specifying the state and behavior of a wide range of components, including simple components, such as input fields, and more complex components, such as scrollable data tables. Developers can also create their own components based on these APIs, and many third parties have already done so and have made their component libraries publicly available.
* A separate rendering model that defines how to render the components in various ways. For example, a component used for selecting an item from a list can be rendered as a menu or a set of radio buttons.
* An event and listener model that defines how to handle events generated by activating a component, such as what to do when a user clicks a button.
* Conversion and validation models for converting and validating component data.

**4. What are the available implementations of JavaServer Faces?**

The main implementations of JavaServer Faces are:

* Reference Implementation (**RI**) by Sun Microsystems.
* Apache **MyFaces** is an open source JavaServer Faces (JSF) implementation or run-time.
* **ADF Faces** is Oracle’s implementation for the JSF standard.

**6. What typical JSF application consists of?**

A typical JSF application consists of the following parts:

* JavaBeans components for managing application state and behavior.
* Event-driven development (via listeners as in traditional GUI development).
* Pages that represent MVC-style views; pages reference view roots via the JSF component tree.

**7. What Is a JavaServer Faces Application?**

JavaServer Faces applications are just like any other Java web application. They run in a servlet container, and they typically contain the following:

* JavaBeans components containing application-specific functionality and data.
* Event listeners.
* Pages, such as JSP pages.
* Server-side helper classes, such as database access beans.

**In addition to these items, a JavaServer Faces application also has:**

* A custom tag library for rendering UI components on a page.
* A custom tag library for representing event handlers, validators, and other actions.
* UI components represented as stateful objects on the server.
* Backing beans, which define properties and functions for UI components.
* Validators, converters, event listeners, and event handlers.
* An application configuration resource file for configuring application resources.

**8. What is Managed Bean?**

JavaBean objects managed by a JSF implementation are called managed beans. A managed bean describes how a bean is created and managed. It has nothing to do with the bean's functionalities.

**9. What is Backing Bean?**

Backing beans are JavaBeans components associated with UI components used in a page. Backing-bean management separates the definition of UI component objects from objects that perform application-specific processing and hold data.

The backing bean defines properties and handling-logics associated with the UI components used on the page. Each backing-bean property is bound to either a component instance or its value. A backing bean also defines a set of methods that perform functions for the component, such as validating the component's data, handling events that the component fires and performing processing associated with navigation when the component activates.

**10. What are the differences between a Backing Bean and Managed Bean?**

Backing Beans are merely a convention, a subtype of JSF Managed Beans which have a very particular purpose. There is nothing special in a Backing Bean that makes it different from any other managed bean apart from its usage.

What makes a Backing Bean is the relationship it has with a JSF page; it acts as a place to put component references and Event code.

|  |  |
| --- | --- |
| **Backing Beans** | **Managed Beans** |
| A backing bean is any bean that is referenced by a form. | A managed bean is a backing bean that has been registered with JSF (in faces-config.xml) and it automatically created (and optionally initialized) by JSF when it is needed. |
|  | The advantage of managed beans is that the JSF framework will automatically create these beans, optionally initialize them with parameters you specify in faces-config.xml, |
| Backing Beans should be defined only in the request scope | The managed beans that are created by JSF can be stored within the request, session, or application scopes |

Backing Beans should be defined in the request scope, exist in a one-to-one relationship with a particular page and hold all of the page specific event handling code.In a real-world scenario, several pages may need to share the same backing bean behind the scenes.A backing bean not only contains view data, but also behavior related to that data.

**11. What is view object?**

A view object is a model object used specifically in the presentation tier. It contains the data that must display in the view layer and the logic to validate user input, handle events, and interact with the business-logic tier. The backing bean is the view object in a JSF-based application. Backing bean and view object are interchangeable terms.

**12. What is domain object model?**

Domain object model is about the business object and should belong in the business-logic tier. It contains the business data and business logic associated with the specific business object.

**13. What is the difference between the domain object model and a view object?**

In a simple Web application, a domain object model can be used across all tiers, however, in a more complex Web application, a separate view object model needs to be used. Domain object model is about the business object and should belong in the business-logic tier. It contains the business data and business logic associated with the specific business object. A view object contains presentation-specific data and behavior. It contains data and logic specific to the presentation tier.

**14. What do you mean by Bean Scope?**

Bean Scope typically holds beans and other objects that need to be available in the different components of a web application.

**15. What are the different kinds of Bean Scopes in JSF?**

JSF supports three Bean Scopes. *viz.,*

* **Request Scope:** The request scope is short-lived. It starts when an HTTP request is submitted and ends when the response is sent back to the client.
* **Session Scope:** The session scope persists from the time that a session is established until session termination.
* **Application Scope:** The application scope persists for the entire duration of the web application. This scope is shared among all the requests and sessions.

**16. What is the difference between JSP-EL and JSF-EL?**

|  |  |
| --- | --- |
| **JSP-EL** | **JSF-EL** |
| In JSP-EL the value expressions are delimited by ${…}. | In JSf-EL the value expressions are delimited by #{…}. |
| The ${…} delimiter denotes the immediate evaluation of the expressions, at the time that the application server processes the page. | The #{…} delimiter denotes deferred evaluation. With deferred evaluation ,the application server retains the expression and evaluates it whenever a value is needed. |

*note:As of JSF 1.2 and JSP 2.1 ,the syntax of both expression languages has been unified.*

[More about Unified Expression Language](http://www.developersbook.com/jsf/unified-expression-language.php)

**17. What are The main 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:

* JSF Core Tags Library.
* JSF Html Tags Library.

**18. How do you declare the managed beans in the faces-config.xml file?**

The bean instance is configured in the *faces-config.xml* file:

<managed-bean>

<managed-bean-name>login</managed-bean-name>

<managed-bean-class>com.developersBookJsf.loginBean</managed-bean-class>

<managed-bean-scope>request</managed-bean-scope>

</managed-bean>

This means: Construct an object of the class *com.developersBookJsf.loginBean*, give it the name *login*, and keep it alive for the duration of the *request*.

**19. How to declare the Message Bundle in JSF?**

We can declare the message bundle in two ways:   
(Let’s assume *com.developersBookJsf.messages* is the properties file)

**1.** The simplest way is to include the following elements in *faces-config.xml file:*

<application>

<resource-bundle>

<base-name>com.developersBookJsf.messages</base-name>

<var>message</var>

</resource-bundle>

</application>

**2.** Alternatively, you can add the *f:loadBundle* element to each JSF page that needs access to the bundle:

<f:loadBundle baseName = “com.developersBookJsf.messages” var=”message”/>

|  |
| --- |
| People who read this, also read:-   * [**AJAX Interview Questions**](http://www.developersbook.com/ajax/interview-questions/ajax-interview-questions-faqs.php) * [**Hibernate Tutorial**](http://www.developersbook.com/hibernate/hibernate-tutorials/hibernate-tutorials.php) * [**Hibernate Tutorial**](http://www.developersbook.com/hibernate/hibernate-tutorials/hibernate-tutorials.php) * [**Hibernate Vs iBatis**](http://www.developersbook.com/articles/hibernate/Hiberanate-iBatis.php) * [**JSF Interview Questions**](http://www.developersbook.com/jsf/interview-questions/jsf-interview-questions-faqs.php) |

**20. How to declare the page navigation (navigation rules) in faces-config.xml file ?**

Navigation rules tells JSF implementation which page to send back to the browser after a form has been submitted. We can declare the page navigation as follows:

<naviagation-rule>

<from-view-id>/index.jsp</from-view-id>

<navigation-case>

<from-outcome>login</from-outcome>

<to-view-id>/welcome.jsp</to-view-id>

</navigation-case>

</naviagation-rule>

This declaration states that the *login* action navigates to */welcome.jsp*, if it occurred inside */index.jsp*.

**21. What if no navigation rule matches a given action?**

If no navigation rule matches a given action, then the current page is redisplayed.

**22. What are the JSF life-cycle phases?**

The six phases of the JSF application lifecycle are as follows (note the event processing at each phase):

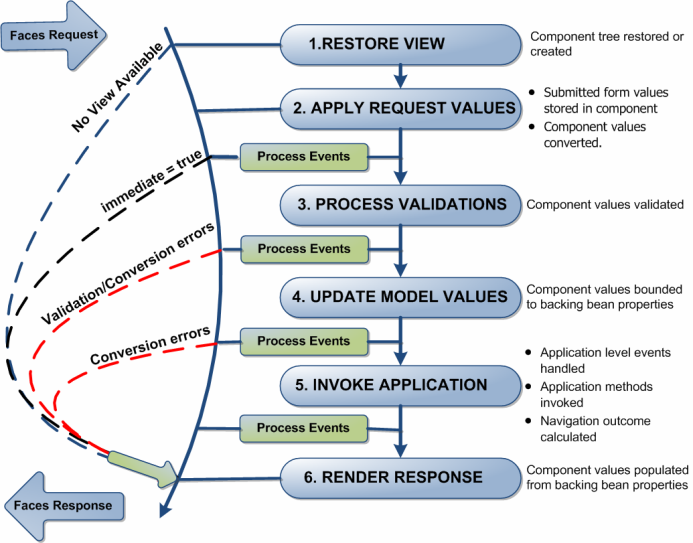
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Don't Miss...** | | [**JSF Basic Tutorial**](http://www.developersbook.com/jsf/jsf-tutorial/jsf-tutorial.php) | | [**JSF-Spring2.0 Integration**](http://www.developersbook.com/articles/jsf/spring2-jsf-integration.php) | | [**Spring FAQs**](http://www.developersbook.com/spring/interview-questions/spring-interview-questions-faqs.php) | | [**Spring Basic Tutorial**](http://www.developersbook.com/spring/spring-tutorials/spring-tutorials.php) | | [**Spring-iBatis Integration**](http://www.developersbook.com/ibatis/iBatis-tutorials/iBatis-tutorials.php) | |

**1.** Restore view   
**2.** Apply request values; process events   
**3.** Process validations; process events   
**4.** Update model values; process events  
**5.** Invoke application; process events  
**6.** Render response

**23. Explain briefly the life-cycle phases of JSF?**

**1. Restore View :** A request comes through the FacesServlet controller. The controller examines the request and extracts the view ID, which is determined by the name of the JSP page.   
**2. Apply request values:** The purpose of the apply request values phase is for each component to retrieve its current state. The components must first be retrieved or created from the FacesContext object, followed by their values.   
**3. Process validations:** In this phase, each component will have its values validated against the application's validation rules.   
**4. Update model values:** In this phase JSF updates the actual values of the server-side model ,by updating the properties of your backing beans.  
**5. Invoke application:** In this phase the JSF controller invokes the application to handle Form submissions.  
**6. Render response:** In this phase JSF displays the view with all of its components in their current state.

[**More about JSF Lifecycle**](http://www.developersbook.com/jsf/jsf-tutorial/jsf-tutorial.php#2)



|  |
| --- |
|  |

**24. What does it mean by render kit in JSF?**

A render kit defines how component classes map to component tags that are appropriate for a particular client. The JavaServer Faces implementation includes a standard HTML render kit for rendering to an HTML client.

**25. Is it possible to have more than one Faces Configuration file?**

We can have any number of config files. Just need to register in *web.xml*.   
Assume that we want to use faces-config(1,2,and 3),to register more than one faces configuration file in JSF,just declare in the *web.xml file*

<context-param>

<param-name>javax.faces.CONFIG\_FILES</param-name>

<param-value>

/WEB-INF/faces-config1.xml,

/WEB-INF/faces-config2.xml,

/WEB-INF/faces-config3.xml

</param-value>

</context-param>

1. What is JSF life cycle and its phases?  
The 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  
  
  
2. What is the role of Renderer in JSF? and justify the statement "JSF supports multiple client devices".  
  
After creating JSF components, it is also necessary for each component to be rendered to the client so that it can be visible to the client’s device. Each of the tag gives rise to an associated component. A renderer is a type of class that is responsible for encoding and decoding components. Encoding displays the component while decoding translates the user’s input into components value i.e. transform it into values the component can understand.  
Now a days there are many devices that are web enabled. So application developers have challenge to develop components that can work across various platforms. For example, if we have an application that works on standard web browser and we want to extend it to make it enable to work on a WAP device. So, to handle this case we need components to be rendered in more than one way. Here JSF can be helpful. This is a simple task for JSF. The solution is to develop separate renderers for the component. JSF components use different renderers depending on the device used.  
  
  
3. What is Render Kit in JSF?  
  
Component classes generally transfer the task of generating output to the renderer. All JSF components follow it. Render kit is a set of related renderers. javax.faces.render.RenderKit is the class which represents the render kit. The default render kit contains renderers for html but it’s up to you to make it for other markup languages. Render kit can implement a skin (a look & feel). Render kit can target a specific device like phone, PC or markup language like HTML, WML, SVG. This is one of the best benefit of JSF because JSF doesn't limit to any device or markup.  
  
  
4. What is conversion and validation? and how are they related?  
  
This is one of the phase of JSF life cycle that happens before binding the component data to the related backing bean in the Update model values phase.  
Conversion is the process of transforming the component data from String to Java objects and vice versa. For example, an user enters a value (String) in an input component and this value is to store in a Date field in the backing bean then this String value is converted to a java.util.Date value when request is sent to the server and vice versa. This process is called Conversion.  
Validation is the process of ensuring data contains the expected content. For example, checking the Date value is in MM./dd/YYYY format or any integer value is between 1 to 10.  
The main purpose of conversion and validation is to ensure the values are of correct type and following the required criteria before updating model data. So this step allows you to focus on business logic rather than working on tedious qualifications of input data such as null checks, length qualifiers, range boundaries, etc.  
  
  
5. When automatic conversion is supplied by JSF Implementation?  
  
JSF implementation automatically converts component data between presentation view and model when the bean property associated with the component is of one of the types supported by the component's data.  
For example, If a UISelectBoolean component is associated with a bean property of type Boolean, then JSF implementation will automatically convert the data from String to Boolean.  
  
  
6. What does component mean and what are its types?  
  
Components in JSF are elements like text box, button, table etc. that are used to create user interfaces of JSF Applications. These are objects that manage interaction with a user. Components help developers to create UIs by assembling a number of components , associating them with object properties and event handlers. Would u like to repeat the same code again & again and waste time if u want to create many tables in hundreds of pages in your web application? Not at all. Once you create a component, it’s simple to drop that component onto any JSP. Components in JSF are of two types :  
1. Simple Components like text box, button and  
2. Compound Components like table, data grid etc.  
  
A component containing many components inside it is called a compound component.  
  
JSF allows you to create and use components of two types:  
  
1. Standard UI Components:   
JSF contains its basic set of UI components like text box, check box, list boxe, button, label, radio button, table, panel etc. These are called standard components.  
2. Custom UI Components:  
Generally UI designers need some different , stylish components like fancy calendar, tabbed panes . These types of components are not standard JSF components. JSF provides this additional facility to let you create and use your own set of reusable components. These components are called custom components.  
  
7. What are third party components and how they are useful?  
One of the greatest power of JSF is to support third party components . Third party components are custom components created by another vendor. Several components are available in the market, some of them are commercial and some are open source . These pre-built & enhanced components can be used in UI of your web application. For example, we are in need of a stylish calendar then we have an option to take it from third party rather than creating it our own. This will help saving time & cost creating effective & robust UI and to concentrate on business logic part of web application.  
  
  
8. 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.

**Code:** [[Select]](javascript:void(0);)

<%@ 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)  
  
9. What is the difference between JSP and JSF?  
  
JSP simply provides a Page which may contain markup, embedded Java code, and tags which encapsulate more complicated logic / html.  
JSF may use JSP as its template, but provides much more. This includes validation, rich component model and lifecycle, more sophisticated EL, separation of data, navigation handling, different view technologies (instead of JSP), ability to provide more advanced features such as AJAX, etc.