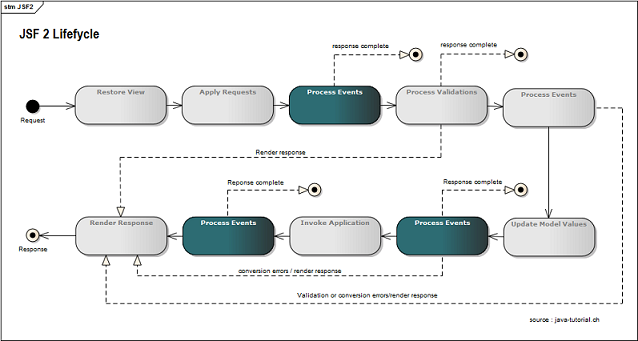
[JSF [http://www.java-tutorial.ch/administrator/components/com_acesef/assets/images/icon-10-external.png](http://java-tutorial.ch/java-server-faces)](http://java-tutorial.ch/java-server-faces" \o "JSF) 2 Lifecycle diagram :



Restore view

RestoreView is the first phase in the [**JSF [http://www.java-tutorial.ch/administrator/components/com_acesef/assets/images/icon-10-external.png](http://java-tutorial.ch/java-server-faces)**](http://java-tutorial.ch/java-server-faces)**lifecycle**. This phase is used for constructing view to display in the front end. Every view has it's own view id and it is stored in the FacesContext's session object. [JSF [http://www.java-tutorial.ch/administrator/components/com_acesef/assets/images/icon-10-external.png](http://java-tutorial.ch/java-server-faces)](http://java-tutorial.ch/java-server-faces) View is collection of components associated with its current state. There is two types of state saving mechanism,

1. Server (default)
2. Client

Default value is server. If you specify state saving method as server, the state of each components will be stored in the server. If it is client, it will be stored in the client side as hidden variables. This value is configured using **[javax.faces.STATE\_SAVING\_METHOD](http://java-tutorial.ch/java-server-faces/jsf-2-configuration-parameters" \l "STATE_SAVING_METHOD" \o "jsf state saving method )** parameter name in your web.xml context params as follows:

jsf restore view state saving configuration in web.xml

1.<context-param>

2.<param-name>javax.faces.STATE\_SAVING\_METHOD</param-name>

3.<param-value>client</param-value>

4.</context-param>

Apply Requests

After the component tree is restored, each component in the tree extracts its new value from the request parameters by using its decode method. The value is then stored locally on the component. If the conversion of the value fails, an error message associated with the component is generated and queued on FacesContext. This message will be displayed during the [**render response phase**](http://java-tutorial.ch/java-server-faces/jsf-2-lifecycle#render_response), along with any validation errors resulting from the process validations phase.

Process Validations

During this phase, the JavaServer Faces implementation processes all validators registered on the components in the tree. It examines the component attributes that specify the rules for the validation and compares these rules to the local value stored for the component.

Update Model Values

After the JavaServer Faces implementation determines that the data is valid, it can walk the component tree and set the corresponding server-side object properties to the components' local values. The JavaServer Faces implementation will update only the bean properties pointed at by an input component's value attribute.

If the local data cannot be converted to the types specified by the bean properties, the **life cycle**advances directly to the [**render response**](http://java-tutorial.ch/java-server-faces/jsf-2-lifecycle#render_response) phase so that the page is rerendered with errors displayed. This is similar to what happens with validation errors.

Invoke Applications

During this phase, the JavaServer Faces implementation handles any application-level events, such as submitting a form or linking to another page.

At this point, if the application needs to redirect to a different web application resource or generate a response that does not contain any JavaServer Faces components, it can call FacesContext.responseComplete.

If the view being processed was reconstructed from state information from a previous request and if a component has fired an event, these events are broadcast to interested listeners.

Render Response

During this phase, the JavaServer Faces implementation delegates authority for rendering the page to the JSP container if the application is using JSP pages. If this is an initial request, the components represented on the page will be added to the component tree as the JSP container executes the page. If this is not an initial request, the components are already added to the tree so they needn't be added again. In either case, the components will render themselves as the JSP container traverses the tags in the page.

Conclusion

If you have any remark or questions feel free to put a comment.If you enjoyed this tutorial and want to promote it don't hesitate to click on

|  |  |  |  |
| --- | --- | --- | --- |
| 1down vote[favorite](http://stackoverflow.com/questions/4323730/jsf-another-question-on-lifecycle)  **2** | Today I'd like to know some features on the JSF Lifecycle. Let me start :  **1 - Phase 2:Apply request Values - During this phase,each component in the view will search for its values in the request and set the new values to them**  Uhm, ok nice. So, the View will be built due to the previous Beans parameters. After, there is a partial View, generated with the request values. (Right? Later, in the 3° phase, they will be compared) . But, for example, if a values in the request list is absent during the creation of this last view? Values will be null?  **2 - Phase 5: Invoke Application - Once all the values of the request has been successfully set to the backing bean the action events queued during the apply request values phase will be processed. In our case the submit buttons action method .**  This is not clear at all. At this moment i have (on the beans) the values updated from the previous Phase (If the validation and the apply request aren't failed). Ok, so now what happens? What means the action events queued during the apply request values phase will be processed? It means that, for example, if the action is Submit the process is finished? That's why an ajax call, if not rendered in the 2° phase, will fail? Or where it fails?  **3 - Phase 6: Render response - In this phase the component tree will be rendered to the client.**  It means that the View on the server is updated by using the updated bean values? And, after this, the HTML code is created from this View? Or just it made the HTML code and save the View status?  Hope you can help me :)  [jsf](http://stackoverflow.com/questions/tagged/jsf) [jsf-2](http://stackoverflow.com/questions/tagged/jsf-2) [facelets](http://stackoverflow.com/questions/tagged/facelets" \o "show questions tagged 'facelets') [page-lifecycle](http://stackoverflow.com/questions/tagged/page-lifecycle) [javabeans](http://stackoverflow.com/questions/tagged/javabeans)   |  |  | | --- | --- | | [share](http://stackoverflow.com/q/4323730)|[improve this question](http://stackoverflow.com/posts/4323730/edit) | asked Dec 1 '10 at 11:12  [[https://www.gravatar.com/avatar/586ed1e5c3543cf7c304861c1240efdf?s=32&d=identicon&r=PG](http://stackoverflow.com/users/365251/markzzz)](http://stackoverflow.com/users/365251/markzzz)  [markzzz](http://stackoverflow.com/users/365251/markzzz) **8,409**29130251 | |
|  | add a comment |

## 2 Answers

[active](http://stackoverflow.com/questions/4323730/jsf-another-question-on-lifecycle?answertab=active#tab-top)[oldest](http://stackoverflow.com/questions/4323730/jsf-another-question-on-lifecycle?answertab=oldest#tab-top)[votes](http://stackoverflow.com/questions/4323730/jsf-another-question-on-lifecycle?answertab=votes#tab-top)

|  |  |
| --- | --- |
| up vote4down voteaccepted | **Phase 2:Apply request Values - During this phase,each component in the view will search for its values in the request and set the new values to them**  Uhm, ok nice. So, the View will be built due to the previous Beans parameters. After, there is a partial View, generated with the request values. (Right? Later, in the 3° phase, they will be compared) . But, for example, if a values in the request list is absent during the creation of this last view? Values will be null?  Basically the following is happening under the covers (here, input is [UIInput](http://download.oracle.com/javaee/6/api/javax/faces/component/UIInput.html) and request is [HttpServletRequest](http://download.oracle.com/javaee/6/api/javax/servlet/http/HttpServletRequest.html)):  if (input.isRendered()) {  String value = request.getParameter(input.getClientId());  if (value != null) {  input.setSubmittedValue(value);  }  }  So, they will be untouched if there's no request parameter. They won't be set with null and just kept default.  **Phase 5: Invoke Application - Once all the values of the request has been successfully set to the backing bean the action events queued during the apply request values phase will be processed. In our case the submit buttons action method .**  This is not clear at all. At this moment i have (on the beans) the values updated from the previous Phase (If the validation and the apply request aren't failed). Ok, so now what happens? What means the action events queued during the apply request values phase will be processed? It means that, for example, if the action is Submit the process is finished? That's why an ajax call, if not rendered in the 2° phase, will fail? Or where it fails?  During 2nd phase basically the following will also happen (here, command is [UICommand](http://download.oracle.com/javaee/6/api/javax/faces/component/UICommand.html), request isHttpServletRequest and ActionEvent is [ActionEvent](http://download.oracle.com/javaee/6/api/javax/faces/event/ActionEvent.html)):  if (command.isRendered()) {  String value = request.getParameter(command.getClientId());  if (value != null) {  command.queueEvent(new ActionEvent(command)); // Queue for INVOKE\_ACTION.  }  }  Then, during invoke application phase, all events which are queued for the particular phase will be invoked.  **Phase 6: Render response - In this phase the component tree will be rendered to the client.**  It means that the View on the server is updated by using the updated bean values? And, after this, the HTML code is created from this View? Or just it made the HTML code and save the View status?  During this phase JSF walks through the component tree and all components will be encoded (will invoke the [Renderer](http://download.oracle.com/javaee/6/api/javax/faces/render/Renderer.html) of all components, by default a HTML renderer). During encoding, the values will just be obtained from the model. The view itself won't be updated. Basically:  facesContext.getViewRoot().encodeAll(); |

JSF 2 experience Interview question

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **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.  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | |  | | --- | | **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) | |  * 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?**   |  | | --- | | People who read this, also read:-  * [**Spring Interview Questions**](http://www.developersbook.com/spring/interview-questions/spring-interview-questions-faqs.php) * [**XML Questions**](http://www.developersbook.com/xml/interview-questions/xml-interview-questions-faqs.php) * [**SCMAD Certification**](http://www.developersbook.com/certifications/scmad/scmad-certification.php) * [**Let Spring Manage JSF Beans**](http://www.developersbook.com/articles/jsf/spring2-jsf-integration.php) * [**Core Java Questions**](http://www.developersbook.com/corejava/interview-questions/corejava-interview-questions-faqs.php) |   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 bean | 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.   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | |  | | --- | | **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) | |   **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:-  * [**XML Interview Questions**](http://www.developersbook.com/xml/interview-questions/xml-interview-questions-faqs-1.php) * [**Servlets Questions**](http://www.developersbook.com/servlets/interview-questions/servlets-interview-questions-faqs.php) * [**Hibernate Tutorial**](http://www.developersbook.com/hibernate/hibernate-tutorials/hibernate-tutorials.php) * [**Integrating Hibernate with Spring**](http://www.developersbook.com/articles/spring-hibernate-integration.php) * [**Servlets Interview Questions**](http://www.developersbook.com/servlets/interview-questions/servlets-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)  JSF Lifecycle     |  | | --- | |  |   **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> What is the JSF architecture? When it comes to Java programming, it is important to note that different sublanguages such as JDBC and JSF have different architectures. Knowing the architecture for the language you are writing in is a pretty basic set of facts that you need to know. You may need to take a **[Javascript for beginners](https://www.udemy.com/beginning-javascript/?tc=blog.jsfinterview.text.p&utm_source=blog&utm_medium=udemyads&utm_content=post13886&utm_campaign=content-marketing-blog&xref=blog)**course to get yourself up to speed. JSF has been built on the Model View Controller (MVC) framework and this allows for applications to be scaled better and has better maintainability. The JSF works with a framework that works well with Java Web and user interfaces that fit well within the MVC. How is JSF different from the conventional JSP / Servlet Model? JSF is basically a much more advanced language than the regular old JSP servlets. Developers need to put together the JSP and servlets by hand whereas JSF is much more high level. You can even think of JSP and servlets as being parts of the engine that help JSF run smoothly rather than features that work side by side. What are the available implementations of JavaServer Faces? There are a couple of different implementations when it comes to JSF. There is Reference Implementation (RI) by Sun Microsystems; Apache MyFacesis an open source JavaServer Faces (JSF) implementation or run-time; ADF Facesis Oracle’s implementation for the JSF standard. All three of these will come in handy in the real world once you have fully come to understand what JSF is and what Java truly offers. What does a typical JSF application consist of? Applications that are built with JSF programming language usually have three specific features. A typical JSF application consists of the following parts: JSF has JavaBeans components for managing the application state and its behavior. Event-driven development is another part of the typical application. The third and final aspect of these applications are pages that represent MVC-style views; pages reference view roots via the JSF component tree.  Have you had some other JSF interview questions that you needed to answer in order to be successful? Share them in the comments below. |  |

|  |  |
| --- | --- |
| 12down voteaccepted | [Why HibernateTemplate isn't recommended? [duplicate]](http://stackoverflow.com/questions/18002768/why-hibernatetemplate-isnt-recommended) Because its main goal was to get a Hibernate session tied to the current Spring transaction, when SessionFactory.getCurrentSession() didn't exist. Since it now exists (and for a long time: HibenateTemplate usage is discouraged even in the hibernate3 package), there is no reason to use this Spring-specific class instead of using SessionFactory.getCurrentSession() to get a session tied to the current Spring transaction.  If you use Spring, then you should use its declarative transaction management, which allows you to avoid opening, committing, closing and flushing. It's all done by Spring automatically:  @Autowired  private SessionFactory sessionFactory;  @Transactional  public void someMethod() {  // get the session for the current transaction:  Session session = sessionFactory.getCurrentSession();  // do things with the session (queries, merges, persists, etc.)  }  In the above example, a transaction will be started (if not already started) before the method invocation; A session will be created by Spring for the transaction, and the session will be automatically flushed before the commit of the transaction, that will be done by Spring automatically when the method returns. |