Register | Login | Logout | Contact Us

Java-Success.com

Industrial strength Java Career Companion

search here … Go

Home | Java FAQs | 600+ Java Q&As | Career | Tutorials | Member | Why?

Can u Debug? | Java 8 ready? | Top X | Productivity Tools | Judging Experience?

Home › Interview › Spring, Hibernate, & Maven Interview Q&A › JSF › More JSF interview Q&A

More JSF interview Q&A

Posted on September 8, 2014 by Arulkumaran Kumaraswamipillai — No Comments ↓

Q1. What is the difference between initial request and postback?

A1. **Initial request** (e.g. HTTP GET) is the request that is made from a browser in order to display a page. **Postback** happens when the browser posts the page back to the server with form values, etc. Initial request is created by clicking a link, pasting an URL in address bar, while a postback request is created by posting a form by **clicking a submit button** or any post request. Initial request passes only restore View & Render Response phases, while postback request process under all phases described in the JSF life cycle diagram.

During the **restore view phase** of the life cycle, ViewHandler retrieves the ResponseStateManager object in order to test if the request is a postback or an initial request. If a request is a postback, the restoreView method of ViewHandler is called. This method uses the ResponseStateManager object to re-build the component tree and restore state. The ResponseStateManager object is the only one that knows what rendering technology is being used and is therefore the

600+ Full Stack Java/JEE Interview Q&As ♥Free ♦FAQs

open all | close all

⊞ Ice Breaker Interview
⊞ Core Java Interview (
⊞ JEE Interview Q&A (3
⊞ Pressed for time? Jav
⊞ SQL, XML, UML, JSC
⊞ Hadoop & BigData Int
⊞ Java Architecture Inte
⊞ Scala Interview Q&As
⊟ Spring, Hibernate, & I
  ⊞ Spring (18)
  ⊞ Hibernate (13)
  ⊞ AngularJS (2)
  ⊞ Git & SVN (6)
  ⊞ JMeter (2)
  ⊟ JSF (2)
    JSF interview Q&
    More JSF intervi
  ⊞ Maven (3)
⊞ Testing & Profiling/Sa
⊞ Other Interview Q&A 1
⊞ ▶ Free Java Interviev

only one that can look at a request, which is rendering-technology specifiec. Here are the basic steps.

– An isPostBack method on ResponseStateManager returns true if the current request is a postback.

– A getState method is called by the restoreView method of ViewHandler to retrieve the component tree state from the current request.

– A writeState method that writes out the state to the client. This method is called by the renderView method of ViewHandler during the render response phase.

Q2. What is the purpose of the attribute immediate=true
A2. The immediate attribute is used in UIInput and UICommand components for the following purposes :

#1. Immediate attribute, when set to true, allows a UICommand component like commandLink or commandButton to process the back-end logic and ignore validation process related to the fields on the page. This allows navigation to occur even when there are validation errors.

In the code below, navigation is performed for button click without validating the required field.

```
<h:inputText id="input1" required="true"/>
<h:message for="input1"/>
<h:commandButton value="submit" immediate="true"a

```

#2. To make one or more UIInput components "high priority" for validation, so validation is performed, if there is any invalid component data, only for high priority input components and not for low priority input components in the page. This helps reducing the number of error messages shown on the page.

16 Technical Key Areas

open all | close all

⊞ Best Practice (6)
⊞ Coding (26)
⊞ Concurrency (6)
⊞ Design Concepts (7)
⊞ Design Patterns (11)
⊞ Exception Handling (3
⊞ Java Debugging (21)
⊞ Judging Experience I
⊞ Low Latency (7)
⊞ Memory Managemen
⊞ Performance (13)
⊞ QoS (8)
⊞ Scalability (4)
⊞ SDLC (6)
⊞ Security (13)
⊞ Transaction Managen

80+ step by step Java Tutorials

open all | close all

⊞ Setting up Tutorial (6)
⊞ Tutorial - Diagnosis (2
⊞ Akka Tutorial (9)
⊞ Core Java Tutorials (2
⊞ Hadoop & Spark Tuto
⊞ JEE Tutorials (19)
⊞ Scala Tutorials (1)
⊞ Spring & HIbernate Tu
⊞ Tools Tutorials (19)
⊞ Other Tutorials (45)

In the code below, validation is performed only for the first component when button is clicked because the immediate=true.

```
<h:inputText id="input1" immediate="true" require
<h:inputText id="input2" required="true"/>
<h:message for="input1"/>
<h:message for="input2"/>
<h:commandButton value="submit" action="send"/>

```

Q3. What are the different types of JSF events?
A3. JSF is an event driven framework.

– **Action Events:** bound to UI Command objects like a Command Button or a Hyper-link. Whenever a user presses a Command Button or clicks a hyperlink these Events get generated.

– **Value Change Events:** bound to UI Components like Text Field, Check-Box, List and Radio Buttons. The Value Change Event is fired as soon as the value that is displayed in the view is modified.

– **Phase Events:** As you saw earlier in the JSF overview blog, the request processing life-cycle in JSF includes six phases and any JSF implementation will fire Phase events during the start and end of each phase. If we want to capture the Phase Events, then can define a Phase Listener. These are handy for debugging as well.

Q4. How are events handled in JSF? What is the difference between these event handling mechanisms?
A4. **Action handlers** and **event listeners** provide an event driven mechanism. Every time a user does something like clicking a button, selecting an item from a drop down, or submitting a form, an event occurs. Event notification is then sent via HTTP to the server and handled by the FacesServlet. Events can invoke custom business logic or initiate page navigation.

100+ Java pre-interview coding tests

open all | close all
⊞ Can you write code? (
⊞ ♦ Complete the given
⊞ Converting from A to I
⊞ Designing your classe
⊞ Java Data Structures
⊞ Passing the unit tests
⊞ What is wrong with th
⊞ Writing Code Home A
⊞ Written Test Core Jav
⊞ Written Test JEE (1)

How good are your .....?

open all | close all
⊞ Career Making Know-
⊞ Job Hunting & Resum

JSF provides two types of methods for handling events; **listeners** and **action handlers**, both of these may be defined within a managed bean. A listener takes an FacesEvent as a parameter and a void return type, while an action handler takes no parameters and returns a String.

Example 1: An event handler

```
<!-- login.xhtml-->
<h:inputText id="useName" />
<h:inputSecret id="password" />
<h:commandButton action="#{login.submit}" />

```

```
<!-- faces-config.xml-->
<navigation-rule>
     <from-view-id>/login.xhtml</from-view-id>
  <navigation-case>
     <from-action>#{login.submit}</from-action>
  <from-outcome>success</from-outcome>
  <to-view-id></to-view-id>
  </navigation-case>
</navigation-rule>

```

```
//Login managed bean
public String submit(  ) {
      //do some action
       ....

       return "success"  //navigate to /home.jsf
}
```

Example 2: An event handler with an event listener.

Action listeners are provided by JSF to make it easier to handle action events. An advantage of using a listener is that the FacesEvent object provides additional information, such as the form element that initiated the event. An action handler in contrast has no knowledge of the source of the event, but based upon its return value, can initiate page navigation. The example below shows using both event handlers and event listeners.

```
<h:commandButton value="Search" actionListener="#
     action="#{orders.search}" />
```

In the above example, when the button is clicked the JSF implementation calls the action listener during the Invoke Application phase. The action listener method then has a chance to perform any processing related to the command element selected by the user. You can perform any processing you need to inside the method. The method can have any name, must be public, return void, and accept an ActionEvent as its only parameter.

```
public void confirm(ActionEvent event) {
    int calculatedAge = calculateAgeFromDOB();
    if (event.getComponent().getId().equals("con
            //perform some action
              ....
       }

    else if(event.getComponent().getId().equals(
          //perform some other action
                ....
    }
}
```

After the action listener method is called, the method bound by the action attribute will be called, and the JSF implementation will determine where to navigate next.

```
public String search( ) {
   //some action logic
   ...
   //navigation logic
   return "success";
}
```

Since the action listener method is called before the action handler method, the action listener method can modify the response that the action method returns. An action listener can also be implemented as shown below.

```
<h:commandButton id="submitOrderSearch" value="Se
     <f:actionListener type="com.MyAppActionListe
</h:commandButton>
```

The listener class can be implemented as shown below:

```
public class MyAppActionListenerImpl implements A

    public void processAction(ActionEvent aev) {
      System.out.println(aev.getId());
   ...
   }
}
```

Q5. What is a value change event, and when will it be useful?

A5. A **ValueChangeEvent** is useful whenever you want to be notified when there is a change in the value of a component, such as text modification in a text field or a check box selection. Most JSF components support the valueChangeListener attribute.

```
<h:inputText  id="orderStatus"  valueChangeListen
```

The managed bean method:

```
public void onStatusChange(ValueChangeEvent vce)
  System.out.println(vce.getId());
  System.out.println(vce.getOldValue());
  System.out.println(vce.getNewValue());
  ....
}
```

The listener class

```
public class MyAppValueChangeActionListenerImpl

    public void processValueChange(ValueChangeEv
      System.out.println(vce.getId());
      System.out.println(vce.getOldValue());
      System.out.println(vce.getNewValue());
      ....
   }
}

```

Popular Member Posts

- ♦ 11 Spring boot interview questions & answers

904 views

- ♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

816 views

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

427 views

18 Java scenarios based interview Questions and Answers

408 views

♦ 7 Java debugging interview questions & answers

324 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

311 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

304 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

301 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

251 views

♦ Object equals Vs == and pass by reference Vs value

234 views

Bio | **Latest Posts**

Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

About Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

‹ JSF interview Q&A

6 Scaling your Java applications interview Q&As ›

Posted in JSF**,** member-paid

Tags: Java/JEE FAQs

Leave a Reply

Logged in as geethika. Log out?

Comment

Post Comment

Empowers you to open more doors, and fast-track

Technical Know Hows

☀ Java generics in no time ☀ Top 6 tips to transforming your thinking from OOP to FP ☀ How does a HashMap internally work? What is a hashing function?
☀ 10+ Java String class interview Q&As ☀ Java auto un/boxing benefits & caveats ☀ Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect

Non-Technical Know Hows

☀ 6 Aspects that can motivate you to fast-track your career & go places ☀ Are you reinventing yourself as a Java developer? ☀ 8 tips to safeguard your Java career against offshoring ☀ My top 5 career mistakes

Prepare to succeed

☀ Turn readers of your Java CV go from "Blah blah" to "Wow"? ☀ How to prepare for Java job interviews? ☀ 16 Technical Key Areas ☀ How to choose from multiple Java job offers?

Select Category ▼

© Disclaimer

The contents in this Java-Success are copy righted. The author has the right to correct or enhance the current content without any prior notice.

These are general advice only, and one needs to take his/her own circumstances into consideration. The author will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. No guarantees are made regarding the accuracy or usefulness of content, though I do make an effort to be accurate. Links to external sites do not imply endorsement of the linked-to sites.