How To... Handle Value Change Events

Applicable Releases:
SAP NetWeaver Composition Environment 7.1

Topic Area:
User Productivity
Development and Composition

Capability:
User Interface Technology
Java

Version 1.0
October 2008
No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, Informix, i5/OS, POWER, POWER5, OpenPower and PowerPC are trademarks or registered trademarks of IBM Corporation.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

MaxDB is a trademark of MySQL AB, Sweden.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies (“SAP Group”) for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

These materials are provided “as is” without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.

SAP shall not be liable for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials.

SAP does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third party web pages nor provide any warranty whatsoever relating to third party web pages.

SAP NetWeaver “How-to” Guides are intended to simplify the product implementation. While specific product features and procedures typically are explained in a practical business context, it is not implied that those features and procedures are the only approach in solving a specific business problem using SAP NetWeaver. Should you wish to receive additional information, clarification or support, please refer to SAP Consulting.

Any software coding and/or code lines / strings (“Code”) included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.

Disclaimer

Some components of this product are based on Java™. Any code change in these components may cause unpredictable and severe malfunctions and is therefore expressively prohibited, as is any decompilation of these components. Any Java™ Source Code delivered with this product is only to be used by SAP’s Support Services and may not be modified or altered in any way.
<table>
<thead>
<tr>
<th>Document Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>First official release of this guide</td>
</tr>
<tr>
<td>Typographic Conventions</td>
<td>Icons</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------</td>
</tr>
<tr>
<td><strong>Type Style</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><em>Example Text</em></td>
<td>Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation</td>
</tr>
<tr>
<td><em>Example text</em></td>
<td>Emphasized words or phrases in body text, graphic titles, and table titles</td>
</tr>
<tr>
<td><em>Example text</em></td>
<td>File and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools</td>
</tr>
<tr>
<td><em>Example text</em></td>
<td>User entry texts. These are words or characters that you enter in the system exactly as they appear in the documentation</td>
</tr>
<tr>
<td>&lt;<em>Example text</em>&gt;</td>
<td>Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system</td>
</tr>
<tr>
<td>EXAMPLE TEXT</td>
<td>Keys on the keyboard, for example, F2 or ENTER</td>
</tr>
<tr>
<td><strong>Icon</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><img src="image" alt="Caution" /></td>
<td>Caution</td>
</tr>
<tr>
<td><img src="image" alt="Note or Important" /></td>
<td>Note or Important</td>
</tr>
<tr>
<td><img src="image" alt="Example" /></td>
<td>Example</td>
</tr>
<tr>
<td><img src="image" alt="Recommendation or Tip" /></td>
<td>Recommendation or Tip</td>
</tr>
</tbody>
</table>
Table of Contents

1. Business Scenario............................................................................................................... 1
2. Background Information..................................................................................................... 1
3. Prerequisites ...................................................................................................................... 2
4. Step-by-Step Procedure.................................................................................................... 3
   4.1 Tutorial Setup ............................................................................................................... 3
   4.2 Create the Java Classes .............................................................................................. 3
   4.3 Create the JSF Page .................................................................................................... 6
   4.4 Implement immediate event handling......................................................................... 13
   4.5 Build, Deploy and Run your application ..................................................................... 15
1. Business Scenario

Web applications may need to respond to changes in the user interface, such as selecting items from a list, clicking a button or changing an input field. These changes are called events and they take place in a client (e.g., a browser), but the changes are evaluated by the request processing lifecycle and are handled in different phases.

The following guide will explain when these events are processed by the JSF request processing lifecycle and will give you a simple example on how to respond to value change events.

2. Background Information

The JSF event model supports three kinds of events:

- **Value change events**: When the event is fired by input components (e.g. InputText, SelectOneRadio, SelectOneMenu)
- **Action events**: When the event is fired by command components (e.g. CommandButton, CommandLink)
- **Phase events**: These events are fired by the JSF life cycle before and after each request processing lifecycle phase.

These events are processed by the JSF request processing lifecycle in different phases. The following image summarizes the different types of events and when they are processed in the request processing lifecycle. (Source: Schalk, Chris, Ed Burns and James Holmes. *JavaServer Faces: The Complete Reference* October 2008)
3. Prerequisites

The following is a list of all you need for developing JSF applications and invoking Enterprise Services.

- AS Java 7.1 (CE 7.1 or NW 7.1)
- NWDS 7.1 (SP3 or higher with latest patch level).

**Note**

While this tutorial is geared towards the SAP AS Java (the build/deploy steps of the guide), it wouldn’t be hard to replace the build/deploy portions with similar steps for any other Java EE 5 platform.

Knowledge

- You have a basic knowledge of Java Enterprise Edition
- You have acquired some basic experience with JSF applications, for example by working through the JSF tutorials (Create a Hello World Application using JavaServer Faces [Extern] and Create Your First JSF Application [Extern])
4. **Step-by-Step Procedure**

In the following sections, you will create a Web Module Development component and an Enterprise Application needed to deploy the web module.

This Web application will consist of one view that has a category and a subcategory menu. When a category is selected by the end user, the *index.jsp* page uses JavaScript to post the form back so the event will be handled by a `valueChangeListener` method (an event handler method), so the corresponding list of subcategories will be displayed.

This guide will also give you an example of immediate event handling, since some of the UI elements are not validated before the event handler is invoked. Instead, the event handler for the category list populates the subcategories and forces JSF to skip to the Render Response phase.

4.1 **Tutorial Setup**

1. Create a Web Module Development Component named `veventjsf/web`.
2. Create an Enterprise Application Development Component named `veventjsf/ear`.

4.2 **Create the Java Classes**

For simplicity, a final Java Class will be created with the categories and subcategories needed for this example, but in a real use case, the book's categories and subcategories should be stored in a backend system.

1. From the context menu of the *Java Resources: source* folder in the *Web Module* project create a Final Java class. Enter *Categories* in the *Name* field, `com.sap.tutorial.jsf.event.util` in the *Package* field and declare the following constants:

   ```java
   public static final String CHILDREN = "Children's book";
   public static final String COMPUTER = "Computer & Internet";
   public static final String NETWORK = "Networking";
   public static final String OS = "Operating Systems";
   public static final String PROG = "Programming";
   public static final String[] CATEGORY_NAMES = { CHILDREN, COMPUTER };
   public static final String[] CHILD_SUBCATEG = { BABY, AGE8, AGE12 };
   public static final String[] COMPUTER_SUBCATEG = { NETWORK, OS, PROG };
   ```
2. From the context menu of the `com.sap.tutorial.jsf.event.util` package in the Web Module project create another Java class. Enter `BookForm` in the Name field, declare the following attributes:

```java
private String title;
private String author;
private String category;
private String subcategory;
```

3. Additionally declare the following collections needed to fill the category and subcategory menu (`selectOneMenu` UI elements) that will display the list of categories and subcategories in the JSF view:

```java
private static ArrayList<SelectItem> categories = null;
private ArrayList<SelectItem> subcategories = null;
private static ArrayList<SelectItem> childrenBooks = null;
private static ArrayList<SelectItem> computerBooks = null;
```

4. Generate the constructor to initialize all the collections with the following code:

```java
public BookForm() {
    super();
    if (categories == null) {
        categories = new ArrayList<SelectItem>();
        for (int i = 0; i < Categories.CATEGORY_NAMES.length; i++) {  
            categories.add(new SelectItem(Categories.CATEGORY_NAMES[i]));
        }
    }
    if (childrenBooks == null) {
        childrenBooks = new ArrayList<SelectItem>();
        for (int i = 0; i < Categories.CHILD_SUBCATEG.length; i++) {
            childrenBooks.add(new SelectItem(Categories.CHILD_SUBCATEG[i]));
        }
    }
    if (computerBooks == null) {
        computerBooks = new ArrayList<SelectItem>();
        for (int i = 0; i < Categories.COMPUTER_SUBCATEG.length; i++) {
            computerBooks.add(new SelectItem(Categories.COMPUTER_SUBCATEG[i]));
        }
    }
}
```
5. Generate getters and setters for the java class attributes

```java
public String getTitle() {
    return title;
}

public void setTitle(String title) {
    this.title = title;
}

public String getAuthor() {
    return author;
}

public void setAuthor(String author) {
    this.author = author;
}

public String getCategory() {
    return category;
}

public void setCategory(String category) {
    this.category = category;
}

public String getSubcategory() {
    return subcategory;
}

public void setSubcategory(String subcategory) {
    this.subcategory = subcategory;
}

public Collection getCategories() {
    return categories;
}

public ArrayList<SelectItem> getSubcategories() {
    return subcategories;
}
```
6. Create a `categoryChanged` method that will handle the value change event in the Category menu by adding the following code to the `BookForm` java class

```java
public void categoryChanged(ValueChangeEvent event) {
    String value = (String) event.getNewValue();
    if (Categories.COMPUTER.equals(value))
        subcategories = computerBooks;
    else
        subcategories = childrenBooks;
}
```

Important  
Like all value change listeners, this method is passed a value change event. The `valueChangeListener` method uses this event to access the UI element’s new value and process it as needed. The new value is obtained through the `event.getNewValue()` method.

7. Save the changes you made

8. Configure the `BookForm` Java class in the application configuration resource file `faces-config.xml` using the managed-bean XML element. Enter `form` in the `Name` field to reference the `BookForm` java class and select `session` in the `Scope` field. The following XML code will be added in the Source tab

```xml
<managed-bean>
    <managed-bean-name>form</managed-bean-name>
    <managed-bean-class>
        com.sap.tutorial.jsf.event.util.BookForm
    </managed-bean-class>
</managed-bean>
```

4.3 Create the JSF Page

1. In the `com.sap.tutorial.jsf.es.util` package, create the `ResourceBundle` entering `messages.properties` in the `File Name`

2. Enter the following keys and values for the English version of the localized messages

```message
pageTitle=Book Search
titlePrompt=Title
authorPrompt=Author
categoryPrompt=Category
subcategoryPrompt=Subcategory
```
submitButton=Submit

3. For simplicity, only the English version of the localized message is created. Optionally you can create other versions of the localized messages and specify which languages are supported for this application as indicated in the Product Offer tutorial Part 3 (International JSF application [extern]).

4. Expose the ResourceBundles by adding the following XML code in the Source tab of the faces-config.xml file

```xml
<application>
   <resource-bundle>
      <base-name> com.sap.tutorial.jsf.event.util.messages</base-name>
      <var>msgs</var>
   </resource-bundle>
</application>
```

5. Create a Style file and define the following CSS classes

```css
.title {
   font-family: Verdana, Arial, Sans-Serif;
   font-weight: bold;
   font-size: 12px;
   color: #0000A0;
   font-style: normal;
}

.label {
   font-family: Verdana, Arial, Sans-Serif;
   font-weight: bold;
   font-size: 12px;
   color: #606060;
   font-style: normal;
}

.errorMessage {
   font-family: Verdana, Arial, Sans-Serif;
   font-size: 10px;
   color: red;
   font-style: normal;
}
```

6. Drill into the Web Module project and right click on the WebContent folder and in the context menu select New → JSP.
7. Enter the file name `index.jsp` and click the `Finish` button. The JSP page will be created. The `index.jsp` page should be opened in the Web Page Editor.

8. Include the style sheet by adding a `link` element inside the `head` element as shown in the following code:

```
<head>
  <link href="styles.css" rel="stylesheet" type="text/css"/>
  ...
</head>
```

9. The following table contains the hierarchy of the UI elements contained in the `index` view:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ViewRoot</code> UI element</td>
<td></td>
</tr>
<tr>
<td><code>Form</code> UI element in the UI-element <code>ViewRoot</code></td>
<td></td>
</tr>
<tr>
<td><code>OutputText</code> UI element in the UI-element <code>Form</code></td>
<td>value: <code>${msgs.pageTitle}</code>; styleClass: <code>title</code></td>
</tr>
<tr>
<td><code>PanelGrid</code> UI element in the UI-element <code>Form</code></td>
<td>Border: 0; Columns: 3</td>
</tr>
<tr>
<td><code>OutputText</code> UI element in the UI-element <code>PanelGrid</code></td>
<td>value: <code>${msgs.titlePrompt}</code>; styleClass: <code>label</code></td>
</tr>
<tr>
<td><code>InputText</code> UI element in the UI-element <code>PanelGrid</code></td>
<td>id: <code>title</code>; value: <code>${form.title}</code>; required: True; label: <code>${msgs.titlePrompt}</code></td>
</tr>
<tr>
<td><code>Message</code> UI element in the UI-element <code>PanelGrid</code></td>
<td>for: <code>title</code>; errorClass: <code>errorMessage</code></td>
</tr>
<tr>
<td><code>OutputText</code> UI element in the UI-element <code>PanelGrid</code></td>
<td>value: <code>${msgs.authorPrompt}</code>; styleClass: <code>label</code></td>
</tr>
<tr>
<td><code>InputText</code> UI element in the UI-element <code>PanelGrid</code></td>
<td>id: <code>author</code>; value: <code>${form.author}</code>; required: True</td>
</tr>
</tbody>
</table>
How To... Handle Value Change Events

<table>
<thead>
<tr>
<th>label</th>
<th>#{msgs.authorPrompt}</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Message UI element in the UI-element PanelGrid</strong></td>
<td></td>
</tr>
<tr>
<td>for</td>
<td>author</td>
</tr>
<tr>
<td>errorClass</td>
<td>errorMessage</td>
</tr>
</tbody>
</table>

10. Results of the index.jsp view

11. Now the list of categories and subcategories should be added in the index view. Click the JSF HTML toolset in the Palette, this will show all the UI elements available within it.

12. Drag and drop an OutputText element to the Web Page Editor and enter the following parameters values:

<table>
<thead>
<tr>
<th>OutputText UI element in the UI-element PanelGrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
</tr>
<tr>
<td>#{msgs.categoryPrompt}</td>
</tr>
<tr>
<td>styleClass</td>
</tr>
<tr>
<td>label</td>
</tr>
</tbody>
</table>

13. Drag and drop a SelectOneMenu element to the Web Page Editor. In the Properties view enter the following values:

<table>
<thead>
<tr>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>The onchange property is used to force a form submit after the menu’s value is changed. The valueChangeListener property has a method expression to a method that responds to value changes</td>
</tr>
</tbody>
</table>
How To... Handle Value Change Events

<table>
<thead>
<tr>
<th>SelectOneMenu UI element in the UI-element PanelGrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>id: category</td>
</tr>
<tr>
<td>value: #{form.category}</td>
</tr>
<tr>
<td>onChange: submit()</td>
</tr>
<tr>
<td>valueChangeListener: #{form.categoryChanged}</td>
</tr>
</tbody>
</table>

14. In the Properties view, click the Add button to add a SelectItems UI element in the Choices table

**Note**

The SelectItems UI element specifies items for the SelectOneMenu UI element

![SelectOneMenu UI element in the Properties view](image)

15. Select the SelectItems UI element in the Web Page Editor. In the Properties view, enter #{form.categories} in the value property

**Note**

The Value property is bound to the categories property of the BookForm java class. When this page is displayed, the getCategories method is called to obtain the element that will fill the SelectOneMenu UI element
16. Drag and drop a Message UI element to the Web Page Editor. In the Properties view enter the following values:

<table>
<thead>
<tr>
<th>Message UI element in the UI-element PanelGrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>for category errorClass errorMessage</td>
</tr>
</tbody>
</table>

17. Repeat steps 12-16 to add the subcategory menu. The following table contains the hierarchy of the UI elements that should be added in the index view.

**Note**
Notice the subcategory menu does not have values for the onChange and the valueChangeListener properties, because the application doesn’t need to handle the events fired by the subcategory menu.

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OutputText UI element in the UI-element PanelGrid</strong></td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>#{msgs.subcategoryPrompt}</td>
</tr>
<tr>
<td>styleClass</td>
<td>label</td>
</tr>
<tr>
<td><strong>SelectOneMenu UI element in the UI-element PanelGrid</strong></td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>subcategory</td>
</tr>
<tr>
<td>value</td>
<td>#{form.subcategory}</td>
</tr>
<tr>
<td><strong>SelectItems UI element in the UI-element SelectOneMenu</strong></td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>#{form.subcategories}</td>
</tr>
<tr>
<td><strong>Message UI element in the UI-element PanelGrid</strong></td>
<td></td>
</tr>
<tr>
<td>for</td>
<td>subcategory</td>
</tr>
</tbody>
</table>
18. Drag and drop a CommandButton element to the Web Page Editor. In the Properties view enter the following values:

<table>
<thead>
<tr>
<th>CommandButton UI element in the UI-element PanelGrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
</tr>
</tbody>
</table>

19. Result of index.jsp view

20. Save the changes you made
4.4 Implement immediate event handling

Let's analyze what would happen if the user doesn't enter any value for the required fields (title and author), but changes the book's category selection.

The validation will result in an error, because the category menu submits its form when its value is changed and the corresponding error message is going to be displayed in the index view.

The validation should be processed when the submit button is activated, but not when the category is changed. So the solution is to make the category menu an immediate UI element and to add some code lines in the valueChangeListener method that will prevent validations for the other elements in the form.

1. In the index.jsp page, select the Category menu, go to the Properties view and enter true in the immediate property

   Important

   The immediate events are fired after the Apply Request Values phase, so the category will be changed before the other UI elements in the form are validated in the Process Validations phase.
2. In the BookForm java class modify the categoryChanged method by adding the following code

```java
public void categoryChanged(ValueChangeEvent event) {
    FacesContext context = FacesContext.getCurrentInstance();
    context.renderResponse();
}
```

*Important*

In general the valueChangeListener methods will be invoked after the Process Validation phase, but in this case the category menu is an immediate UI element, so this method will be invoked before the Process Validation phase. To prevent validations for the other UI elements, the context renderResponse method could be used in the valueChangeListener method to skip the rest of the life cycle up to the Render Response phase.
3. Save the changes you made

4.5 Build, Deploy and Run your application

1. Create the application.xml deployment descriptor, sets the WAR file to “demo.sap.com~veventjsf~web.war” and the context root to “veventjsf” as indicated in the Hello World JSF tutorial (Create a Hello World Application using JavaServer Faces [Extern]).

2. Save changes.

3. Build and deploy the application.

4. Run the application using the following simplified URL:
   
   http://<servername>:<httpport>/veventjsf/faces/index.jsp

5. Results

   The category can be changed without filling the required Title and Author fields and no error messages are displayed.