How To... Convert And Validate Data

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
## Document History

<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>
</tbody>
</table>
### Typographic Conventions

<table>
<thead>
<tr>
<th>Type Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example Text</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>Example text</td>
<td>Emphasized words or phrases in body text, graphic titles, and table titles</td>
</tr>
<tr>
<td>Example text</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>Example text</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;Example text&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>
</tbody>
</table>

### Icons

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🚨</td>
<td>Caution</td>
</tr>
<tr>
<td>📜</td>
<td>Note or Important</td>
</tr>
<tr>
<td>📚</td>
<td>Example</td>
</tr>
<tr>
<td>📘</td>
<td>Recommendation or Tip</td>
</tr>
</tbody>
</table>

**EXAMPLE TEXT**  Keys on the keyboard, for example, F2 or ENTER.
# Table of Contents

1. **Business Scenario** ................................................................. 1

2. **Background Information** ..................................................... 1

3. **Prerequisites** ................................................................. 1

4. **Step-by-Step Procedure** .................................................. 2
   4.1 Tutorial Setup ........................................................................ 2
   4.2 Create a Java Class ............................................................ 2
   4.3 Create a Style file .............................................................. 4
   4.4 Create the JSP Pages .......................................................... 5
   4.5 Converters ........................................................................ 15
   4.6 Validators ........................................................................ 18
   4.7 Bypassing Validation ......................................................... 19
   4.8 Build, Deploy and Run your application .......................... 20
1. Business Scenario

The following guide will explain how input data is converted to Java objects and how the data is validated against some application-specific constraints before the model is updated, so invalid inputs won’t affect the model integrity.

The user interface for this Web application will allow you to create special offers for a company. The user will need to provide the following information:

- The product short description
- The product price
- The offer expiration date
- The stock availability

JSF will convert and validate all user input. If there are any errors, the page is redisplayed with the values entered by the user in order to correct them. If all validations are successful, the application will continue with the creation process.

2. Background Information

The execution of a JavaServer Faces page follows the Web request/response paradigm. When a client makes a request to a JSF page, the server side JSF runtime processes the request and delivers back a response. One of the distinguished features possessed by JSF is the division of the request processing into multiple phases. Those phases deal with common problems which any modern Web application solves either in a proprietary way or with a help of an additional framework. Among those problems are the conversion of the client request data into application depended model and ensuring the validity of the data before executing any back-end business logic. To learn more about the JSF request processing lifecycle you can visit Sun's java website for JSF.

You will now be familiarized with the phase of the lifecycle that deal with conversion and validation and how the conversion and validation process protect the business logic from invalid data.

3. Prerequisites

The following is a list of all you need for developing JSF applications.

- 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. You will get to know how to convert data types and how to ensure that the correct values are entered by a user.

This Web application will consist of three views. In the first view, using the Process button, the user can navigate to a second view in which the information can be reviewed. This navigation button should only be followed up by the JSF framework if the entries for all the input fields are correct. If this is not the case, a corresponding error message is displayed. Using the Cancel button, the user can navigate to a third view and bypass the data validation.

4.1 Tutorial Setup

1. Create a Web Module Development Component named converterjsf/web as indicated in the Hello World JSF tutorial (Create a Hello World Application using JavaServer Faces [Extern]).

2. Create an Enterprise Application Development Component named converterjsf/ear as indicated in the Hello World JSF tutorial (Create a Hello World Application using JavaServer Faces [Extern]).

4.2 Create a Java Class

1. From the context menu of the Java Resources: source folder in the Web Module project create a Java class as indicated in the Navigation tutorial (Create Your First JSF Tutorial [Extern]). Enter Product in the Name field, com.sap.tutorial.jsf.conv.beans in the Package field, declare the attributes and generate the corresponding Getters and Setters methods shown in the following code.

```java
public class Product {
    private String description = new String();
}```
private double price;
private Date expirationDate = new Date();
private int stock;

public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}

public double getPrice() {
    return price;
}

public void setPrice(double price) {
    this.price = price;
}

public Date getExpirationDate() {
    return expirationDate;
}

public void setExpirationDate(Date expirationDate) {
    this.expirationDate = expirationDate;
}

public int getStock() {
    return stock;
}

public void setStock(int stock) {
    this.stock = stock;
}

2. Configure the Product Java class in the application configuration resource file faces-config.xml using the managed-bean XML element as indicated in the Navigation tutorial (Create Your First JSF Tutorial [Extern]). Enter product in the Name field to reference the Product java class and select session in the Scope field.

<managed-bean>
    <managed-bean-name>product</managed-bean-name>
</managed-bean>
<managed-bean-class>
    com.sap.tutorial.jsf.conv.beans.Product
</managed-bean-class>

<managed-bean-scope>session</managed-bean-scope>
</managed-bean>

### 4.3 Create a Style file

Instead of using a hardwired style, it is better to use a style sheet.

1. Drill into the Web Module project and right click on the `WebContent` folder and in the context menu select `New` → `Other`… In the popup window select `Web` → `CSS` and click the `Next` button.

2. Enter `styles.css` in the `File Name` field and click the `Finish` button.

3. Define the following CSS classes:

   ```css
   .errorMessage {
       font-family: Verdana, Arial, Sans-Serif;
       font-size: 10px;
       color: red;
       font-style: normal;
   }
   
   .title {
       font-family: Verdana, Arial, Sans-Serif;
       font-weight: bold;
       font-size: 12px;
       color: #edbb10;
       font-style: normal;
   }
   
   .label {
       font-family: Verdana, Arial, Sans-Serif;
       font-weight: bold;
       font-size: 12px;
       color: #808080;
       font-style: normal;
   }
   ```
4. Save the changes you made.

4.4 Create the JSP Pages

1. Drill into the Web Module project and right click on the WebContent folder and in the context menu select New → JSP.

2. 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

3. 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>

4. Drag and drop a Form element (found in the JSF HTML elements) to the Web Page Editor

5. Place an Output Text UI element between the <h:view>…< /h:view> tags and select the Properties view in the bottom window pane. Click in the Attributes tag on your left and enter the following values in the corresponding properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Product Offer</td>
</tr>
<tr>
<td>styleClass</td>
<td>title</td>
</tr>
</tbody>
</table>

   Important

   The value property is the component’s value
   The styleClass property is the CSS class name included in step 3

6. Place a Panel Grid element between the <h:form>… </h:form> tag. You are going to need an extra column to show the error messages next to the components that reported them. Set the Columns property to 3 in the Properties view (bottom window pane).

7. Take a look at the tags that were inserted into the JSP page. Put the cursor on the first Output Text UI element and then select the Properties view in the bottom window pane. click in the Attributes tag on your left and enter the following values in the corresponding properties:
8. Drag and drop a Text Input UI element in the second column. In the Properties view in the bottom window pane, click in the Attributes tag on your left and enter the following values in the corresponding properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>description</td>
</tr>
<tr>
<td>value</td>
<td>#{product.description}</td>
</tr>
<tr>
<td>label</td>
<td>Description</td>
</tr>
<tr>
<td>required</td>
<td>true</td>
</tr>
</tbody>
</table>

**Important**

The id property is the identifier for the component and allows other JSF tags to access it. The value property bounds to properties of the Product java class. The label property is the description of the component for use in error messages.
The *required* property specifies if requires a value to be entered.

9. Drag and drop a *Message* element to the Web Page Editor. In the Properties view in the bottom window pane, click in the *Attributes* tag on your left and enter the following values to the corresponding properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>for</td>
<td>description</td>
</tr>
<tr>
<td>errorClass</td>
<td>errorMessage</td>
</tr>
</tbody>
</table>

**Important**

The *for* property is the component id for which to display the message. The *errorClass* property is the CSS class applied to error messages.
How To... Convert And Validate Data

10. Repeat steps 7-9 to incorporate the following UI elements:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property Value</strong></td>
<td><strong>Price</strong></td>
</tr>
<tr>
<td><strong>styleClass</strong></td>
<td><strong>label</strong></td>
</tr>
<tr>
<td><strong>InputText UI element</strong></td>
<td><strong>price</strong></td>
</tr>
<tr>
<td><strong>value</strong></td>
<td>#{product.price}</td>
</tr>
<tr>
<td><strong>label</strong></td>
<td>Price</td>
</tr>
<tr>
<td><strong>required</strong></td>
<td>true</td>
</tr>
<tr>
<td><strong>Message UI element</strong></td>
<td></td>
</tr>
<tr>
<td><strong>for</strong></td>
<td>price</td>
</tr>
<tr>
<td><strong>errorClass</strong></td>
<td>errorMessage</td>
</tr>
</tbody>
</table>
OutputText UI element in the UI-element PanelGrid
value Expiration Date
styleClass label
InputText UI element
Id expirationDate
value #{product.expirationDate}
label Expiration Date
required true
Message UI element
for expirationDate
errorClass errorMessage
OutputText UI element in the UI-element PanelGrid
value In Stock
styleClass label
InputText UI element
Id stock
value #{product.stock}
label In Stock
required true
Message UI element
for stock
errorClass errorMessage

11. Drag and drop two (2) CommandButton elements to the Web Page Editor. Enter the following values on the corresponding properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CommandButton1 UI element</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>Process</td>
</tr>
<tr>
<td>action</td>
<td>process</td>
</tr>
<tr>
<td>CommandButton2 UI element</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>Cancel</td>
</tr>
<tr>
<td>action</td>
<td>cancel</td>
</tr>
</tbody>
</table>

12. Result of index.jsp
13. Save the changes you made.

14. Create the second view result.jsp. This page will show the inputs that the user provided, using a different converter for the price amount. Right click on the WebContent folder and in the context menu and select New → JSP.

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

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

16. The following table contains the hierarchy of the UI elements contained in the result view:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ViewRoot UI element</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Form UI element in the UI-element ViewRoot</strong></td>
<td></td>
</tr>
<tr>
<td><strong>OutputText UI element in the UI-element Form</strong></td>
<td>Offer created successfully</td>
</tr>
<tr>
<td>value</td>
<td></td>
</tr>
<tr>
<td>styleClass</td>
<td>title</td>
</tr>
<tr>
<td><strong>PanelGrid UI element in the UI-element Form</strong></td>
<td>Border 0</td>
</tr>
<tr>
<td>Border</td>
<td>0</td>
</tr>
</tbody>
</table>
### Data Conversion and Validation

<table>
<thead>
<tr>
<th>Columns</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OutputText UI element in the UI-element</strong> PanelGrid</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>Description</td>
</tr>
<tr>
<td>styleClass</td>
<td>label</td>
</tr>
<tr>
<td><strong>OutputText UI element in the UI-element</strong> PanelGrid</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>#{product.description}</td>
</tr>
<tr>
<td>styleClass</td>
<td>text</td>
</tr>
<tr>
<td><strong>OutputText UI element in the UI-element</strong> PanelGrid</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>Price</td>
</tr>
<tr>
<td>styleClass</td>
<td>label</td>
</tr>
<tr>
<td><strong>OutputText UI element in the UI-element</strong> PanelGrid</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>#{product.price}</td>
</tr>
<tr>
<td>styleClass</td>
<td>text</td>
</tr>
<tr>
<td><strong>OutputText UI element in the UI-element</strong> PanelGrid</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>Expiration Date</td>
</tr>
<tr>
<td>styleClass</td>
<td>label</td>
</tr>
<tr>
<td><strong>OutputText UI element in the UI-element</strong> PanelGrid</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>#{product.expirationDate}</td>
</tr>
<tr>
<td>styleClass</td>
<td>text</td>
</tr>
<tr>
<td><strong>OutputText UI element in the UI-element</strong> PanelGrid</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>In Stock</td>
</tr>
<tr>
<td>styleClass</td>
<td>label</td>
</tr>
<tr>
<td><strong>OutputText UI element in the UI-element</strong> PanelGrid</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>#{product.stock}</td>
</tr>
<tr>
<td>styleClass</td>
<td>text</td>
</tr>
</tbody>
</table>

**CommandButton UI element**

| value | Continue |
| action | continue |

17. Result of result.jsp
18. Save the changes you made.

19. Create the third view canceled.jsp. This page will allow the user to bypass the validation and cancel the transaction. Right click on the WebContent folder and in the context menu and select New → JSP.

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

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

21. The following table contains the hierarchy of the UI elements contained in the canceled view:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViewRoot UI element</td>
<td></td>
</tr>
<tr>
<td>Form UI element in the UI-element ViewRoot</td>
<td></td>
</tr>
<tr>
<td>OutputText UI element in the UI-element Form</td>
<td>The transaction has been canceled.</td>
</tr>
<tr>
<td>value</td>
<td></td>
</tr>
<tr>
<td>styleClass</td>
<td>title</td>
</tr>
</tbody>
</table>
How To... Convert And Validate Data

---

**CommandButton UI element**

<table>
<thead>
<tr>
<th>value</th>
<th>Back</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>back</td>
</tr>
</tbody>
</table>

22. Result of `canceled.jsp`

23. Save the changes you made.

24. To complete the JSF application, we need to specify the navigation rules, drill into the Web Module project, in the `WebContent → WEB-INF` folder and open the `faces-config.xml` file.

25. Go to the Navigation Rule tab and define the navigation as indicated in the Navigation tutorial (Create Your First JSF Tutorial [Extern]). The navigation flow should look like the following image:
26. Select the Source tab. The following XML code should be added automatically between the `<faces-config ... >` `<faces-config>` tags

```xml
<navigation-rule>
    <display-name>index</display-name>
    <from-view-id>/index.jsp</from-view-id>
    <navigation-case>
        <from-outcome>process</from-outcome>
        <to-view-id>/result.jsp</to-view-id>
    </navigation-case>
</navigation-rule>

<navigation-rule>
    <display-name>index</display-name>
    <from-view-id>/index.jsp</from-view-id>
    <navigation-case>
        <from-outcome>cancel</from-outcome>
        <to-view-id>/canceled.jsp</to-view-id>
    </navigation-case>
</navigation-rule>

<navigation-rule>
    <display-name>result</display-name>
    <from-view-id>/result.jsp</from-view-id>
    <navigation-case>
        <from-outcome>continue</from-outcome>
        <to-view-id>/index.jsp</to-view-id>
    </navigation-case>
</navigation-rule>
```
27. Save the changes you made.

4.5 Converters

Conversion is a two-way process by which data is converted from the String-based representation of the user interface to the Object-based representation demanded by the web application and back again.

1. In the index.jsp page select the Price InputText UI element, in the Properties view in the bottom window pane, click the Quick Edit tab and go to the Converter section. In the drop down list, select the Number option and click the Add button.

2. Double click the ConvertNumber you just added, click the Attributes tab and set the minFractionDigits property to 2.

Note

This converter will format the current value of the Price UI element with at least two digits after the decimal point.
3. In the index.jsp page select the expirationDate InputText, in the Properties view click the Quick Edit tab and go to the Converter section. In the drop down list, select the DateTime option and click the Add button.

4. Double click ConvertDateTime you just added and set the Pattern property to **MM/dd/yyyy**.

**Note**

This converter will format the current value of the expirationDate UI element with the MM/dd/yyyy pattern.
5. In the result.jsp page, follow steps 1-2 to attach a ConvertNumber converter to the Price OutputText UI element and enter the following values in the corresponding properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>currency</td>
</tr>
<tr>
<td>Currency Symbol</td>
<td>$</td>
</tr>
</tbody>
</table>

**Note**

This converter automatically supplies a currency symbol and decimal separators.
6. In the `result.jsp` page, follow steps 3-4 to attach a `ConvertDateTime` to the `expirationDate` OutputText UI element and set the `Pattern` property to `MM/dd/yyyy`.

7. Save the changes you made.

### 4.6 Validators

Validation is the process by which a piece of converted data has one or more correctness checks applied to it.

1. In the `index.jsp` page select the `Price` InputText UI element, in the `Properties` view in the bottom window pane, click the `Quick Edit` tab and go to the `Validators` section. In the drop down list, select the `DoubleRange` option and click the `Add` button.

2. Double click `validateDoubleRange` you just added and set the `Minimum` property to 1.

**Note**

This validator will check limits for a numerical value (for example, >1). All the standard validator tags have minimum and maximum properties. You need to supply one or both values.
3. (Optional) Attach a validateLongRange validator to the stock InputText UI element and set the Minimum property to 10.

4. Save the changes you made

### 4.7 Bypassing Validation

Validation errors force a redisplay of the current page, but this behavior can be problematic with certain navigation actions, for example, a Cancel button. When a command has the immediate attribute set, it is executed during the Apply Request Values phase and bypass the validation.

1. In the index.jsp page select the Cancel commandButton UI element, go to the Properties view, click the Attributes tab and set the Immediate property to true.
2. Save the changes you made.

4.8 Build, Deploy and Run your application

1. Create the application.xml deployment descriptor, sets the WAR file to “demo.sap.com~converterjsf~web.war” and the context root to “converter” 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>/converterjsf/faces/index.jsp

5. Results:
How To... Convert And Validate Data

Product Offer
Description: 37" Flat-Panel LCD HDTV
Price: 799
Expiration Date: 03/31/2009
In Stock: 100

Process | Cancel

Offer created successfully
Description: 37" Flat-Panel LCD HDTV
Price: $799.00
Expiration Date: 03/31/2009
In Stock: 100

Continue