SAP NetWeaver
How-To Guide

How To... Develop International JSF Applications

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

Document Version	Description
1.00	First official release of this guide

Typographic Conventions

Type Style	Description
Example Text	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
Example text	Emphasized words or phrases in body text, graphic titles, and table titles
`Example text`	File and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
`Example text`	User entry texts. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example text>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE TEXT`	Keys on the keyboard, for example, F2 or ENTER.

Icons

Icon	Description
	Caution
	Note or Important
	Example
	Recommendation or Tip

Table of Contents

1. Business Scenario

In the previous tutorial you created a JSF application that used standard and custom converters and validators to maintain the integrity of your model.

The following guide will show you how to organize the texts (messages, labels, titles) on your Product Offer application so it will meet the requirements of any specific geographic region of the world.

2. Background Information

JSF handles localization in the same way other Java Web-based applications handle it: using ResourceBundles. Resource bundles contain locale-specific objects. When your program needs a locale-specific resource, a String for example, your program can load it from the resource bundle that is appropriate for the current user's locale. In this way, you can write program code that is largely independent of the user's. Resource bundles contain key/value pairs. The keys uniquely identify a locale-specific object in the bundle.

JSF takes advantage of the user-configurable language information sent by a Web browser with each request to choose the best locale among the choices available to the application

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 to 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])
- You have successfully completed the Product Offer tutorial Part 2 (Custom Converters And Validators [extern])

4. Step-by-Step Procedure

In the following sections you will optimize the Product Offer tutorial Part 2 (Custom Converters And Validators [extern]) by collecting labels, titles and messages in a central location.

Before looking at the details of creating the resource bundle, it would be helpful to summarize the message strings that need to be collected

JSP Page	Message strings
Index.jsp	Product Offer title, Description label, Code label, Price label, Expiration Date label, In Stock label, Process button, Cancel button, Code Converter error message, Code Validator error messages
Result.jsp	Success message, Description label, Code label, Price label, Expiration Date label, In Stock label, Continue button
Canceled.jsp	Cancel message, Back button

The JSF framework is fully internationalized. However, when you developed your own components, in this case custom converters and validators, you must retrieve your error messages from a message bundle, so that your components have the same level of localizability that users expect from the JSF framework.

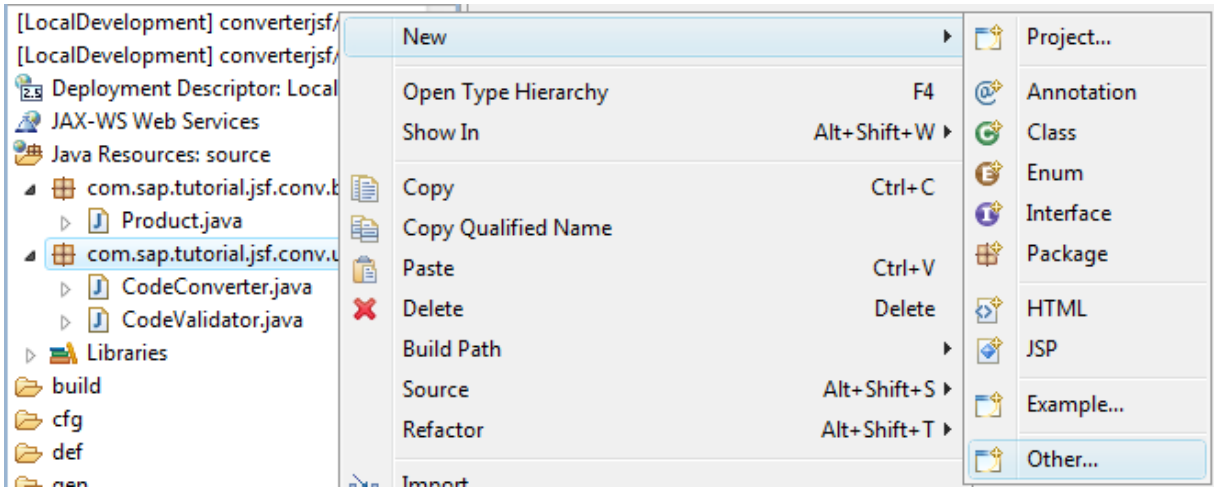
You will use this information to create a resource bundle file in the *properties* format with all the message strings. Then you will expose it to the *Faces* runtime and use it from the JSF views.

4.1 Tutorial Setup

1. Download the ZIP file *05_converterjsf_init.zip*, which contains the initial JSF project for this tutorial. Save it in a local directory.
2. Unzip the contents and import the *Development Components* in the JDI workspace of the SAP NetWeaver Developer Studio as indicated in the Product Offer tutorial Part 2 (Custom Converters And Validators [extern])

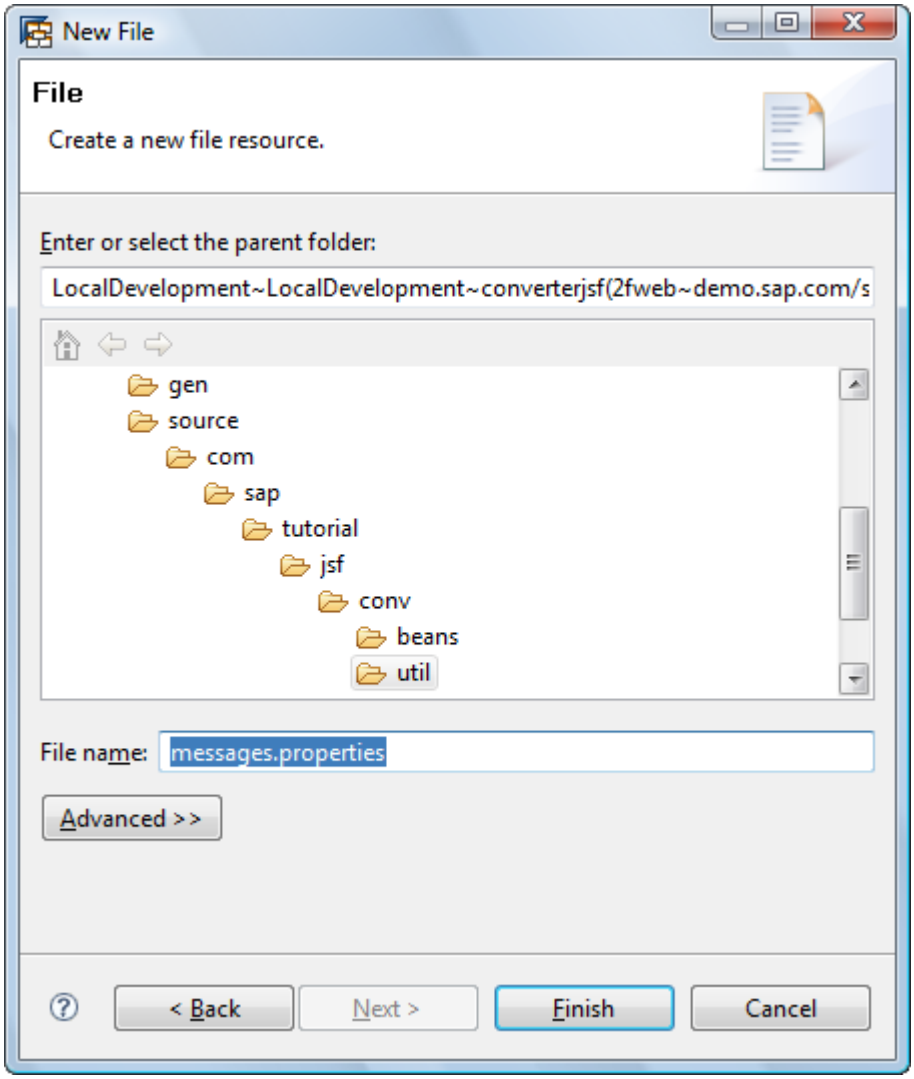
4.2 Create the ResourceBundles

1. From the context menu of the *com.sap.tutorial.jsf.conv.util* package in the *Java Resources: source* folder, select *New* → *Other…*



2. In the popup window select *General* → *File* and click the *Next* button.
3. Enter **`messages.properties`** in the *File Name* field and click the *Finish* button.

> **Note**
>
> You can choose any directory path and file name, but you must use the extension **`.properties`**



4. The following code shows the keys and values for the English version of the localized messages

```
offer_title=Product Offer
description=Description
code=Code
price=Price
expiration_date=Expiration Date
in_stock=In Stock
process_button=Process
cancel_button=Cancel
success_title=Offer created successfully
continue_button=Continue
cancel_title=The transaction has been canceled
back_button=Back
invalidCodeCharacter=Code: Convertion error: Invalid character found.
invalidCodeCharacter_detail=Code: Convertion error: The code contains
invalid characters.
invalidCodeFormat=Code: Validation error: Invalid code format.
invalidCodeFormat_detail=Code: Validation error: The code contains invalid
format. It should be 'XYZ1234' or 'XYZ-1234'.
invalidCodeLength=Code: Validation error: Invalid length.
invalidCodeLength_detail=Code: Validation error: The code contains invalid
format. Length allowed is 7 or 8.
```

5. Create the Spanish version of the localized messages. When you localize a bundle file, you need to add a locale suffix to the file name, that is, an underscore followed by the lower case, two-letter ISO-639 language code. To do this, create a new file called **`messages_es.properties`** and add the following code

> **Note**
>
> Special characters are encoded as \uxxxx escape sequences. To learn more about the Unicode characters you can visit Unicode website

```
offer_title=Producto en Promoci\u00f3n
description=Descripci\u00f3n
code=C\u00f3digo
price=Precio
expiration_date=Fecha de Expiraci\u00f3n
in_stock=En Almac\u00e9n
process_button=Procesar
cancel_button=Cancelar
```

```
success_title=Oferta creada exitosamente

continue_button=Continuar

cancel_title=La transacci\u00f3n ha sido cancelada

back_button=Regresar

invalidCodeCharacter=C\u00f3digo: Error de Converci\u00f3n: Caracter
inv\u00e1lido.

invalidCodeCharacter_detail=C\u00f3digo: Error de Converci\u00f3n: El
c\u00f3digo contiene caracteres inv\u00e1lidos.

invalidCodeFormat=C\u00f3digo: Error de Validaci\u00f3n: Formato
inv\u00e1lido.

invalidCodeFormat_detail=C\u00f3digo: Error de Validaci\u00f3n: El
c\u00f3digo contiene formato inv\u00e1lido. Formato permitido es 'XYZ1234'
o 'XYZ-1234'.

invalidCodeLength=C\u00f3digo: Error de Validaci\u00f3n: Longitud
inv\u00e1lido.

invalidCodeLength_detail=C\u00f3digo: Error de Validaci\u00f3n: El
c\u00f3digo contiene longitud inv\u00e1lida. Longitud permitida es 7 u 8.
```

6. Save the changes you made

4.3 Get Error Messages from ResourceBundles

1. In the *com.sap.tutorial.jsf.conv.util* package you will find a *Messages* java class provided by Geary, David and Cay Horstmann. Core JavaServerTM Faces. 2nd ed. This class will help you to retrieve the error messages from a ResourceBundle by:

   a. Getting the locale:

   ```
   Locale locale = context.getViewRoot().getLocale();
   ```

   b. Getting the class loader needed to locate the resource bundle

   ```
   ClassLoader loader =
   thread.currentThread().getContextClassLoader();
   ```

   c. Getting the resource bundle

   ```
   bundle = ResourceBundle.getBundle(bundle1, locale, loader);
   ```

   d. Getting the message string from the resource bundle

   ```
   resource = bundle.getString(resourceId);
   ```

2. Open the *CodeConverter* class and use the *Messages* class to get the error messages from the ResourceBundle by changing the getAsObject method as follows:

```
public Object getAsObject(FacesContext arg0, UIComponent arg1, String
arg2) {
  int i = 0;
  boolean foundInvalidCharacter = false;
  if (arg2 == null){
```

```
            return null;
        }
        arg2 = arg2.replace("-", "").trim();
        StringBuilder builder = new StringBuilder(arg2);
        while (i < builder.length() && !foundInvalidCharacter) {
            char ch = builder.charAt(i);
            if (Character.isLetter(ch) || Character.isDigit(ch))
                i++;
            else if (Character.isWhitespace(ch))
                builder.deleteCharAt(i);
            else {
                foundInvalidCharacter = true;
            }
        }
        if (foundInvalidCharacter) {
                FacesMessage message = Messages.getMessage(
                   "com.sap.tutorial.jsf.conv.util.messages",
                   "invalidCodeCharacter", null);
                  message.setSeverity(FacesMessage.SEVERITY_ERROR);
                  throw new ConverterException(message);
        }
        return builder.toString();
    }
```

3. Open the *CodeValidator* class and use the *Messages* class to get the error messages from the ResourceBundle by changing the validate method as follows:

```
    public void validate(FacesContext arg0, UIComponent arg1, Object arg2)
            throws ValidatorException {
        int i = 0;
        boolean foundError = false;
        FacesMessage message = new FacesMessage();

        String code = arg2.toString();
        code = code.replace("-", "").trim();
        StringBuilder builder = new StringBuilder(code);
        if (builder.length() == 7) {
            while (i < builder.length() && !foundError) {
```

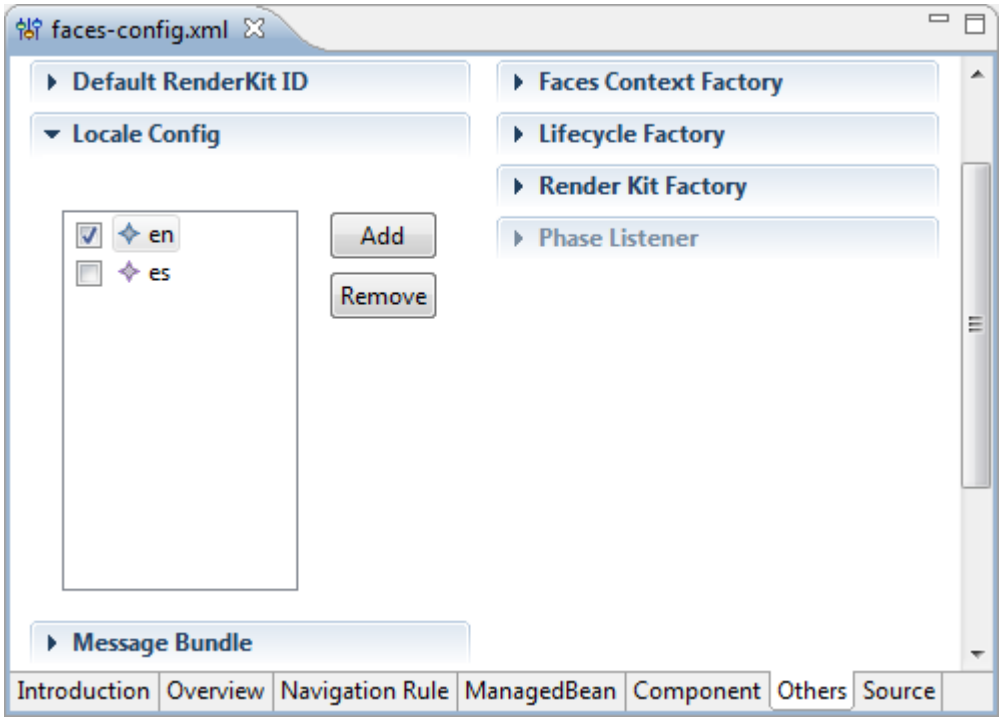
```
            char ch = builder.charAt(i);
            if (Character.isLetter(ch) && i < 3)
               i++;
            else if (Character.isDigit(ch) && i >= 3)
               i++;
            else if (Character.isWhitespace(ch))
               builder.deleteCharAt(i);
            else {
               foundError = true;
                 message = Messages.getMessage(
                  "com.sap.tutorial.jsf.conv.util.messages",
                  "invalidCodeFormat", null);
            }
         }
      } else {
         foundError = true;
           message = Messages.getMessage(
               "com.sap.tutorial.jsf.conv.util.messages",
               "invalidCodeLength", null);
      }
      if (foundError) {
         message.setSeverity(FacesMessage.SEVERITY_ERROR);
         throw new ValidatorException(message);
      }
   }
```

4. Save the changes

4.4 Set Locale and Expose the ResourceBundles

It is important to define the default locale and supported locales in order for Faces to find the right Local instance for the user. This is done in the <application> tag of the faces-config.xml file.

1. Open the *faces-config.xml* file, go to the *Other* tab
2. In the *Locale Config* section, add *en* (english) and *es* (spanish)



3. The following XML code will be added in the *Source* tab

```
<application>
   <locale-config>
      <default-locale>en</default-locale>
      <supported-locale>es</supported-locale>
   </locale-config>
</application>
```

4. To make available the ResourceBundle to the Application, in the Source tab of the *faces-config.xml* file, add the following XML code between the <application>… </application tags>.

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

Important

Now the messages in the *bundle* are accessible through a map variable with the name *msgs*. The other way to expose the ResourceBundle is adding the **`f:loadbundle`** element to each JSF page that needs access to the bundle. The **`resource-bundle`** element is more efficient that the **`f:loadBundle`** action because the bundle is created

once for the application. However it is a JSF1.2 feature. To make your application compatible with JSF 1.1, you must use **`f:loadBundle`**

5. Save your changes

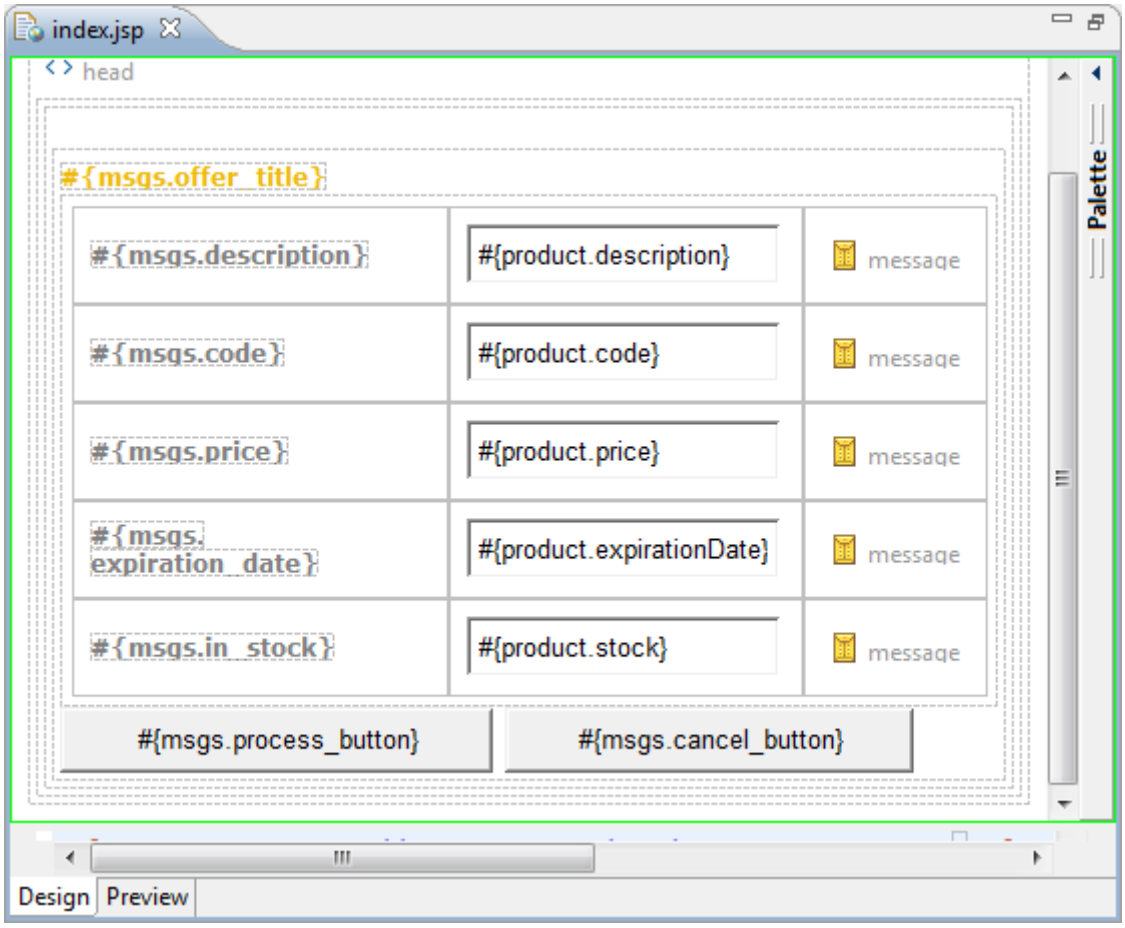
4.5 Modify the JSP Pages

1. In the index.jsp page replace the *value* property in the following UI elements:

UI element	***Value*** **property**
Product Offer *outputText*	#{msgs.offer_title}
Description *outputText*	#{msgs.description}
Code *outputText*	#{msgs.code}
Price *outputText*	#{msgs.price}
Expiration Date *outputText*	#{msgs.expiration_date}
In Stock *outputText*	#{msgs.in_stock}
Process *commandButton*	#{msgs.process_button}
Cancel *commandButton*	#{msgs.cancel_button}

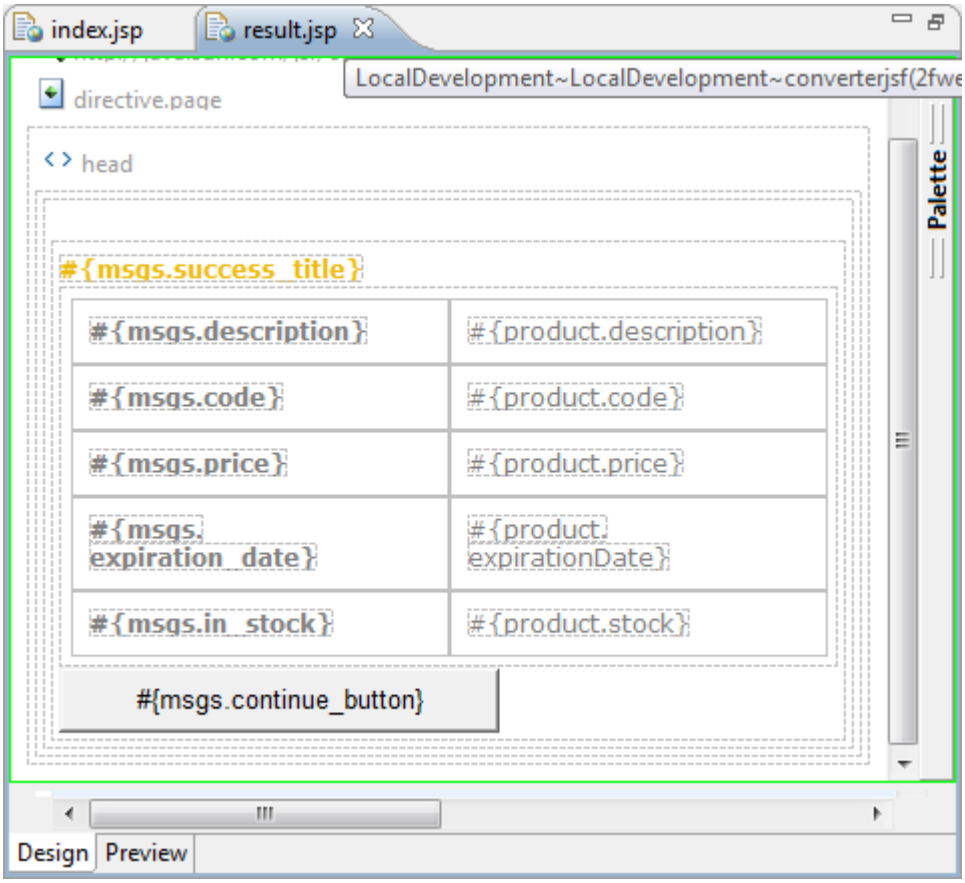
2. In the index.jsp page replace the *label* property in the following UI elements:

UI element	***Label*** **property**
Description *inputText*	#{msgs.description}
Code *inputText*	#{msgs.code}
Price *inputText*	#{msgs.price}
Expiration Date *inputText*	#{msgs.expiration_date}
In Stock *inputText*	#{msgs.in_stock}



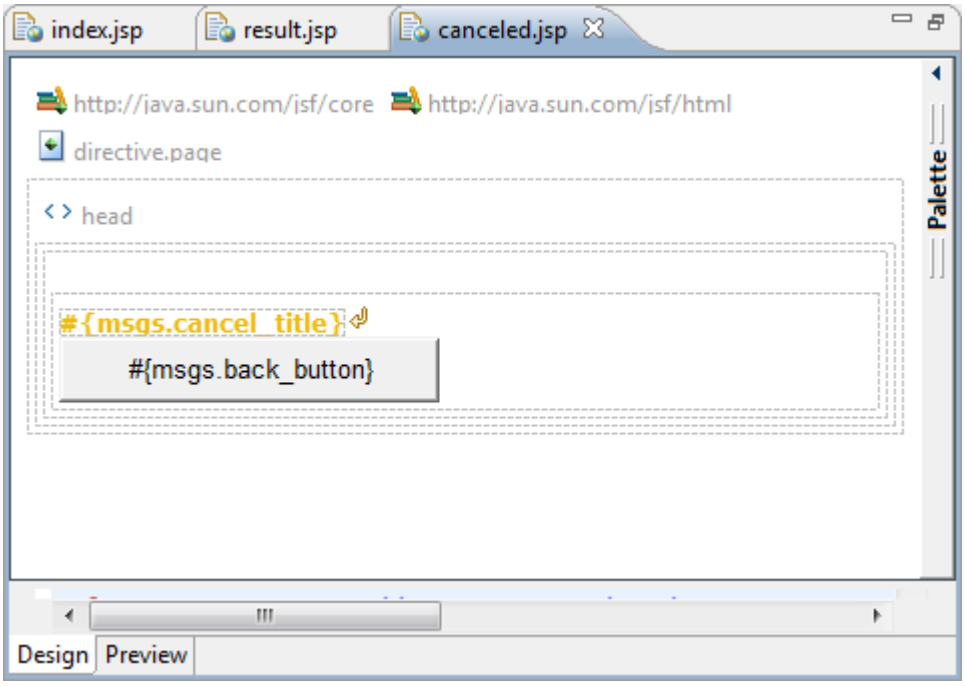
3. In the result.jsp page replace the *value* property in the following UI elements:

UI element	***Value* property**
Offer created successfully *outputText*	#{msgs.success_title}
Description *outputText*	#{msgs.description}
Code *outputText*	#{msgs.code}
Price *outputText*	#{msgs.price}
Expiration Date *outputText*	#{msgs.expiration_date}
In Stock *outputText*	#{msgs.in_stock}
Process *commandButton*	#{msgs.continue_button}



1. In the result.jsp page replace the *value* property in the following UI elements:

UI element	***Value* property**
The transaction has been canceled *outputText*	#{msgs.cancel_title}
Process *commandButton*	#{msgs.back_button}



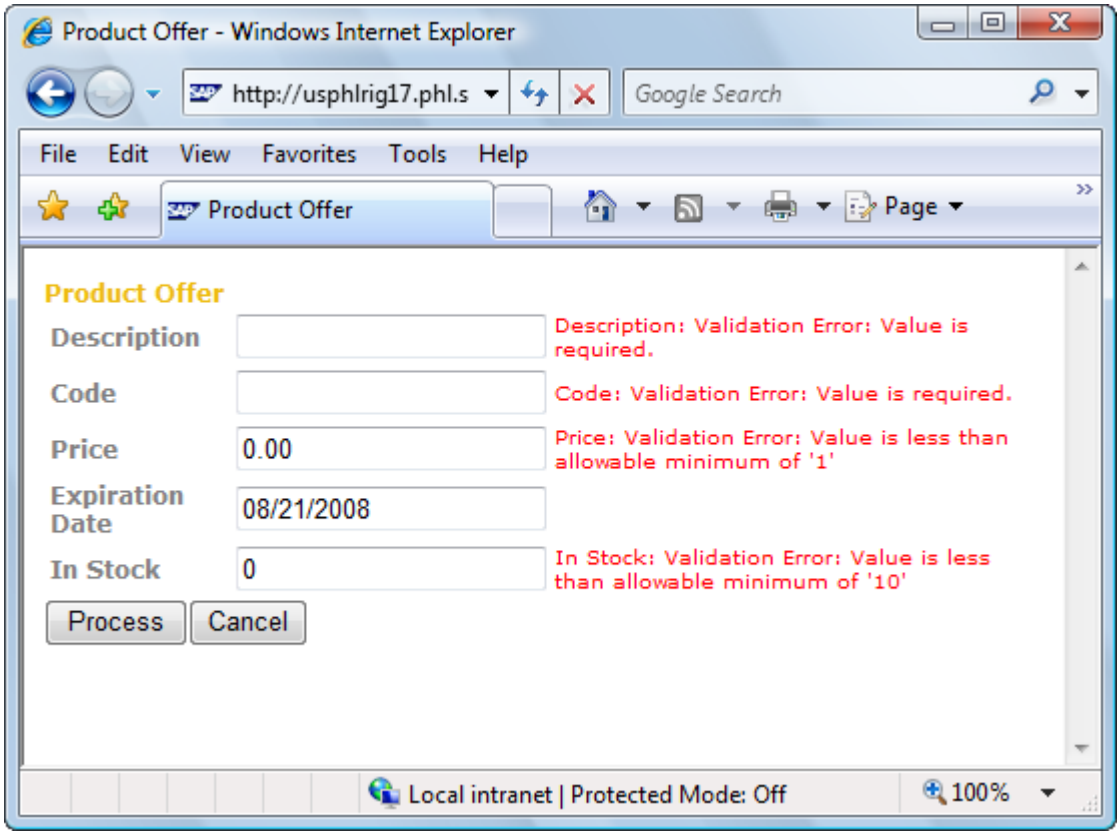
2. Save your changes

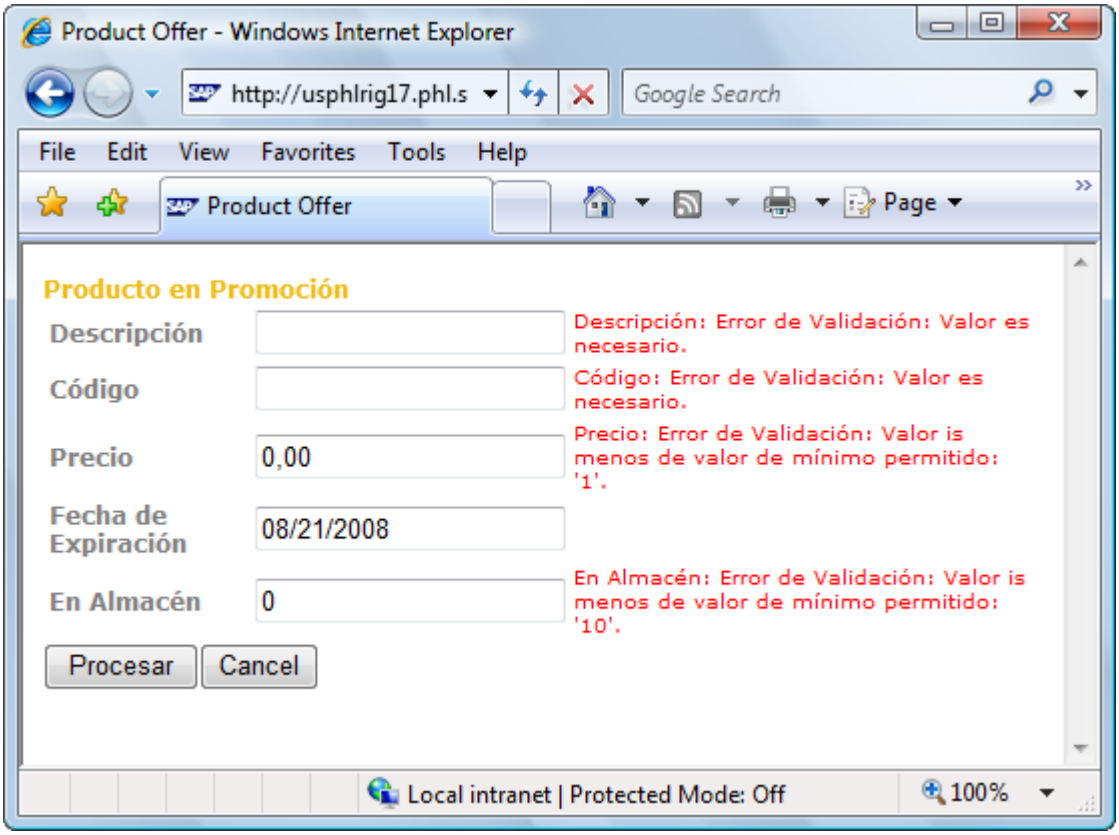
4.6 Build, Deploy and Run your application

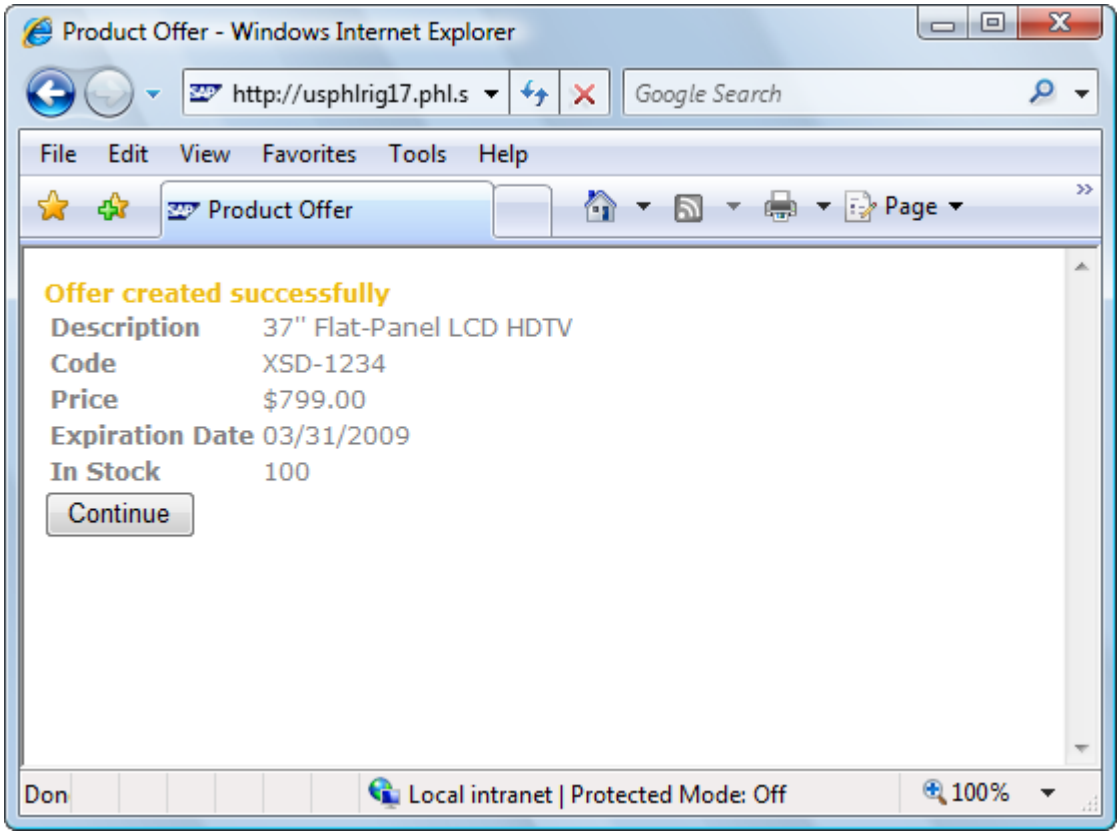
3. Build and deploy the application.
4. Run the application using the simplified URL:

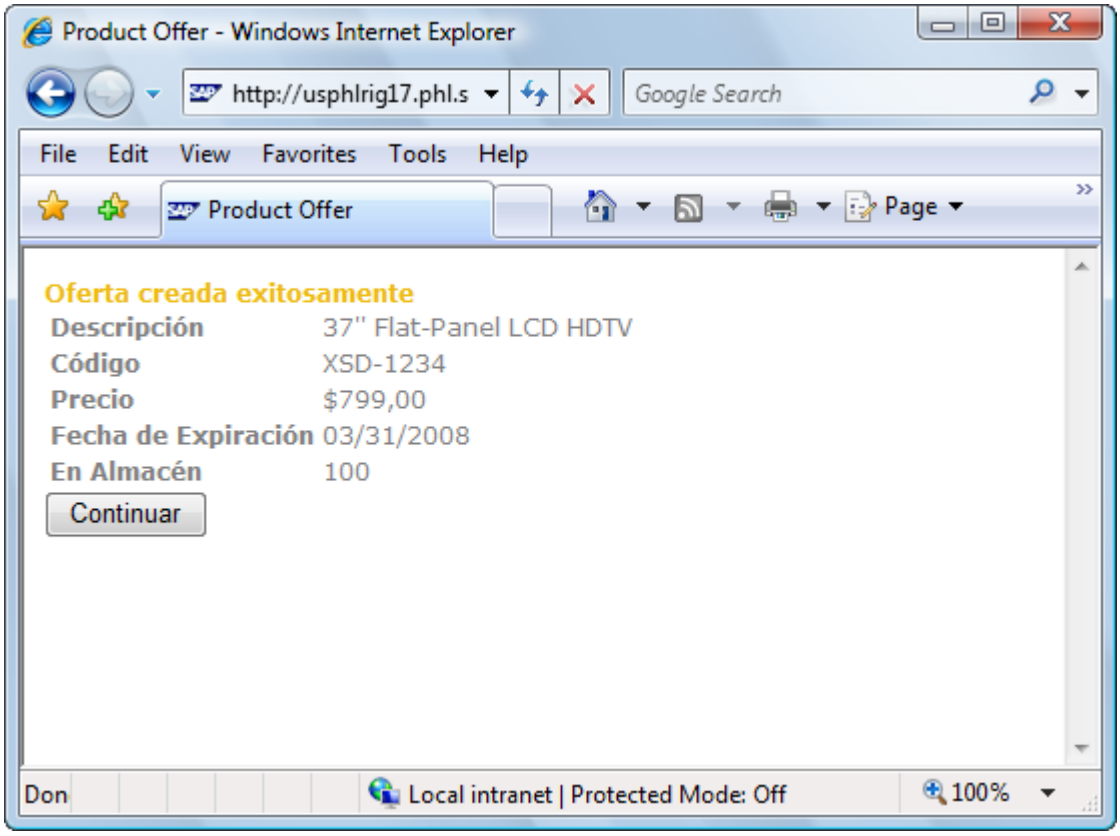
   http://<servername>:<httpport>/converterjsf/faces/index.jsp

5. Results:


Product Offer - Windows Internet Explorer
http://usphlrig17.phl.s
Google Search
File Edit View Favorites Tools Help
Product Offer
Page
Product Offer
Description
Description: Validation Error: Value is required.
Code
Code: Validation Error: Value is required.
Price
0.00
Price: Validation Error: Value is less than allowable minimum of '1'
Expiration Date
08/21/2008
In Stock
0
In Stock: Validation Error: Value is less than allowable minimum of '10'
Process
Cancel
Local intranet | Protected Mode: Off
100%


Product Offer - Windows Internet Explorer
http://usphlrig17.phl.s
Google Search
File Edit View Favorites Tools Help
Product Offer
Page
Producto en Promoción
Descripción
Descripción: Error de Validación: Valor es necesario.
Código
Código: Error de Validación: Valor es necesario.
Precio
0,00
Precio: Error de Validación: Valor is menos de valor de mínimo permitido: '1'.
Fecha de Expiración
08/21/2008
En Almacén
0
En Almacén: Error de Validación: Valor is menos de valor de mínimo permitido: '10'.
Procesar
Cancel
Local intranet | Protected Mode: Off
100%


Product Offer - Windows Internet Explorer
http://usphlrig17.phl.s
Google Search
File Edit View Favorites Tools Help
Product Offer
Page
Offer created successfully
Description 37" Flat-Panel LCD HDTV
Code XSD-1234
Price $799.00
Expiration Date 03/31/2009
In Stock 100
Continue
Don
Local intranet | Protected Mode: Off
100%


Product Offer - Windows Internet Explorer
http://usphlrig17.phl.s
Google Search
File Edit View Favorites Tools Help
Product Offer
Page
Oferta creada exitosamente
Descripción 37" Flat-Panel LCD HDTV
Código XSD-1234
Precio $799,00
Fecha de Expiración 03/31/2008
En Almacén 100
Continuar
Don
Local intranet | Protected Mode: Off
100%

www.sdn.sap.com/irj/sdn/howtoguides