Applies to:
SAP enhancement package 1 for SAP NetWeaver Composition Environment 7.1

Summary
This guide explains how to modify an existing Web Dynpro Java user interface so that it can be used within a human activity that is part of an SAP NetWeaver BPM process. Web Dynpro Java is the first user interface technology that will be supported in wave one of SAP NetWeaver BPM. Download the Source Files here.

Author: Jie Deng
Company: SAP AG
Created on: 28 November 2008

Author Bio
Jie Deng is a Product Manager in SAP AG Germany. In this role, Jie focuses on the area of Business Process Management, Business Activity Monitoring as well as process monitoring and reporting. She is responsible for the rollout to customers as well as roll-out within the organizations. Jie has been working for the SAP organization for eight years and has been part of the Product Management Group in the SAP NetWeaver BI area for six years. In 2006, Jie joined the Product Management Group in the BPM area.
**Table of Contents**

Step 1 — Locate the Web Dynpro Java Project and Run the Existing View.............................................................3  
Step 2 — Create a New Component Interface to the Web Dynpro Java Project..........................................................4  
Step 3 — Add the Newly-Created Component Interface to the Implemented Interfaces of the PONotifyComp Component ........................................................................................................................................7  
Step 4 — Implement the Local Component Interface of the PONotifyComp Component ...........................................8  
Step 5 — Add Your Newly-Created Component to a New Public Part ......................................................................10  
Step 6 — Add the Required Java Code to Fire the Complete Event to the Complete Method ..................................11  
Step 7 — Adapt the PONotifyCompView View to call the Complete() Method ......................................................13  
Step 8 — Deploy and Run Your Modified View.....................................................................................................15  
Step 9 — Add your Web Dynpro Java Component as a Dependency to Your Process Composer Development Component..............................................................................................................16  
Step 10 — Add the Human Activity to View the Notification to Your Process..........................................................18  
Step 11 — Rebuild the Project and Deploy Your Process ........................................................................................23  
Step 12 — Test Your Modified Process ..................................................................................................................25  
  Troubleshooting — If Your Modified Process Is Not Working Correctly .............................................................27  
Related Content.........................................................................................................................................................28  
Copyright.................................................................................................................................................................29
Step 1 — Locate the Web Dynpro Java Project and Run the Existing View

1. In NetWeaver Developer Studio (NWDS), open the Web Dynpro perspective and locate the bpm160_excercise_wd project.

2. Expand the Web Dynpro tree and take note of the PONotifyApp application and PONotifyComp component.

3. Right-click on the PONotifyApp application and select “Run”.

You should now see your Web Dynpro user interface in your browser window. The form data is empty and the confirm button does not respond to clicks because there is no context data and the event handler for confirm has not been implemented.
Step 2 — Create a New Component Interface to the Web Dynpro Java Project

1. Create a new Component Interface definition and name it **Notify**

2. Expand **Notify** and double click on Interface Controller to open it – You should see an empty Context

3. Select the Events tab and add events, **Complete** and **Error** (case sensitive!)
4. Select the Methods tab and add a new method named \textit{complete} (case sensitive!) with return type \texttt{void}.

5. Select the Context tab and right-mouse click on Context and select \texttt{New -> Node…}

6. In the wizard that opens, select \texttt{Create with Manually} and name it \texttt{PurchaseOrder}.
7. In the **Properties tab** for the **PurchaseOrder node**, change the **Collection Cardinality** to **1..1**

8. Right-mouse click on **PurchaseOrder** and create manually the following attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>orderID</td>
<td>string</td>
</tr>
<tr>
<td>orderComment</td>
<td>string</td>
</tr>
<tr>
<td>orderDate</td>
<td>date</td>
</tr>
<tr>
<td>orderTotalAmount</td>
<td>decimal</td>
</tr>
</tbody>
</table>

9. Save your work. **File -> Save All**

You should now have a new interface definition with a populated context, method, and events.
Step 3 — Add the Newly-Created Component Interface to the Implemented Interfaces of the PONotifyComp Component

1. Expand the Components tree and the PONotifyComp tree and right-mouse click on **Implemented Interfaces** and select **Add**.

2. In the pop-up window, under the Implement Component Interfaces section, click **Add**... and then select the **Notify** component interface that you previously created and click **OK**.

3. Note the status at the top of the wizard window. If you have performed the steps correctly, you should have an informational message. Double click on the message to review details if you need to make corrections.

4. Click **Finish** on the wizard window and you should see an informational pop-up window displaying your newly created objects. Click **OK**.

You should now see **Notify** under the Implemented Interfaces node of **PONotifyComp** in the Web Dynpro Explorer.
Step 4 — Implement the Local Component Interface of the PONotifyComp Component

1. Expand the Components tree, the PONotifyComp tree and right-mouse click on Component Controller and select Open -> Controller Editor

2. Right-mouse click on PurchaseOrder and select Copy

3. Under node Local Component Interface, right-mouse click on Interface Controller and select Open -> Controller Editor

4. Notice that the Context is empty. Right-mouse click on Context and select Paste
5. Select the Methods tab and add a new method (new…) named `complete` with return-type `void`.

6. Select the Events tab and add events `Complete` and `Error`.

7. Save your work. File -> Save All.

You should now have a local component interface that is completed and is ready to be exposed to other components.
Step 5 — Add Your Newly-Created Component to a New Public Part

1. Right-mouse click on the PONotifyComp component and select Add to Public Part

2. In the pop-up window, select the New... button to create a new public part and name it API and then click Finish

3. Select API and click Finish

4. Save your work. File -> Save All

Your Web Dynpro Java project now has a public part named API which exposes the context, method, and events that you previously defined.
Step 6 — Add the Required Java Code to Fire the Complete Event to the Complete Method

1. Under PONotifyComp, right-mouse click on Component Controller and select Open -> Controller Editor. You should now see the PONotifyComp editor on the right.

2. Select the Methods tab of the PONotifyComp editor.

3. Navigate to the implementation of the complete method by right-mouse clicking on complete within the Methods table and selecting Navigate to -> Implementation.

4. You should now see Java code for the complete() method. All code entered should be place between the special markers //@@begin and //@@end.

5. Insert a new line between the //@@begin complete() line and //@@end line by placing your cursor at the end of complete() and press return.
6. Enter code to fire the Complete event. Enter `wdThis` and pause a moment and then you will see the code completion window. Now press `w` and select `wdFireEventComplete()` now enter a semi-colon (`;`) to complete the statement.

   ```java
   wdThis.wdFireEventComplete();
   ```

7. Save your work. File -> Save All

You now have a `complete()` method that will fire the Complete event. You could also implement the Error method, but due to time constraints this is not done here.
Step 7 — Adapt the PONotifyCompView View to call the Complete() Method

1. Open the PONotifyCompView view by double clicking on PONotifyCompView under the Views node.

2. Click on Confirm (either the button on the view editor or the text under the RootElement in the outline) and scroll down to Events -> onAction in the Properties editor. Now click on Go to navigate to the implementation.

3. You should now be in the Java editor at the onActionConfirm method.

4. At the end of the line that starts with //@begin press return to open up a new line. You will be prompted whether or not you wish to edit the file. Click on Yes.
5. Enter the code to call the `complete()` method on the `PONotifyComp` controller. You should use code completion as you did before. Enter `wdThis` and pause, then enter `wdGet` and then select `wdGetPONotifyCompController()` and now enter `.` and select the `complete()` method. End your statement with a semicolon;

    Your line should read:

    ```
    wdThis.wdGetPONotifyCompController().complete();
    ```

6. Save your work. File -> Save All

You now have adapted your view to call the `complete()` method which fires the Complete event when you click on the button Confirm.
Step 8 — Deploy and Run Your Modified View


2. Enter the user credentials and press OK. (bpm160user/welcome123)

3. You should now see a confirmation dialog. Click on OK.

Your view should once again show up in your browser. Close the window to continue.
Step 9 — Add your Web Dynpro Java Component as a Dependency to Your Process Composer Development Component

1. Change to the Process Composer perspective

2. Open the InvestmentApprovalProcess process in the bpm160exercise project.

3. Open the development component properties of the process project by right-clicking on bpm160exercise and selecting Development Component -> Shown In -> Component Properties.

4. Double click on the tab Component Properties to expand the editor. Now select the Dependencies tab and click on Add...
5. Type `bpm` to filter the projects, and check `bpm160_exercise_wd`.

6. Check the Dependency Details for Design Time, Deploy Time, and Run Time, and then click Finish.

7. Double click on the tab **Component Properties** to restore the editor back to its original size.

Your WD Java component is now ready to be used within Process Composer.
Step 10 — Add the Human Activity to View the Notification to Your Process

1. Change to the Process Composer perspective (if not already here).

2. Open the InvestmentApprovalProcess process in the bpm160exercise project.

3. Delete the line from the Create Purchase Order activity to the End event.

4. Using the speed button on the activity Create Purchase Order, create a new human activity in the Purchase Requester lane. This will be for viewing the notification. Name it View Notification.
5. Using the speed button, connect the View Notification activity to the End event.

6. Double click on the View Notification activity and select **Task Properties** within the Properties pane, and then select **New...** to create a new task.

7. Name the new task **View Notification** and click **Finish**.
8. Click on the **Task** link to open the task editor.

9. Click on the **Choose...** button to assign the WD Java user interface.

10. Select the Web Dynpro Java development component `bpm160_exercise_wd` and the public part `API`, and then click **Next**.

11. Select the `PONotifyComp` component and the interface view `PONotifyCompInterfaceView` and then click **Next**.
12. Assign **Complete** to the completion event.

13. Assign user ‘**BPM160user**’ to this task in task editor. Detailed descriptions please refer to exercise B step 16 and 17.

14. Select the **Input Mapping** tab of the Properties and expand the **TaskInput** node.
15. Right mouse click on **PurchaseOrder** on the left and drag to **PurchaseOrder** on the right. When you release, select **Map automatically**

16. Save your work. File -> Save All

You now have completed the configuration of the task associated with the human activity for viewing the notification.
Step 11 — Rebuild the Project and Deploy Your Process

1. Right-mouse click on the `bpm160exercise` project and select **Development Component -> Build…**

2. Select only the `bpm160exercise` development component and click **OK**.
3. Once the build is complete, deploy the process by right-mouse clicking on `bpm160exercise` and selecting Development Component -> Deploy...

4. Uncheck all checkboxes except the `bpm160exercise` development component and then click OK.

5. Enter your username and password if prompted (bpm160user / welcome123)

You now have completed deployed your process and are ready to test it.
Step 12 — Test Your Modified Process

1. Log into NetWeaver Administrator and then select **Process Repository**.

2. In the Component section, select your process, **bpm160exercise**. In the Component version section make sure that the correct deployment date and time is selected. Finally in the lower section, make sure the **Process Definition** is selected and click on **Start Process**.

3. In the pop-up window, select the line with **Empty** (no parameters for the Start event), and then click **Start Process**.

4. In the lower section of your browser window, you should see a success message.
5. Now, log into the SAP NetWeaver Portal and your work list should contain a task for Enter Purchase Request.

6. Be sure to refresh your work list after completed tasks.

7. Continue as you did in the previous exercise and make sure your Notification human activity executes.

8. You should now see your new activity, View Notification.

9. And when you open your task, you should see the Web Dynpro Java screen with the data populated. Now you will see that the order number 2000000053 is created in the database.

Congratulations! You now have modified your process to consume a new Web Dynpro Java user interface.
Troubleshooting — If Your Modified Process Is Not Working Correctly

1. If your process is not working, you can review the log viewer in NetWeaver Administrator to investigate the cause.

2. Select Default Trace (Java) from Show View -> General

3. You can filter the messages by entering text into the while line under the column heading Message. The Details of an error can be expanded by click on the arrow icon under the Details column.

You now have completed deployed your process and are ready to test it.
Related Content

Building a Composite Business Process from Scratch with SAP NetWeaver BPM - Guide 1
Building a Composite Business Process from Scratch with SAP NetWeaver BPM - Guide 3

For more information, visit the Business Process Modeling homepage.
Copyright

© 2008 SAP AG. All rights reserved.

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, System i, System i5, System p, System p5, System x, System z, System z9, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, Informix, i5/OS, POWER, POWER5, 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.

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.