Using Inter-Portlet Communication with IBM WebSphere Portlet Factory

June, 2007

This article with the accompanying sample shows you how to use IBM® WebSphere® Portlet Factory Version 6 (hereafter called Portlet Factory) to implement inter-portlet communication including Click-to-Action, Property Broker, Portlet Factory events, and shared variables.

This article is one in a collection of articles and samples that illustrate techniques for developing with Portlet Factory. See the Portlet Factory Product Documentation page for a complete list of these. For an introduction to developing with Portlet Factory, you may want to look at the introductory tutorials that are available both in the product help and on that web site.

Prerequisites

You should have a basic familiarity with Portlet Factory and be able to create and run Portlet Factory portlets in WebSphere Portal.

Introduction to inter-portlet communication

There are three general types of inter-portlet communication supported by Portlet Factory builders:

1. **Portlet Factory events**. These events are a simple way to implement inter-model communication, both in Portal and when running standalone. These events work in both IBM Portlet API mode and Standard portlet API mode. When an event is fired, it is broadcast to all models in the same user session, and any models that are listening for the event will have their event handler called. Events can have any number of arguments, of simple or complex types. These events are fired explicitly and don't have any user interface of their own. In the sample, the events are fired when a link is clicked in a column of the table. The builders used for Portlet Factory events are Event Declaration and Event Handler.
2. **Cooperative Portlet events: Click-to-Action and Property Broker.** These are mechanisms defined by WebSphere Portal for inter-portlet communication, and these mechanisms are only available when running in Portal. Click-to-Action is a Portal mechanism that uses drop-down menus to control portlet interaction, and is available only when running a project in IBM Portlet mode. In Standard portlet mode, Cooperative Portlets must be configured using the "Wires" tool described below. The builders used for Cooperative Portlet events are Cooperative Portlet Source, Cooperative Portlet Target, and Event Handler.

3. **Shared variables**. This is a Portlet Factory feature that lets you share a variable across all the models in a user session. This feature works both in Portal (either portlet mode) and when running standalone. The models do not have to be on the same portal page. The builder used for this is Shared Variable. This feature is often coupled with one of the event mechanisms above, so that a when portlet modifies a shared variable it can notify other portlets that a value has changed.

There are four types of inter-portlet events available in Portlet Factory: Click-to-Action, Property Broker Link, Property Broker Action, and Portlet Factory events. The key differences between these event types are summarized in the table below. There are also differences in behavior depending on whether portlets are using IBM Portlet mode or Standard mode (JSR 168).

Table 1. Summary of key features of the four event mechanisms

	Click-to-Action	**Property Broker Link**	**Property Broker Action**	**Portlet Factory Events**
Platform support	Portal only, IBM Portlet mode only*	Portal only (either mode)	Portal only (either mode)	Portal (either mode) and standalone J2EE supported
User interface	C2A drop-down menu	Link in Standard portlet mode; C2A drop-down menu in IBM Portlet mode	No UI (event must be explicitly fired by application)	No UI (event must be explicitly fired by application)
Arguments supported	Simple argument value	Simple argument value	Simple argument value	Multiple complex argument types
Event Configuration	No wiring required – matches event names and types	Requires explicit Portlet Wiring in Standard portlet mode	Requires explicit Portlet Wiring in Standard portlet mode	No wiring required – event broadcast using the event name

* Note: Click-to-Action menus are displayed only in IBM Portlet mode, but the same models can be configured with Portlet Wiring in Standard portlet mode and will display a simple link UI instead of a drop-down menu.

Sample description

Here are some of the techniques illustrated in the sample code.

Click-to-Action and Property Broker events. For both of these event types, the source models uses the Cooperative Portlet Source builder, and the target model uses Cooperative Portlet Target. The `ORDER_ID` from the source table is used as the argument to pass to the target model. In the target model, the Event Handler builder is used to

save the ORDER_ID value into a variable and then call the method that updates and displays the details data. For the Click-to-Action portlets, the Output Name and Output Type inputs in the source model must match the Input Name and Input Type Name builder inputs in the target model.

Portlet Factory events. The GetOrderDetails event is defined in a common base model called OrdersEventDeclarationBase. This event has one argument which is the ORDER_ID. The source model has an Action List called selectRowFireEvent that fires the event, using the ORDER_ID value from the SelectedRowData variable. The target model has an Event Handler builder that listens for the event.

Using shared variables. The SharedVariableDateFilter and SharedVariableOrderList use two shared variables that are defined in the SharedVariableBase model. The variables are set when you change a date in the SharedVariableDateFilter model. In the SharedVariableOrderList model, the variables are used to filter the list of orders. This is done every time the page is rendered, using an OnPageLoad event handler. Another technique commonly used is to explicitly fire an event whenever a shared variable value changes.

Using Model Container as a simple container for standalone testing of events and shared variables. For inter-portlet communication that doesn't use Portal-specific functionality such as Cooperative Portlets, it may be convenient to create a simple container model, then use the Model Container builder to place models on the same container page for testing. See the OrdersContainmentViewer and SharedVariableContainmentViewer models for examples of this technique.

Using View & Form builder for a details page. For the portlets that show a details page (responding to an event that determines which data to display), the details page is created with a View & Form builder. Since the operation selected for viewing returns a single record as opposed to a repeating table, the page is automatically generated to display a single record. These details models have a special page which is initially displayed that tells the user to click an item to see the data. The main method, which is usually generated by the View & Form builder, is disabled in the Advanced section of that builder.

Figure 1. This screen capture shows the running Property Broker sample page in Standard portlet mode

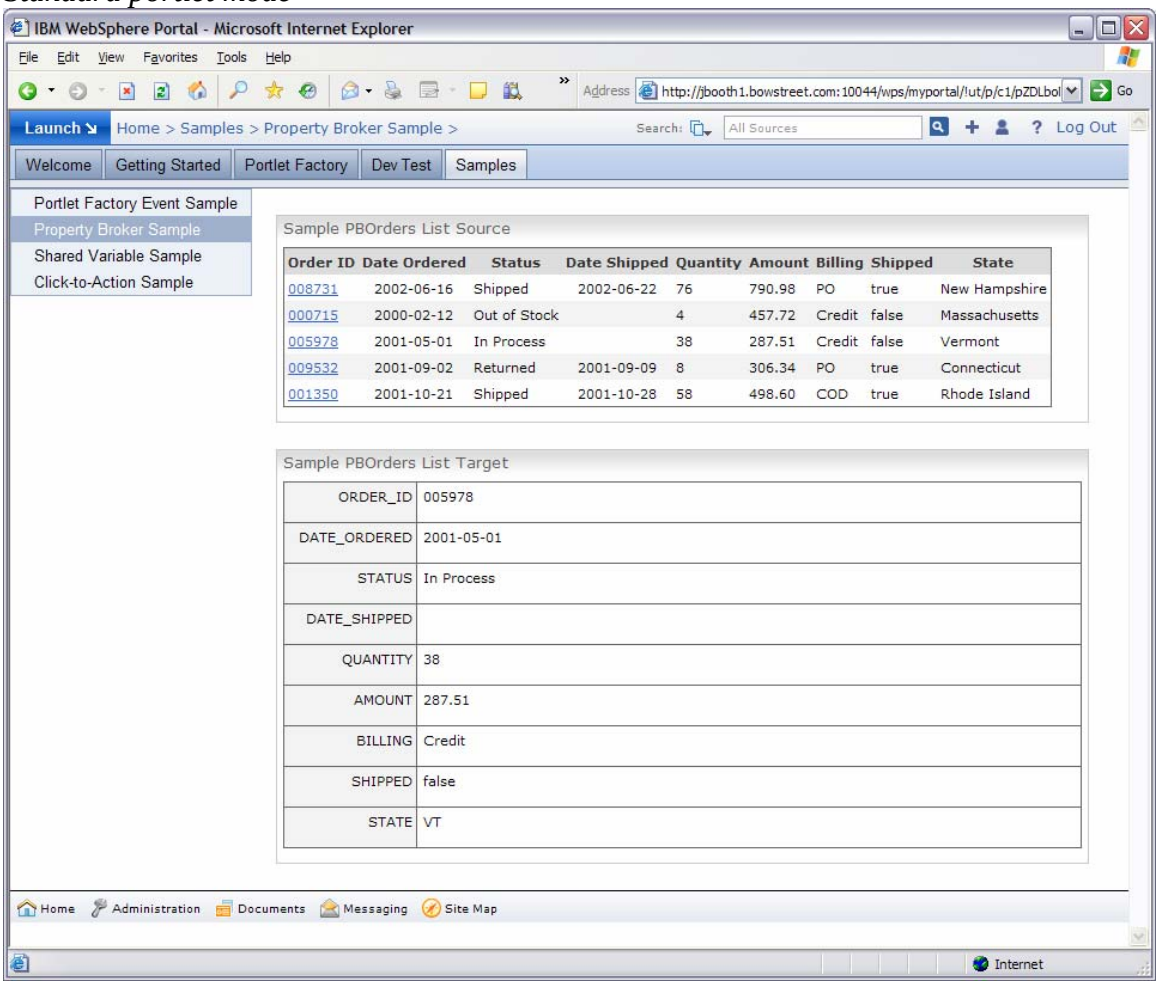

Figure 2. This screen capture shows the Click-to-Action portlets running in Portal using IBM Portlet mode

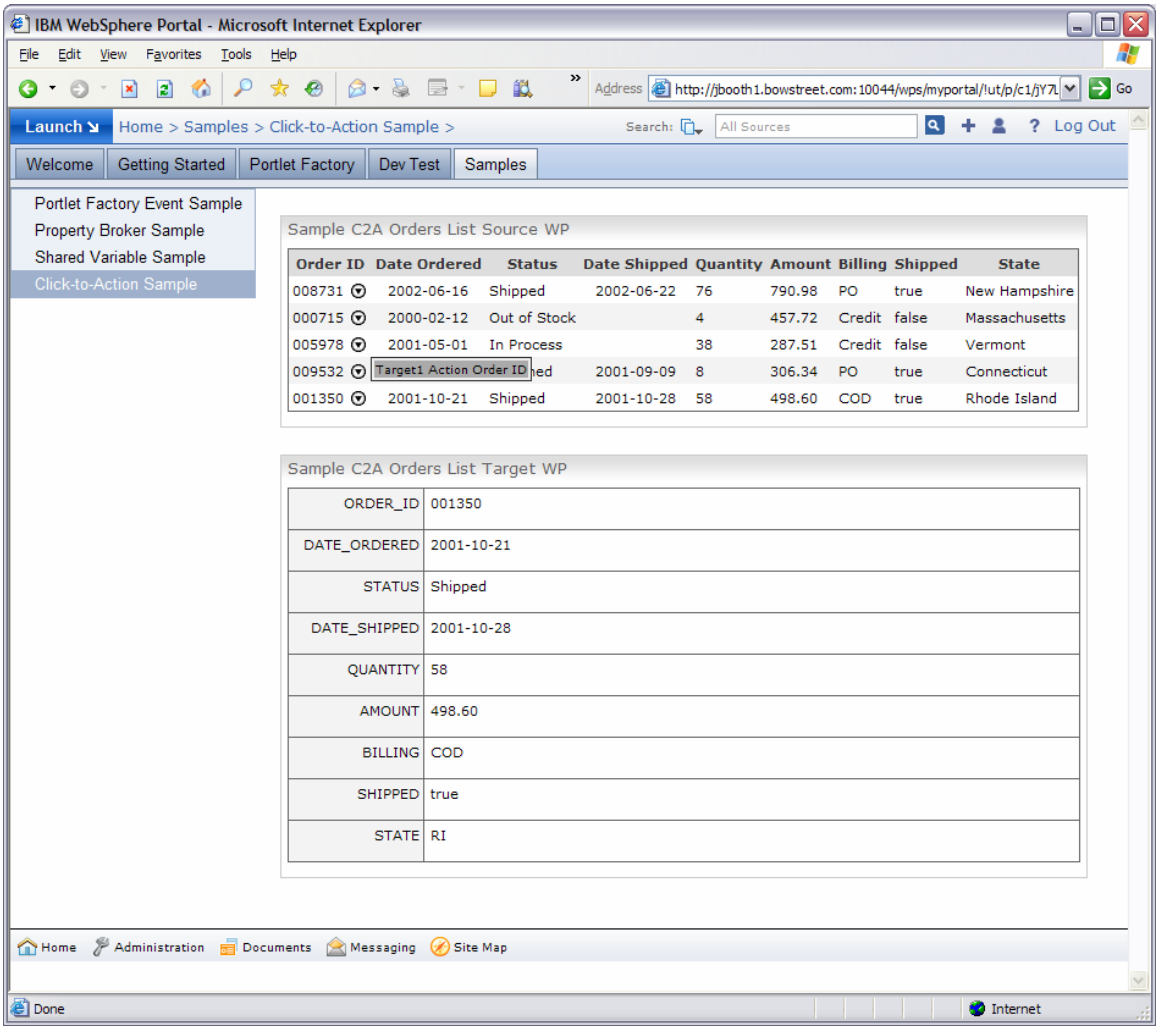

Table 1. Sample package contents

Filename and location	**Description**
`WEB-INF/models/samples/portlet_communication/OrdersSelect.model`	Source portlet for Portlet Factory event sample.
`WEB-INF/models/samples/portlet_communication/OrdersView.model`	Target portlet for Portlet Factory event sample.
`WEB-INF/models/samples/portlet_communication/OrdersContainmentViewer.model`	Container model for the previous two models, for standalone testing.
`WEB-INF/models/samples/portlet_communication/C2AOrdersListSource.model`	Source portlet for Click-to-Action sample.
`WEB-INF/models/samples/portlet_communication/C2AOrdersListTarget.model`	Target portlet for Click-to-Action sample.

WEB-INF/models/samples/ portlet_communication/ PBActionOrdersSource.model	Source portlet using "Property Broker Action" type for Property Broker sample.
WEB-INF/models/samples/ portlet_communication/ PBOrdersListSource.model	Source portlet for Property Broker sample.
WEB-INF/models/samples/ portlet_communication/ PBOrdersListTarget.model	Target portlet for Property Broker sample.
WEB-INF/models/samples/ portlet_communication/ SharedVariableDateFilter.model	Part of the shared variable sample. This model has a page for setting start and end dates used to filter data in the following model.
WEB-INF/models/samples/ portlet_communication/ SharedVariableOrderList.model	List of orders which is filtered using a shared variable value.
WEB-INF/models/samples/ portlet_communication/ SharedVariableContainmentViewer.model	Container model for the previous two models, for standalone testing.
WEB-INF/models/samples/ BaseModels/ OrdersEventDeclarationBase.model	Event declaration that is used by both OrdersSelect and OrdersView models.
WEB-INF/models/samples/ BaseModels/ SharedVariableBase.model	Definition of the shared variable that is used by both SharedVariableDateFilter and SharedVariableOrderList models.
WEB-INF/models/samples/ data/OrdersServiceProvider.model	Service provider model used with the shared variable sample.

Instructions for running the sample

Note: There can be an issue with path names that exceed the Windows limit when deploying Cooperative Portlets in Portal. For this reason, you may want to look at the workaround below in the Troubleshooting section, and create a new project that has the workaround in place before you begin.

To run the sample application:

1. Download the sample ZIP file and import it into a project using the `File, Import, WebSphere Portlet Factory Archive` command. The project must have the `Tutorials and Samples / Applications` feature set installed and it must be enabled for portlet creation. The features and functionality are slightly different between Java Standard and WebSphere Portal Native portlets, as noted below.
2. Test the shared variable sample models and the event sample models in a standalone mode. For events, open the `OrdersContainmentViewer` model and run it. Click on one of the order ID's in the top model to see the order details in the bottom model. For shared variable, open the `SharedVariableContainmentViewer` model and run it. Try selecting a different

start date such as `June 1, 2001`. Notice that the list of orders is filtered using the shared variable data.

3. Update the portlet WAR to make the portlet models available in Portal. To do this, right-click on the project and select `Rebuild WAR / Rebuild Portlet WAR`. If you do not have auto-deploy enabled for portlets in your project, you will need to manually update the portlet WAR using the Portal's Administration tools.
4. Create portal pages for the test portlets and place the portlets on them. Create pages similar to the following, with two/three portlets on each page as shown:

Page	**Portlets**
Portlet Factory Event Sample	Sample Orders Select Sample Orders View
Property Broker Sample	Sample PBOrders List Source Sample PB Action Orders List Source Sample PBOrders List Target
Click to Action Sample	Sample C2A Orders List Source Sample C2A Orders List Target
Shared Variable Sample	Sample SharedVariable Date Filter Sample SharedVariable Orders List

5. View the Portlet Factory Event Sample page. Click on an `Order ID` in the list portlet to see the order details in the second portlet.
6. View the Shared Variable Sample page. Try selecting a different start date such as `June 1, 2001`, and notice that the second portlet updates to reflect the change.
7. If you are running WebSphere Portal Native mode, view the Click to Action Sample page, and click an `Order ID` to see the order details in the other portlet.
8. If you are using Java Standard mode, in order to run the Property Broker or Click-to-Action samples you will need to use the Portlet Wiring tool to connect the two portlets.
    a. Go to Edit Page Layout for the page and click on the **Wires** tab at the top.
    b. Select the following values:

Input	**Selection**
Source portlet	`Sample PBOrders List Source`, `Sample PB Action Orders List Source`, or `Sample C2A Orders List Source`
Sending	`Order ID`
Target Page	(leave at default value for this page)
Target Portlet	`Sample PBOrders List Target` or `Sample C2A Orders List Target`
Receiving	`Target1 Action Order ID`, `Order List Order ID`

    c. Important: Click the "plus" sign on the right to add this wiring.
    d. Click **Done**.
9. View the Property Broker Sample page and select an `Order ID` to see the order details in the other portlet.

Troubleshooting

Working around the "Path too long" error

There can be an issue with path names that exceed the Windows limit when deploying Cooperative Portlets in Portal. The issue arises with the WSDL files that are generated when you use Click-to-Action or Portlet Wiring. You can follow the steps here to make projects that have a somewhat shorter deployed folder location. After making this change, any new projects you create will have the shortened folder location, making them less likely to have the problem with long pathnames.

1. Locate this folder in your Portlet Factory installation folder:
   `WPFDesigner\FeatureSets\Web-App_<version>\`
   `Templates\Project\wpf.war\WEB-INF\bin\deployment`
2. Open these two files under that folder for editing:
   `\jsr168\web.xml`
   `\wp\web.xml`
3. In both files, change this line:
   `<display-name>WebAppRunner Portlet Application</display-name>`
   to this:
   `<display-name>WPF</display-name>`
4. Save the files.
5. Create a new project, with the `Tutorials and Samples / Applications` feature set.
6. Continue with the steps above to install and run the samples.

Resources

WebSphere Portlet Factory product documentation

http://www.ibm.com/developerworks/websphere/zones/portal/portletfactory/proddoc.html

WebSphere Portlet Factory support

http://www.ibm.com/software/genservers/portletfactory/support/

developerWorks forums

http://www.ibm.com/developerworks/forums/wsdd_forums.jsp

Trademarks