How To…

Use Portal Eventing

ENTERPRISE PORTAL 6.0

PUBLIC

ASAP "How to…" Paper

Applicable Releases: EP 6.0

September 2003

1 Scenario

The SAP Enterprise Portal 6.0 supports client-side eventing. This not only allows different Web Dynpro applications to communicate with one another but also with any iView with portal content (BSP, ITS).

Although you cannot write Java Script code in a Web Dynpro application, a Java wrapper enables you to implement client-side eventing and so use portal eventing.

You can subscribe to- and unsubscribe from specific portal events. This means that the SAP Enterprise Portal can react to an event that is triggered by a Web Dynpro application.

To use portal eventing you have to define which Web Dynpro action is used as the event handler for the portal event. You can fire any portal event.

Note that portal eventing only works properly if all participants (i.e. the portal server itself and all the servers used) are in the same domain. Otherwise portal eventing does not work. If the SAP J2EE Engine on which the Web Dynpro application is deployed is in another domain, you have to edit the configuration file `hosts` and map the IP address of the SAP J2EE Engine to the domain name of the SAP J2EE Engine on which the SAP Enterprise Portal is running.

1.1 Subscribing to a Portal Event

The following code shows the signature of the **subscribe** method:

```
WDPortalEventing.subscribe(java.lang.String nameSpace, java.lang.String
event, IWDAction action);
```

for example

```
WDPortalEventing.subscribe (“urn:com.sap.tc.webdynpro.test.portal”,
                            “TestEvent”,
                            wdThis.wdGetTestEventAction());
```

You have to define the event’s `name` and its `namespace`. The combination of these two names must be unique.

The third parameter is the Web Dynpro `action`, which should be mapped to the portal event. The action event handler is called if the Web Dynpro application receives the specified portal event on the client side. The Client Side Framework (CSF) handles the mapping between a portal event and a Web Dynpro action and it is completely transparent for the Web Dynpro application developer.

You can reuse a Web Dynpro action for several portal events. If you want to receive the portal event’s data you have to define the following parameters for your Web Dynpro action:

- `dataObject`
  The `dataObject` parameter contains the portal event parameter.
- `nameSpace`
  The `namespace` parameter contains the portal event’s namespace.
- `name`
  The `name` parameter contains the portal event’s name.

It is useful to add the `nameSpace` and `name` parameters to the Web Dynpro action if the action is reused for several portal events because you can use this information to differentiate between the portal events.

Note: In the current version a portal event subscription is valid for a Web Dynpro view. Therefore you should add the required Java coding, for example in the `wdDoInit`() method of the generated view class. If you navigate between different views, you have to subscribe to every view for the portal event required.

1.2 Unsubscribe from a portal event

Unsubscribing from a portal event is very similar to subscribing. The following code shows the signature of the `unsubscribe` method:

```
WDPortalEventing.unsubscribe(java.lang.String nameSpace, java.lang.String
event, IWDAction action);
```

for example

```
WDPortalEventing.unsubscribe (“urn:com.sap.tc.webdynpro.test.portal”,
                              “TestEvent”,
                              wdThis.wdGetTestEventAction());
```

Note: You must unsubscribe every Web Dynpro view, because subscription and unsubscription is valid only for the current view.

1.3 Raise a portal event

The following example shows how to raise a portal event. The signature is:

```
WDPortalEventing.fire(java.lang.String nameSpace, java.lang.String event,
java.lang.String parameter);
```

for example

```
WDPortalEventing.fire (“urn:com.sap.tc.webdynpro.test.portal”,
                       “TestEvent”,
                      “AParameter”);
```

You can fire a portal event anywhere in your Web Dynpro application. The event is sent to the client with the next response. You can also raise more than one portal event in a request-response cycle. Typically, you will fire a portal event in a Web Dynpro action event handler (for example, as a response to pressing a button).

The following step-by-step example shows how to carry out portal eventing within two simple Web Dynpro example applications.

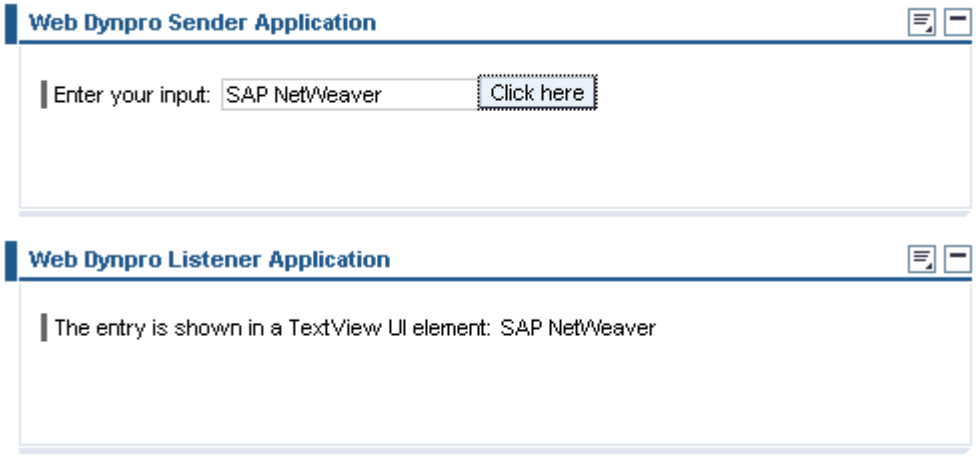
2 Result

A page preview will demonstrate portal eventing.

3 The Step By Step Solution

3.1 Example Description

The user can enter an arbitrary string in an input field. If the user press the pushbutton `click here` in the sender view, the input string will be displayed in a TextView UI element of the listener application.



3.2 Prerequisites

You have built two Web Dynpro applications. How to built Web Dynpro applications is described in the How-To Guide **Creating a Simple Web Dynpro Application**. In each application you have created a Web Component with a view that uses the portal eventing.

You can find the detailed procedure describing how to insert UI elements into a view in the How-To Guide **Creating a Simple Web Dynpro Application**.

3.3 Creating the Web Dynpro applications

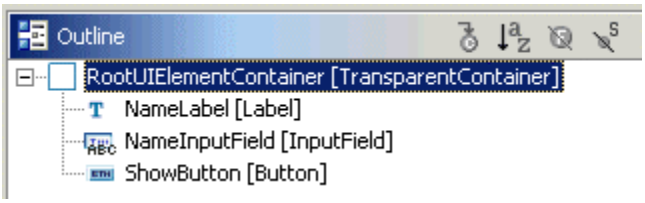
- Create the Web Dypro project. Call it **_webdynpro_example_portal_eventing**.
- Create two Web Dynpro components. Call them:
  - **EventSenderComponent**
  - **EventListenerComponent**
- Create two Web Dynpro applications because only an application can be called by the browser. Call them:
  - **EventSenderApplication**
  - **EventListenerApplication**

The How-To Guide **Creating a Simple Web Dynpro Application** describes how to create a:

- Web Dynpro project
- Web Dynpro component
- Web Dynpro application

3.4 Creating the necessary views:

3.4.1 Create the layout of the EventSenderView in the EventSenderComponent as follows:



3.4.2 Create the context structure of the EventingSender View:

The context value attribute **Name** must be of type `String`.

3.4.3 Create an action “Show”

3.4.4 Define the data binding of corresponding UI element properties:

The appropriate properties of the UI elements must be bound to the context that contains and displays the data.

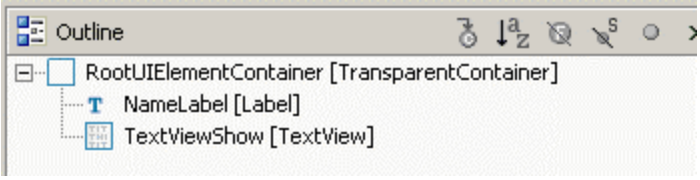
- Define a text, for instance, “Enter a text:” for the label UI element (ID: NameLabel) using the property `text`.
- Bind the property `value` of the input field (ID: NameInputField) to the context attribute `Name`.
- Bind `onAction` of the button (ID:ShowButton) to the corresponding action `Show`.

3.4.5 Implementation of the view controller

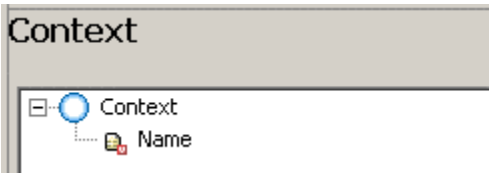
- If you create an action declaratively within the SAP NetWeaver Developer Studio, the Web Dynpro framework automatically generates an `onAction` method in the controller’s implementation. The generated method enables you to write code to fetch the input string and to fire the portal event. The fire method of *WDPortalEventing* passes the data and the name of the event that should be handled by the second application as parameters. The following code shows the implementation of `onActionShow` method:

```
public void onActionShow(com.sap.tc.webdynpro.progmodel.api.IWDCustomEvent
wdEvent )
  {
    //@@begin onActionShow(ServerEvent)
      String name = wdContext.currentContextElement().getName();

WDPortalEventing.fire("urn:com.sap.tc.webdynpro.example.portaleventing",
                                                "Show",
                                                name);
    //@@end
  }
```

3.4.6 Create the layout of the EventListenerView in the EventListenerComponent as follows:



3.4.7 Create the context structure of the EventingListenerView:



The context attribute **Name** must be of type `String`.

3.4.8 Create an action, for example, ReactPortalEventing which should be mapped to the portal event.

Create an action `ReactPortalEventing`.

Add the parameter `dataObject`. It should have the type java.lang.String.

3.4.9 Define the data binding of the corresponding UI element properties:

The corresponding properties of the UI elements must be bound to the context that contains and displays the data.

- Define a text for the `NameLabel` for example, `The entry is displayed in a TextView UI element:`
- Bind the property `text` of the TextView UI element (ID: TextViewShow) to the context attribute `Name`.

3.4.10 Implementation of the view's controller

- You have to subscribe to the event within the `wdDoInit` method. You have to define the `namespace` of the event and the `name` of the event, for example `Show`. The third parameter is the Web Dynpro `action`, which should be mapped to the portal event.
- You have to implement the action (`ReactPortalEventing`) that passes the `dataObject` and fills the context with data. The following code shows the implementation of the `wdDoInit` and the `reactPortalEventing` methods:

```
public void wdDoInit()
  {
    //@@begin wdDoInit()

WDPortalEventing.subscribe("urn:com.sap.tc.webdynpro.example.portaleventing",
"Show",
    wdThis.wdGetReactPortalEventingAction() );
    //@@end

  }
```

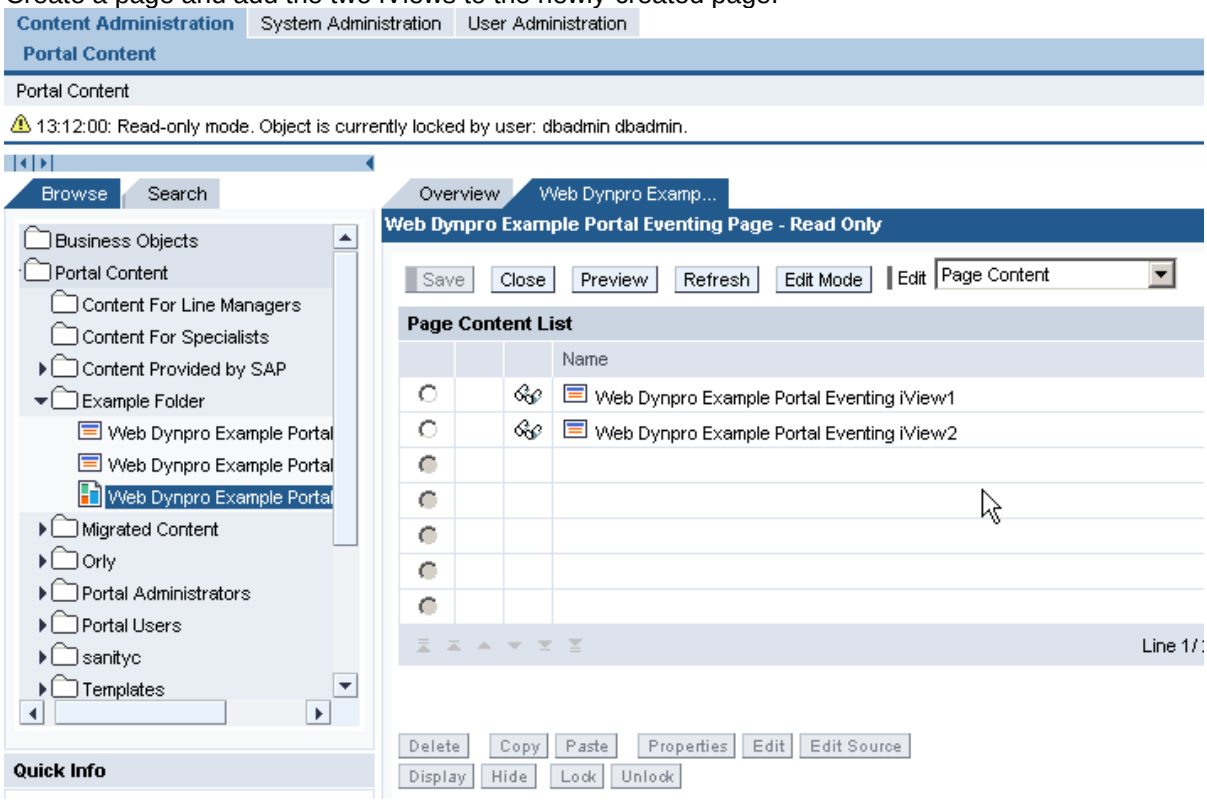
```
public void
onActionReactPortalEventing(com.sap.tc.webdynpro.progmodel.api.IWDCustomEvent
wdEvent, java.lang.String dataObject )
  {
    //@@begin onActionReactPortalEventing(ServerEvent)
    wdContext.currentContextElement().setName(dataObject);
    //@@end

  }
```

3.4.11 Deploying the Web Dynpro Application

Before you can call the Web Dynpro application, you have to built the Web Dynpro project and deploy the application on the SAP J2EE Engine.

Simple Testing of the Portal Eventing and Calling the Web Application

- Create two iViews in the newly-created example folder as described in the How-To Guide **How to Integrate a Web Dynpro Application into the SAP Enterprise Portal 6.0**.
- Create a page and add the two iViews to the newly-created page.



- To see the Web Dynpro applications communicate with each other press the button *Preview* in edit mode. Enter your input and press button *Click here*.