Accessing UWL Notifications from Web Dynpro Using UWL API

Applies to:

SAP NetWeaver 7.0 SP13 & SAP NetWeaver Developer Studio 7.0.10 (May be applicable to previous versions). For more information, visit the User Interface Technology homepage.

Summary

The objective of this tutorial is to demonstrate the use of the UWL API in external applications. Web Dynpro Java is being used for this purpose. A small application is developed that enables the user to send notifications to other portal users and also view the notifications received. The notifications received are filtered according to the application. Download the application here

Author: Mayuresh Kanvinde

Company: Tata Consultancy Services Ltd.

Created on: 15 October, 2008

Author Bio

Mayuresh Kanvinde is working in Tata Consultancy Services Ltd. as a NetWeaver Consultant.

Table of Contents

Prerequisites

- Knowledge Requirement – Basic Web Dynpro Hands-On Experience
- UWL services in the Enterprise portal have been started.
- AdHocWorkflowConnector is enabled in UWL Configurations. (Refer to Related Content for more information).

  By default the `AdHocWorkflowConnector` is enabled. If it is not, then make sure that while linking connectors to the system names, the system alias `AdHocSystem` must map to `AdHocWorkflowConnector.`

Software Component (API) Requirements

The following jar files are required. These are provided with NWDS 7.0 and are located in the folder \SAP\IDE\IDE70\eclipse\plugins.

- bc.uwl.service.api_api.jar \com.sap.netweaver.bc.uwl.plugin_1.0.0\lib
- bc.uwl.service.api_core.jar \com.sap.netweaver.bc.uwl.plugin_1.0.0\lib
- com.sap.security.api.jar \com.sap.security_2.0.0\lib

Implementation

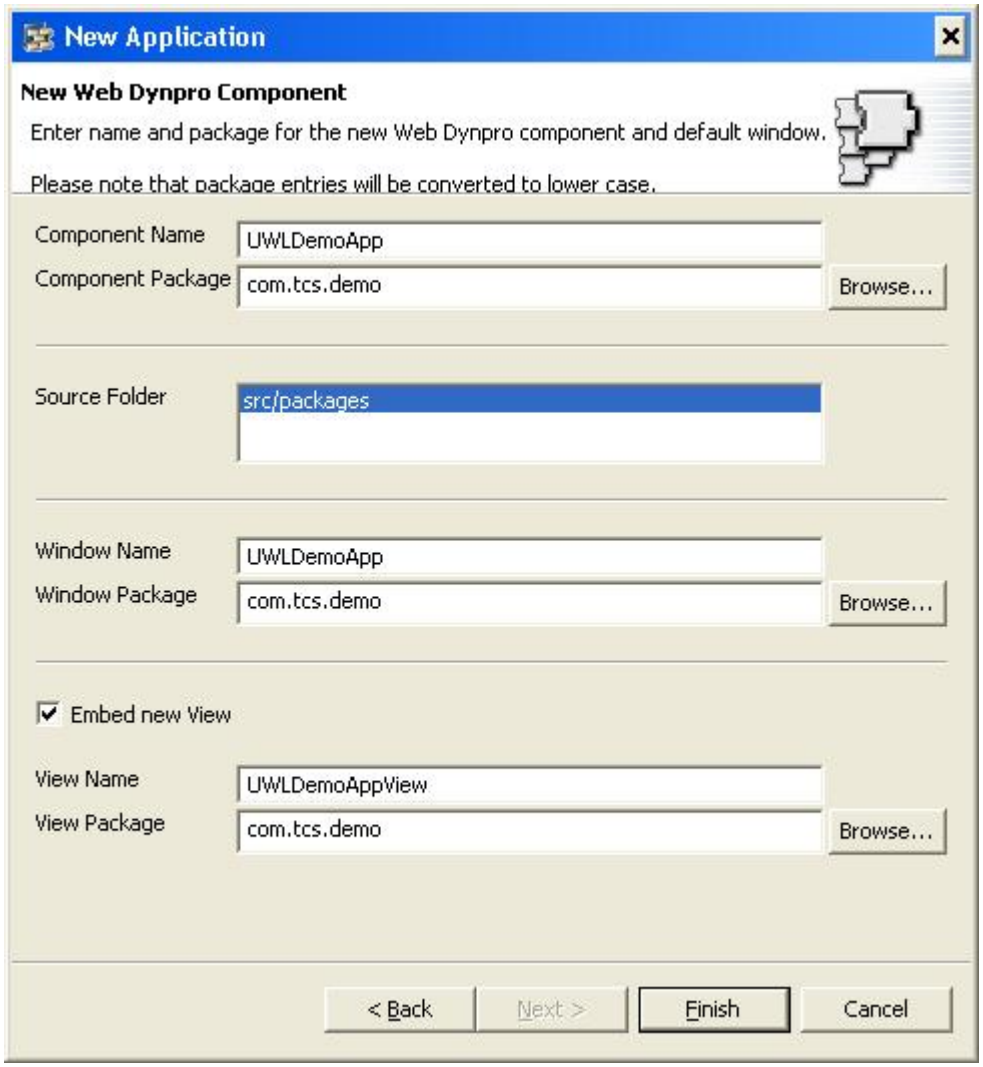
Step 1: Create Web Dynpro Project, Application and Component

Create a Web Dynpro project with the name “UWLDemo”. Create an application “UWLDemoApp”.

It would be ideal to provide authentication if the application is to be accessed without the portal. Check the “Authentication” check box for this purpose.

If the application is to be accessed from within the portal by means of an iview then authentication may not be required.

Create a component “UWLDemoApp”, a window “UWLDemoApp”, and a view “UWLDemoAppView”.



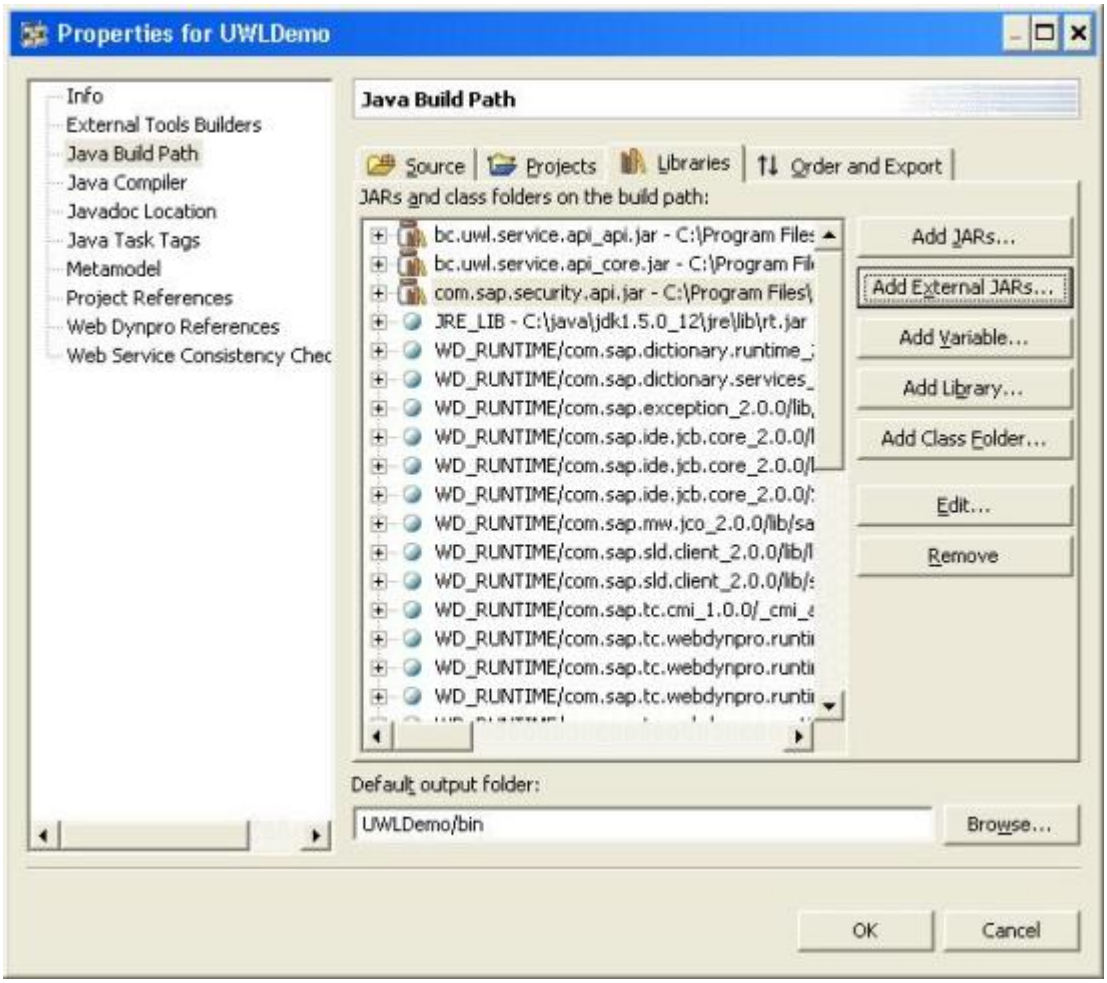
Step 2: Add Libraries and References

Right click on the project folder and select properties.

Select “Java Build Path” -> “Libraries” and click on “Add External Jars”.

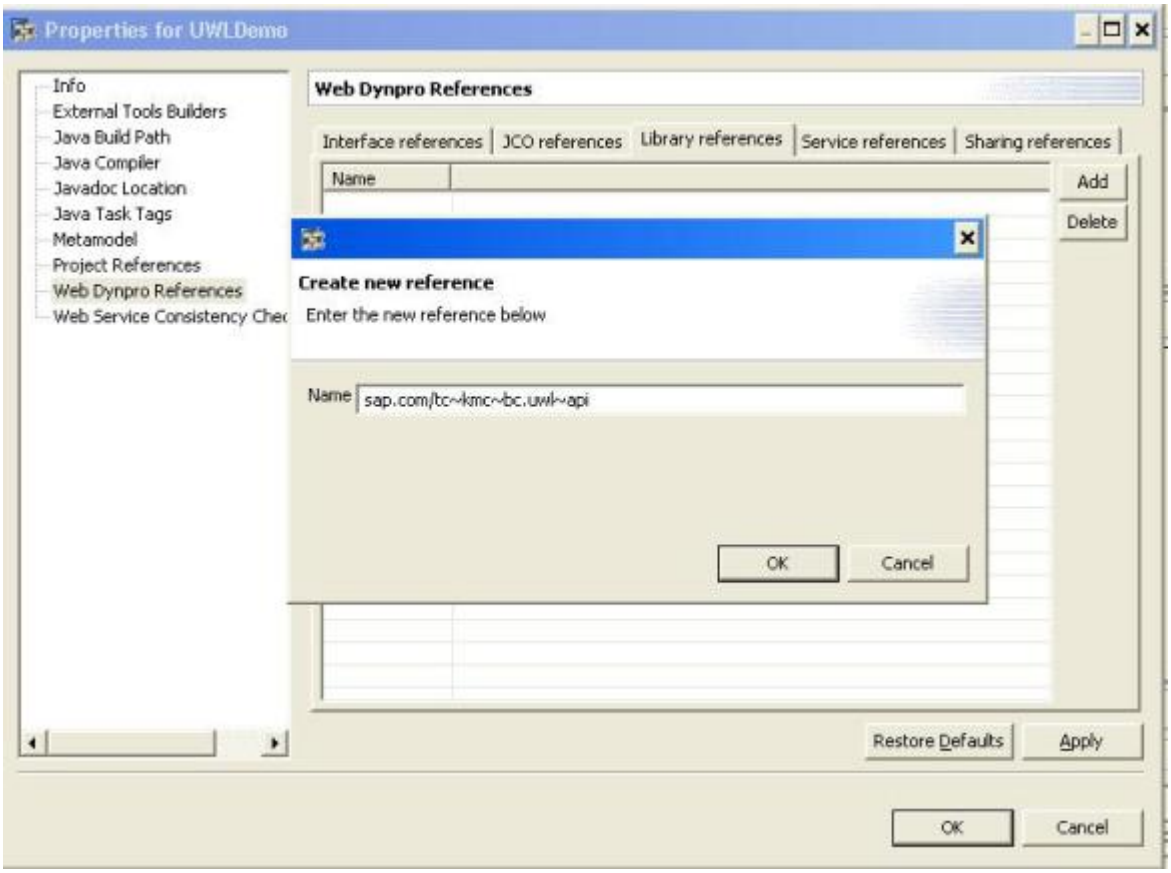
Add the following jar files: (These are located in the folder \SAP\IDE\IDE70\eclipse\plugins).

- bc.uwl.service.api_api.jar \com.sap.netweaver.bc.uwl.plugin_1.0.0\lib
- bc.uwl.service.api_core.jar \com.sap.netweaver.bc.uwl.plugin_1.0.0\lib
- com.sap.security.api.jar \com.sap.security_2.0.0\lib



Select “Web Dynpro References” -> “Library References” and click on “Add”.

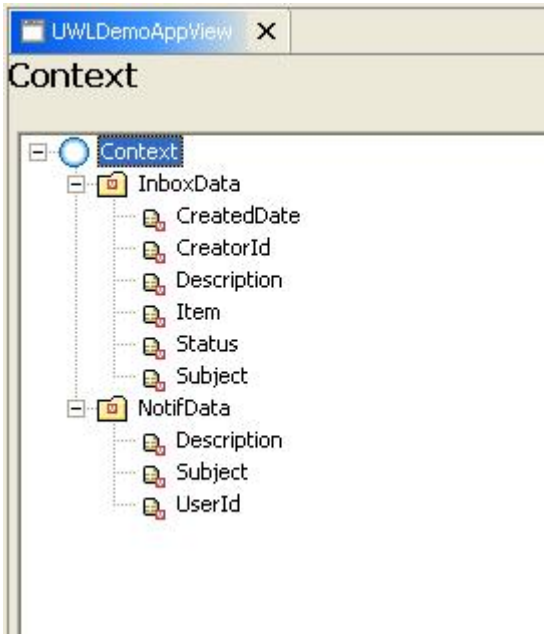
Add a reference with the name “sap.com/tc~kmc~bc.uwl~api”.



Step 3: Create Web Dynpro Context

Create context for the view “UWLDemoAppView”.

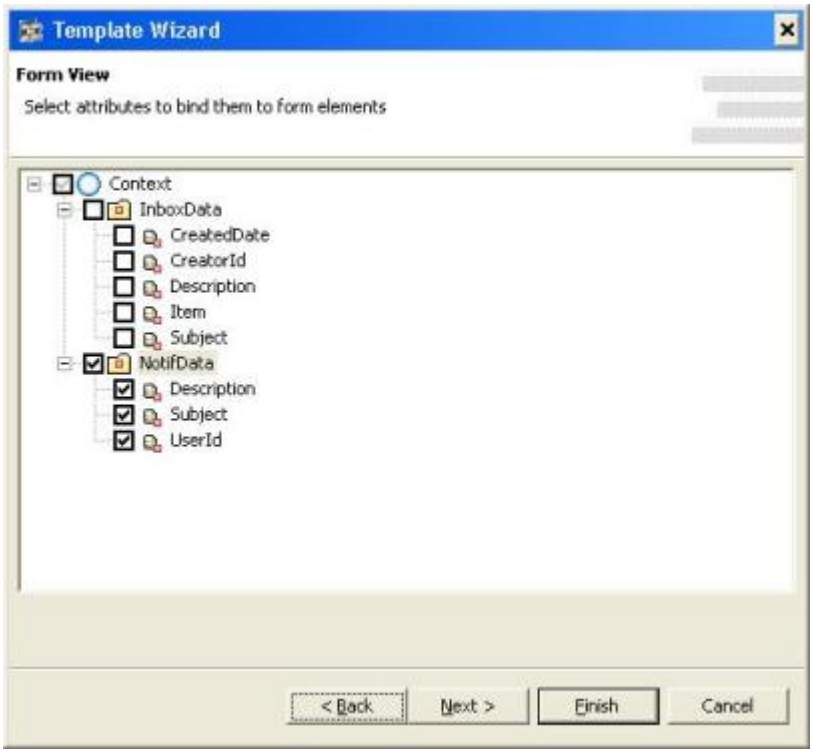
All attributes except for InboxData.Item are of type “string”. For Item, select item type as com.sap.netweaver.bc.uwl.Item from Java native types.



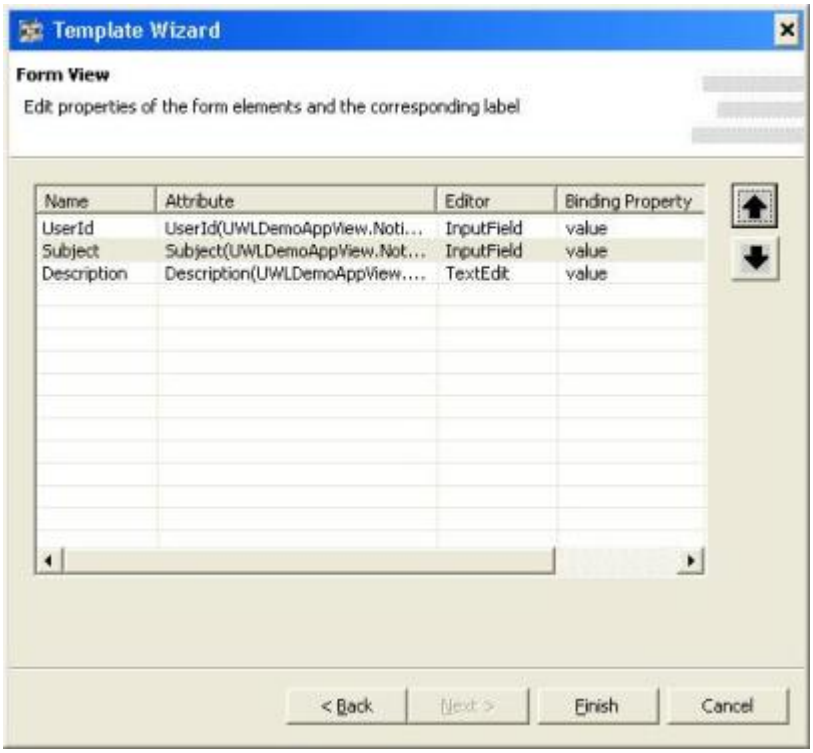
Step 4: Create User Interface

Switch to the layout of the view “UWLDemoAppView”.

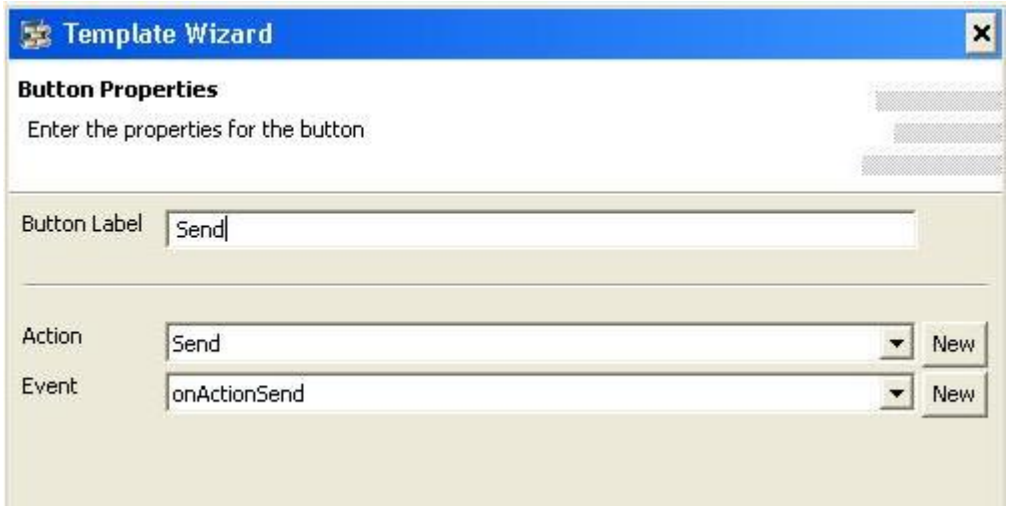
Add a Transparent Container to the “RootUIElementContainer”. Create a form using “Apply Template” with following fields.



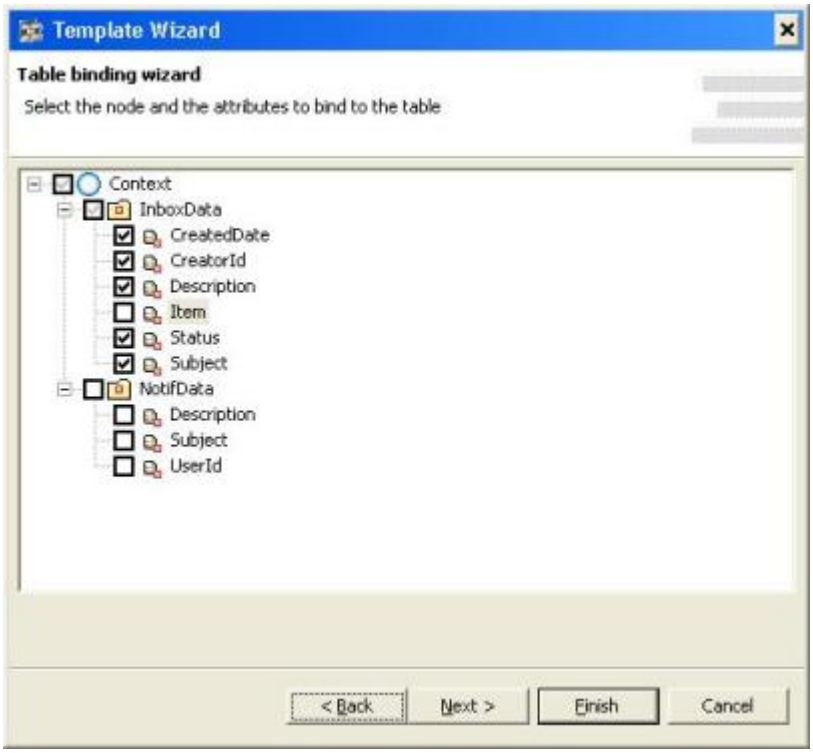
Change the Editor for “Description” Field to TextEdit.



Add a button “Send” to the transparent container. Create an action “Send” and bind it to the button.

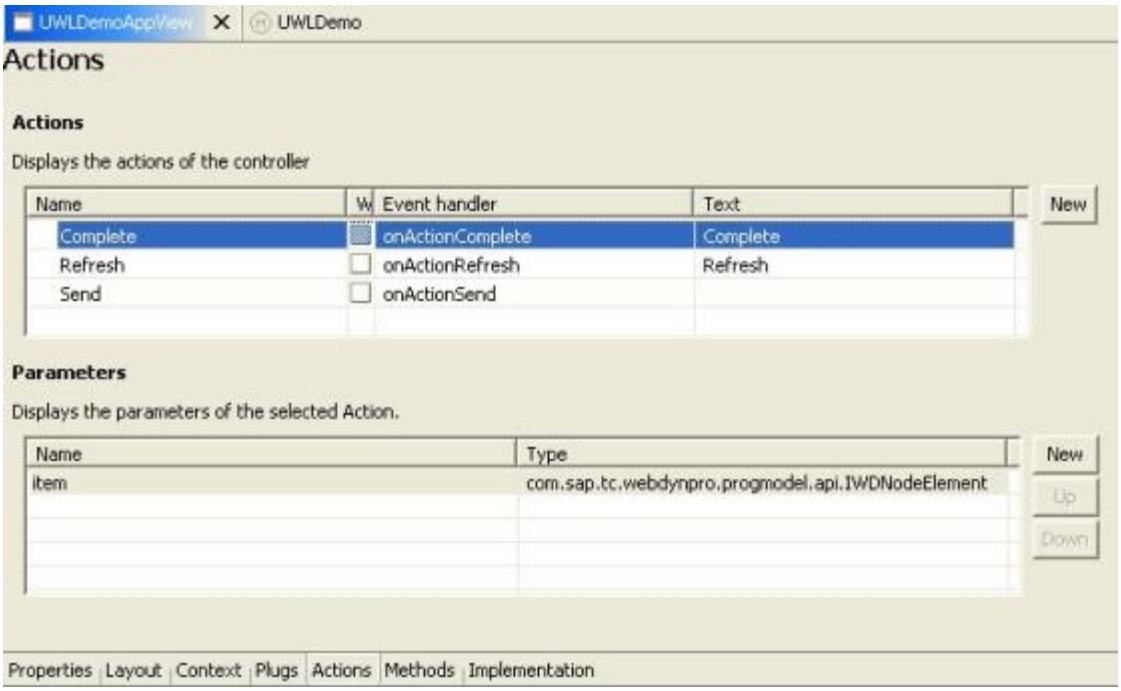


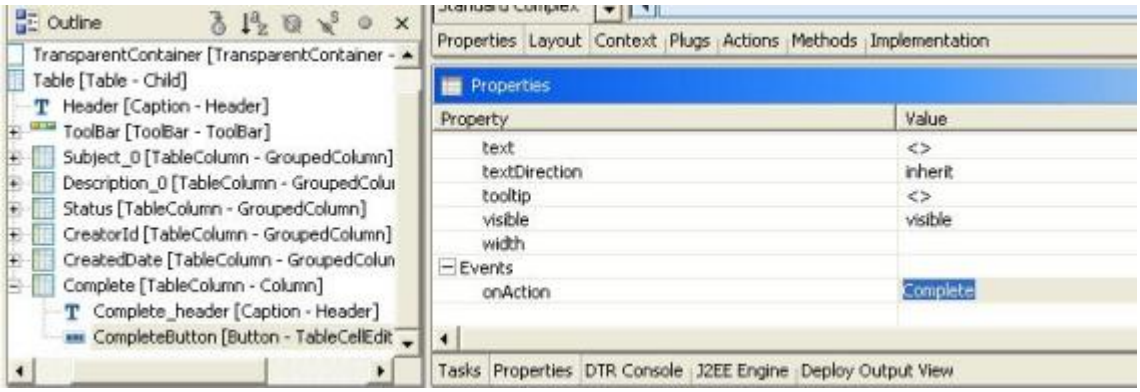
Create a Table using “Apply Template” with following columns.



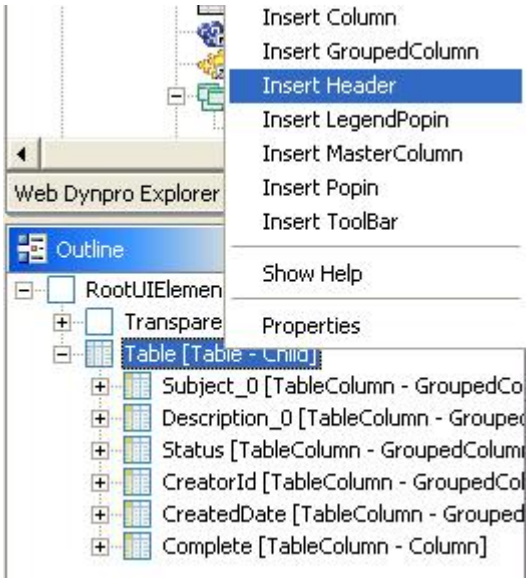
Insert another column into the table with id “Complete” and insert a Button “CompleteButton” as the column’s TableCellEditor.

Create an action “Complete” with a parameter “item” of type com.sap.tc.webdynpro.progmodel.api.IWDNodeElement and bind it to the button “CompleteButton”.

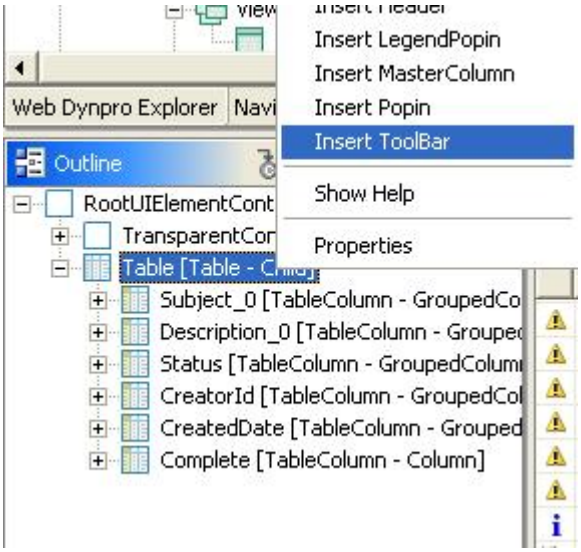




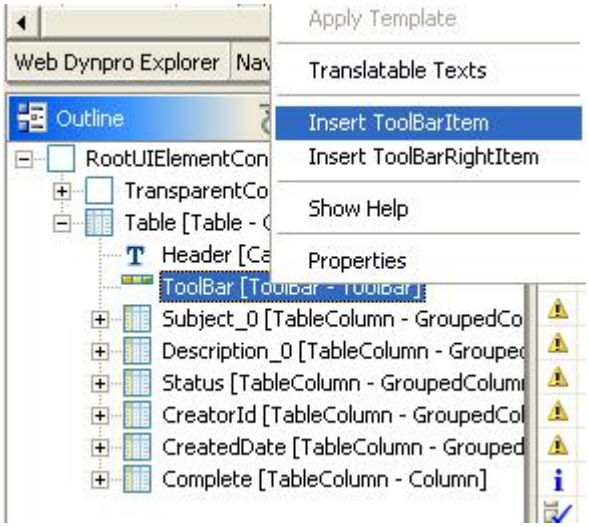
Insert a Header for the table and change its Text property to “Inbox”.

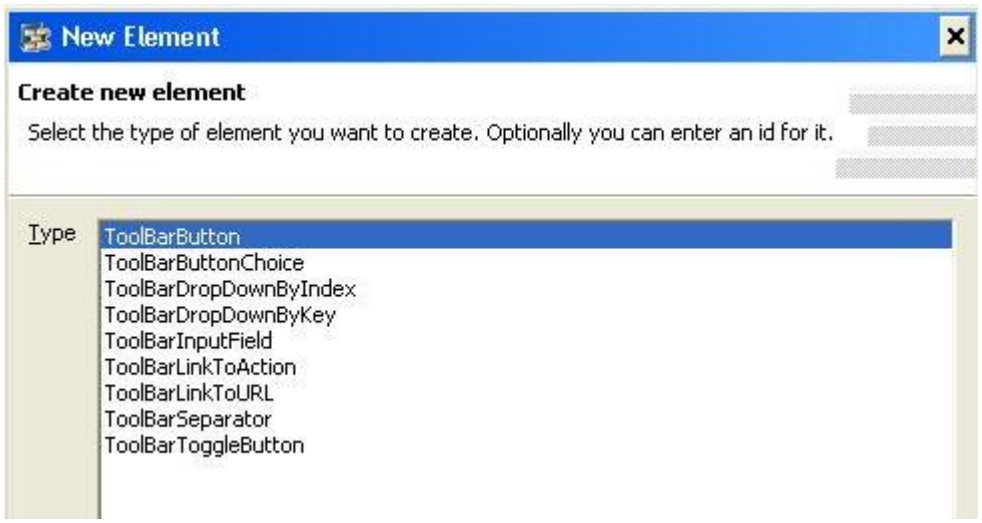


Insert a ToolBar for the table.

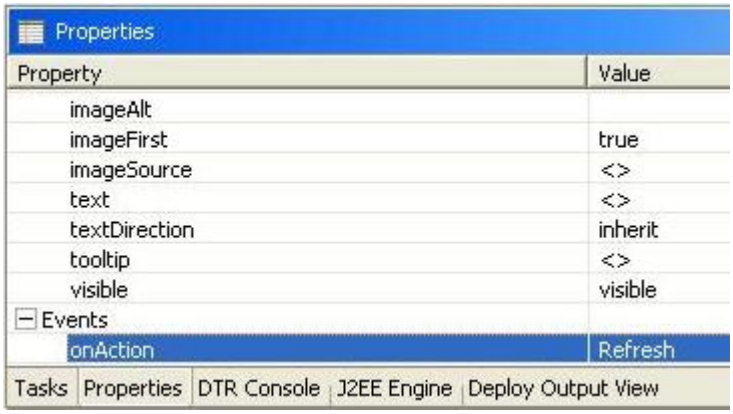


Insert a ToolBarButton “Refresh” for the table.

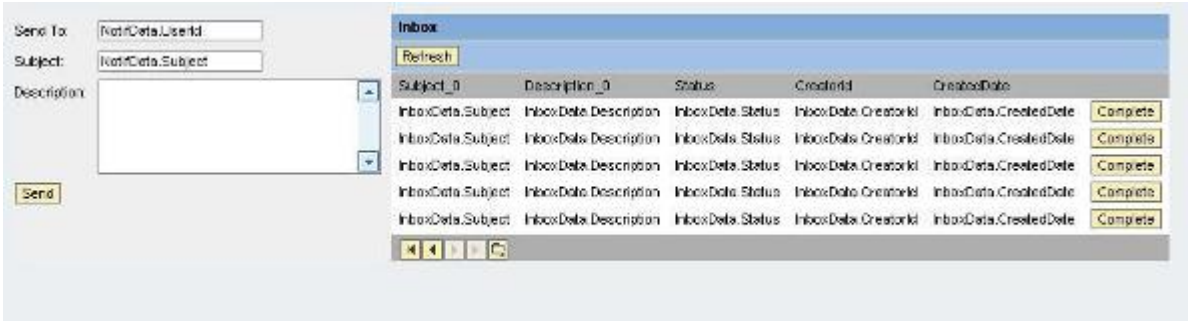




Create an Action “Refresh” and bind it to the ToolBarButton.



We get the final layout for the view as follows:



Step 5: Code

Create 3 methods in the view “UWLDemoAppView” as follows:

- getInbox ( ) with return type void and no parameters.
- sendNotification ( ) with return type void and no parameters.
- completeNotification ( ) with return type void and 1 parameter.
    - Parameter Name – item
    - Parameter Type - com.sap.tc.webdynpro.progmodel.api.IWDNodeElement

The code for these functions is as follows (It can be used in any other Java-based Application provided the necessary classes are provided using the API jar files):

```
//@@begin javadoc:getInbox()
 /** Declared method. */
//@@end
public void getInbox( )
{
  //@@begin getInbox()

        wdContext.nodeInboxData().invalidate();

        try {
               IUWLService uwlService =
                      (IUWLService) WDPortalUtils.getServiceReference(
                             IUWLService.ALIAS_KEY);
               UWLContext myContext = new UWLContext();
               IWDClientUser user;
               user = WDClientUser.getLoggedInClientUser();
               IUser loggedInUser = user.getSAPUser();

               myContext.setUser(loggedInUser);

               IUWLSession mySession =
                      uwlService.getUwlSessionForWebDynproClient(myContext);

               IUWLItemManager itemMan = uwlService.getItemManager(myContext);
               QueryResult result =
                      itemMan.getItemsForItemType(
                             myContext,
                             ItemType.UWL_ITEM_NOTIFICATION,
                             null,
                             null);
               ItemCollection coll = result.getItems();
               List l = coll.list();

               Iterator iter = l.iterator();
               while (iter.hasNext()) {
                      Item item = (Item) iter.next();

// External Type is used to filter the notifications according to application

                      if (item.getExternalType().equals("UWLNotification") &&
!item.getStatus().equals(StatusEnum.COMPLETED)) {
```

```
                        IPrivateUWLDemoAppView.IInboxDataElement data =

wdContext.nodeInboxData().createInboxDataElement();
                        data.setItem(item);
                        data.setSubject(item.getSubject());
                        data.setDescription(item.getDescription());
                        data.setStatus(item.getStatus().getText());
                        data.setCreatorId(item.getCreatorId());
                  data.setCreatedDate(item.getCreatedDate().toString());
                        wdContext.nodeInboxData().addElement(data);

                  }
            }

            uwlService.endSession(myContext);
      } catch (UWLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            wdComponentAPI.getMessageManager().reportSuccess(e.getMessage());

      }

  //@@end
}

//@@begin javadoc:sendNotification()
 /** Declared method. */
//@@end
public void sendNotification( )
{
  //@@begin sendNotification()

      try {
            IUWLService uwlService =
                  (IUWLService) WDPortalUtils.getServiceReference(
                        IUWLService.ALIAS_KEY);
            UWLContext myContext = new UWLContext();
            IWDClientUser user;
            user = WDClientUser.getLoggedInClientUser();
            IUser loggedInUser = user.getSAPUser();

            myContext.setUser(loggedInUser);

            IUWLSession mySession =
                  uwlService.getUwlSessionForWebDynproClient(myContext);

            IUWLItemManager itemMan = uwlService.getItemManager(myContext);

            IPushChannel push = uwlService.getPushChannel();
            IProviderConnector procon =
                  uwlService.getRegisterProviderConnector(
                        IProviderConnector.ADHOC_WORKFLOW_CONNECTOR_ID);
            IUserFactory usf = UMFactory.getUserFactory();
            IUser sendToUser =
                  usf.getUserByLogonID(
                        wdContext
```

```
                                        .nodeNotifData()
                                        .currentNotifDataElement()
                                        .getUserId());
                        Item item =
                                new Item(
                                        IProviderConnector.ADHOC_WORKFLOW_CONNECTOR_ID,
                                        IProviderConnector.ADHOC_WORKFLOW_SYSTEM,
                                        "" + new Date().getTime(),
                                        sendToUser.getUniqueID());

                        item.setUser(sendToUser.getUniqueID());
                        item.setSubject(
                                wdContext
                                        .nodeNotifData()
                                        .currentNotifDataElement()
                                        .getSubject());
                        item.setDescription(
                                wdContext
                                        .nodeNotifData()
                                        .currentNotifDataElement()
                                        .getDescription());
                        item.setItemType(ItemType.UWL_ITEM_NOTIFICATION);
                        item.setCreatedDate(new Date());
                        //item.setDueDate(new Date());
                        item.setCreatorId(loggedInUser.getUniqueID());
                        item.setStatus(StatusEnum.COMPLETED);
                        item.setCompletedDate(new Date());

    // External Type is used to filter the notifications according to application

                        item.setExternalType("UWLNotification");

                        Set users = new HashSet();
                        users.add(sendToUser.getUniqueID());
                        //
       users.add("USER.PRIVATE_DATASOURCE.un:testuser1");

                        push.pushSharedItem(procon, item, users);

                        uwlService.endSession(myContext);

                        wdComponentAPI.getMessageManager().reportSuccess(
                                "Notification Sent");
                } catch (UWLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        wdComponentAPI.getMessageManager().reportSuccess(
                                e.getCause() + "\n" + e.getMessage());
                } catch (NullPointerException e) {
                        wdComponentAPI.getMessageManager().reportSuccess(
                                e.getCause() + "\n" + e.getMessage());
                } catch (UMException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        wdComponentAPI.getMessageManager().reportSuccess(
                                e.getCause() + "\n" + e.getMessage());
```

```
        }

    //@@end
  }

  //@@begin javadoc:completeNotification()
  /** Declared method. */
  //@@end
  public void completeNotification( com.sap.tc.webdynpro.progmodel.api.IWDNodeElement
item )
  {
    //@@begin completeNotification()

   try {

                        IUWLService uwlService =
                              (IUWLService) WDPortalUtils.getServiceReference(
                                    IUWLService.ALIAS_KEY);
                        UWLContext myContext = new UWLContext();
                        IWDClientUser user;
                        user = WDClientUser.getLoggedInClientUser();
                        IUser loggedInUser = user.getSAPUser();

                        myContext.setUser(loggedInUser);

                        IUWLSession mySession =
                              uwlService.getUwlSessionForWebDynproClient(myContext);

                        IUWLItemManager itemMan =
uwlService.getItemManager(myContext);

                        Item notifItem = (Item) item.getAttributeValue("Item");

                        notifItem.setStatus(StatusEnum.COMPLETED);

                        IPushChannel push = uwlService.getPushChannel();
                        IProviderConnector procon =
                              uwlService.getRegisterProviderConnector(
                                    IProviderConnector.ADHOC_WORKFLOW_CONNECTOR_ID);

                        push.updateItem(procon,myContext,notifItem);

                        uwlService.endSession(myContext);

                        getInbox();


                } catch (UWLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();

   wdComponentAPI.getMessageManager().reportSuccess(""+e.getCause());
                }


    //@@end
```

```
  }
```

The methods created above can then be called from the respective action handlers as follows:

```
  //@@begin javadoc:onActionSend(ServerEvent)
  /** Declared validating event handler. */
  //@@end
  public void onActionSend(com.sap.tc.webdynpro.progmodel.api.IWDCustomEvent wdEvent
)
  {
    //@@begin onActionSend(ServerEvent)

          wdThis.sendNotification();

    //@@end
  }

  //@@begin javadoc:onActionRefresh(ServerEvent)
   /** Declared validating event handler. */
  //@@end
  public void onActionRefresh(com.sap.tc.webdynpro.progmodel.api.IWDCustomEvent
wdEvent )
  {
    //@@begin onActionRefresh(ServerEvent)

          wdThis.getInbox();

    //@@end
  }

  //@@begin javadoc:onActionComplete(ServerEvent)
  /** Declared validating event handler. */
  //@@end
  public void onActionComplete(com.sap.tc.webdynpro.progmodel.api.IWDCustomEvent
wdEvent, com.sap.tc.webdynpro.progmodel.api.IWDNodeElement item )
  {
    //@@begin onActionComplete(ServerEvent)

                wdThis.completeNotification(item);

    //@@end
  }
```

The getInbox( ) method is also invoked from wdDoInit( )

```
//@@begin javadoc:wdDoInit()
   /** Hook method called to initialize controller. */
//@@end
public void wdDoInit() {
   //@@begin wdDoInit()
```

```
            wdThis.getInbox();

      //@@end
   }
```

Following is the code to be entered in the wdDoModifyView function:

```
public static void wdDoModifyView(IPrivateUWLDemoAppView wdThis,
IPrivateUWLDemoAppView.IContextNode wdContext,
com.sap.tc.webdynpro.progmodel.api.IWDView view, boolean firstTime)
  {
    //@@begin wdDoModifyView

   if (firstTime) {

                        IWDButton button = (IWDButton)
view.getElement("CompleteButton");
                        button.mappingOfOnAction().addSourceMapping("nodeElement",
"item");
                }

    //@@end
  }
```

Run the Application

Login into the application with the portal user.

Send a notification to another portal user.

Login with the user to whom the notification was sent to find the notification in his Inbox.

On clicking Complete, the inbox gets refreshed.


Send To:
Subject:
Description:
Send
Inbox
Refresh
Subject_0
Description_0
Status
CreatorId
CreatedDate
NEW
Complete
1111
NEW
Complete
NEW
Complete

Related Content

UWL API Documentation

UWL Configuration

How to Configure the Universal Worklist (NW2004)

Developing Connectors with the Universal Worklist API

For more information, visit the User Interface Technology homepage.

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.