Tutorial: Consuming Web Services in Web Dynpro Java

Applies to:

Web Dynpro for Java applications for SAP enhancement package 1 for SAP NetWeaver CE 7.1.

For more information, visit the User Interface Technology homepage.

Summary

The tutorial describes how to create and implement a Web Dynpro application consuming a Web service using the Adaptive Web Service Model. This tutorial focuses on the Java Demo Enterprise Service *ProductByIDQueryResponseIn* from the ITelO demo company. In addition you learn how to create and configure a Service Group and a Provider System with SAP NetWeaver Administrator.

Prerequisites

- Systems, installed applications, and authorizations
- The SAP NetWeaver Developer Studio is installed on your computer.
- You have access to the SAP J2EE Engine.
- You have acquired some basic experience with Web Dynpro applications - for example, by working through the Welcome Quick start Guide (Web Dynpro Java for Newbies).
- Basic knowledge of Java would be an advantage.

Details

Level of complexity: Beginner

Time required for completion: 60 min.

Author: Stefanie Bacher with grateful thanks to **Swen König**, who created this tutorial during his practical phase in our team.

Company: SAP AG

Created on: 10 October 2008

Author Bio

Stefanie Bacher works as a product specialist within the SAP NetWeaver Product Management UI Technology Team. She focuses on rollout and knowledge transfer for Web Dynpro Java.

Table of Contents

Introduction

This tutorial is an example for consuming a Web service in Web Dynpro Java. You will import a Web service using the Adaptive Web Service Model and map the contexts.
In addition you will create a Service Group and a Provider System. These are needed for authorization of the Web service.
Finally this tutorial shows you how to design the view for the appropriate application.

This will be the result of the tutorial:

The application is consuming the Java Demo Enterprise Service *ProductByIDQueryResponseIn*. It is provided by ITelO, a demo model for SAP NetWeaver. Based on data on the database you get details for products.

You have to enter a Product ID (e.g.: **HT-1000**) and click *Get Details*. If the product exists you will get some product details such as name, weight, price and picture.

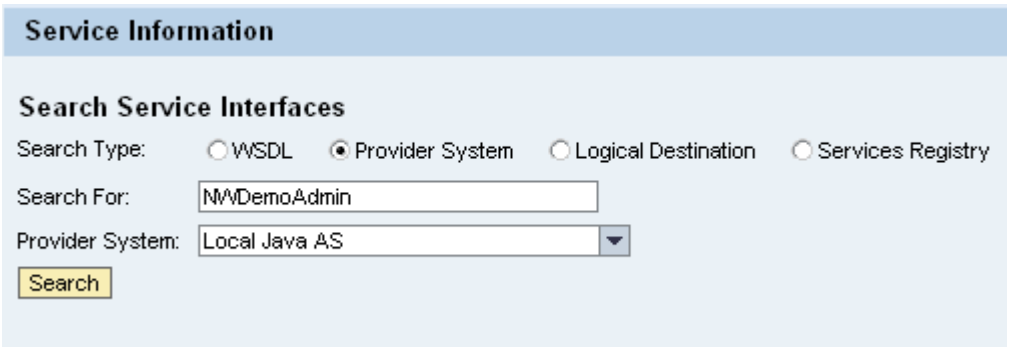
In this tutorial, you will learn how to:

- Create a Adaptive Web Service Model to be used for connecting an external Web service from within the Web Dynpro project
- Create and configure a Service Group
- Create and configure a Provider System with SAP NetWeaver Administrator
- Apply the Service Controller Code template for creation of a binding relation to the model
- Design a simple view layout applying the *Form* and *Action Button* templates
- Perform the implementation for availing of the Web service used.

Prerequisite

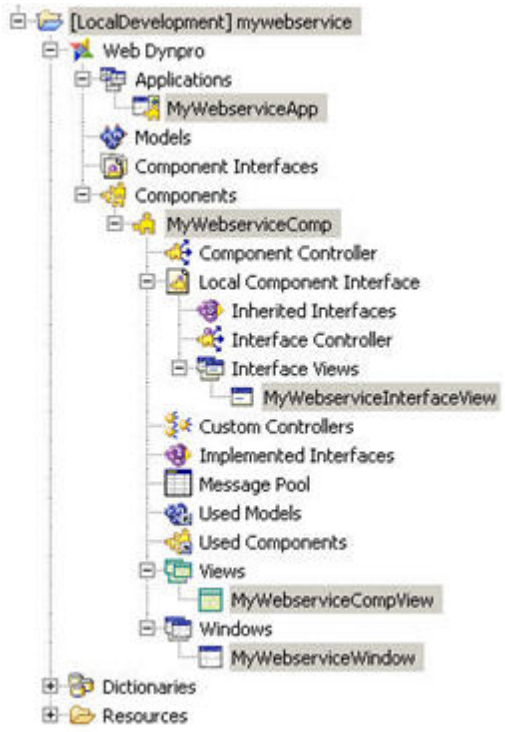
To be able to use these Demo Enterprise Services you have to execute the following procedure:

1. Open the start page of the SAP NetWeaver Application Server Java using the following URL: `http://<host>:<port>` (e. g. http://localhost:50000).
2. Choose Web Services Navigator.
3. Select *Provider System* as *Search Type* and search for **NWDemoAdmin.**



4. Select the Service Interface **NWDemoAdmin** and choose Next.
5. Select the *generateData* operation and choose *Next* once more.
6. Choose *Invocation Parameters* and change the *Timeout* value to **600**.
7. Choose *Next* to execute the *generateData* operation. The demo data is generated.

Creating a Web Dynpro Development Component (DC) and Application



1. To create a new Web Dynpro DC choose *File –> New -> Web Dynpro Development component,* select the *MyComponents* Software component and click *Next.* Name the component **mywebservice** and complete the wizard.

2. Select *Applications*, open the context menu and choose *Create Application*. Name the application **MyWebserviceApp** and the package *com.sap.examples.mywebserviceapp*.
   The component is called **MyWebserviceComp**, and the package is *com.sap.examples.webservicecomp.*

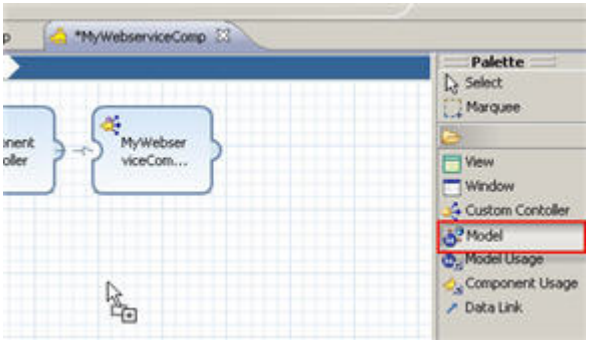
3. Leave the *Default Window and Views* option selected in order to get them applied by default.

Creating an Adaptive Web Service Model

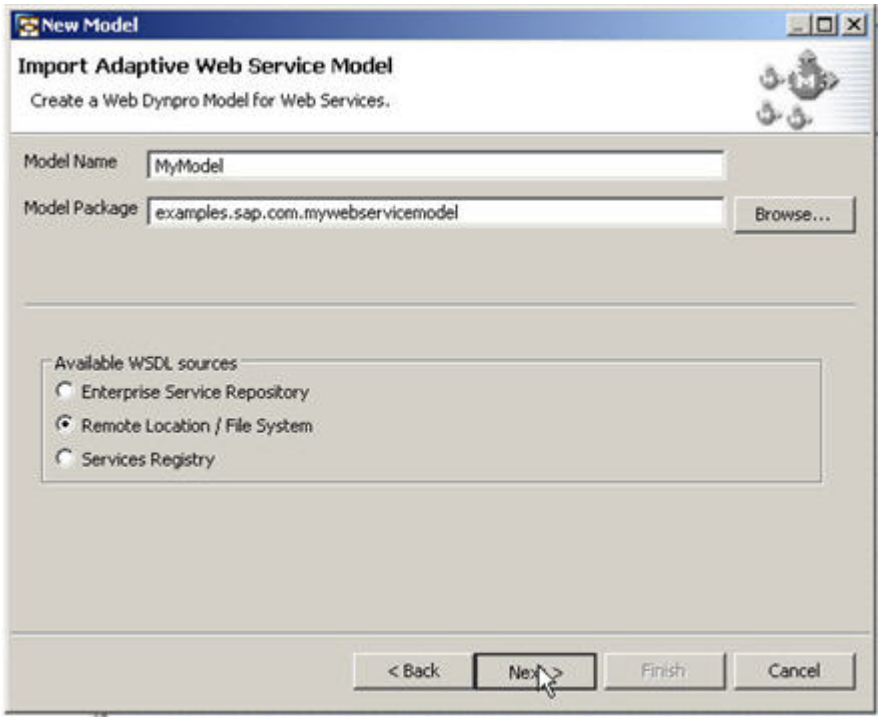
To provide a Web service for your Web Dynpro application you have to create a new model.

The model holds the data for the request and response of the Web service.

1. Double-click the *MyWebserviceComp* to open the Data Modeler.
2. Select *Create a new model* and click in the area of your component to open the *New Model* wizard.



3. From the wizard choose *Adaptive Web Service Model* and click *Next*.
4. Now enter **MyModel** for name, *com.sap.examples.webservicemodel* for package and choose *Remote Location / File System* from the radio button group and confirm with *Next*.



5. In this step you have to enter the URL of the Web service you want to use in your Web Dynpro application. Due to this enter `http://<host>:<port>/ProductByIDQueryResponseInService/ProductByIDQueryResponseIn?wsdl`

by replacing <host> and <port> with the appropriated values of your SAP NetWeaver Application Server.

6. Wait for system validating the WSDL.

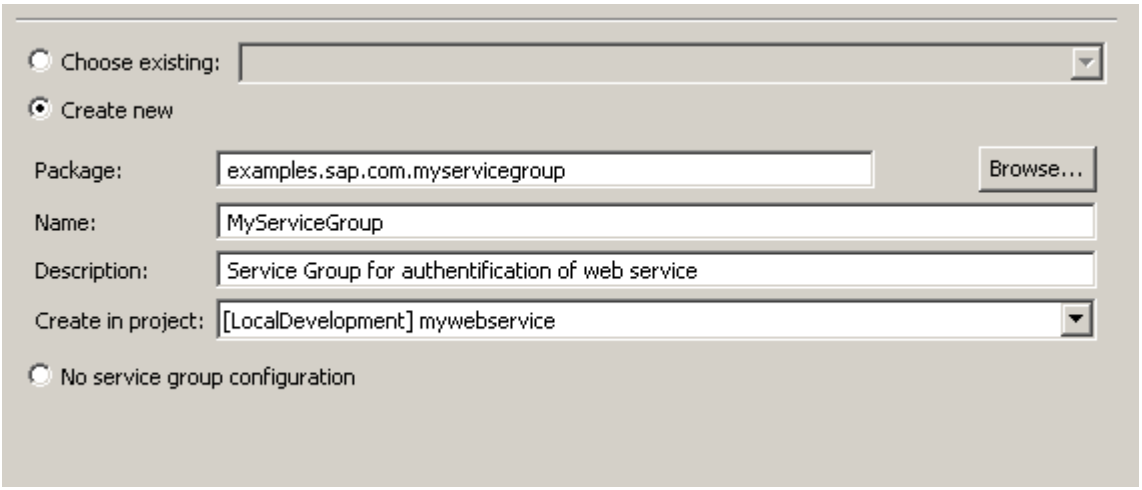
Note: If validation was not successful you should check the proxy settings at the Network settings preference page and the URL of your web service. As long as the WSDL cannot be validated, you cannot create the model, because the wizard creates the model based on the WSDL description of the Web service.

7. Afterwards you can click *Next* to continue.

8. Select *Create new* choose package *examples.sap.com.myservicegroup,* enter **MyServiceGroup** as *Name* and **Service Group for Authorization of Web Service** for *Description*.

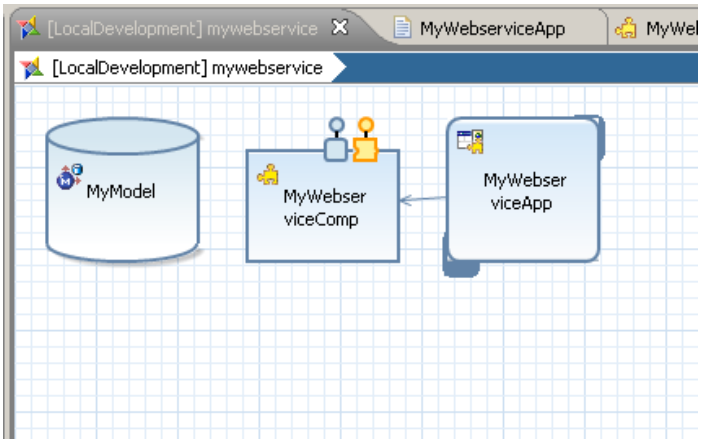
Web services usually require authorization. For that reason you have to create a new Service Group for your Web Dynpro application.

9. Leave the default option for *Create in project* in order to create the Service Group in your *mywebservice* project and confirm with *Next*.



Note: There are some more steps needed to configure your Service Group on the SAP NetWeaver Application Server. These steps will be explained in detail later in this tutorial. For now we finish creation of the model.

10. If you are asked for Logical Destinations, choose *No logical destinations…* and click *Next*.
11. Click *Next* to start the importing process and leave the wizard with *Finish*.
12. The new Adaptive Web Service Model *MyModel* was created and is displayed in the Component Modeler.



13. This is a good point in time to save your project and rebuild it.
14. To be able to configure the Service Group in the next step, you have to deploy your DC. To do this, select the *mywebservice* DC in the Web Dynpro Explorer and choose *deploy* from the context menu.

Configuration for Service Group

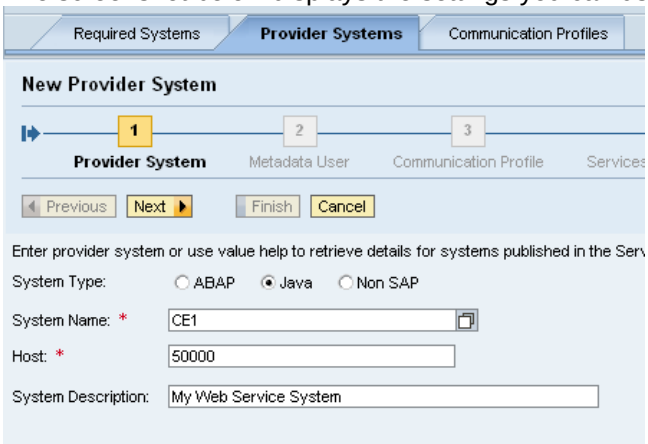
In the previous chapter you created a Service Group. In the following steps you will create a Provider System with the needed credentials for the Web service and assign it to your Service Group. These steps will be needed to automatically authorize your application for using the Web service on runtime.

Creating Provider System

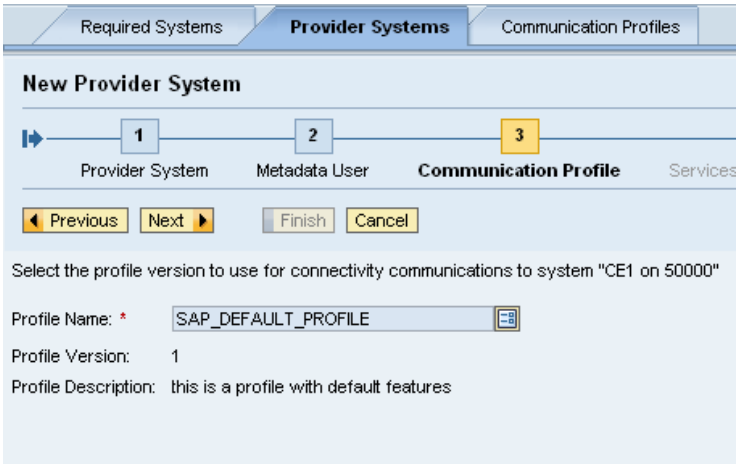
1. To open the SAP NetWeaver Administrator on your SAP NetWeaver Application Server, enter **http://<host>:<port>/nwa** in your Internet Browser.
2. Navigate to *SOA Management* - > *Technical Configuration* - > *System Connections*.
3. To create a new Provider System select the *Provider Systems* tab and click *New*.
4. Enter the following
   - System Type: *Java*
   - System Name: **<System ID>**
   - Host: **<Host>**
   - System Description: **My Web Service System**

   and confirm with *Next.*

   The screenshot below displays the settings you can use for the CE Trial version EHP1:



5. Enter username and password and confirm with *Next*.
6. In the *Communication Profile* step, leave all default settings unchanged and confirm once more with *Next*.



7. Configure the Services Search Settings:
   - Mode: *Multiple Services*
   - Services Source: *WSIL*
   - WSIL URL: **http://<host>:<port>/inspection.wsil**
   - Socket Timeout: **60.000**

   Confirm with *Finish*.

The Provider System appears in the Provider Systems Table and the status is active.

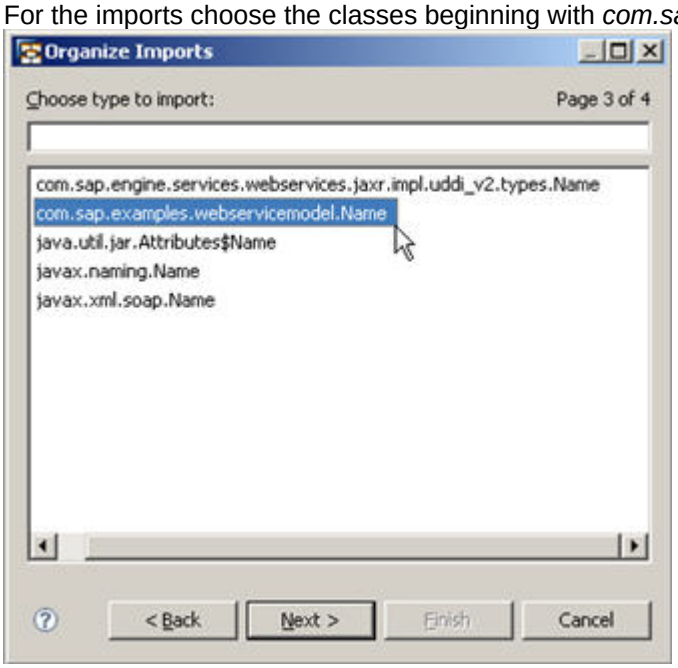
Assigning Provider System to Service Group

1. To access the Application Communication Configuration return to *SOA Management,* select *Application and Scenario Communication* and open *Application Communication*.
2. Now search for your DC **mywebservice**, select your project and click *Edit* to configure the Service Group in your project.
3. Afterwards you can assign a Provider System to your Service Group by clicking on the button. Choose your recently created Provider System and click *OK*.
4. Finally complete the configuration by clicking *Save*.
   You can close the Internet Browser and come back to your Web Dynpro DC in the NWDS.

Apply Service Controller Code Template

Next step you have to do is to create the context in the *Component Controller* of your component *MyWebserviceComp* and create a binding relation to the model. Therefore we apply the Service Controller Code Template.

1. Click the *Component Controller* in the Web Dynpro Explorer.
   From context menu choose *Template* → *Apply…*
2. Now select *Service Controller* and continue with *Next*.
   After that take the *Request_ProductByIDQueryResponseIn* class followed by clicking *Next*.
3. In this step you have to choose the nodes and attributes from the model you want to map to your Component Controller.
   In this case mark all attributes to map the complete context and click *Next*.
4. Leave the default settings in this step in order to generate the appropriate execute method for the Web service.
   Complete the wizard by clicking *Finish*.
5. At the moment the Component Controller displaying some errors. These errors occur because of missing imports in the Component Controller.
   To fix that, choose *Open* → *Java Editor* from context menu of the Component Controller.
   You can now see the Java code generated by the NWDS. To organize the imports make a right-click in the code and select *Source* → *Organize Imports* (or simply use *Ctrl+Shift+O)*.
   For the imports choose the classes beginning with *com.sap.examples.webservicemodel...*

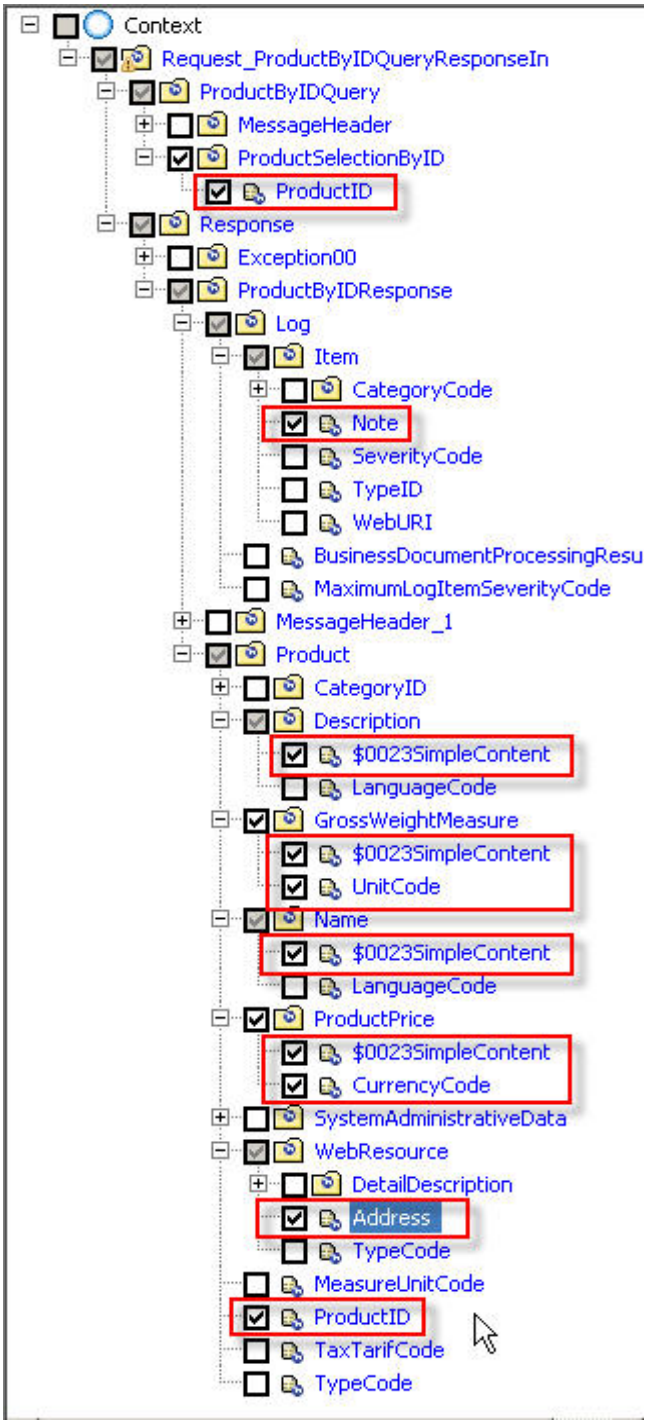


6. Save and rebuild your project.

Mapping Context for View

To provide the view with data from the Web service, you also need to map the Context of the view to the Component Controller.

1. Therefore create a *Data Link* between MyWebserviceCompView and *ComponentController* in the *MyWebserviceComp*.
2. In the *Context Mapping* wizard drag the *Request_ProductByIDQueryResponseIn* context node and drop it on the Context of your view. Now you can select the attributes you need for your view. In this tutorial we will use the following:



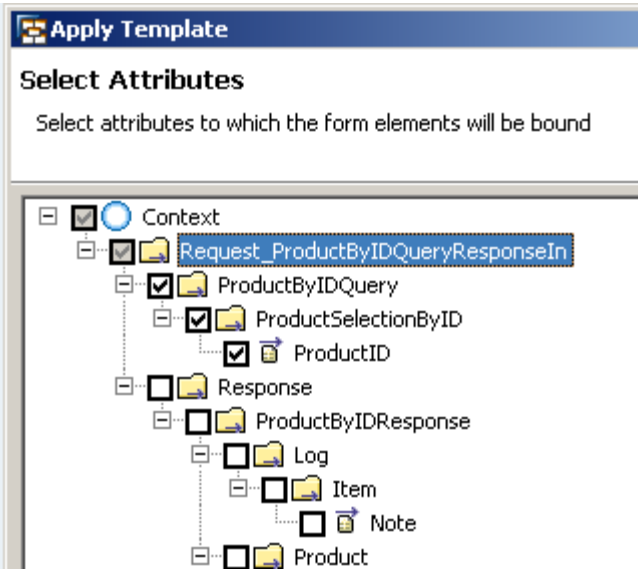
3. Save and rebuild your project.

Designing View

Now it is time for designing the view layout.

Creating the UI elements for the query:

1. To open the View Editor for the *MyWebserviceCompView,* choose *Open - > View Editor* from the context menu.
2. To arrange the UI elements, change the following properties:
   - For the *RootElement* change the layout property to *GridLayout*.
   - For the *DefaultTextView* change following properties:
     - Design: *header1*
     - Text: **Get Product Details**
     - hAlign: *center*
3. Choose *Apply Template* from the context menu of the *RootElement*.
4. Select *Form* and select the *ProductID* attribute from the *ProductByIDQuery* node in the next step. Confirm with *Finish*.



5. Open the context menu for *TransContainer_0* and choose *Apply Template* and select the Action Button template.
6. In the following screen enter **Get Details** for *Button Label* and click *Next.* The respective action and event handler are generated after having finished this wizard.
7. Now select *Call Method*, select your Component Controller **MyWebserviceComp** as *Controller* and **executeProductByIDQueryResponseIn** as *Method* and confirm with *Finish*.

With the last step the following action handler is created and assigned to the button's onAction event. If this event is triggered, the component controller's method to execute the model is called.

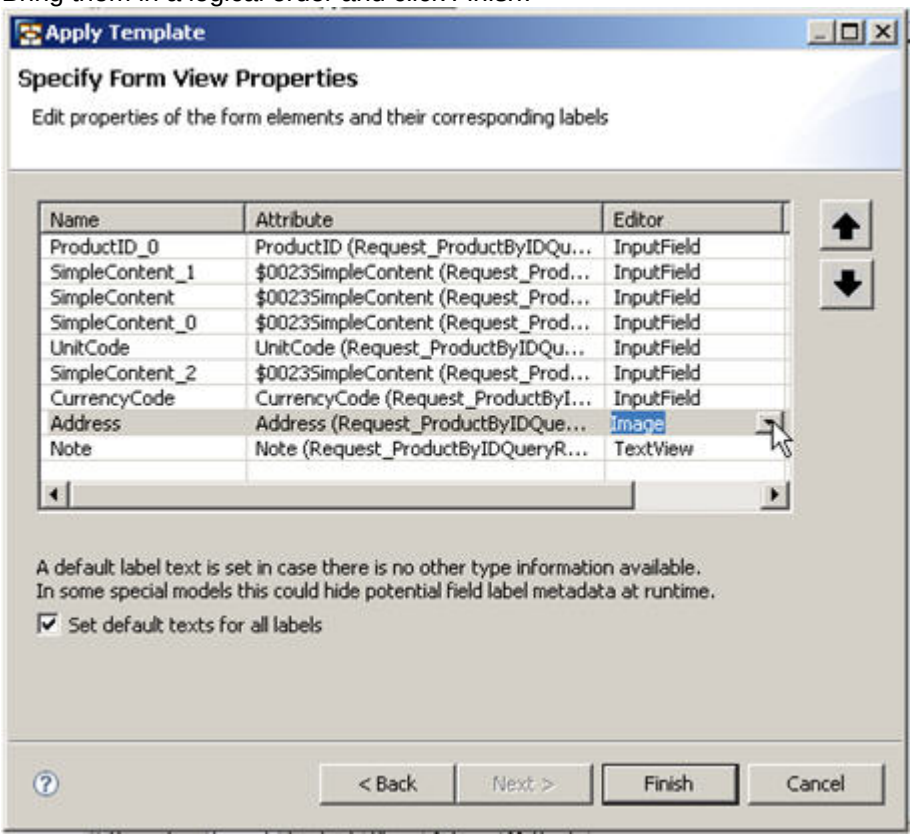
```
public void onActionGetDetails(com.sap.tc.webdynpro.progmodel.api.IWDCustomEvent
wdEvent )
  {

        //@@begin onActionGetDetails(ServerEvent)
        //$$begin Action Button(-1769999083)
       wdThis.wdGetMyWebserviceCompController().executeProductByIDQueryResponseIn();
        //$$end
        //@@end

  }
```

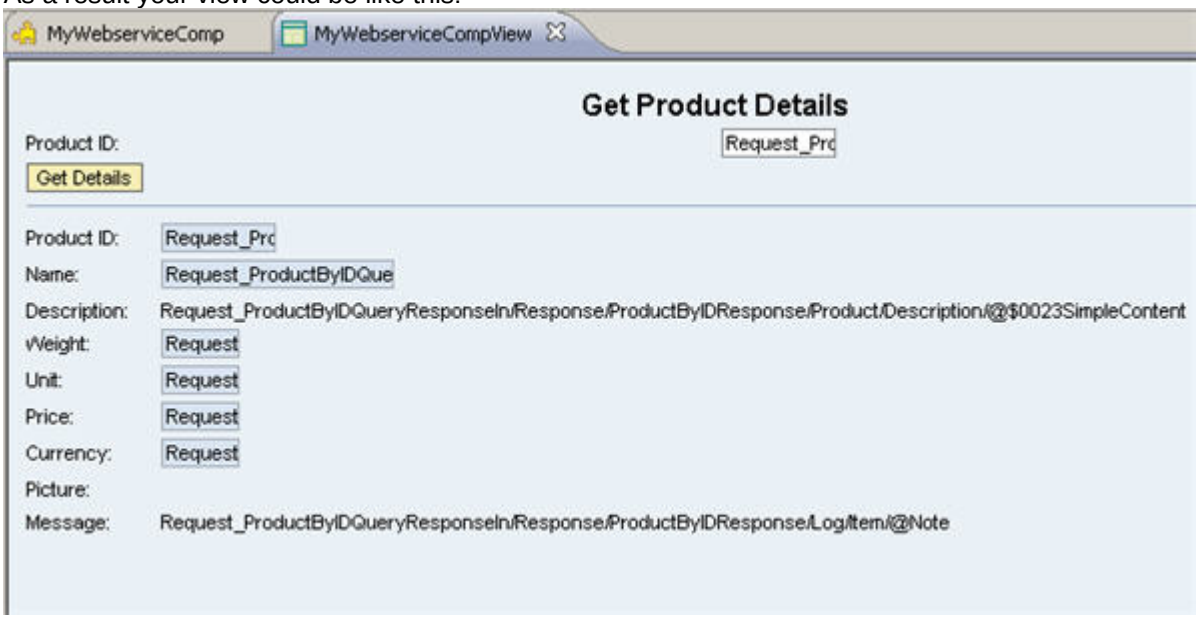
Creating the UI elements for displaying the results

1. In order to structure your view, insert the UI element *HorizontalGutter* to the *RootElement*.
2. Afterwards apply a second *Form* template for the response parameters.
   Mark the *Response* node and click *Next*.
   Next change the *Editor* for the *Address* to *Image*, for the *SimpleContent* and the *Note* to *TextView*.
   Bring them in a logical order and click *Finish*.



3. You can now change the following attributes to get a good-looking layout.
   - For the two *TransContainer* set property *colCount* to **2**.
   - For the input fields in the *TransConteiner_0_0* change the property *readOnly* to *true*.
   - For all labels change the *text* property.
   - For the two Product ID input fields change the property *length* to **7**.
   - For the Weight, Unit, Price and Currency input fields change the property *length* to **3**.

As a result your view could be like this:



Finally save your project and deploy and run your application.

Note: Currently there is some additional coding needed to run the application. This is because the NW Demo Enterprise Services are still in development and not yet finished. At this time the WSDL definition for the web service is very strong. There are some fields that must not be null. So, if you get an error while executing the application, you have to modify the *wdDoInit()* method of your *Component Controller*.
Therefore in the Web Dynpro Explorer select the Component Controller and choose *Open* → *Java Editor* from context menu.
Scroll down to the implementation of the *wdDoInit()* method and insert the following code.
Afterwards the application should run correctly.

BusinessDocumentMessageHeader messageheader:

```
BusinessDocumentMessageHeader messageHeader = new
BusinessDocumentMessageHeader(myModelModel);
//Insert this here ------------------------------------------------------
Date date = Date.valueOf("2008-08-01");
Timestamp timestamp = new Timestamp(date.getTime());
messageHeader.setCreationDateTime(timestamp);
messageHeader.setTestDataIndicator(true);
messageHeader.setReconciliationIndicator(false);
messageHeader.setSenderBusinessSystemID("001");
messageHeader.setRecipientBusinessSystemID("002");
//end of Insert---------------------------------------------------------
productByIDQuery.setMessageHeader(messageHeader);
```

BusinessDocumentMessageID iD_1:

```
BusinessDocumentMessageID iD_1 = new BusinessDocumentMessageID(myModelModel);
//Insert this here ------------------------------------------------------
iD_1.setSchemeAgencyID("001");
iD_1.setSchemeID("000");
iD_1.setSchemeAgencySchemeAgencyID("002");
iD_1.set$0023SimpleContent("003");
//end of Insert---------------------------------------------------------
```

```
messageHeader.setID(iD_1);
```

BusinessDocumentMessageID referenceID:

```
BusinessDocumentMessageID referenceID = new BusinessDocumentMessageID(myModelModel);
//Insert this here ------------------------------------------------------------
referenceID.setSchemeAgencyID("001");
referenceID.setSchemeID("000");
referenceID.setSchemeAgencySchemeAgencyID("002");
referenceID.set$0023SimpleContent("003");
//end of Insert---------------------------------------------------------------
messageHeader.setReferenceID(referenceID);
```

ContactPersonInternalID internalID_2:

```
ContactPersonInternalID internalID_2 = new ContactPersonInternalID(myModelModel);
//Insert this here ------------------------------------------------------------
internalID_2.setSchemeAgencyID("001");
internalID_2.setSchemeID("000");
internalID_2.set$0023SimpleContent("003");
//end of Insert---------------------------------------------------------------
ContactPerson_1.setInternalID(internalID_2);
```

PartyInternalID internalID_3:

```
PartyInternalID internalID_3 = new PartyInternalID(myModelModel);
//Insert this here ------------------------------------------------------------
internalID_3.setSchemeAgencyID("001");
internalID_3.setSchemeID("000");
internalID_3.set$0023SimpleContent("003");
//end of Insert---------------------------------------------------------------
senderParty.setInternalID(internalID_3);
```

java.util.List<Name> name:

```
java.util.List<Name> name = new ArrayList<Name>();
//Insert this here ------------------------------------------------------------
Name n = new Name(myModelModel);
n.set$0023SimpleContent("");
n.setLanguageCode("DE");
name.add(n);
//end of Insert---------------------------------------------------------------
product.setName(name);
```

Related Content

Parameter Mapping

ITelO – The Demo Company

Java Demo Enterprise Services

Integrating Services

For more information, visit the User Interface Technology homepage.

Copyright