How to Integrate SAP xMII Services with Web Dynpro Java

Applies to:

SAP xMII 11.5
SAP Netweaver 04s

Summary

This document gives a step by step description on how SAP xMII services and objects can be exposed as web services or HTTP services and can be integrated with Web Dynpro Java.

Author(s): Dipankar Saha

Company: IBM India Pvt. Ltd

Created on: 10 November 2006

Author Bio

Dipankar Saha is presently working in IBM India as Advisory System Analyst developing composite applications using SAP xMII. Previously he worked for SAP Labs India and was involved in development of xMII product, composite applications using Web Dynpro Java and ISA95 business package. Dipankar has about 5 years of experience in application software development.

Table of Contents

Introduction

Composite applications developed using SAP xMII uses the xMII Business Logic Transaction for developing business logic, Query Templates and Display Templates to aggregate and process data from various shop-floor systems and display them using graphical and statistical charts. But the UI development of SAP xMII composite application has no well-defined framework and most of the times xMII developers develops the UI in HTML and Javascript using notepad or other third-party web editors. But xMII developers can leverage the power of Web Dynpro Java to easily generate powerful UI embedding the xMII charts and BLS data in them.

Scenario

To aggregate and process shop-floor data we use the SAP xMII Business Logic transactions. These BLS transactions can be easily exposed as web services to be used by other applications like Web Dynpro. SAP xMII charts (iChart object) can also be exposed using a http service which gives a gif image of the chart.

Web Dynpro Java is based on MVC (model-view-controller) architecture and uses a backend model to retrieve data. This model can be a web-service (SAP xMII BLS Transaction) as well. Any image with a HTTP URL can also be embedded in a Web Dynpro page. We'll use these features to develop a composite application using Web Dynpro Java accessing xMII services on the backend.

Developing the Pieces

SAP xMII BLS Transaction As Web Service

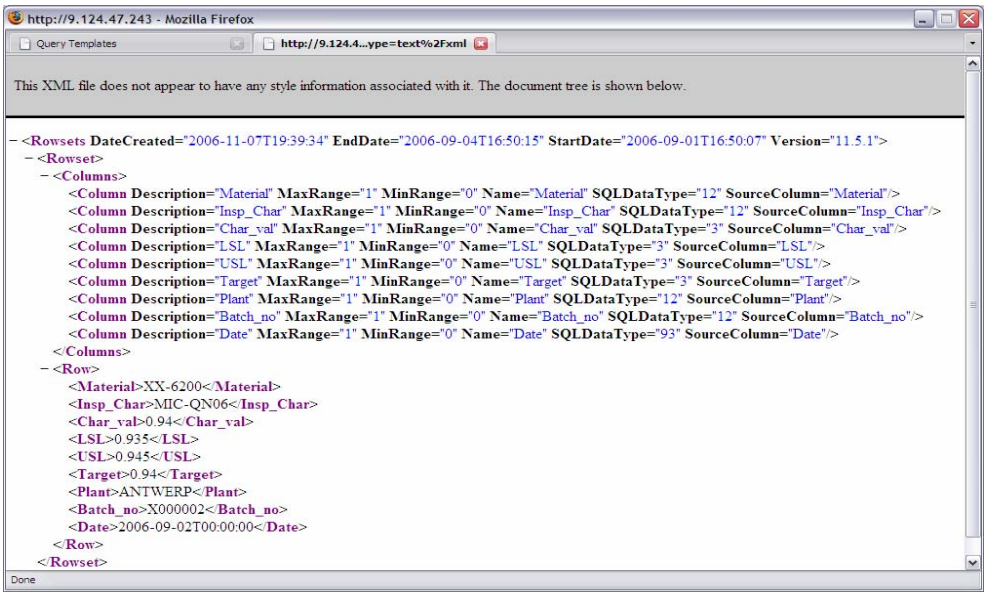
Any SAP xMII Business Logic Service Transaction can be exposed as a web service (WSDL) by the following URL:

http://<server>:<port>/Lighthammer/WSDLGen/<BLSTransactionName>

The BLS web service can be executed as a web service by the following URL:

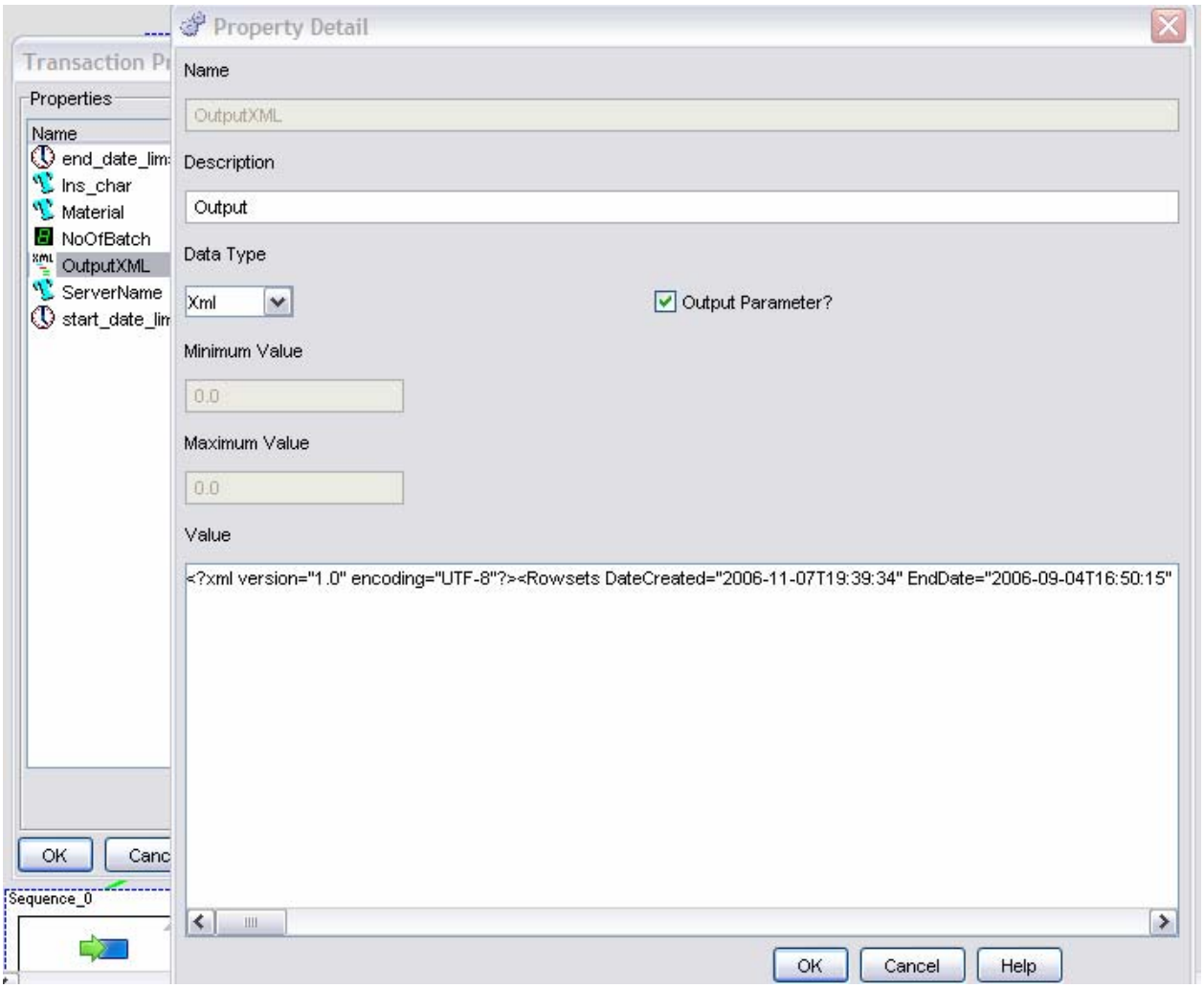
http://<server>:<port>/Lighthammer/SOAPRunner/<BLSTransactionName>

In this scenario we have a BLS Transaction to do QM analysis on data collected from LIMS, which accepts some input parameters and returns the data in XML format. To define the output XML structure we need to create a XML in SAP xMII format and assign that to the output parameter's value. To do this, run the BLS transaction as a XacuteQuery using a Query Template and get the output in XML format as below (Select text/xml option while testing the query).




```
- <Rowsets DateCreated="2006-11-07T19:39:34" EndDate="2006-09-04T16:50:15" StartDate="2006-09-01T16:50:07" Version="11.5.1">
  - <Rowset>
    - <Columns>
        <Column Description="Material" MaxRange="1" MinRange="0" Name="Material" SQLDataType="12" SourceColumn="Material"/>
        <Column Description="Insp_Char" MaxRange="1" MinRange="0" Name="Insp_Char" SQLDataType="12" SourceColumn="Insp_Char"/>
        <Column Description="Char_val" MaxRange="1" MinRange="0" Name="Char_val" SQLDataType="3" SourceColumn="Char_val"/>
        <Column Description="LSL" MaxRange="1" MinRange="0" Name="LSL" SQLDataType="3" SourceColumn="LSL"/>
        <Column Description="USL" MaxRange="1" MinRange="0" Name="USL" SQLDataType="3" SourceColumn="USL"/>
        <Column Description="Target" MaxRange="1" MinRange="0" Name="Target" SQLDataType="3" SourceColumn="Target"/>
        <Column Description="Plant" MaxRange="1" MinRange="0" Name="Plant" SQLDataType="12" SourceColumn="Plant"/>
        <Column Description="Batch_no" MaxRange="1" MinRange="0" Name="Batch_no" SQLDataType="12" SourceColumn="Batch_no"/>
        <Column Description="Date" MaxRange="1" MinRange="0" Name="Date" SQLDataType="93" SourceColumn="Date"/>
      </Columns>
    - <Row>
        <Material>XX-6200</Material>
        <Insp_Char>MIC-QN06</Insp_Char>
        <Char_val>0.94</Char_val>
        <LSL>0.935</LSL>
        <USL>0.945</USL>
        <Target>0.94</Target>
        <Plant>ANTWERP</Plant>
        <Batch_no>X000002</Batch_no>
        <Date>2006-09-02T00:00:00</Date>
      </Row>
    </Rowset>
```

Copy the XML from the web page source. Open the BLS transaction and edit the output parameter (of type Xml). Paste the copied XML into the Value field as below.



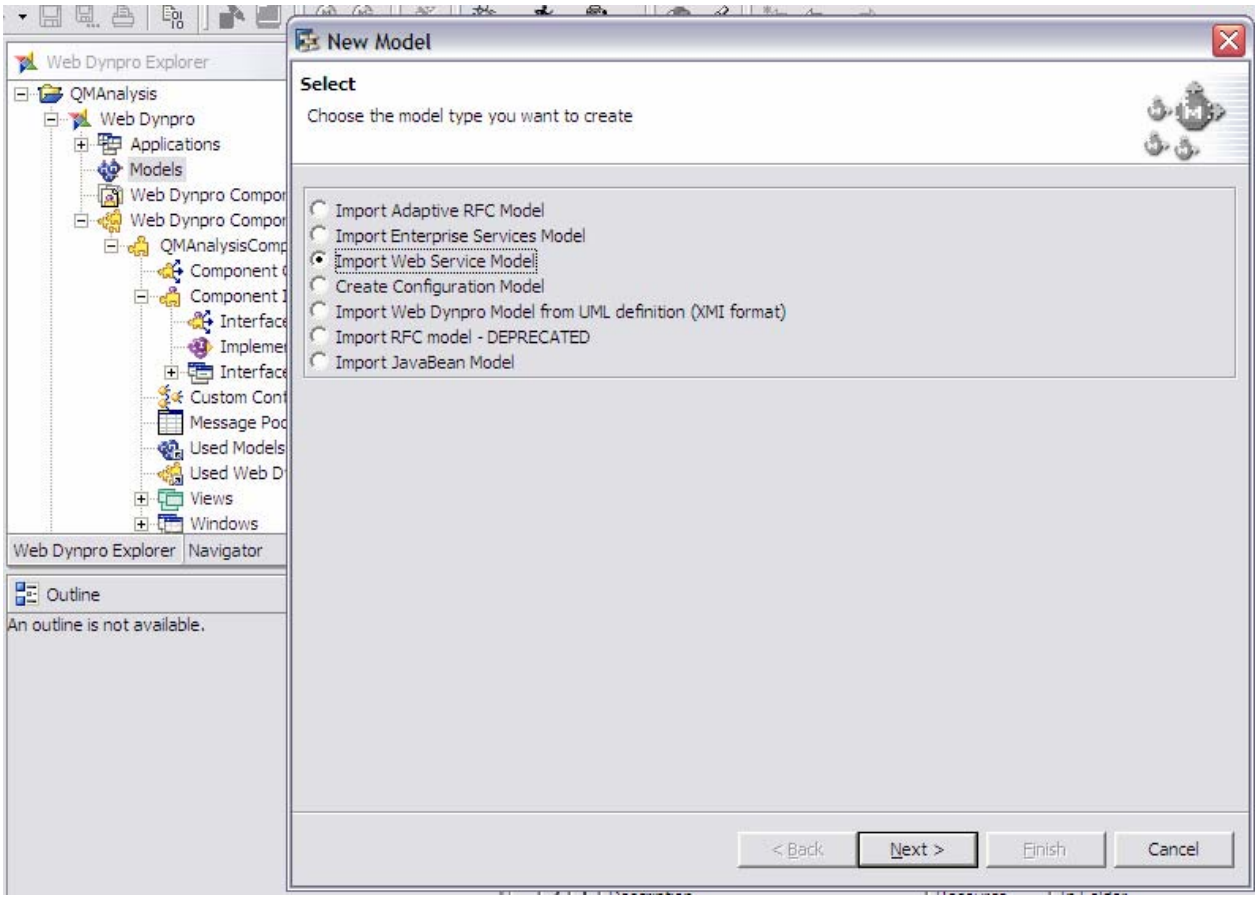
Save the transaction and from now on while generating the WSDL the correct structure of the output parameter will be reflected in the WSDL.

Creating the Web Dynpro Java Project

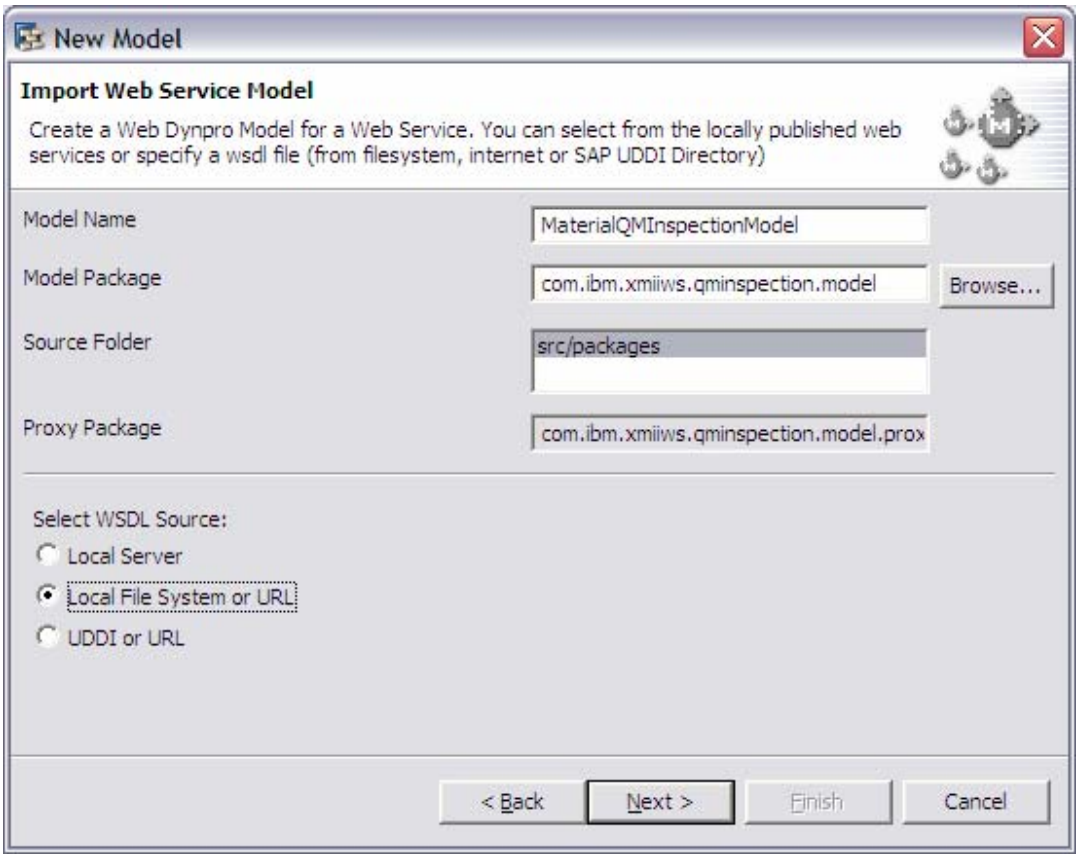
Open the Netweaver Developer Studio in Web Dynpro perspective to create the Web Dynpro Java project. Create a new Webdynpro Java project and create a Webdynpro component also.

Creating the Web Dynpro Model Object

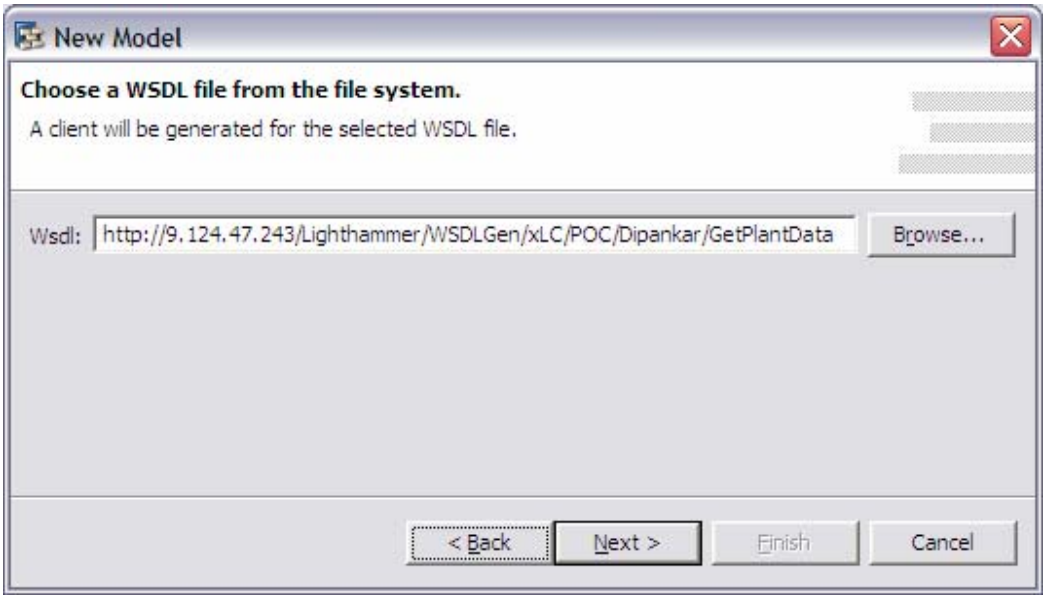
You need to import the model which is the SAP xMII BLS transaction exposed as a web service. Select Create Model option and on the dialog box select Import Web Service Model option as below and click on Next.



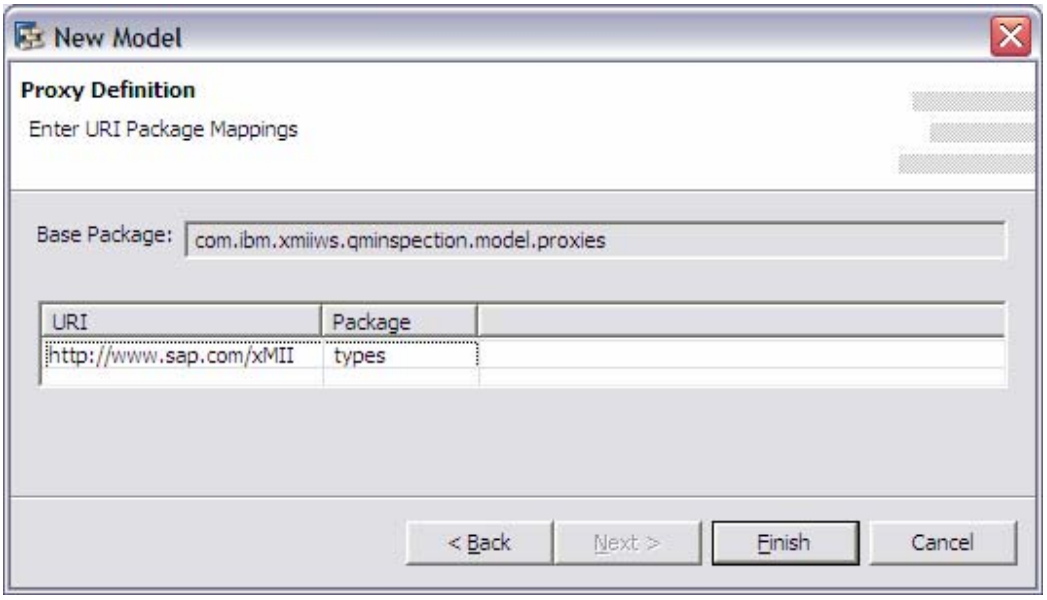
In the next step specify a Model name and model package. Select Local File System or URL option and click on Next.



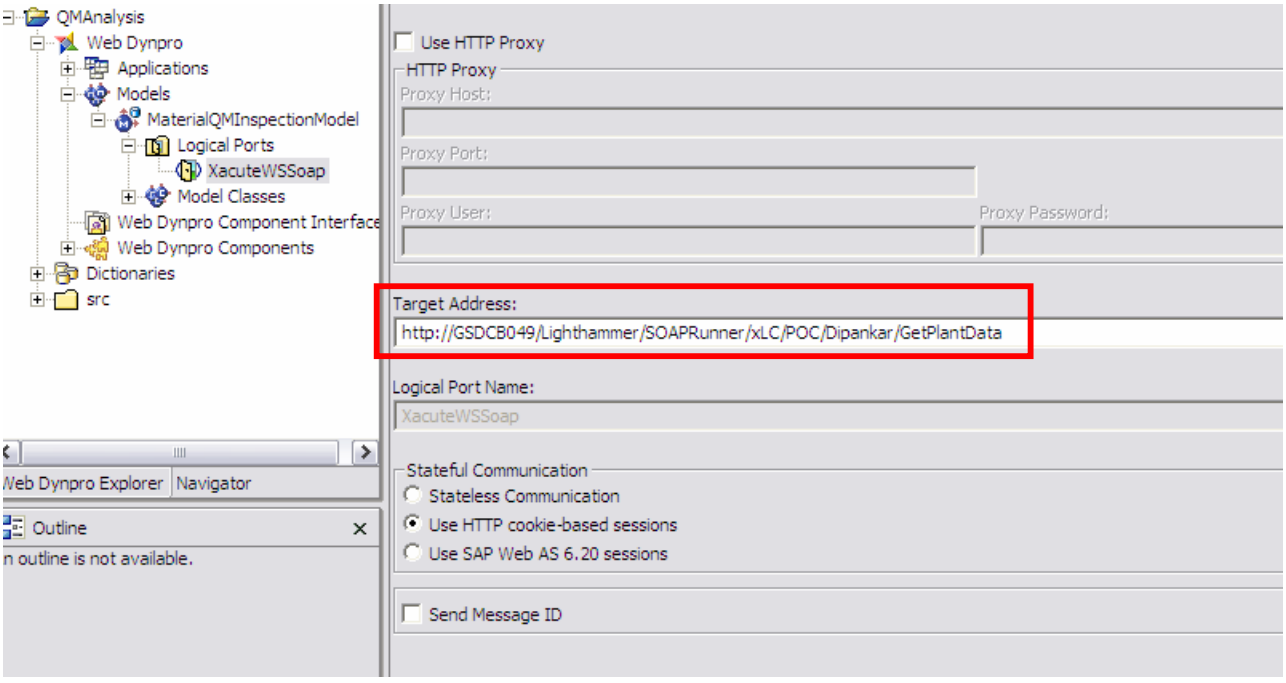
In the next step enter the WSDL URL for the SAP xMII BLS web service. It's typically in the format http://<server>:<port>/Lighthammer/WSDLGen/<BLSTransactionName>. Click on Next



In the next screen Proxy Definitions are shown. Click on Finish.

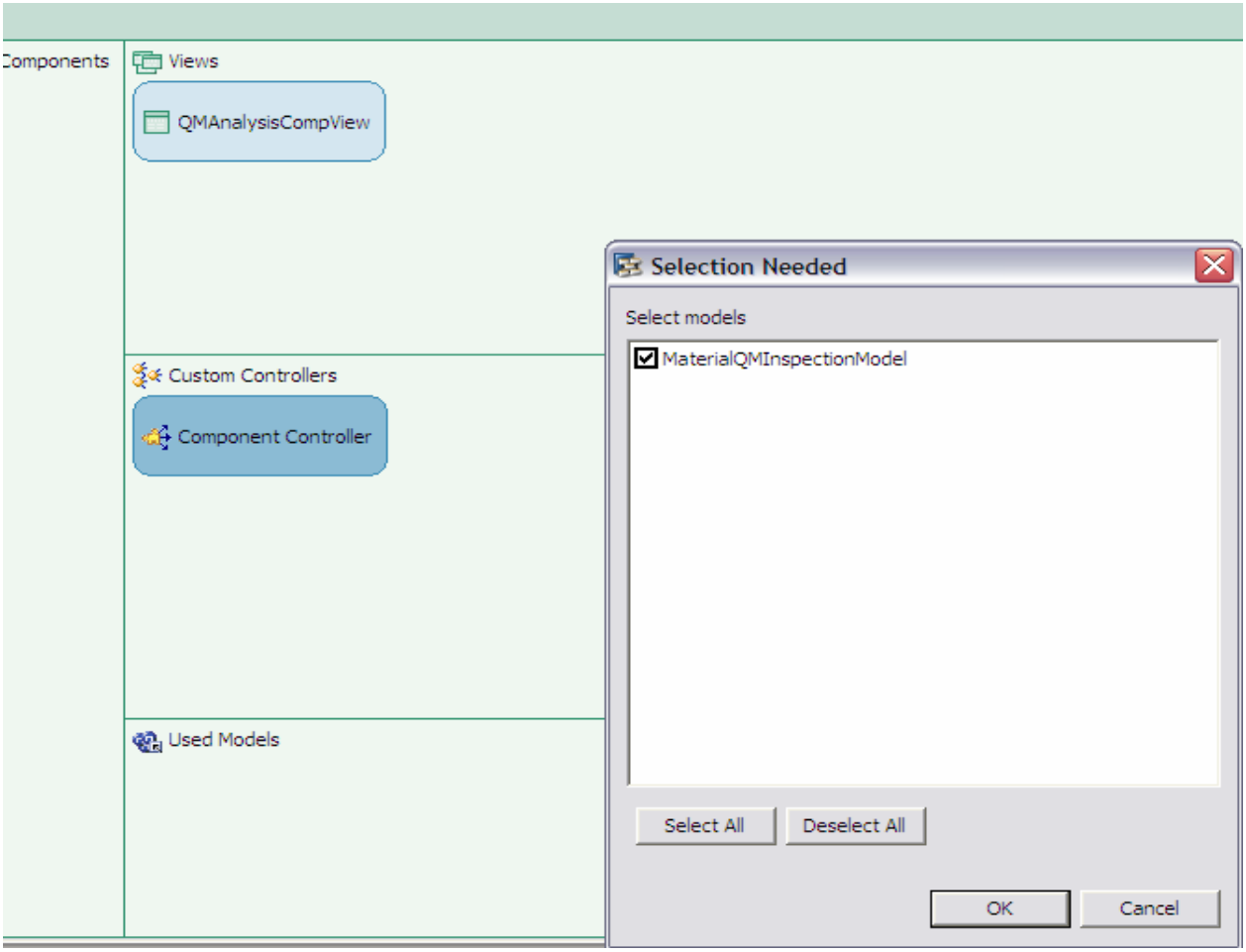


The logical port created while importing the model can be configured later as well. For Example if you want to change the xMII server's host name/IP you can change it in the Logical Port's Target Address as shown below.

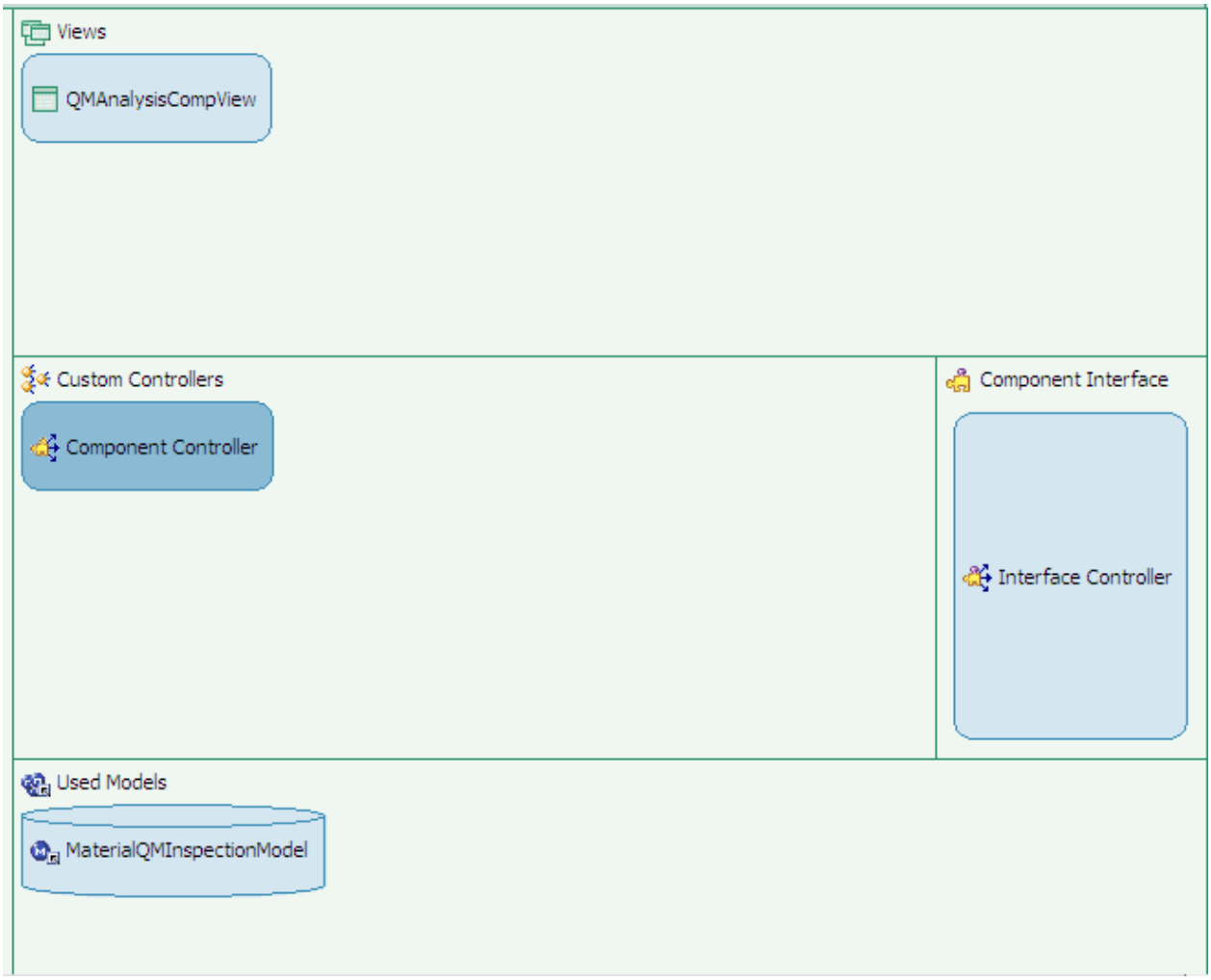


Defining Model Usage

Open the Web Dynpro component. We need to add the model created above as a Used Model for the component. Right-click on the Used Models area and select the SAP xMII web service Model and click on OK.



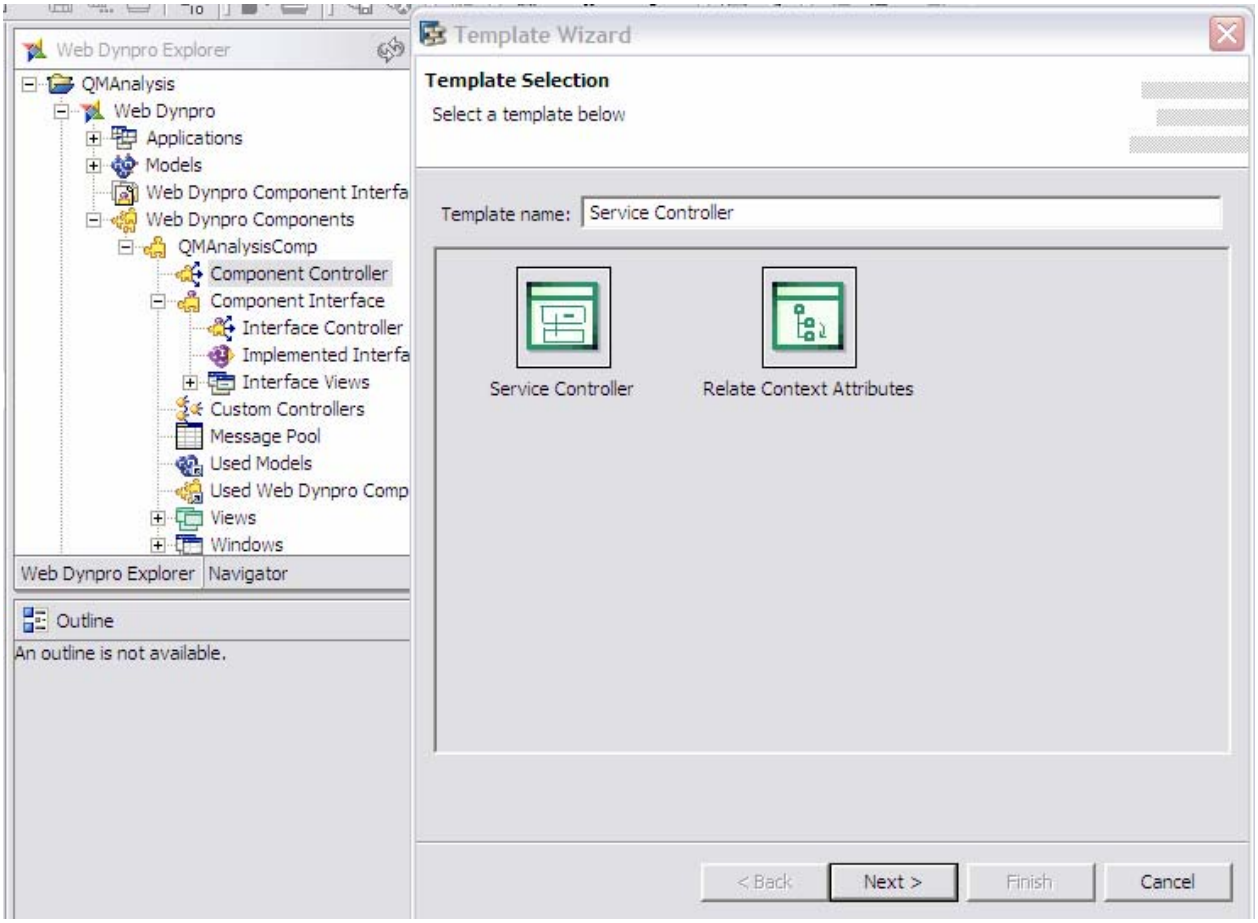
Now the screen should look as below.



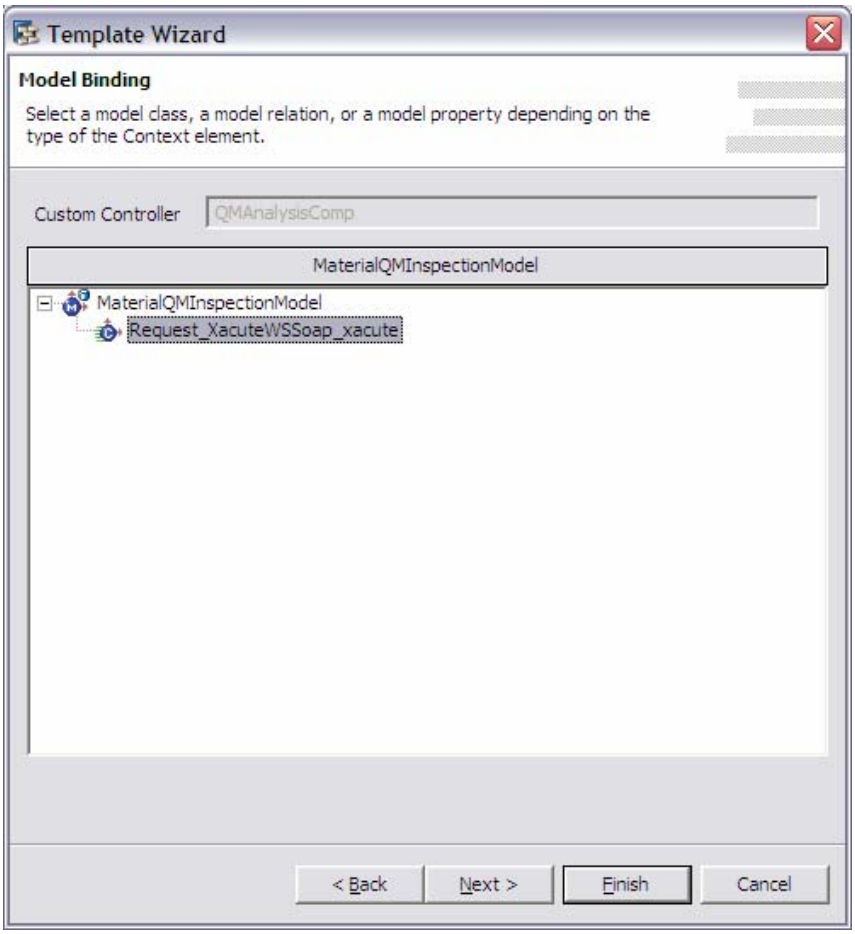
Applying Web Dynpro Pattern to the Component Controller

Web Dynpro component controller is the default controller of the Web Dynpro component. It calls the web service model object and gets the data to be displayed in the view. You can use the Service Controller pattern to create the necessary context nodes, mappings and methods.

Select the Component Controller and right-click on it. Select the Apply > Template option. Select the Service Controller option as below and click on Next.

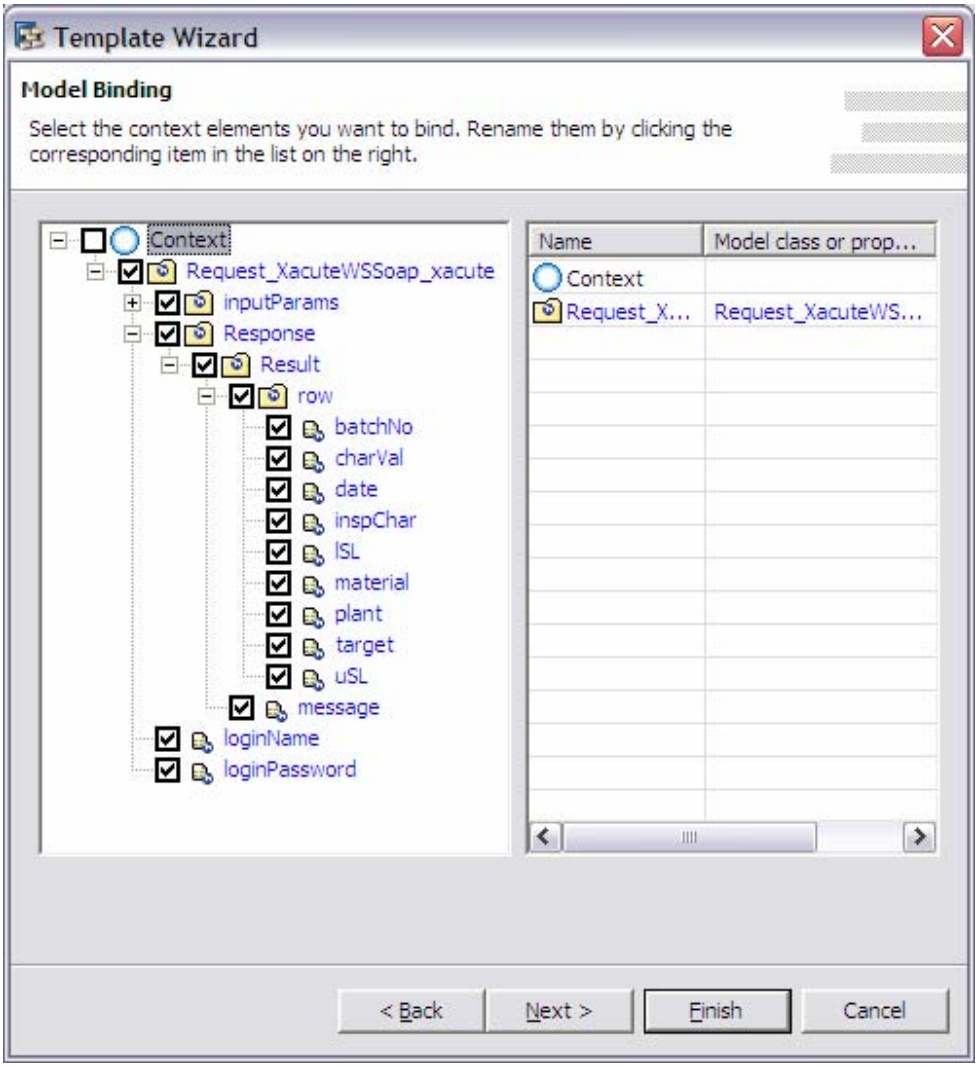


Select the model class and click on Next.

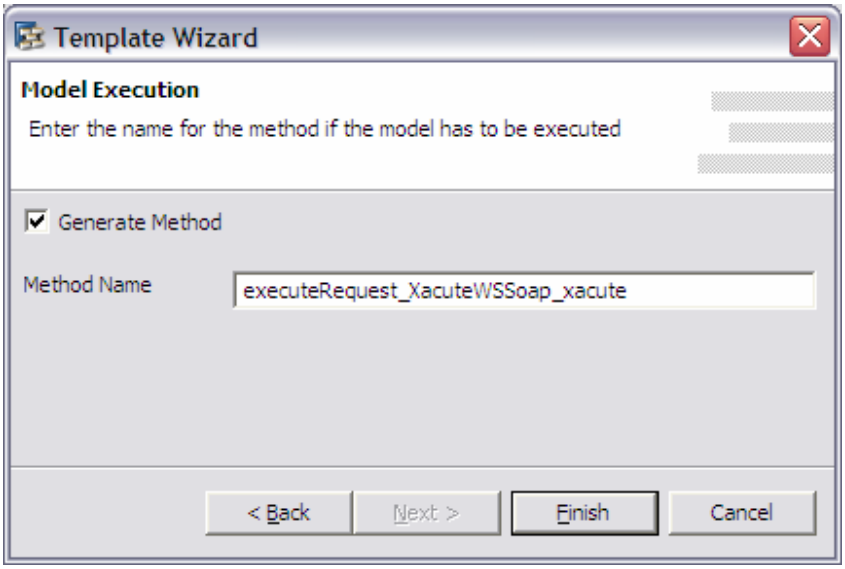


Select the required context elements from the model parameter and click on Next.

You must select loginName and loginPassword attributes. This is due to the fact that SAP xMII doesn't support basic authentication and we need to pass the login username and password as web service parameters only.

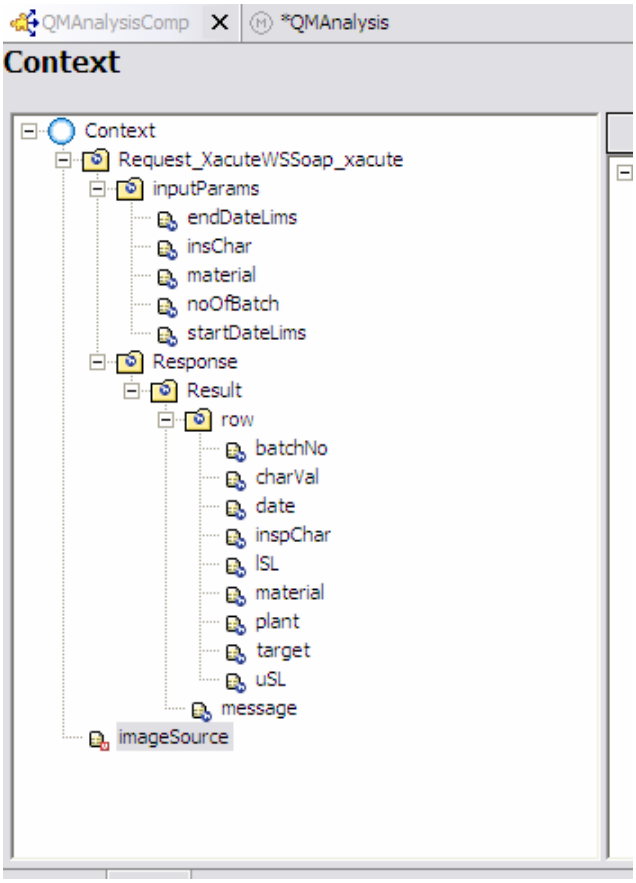


Select the Generate method option. You can also change the name of the method if you wish. This will be the controller method created by the wizard to invoke the model object's execute method (calling web service). Click on Finish.



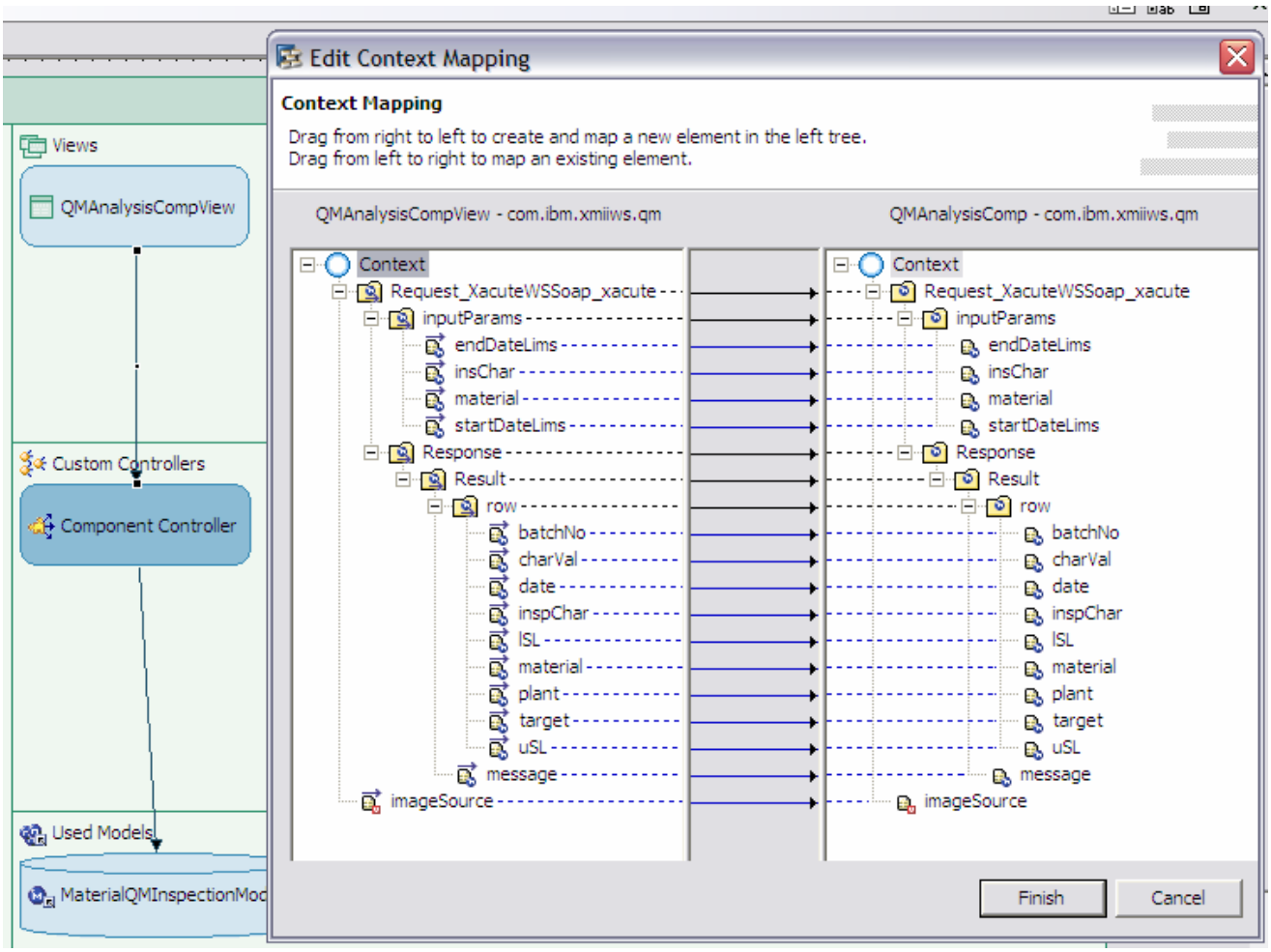
Create Context Attribute for SAP xMII iChart Object Image Source

We also want to display a SAP xMII iChart object in the UI. For this we need to create another context attribute to hold the iChart image URL.
Open the controller's context and create a new context attribute (value attribute) called imageSource.



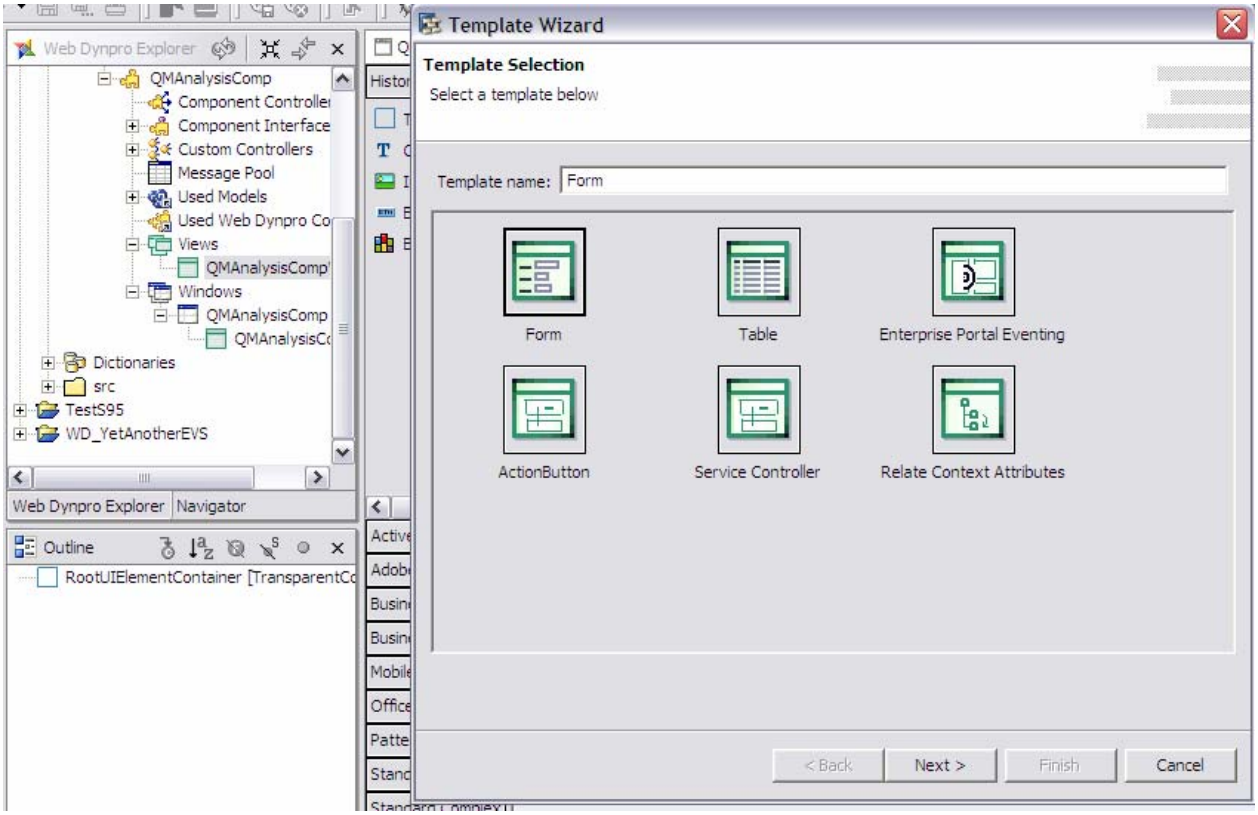
Map Controller and View Context

We need to map the controller's context to the view context before we create the view layout elements. Open the component and create a mapping link between the view and the controller and map the context elements as below.

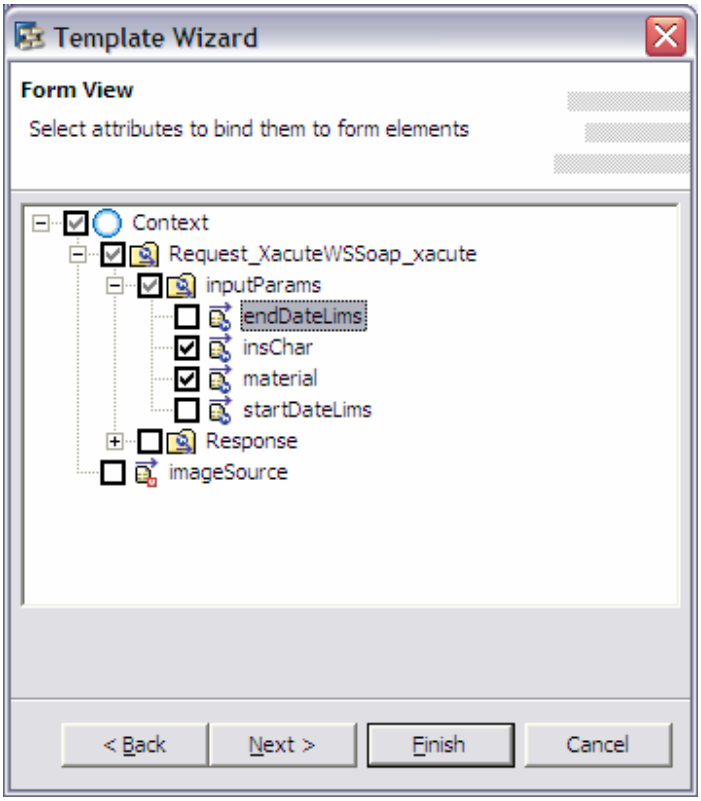


Adding UI Elements to View

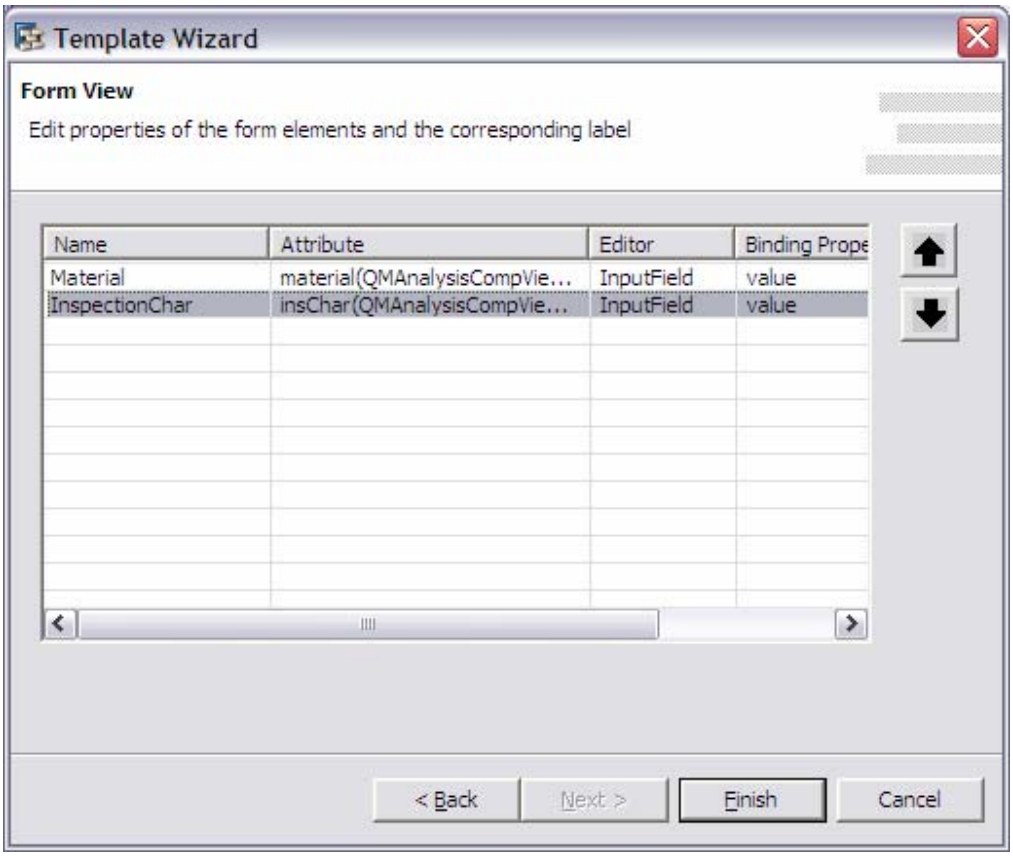
Now we'll create the UI elements in the default view using the Template wizard. Select the RootUIElementContainer and right-click and select apply template. Select the Form option and click Next.



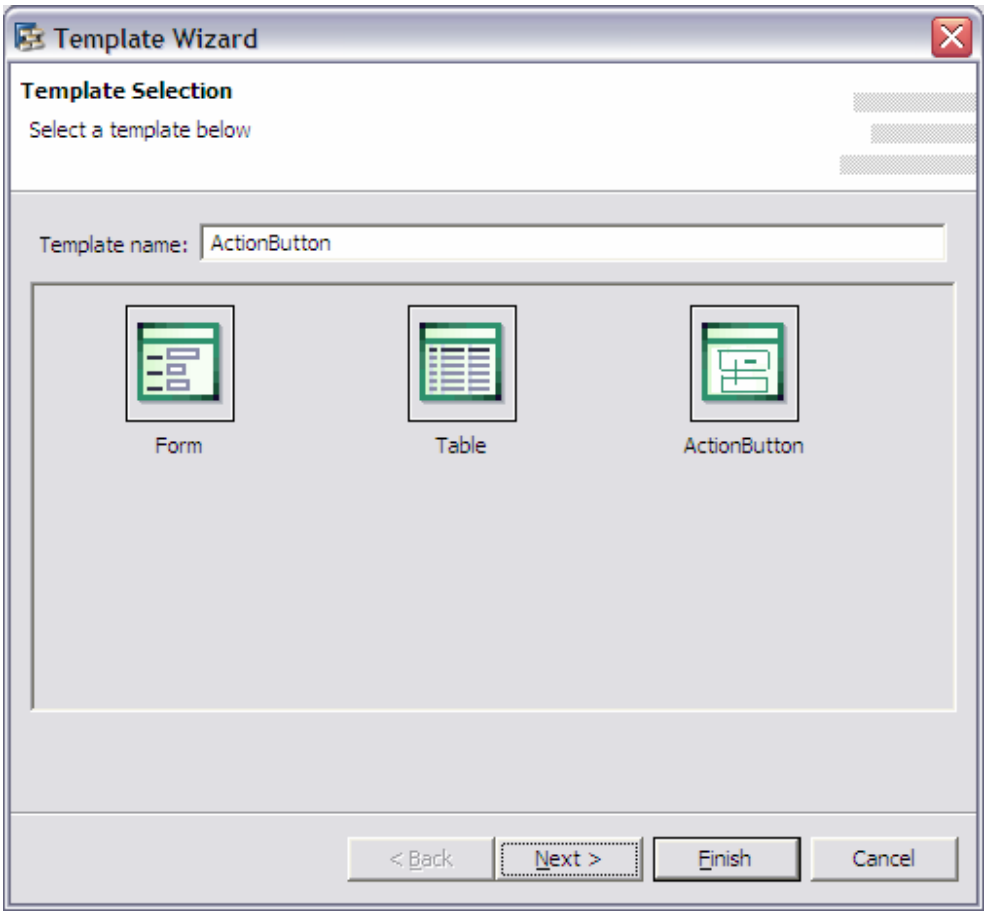
Select the required context elements and click Next.
You cannot select the date elements here because while creating the Web service model a XML dateTime field is converted to java.util.calendar in the Web Dynpro proxy class. As java.util.calendar is a complex data type you can't link that to an input field (textbox). If you want to use the date fields then we need to manually create the date context elements of type java.sql.Timestamp and then link them to the corresponding input fields. In the controller method for executing the web service you can write a line of code to read the date fields and then assign them to the corresponding model context attributes.



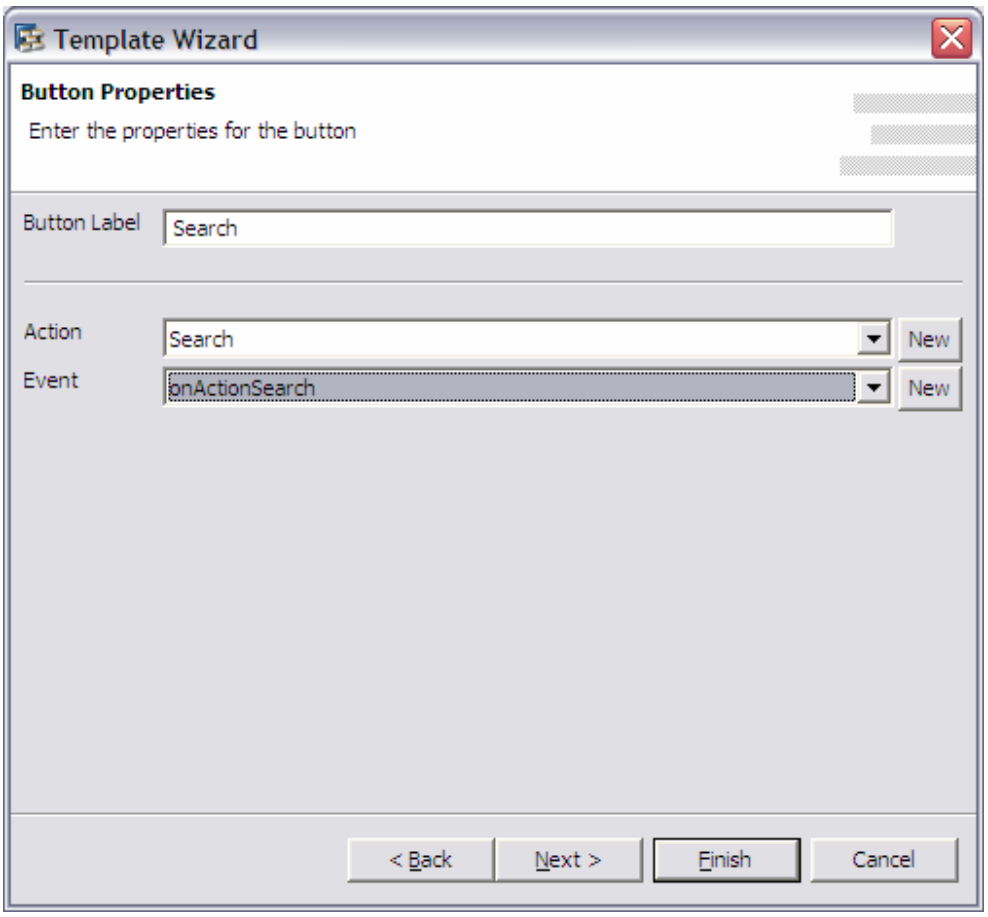
In the next screen you can modify the field names and labels and order of display. Click on Next.



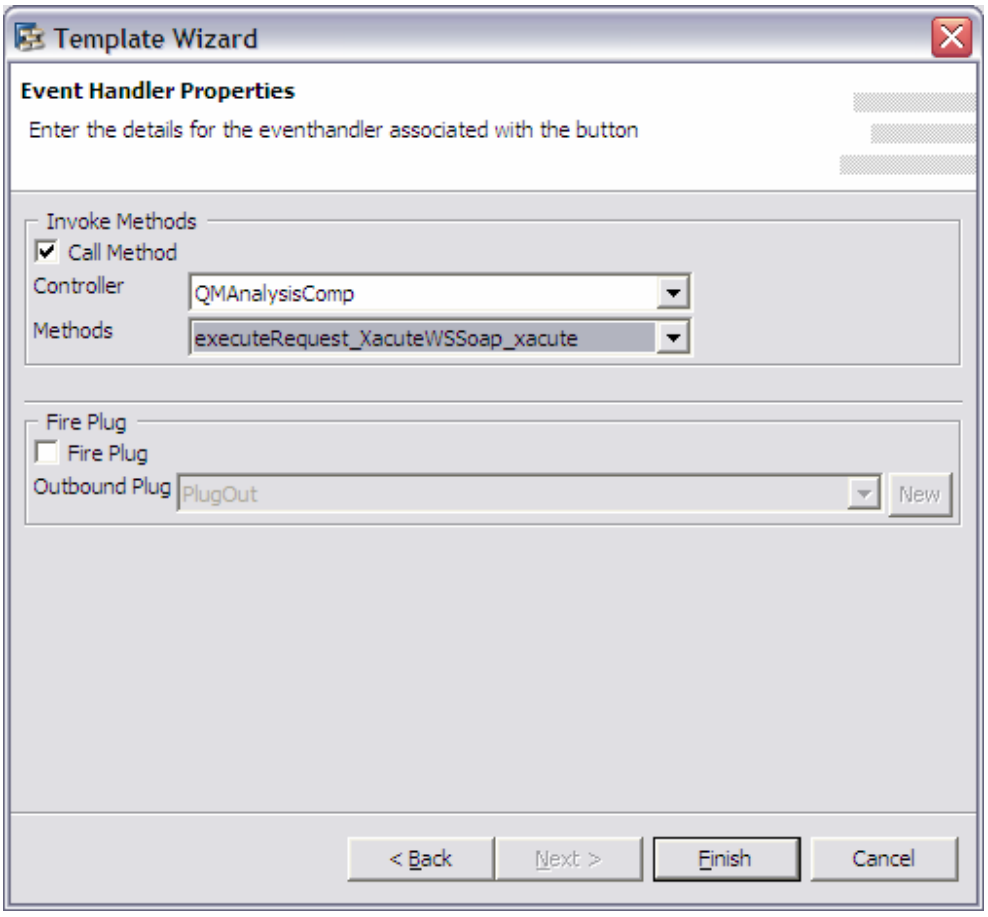
Now we have to add an action button to the view. Select the view and click on Apply Template option. Select the ActionButton option and click Next.



Enter the label of the action button and click Next. Automatically the wizard will select the Action and Event to be linked with the button click.

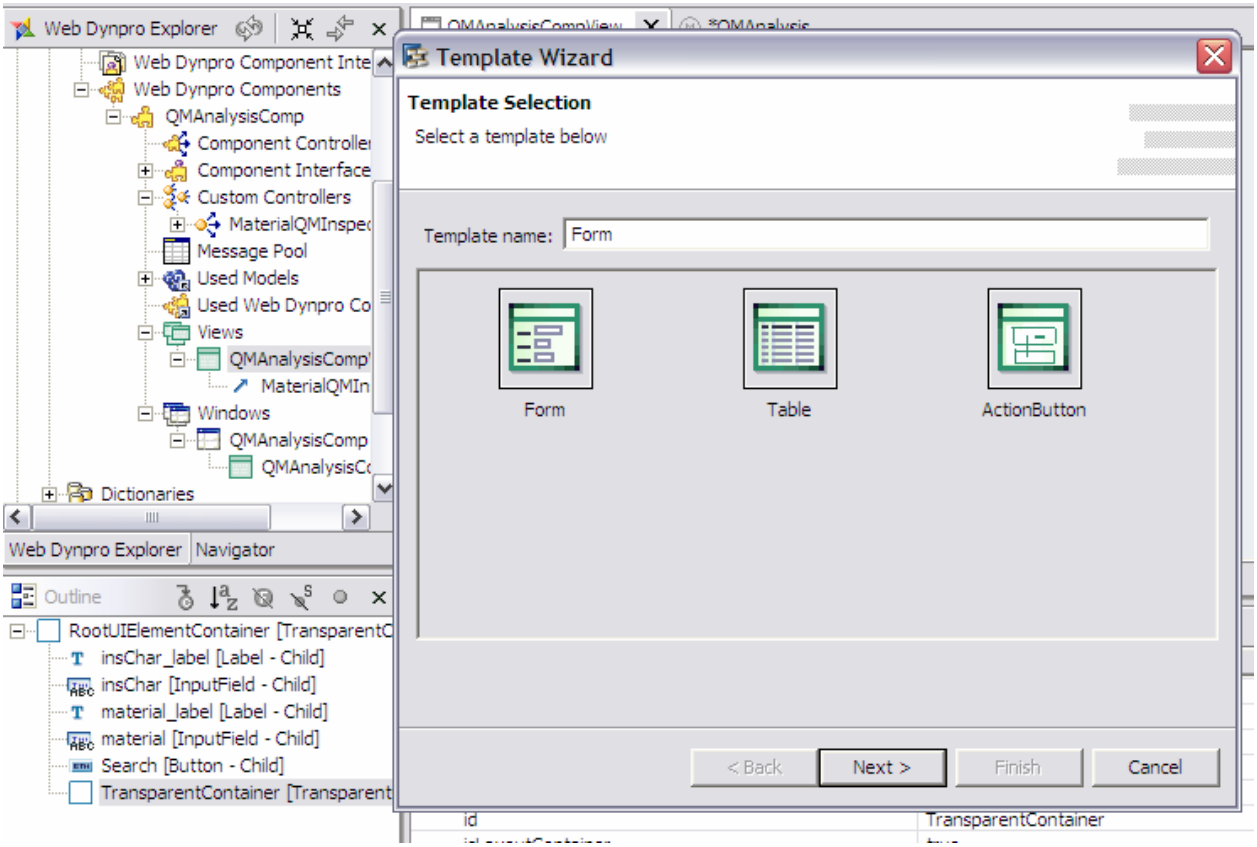


Select the Controller and the execute method and click Finish.

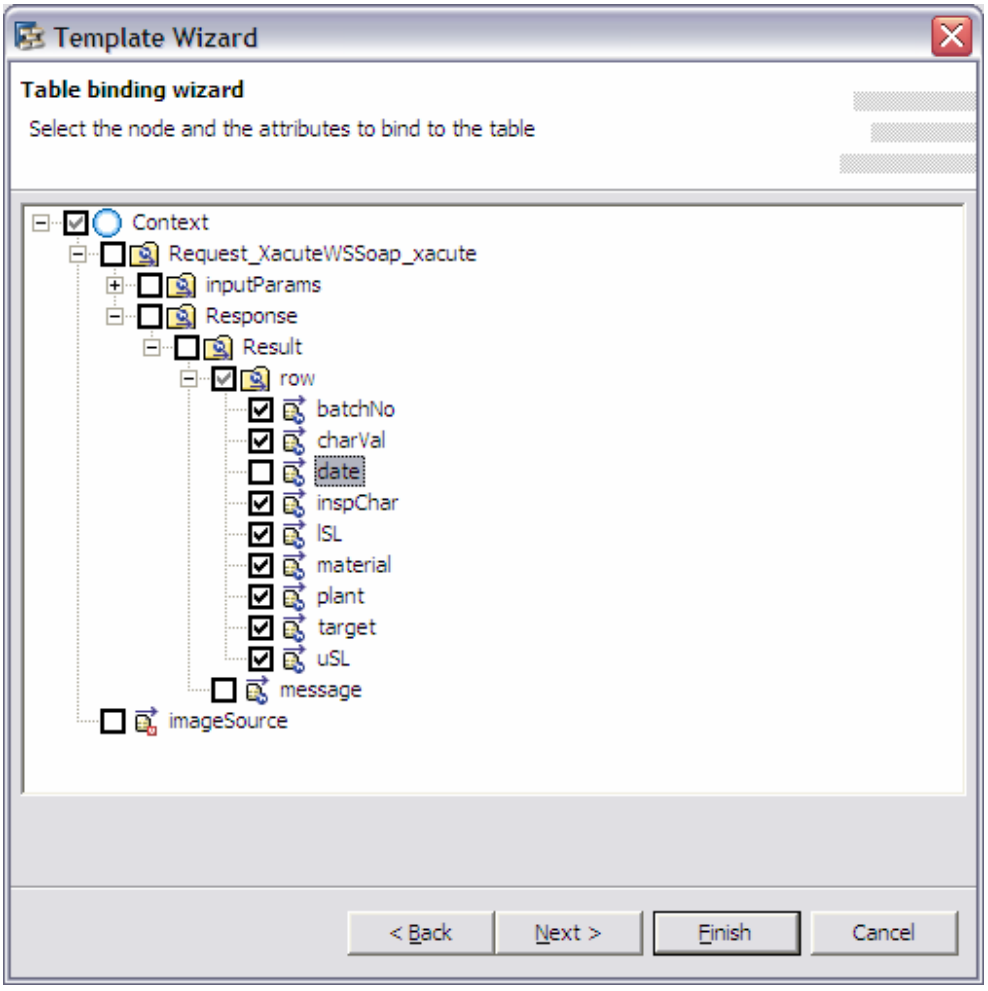


The wizard will add the action button to the view and generate the necessary code for it.

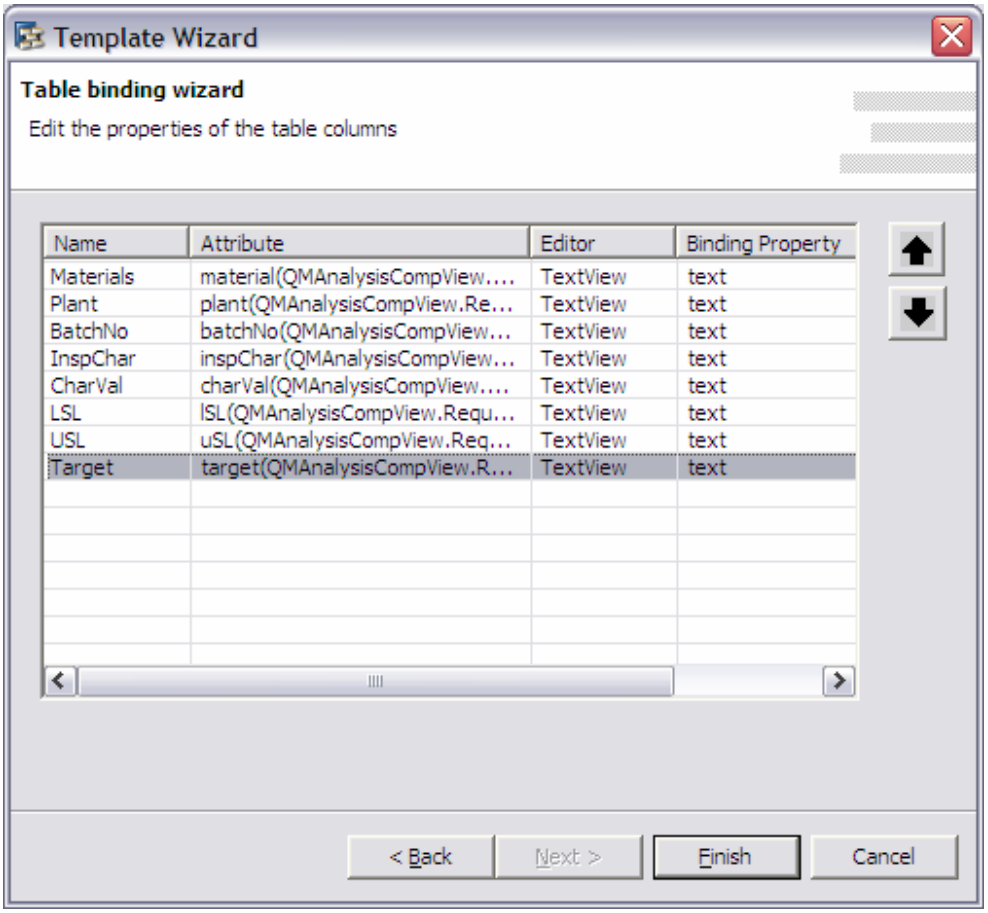
Now we need to add a table to the view for displaying the output data. To do this we need to again apply a Template to the view of type table as below.



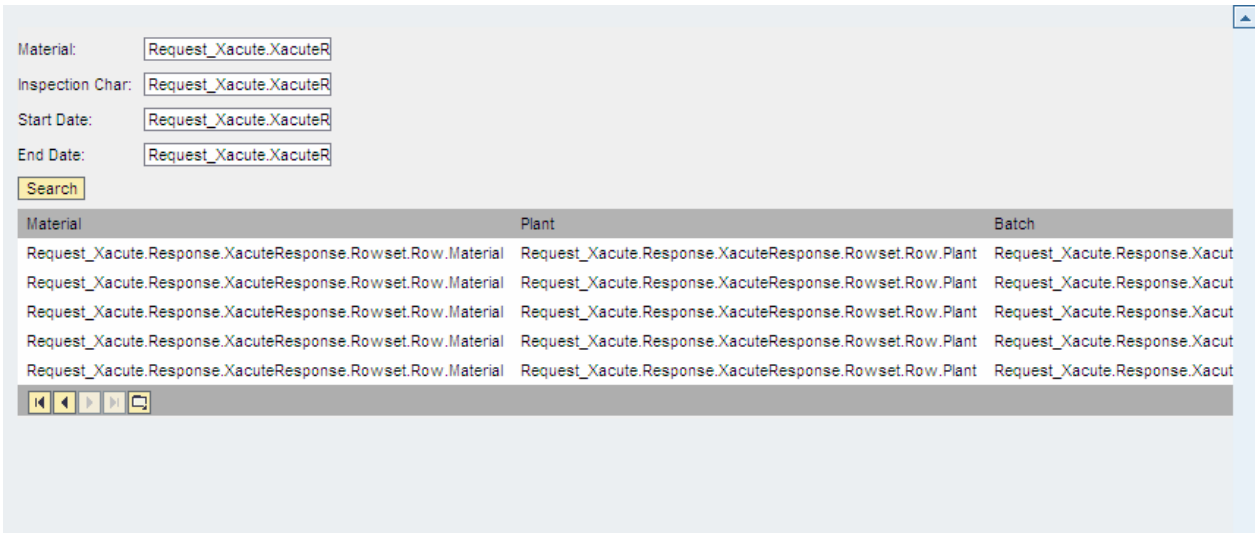
Select the required context attributes to be bound to the table output.
Again here also you can't select the date element to be bound to the table column. Instead if you want to display the dates you need to create a new context attribute of type java.sql.Timestamp, bind it to the table output and then have to write a line of code in the controller execute method to assign the model output to the custom context attribute.



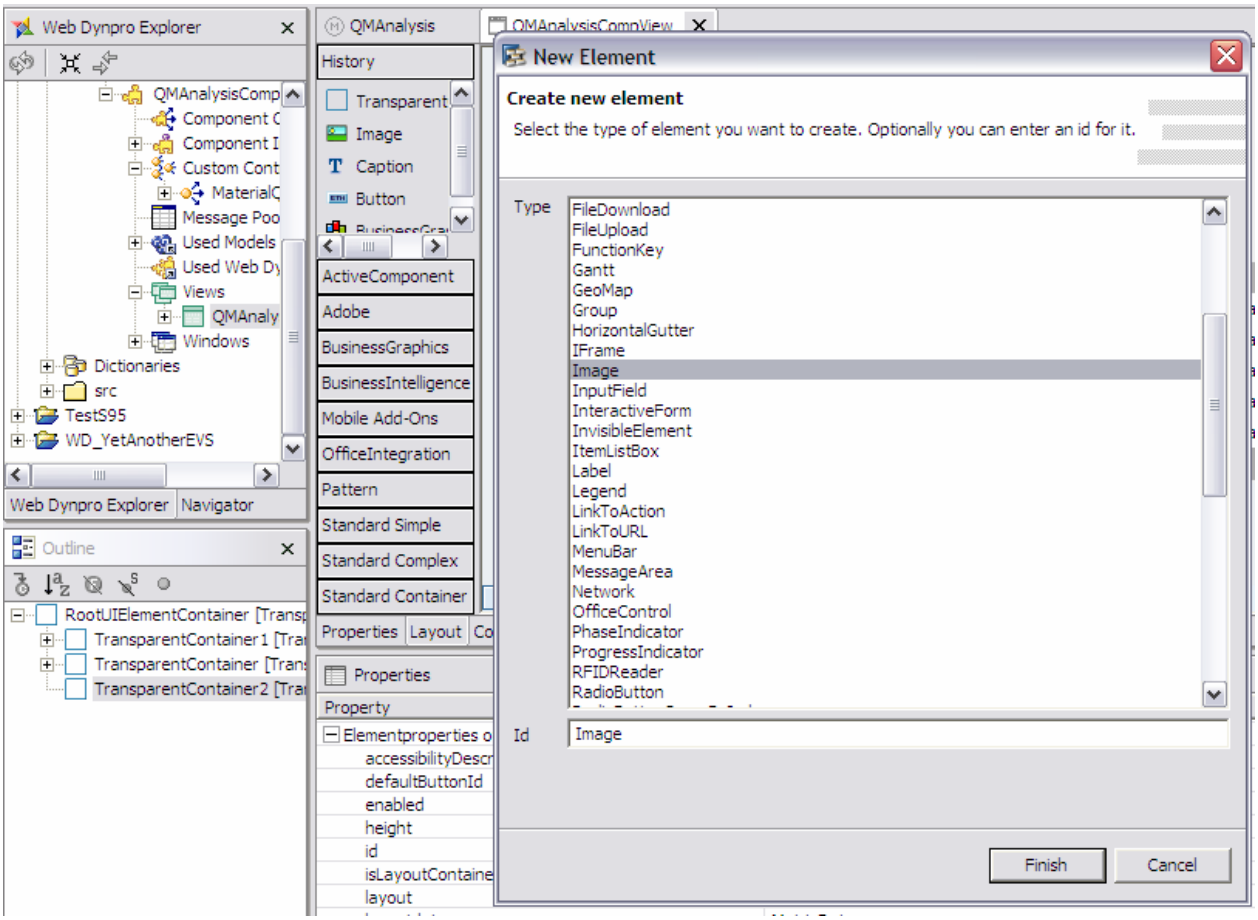
In the next screen you can modify the table column names, arrange the order of columns and select the output type. Click on Finish.



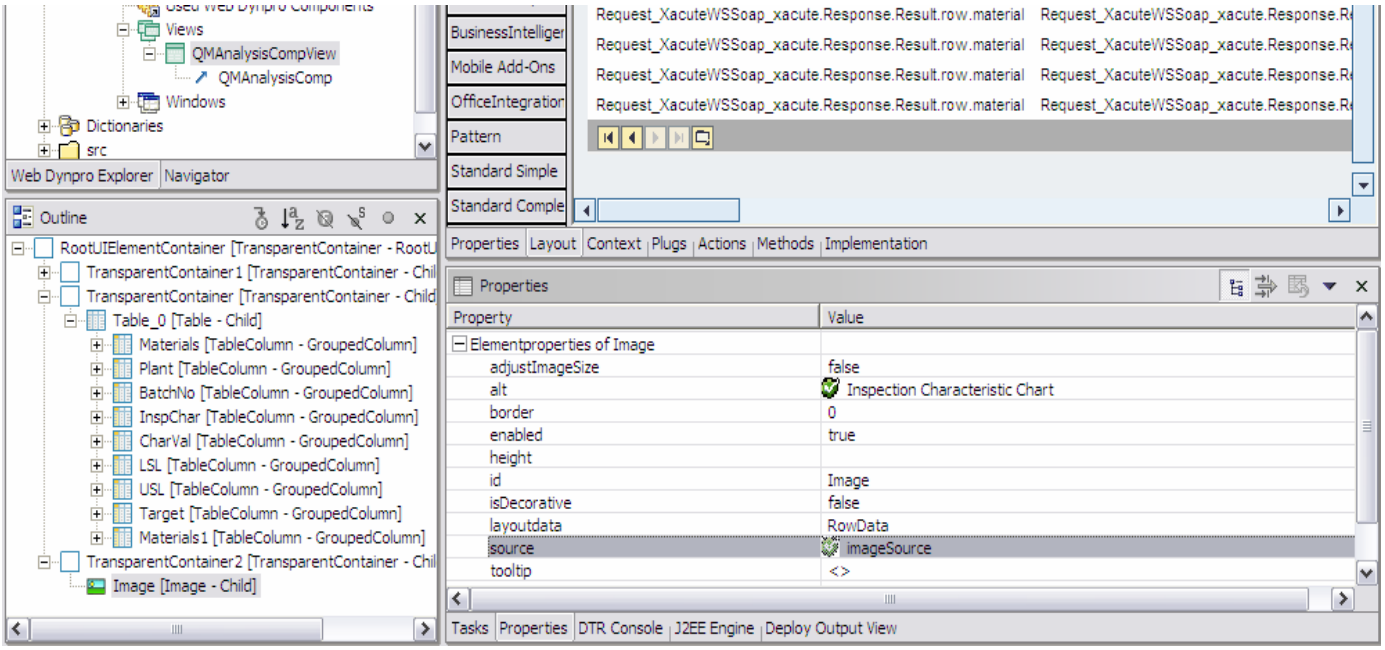
Now the view will look as below.



You also need to add an image output element to display the xMII chart. Add a new Image element below the table.



The image added should be bound to a context attribute which will provide it's image location. Bind the image source property to the imageSource context attribute.



Modifying the Method

Open the Implementation tab of the view and go to the onActionSearch method. This is the method called when the action button on the view is clicked. Notice that the code for calling the controller's execute method has been automatically added by the template wizard.

```
//View Method
  //@@begin javadoc:onActionSearch(ServerEvent)
  /** Declared validating event handler. */
  //@@end
  public void onActionSearch(com.sap.tc.webdynpro.progmodel.api.IWDCustomEvent
wdEvent )
  {
    //@@begin onActionSearch(ServerEvent)
    //$$begin ActionButton(-223265440)
      wdThis.wdGetQMAnalysisCompController().executeRequest_XacuteWSSoap_xacute();
    //$$end
    //@@end

  }
```

Go to the controller and open the Implementation tab. Go to the executeRequest_XacuteWSSoap_xacute() method. Notice that the code for calling the model object's web service has already been added by the template wizard. You need to add a few lines of code here.

As SAP xMII doesn't support BASIC authentication you need to pass the logon information programmatically along with the web service parameters.

Also as you need to display the chart plotted by SAP xMII we need to get the URL of the image source and assign it the context attribute. Any SAP xMII iChart object can be accessed by the following URL as a gif image:

http://<server>:<port>/Lighthammer/ChartServlet?Width=<width pixel>&Height=<height pixel>&QueryTemplate=<query template name>&DisplayTemplate=<display template name>&Content-Type=image/gif&IllumLoginName=<login-name>&IllumLoginPassword=<login-password>&Param.1=<param 1 for query template>

So after the execution of modelobject get the endpoint address i.e. the web service execution URL. Typically any web service (BLS transaction) of SAP xMII can be executed by the following URL:

http://<server>:<port>/Lighthammer/SOAPRunner/<Transaction name>

So the end point address returns the above URL with actual server name. You need to read the server name from this URL and construct the image source URL. By substring operations read the server name (and port also if present) and construct the URL to access the iChart object's image. Assign the new URL to the imageSource context attribute.

```
//Controller Method
  //@@begin javadoc:executeRequest_XacuteWSSoap_xacute()
  /** Declared method. */
  //@@end
  public void executeRequest_XacuteWSSoap_xacute( )
  {
    //@@begin executeRequest_XacuteWSSoap_xacute()
    //$$begin Service Controller(-1387999182)
    IWDMessageManager manager = wdComponentAPI.getMessageManager();
    try
    {
      wdContext.currentRequest_XacuteWSSoap_xacuteElement().setLoginName("demo2");

wdContext.currentRequest_XacuteWSSoap_xacuteElement().setLoginPassword("demoxm2");


wdContext.currentRequest_XacuteWSSoap_xacuteElement().modelObject().execute();
      wdContext.nodeResponse().invalidate();
      wdContext.nodeResult().invalidate();
      wdContext.nodeRow().invalidate();

        //get endpoint address
        String strDestination = (String)
MaterialQMInspectionModel.getServiceImpl().getXacuteWSSoap()._getProperty(Material
QMInspectionModel.getServiceImpl().getXacuteWSSoap().ENDPOINT_ADDRESS_PROPERTY);
        strDestination = strDestination.substring(7, strDestination.indexOf('/',
7));

        //construct image source URL
        String imageSource = "http://" + strDestination +
"/Lighthammer/ChartServlet?Width=400&Height=300&QueryTemplate=UserTemplates/PoC/Di
pankar/XAGetPlantICList&DisplayTemplate=UserTemplates/PoC/Dipankar/iChartPlantICLi
st&Content-
Type=image/gif&IllumLoginName=demo2&IllumLoginPassword=demoxm2&Param.1="
            + (String)
wdContext.currentInputParamsElement().getAttributeValue("material")
```



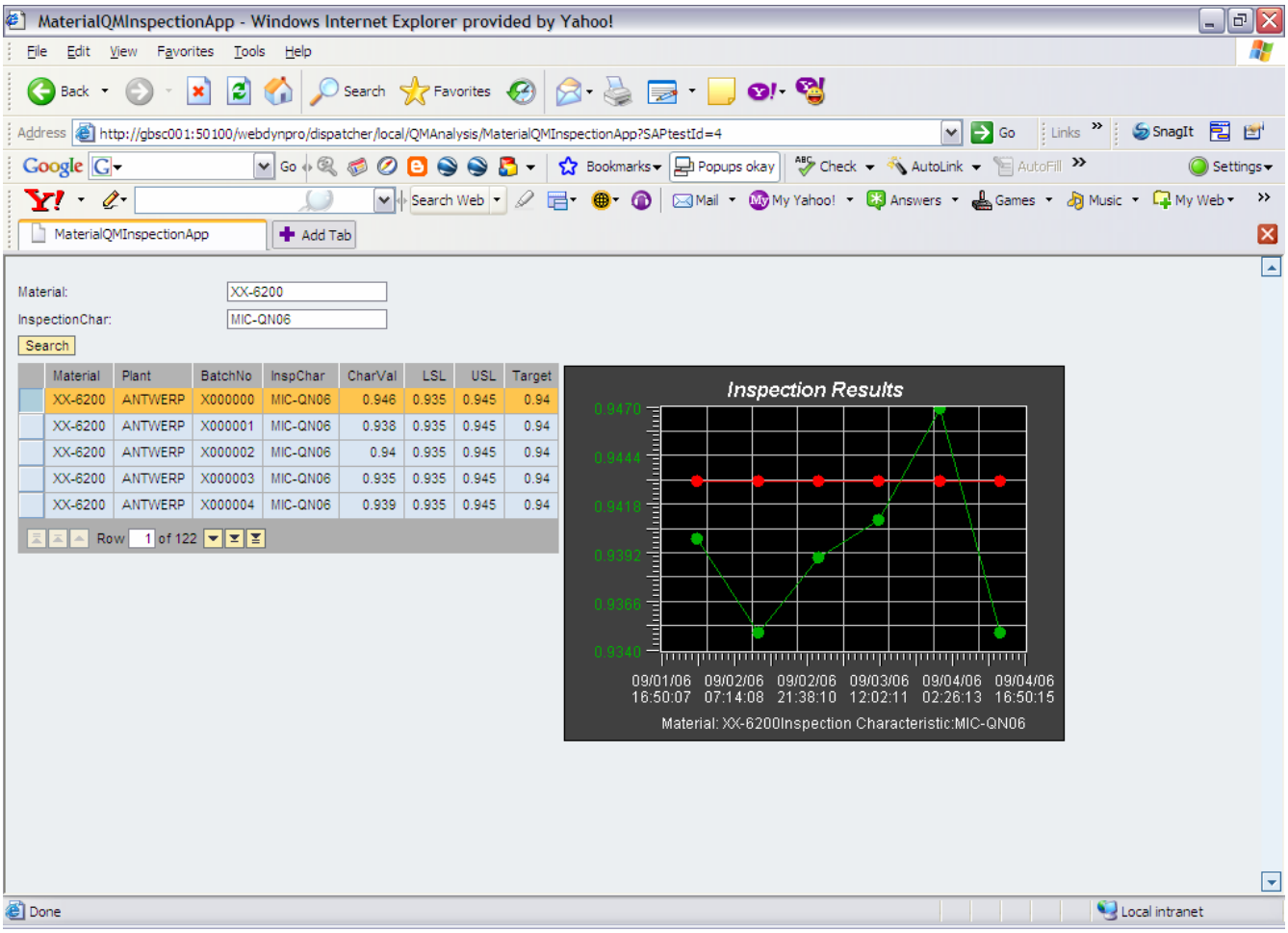
```
            + "&Param.2=" +(String)
wdContext.currentInputParamsElement().getAttributeValue("insChar");

        wdContext.getCurrentElement().setAttributeValue("imageSource",
imageSource);

    }
    catch(Exception e)
    {
      manager.reportException(e.getMessage(), false);
    }
    //$$end
    //@@end
  }
```

Execute the Web Dynpro Application

Build the Web Dynpro project, create archive, deploy and run the application. Enter the material and inspection characteristic name and click on Search. The web service (BLS Transaction) is called in SAP xMII and the data is retrieved which is displayed in the table below. Also the chart plotted by SAP xMII is displayed as a gif image.



Conclusion

SAP xMII composite application developers can leverage the Web Dynpro Java UI development framework to develop SAP xMII based composite applications using the xMII services as described above.

Related Content

- SAP XMII Getting Started Guide
- Calling Services and Queries in SAP xMII 11.5 from ABAP

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.