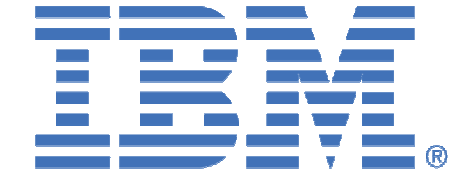


IBM Integration Bus

Creating an Integration Service

Featuring:

Creating schemas from a database
Placing the schemas in a Shared Library
Using the schemas in an Integration Service
Placing a Mapping Node in a Shared Library
Using the Mapping Node to read a database
Introduction to the Flow Exerciser

October 2015
Hands-on lab built at product
Version 10.0.0.2

1. Introduction

An integration service is a specialized application with a defined interface and structure that acts as a container for a Web Services solution.

You can integrate applications by using a service-oriented architecture (SOA). In an SOA, a service is often defined as a logical representation of a repeatable activity that has a specified outcome. Typically, a service is self-contained and its implementation is hidden from its consumers. To facilitate their reuse, services define a prescribed interface, which specifies how data is exchanged with the service.

In IBM Integration Bus, an integration service is a specialized application with a defined interface that acts as a container for a Web Services solution:

- It contains message flows to implement the specified web service operations.
- The interface is defined through a WSDL file.

1.1 Scenario Overview

In this lab, you will create an integration service and all the IBM integration resources required to implement the service. You will define the service data model and the WSDL that describes the service. You will implement and test the service operation.

The service accesses an external database and responds with data retrieved from the database. You will need to configure IBM Integration Bus to work with an external database system, in this lab, DB2

1.2 Outline of tasks

The tasks to complete in this lab are the following:

1. Configure IBM Integration Bus to work with DB2 (optional)
    a. Runtime configuration: Configure a JDBCProvider configurable service. Configure the security credentials to work with the database.
    b. Development configuration: Create a Database Definition project, and the database connectivity configuration.
2. Create a Shared Library. This library will contain your message models, and the WSDL that defines the new service.
3. Create a Message Model
    a. Create a message model based on a database table definition
    b. Extend the message model with other schema available in the library
4. Create an integration service, and define the WSDL associated with it
5. Implement the integration service operation
    a. Create the Message Map in the Shared Library
    b. Develop the Message Map to read the EMPLOYEE table
6. Test the service using the Integration Toolkit Flow Exerciser
7. Test the integration service using external tools (optional)
    a. Export the Service Interface definition from the IIB node
    b. Test the service using SOAPUI

1.3 Configure Integration Bus node to work with DB2

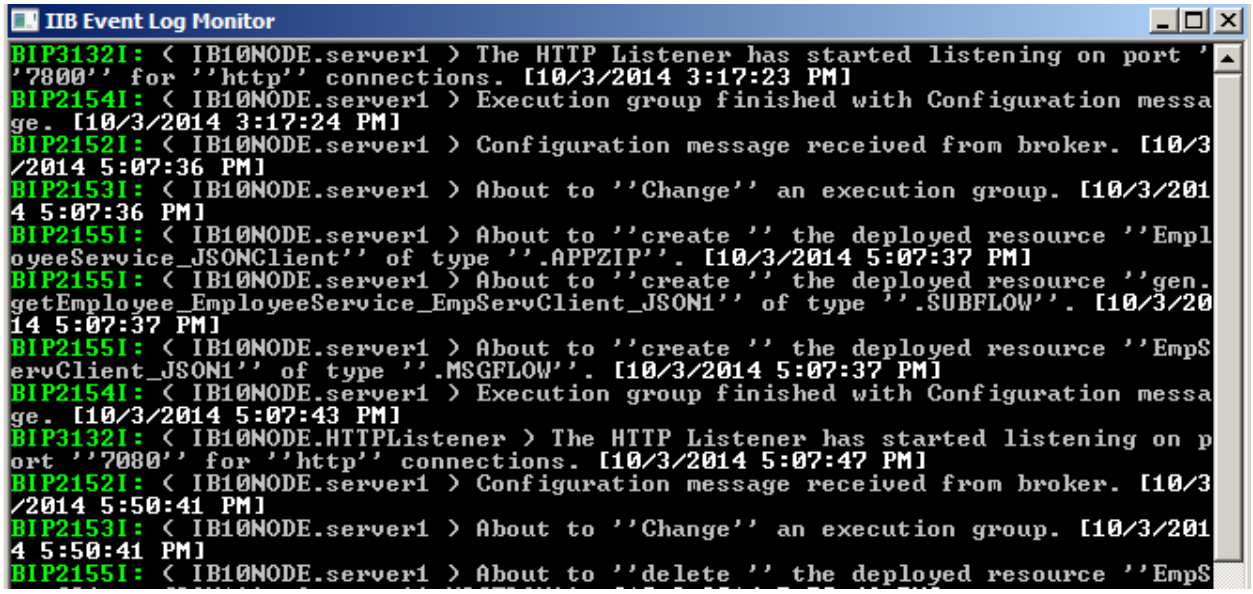
To run this lab, the Integration Bus node must be enabled to allow a JDBC connection to the SAMPLE database.

For this session, this configuration has already been done for you. You can look at the relevant script file in **c:\student10\common\DBSetup** if you want to review the commands for this.

1.4 Open the Windows Log Monitor for IIB

A useful tool for IIB development on Windows is the IIB Log Viewer. This tool continuously monitors the Windows Event Log, and all messages from the log are displayed immediately.

From the Start menu, click IIB Event Log Monitor. The Monitor will open; it is useful to have this always open in the background.



1.5 Create a Database Definition (optional)

This section is optional, but included to show you how this is done. If you don't do this section, you should import the pre-built database definition from the PI file

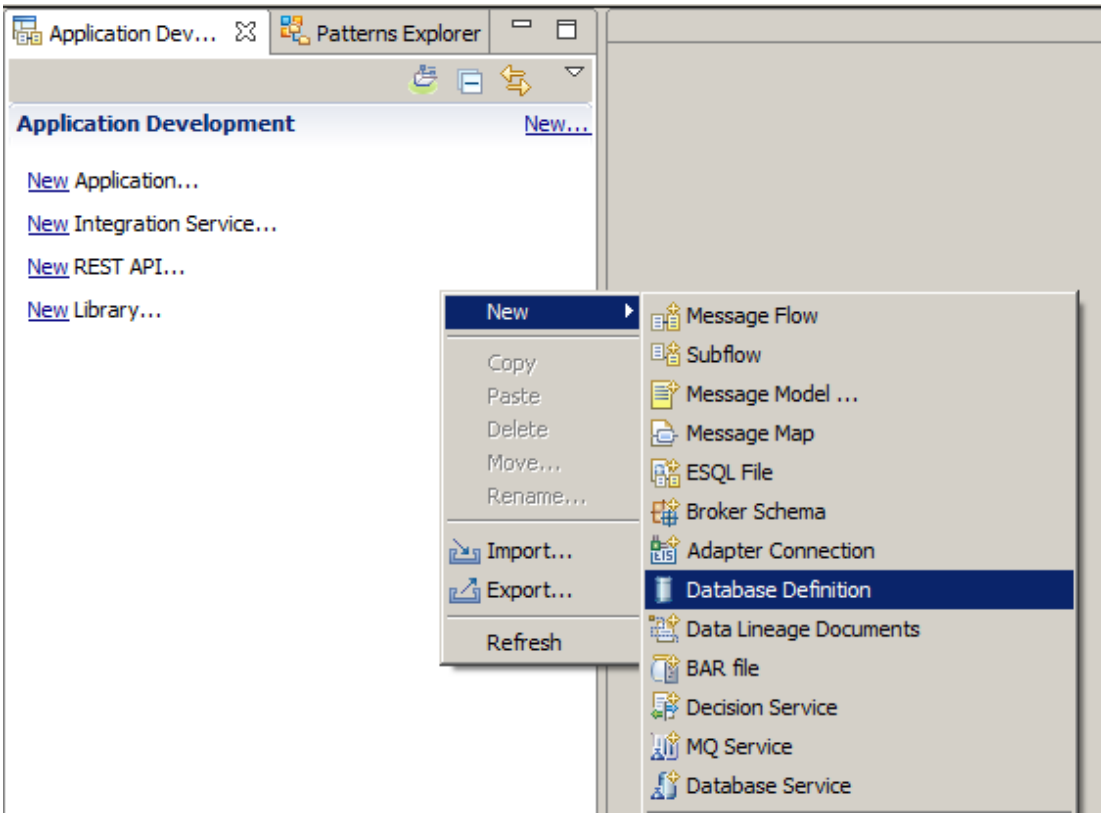
```
c:\student10\integration_service\resources\SAMPLE.zip
```

Invoke the import by right-clicking in the navigator, select Import, and choose Project Interchange file. Use the Browse button to locate the zip file.

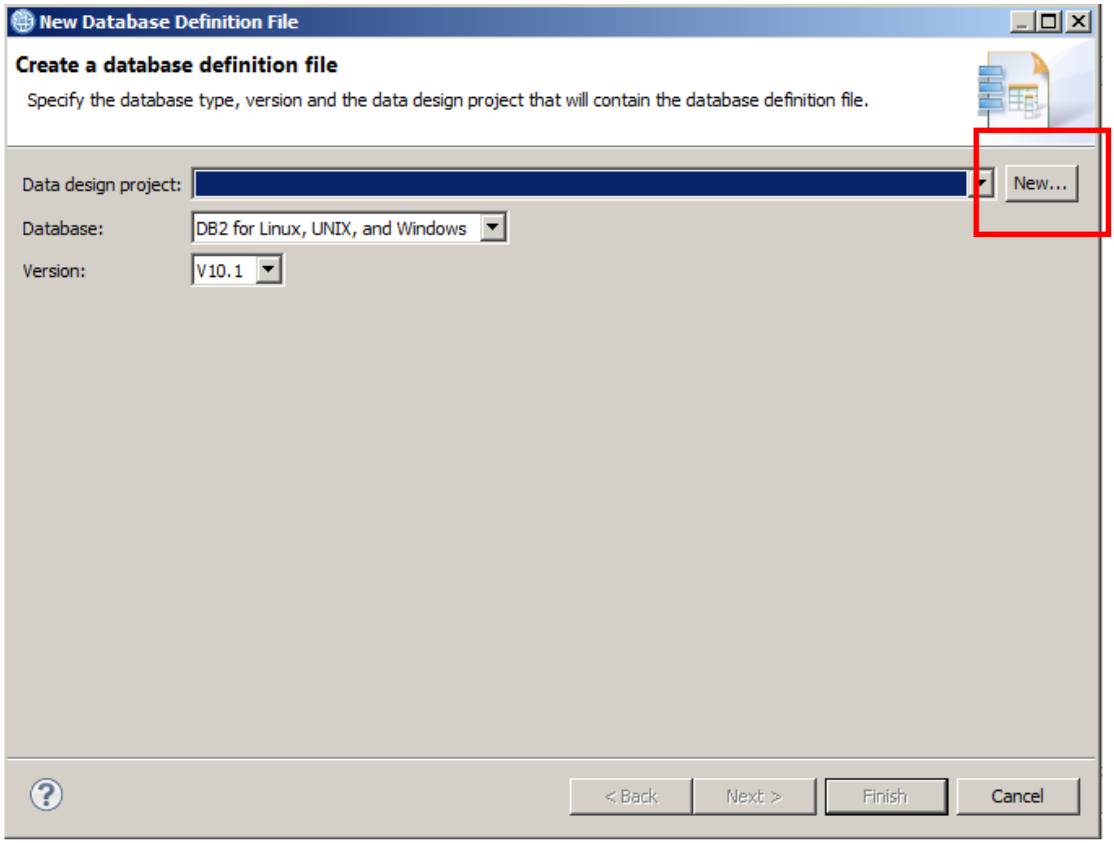
Then proceed to Create the schema on page 11.

Manual steps for creation of Database Definition

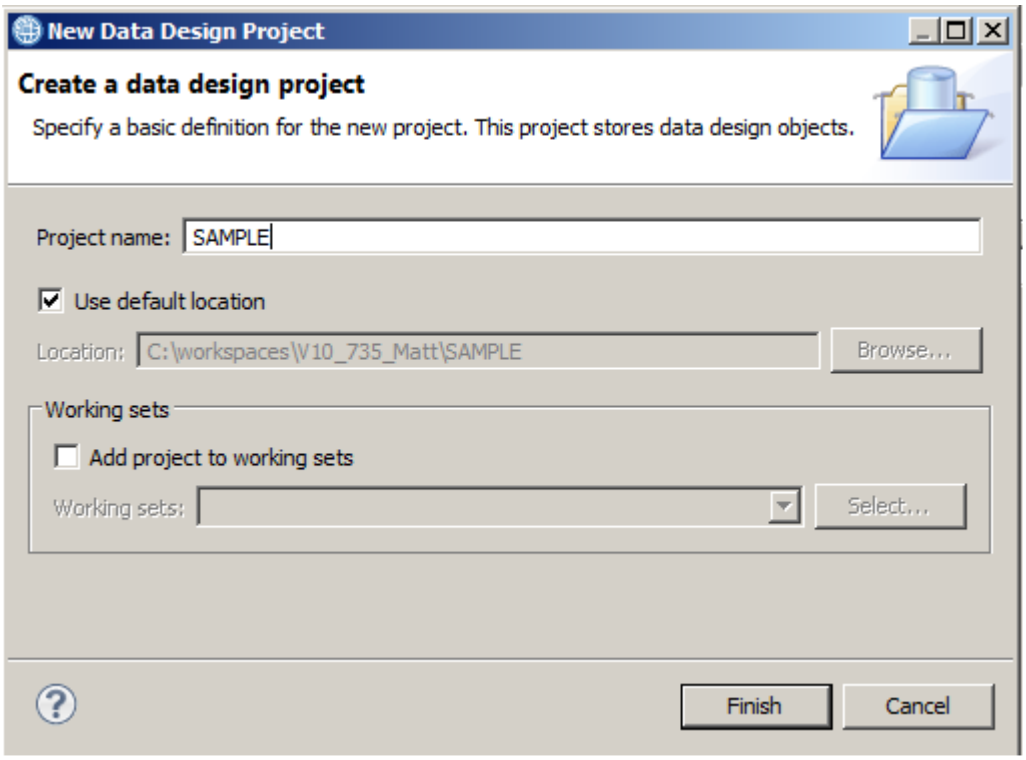
1. In the Toolkit navigator, right-click in blank space, select New, Database Definition.



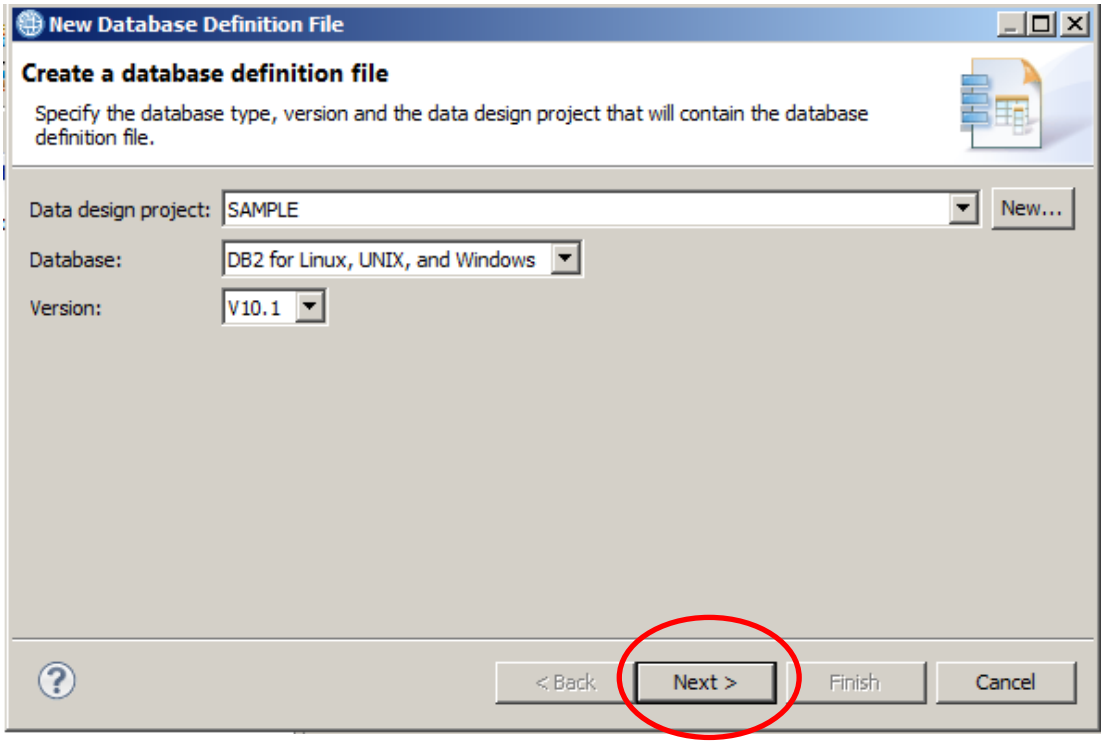
2. Click New to create a data design project.



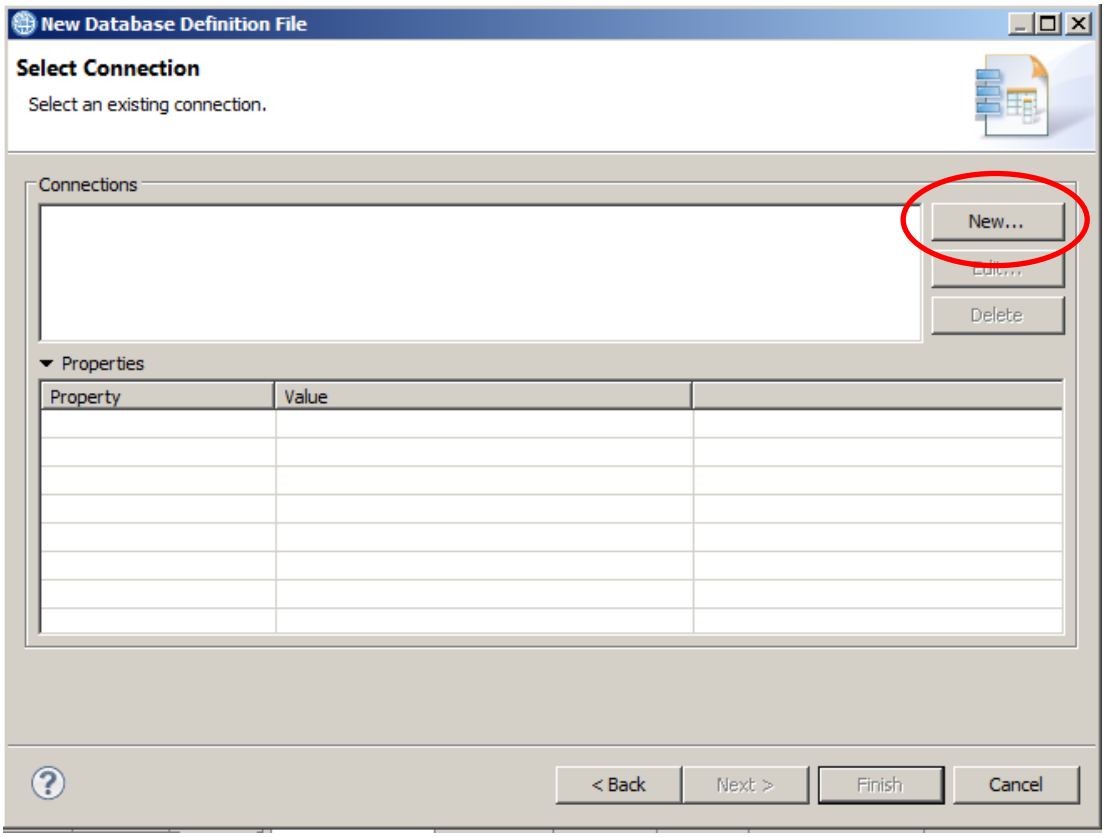
3. Name it SAMPLE. Click Finish.



4. Click Next.



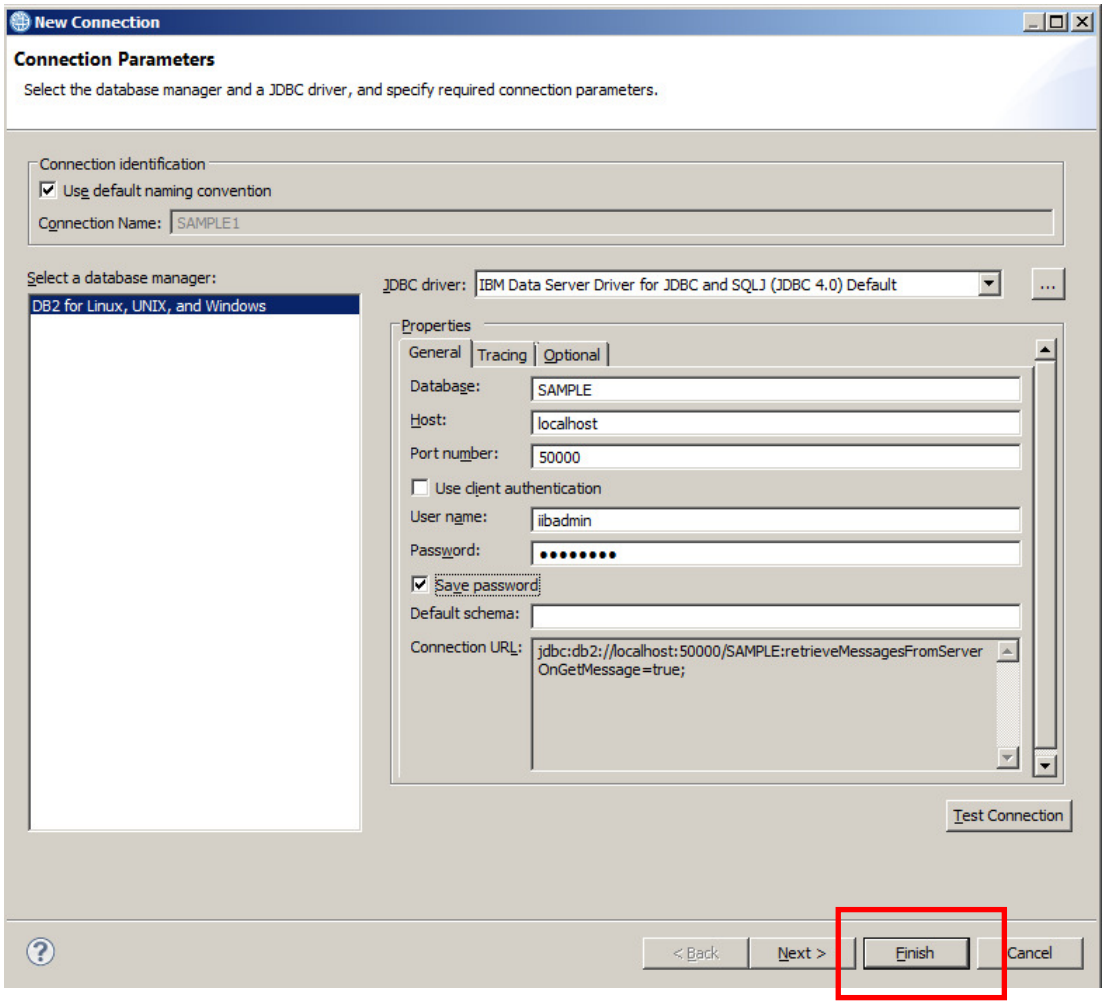
5. Click New to create a new connection.



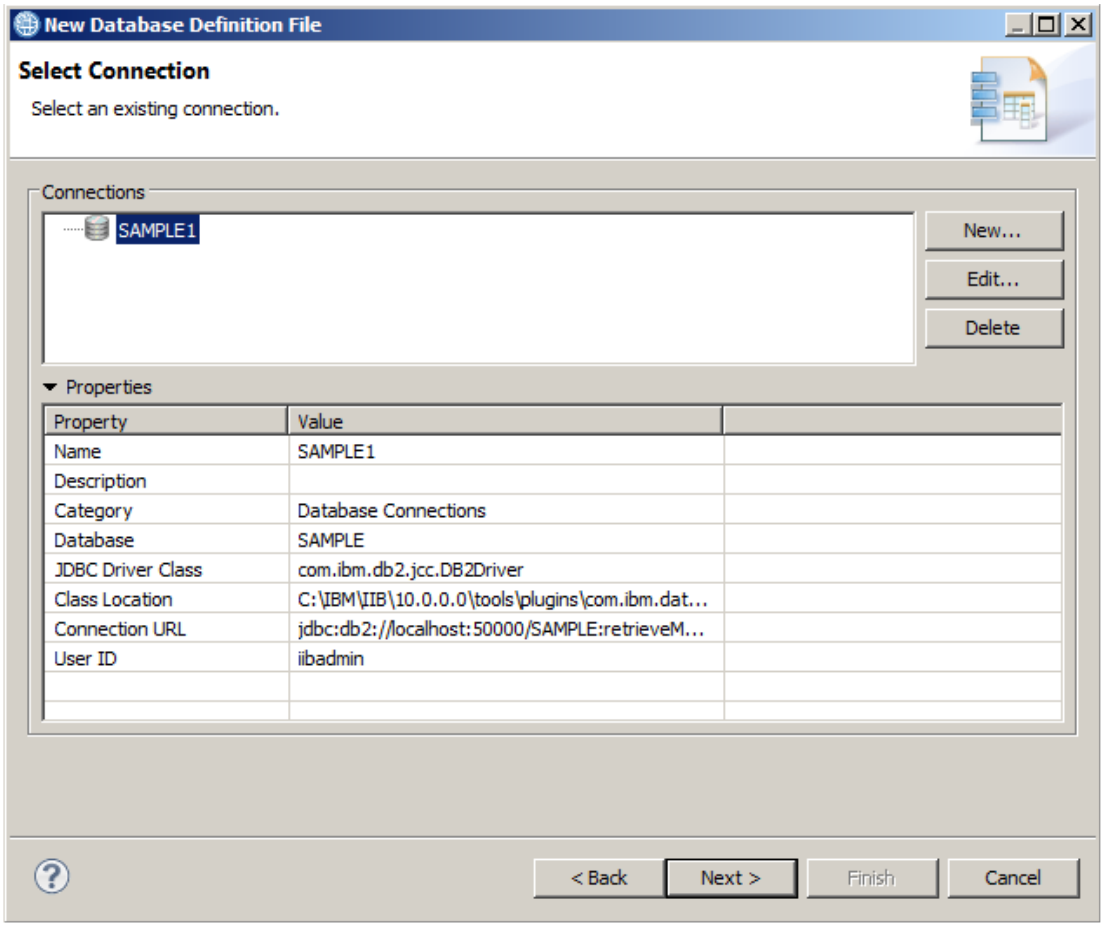
6. Specify connection credentials (user=iibadmin, password=passw0rd),

   Save the password, and click Finish.

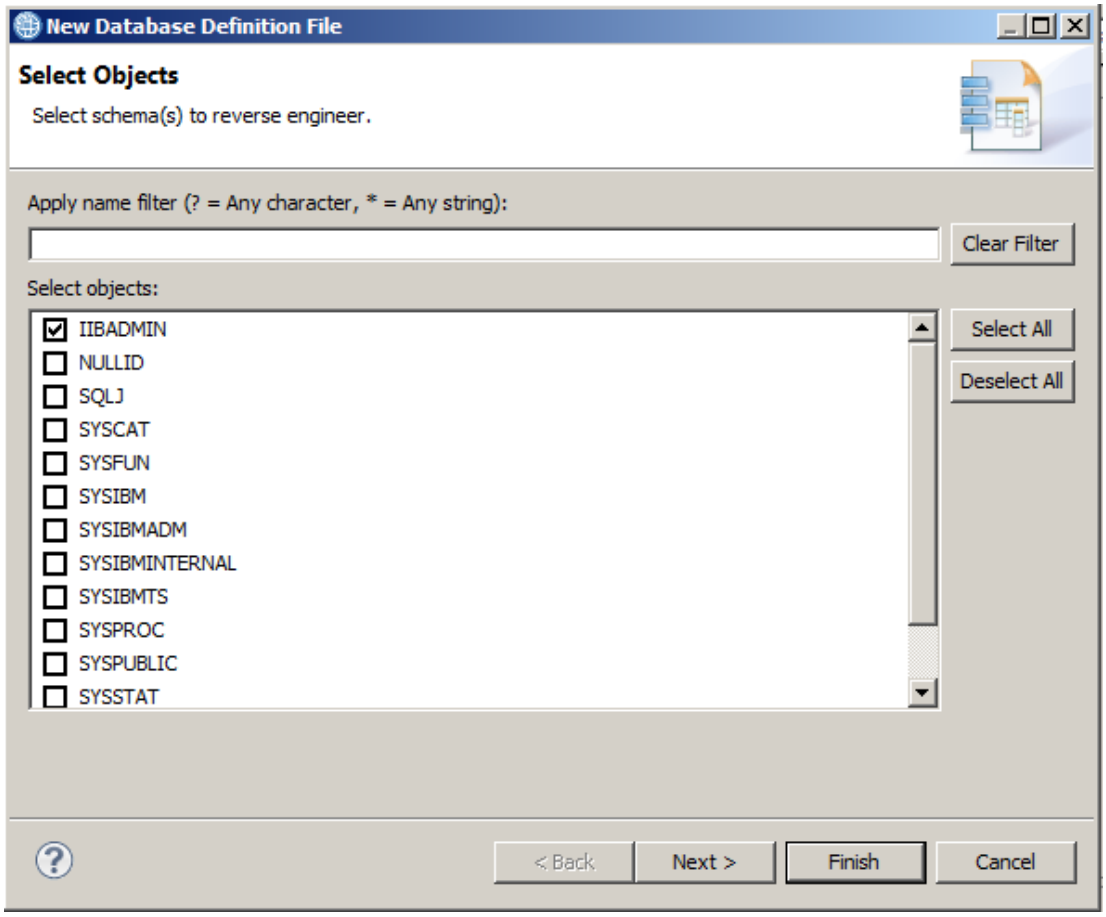
   (You may need to expand the dialog window to see the Save Password checkbox).



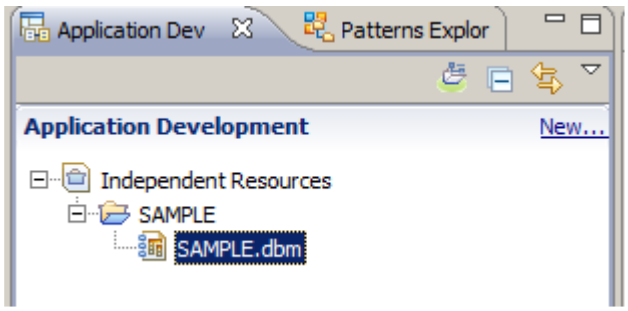
7. Highlight the new connection, and click Next.



8. Select the iibadmin schema, and click Finish.



9. The database definition for the SAMPLE database is now present as a project in the Independent Resources part of the navigator.

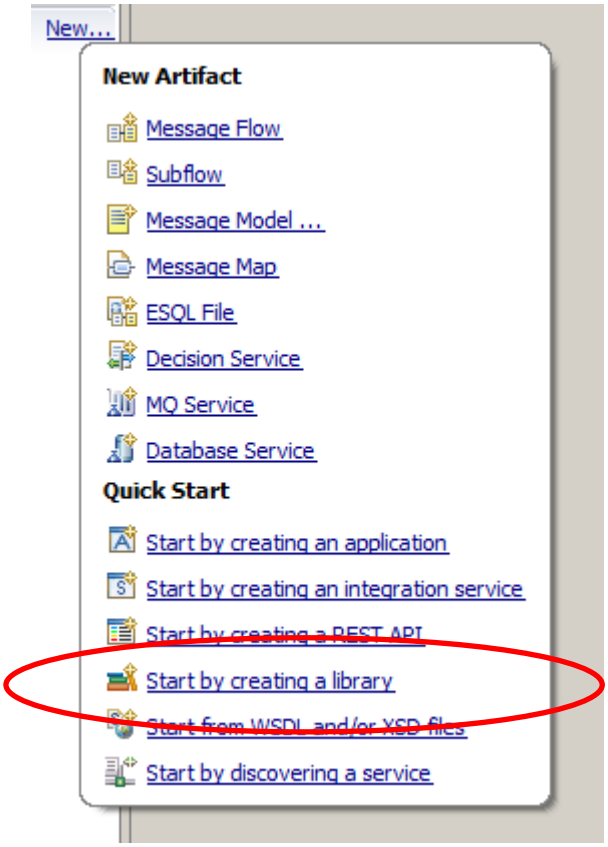


2. Create the schema

The schema will be derived from the DB2 SAMPLE database definition that was created in chapter 1. This lab will use the EMPLOYEE table, so you will generate just the EMPLOYEE schema.

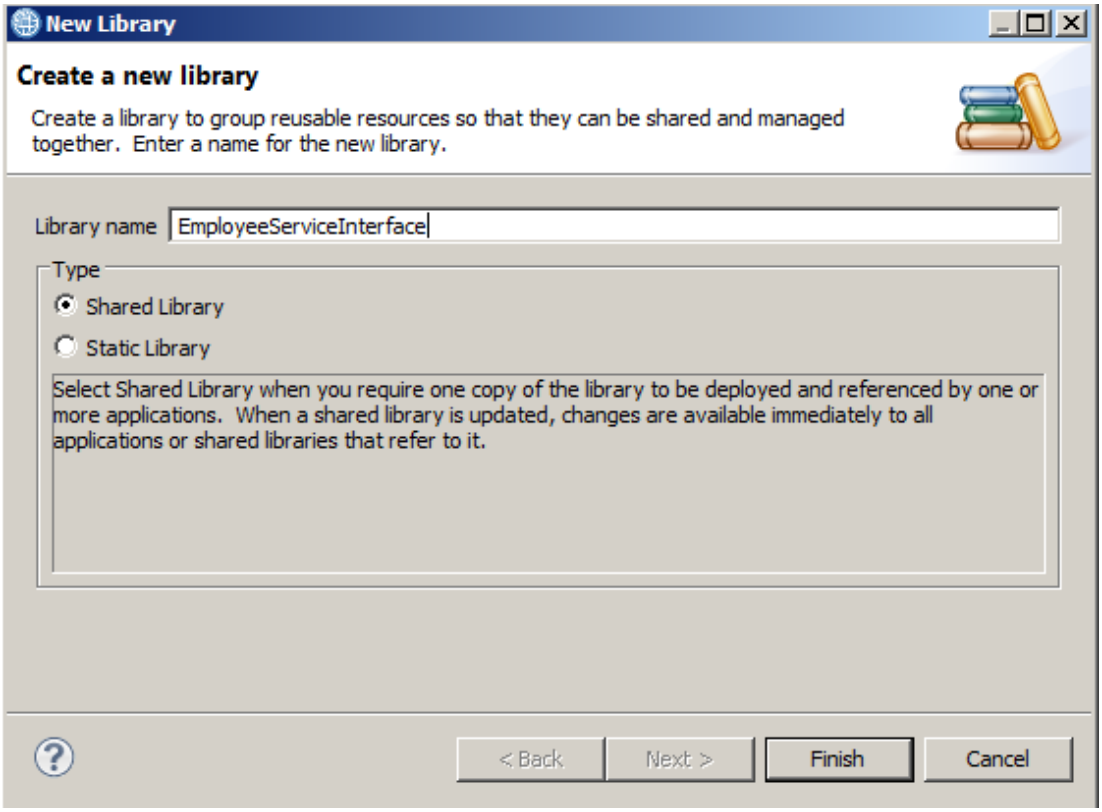
2.1 Create the Message Model (XML schema)

1. Create a new Library. In the Navigator view, click New, and select "Start by creating a library".

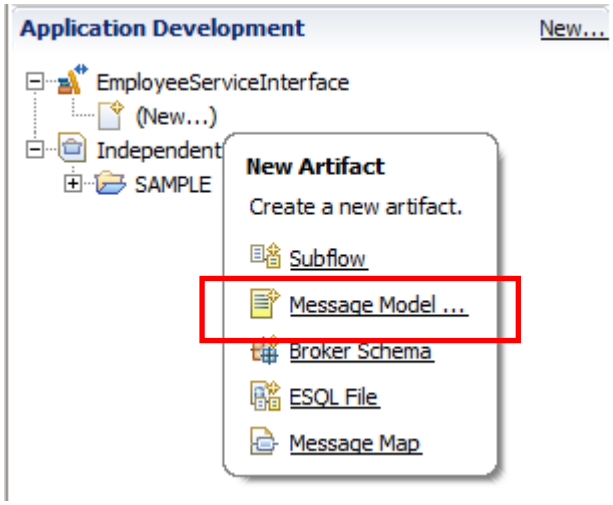


2. You can choose whether to create a Shared or Static library. Choose Shared (the default), and name the library EmployeeServiceInterface.

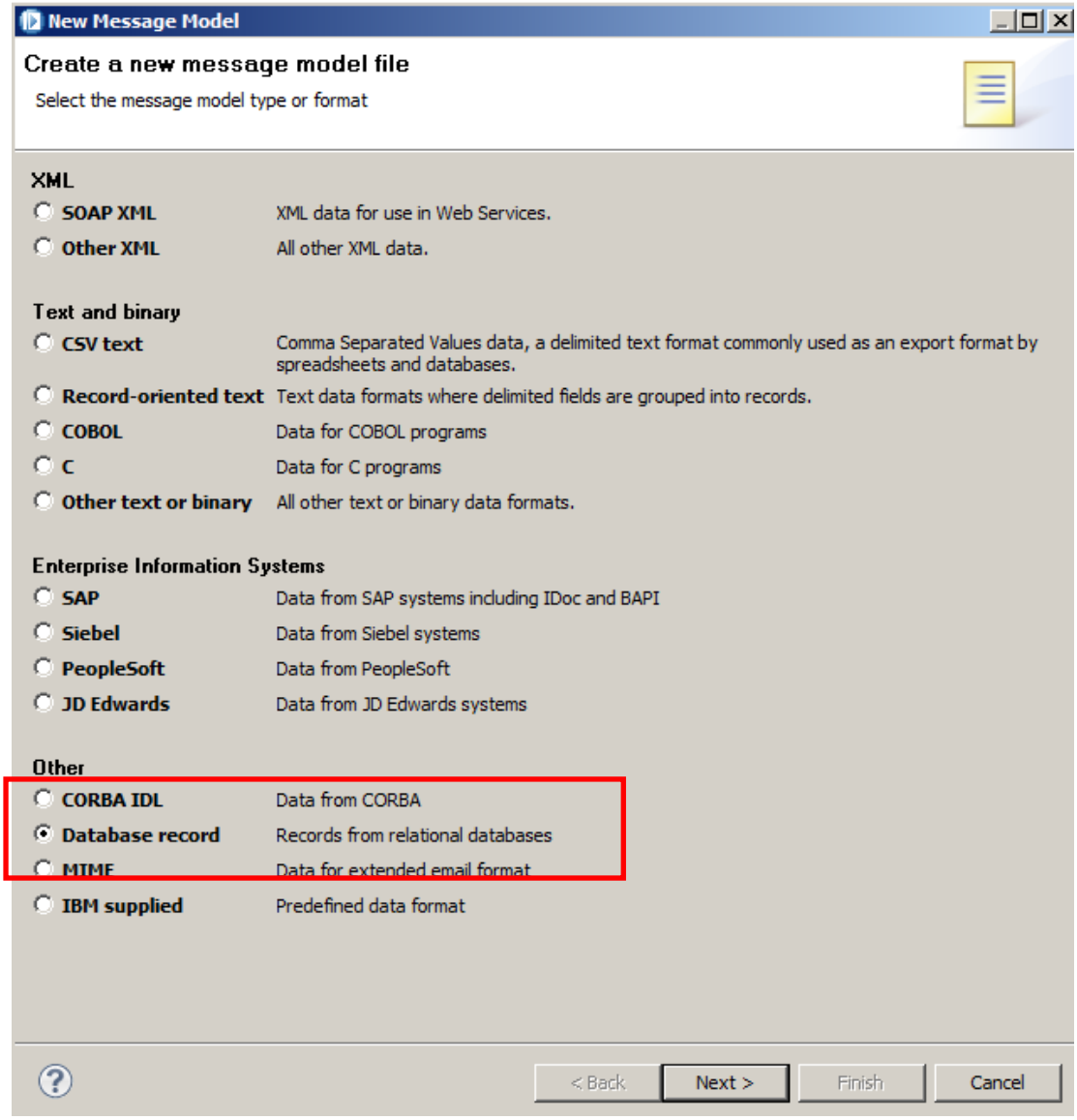
   Click Finish.



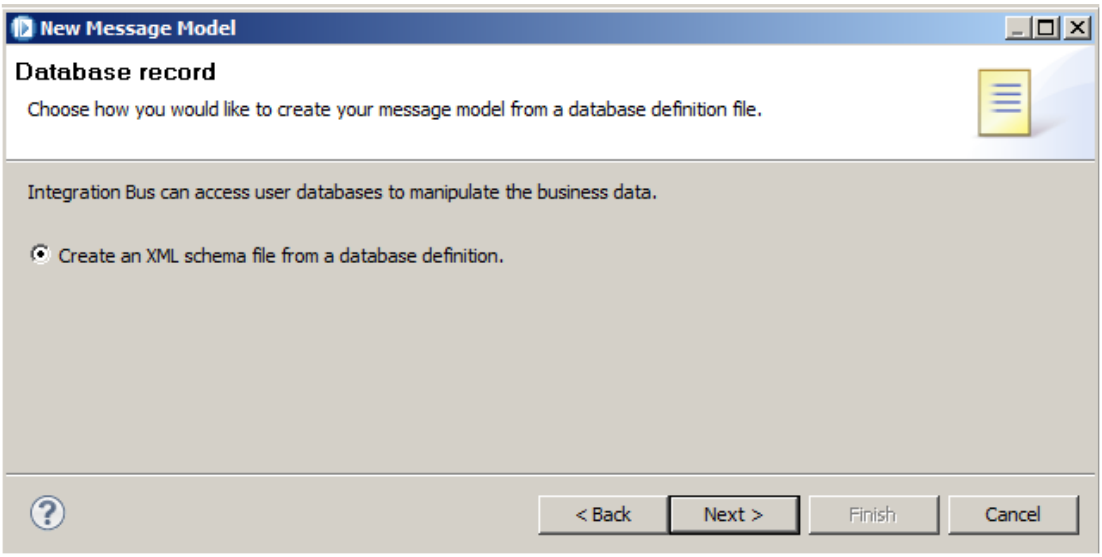
3. In this library create a new Message Model (click "New" under the library name), then select "Message Model".



Select the Database Record type, and click Next.



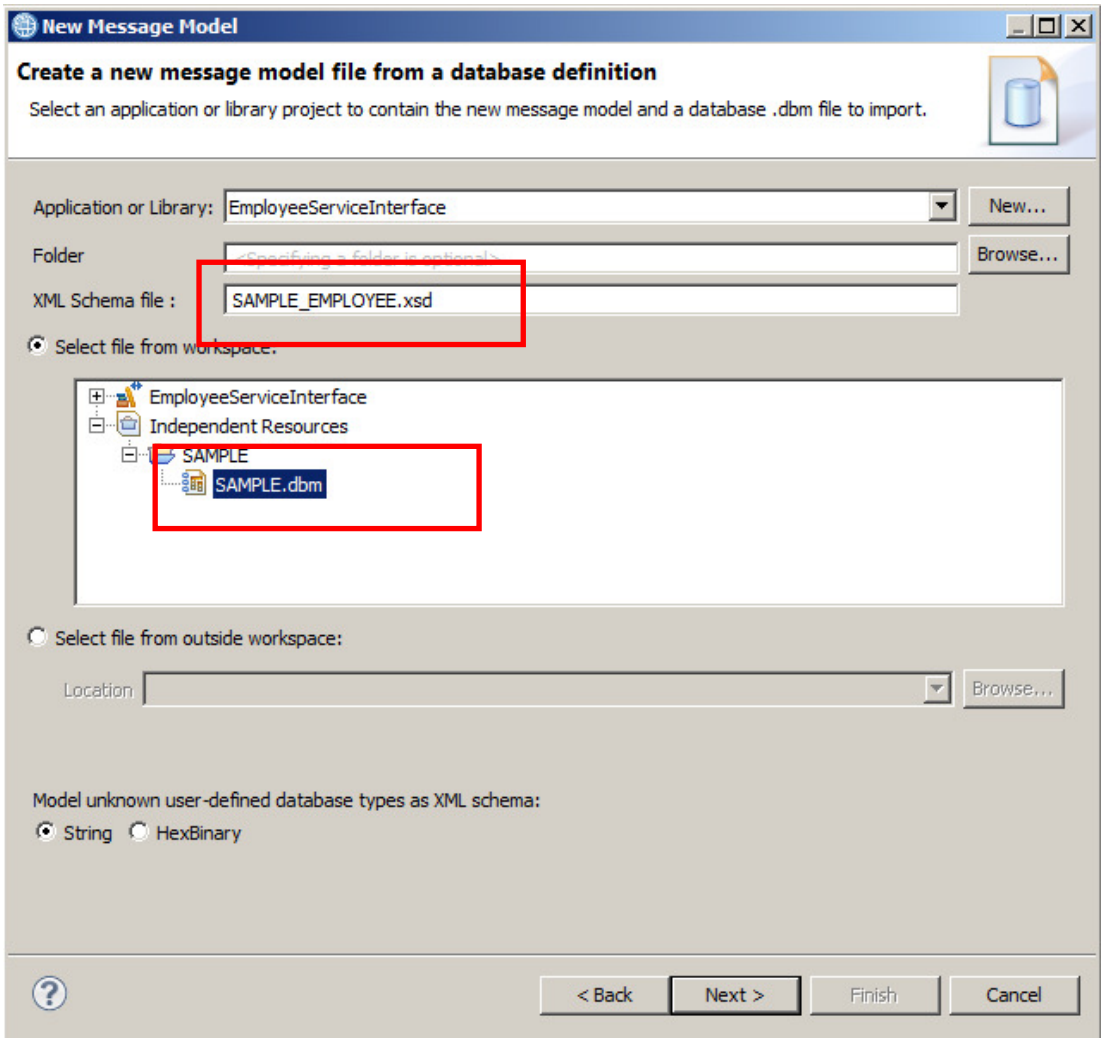
4. Create an XML schema - click Next.



5. Specify the name of the XML schema file that you will generate:

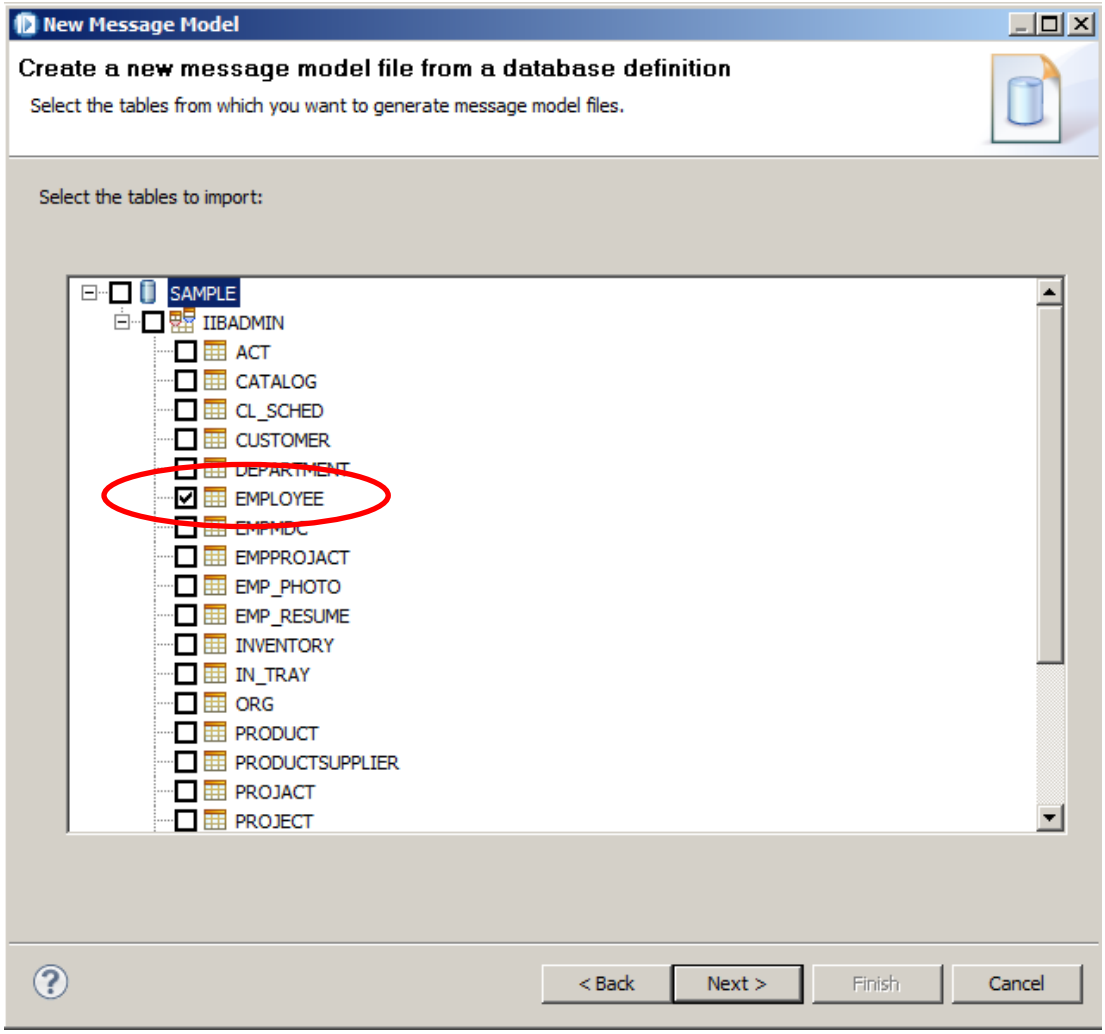
   - Highlight the SAMPLE.dbm file from the database definition project in the workspace (under Independent Resources).
   - For simplicity, you will create a schema containing only the elements for the EMPLOYEE table, so set the XML Schema file to **SAMPLE_EMPLOYEE.xsd**

   Click Next.



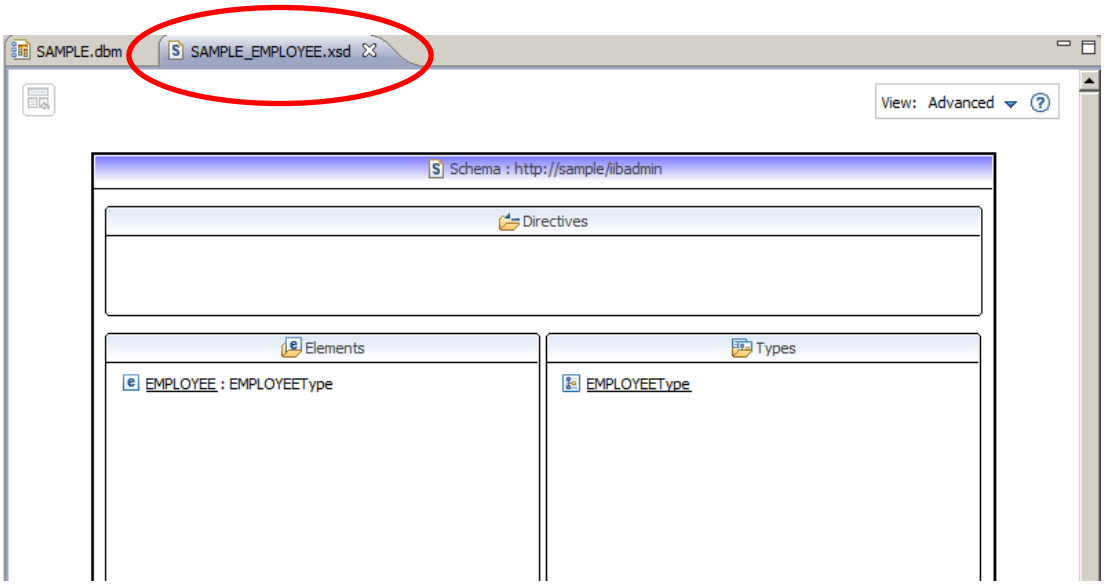
6. Select the EMPLOYEE table.

   Click Next, then Finish.



7. The schema will be created. Note that the name of the schema is SAMPLE_EMPLOYEE.

   If you forgot to set this name correctly, please delete and start again from step 2.1 The name of the schema is used frequently in several subsequent labs.



2.2 Add additional references to the schema

This lab will use a database scenario to illustrate the use of the Graphical Mapping Node within an Integration Service.

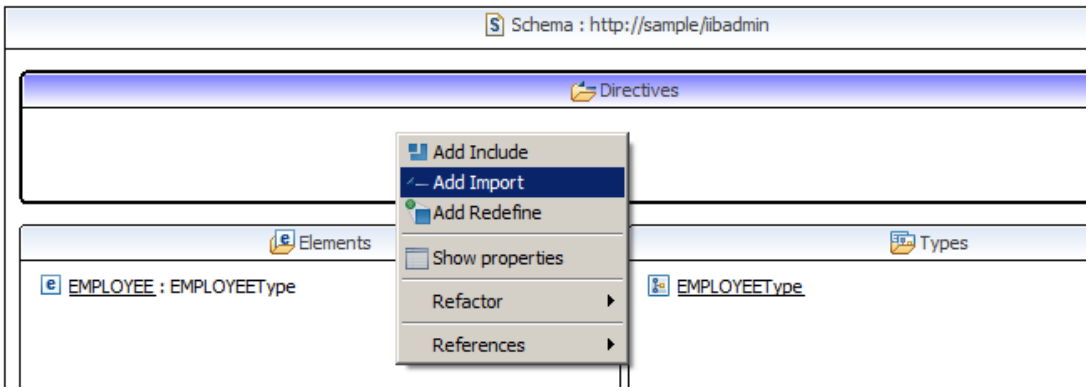
To do this, the Integration Service will pass a return code, plus details of any SQL messages, to the requesting application. You will therefore import a second schema, DBResp, into the SAMPLE_EMPLOYEE schema.

The DBResp schema contains several elements that can be used to report SQL responses, as well as a user return code.

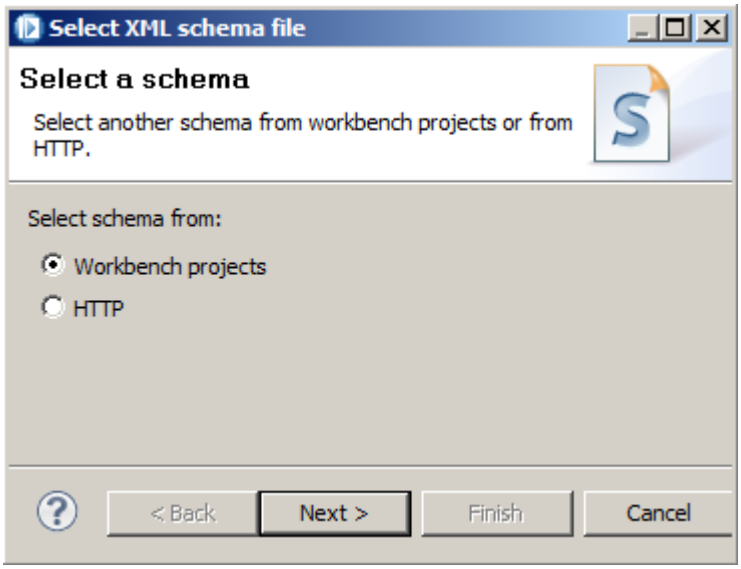
The SAMPLE_EMPLOYEE schema will be augmented with a new complex type, EmployeeeResponseType, which contains the EMPLOYEE and DBResp elements.

1. In the SAMPLE_EMPLOYEE schema, right-click in the Directives section.

   Click Add Import.

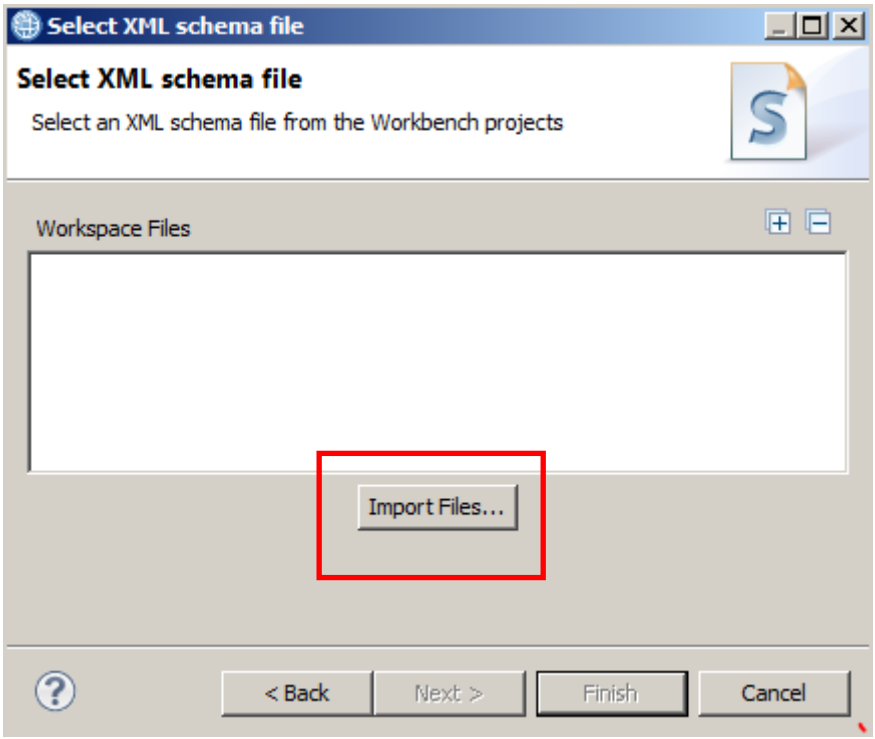


2. Select Workbench projects, and click Next.



3. Click Import Files.

   (You may see the EmployeeServiceInterface library mentioned in the Workspace Files pane below).



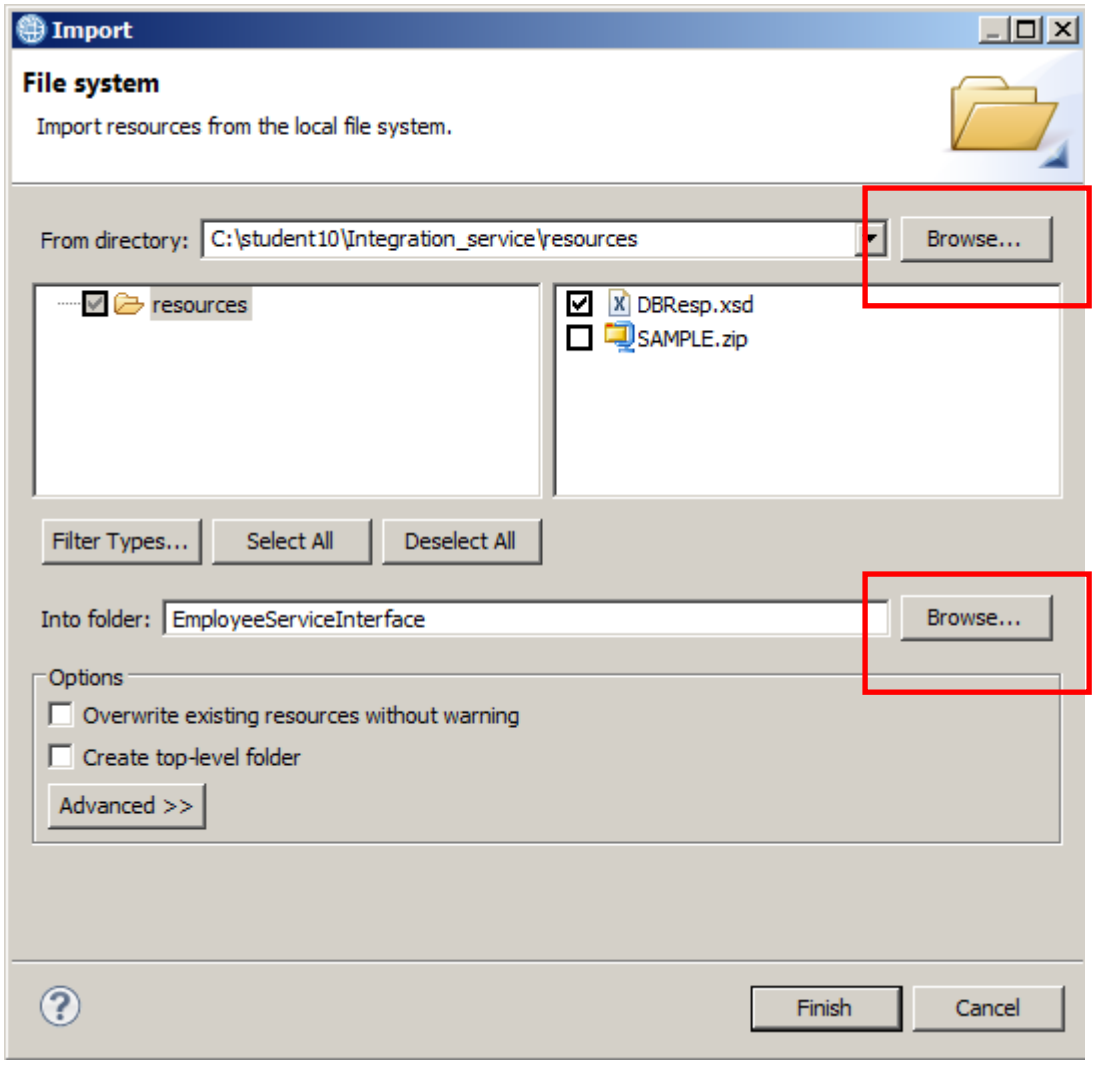
4. In the "From directory", use the browse button to navigate to

   `c:\student10\integration_service\resources`

   Select the DBResp.xsd schema.

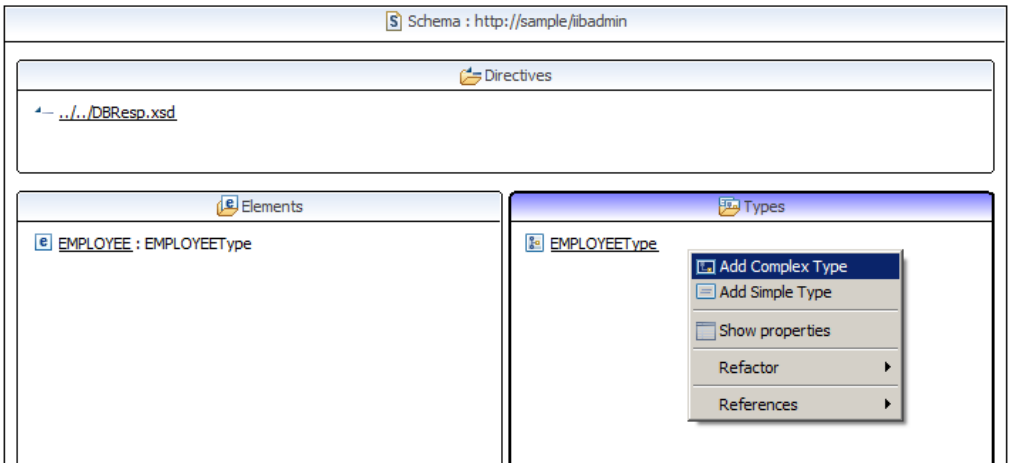
   Use the second Browse button to select the target folder - EmployeeServiceInterface.

   Click Finish, and then Finish again on the next window.

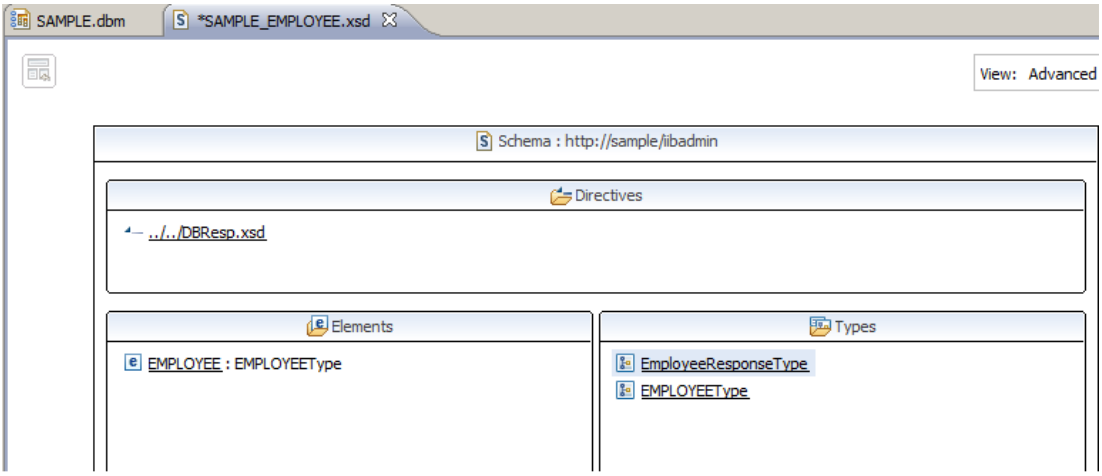


5. Note that the DBResp schema has been imported.

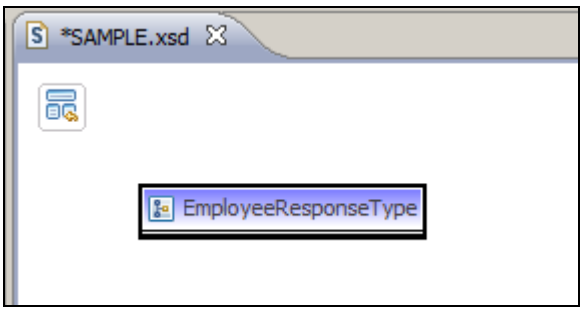
   In the Types pane, right-click in an open area, and select Add Complex Type.



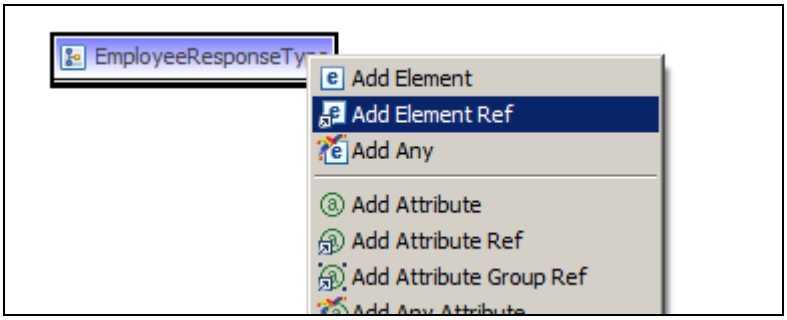
6. Set the name of the new Complex Type to EmployeeResponseType.



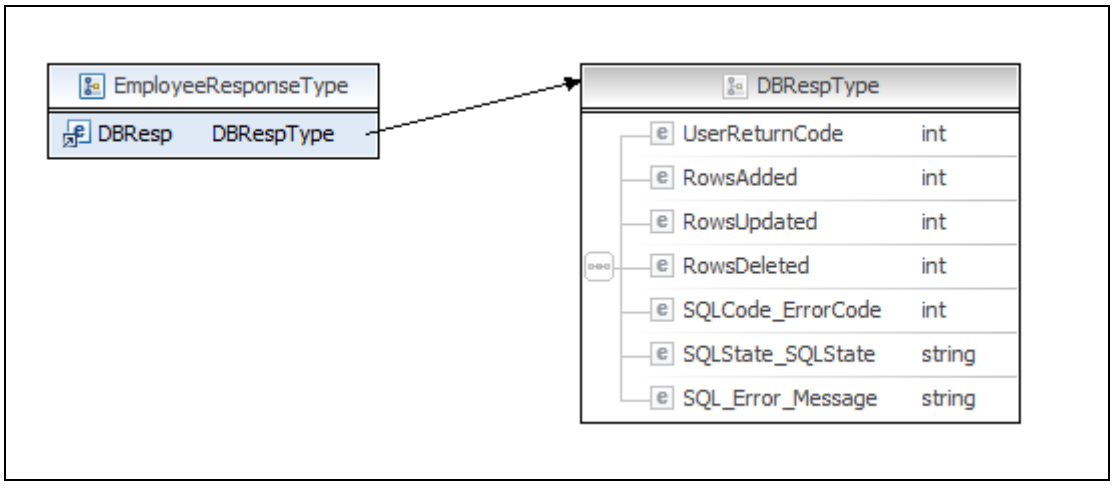
7. Double-click EmployeeResponseType which will open the schema editor for this type.



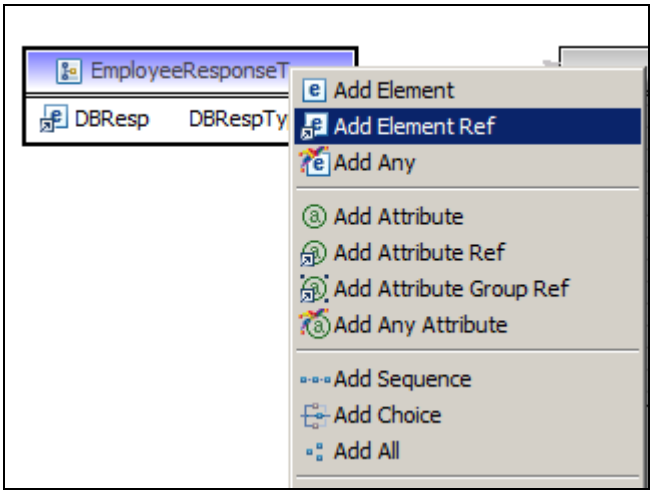
8. Right-click EmployeeResponseType and select Add Element Ref.



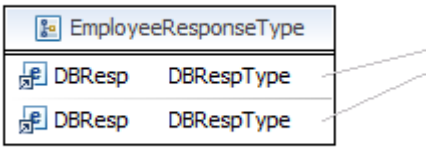
9. The editor will add a default element using the first alphabetically available. Luckily, since DBResp is the first available, this is the one that we want.



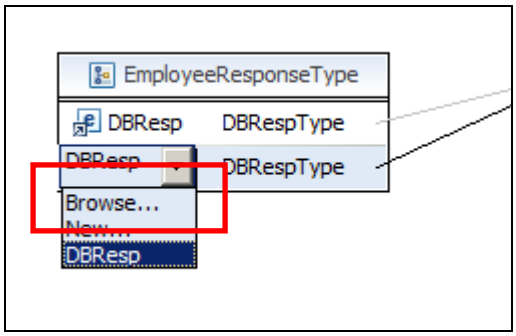
10. Now, add a second element reference, by right-clicking the EmployeeResponseType again.



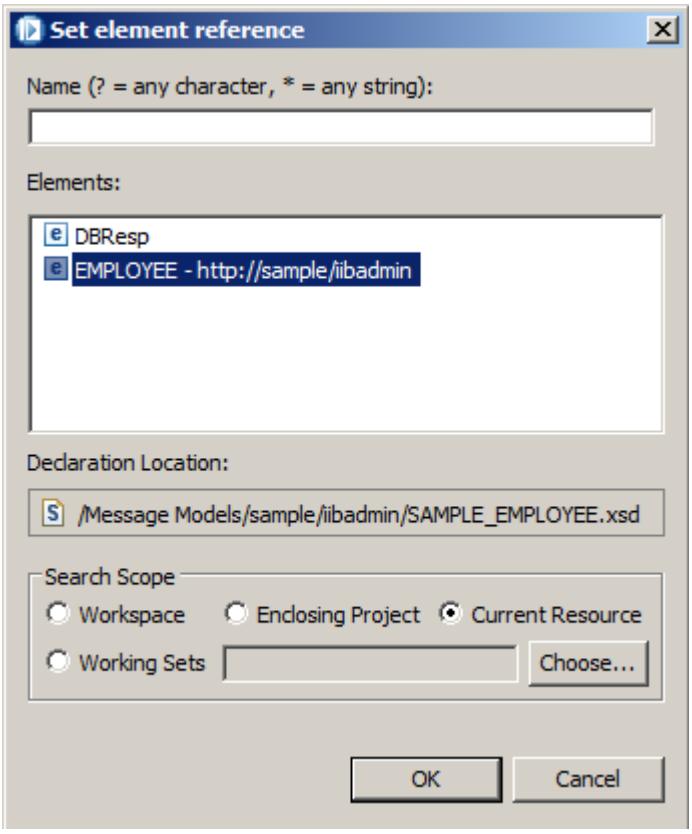
This will add a second reference to DBResp.



11. To change this second reference, click the second instance of DBResp (click DBResp, not DBRespType). A drop-down menu will be shown. Click Browse.

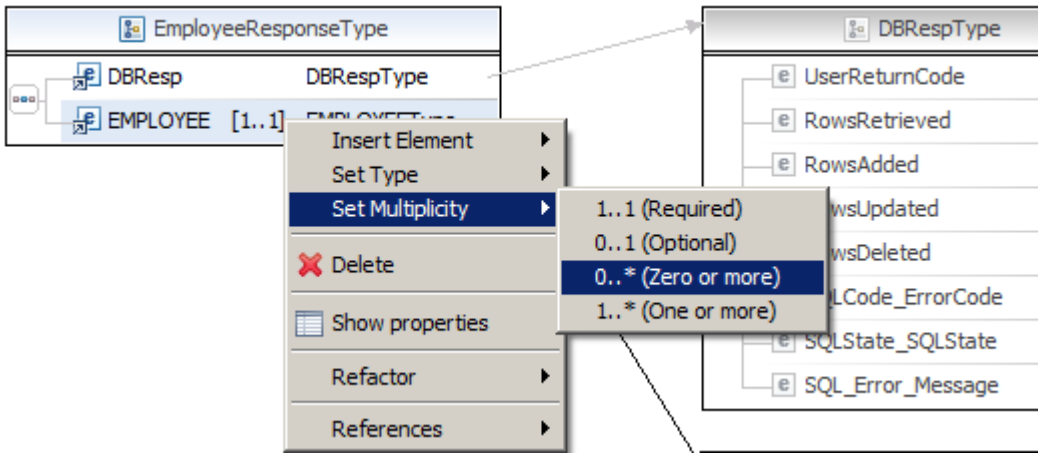


12. Highlight the EMPLOYEE reference, and click OK.

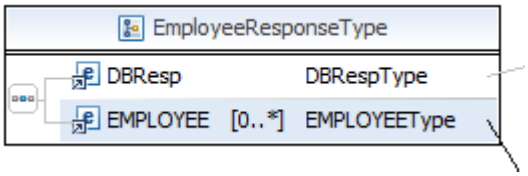


13. For some later scenarios, it will be necessary to be able to handle multiple instances of the EMPLOYEE element within EmployeeResponse, so set the Multiplicity as follows:

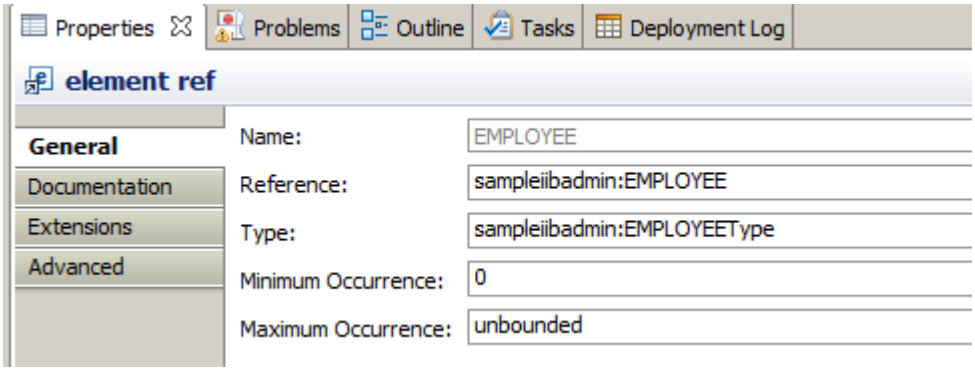
    Right-click EMPLOYEE, click "Set Multiplicity", then "0...*".



When you've done this, the complex element will look like this (click the EMPLOYEE element):



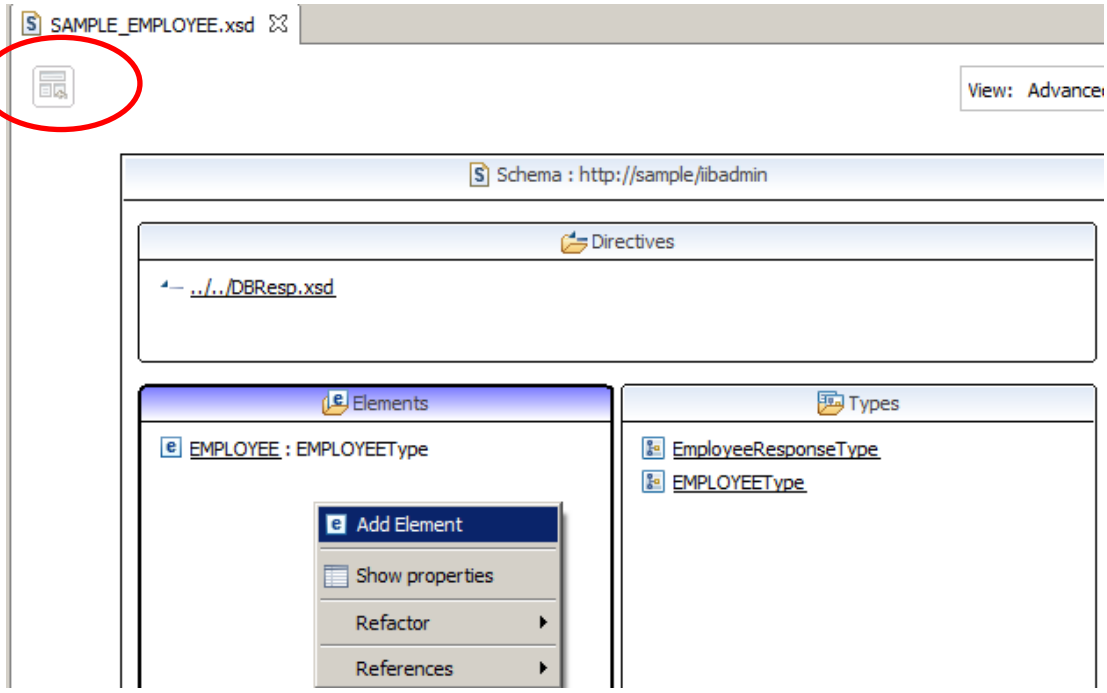
And the properties will look like this (note the minimum and maximum occurrences):



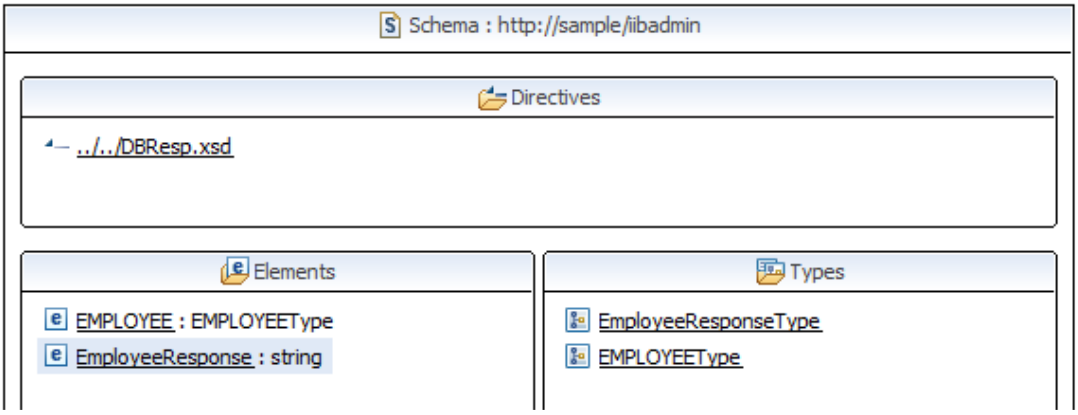
14. Return to the high-level definition of the schema (click the highlighted icon shown below).

    The Mapping Node bases its message inputs and outputs on element definitions, rather than element types, so it is necessary to define a new element for the EmployeeResponse.

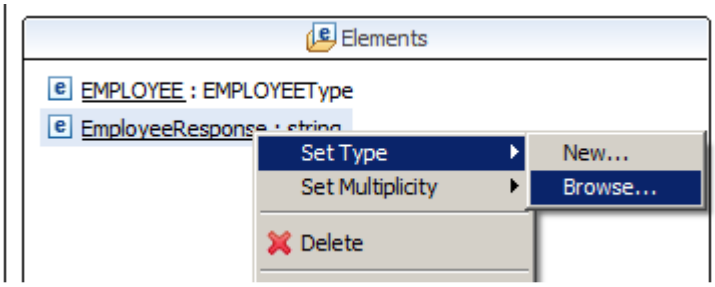
    In the Elements pane, right-click, and select "Add Element".

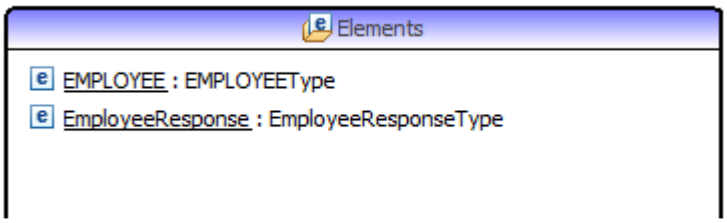


15. Name the new element EmployeeResponse.



16. Set the Type of the new element to EmployeeResponseType (follow the dialogue as before, using the Browse function).





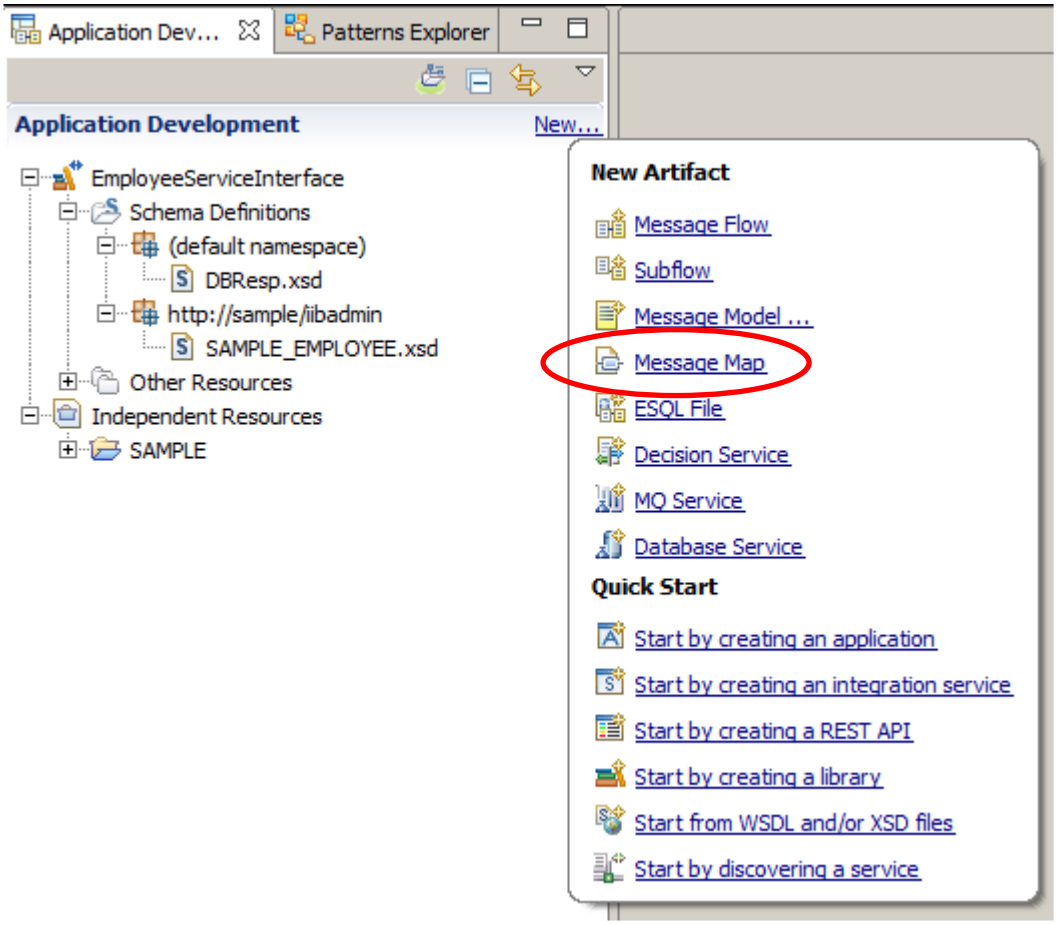
17. The schema is now ready for use, so Save and Close.

3. Create the Employee Service

3.1 Create the Mapping Node

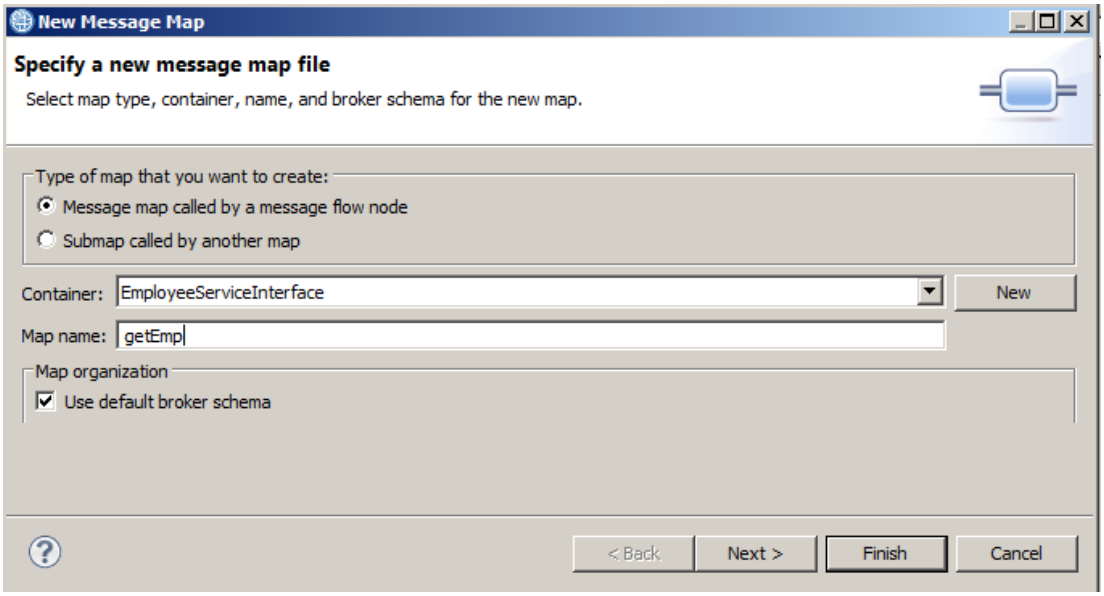
The new service will contain just one operation. This operation will use a Mapping Node to retrieve employee details from the EMPLOYEE table in the SAMPLE database. The Mapping Node will be created and stored in the Shared Library, because it will be used by other applications later on.

1. Highlight the EmployeeServiceInterface library, and click New. Select Message Map.



2. Make sure the Container is EmployeeServiceInterface.

   Set the Map name to "getEmp". Click Next.

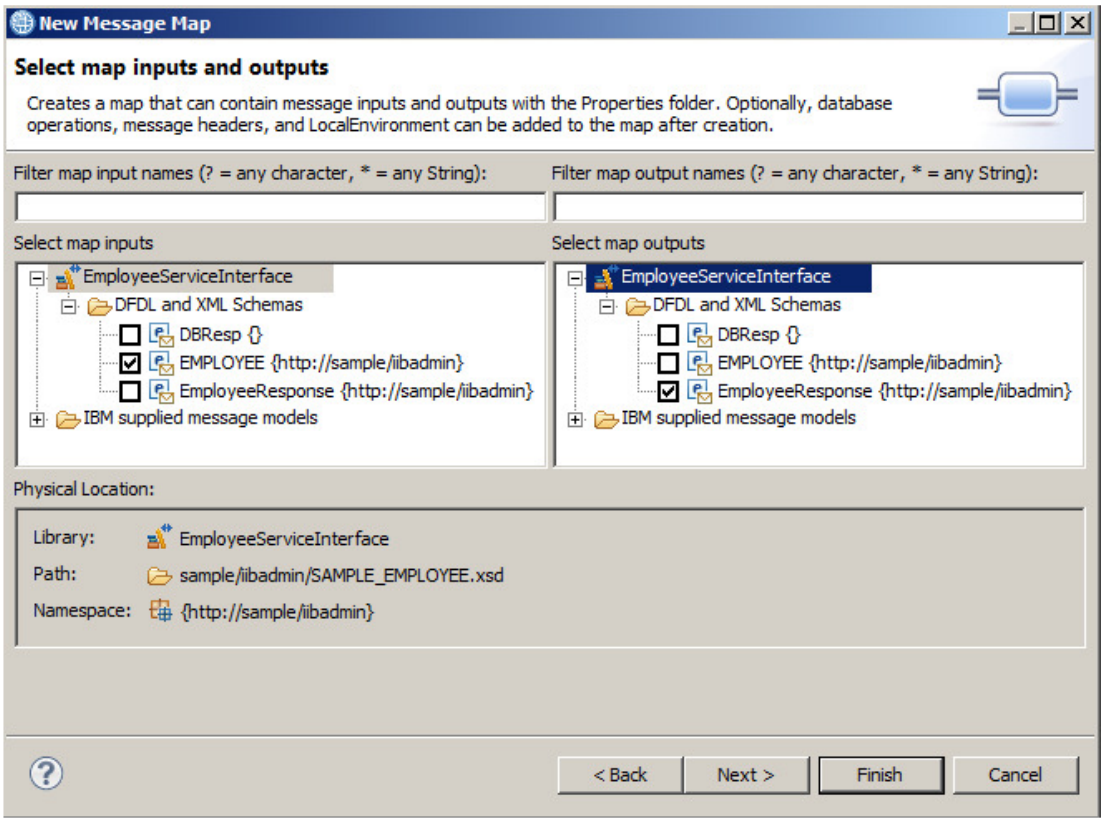


3. Expand the EmployeeServiceInterface library for both input and output to the map. Expand DFDL and XML Schemas.

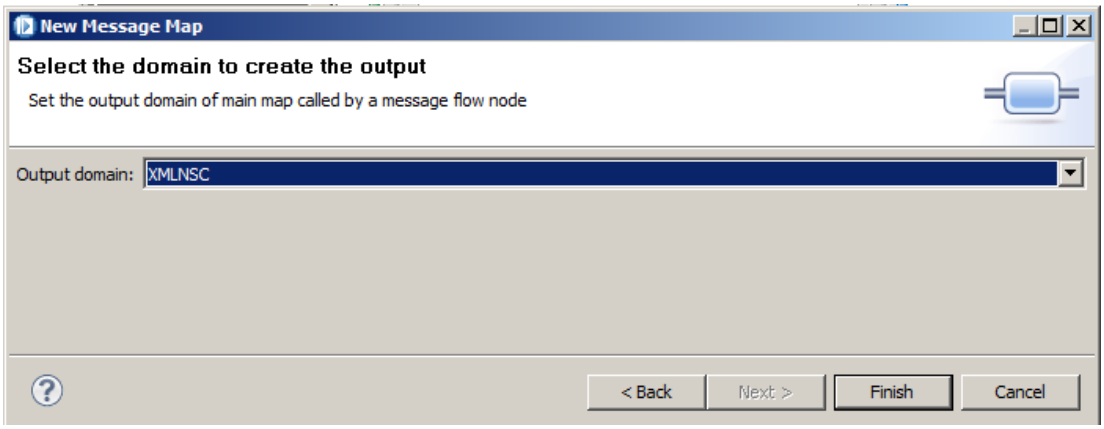
   For the input, select the input message EMPLOYEE.

   For the output, select the output message EmployeeResponse.

   Click Next.

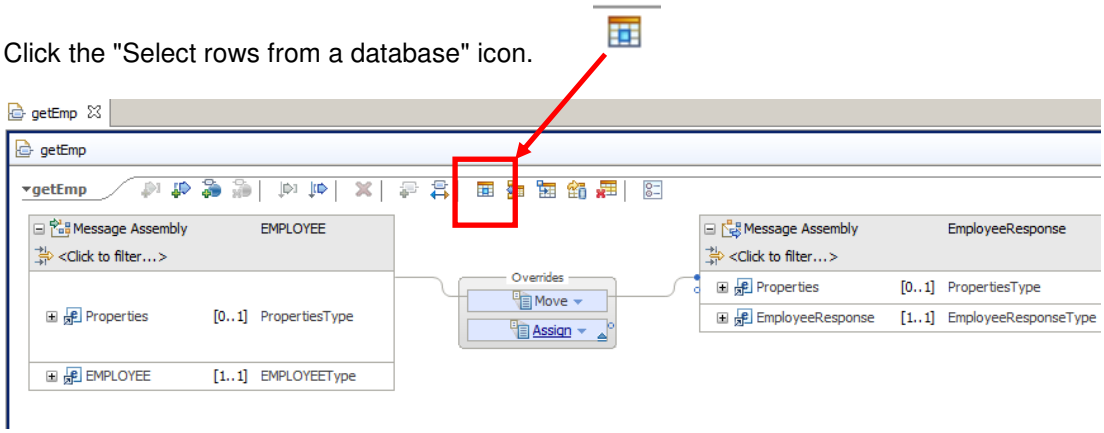


4. Select the output domain (XMLNSC in this case). Click Finish.

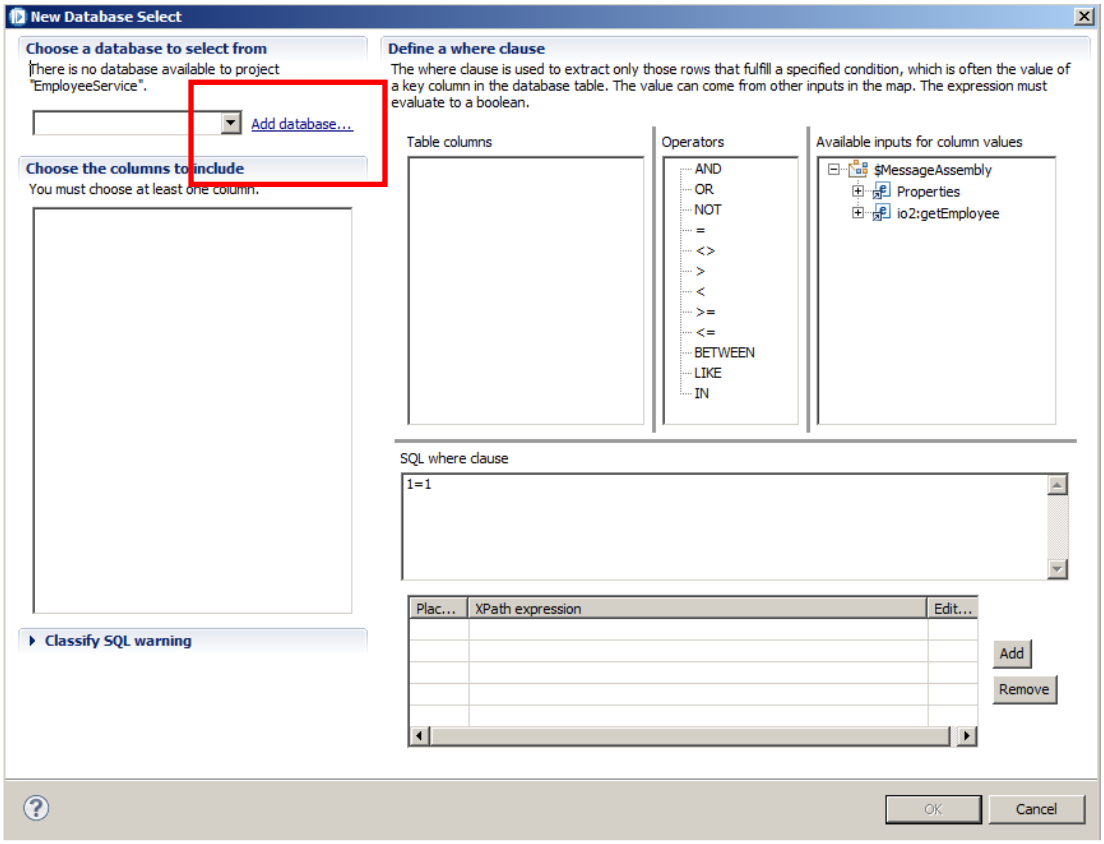


5. The basic map will be shown, with the Message Properties mapped by default, but the message body not yet mapped.

Click the "Select rows from a database" icon.

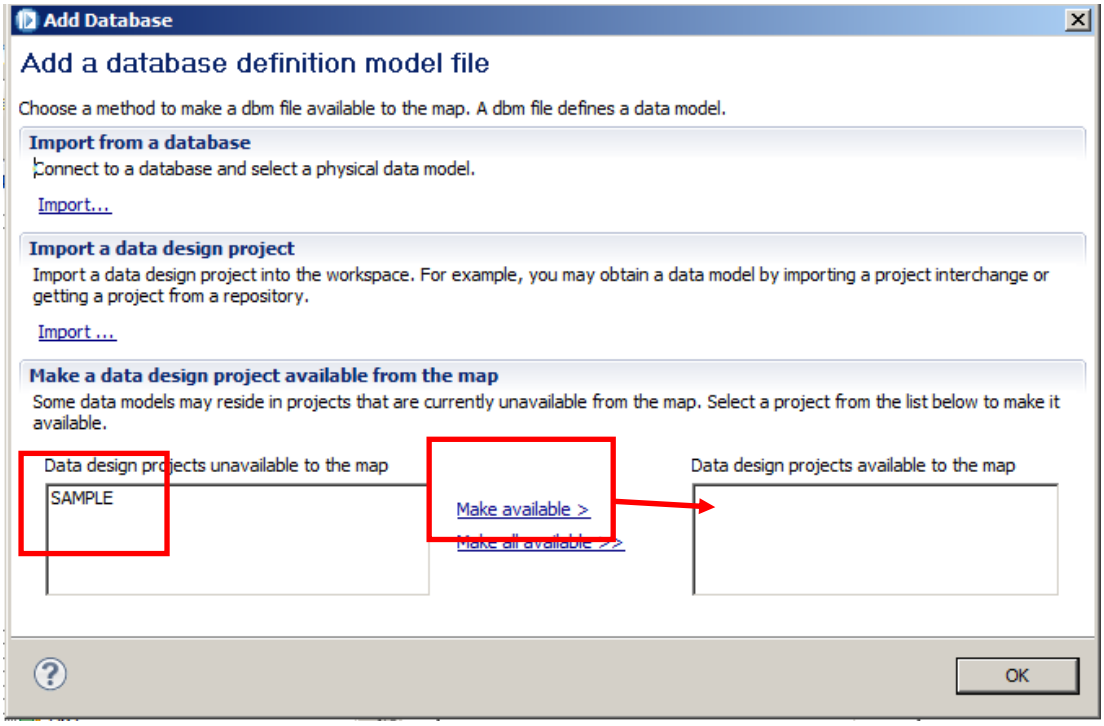


6. Since the integration service does not have a reference to the SAMPLE.dbm, click the "Add database" link.



7. Select the SAMPLE database, and "Make available".

   When SAMPLE appears in the pane on the lower right, click OK.



8. In the Database Select wizard:

    1. Select (tick) the EMPLOYEE table.

    2. Remove the "1=1" phrase from the "SQL where clause" pane.

    3. Construct the SQL where clause:
        a. Double-click the EMPNO column
        b. Double-click the "=" sign
        c. Double-click the EMPNO element from the io:EMPLOYEE input (under available inputs)

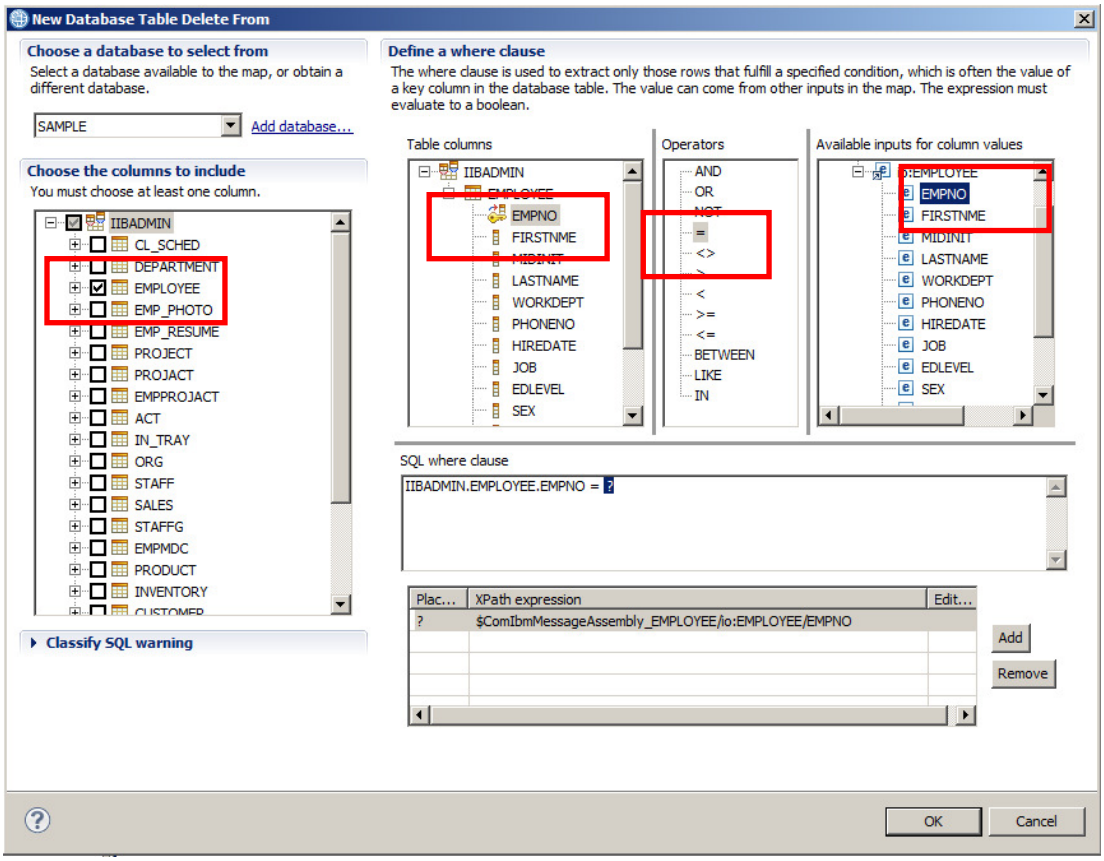
As a check, the SQL clause that should be generated should be:

```
IIBADMIN.EMPLOYEE.EMPNO = ?
```

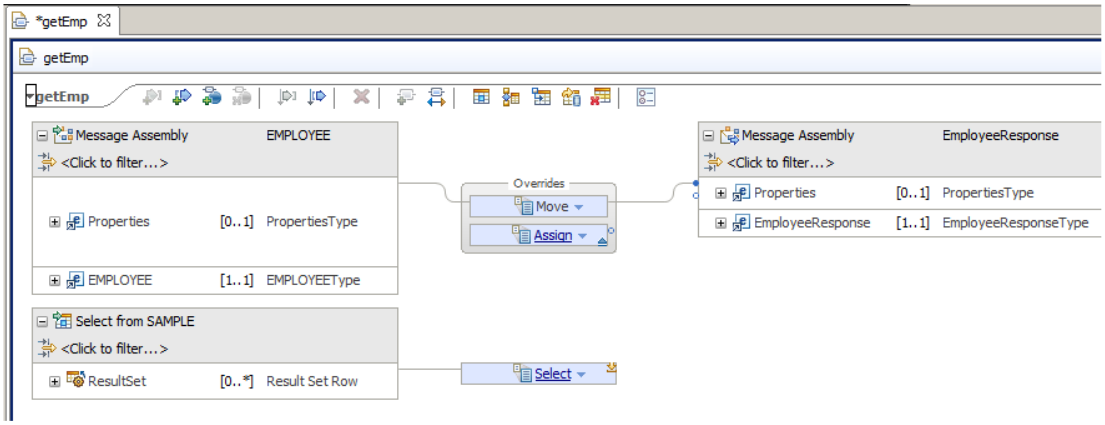
And the generated XPath expression should be:

```
$ComIbmMessageAssembly_EMPLOYEE/io:EMPLOYEE/EMPNO
```

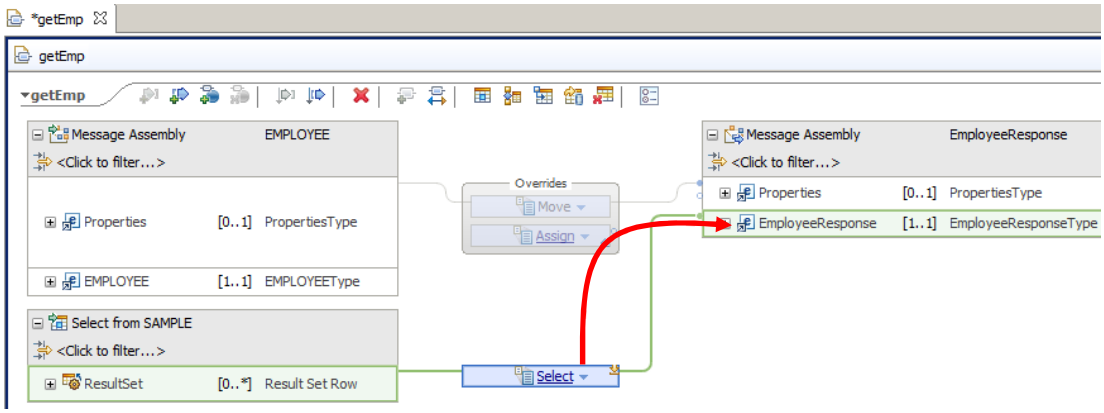
Click OK.



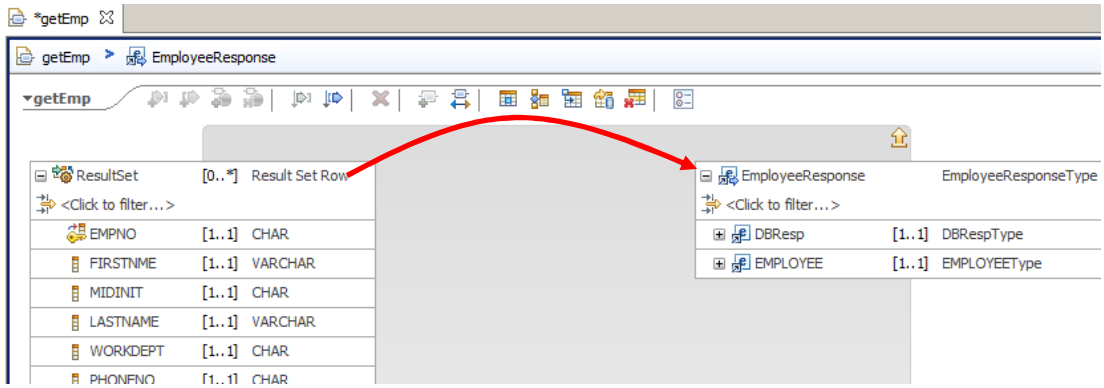
9. The map will be shown as follows:



10. In the output message assembly, connect the Select transform to EmployeeResponse.

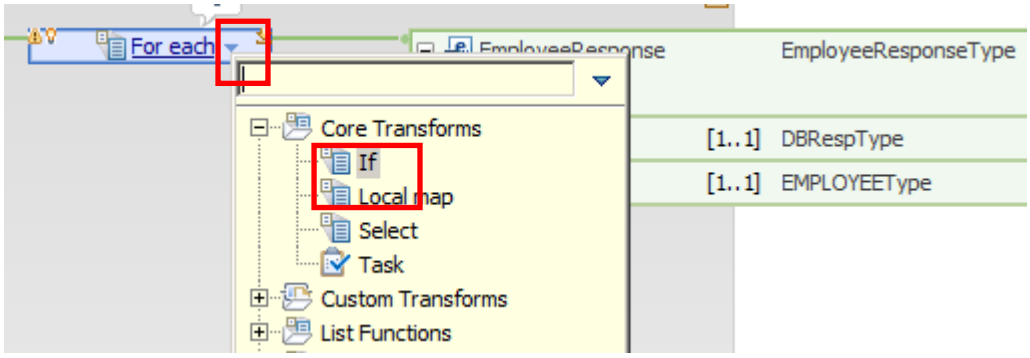


11. Click "Select" to enter the next logical level of the map.
Connect ResultSet to EmployeeResponse. This will generate a "For each" transform.

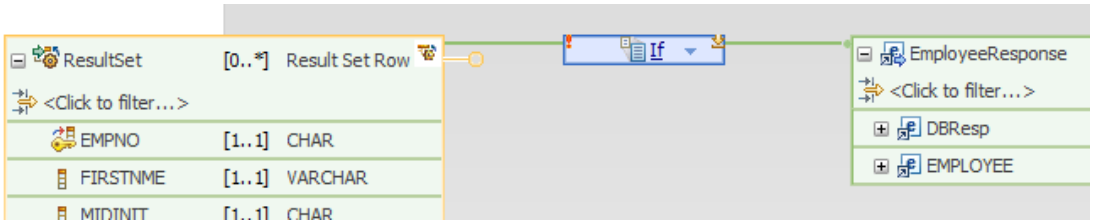


12. Click the small BLUE arrow next to the "For each" words. This will open a dialogue where you can select the type of function for the transformation. Click the "If" transform. This part of the map will be the "success" path (ie. the database read has returned at least one row).

.

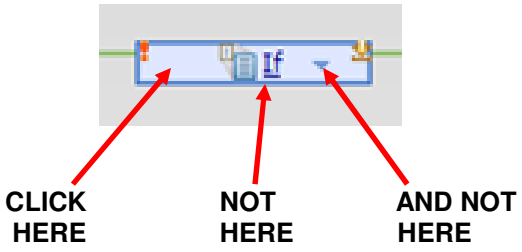


13. You will see that the "If" transform is currently showing an error, indicating that no condition has been set.



14. Now set the Condition for the "If" transform.

    Make sure the "If" transform is selected, by clicking the blue box surrounding the "If" transform),



    Now select the Properties pane, which will show all the Properties of the "If" transform.

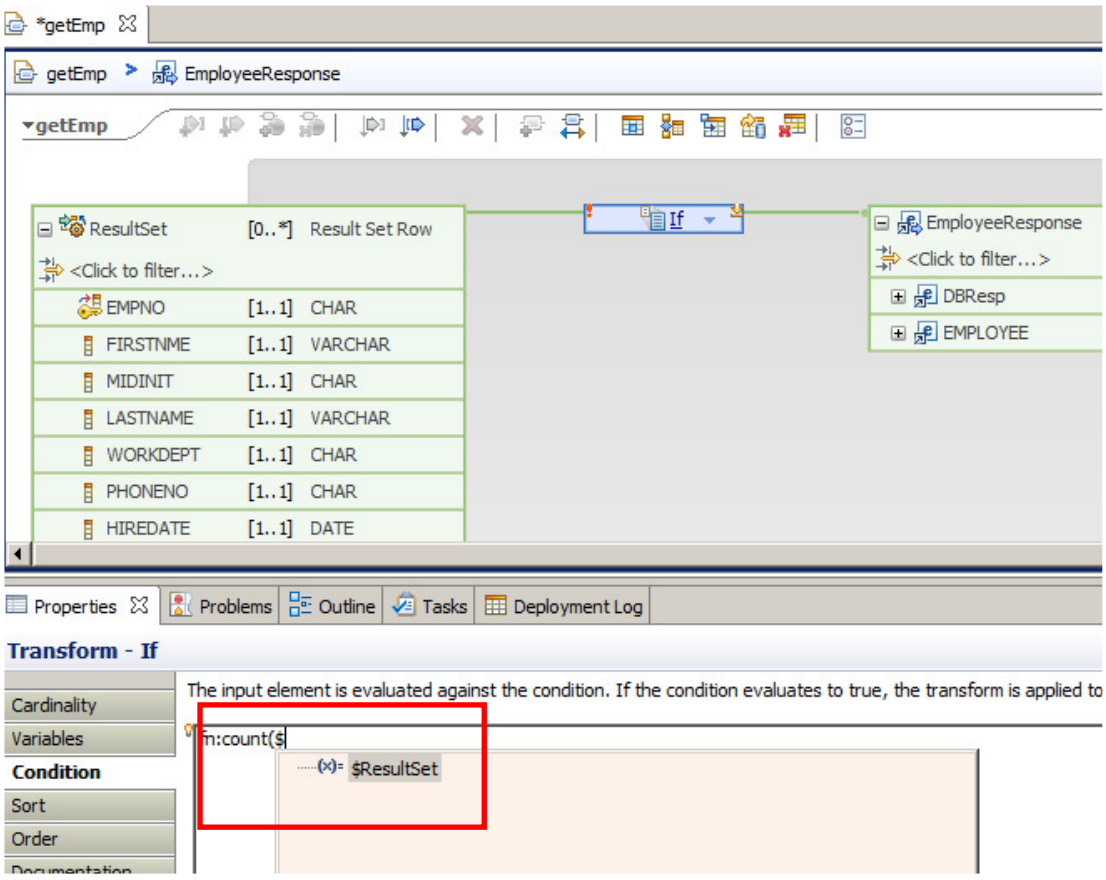
    On the Condition tab, type the following into the Condition editor:

```
fn:count($
```

    The content assist function should then automatically show the possible variables. Assuming that $ResultSet is displayed, select this variable for completion. If not, you can manually start this by using the Ctrl-Space key combination.

    Then, manually complete the condition as follows:

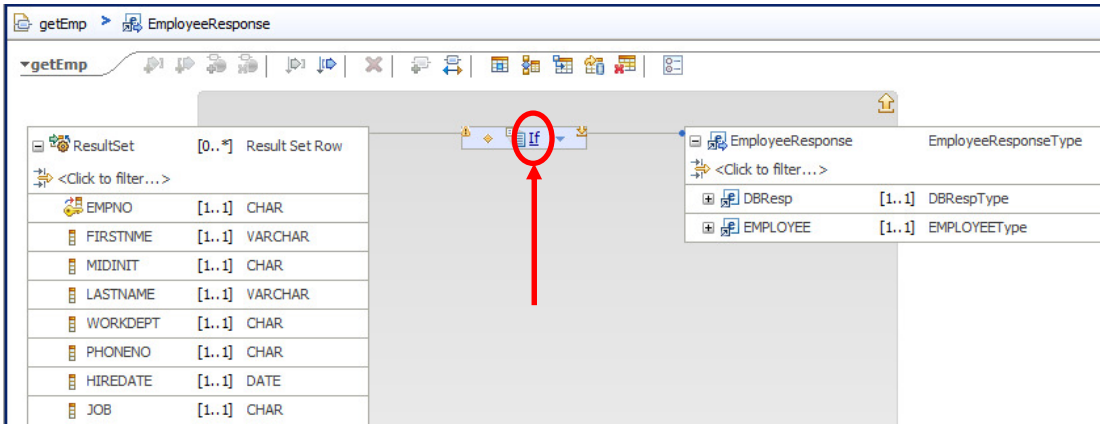
```
fn:count($ResultSet) > 0
```



    (Note - you should always use the content assist feature to choose the available variable ($ResultSet in this case). If you subsequently remove and re-add the "If" transform, a differently-named variable would be generated (eg. $ResultSet1), and the content assist feature will always generate the correct corresponding XPath statement).

15. Save the map. The error on the If transform will have been resolved.

    Now click "If" (the word, not the surrounding box) to implement the required transforms.

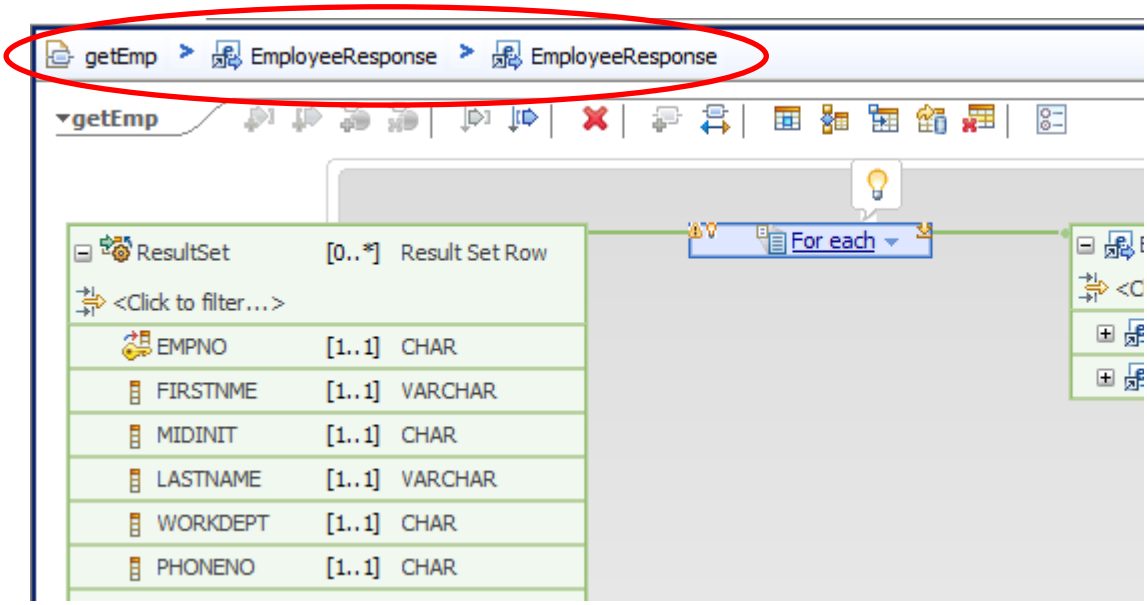


    Be careful not to create more than one logical level of "If" transforms. Clicking "If" will automatically create a new logical level of the map. In this lab, you should only create one "If" transform level.

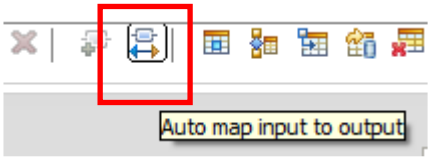
    The same effect is observed with other transforms such as "for each", "select", "else", etc.

16. Initially, you will see a "For each" transform. Delete this transform (ensure the transform is highlighted, then use the delete key, or the red cross).

    By the way, here is a good time to point out the names of the logical levels of the map. You can click any of these levels, and the editor will take you direct to the selected level.

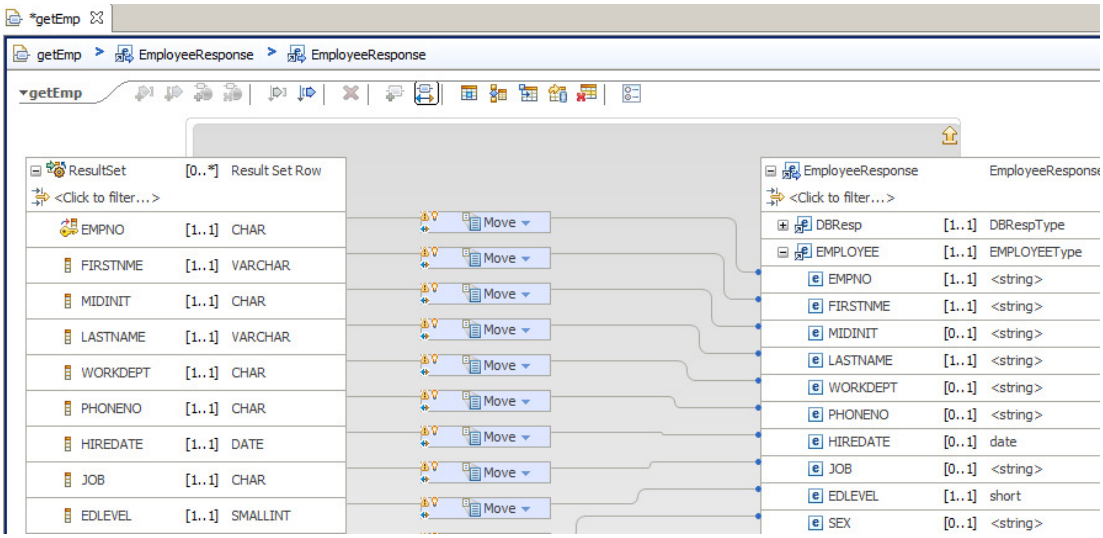


17. To map the required elements, use the Automap function, using the icon shown below.

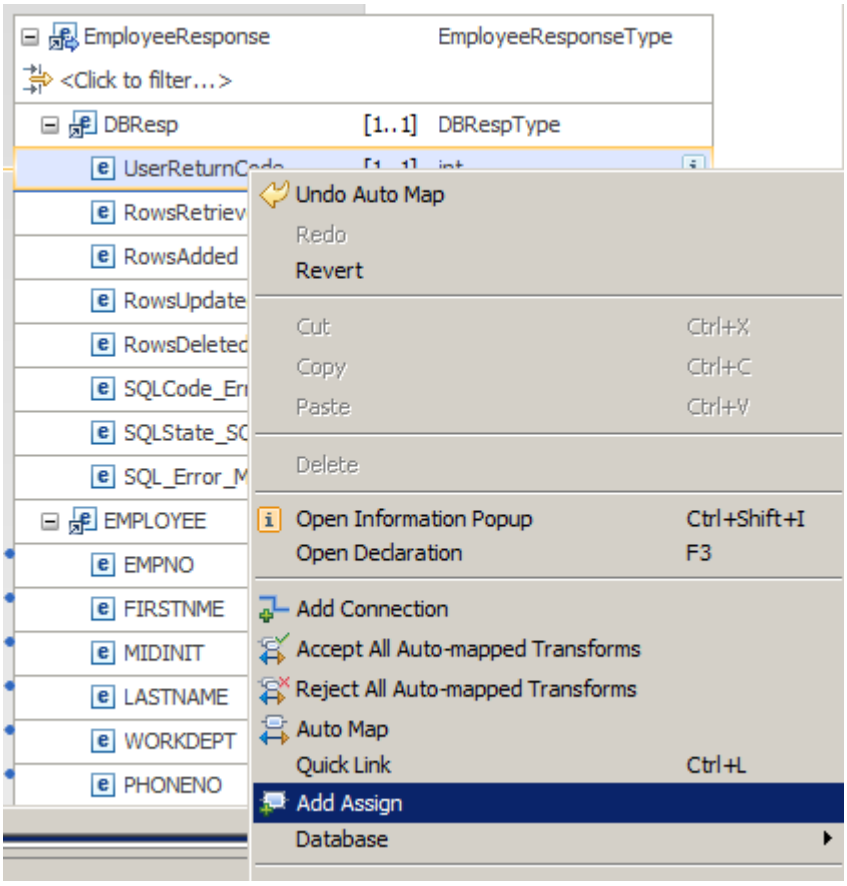


Hover over each icon to find the Automap icon.

At the AutoMap window, for this scenario, you can click Finish immediately, and all the correct mappings will be made, as shown:

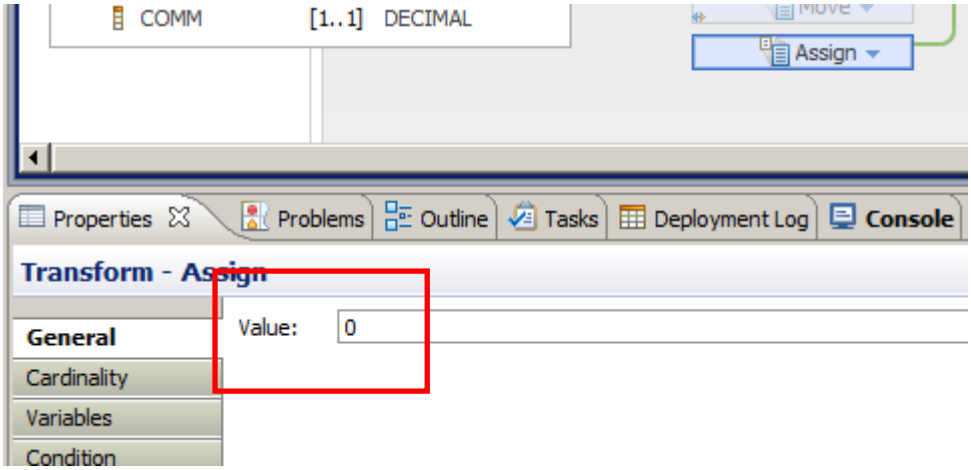


18. To set the value for UserReturnCode, expand DBResp, right-click UserReturnCode and select Add Assign.



19. Highlight the Assign transform, and select the Properties of the transform.

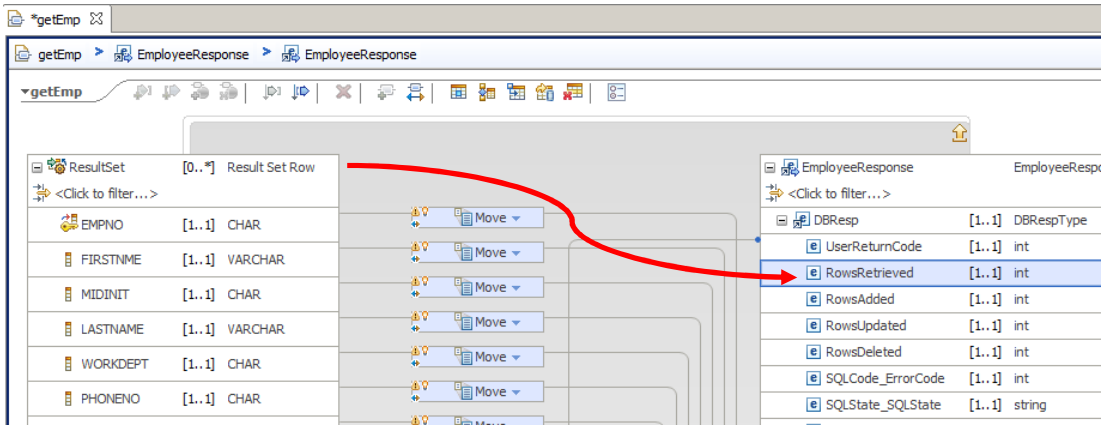
    On the Properties General tab, set the value of the Assign to 0. This value will be set if the database has been successfully accessed, even though the number of rows returned may be zero (the default value is 0).



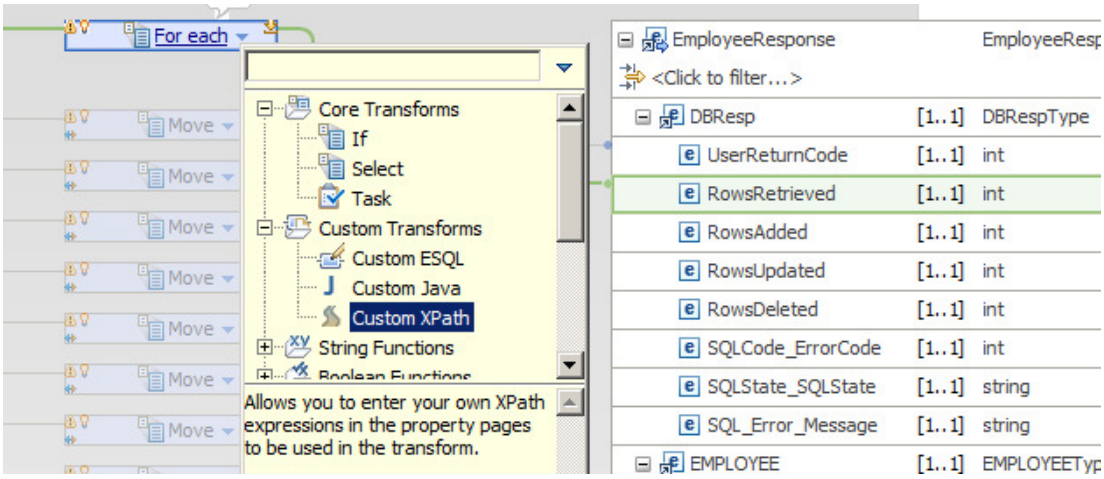
20. Since this logic applies if at least one row has been retrieved from the DB table, the RowsRetrieved element should be set to the appropriate value.

    Connect the input ResultSet to the RowsRetrieved output element.

    Connecting the input ResultSet ensures that when you construct the XPath statement in the next few steps, the content assist tool will show you the correct variable name.



21. This will create a "For each" transform. Click the blue down-arrow on the transform, and change it to Custom XPath.



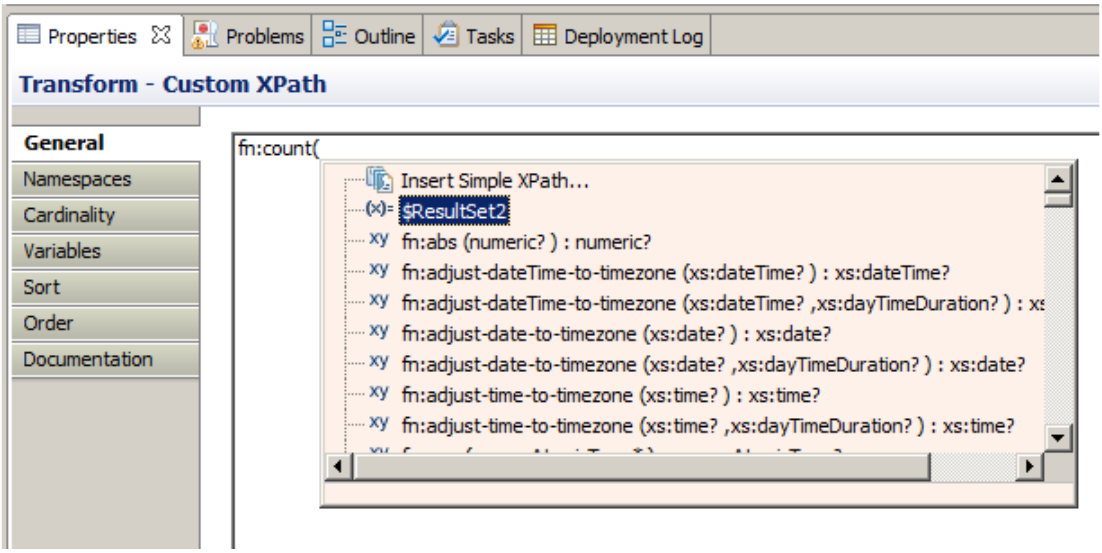
22. Make sure the Custom XPath transform is selected.

    On the Properties tab for this transform, select "General" and type the following into the XPath editor:

    ```
    fn:count(
    ```

    Then, invoke the content assist function (Ctrl-space) which will show you the possible values for completion. Select the value that shows $ResultSet, or similar. Depending on whether you have made other editing changes, the value shown may be $ResultSet or have a suffixed number. The example shown below is $ResultSet2.

    Complete the XPath expression with a ")". This expression will calculate the number of rows retrieved from the database.

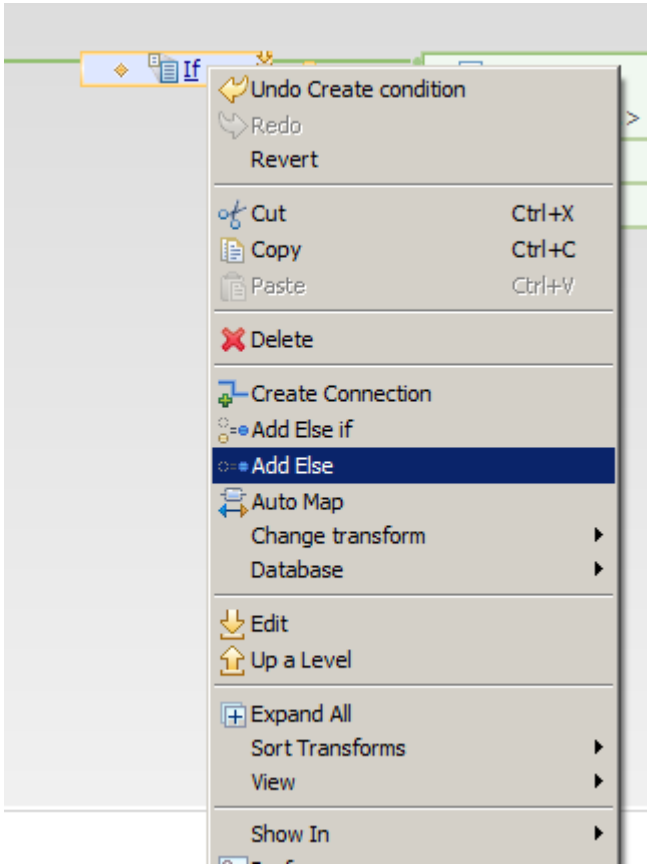


23. Now add the Else transform to the map.

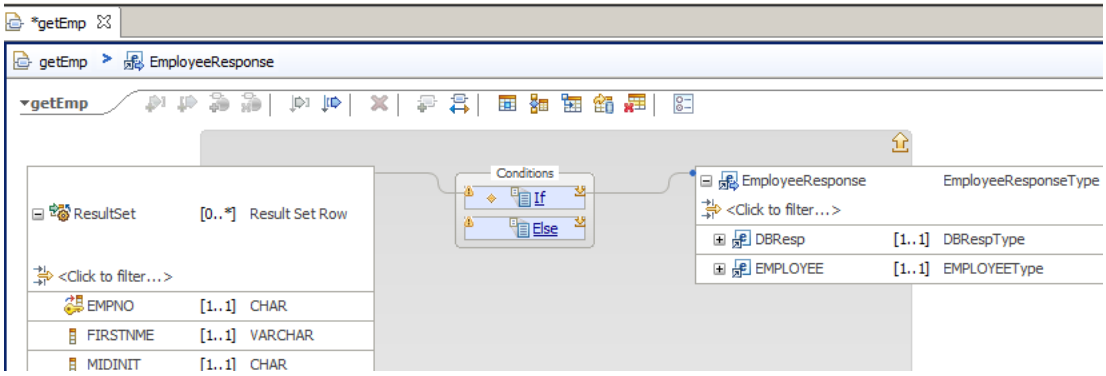
    To do this, move up one logical level in the map to return to the If transform (use the yellow up arrow).

    If no rows are returned from the database, the map will return a value of "0" in the UserReturnCode field (indicating that the database access has worked), and a value of "0" in the RowsRetrieved element.

    To specify this in the map, add an Else transform, by right-clicking the If, and selecting "Add Else".

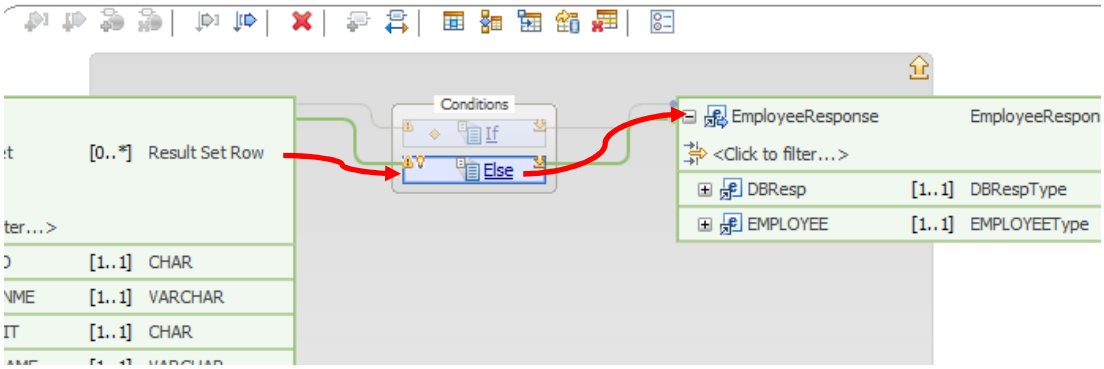


24. The Else transform will be added as shown.

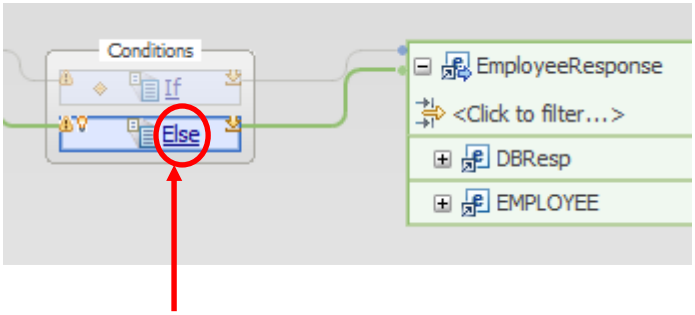


25. Connect the "ResultSet Row" to the Else transform.

    Connect the Else transform to EmployeeResponse (this is required so that the XPath content assist references the correct XPath variable).



26. Click the Else transform (click the "Else" word) to implement the required transforms.

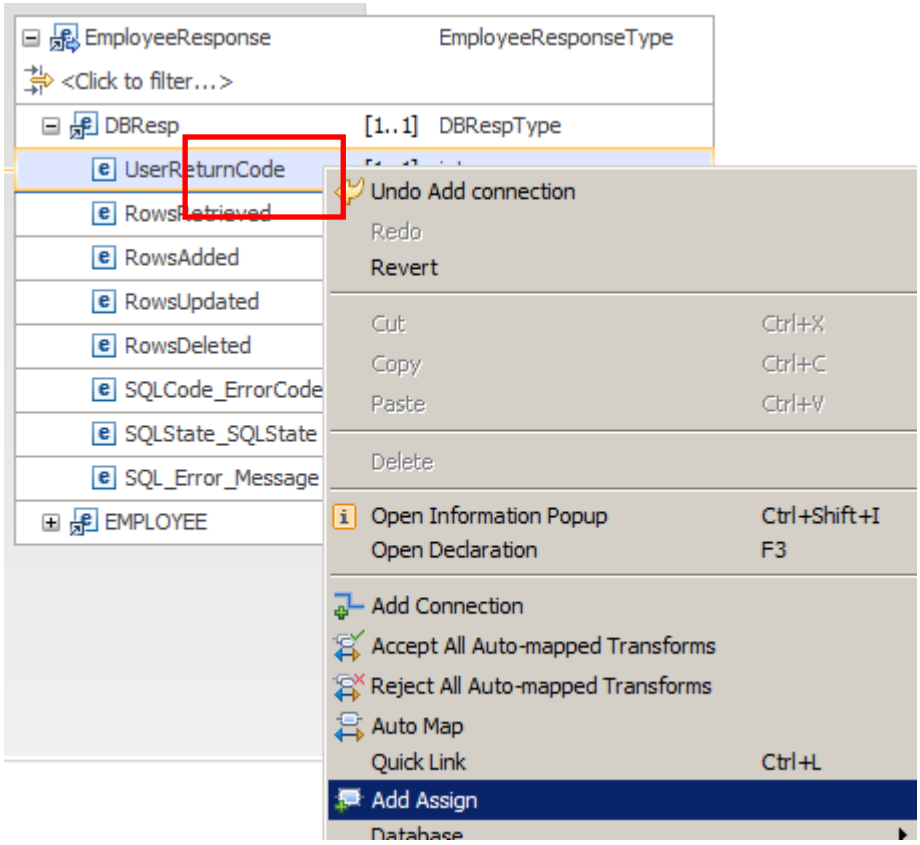


27. This will take you to the next logical level of the map.

    Expand DBResp.

    Right-click the UserReturnCode, and select Add Assign.

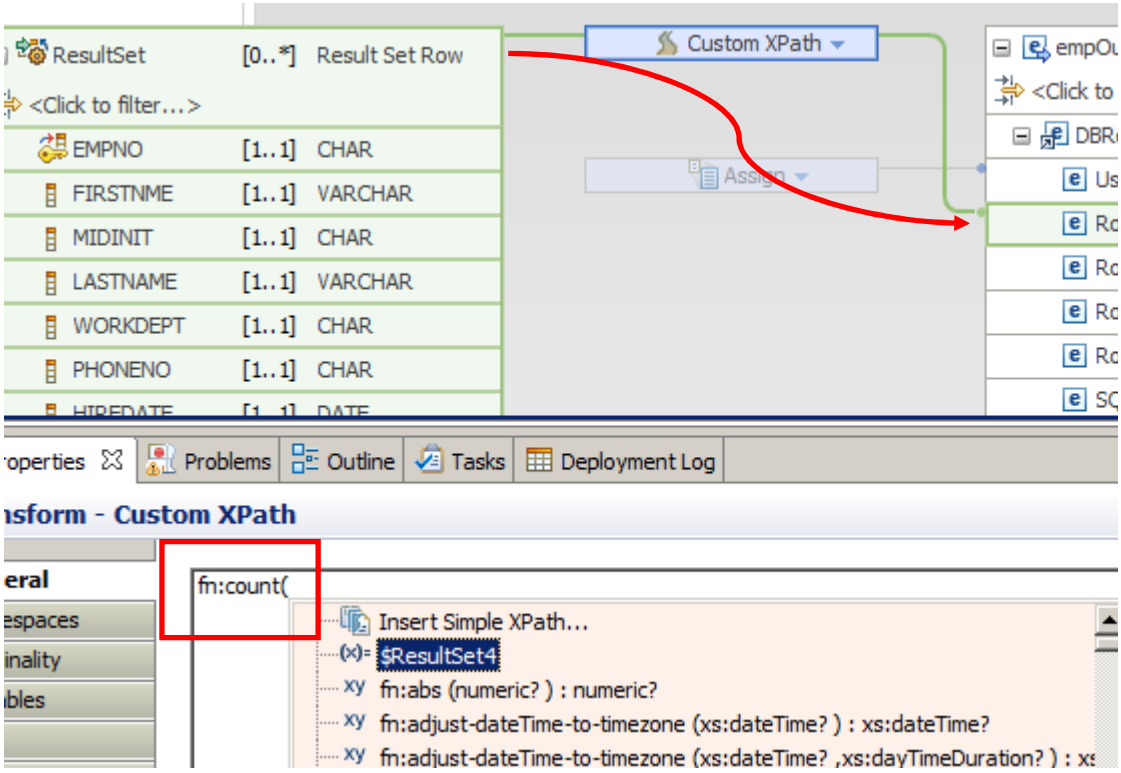
    In the General Properties of the Assign, set the value to 0, as before.



28. Connect the input ResultSet to the output RowsRetrieved. As before, change the "For each" to Custom XPath.

    Using the same technique as earlier, type "fn:count(" into the General editor, and then use the content-assist tool to generate the correct $ResultSet variable.

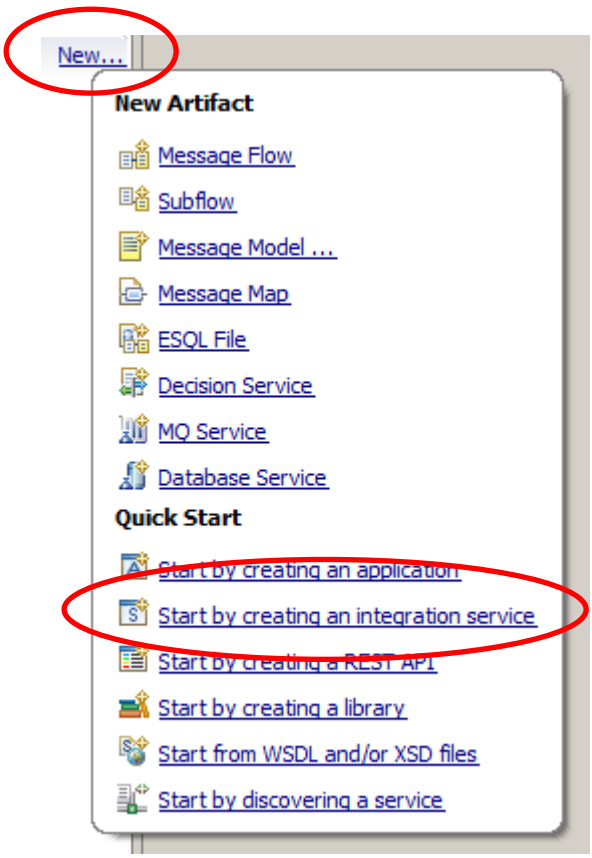
    Type the ")" to complete the XPath



    Save and close the map.

3.2 Create the Integration Service

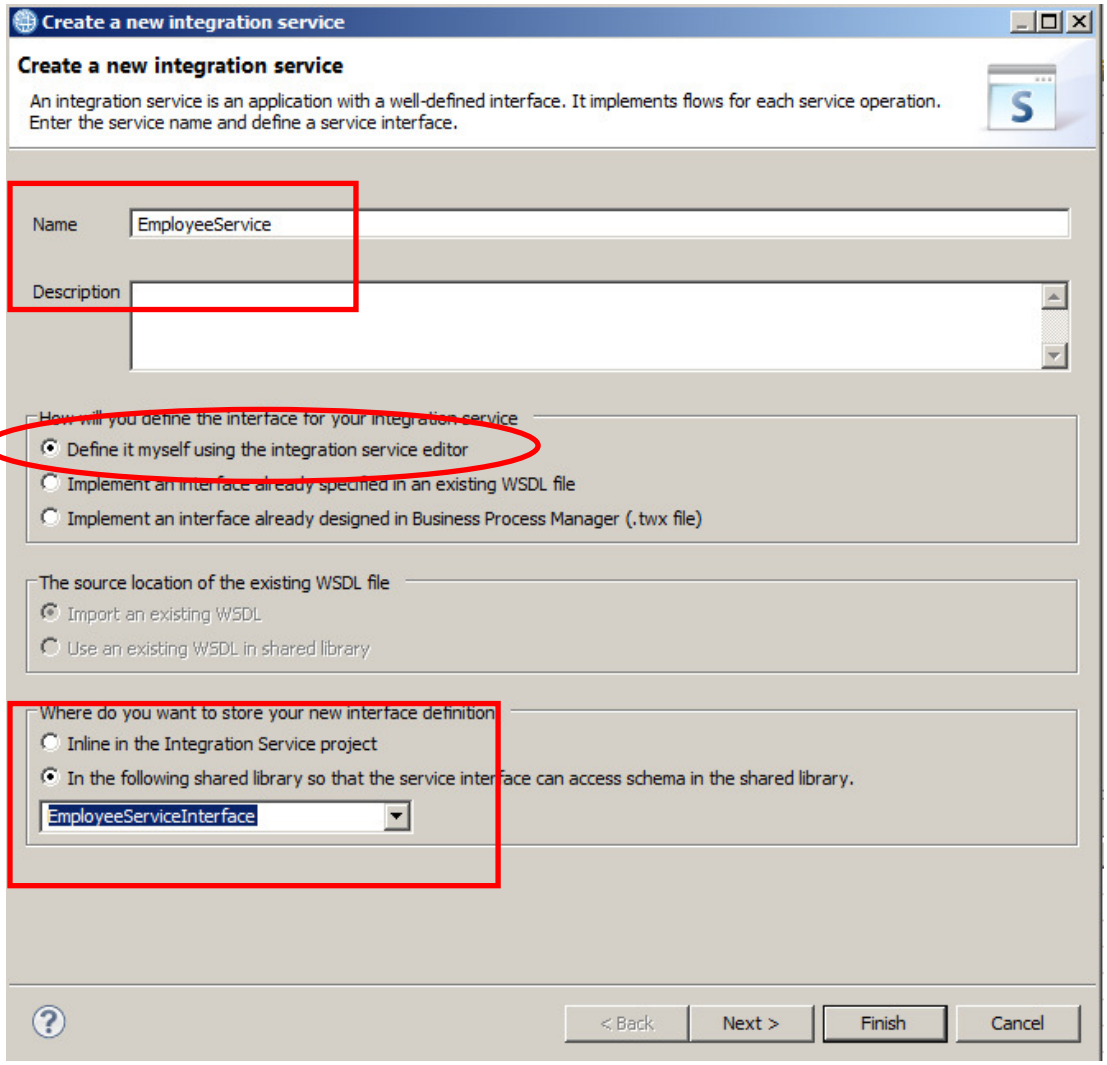
1. In the Application Development navigator, create a new Integration Service.



2. Name the service EmployeeService, and accept the default to "Define it myself ...".

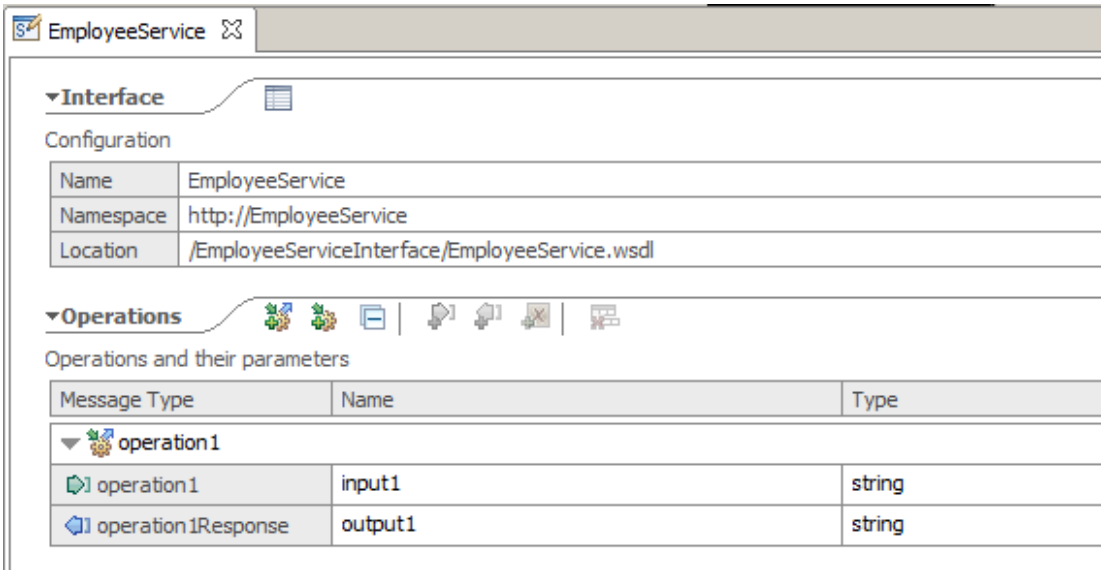
   Since the schemas have been defined in a Shared Library, you should choose the option to store the interface definition in the same library. Select this option, and specify the name of the EmployeeServiceInterface.

   Click Finish.



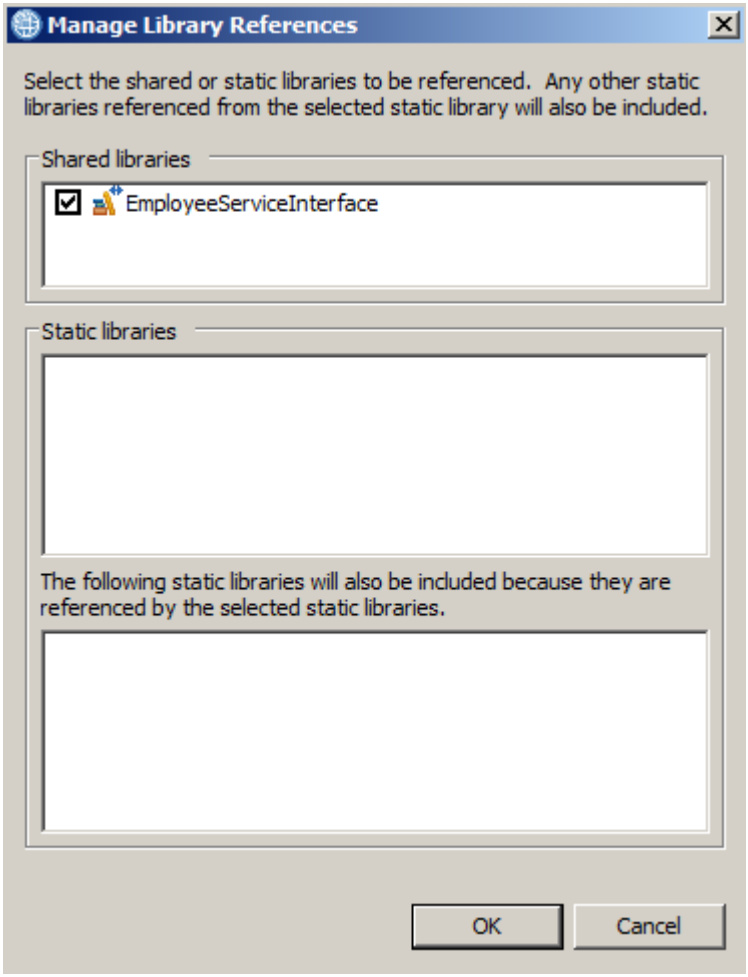
3. The following service interface will be generated. The service will contain a request/response operation named operation1.

   Note that the location of the service interface is EmployeeServiceInterface, with the name of the wsdl = EmployeeService.wsdl.

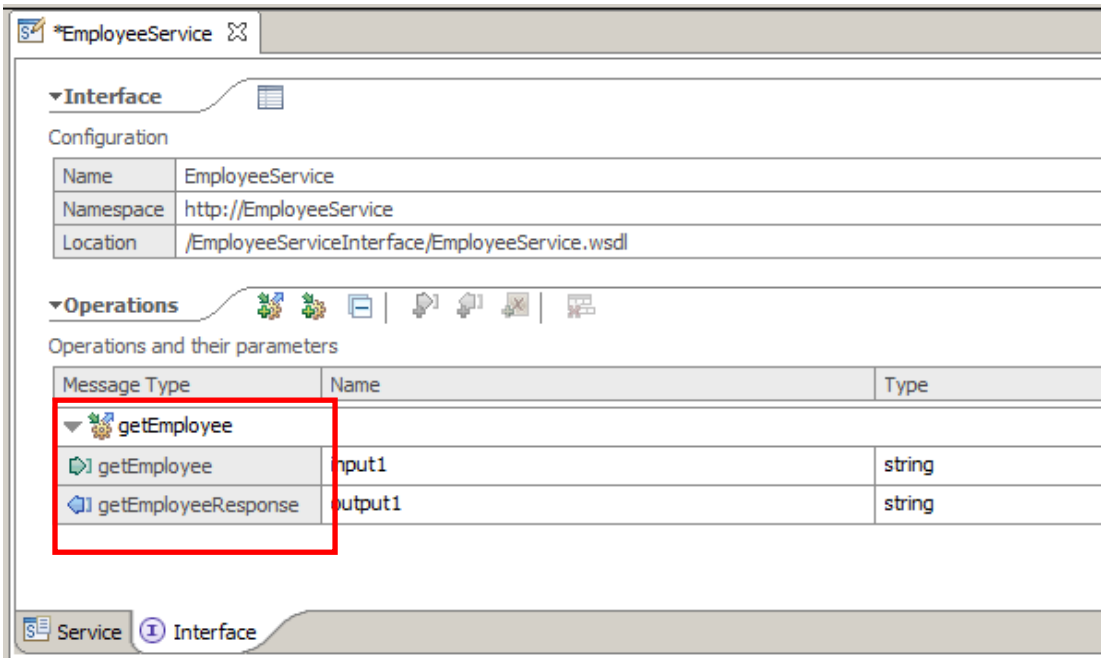


4. Note that the Library Reference of the new service has automatically been set to reference the Library where the SAMPLE_EMPLOYEE schema is defined (right-click the new Integration Service and select Manage Library References).

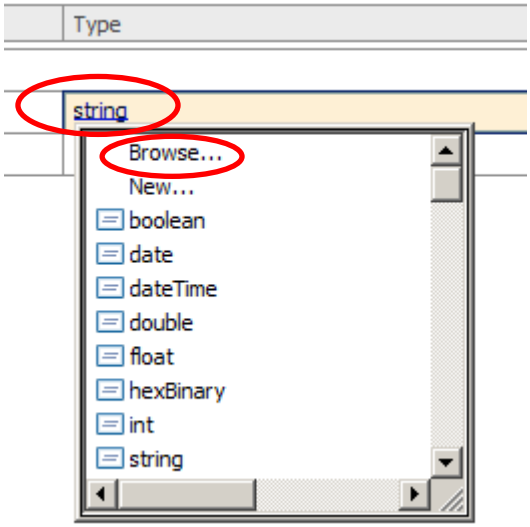
   After reviewing, cancel this window.



5. Change the name of operation1 to getEmployee:
   - Highlight operation1, and overtype with getEmployee
   - Click return after renaming, which will automatically allocate the names of the message types for the operation

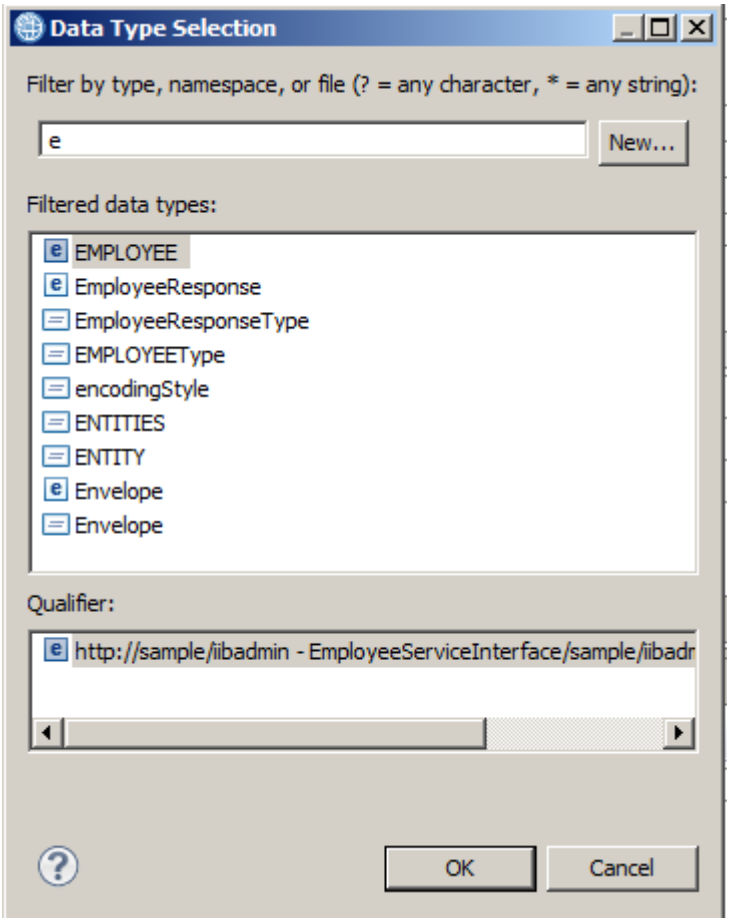


6. Change the Type of the input message to EMPLOYEE. (Click the current value "string"), then click Browse).

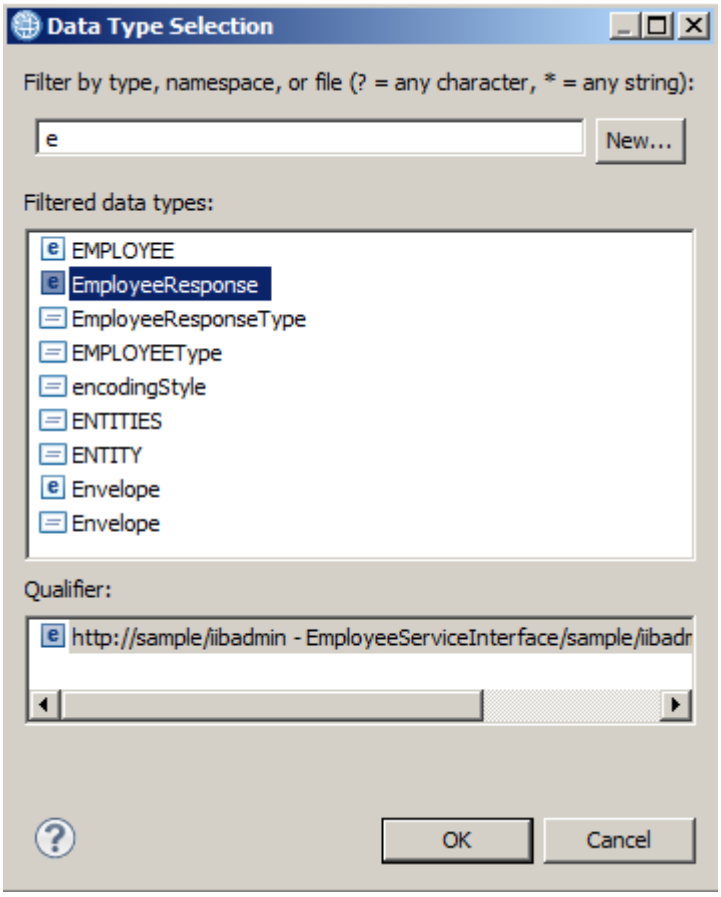


7. Type "E" into the filter, and select EMPLOYEE.

   Click OK.



8. For the output message, use the same technique and select EmployeeResponse.

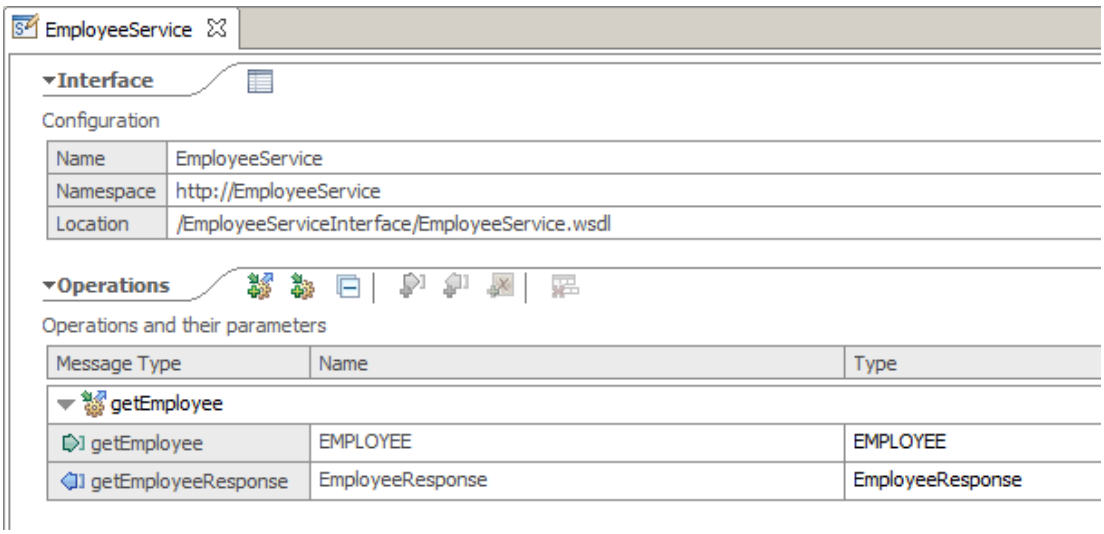


9. The getEmployee operation interface is now fully defined, and should look like this.

   Note that the name of the input message will have been changed to EMPLOYEE. This is because EMPLOYEE is an XSD element, and the "name" attribute is not permitted when using an XSD element.

   Similarly, for the output message, you have used an element type (EmployeeResponse). You could have used the element type (EmployeeResponseType), which would not have changed the name of the operation types.

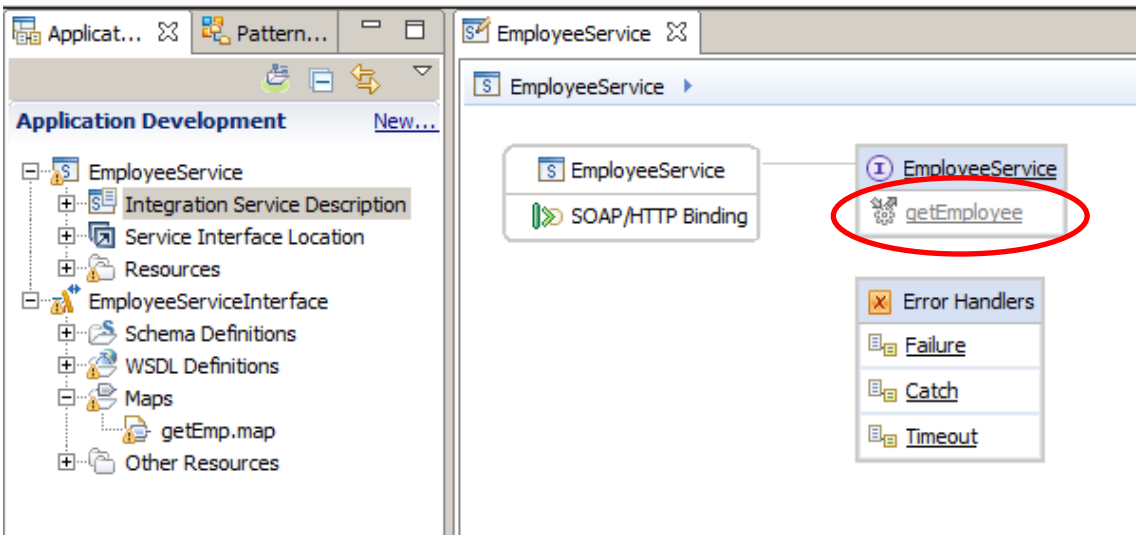
   Save the service at this point (Ctrl-S).



3.3 Implement the getEmployee Operation

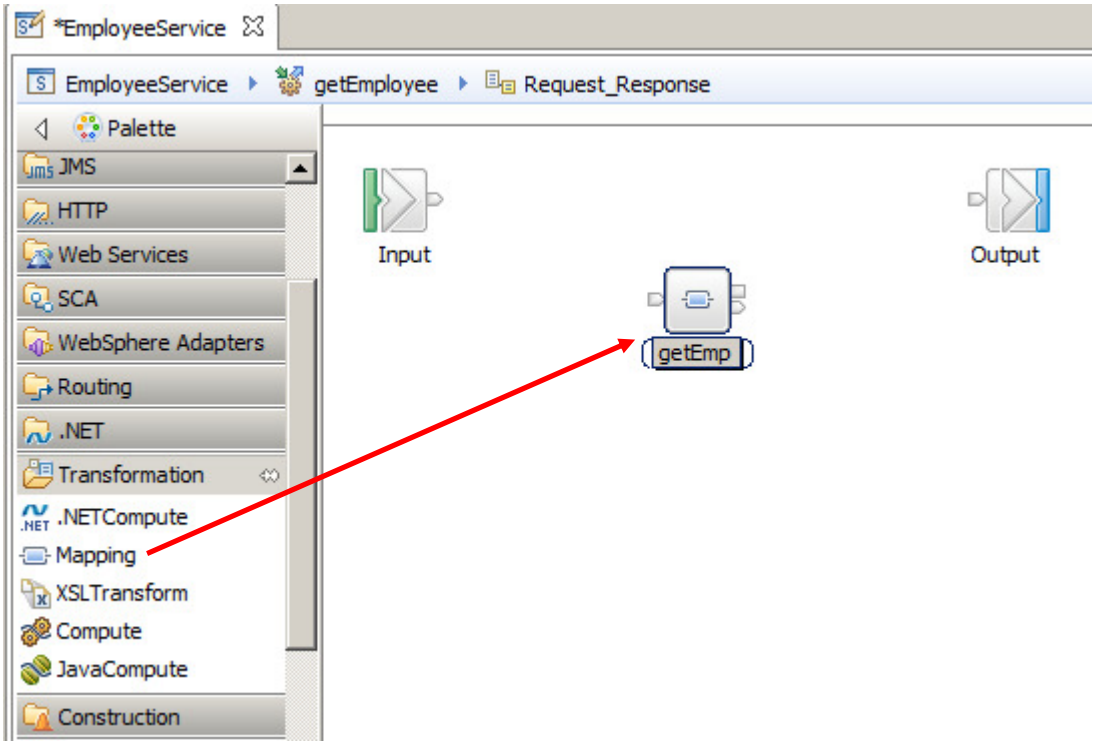
1. In the EmployeeService, click the Service tab.

   You will see that the getEmployee operation is not implemented, so is greyed out. Click getEmployee to implement it.

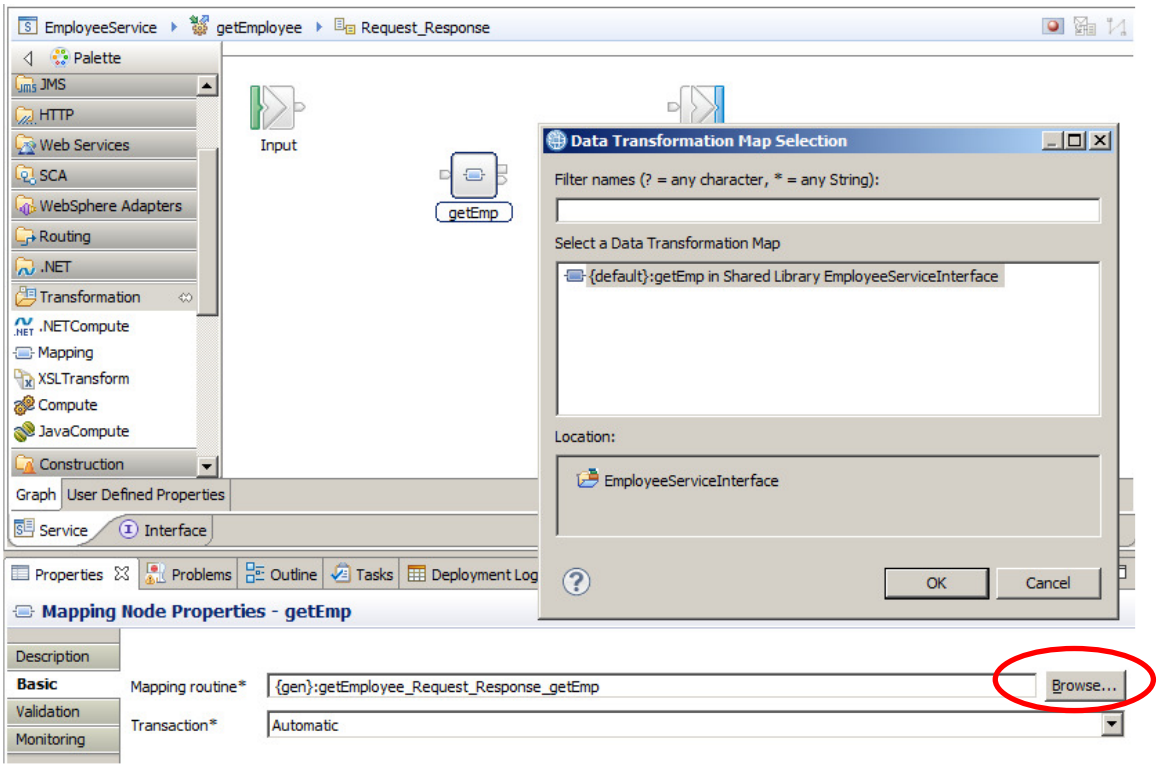


2. From the Transformation folder, drop a Mapping Node onto the flow editor.

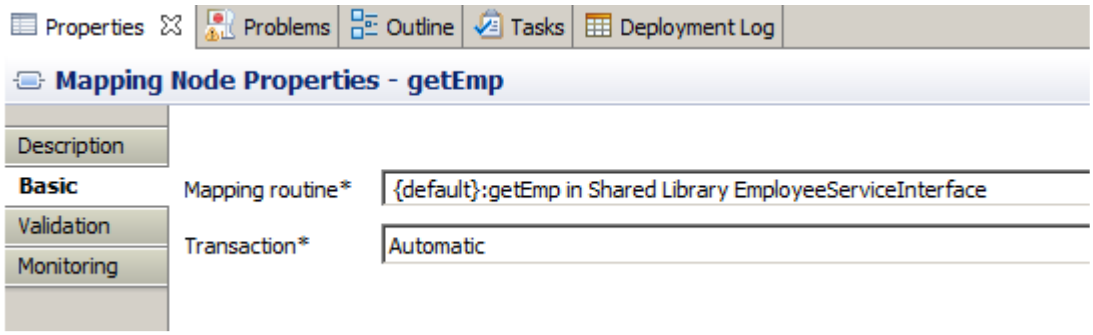
   Name it getEmp.



3. In the Properties (Basic) of the new Mapping Node, click Browse to select the map you created earlier. Select the map in the Shared Library, and click OK.



4. The new map node properties will show the map in the Shared Library.

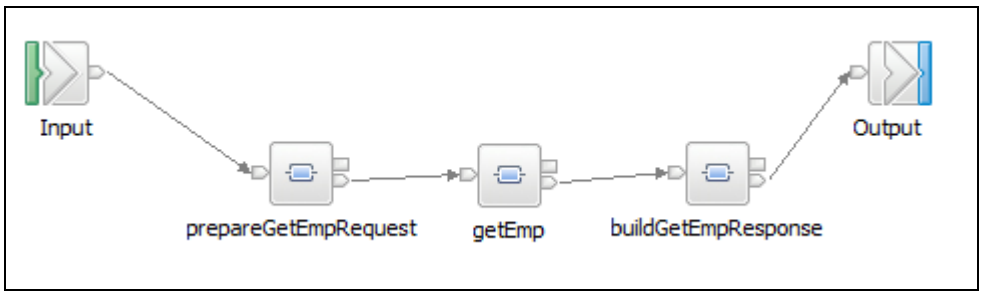


5. Because the operation is going to invoke a map that is expecting to receive only the EMPLOYEE message, you must remove the high-level message wrapper (getEmployee) that the EmployeeService has generated from the incoming message. You must also add the corresponding output wrapper (getEmployeeResponse).

   This will be done with two simple maps. Since these maps will be unique to the EmployeeService, they will be located within the integration service.

   Drop two new maps onto the flow editor. Name them:
   - prepareGetEmpRequest
   - buildGetEmpResponse

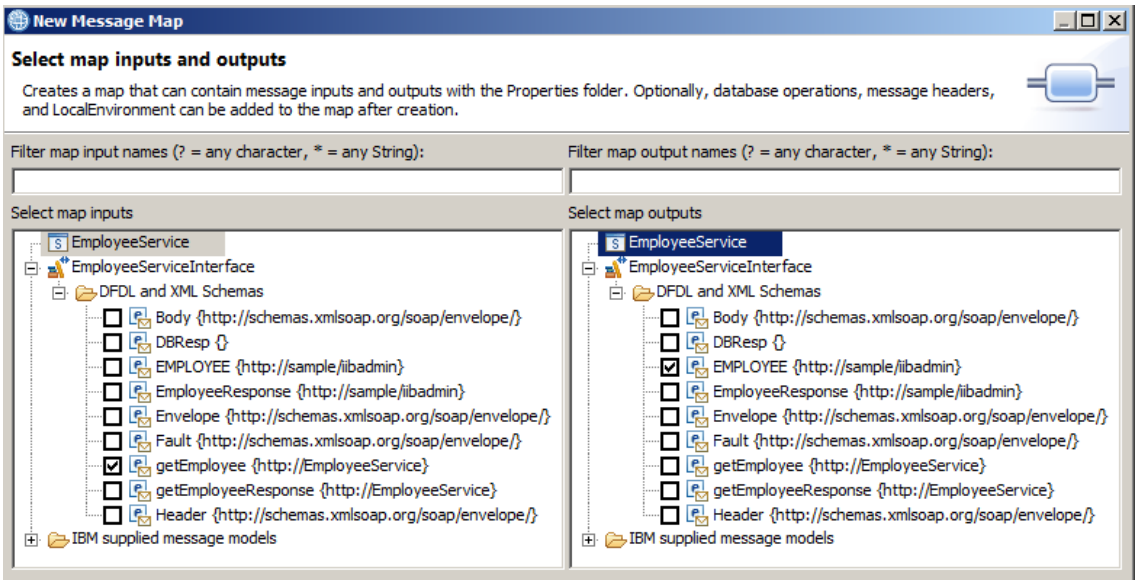
   Connect the nodes as shown.



6. Open the first new map, and at the dialogue to define the map inputs and outputs, you will now see a larger number of available schemas. These include the same schemas as before, but also the definitions of the inputs and outputs to the service operation.

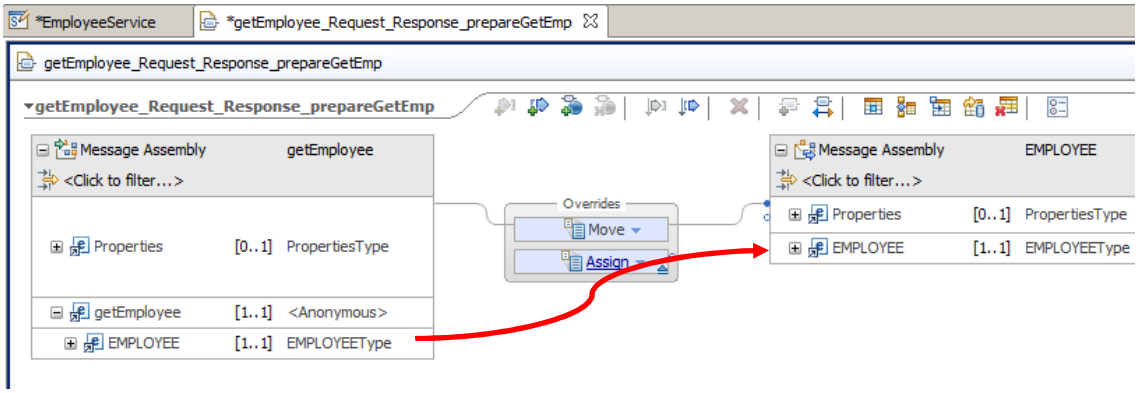
   For the input, select "getEmployee".
   For the output, select "EMPLOYEE".

   Click Next and the Finish to go to the map editor.



7. Expand getEmployee. Connect EMPLOYEE to EMPLOYEE.

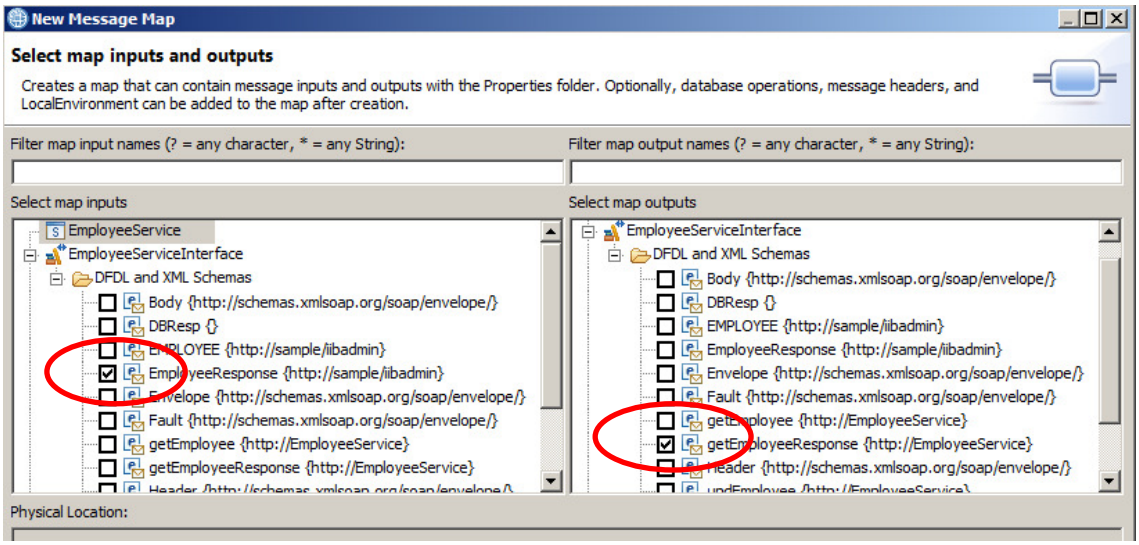
   Save and close the map.



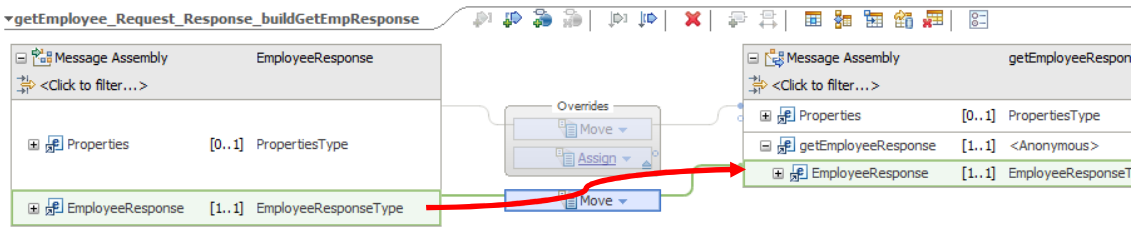
8. Open the second new map. At the dialogue to create the map inputs and outputs:

   For the input, select "EmployeeResponse".
   For the output, select "getEmployeeResponse".

   Click Next and Finish to proceed to the map editor.



   In the map editor, expand getEmployeeResponse (output), and connect EmployeeResponse to EmployeeResponse (a Move transform).



   Save and close the map.

3.4 Remove schema validation

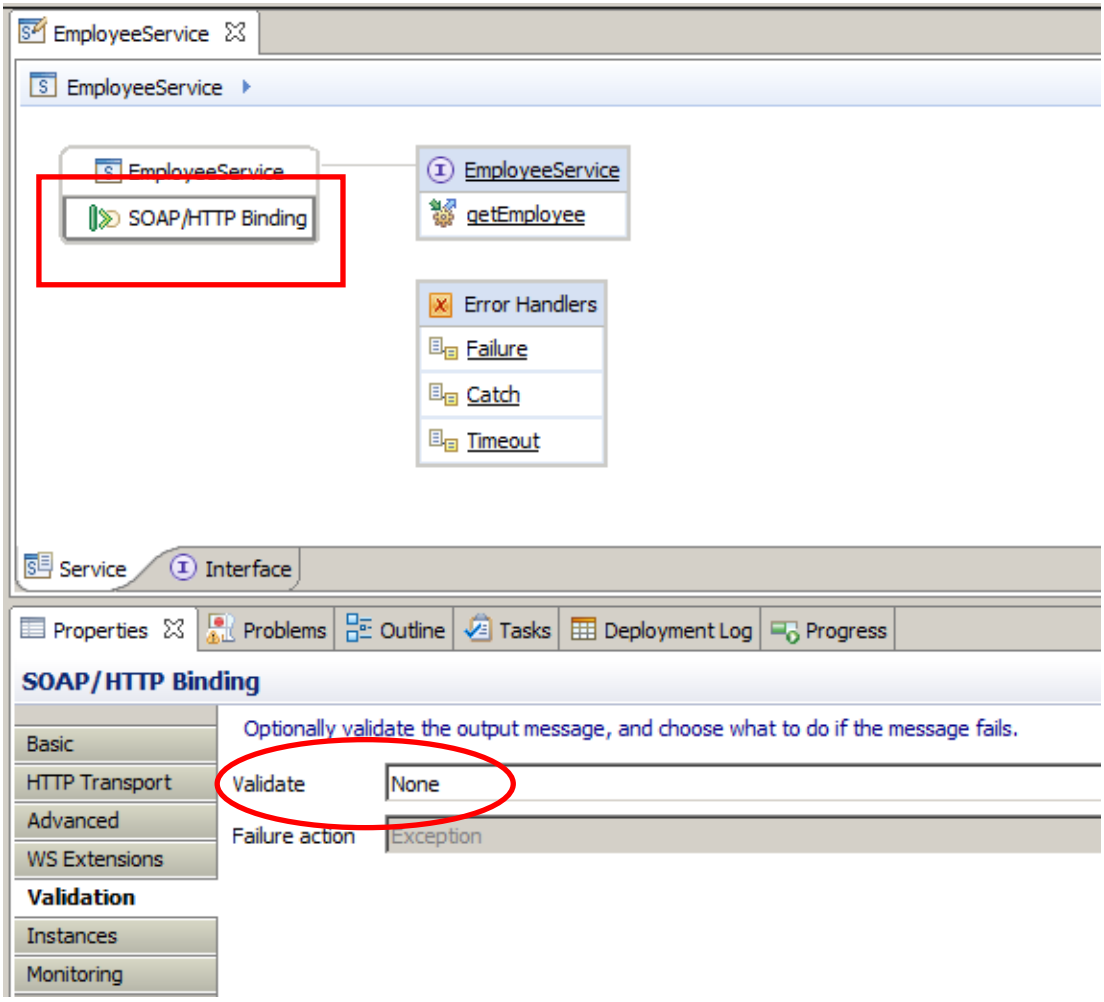
1. Finally, for ease of testing, remove the schema validation for incoming SOAP requests. This is necessary because some of the columns in the EMPLOYEE table definition had certain value constraints, and these have been inherited by the Message Model schema.

   Highlight the EmployeeService, and click the "SOAP/HTTP Binding".

   In the Properties pane, click the Validation tab.

   Set the Validate property to None.

   Save the service.

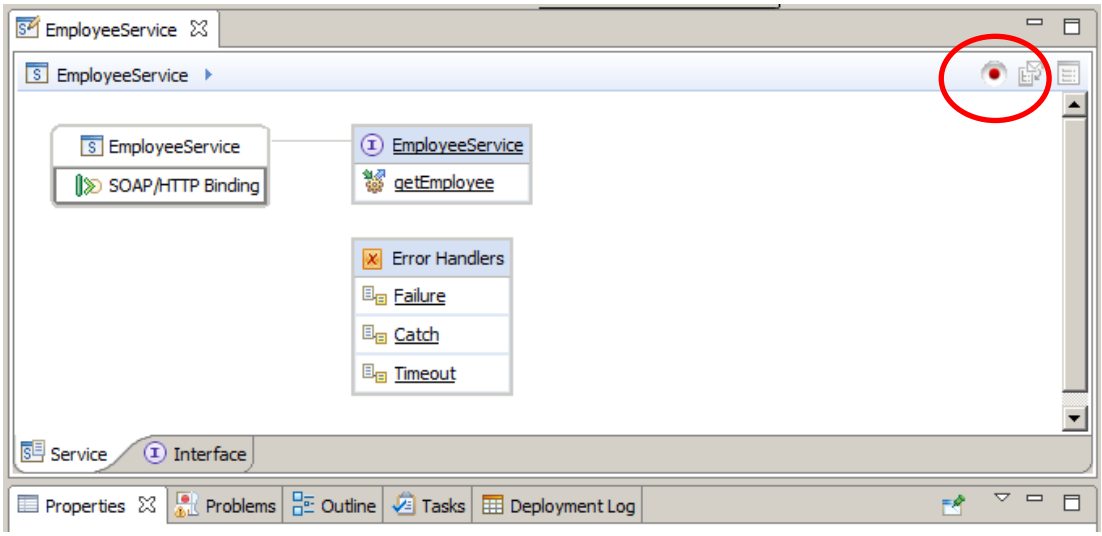


4. Test the Integration Service with the Flow Exerciser

The Flow Exerciser is a new function in IIB v10. You will use this to perform a simple unit test of the service you have just developed. You will see additional Flow Exerciser functions in later labs.

First, make sure that the node IB10NODE is started. You can check this in the Toolkit Integration Nodes pane. If it isn't started, right-click and select Start.

1. In the service editor, make sure the Service tab is selected. You will see a red button in the top right of the editor pane.

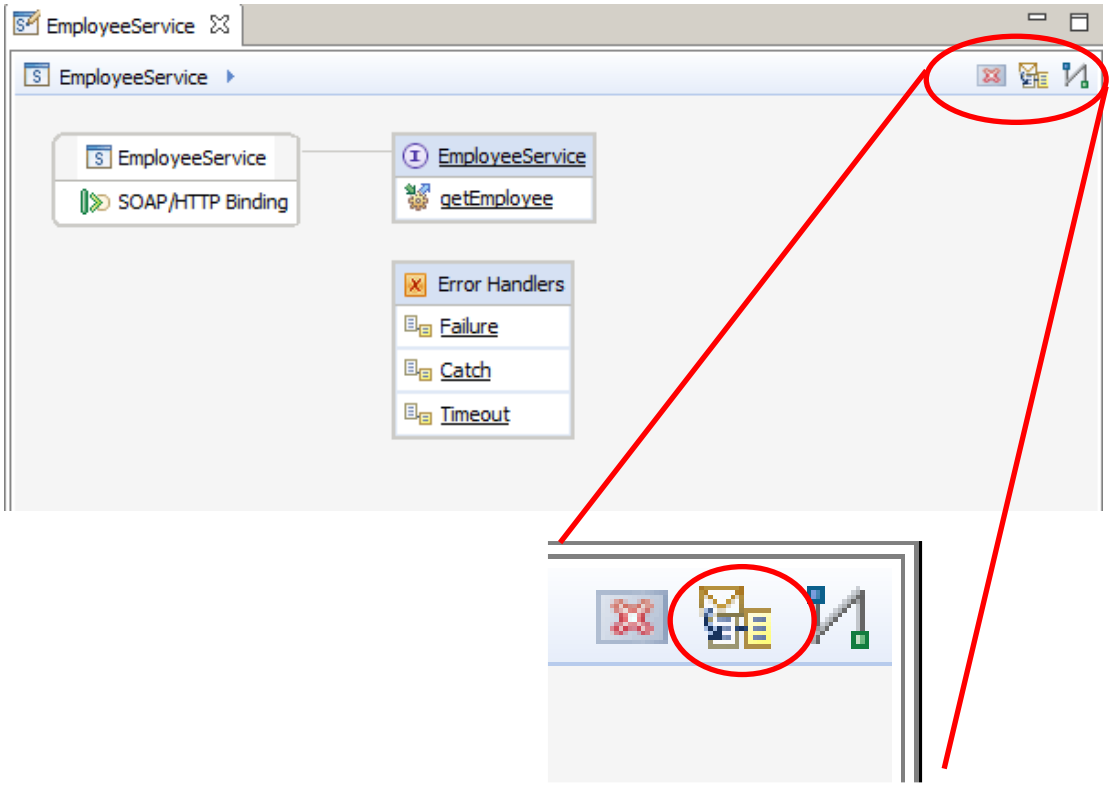


2. Click the red button. The integration service will be deployed to the IB10NODE/server1. Note that if you have other nodes or servers running, you will be presented with a selection dialogue.

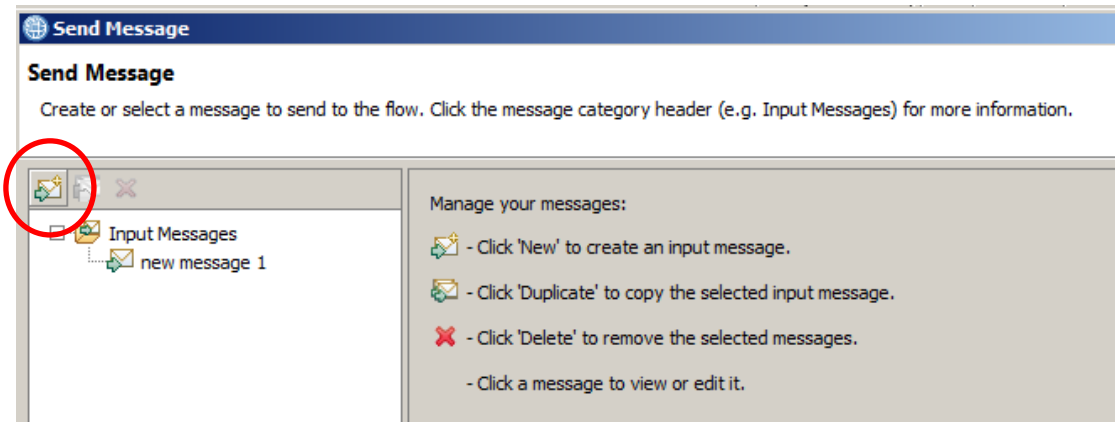
   When the service is deployed, you will see an information message. After reading this, select the box to not see in the future, and click Close.

3. The flow editor will turn grey, indicating that the Flow Exerciser is ready for use.

   In the top right, select the "Send message" icon (the middle icon).

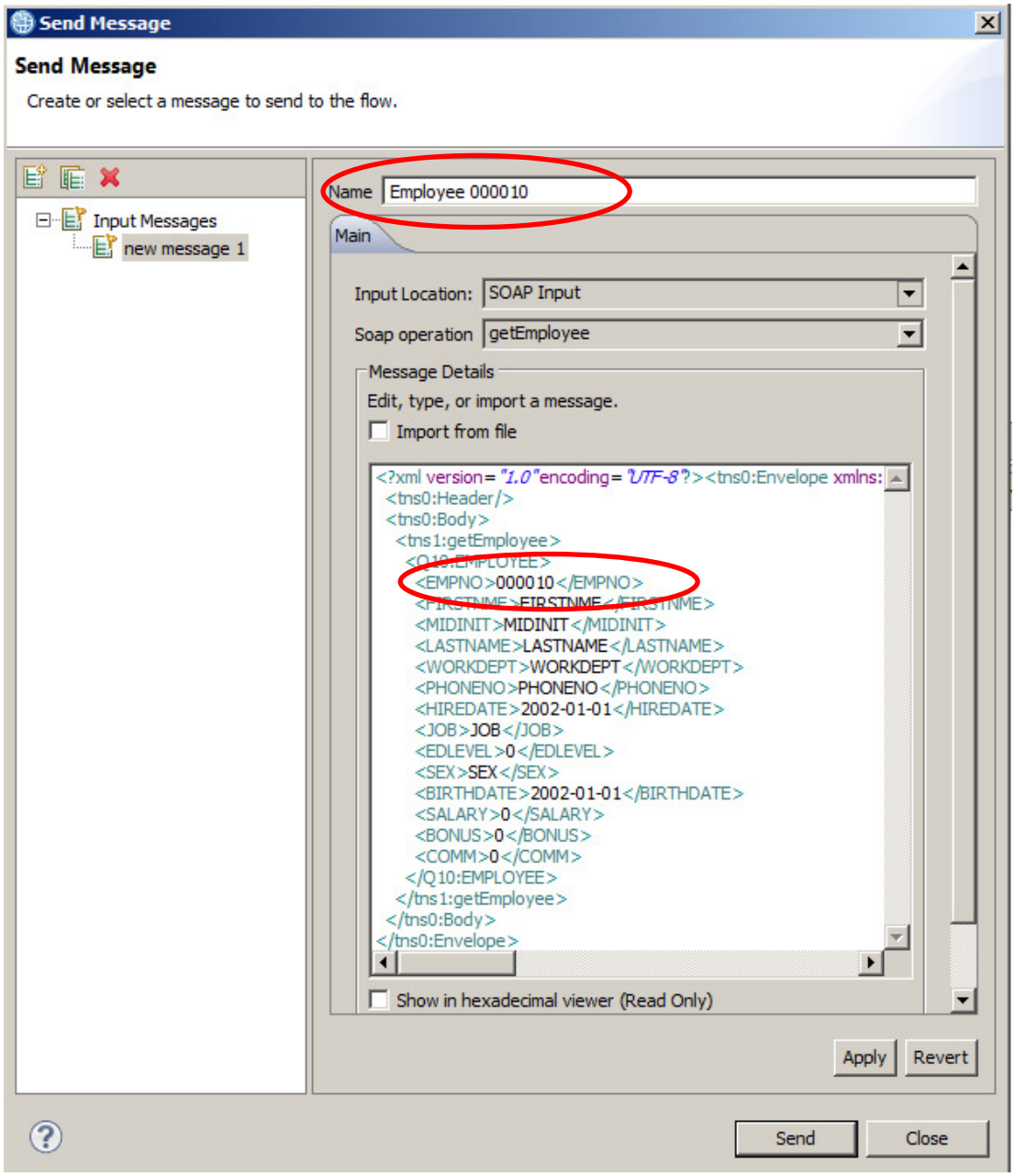


4. In the Send Message window, click the button to create a new message.

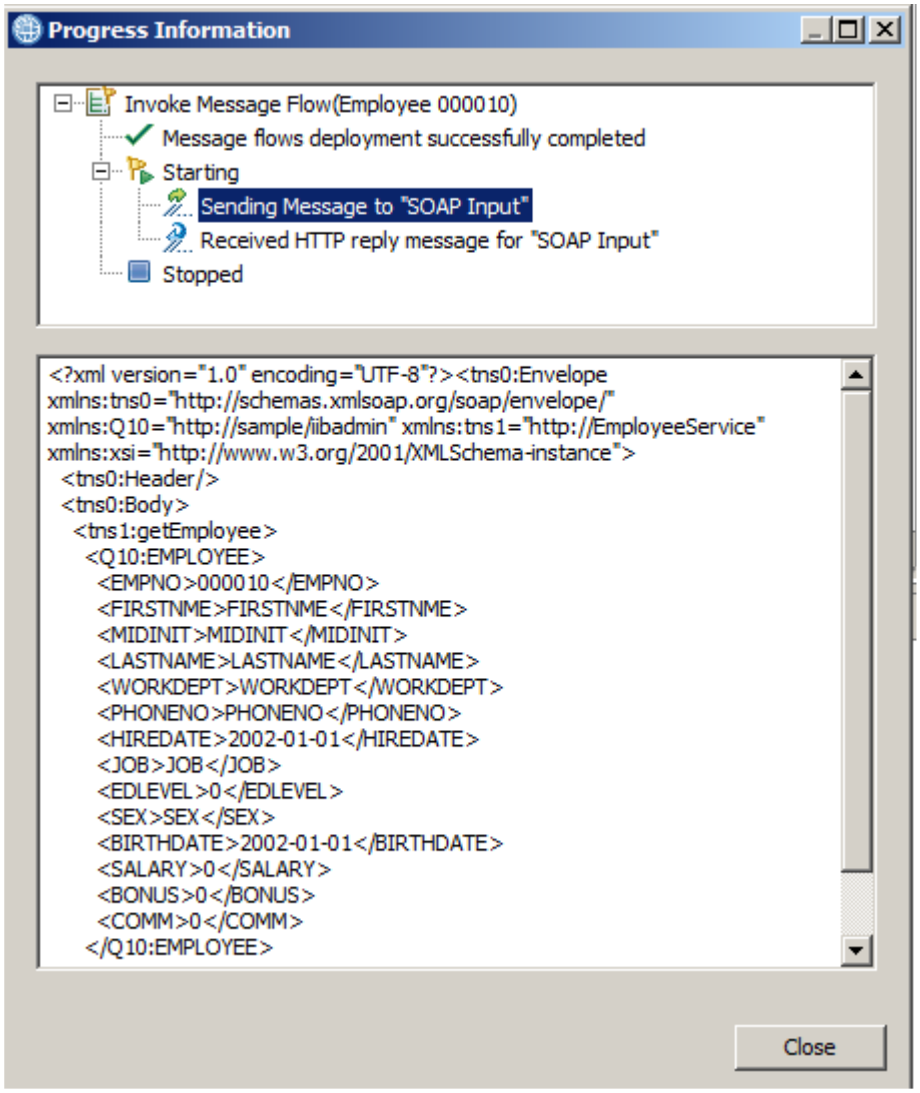


5. Name the new message "Employee 000010".

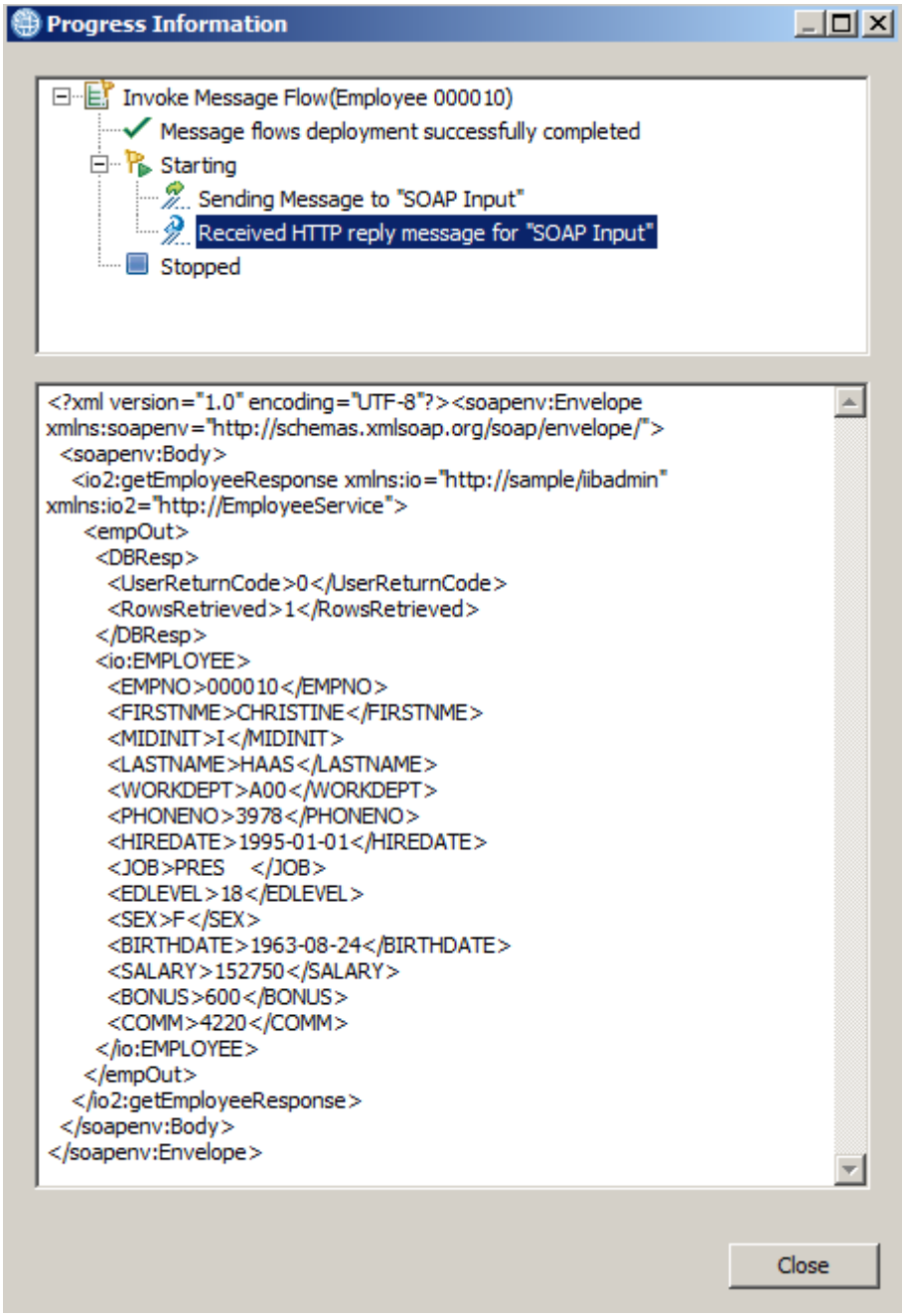
   The Flow Exerciser will have automatically populated a template input message, based on the WSDL schema. In the EMPNO element, change the value to 000010, and click Send.



6. The service operation will be executed. Highlighting the "Sending Message" line will show the message that was sent to the service.

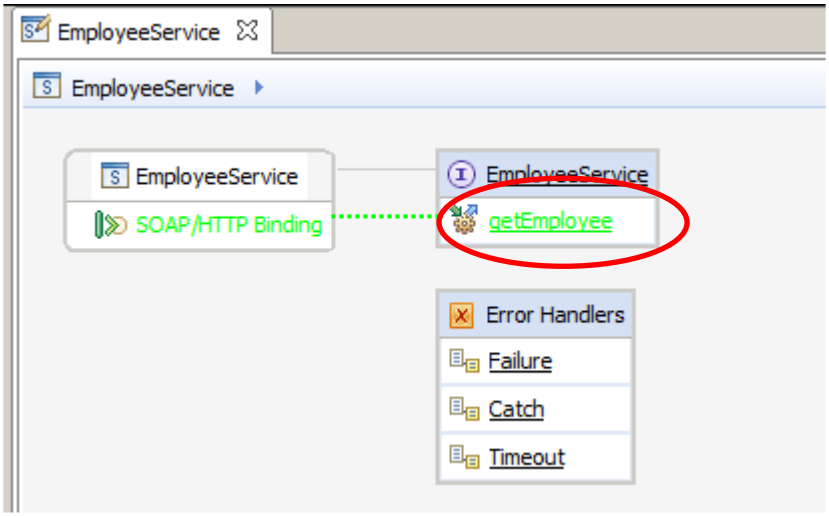


7. Highlighting the "Received HTTP" line will show the message returned from the service. Note that the employee record for the selected employee number has been retrieved from the EMPLOYEE table.

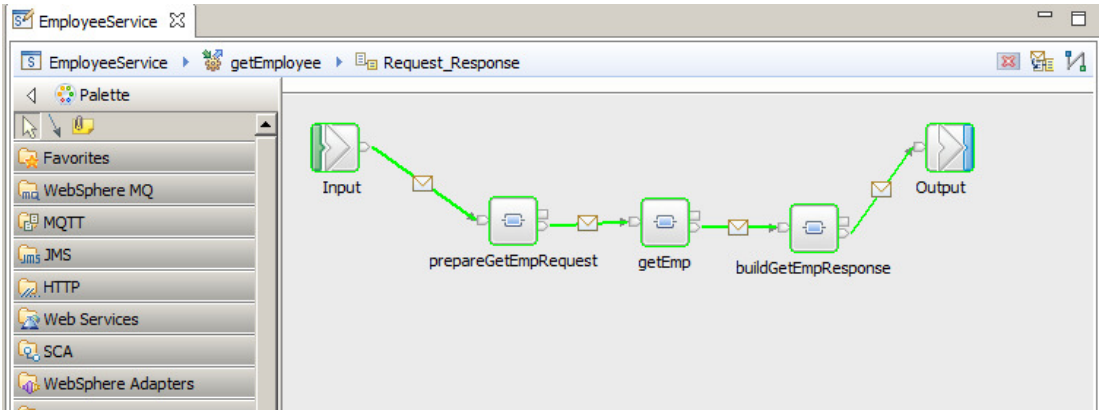


Close the Progress Information window.

8. In the flow editor, you will now see that some parts of the service definition have turned green. Click the "getEmployee" operation name.



9. In the flow editor, you will see the subflow that implements this operation. You will see that the connectors are green, indicating that the message flowed down this path.

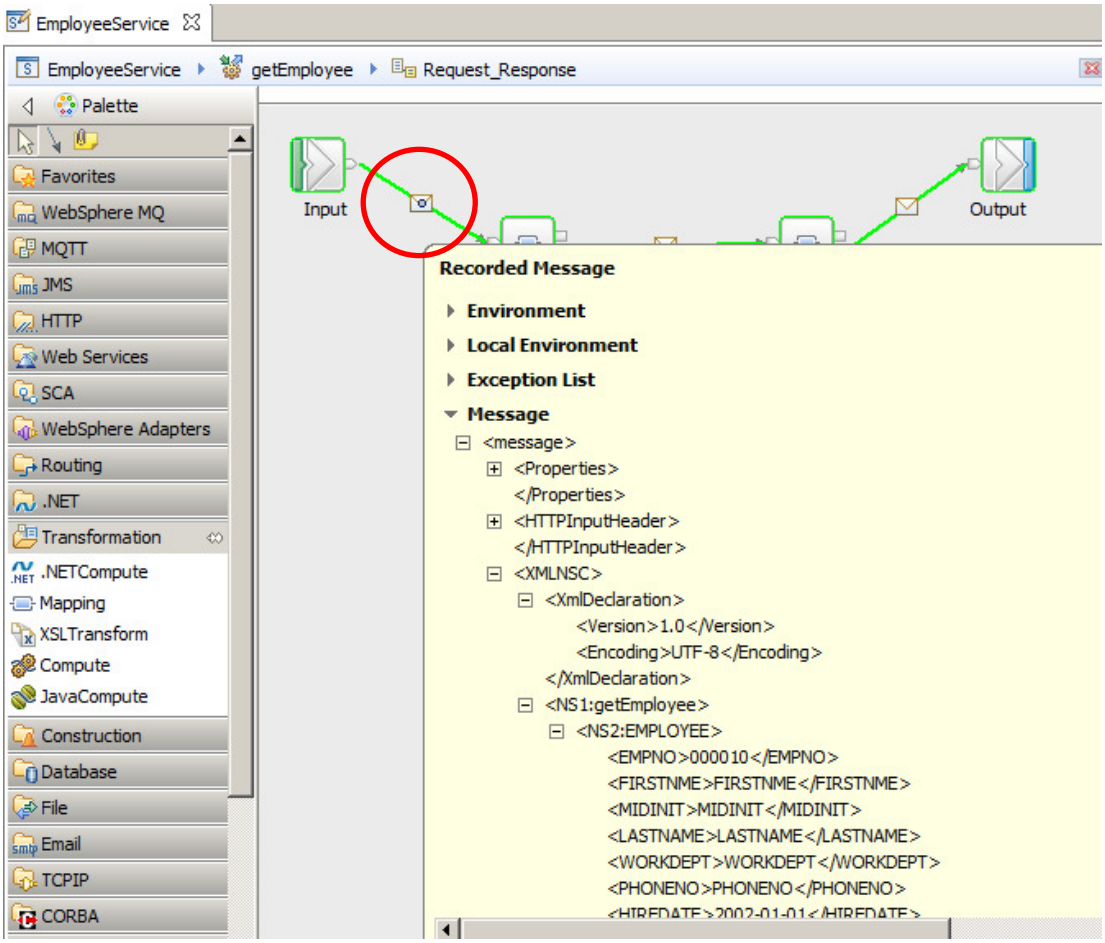


10. Click the icon on the first connector.

    The message tree that was current at the time it passed through this connector will be retrieved and displayed. Note that this is the full message tree, not just the user data.

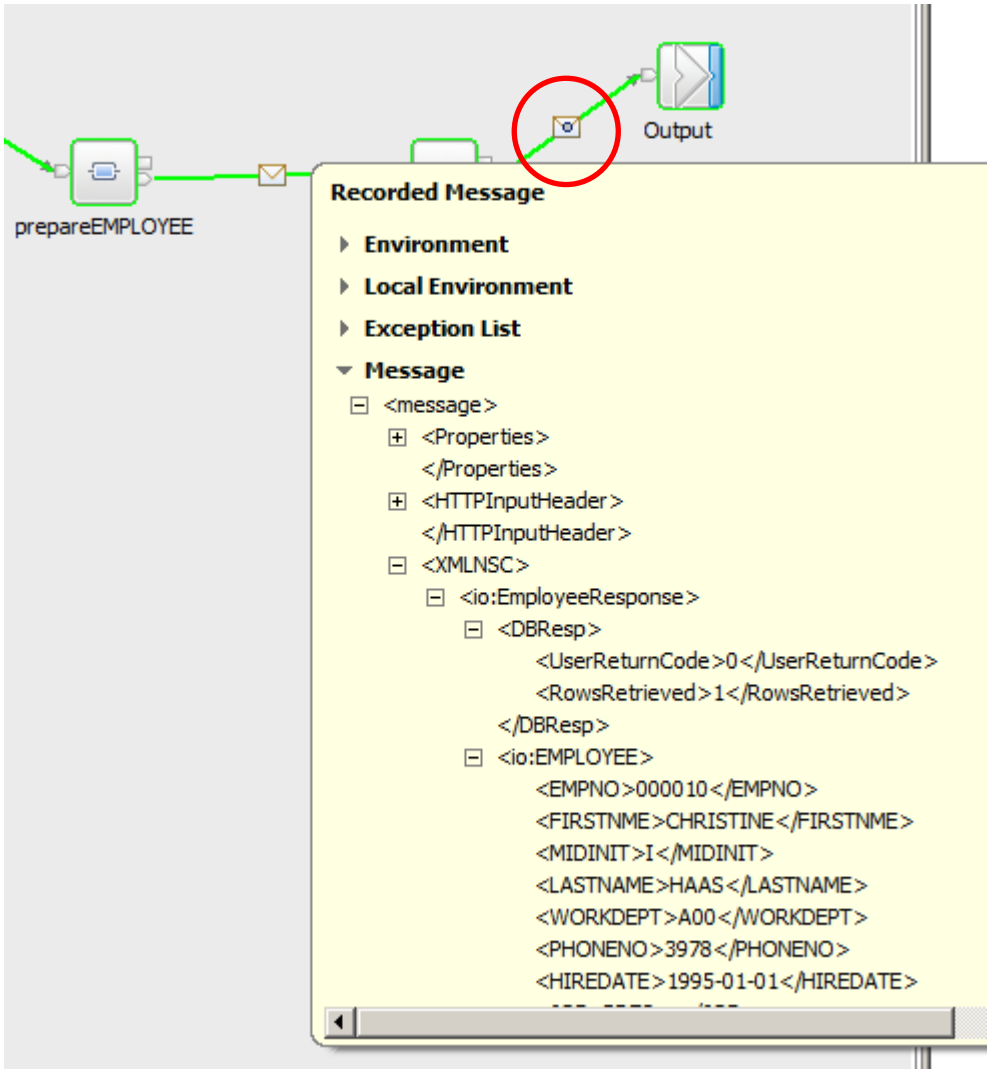
    You can expand various parts of the tree, such as LocalEnvironment, Environment, ExceptionList, as well as the Message.

    Note that the domain of the message tree is shown. In this case, it is XMLNSC.

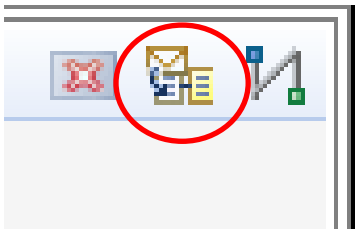


11. Click the icon on the final connector. This time, you will see the message tree after the employee record has been retrieved from the database.

    Note that you can return to the first connector icon to review the message tree again, at that point. The data is not discarded.

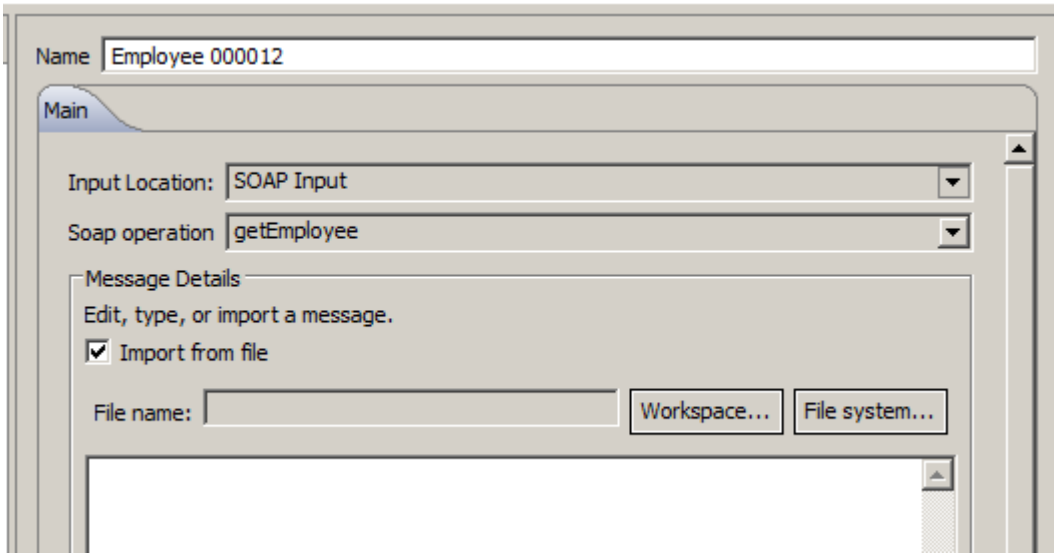


12. Click the Send Message icon again.

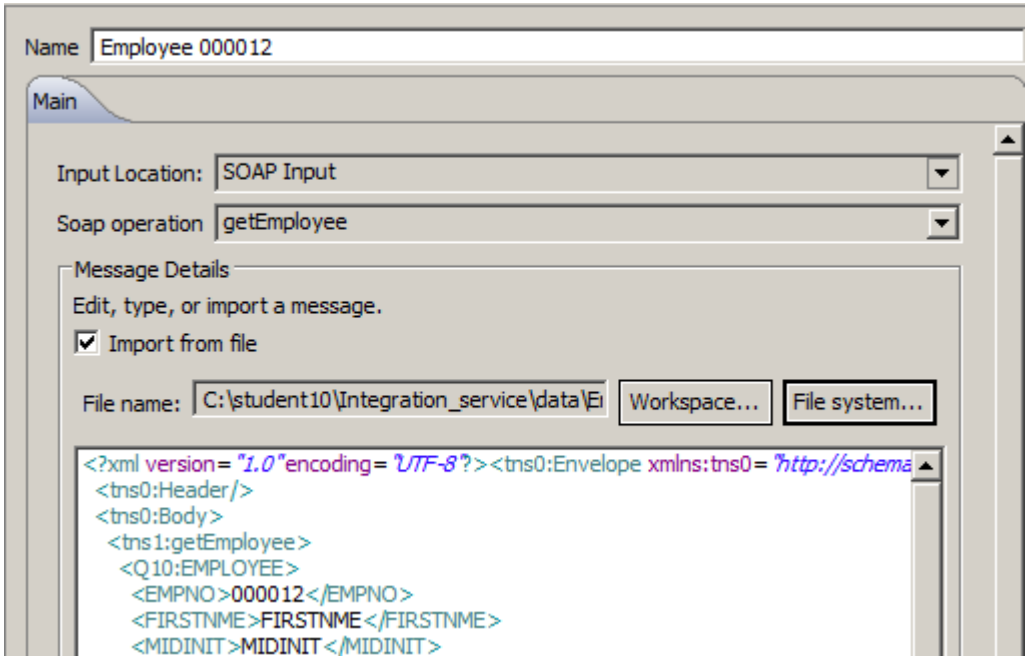


13. Create a new test as before. Name it "Employee 000012".

    This time, select "Import from file". Using the File system button, navigate to c:\student10\integration_service\data\Emp000012.xml.



14. Click Send, and use the same tools to review the test outcome.



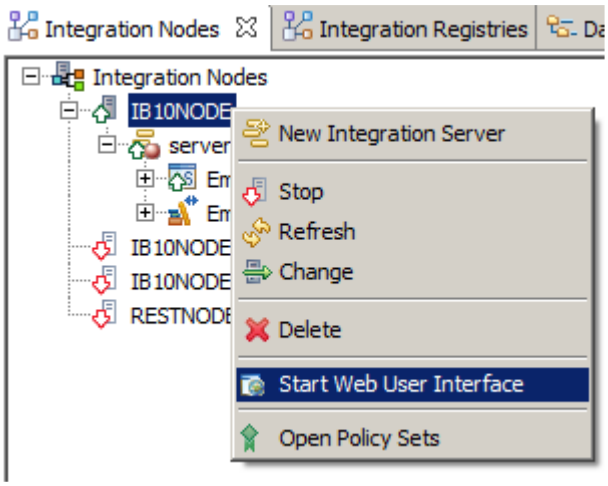
5. Export the Service Interface and retest

5.1 Export the Service Interface from the IIB Node

1. The service interface files (WSDL and XSDs) can be obtained either from the development artefacts in the IIB Toolkit, or from the deployed version of the service. In this lab, you will obtain the interface from the deployed version; this will ensure that the port number of the integration server is specified correctly.

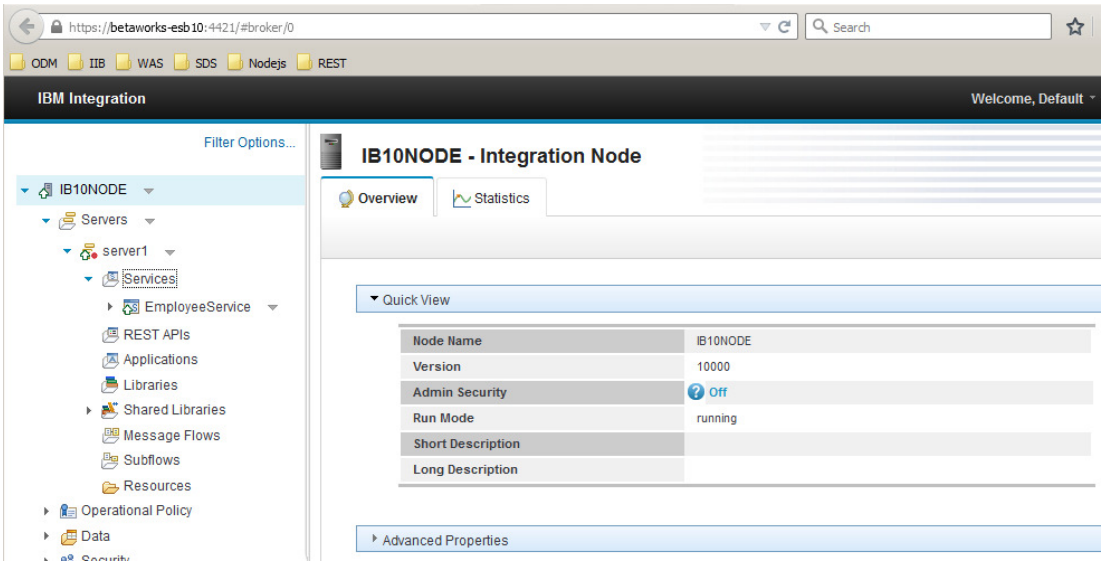
   Open a Firefox browser, and connect to IB10NODE using the shortcut in the IIB folder. Note that this will connect to IB10NODE using a HTTPS connection.

   Or, you can open the browser direct from the IIB Toolkit, by right-clinking IB10NODE, and selecting "Start Web User Interface".



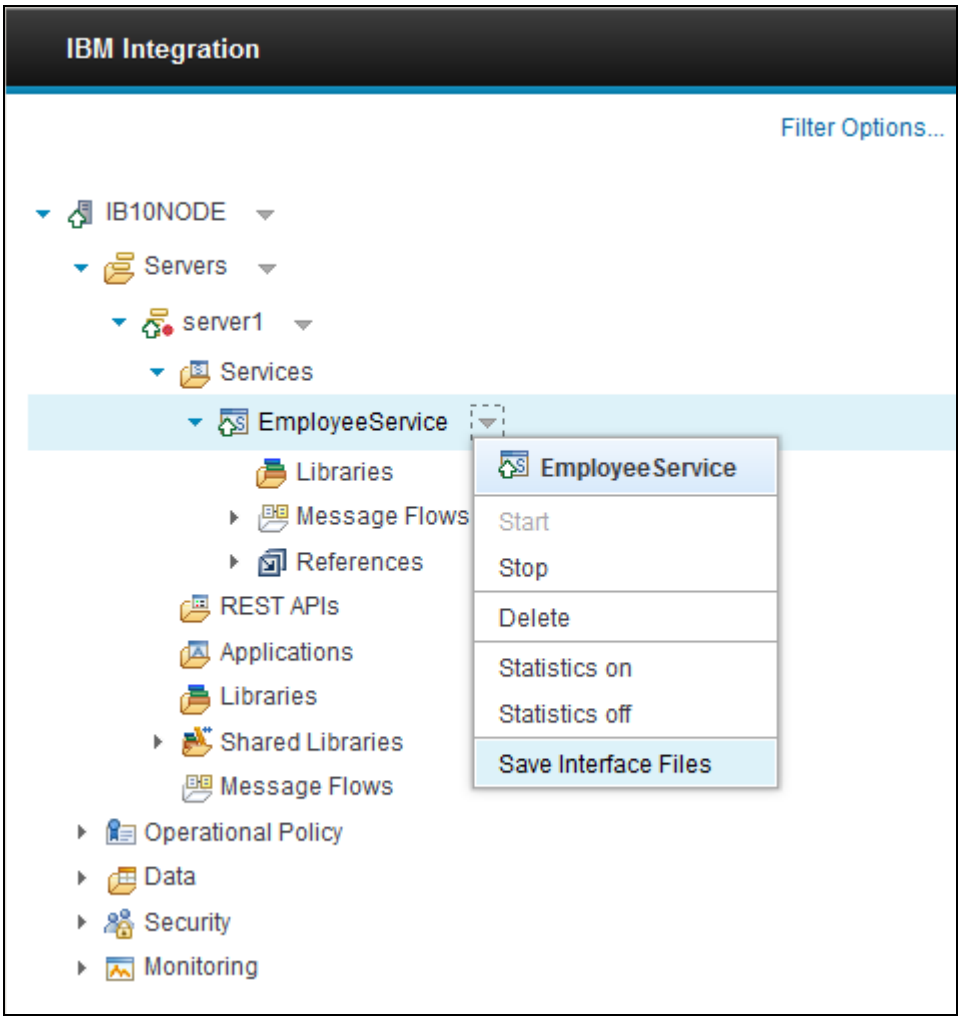
   Expand IB10NODE, Servers, server1, and display the EmployeeService service.

   Note various deployed properties of the service in the Overview pane.

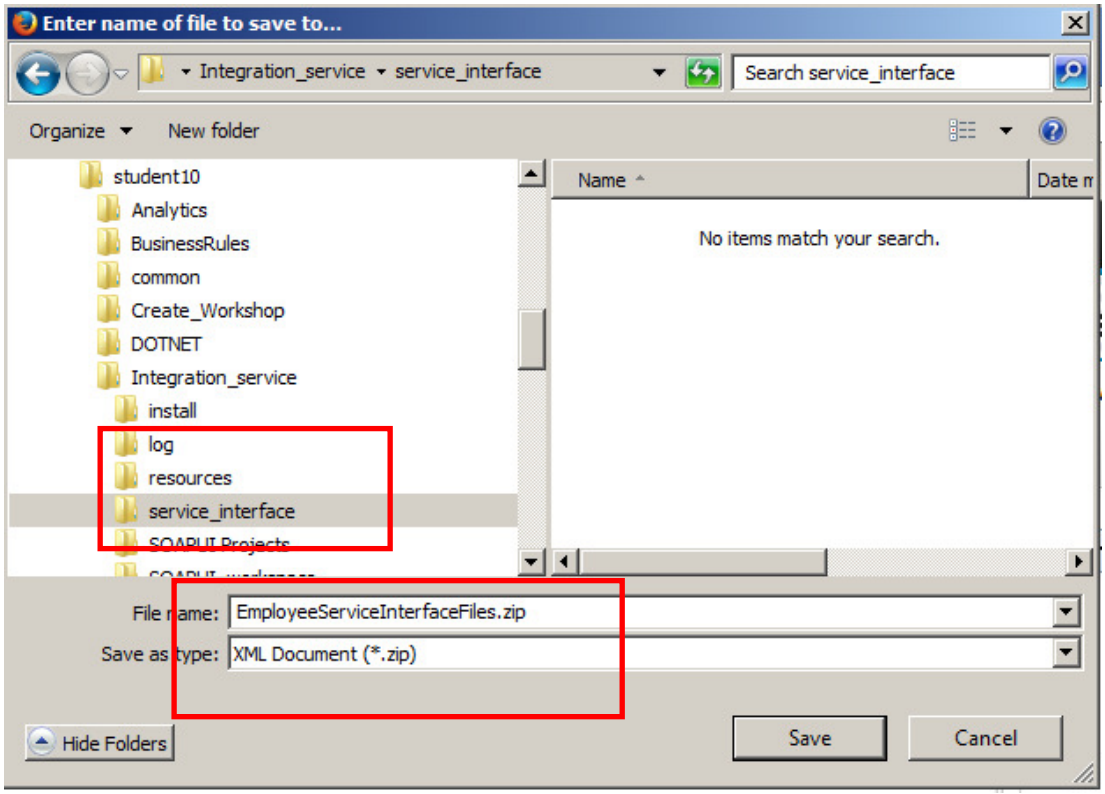


2. Click the drop-down arrow on EmployeeService, and select "Save Interface Files".

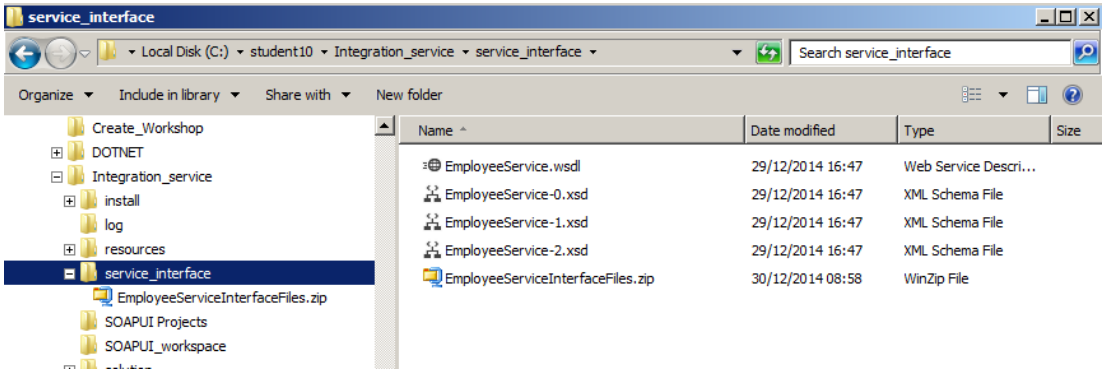
   (Also note that this window shows the "Flow Exerciser" recorder as being active - observe the red circle on the integration server "server1". You can stop recording through this web browser by clicking the drop-down on server1 and selecting Stop Recording).



3. Save the exported file as EmployeeServiceInterfaceFiles.zip under c:\student10\Integration_service\service_interface.



4. Open the downloaded zip file (7-Zip is installed on the VM image), and extract the included files into the same folder.



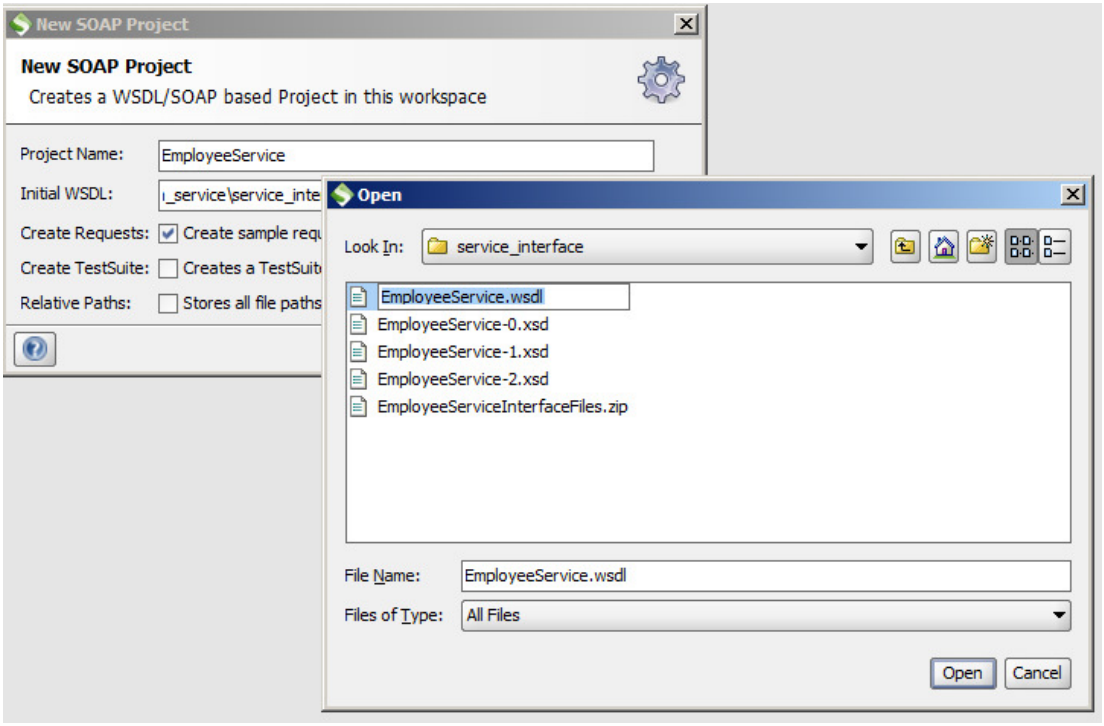
5.2 Test the service with SOAPUI

1. Open SOAPUI (on the Windows Start menu). Ensure you are using the SOAPUI workspace **EmployeeService_PrebuiltWorkspace**.

   Create a new SOAPUI project (File, New SOAP Project).

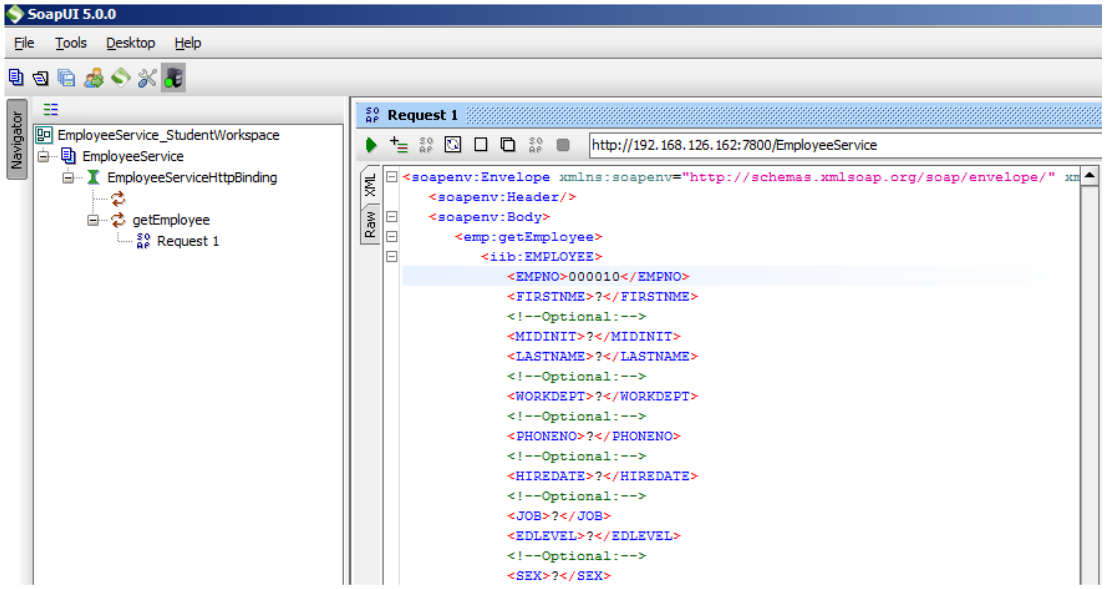
   Provide a suitable name of your choosing for the project (not one of the existing projects), and use the Browse button to navigate to the WSDL file that you exported from IB10NODE.

   - Navigate to the c:\student10\integration_service\service_interface folder.
   - Select the EmployeeService.wsdl file, click Open, then OK.

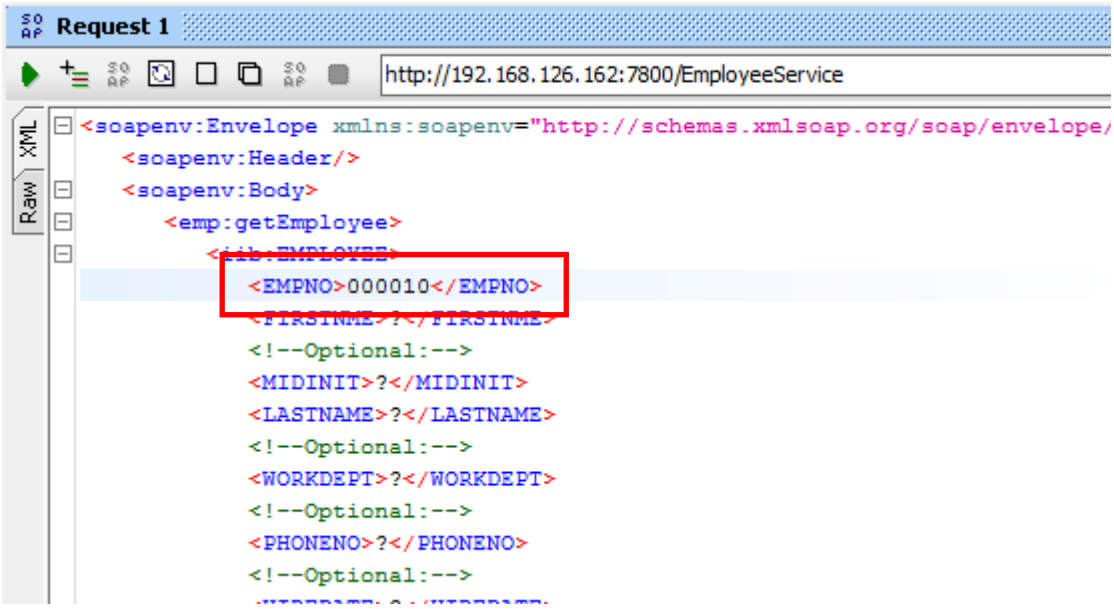


2. Expand the new project, and open Request1. You will see that SOAPUI has pre-populated the web service input with the required schema elements. In this case, these are derived from the EMPLOYEE table.

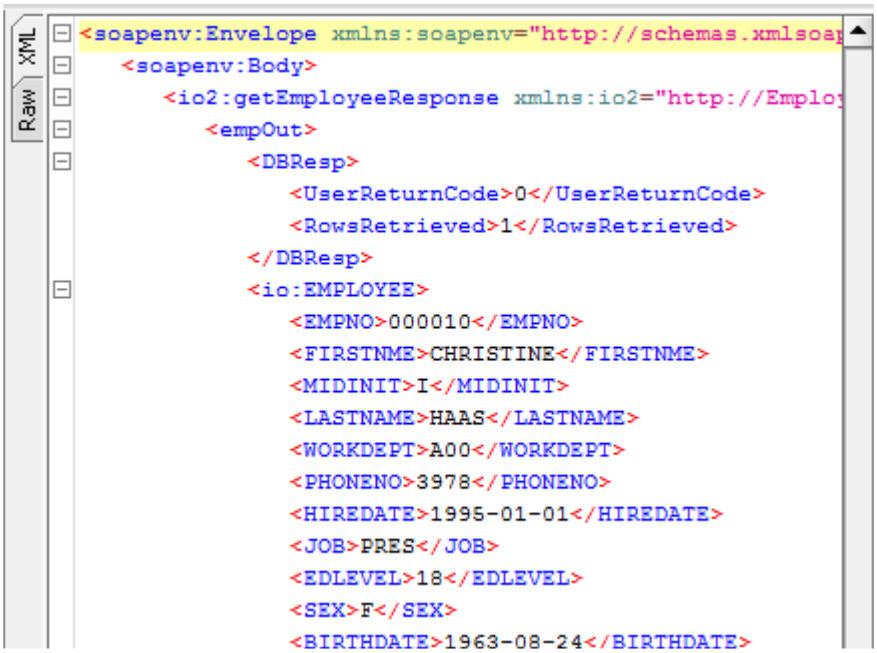
   Also note that the service port number has been set to 7800. This is the port number that has been allocated for the IB10NODE server1 server.



3. Provide a value for the EMPNO element. We suggest using value 000010 (replace the ? with this value, as shown).



4. Clicking the green arrow (top left), and the Integration Service should run and return the data to the client.



5. Create a second SOAPUI request in the same project. This time, use the value 000012, which does not exist in the EMPLOYEE table.

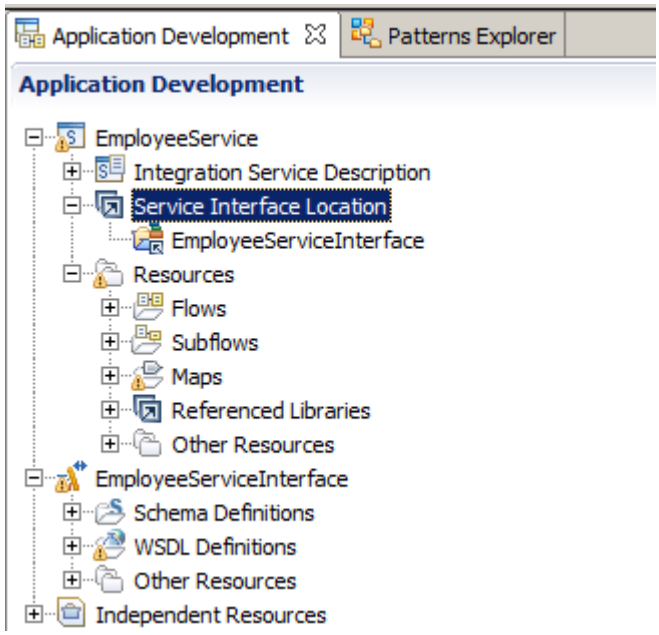
   When you click the green arrow, you should see the response indicating that the flow successfully accessed the tables (user return code 0), and no rows were returned.

5.3 Export the Service Interface from the IIB Toolkit (optional)

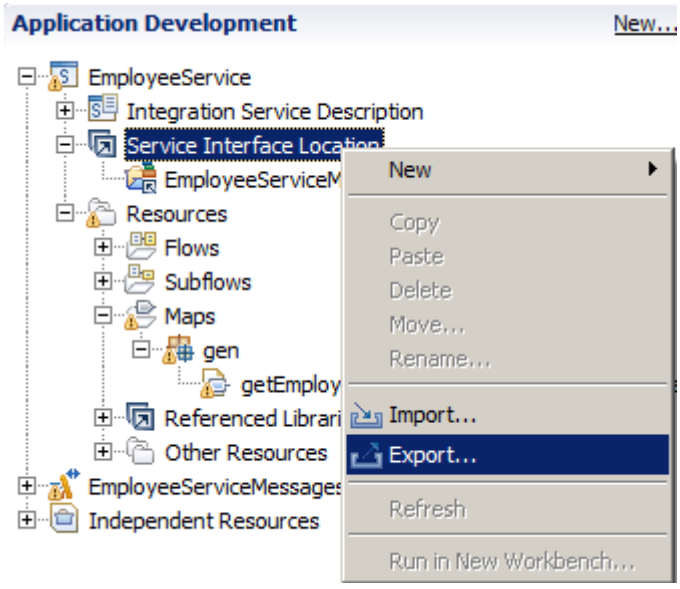
Instead of exporting the service interface from the runtime environment, you can obtain the same files direct from the Toolkit. This section is not required for the lab, but is included for reference.

Note that you would need to provide the connection details of the service (IP address, etc) if you use this method when running the service.

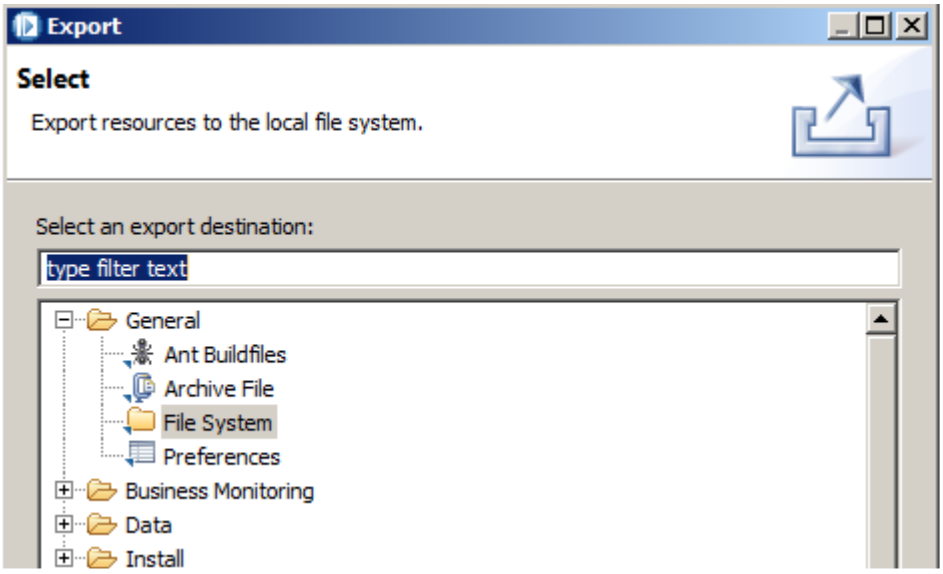
1. In the Toolkit Navigator, note that the EmployeeService has a folder called Service Interface Location.



Right-click this and select Export.



2. Expand General and click File System.



3. In the dialogue window, a number of files need to be selected.

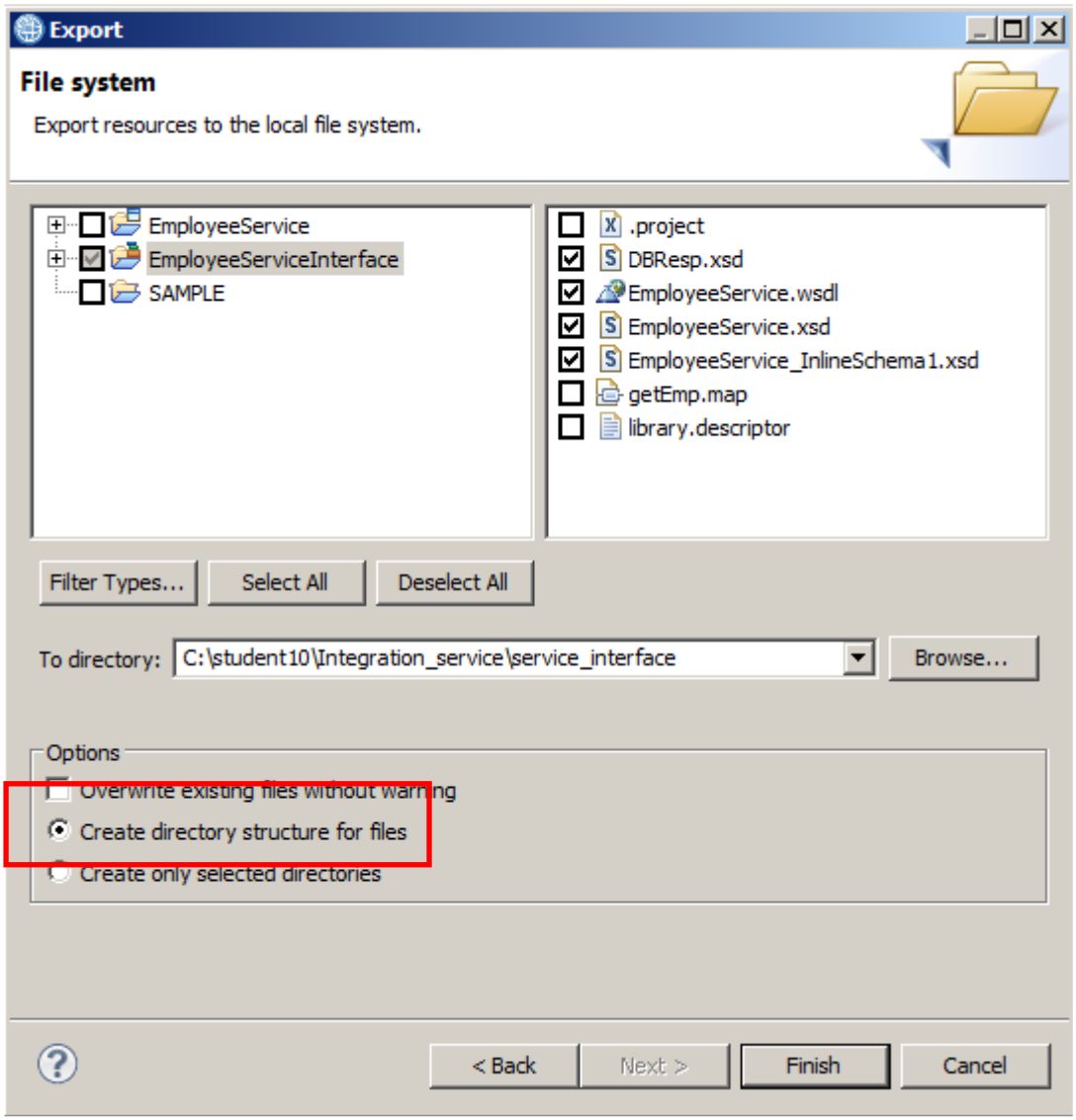
   First, Deselect All.

   Highlight EmployeeServiceInterface, and select (tick) the following files:
   - DBResp.xsd
   - EmployeeService.wsdl
   - EmployeeService.xsd
   - EmployeeService_InlineSchema1.xsd

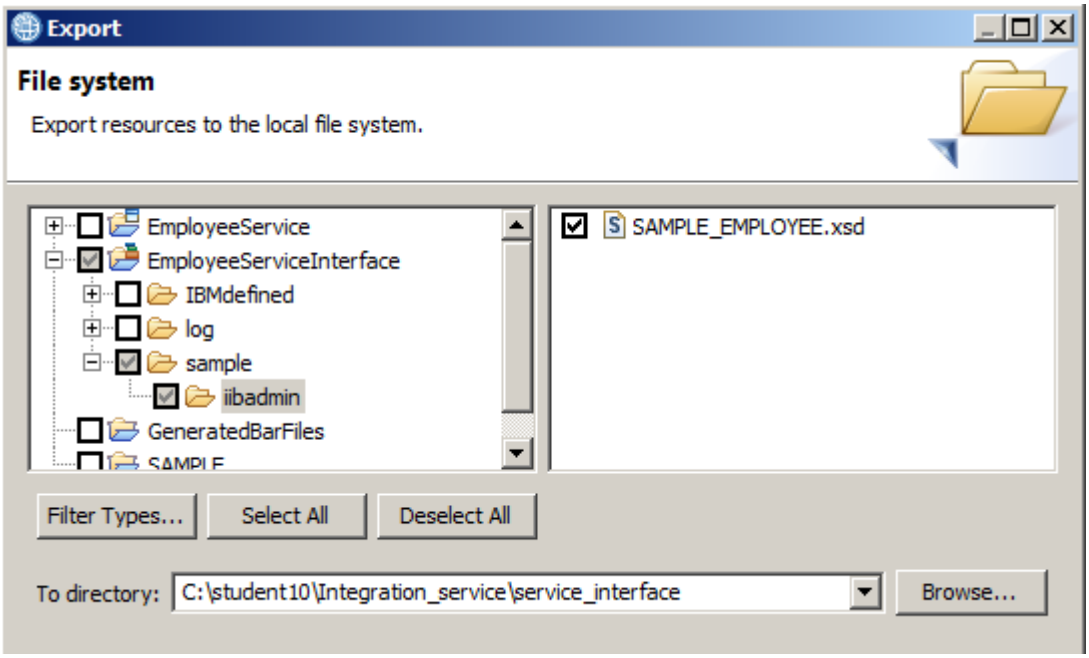
   Set the "To directory" to c:\student10\integration_service\service_interface.

   Choose "Create directory structure for files".

   DO NOT click Finish.



4. Expand EmployeeServiceInterface, sample, iibadmin, and select SAMPLE_EMPLOYEE.xsd. Click Finish.



END OF LAB GUIDE