IBM

BetaWorks

IBM Integration Bus

Pattern Authoring Lab 6

Groups and Table Parameters

June, 2013

Hands-on lab built at product
code level Version 9.0

1. Introduction

In this lab you will use 3 new Pattern Parameters features introduced in WebSphere Message Broker version 8.

This lab has been tested using IBM Integration Bus V9.0:

1. Parameter Group Hiding
2. Project Name Parameter
3. Table Parameter

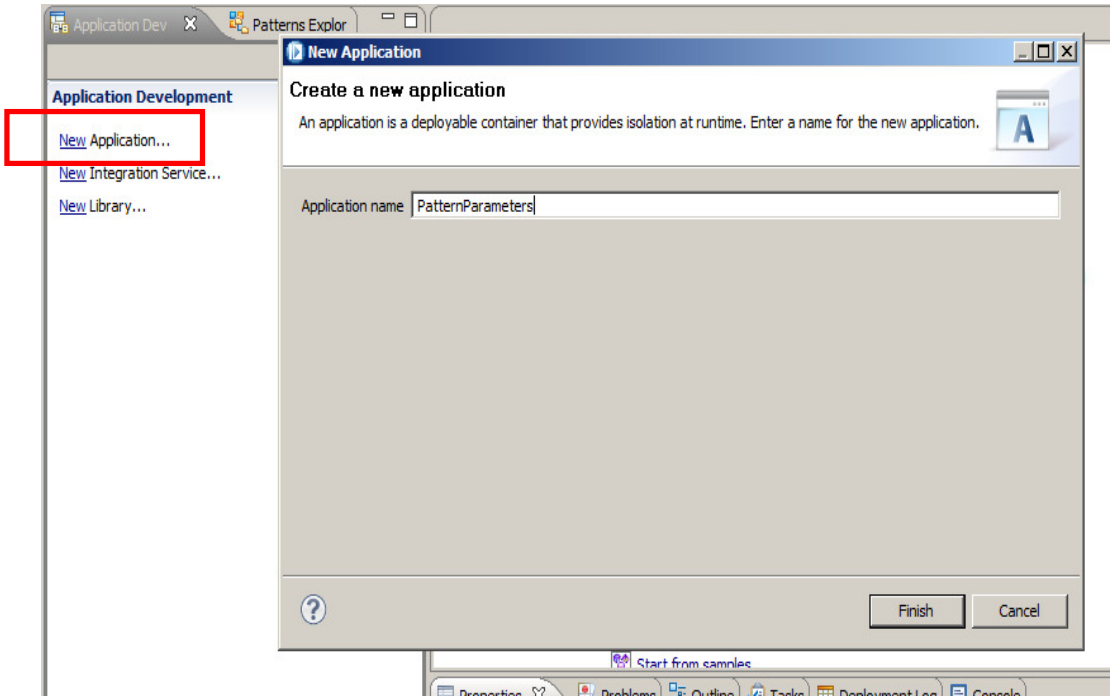
The objective of the lab is to create a Routing Pattern that will allow you to generate a dynamic routing message flow. The pattern will take the number of routing destinations (MQOutput nodes) and the routing table as parameters. For each destination, the Pattern Configuration Editor will show one parameter group with its corresponding parameters.

You will also define a parameter to be used as the Project Name for the generated Pattern instance.

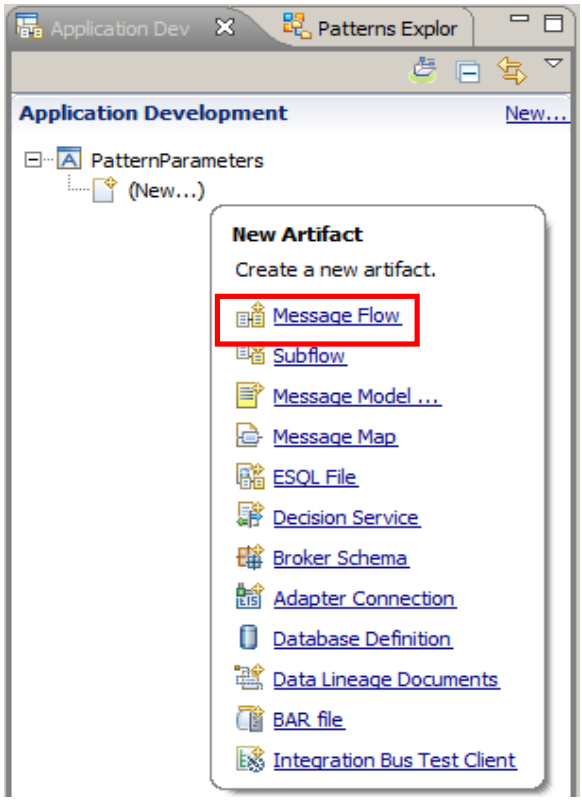
2. Creating the Pattern

1. Create a new Application by clicking on "New Application". (The example below shows an empty workspace, but may be different if you already existing items in the workspace).

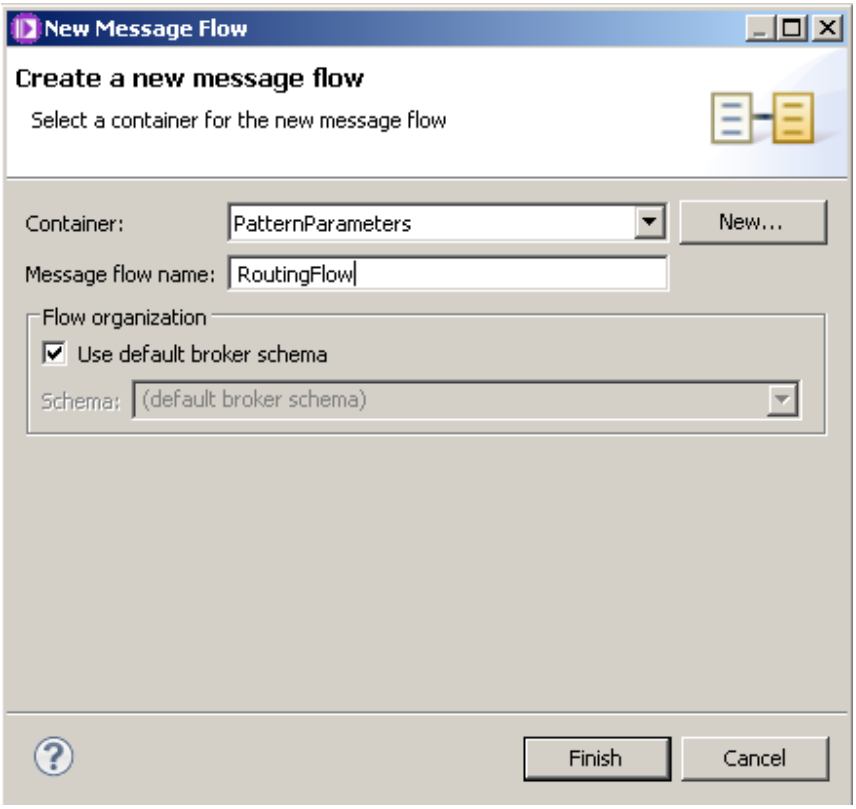
   Enter "PatternParameters" as the application name (this exact name is required, since it will match some java code later in the lab). Click Finish.



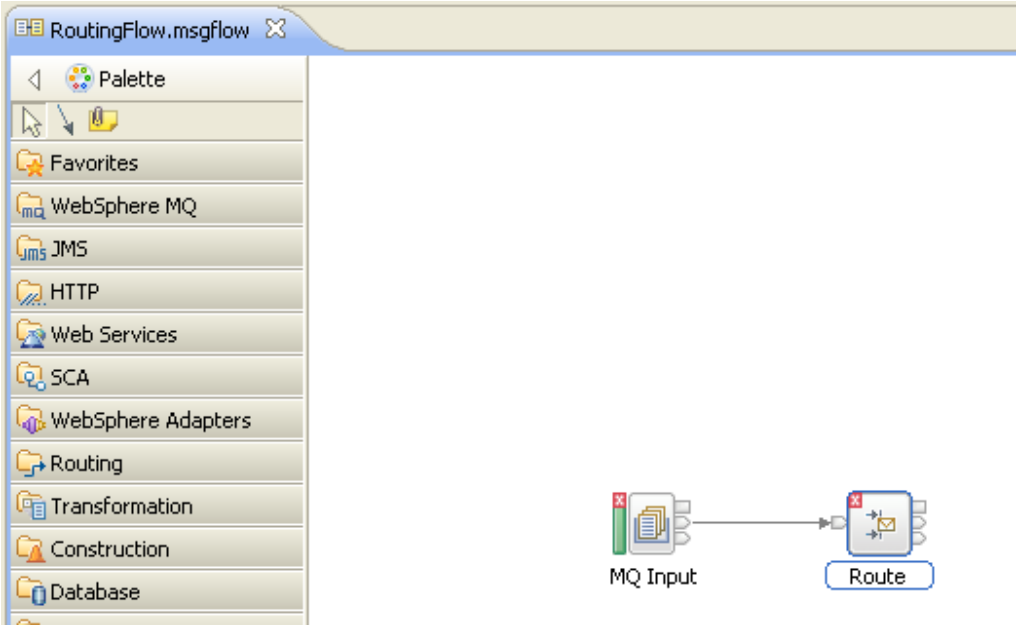
2. Create a new Message Flow by clicking on New -> Message Flow under the PatternParameters application.



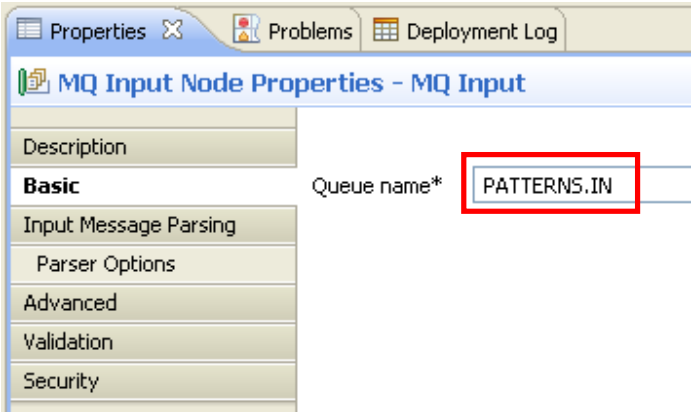
3. Enter "RoutingFlow" as the message flow name. Click Finish.



4. Drop an MQInput node and a Route node onto the canvas. Connect them as shown:



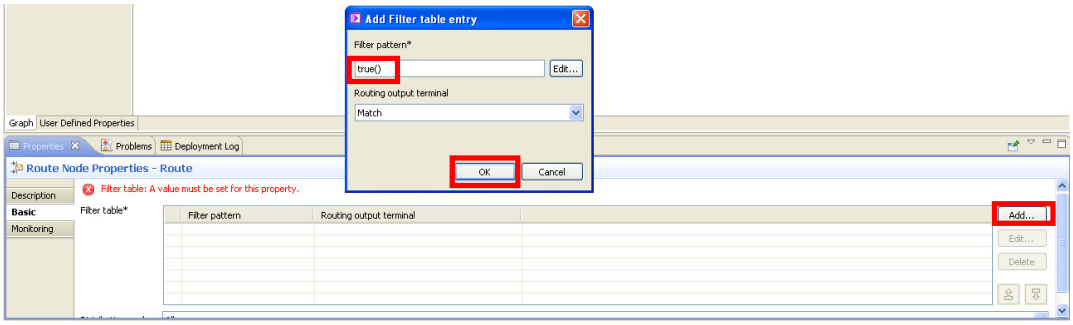
5. Set the MQInput node's queue name to "PATTERNS.IN".



6. Click on the Route node, and select the properties tab.

   On the Basic tab, click on the Add button to insert a new Filter Pattern.

   Enter "true()" as the Filter Pattern and click OK. Leave the Routing output terminal to "Match".



   Note that the only purpose of this step is to avoid the error that would appear if the Filter table was left empty.

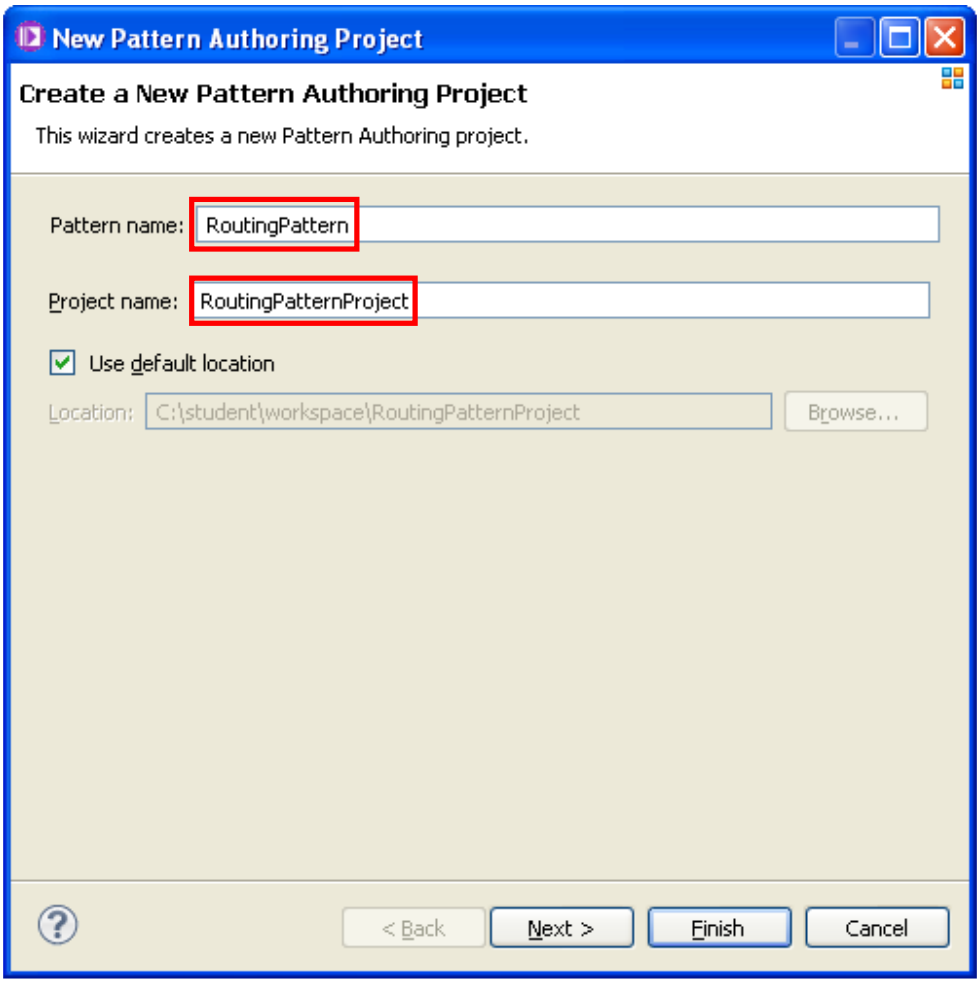
7. Press Ctrl+S or File->Save to save the message flow.

   Check that the Problems view has no errors (apart from the "no outbound connections" warning).

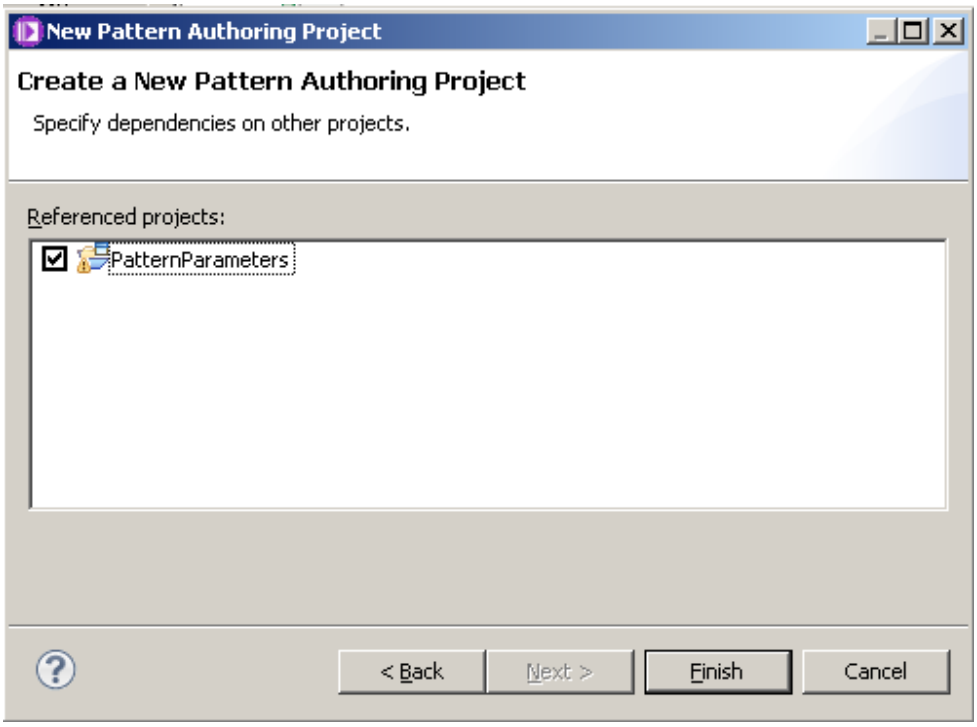
8. Create a new Pattern Authoring project by clicking File->New->Pattern Authoring Project.

   Enter "RoutingPattern" as the Pattern name and "RoutingPatternProject" as the Project name.

   Click Next.

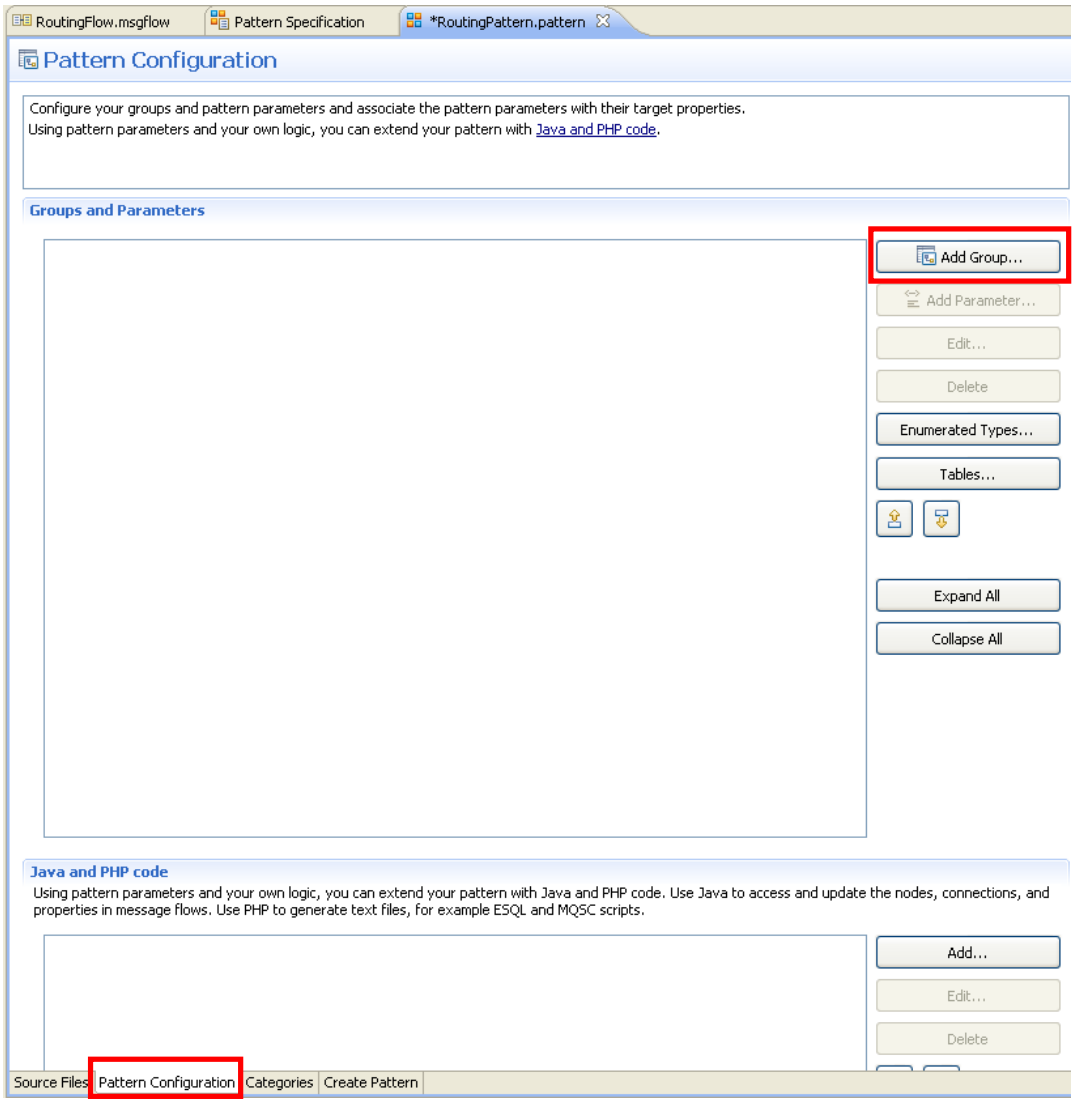


9. Select the "PatternParameters" project, and click Finish.



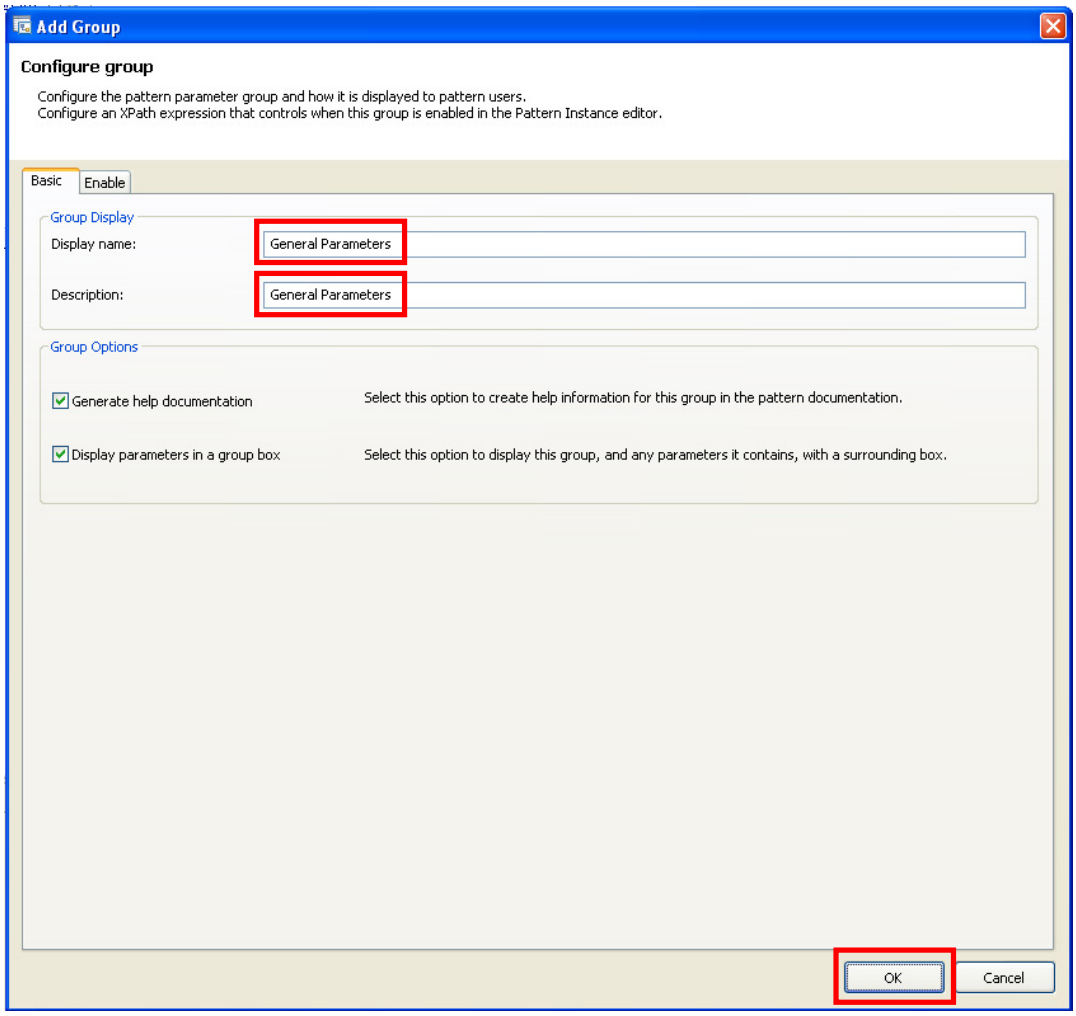
10. When the Pattern Authoring Editor opens, click on the "Pattern Configuration" tab.

    Click on the "Add Group" button.



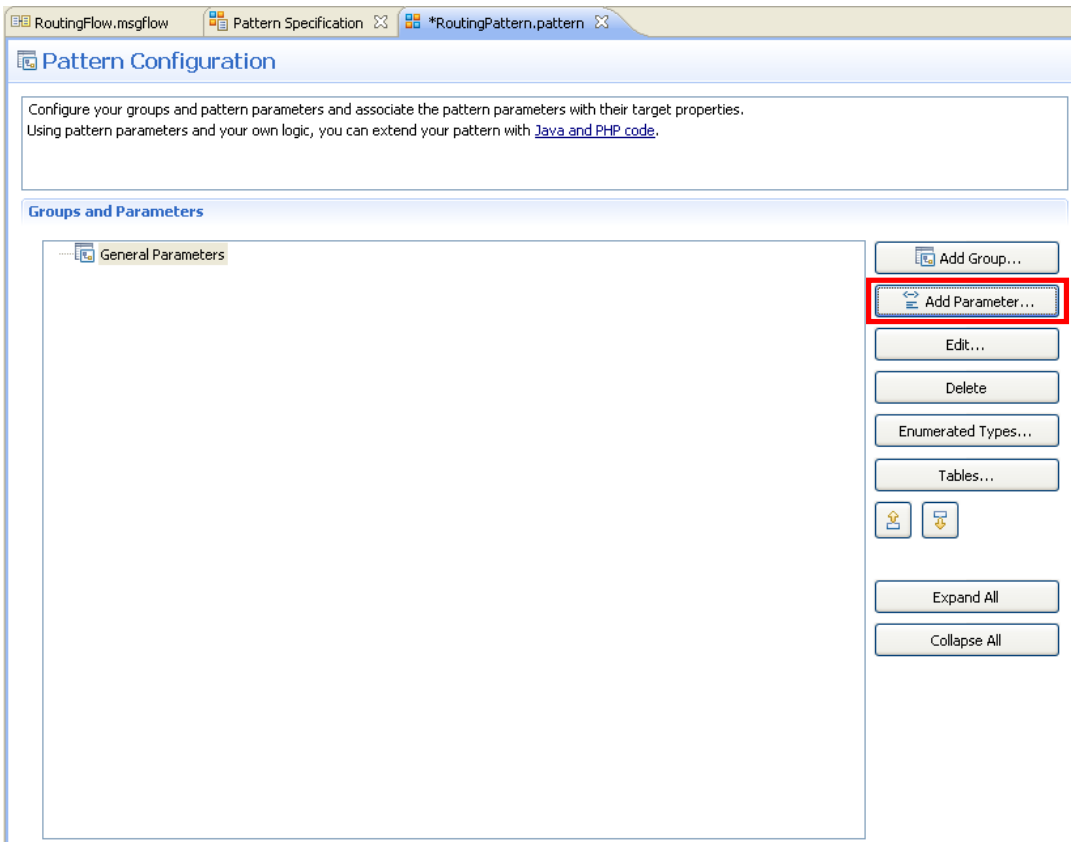
11. Enter "General Parameters" as both the Group Display Name and Description.

    Click OK.

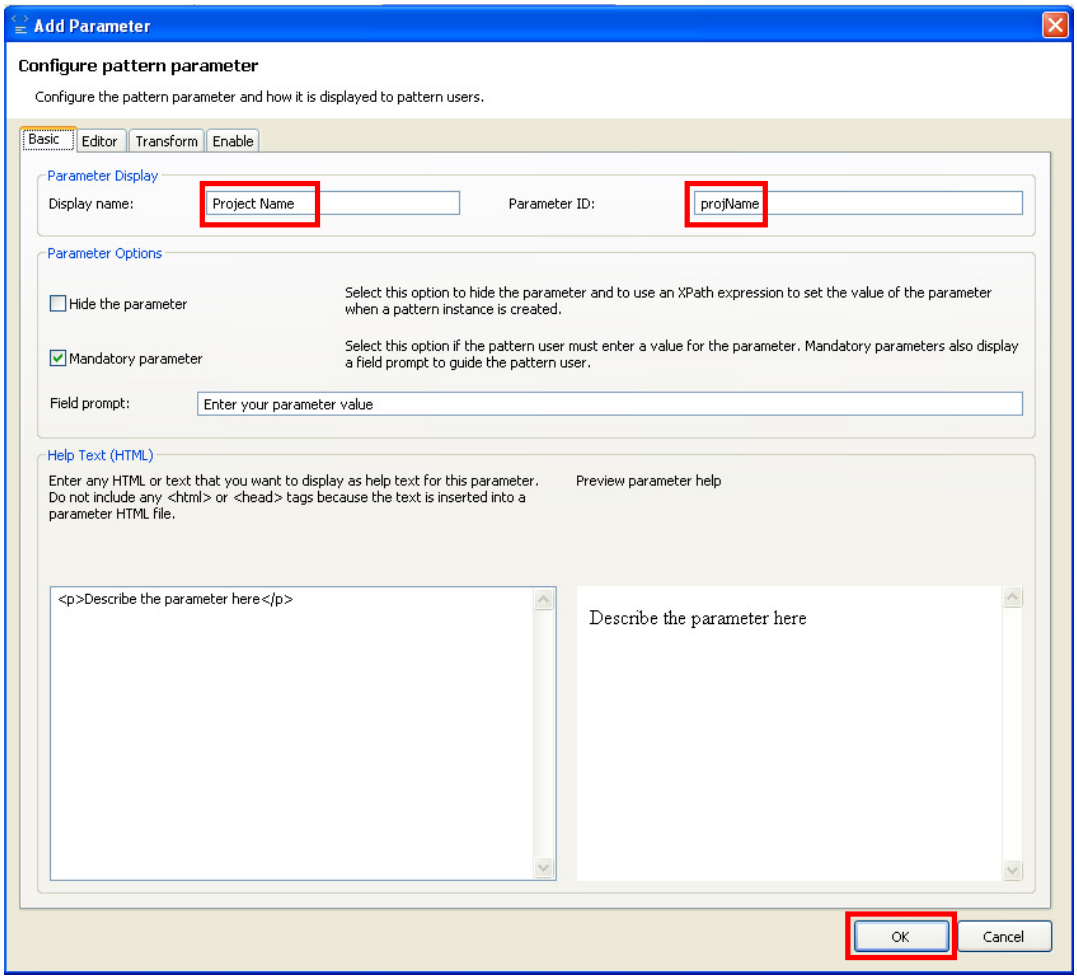


12. Now add three parameters to the "General Parameters" group: Project Name, OutputsAmount and RoutingTable.

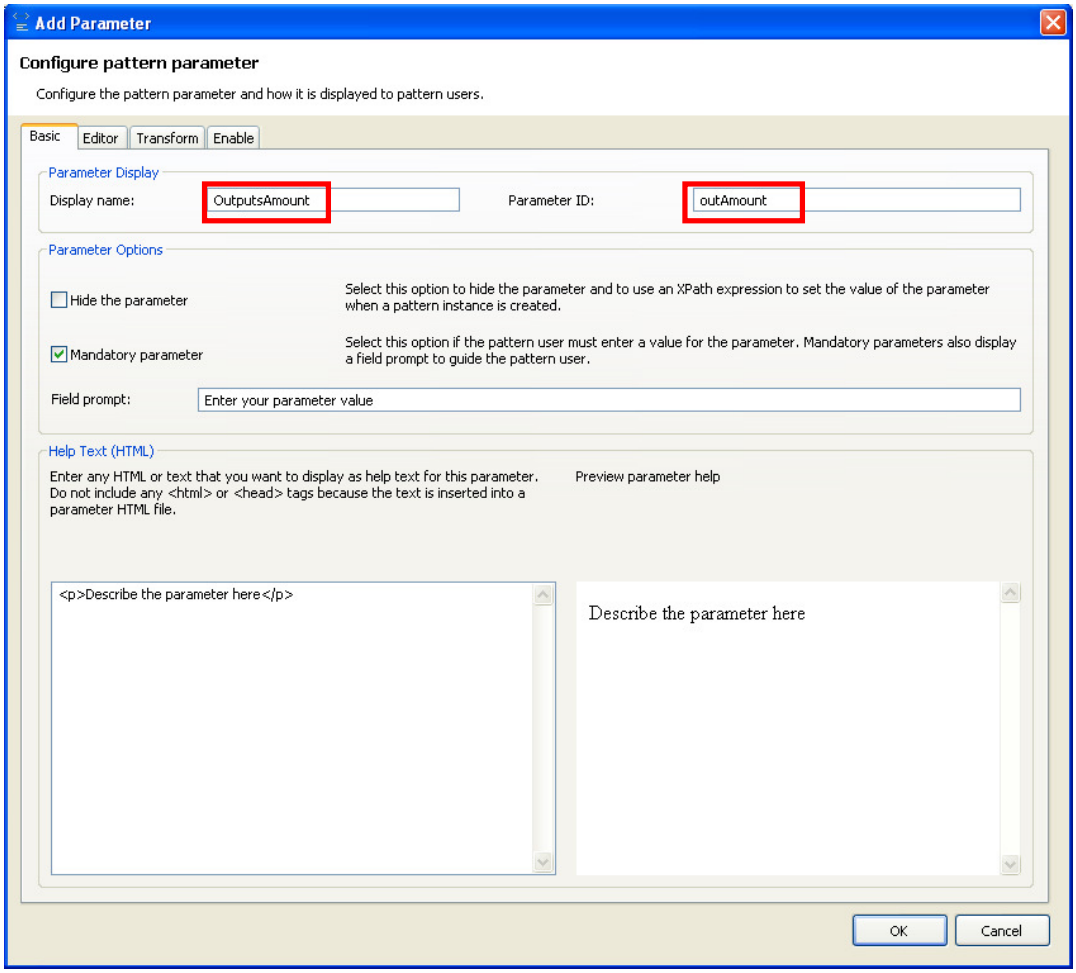
    Click on the "General Parameters" group and click on the "Add Parameter" button.



13. Enter "Project Name" as the Display Name, and "projName" as the Parameter ID. Click OK.



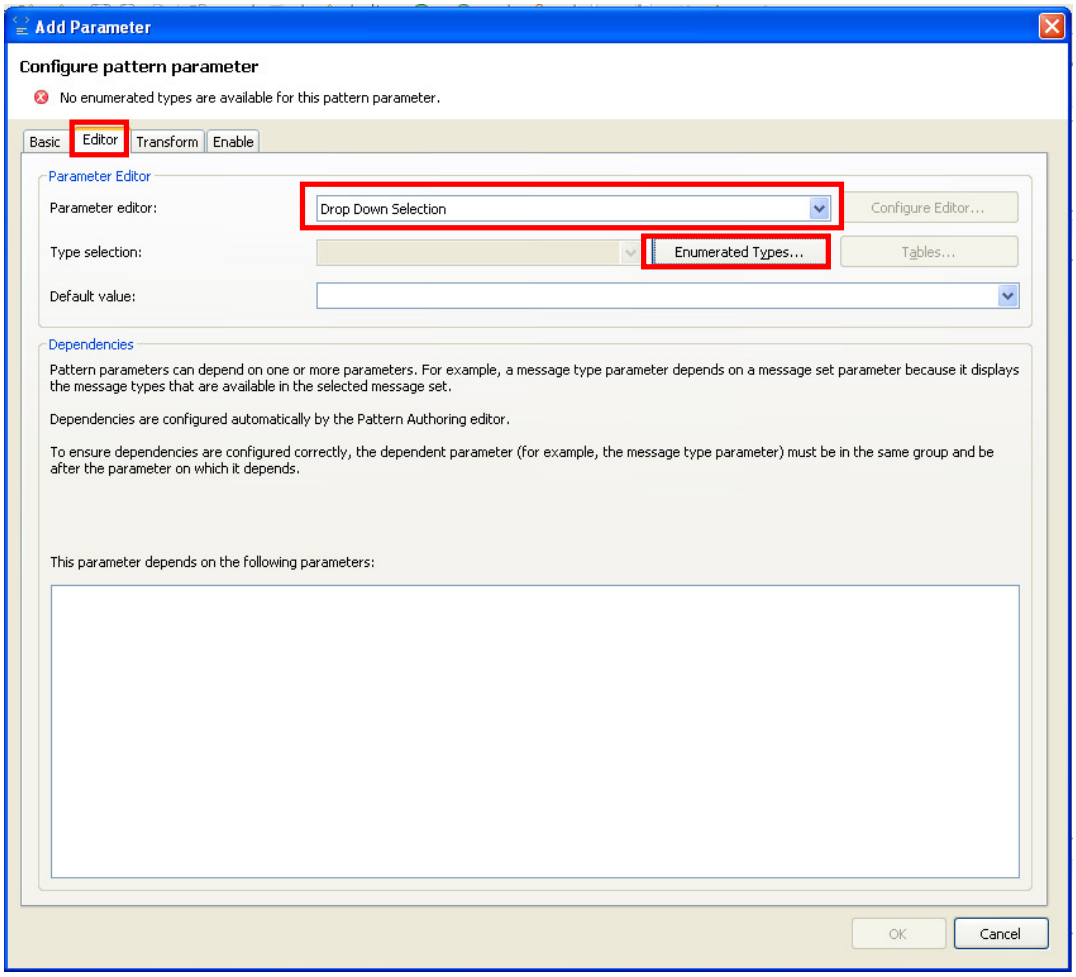
14. Click on the "Add Parameter" button again, and enter "OutputsAmount" as the Display Name, and "outAmount" as the Parameter ID.



15. Then click on the Editor tab.

    Select "Drop Down Selection" as the Parameter Editor.

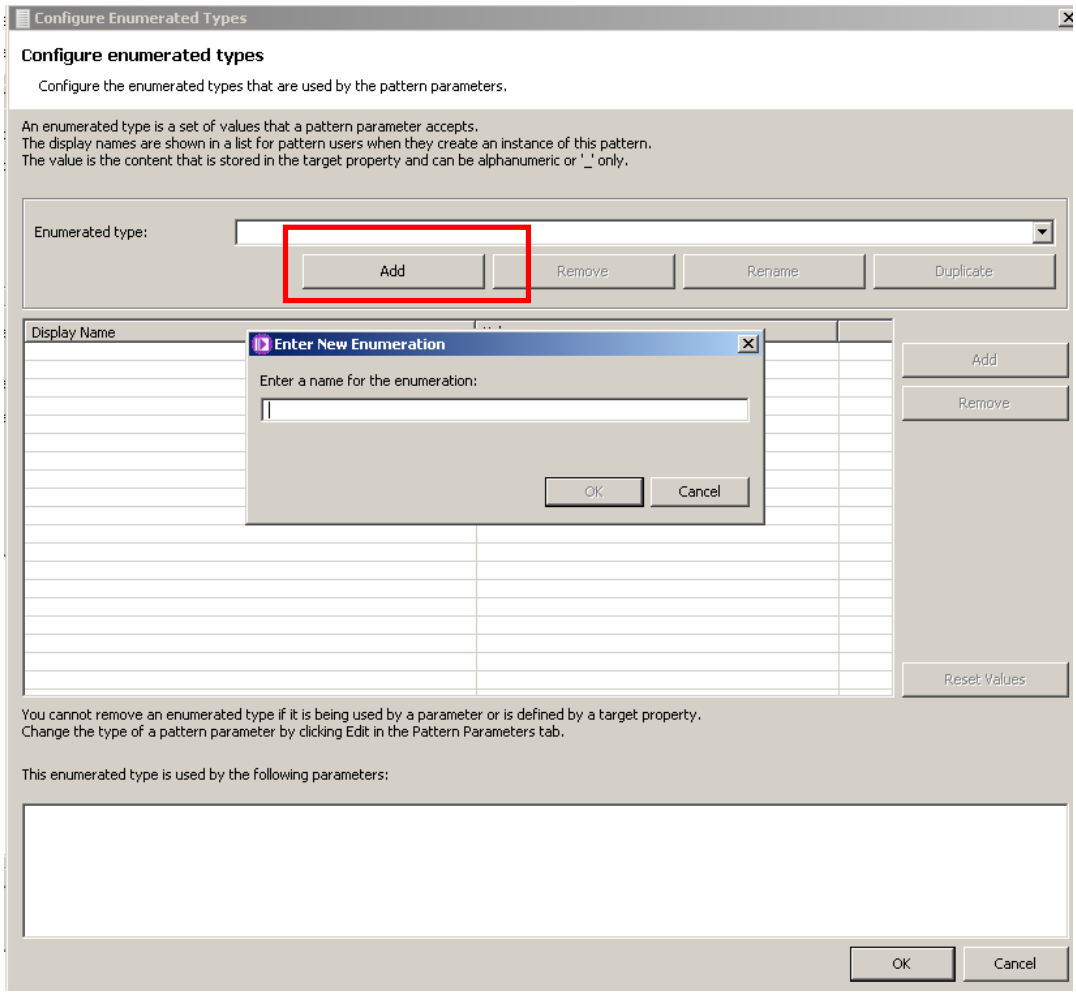
    Click on the Enumerated Types button to define the possible values of the parameter.



16. In the "Configure Enumerated Types" window, click on the Add button to create a new enumerated type.

    Enter "outputsAmount" as the Enumeration name.

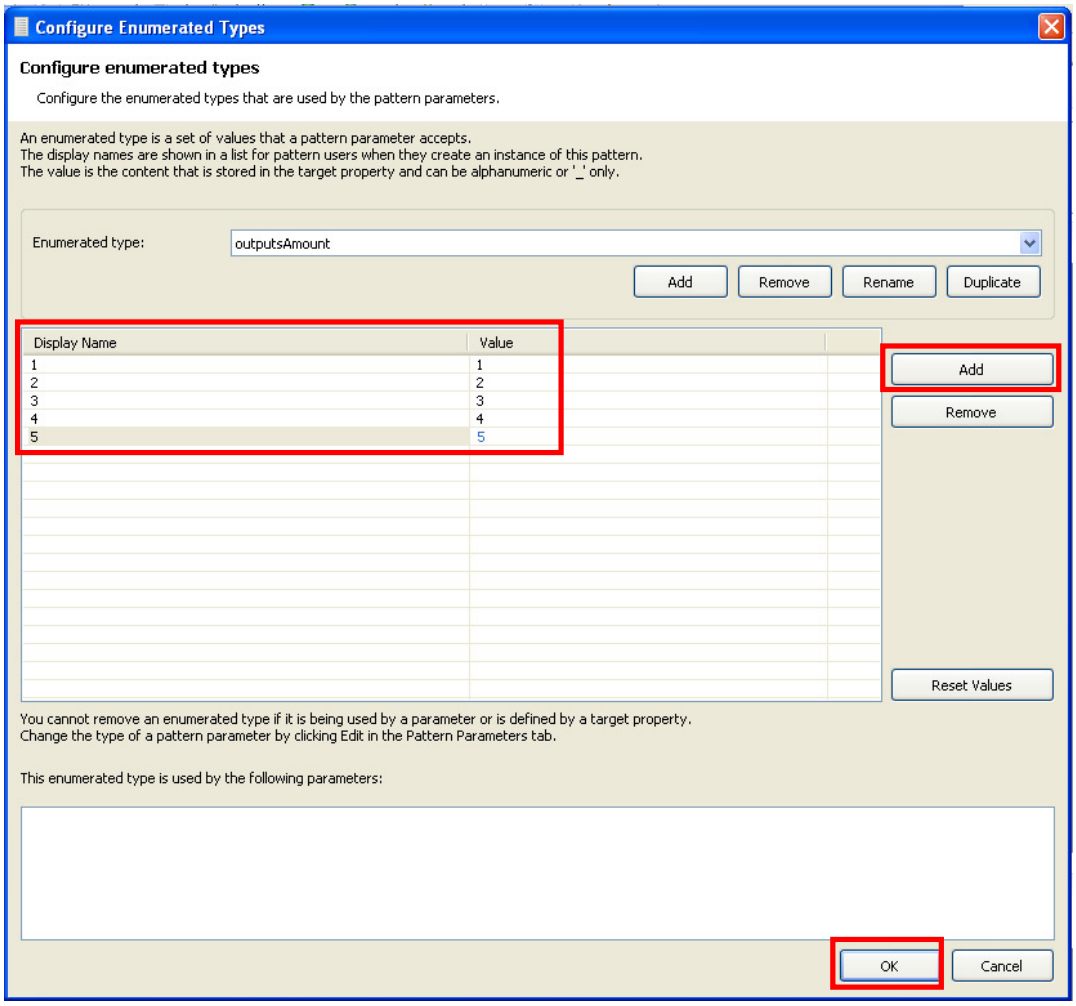
    Click OK.



17. Enter the following values in the table using the Add button for each row.

    To edit the values, click on the row, delete the existing value (e.g. "Display Name1"), overtype with the new value, and press Enter to make the change.

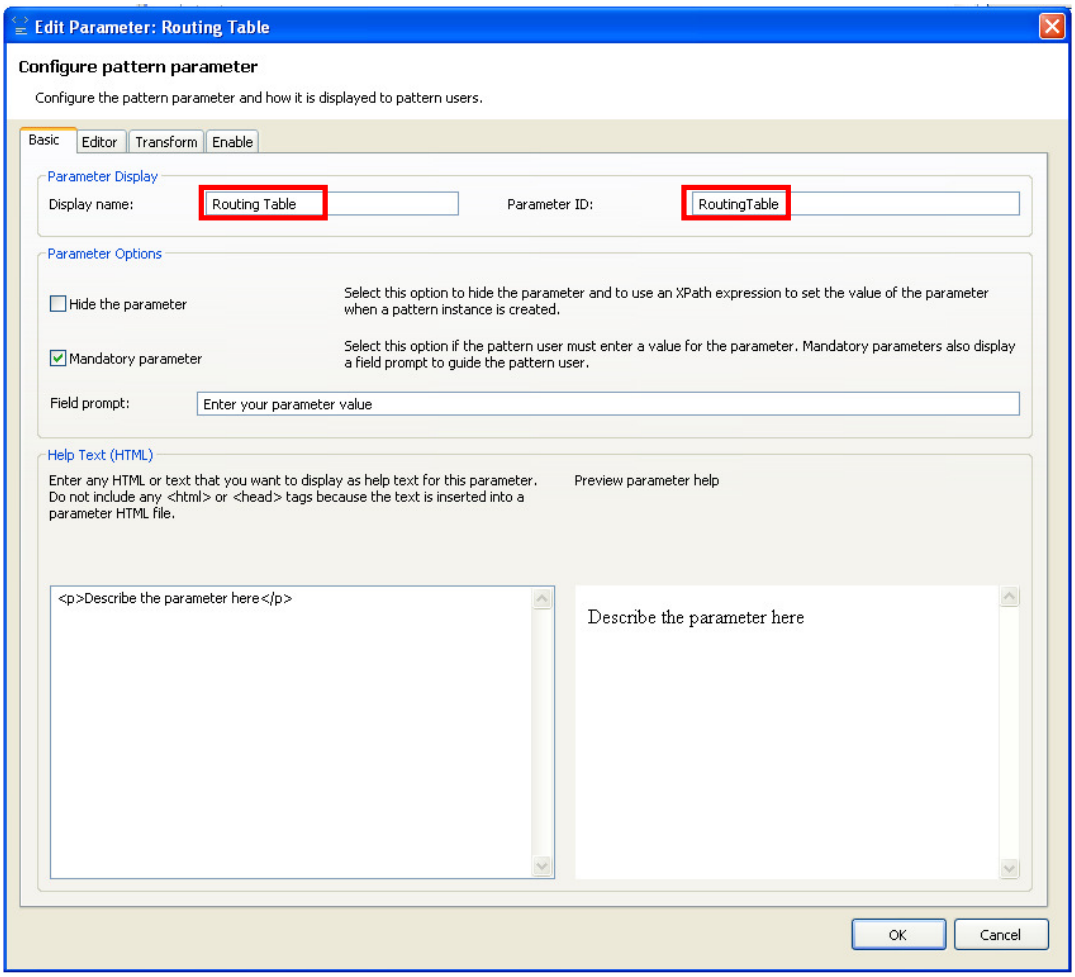
    Click OK.



18. Back in the "Add Parameter" window, click OK.

19. Now you will add the Routing Table parameter. To do this you will use a new type of parameter, which allows you to enter tables as parameters of the pattern.

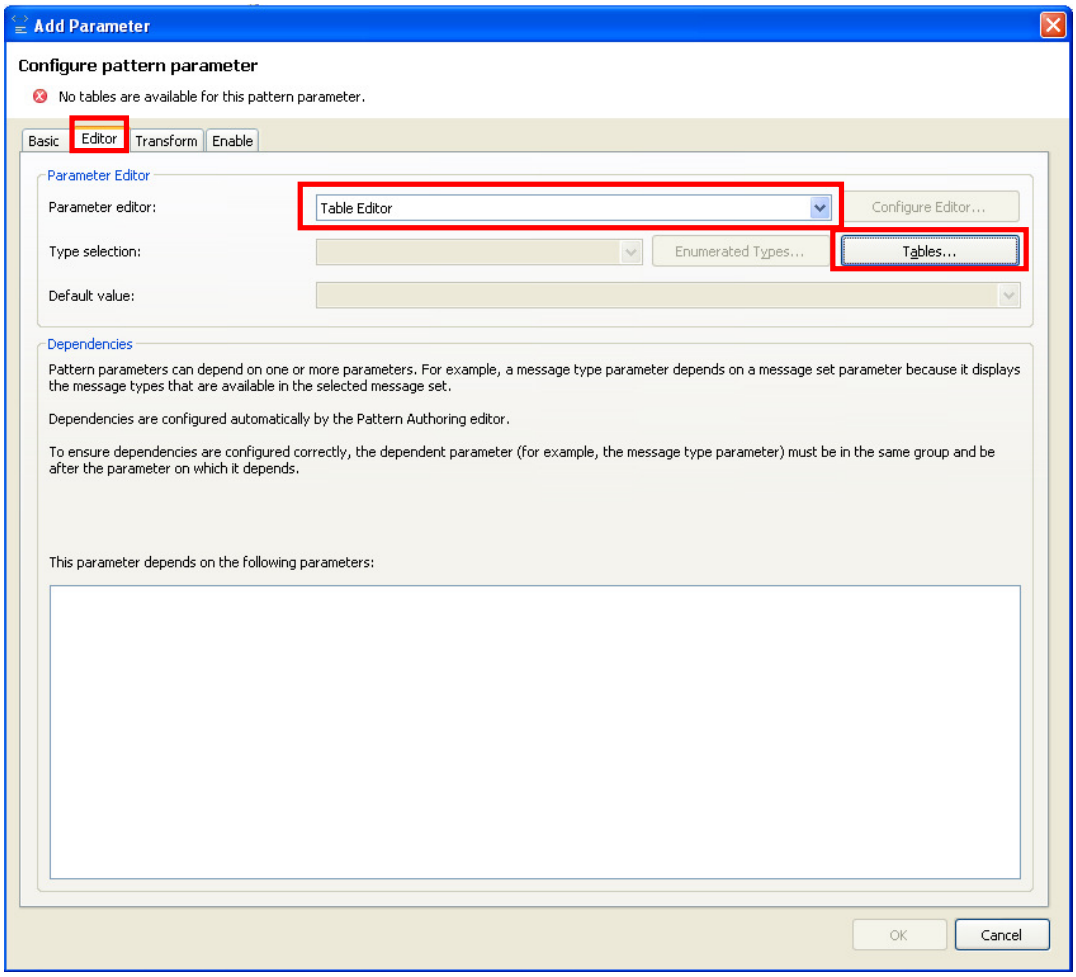
    Click on the Add Parameter button again, and enter "Routing Table" as the Display name, and "RoutingTable" (no spaces) as the Parameter ID.



20. Then click on the Editor tab to select the appropriate Editor for this parameter.

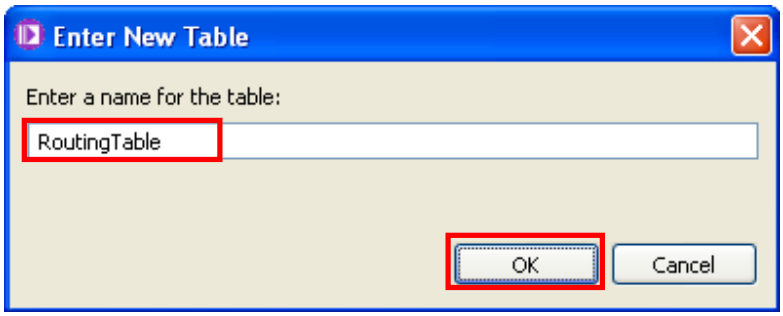
    Select "Table Editor" as the Parameter editor.

    Then click on the Tables button to create the required table.



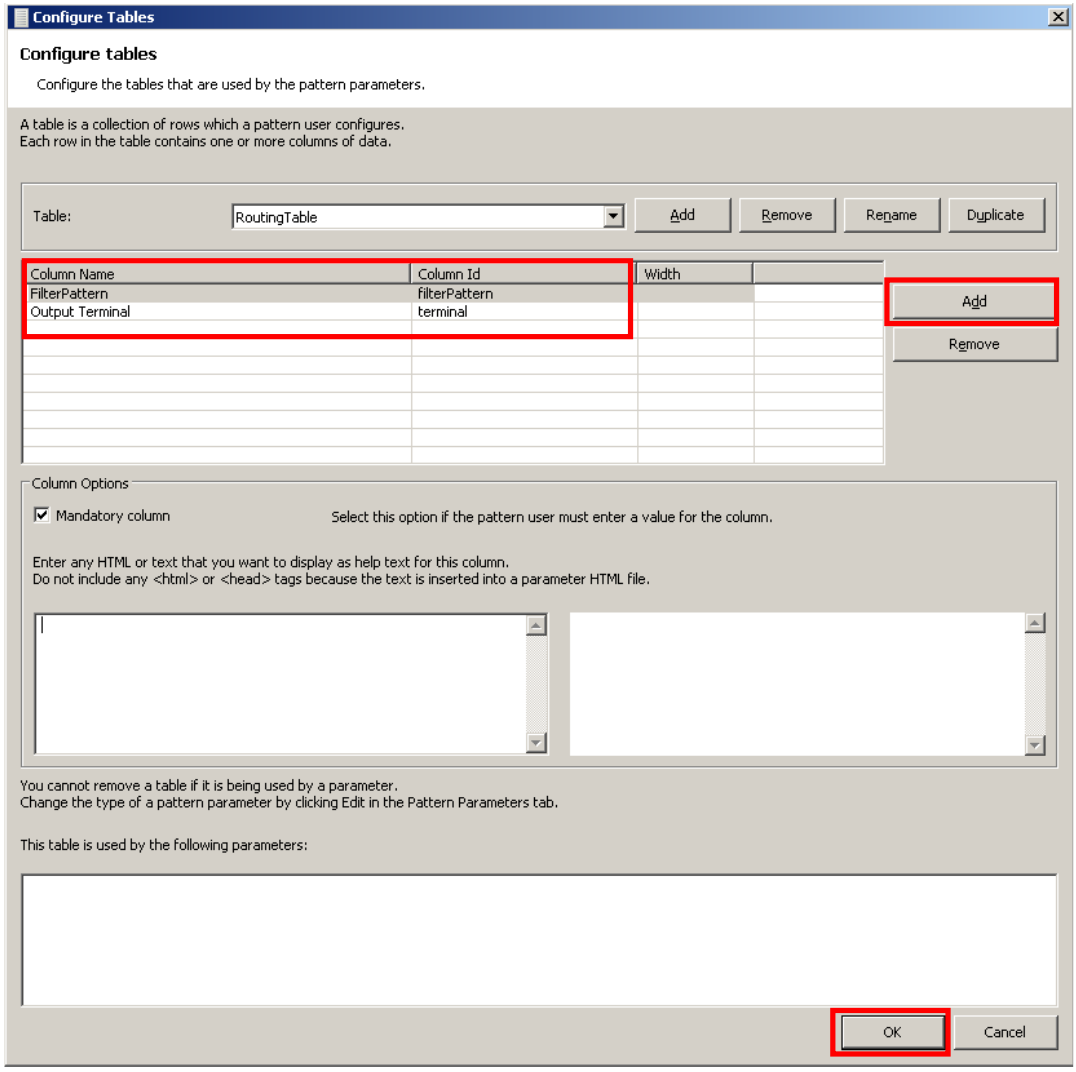
21. Click on the Add button and enter "RoutingTable" as the table name.

    Click OK.



22. Add two columns (two rows in the table) using the Add button, to reflect the Route node Filter table, like shown, with the following values:

Column Name	Column ID
FilterPattern	filterPattern
Output Terminal	terminal

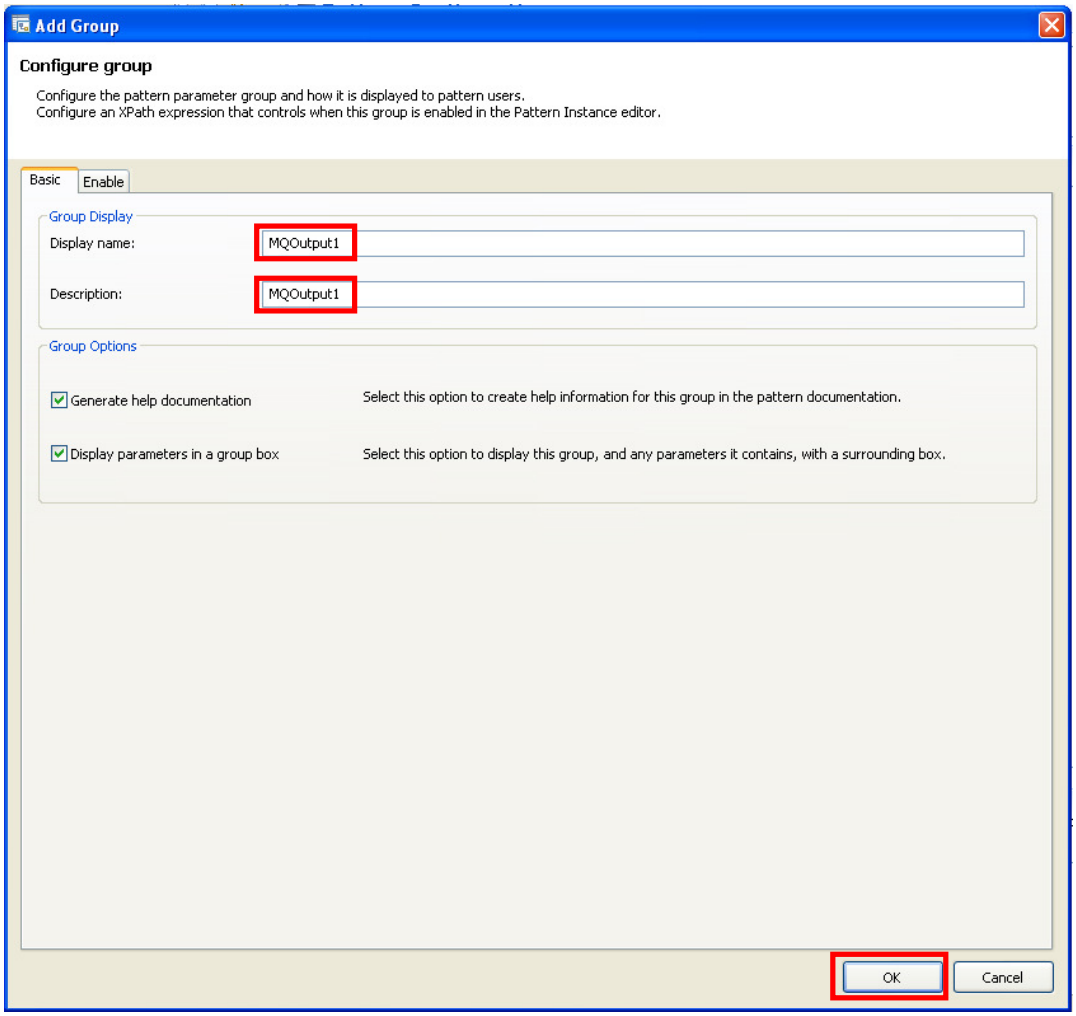


Then click OK.

23. In the Add Parameter window, click OK.

24. Then you are going to add 5 groups, one per each possible MQOutput node the user can generate with the pattern.

    Click on the Add Group button, enter "MQOutput1" as the Display name and Description.



    Click OK.

25. Select the MQOutput1 group and click on the Add Parameter button.

    Enter "Queue Name" as the Display name and "qn1" as the Parameter ID.

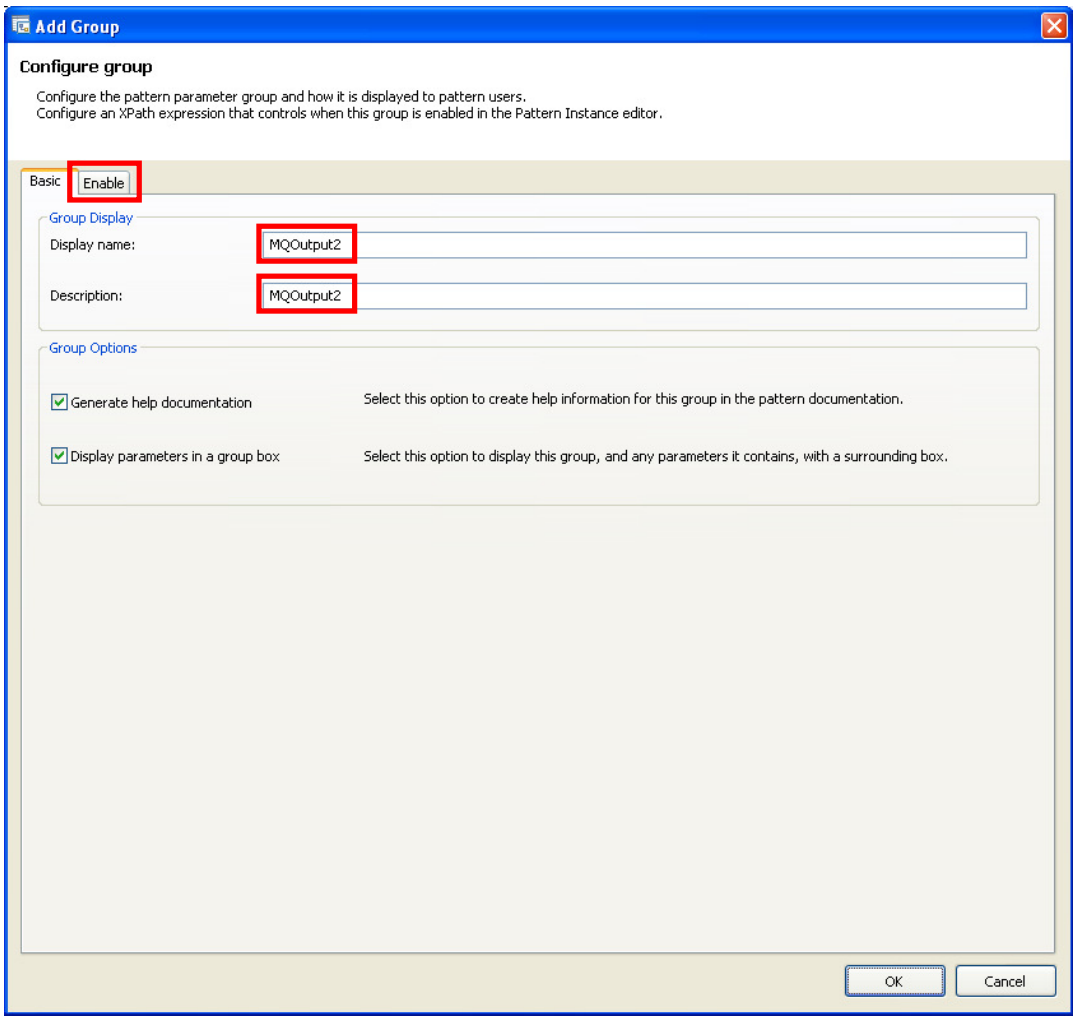
    Click Ok.

26. Repeat the last step to create a parameter with "Queue Manager" as the Display name and "qm1" as the Parameter ID.

27. Save your progress pressing Ctrl+S or File->Save.

28. Create a new group by clicking on the Add Group button.

    Enter "MQOutput2" as the Display Name and Description.



This time, click on the "Enable" tab.

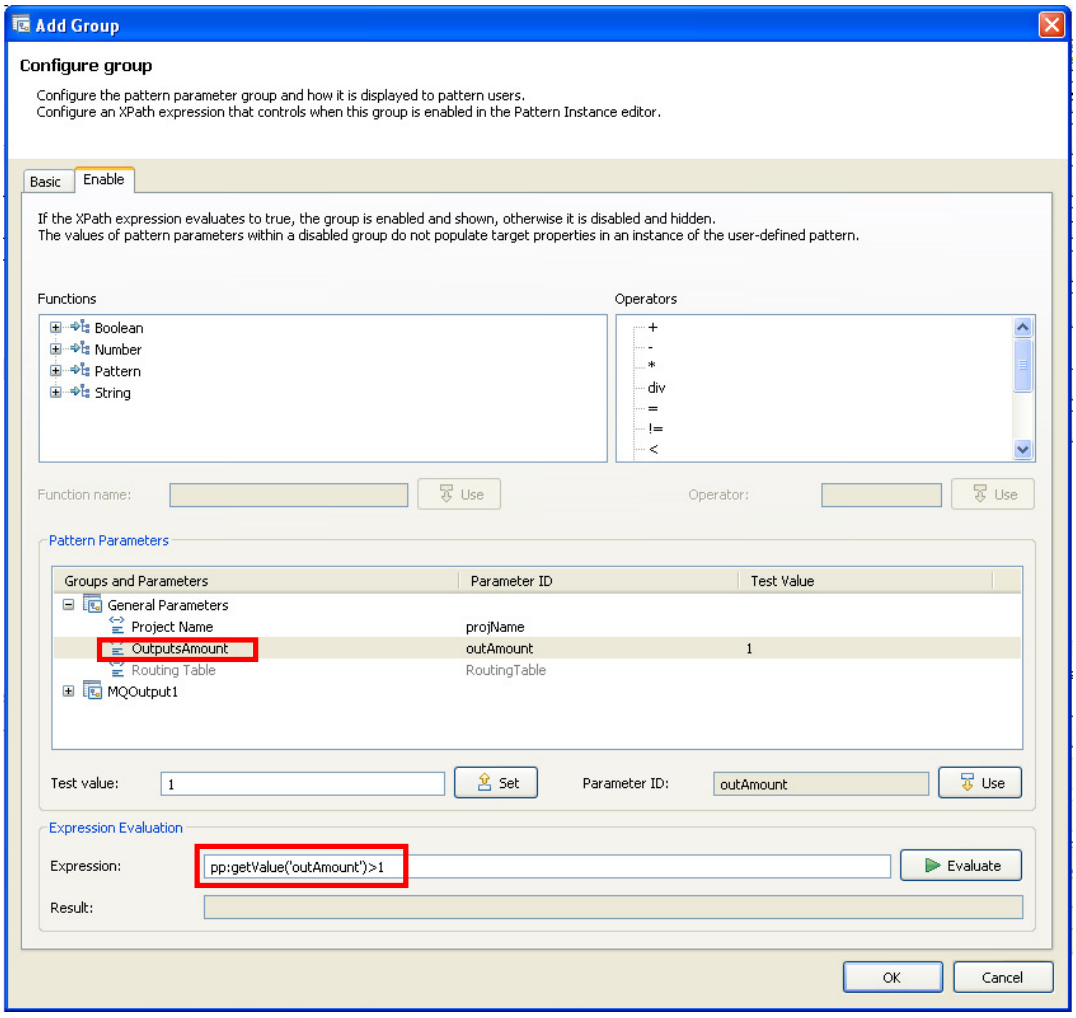
29. Now you are going to define when should this group be shown.

    The objective is to show the MQOutput2 group only if the OutputsAmount was set to a value greater than 1.

    Double-click on the "OutputsAmount" parameter (under General Parameters group)

    This will insert the "pp:getValue("outAmount") sentence in the Expression field.

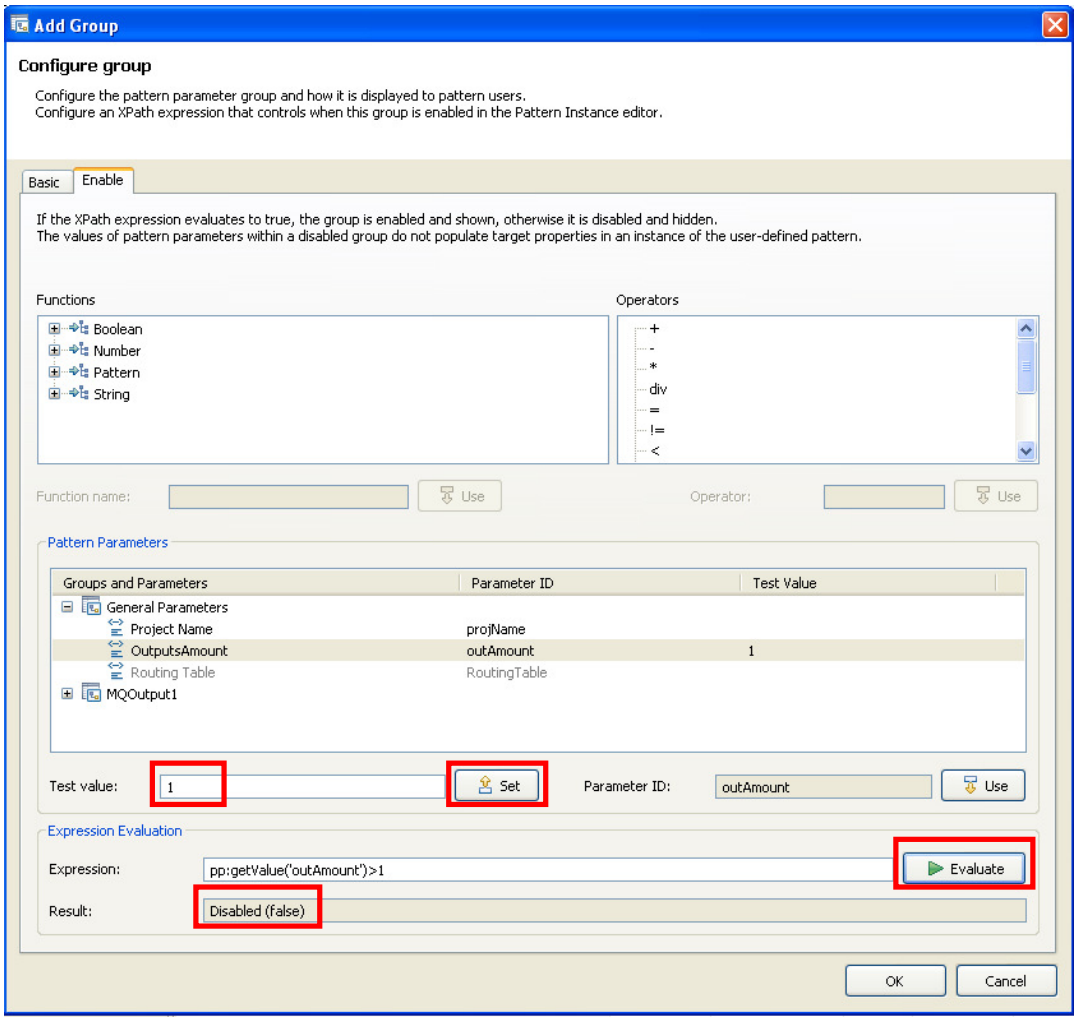
    Append ">1" to the expression.



Note that "outAmount" was used instead of "OutputsAmount" because the first is the Parameter ID while the last one is the Parameter display name.

30. You can test your expression by using the Evaluate button.

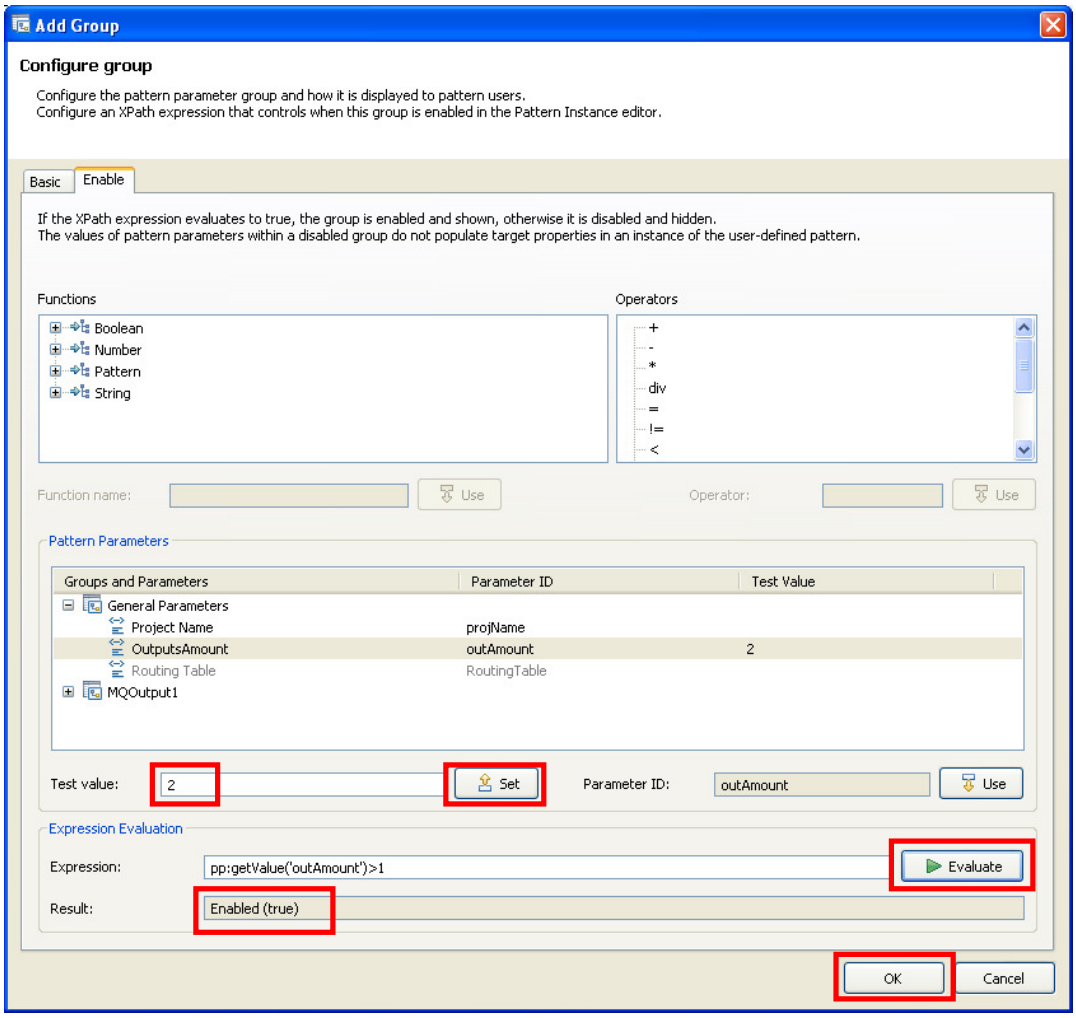
Enter "1" as the Test Value and click Set. Then click the Evaluate button.



You can see the result of the evaluation in the Result field.

In this case, the Result is Disabled, because OutputsAmount value was set to 1.

31. Now set the Test Value to "2", click the Set button. Click on the Evaluate button.



Check that the Result now is Enabled, because OutputsAmount value was greater than 1.

Click OK.

32. Add two parameters to the MQOutput2 group (Queue Name and Queue Manager) the same way you did before, with these values:

Display Name	Parameter ID
Queue Name	qn2
Queue Manager	qm2

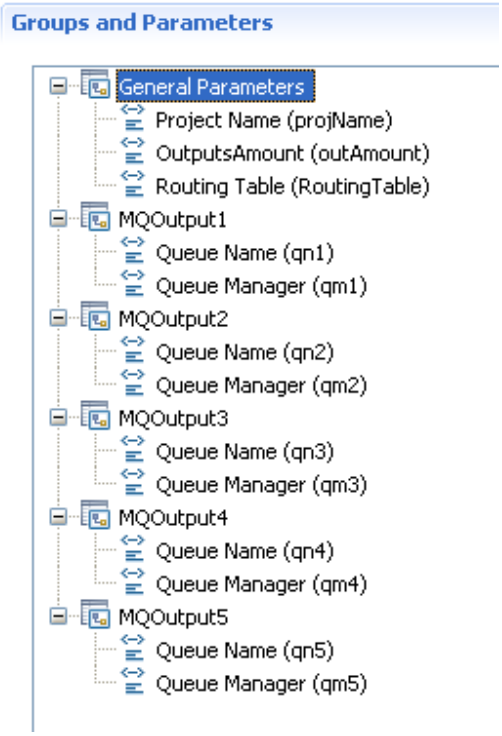
33. Repeat 3 times the steps 29-30 with the following values:

Group Display name	Enable Expression
MQOutput3	pp:getValue('outAmount')>2
MQOutput4	pp:getValue('outAmount')>3
MQOutput5	pp:getValue('outAmount')>4

Then add two parameters to each group with the following values:

Group	Display Name	Parameter ID
MQOutput3	Queue Name	qn3
	Queue Manager	qm3
MQOutput4	Queue Name	qn4
	Queue Manager	qm4
MQOutput5	Queue Name	qn5
	Queue Manager	qm5

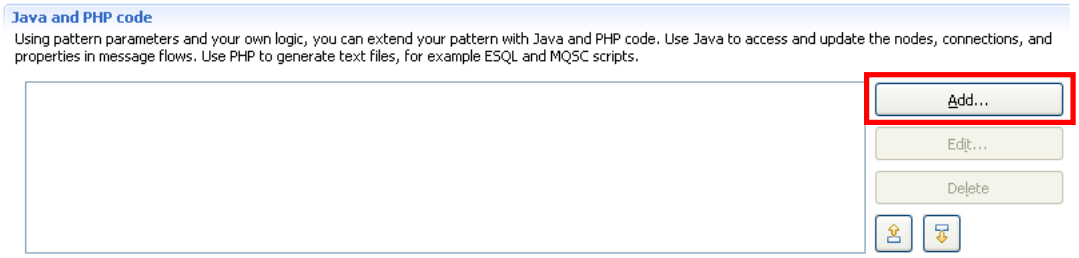
It should look like this:



34. Press Ctrl+S or File->Save to save your Pattern.

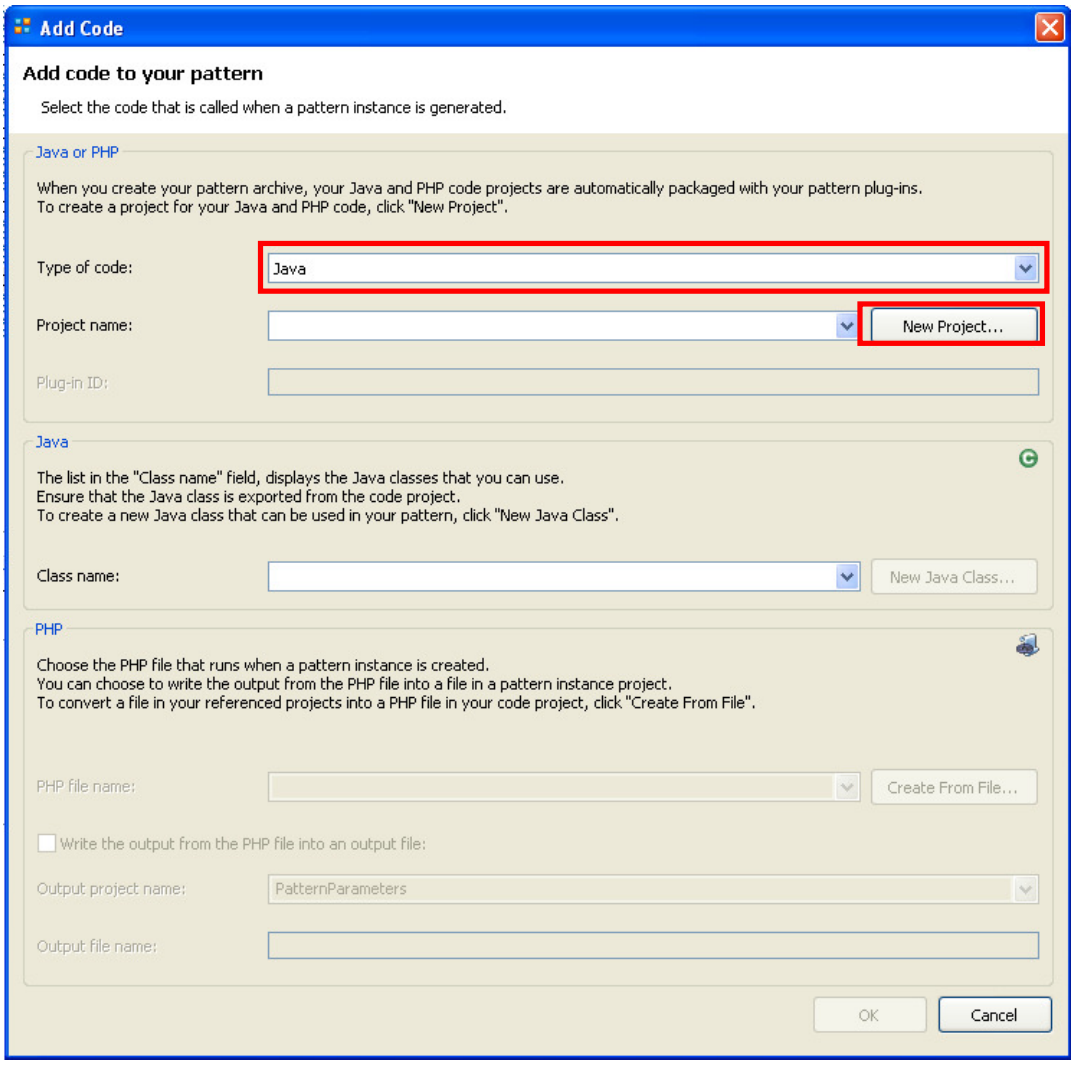
35. Now you are going to create a Java class that will create the MQOutput nodes dynamically (according to the OutputsAmount parameter) and will associate the table parameter with the Router node Filter table.

    In the "Java and PHP code" section (at the bottom of the Pattern Configuration tab, click on the Add button.



    This will start the "Add Code" wizard.

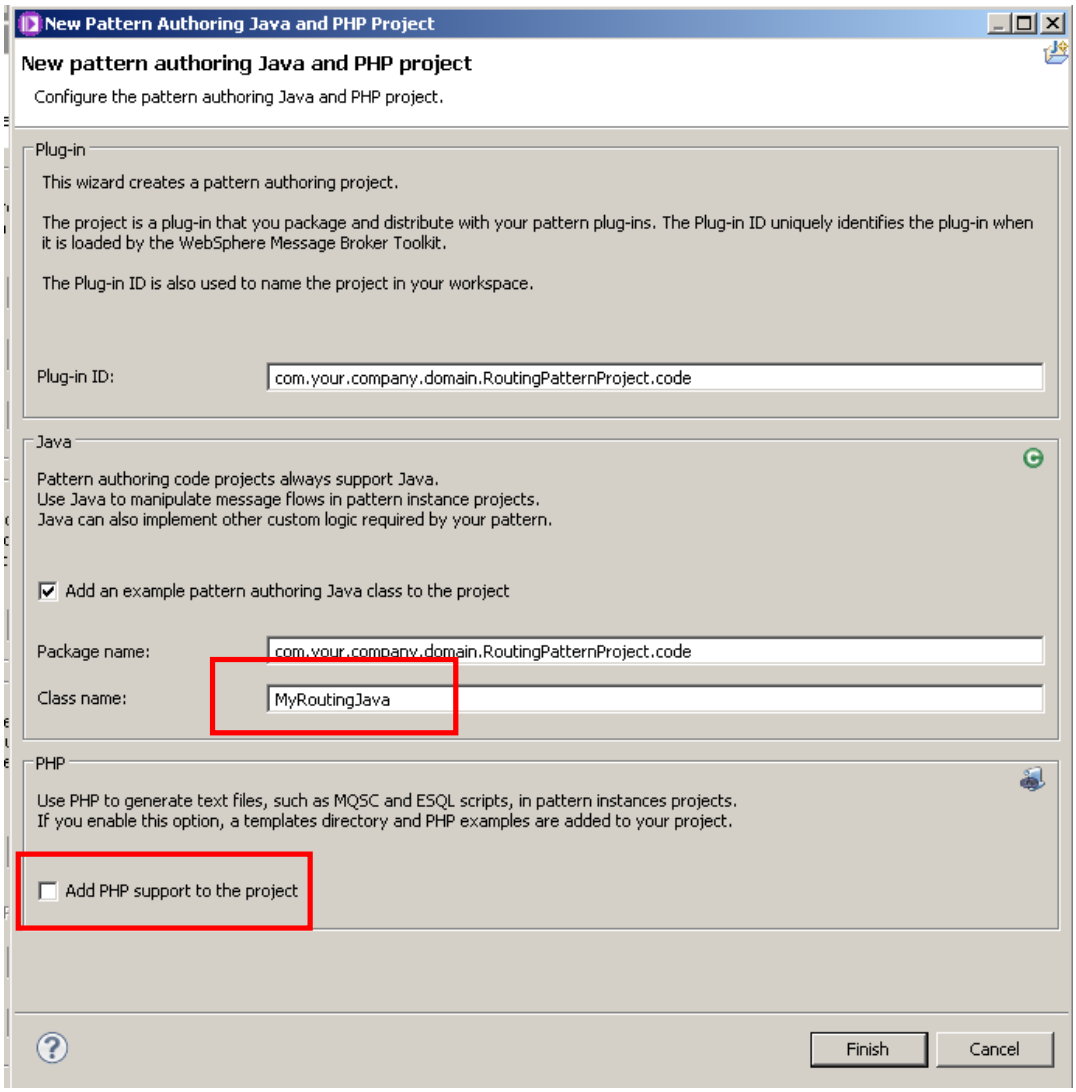
36. Leave the "Type of code" field as "Java", and click on the "New Project" button.



37. Change the java Class name to MyRoutingJava.

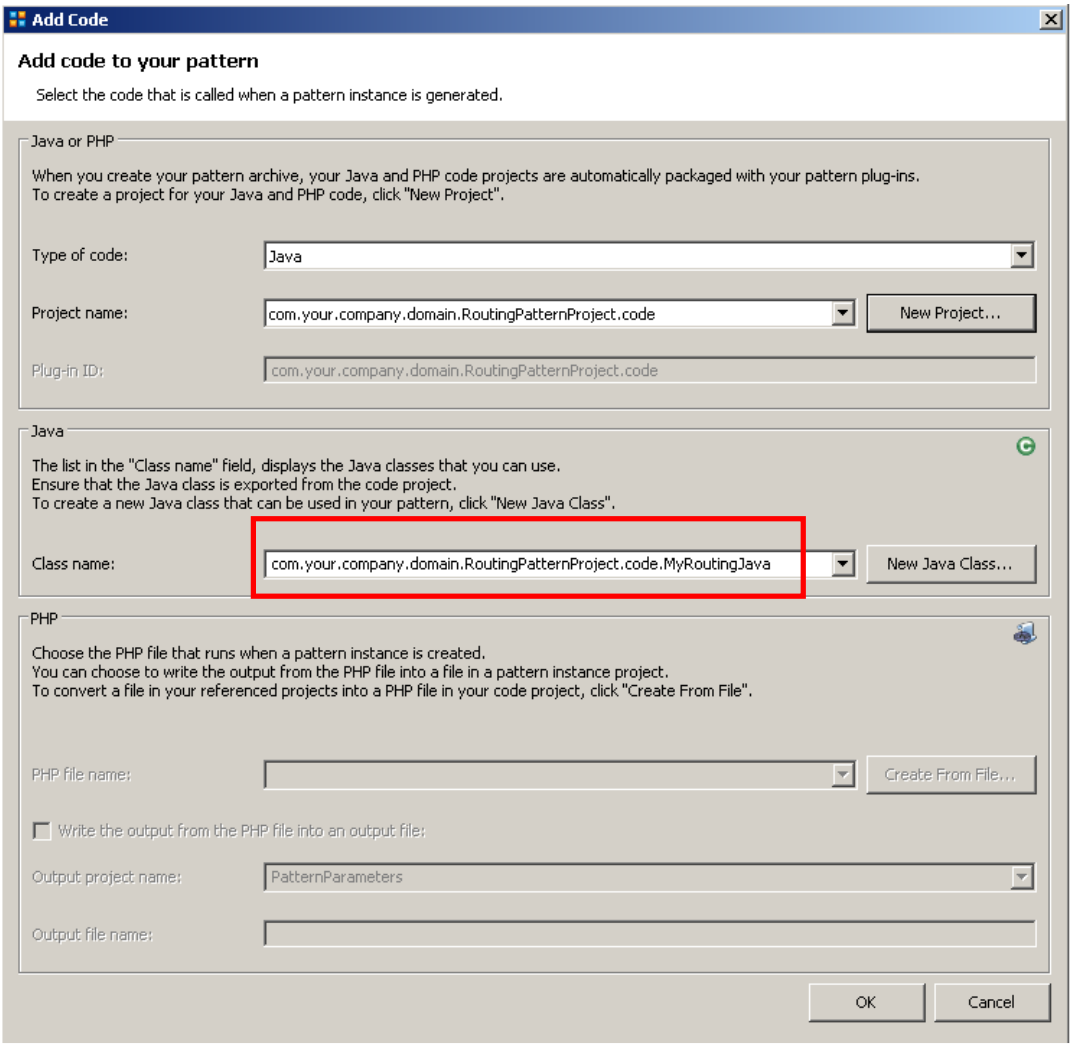
    Uncheck the "Add PHP support to the project" option since you won't need it.

    Click Finish.



38. Back in the "Add Code" window, leave the default value for the Java Class name (it should have been completed automatically with the Java class generated by the wizard)

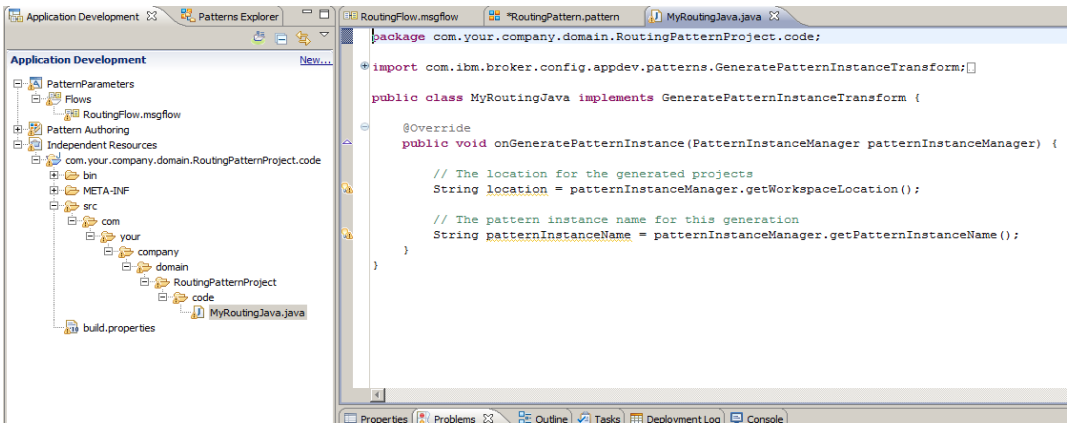
    Click OK.



39. In the Application Development view, under Independent Resources, expand the newly created Java Project (com.your.company.domain.RoutingPatternProject.code) .

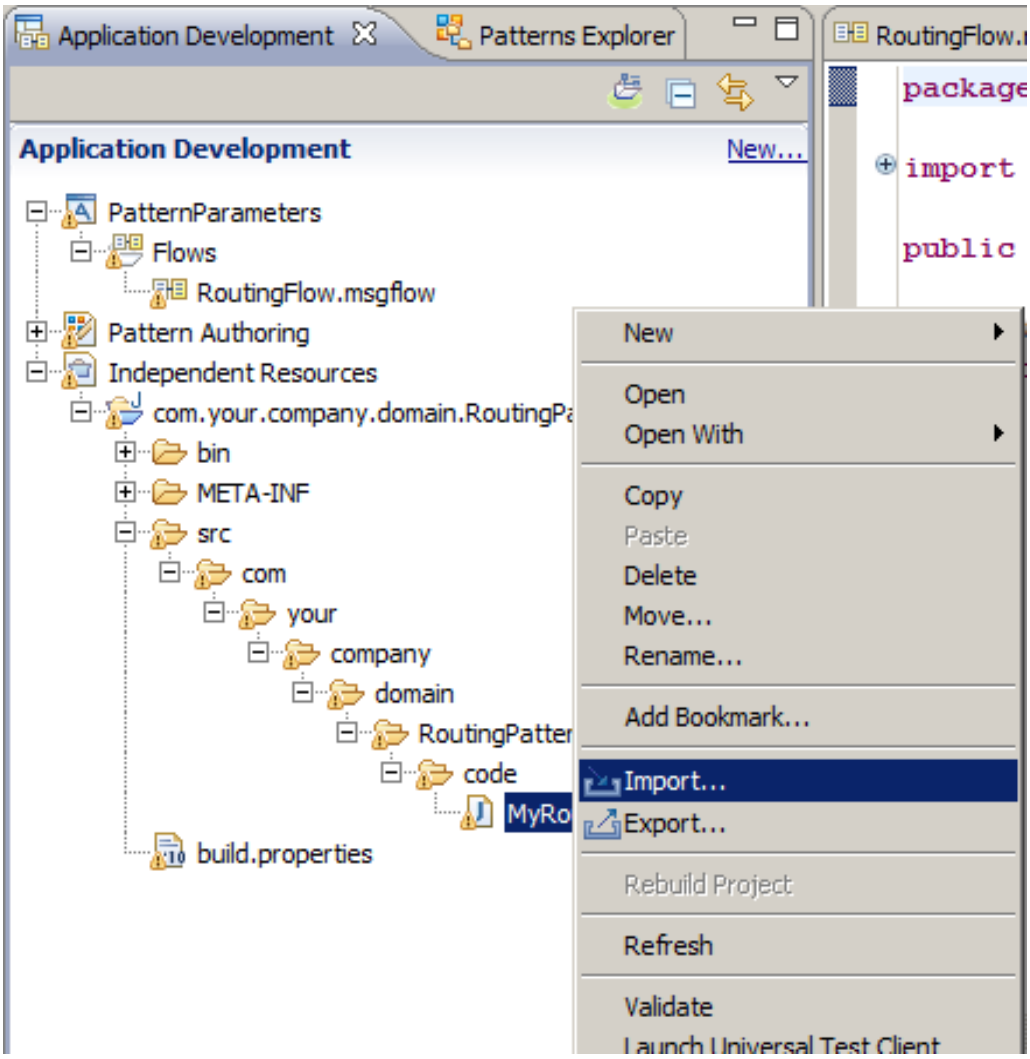
    Navigate to com.your.company.domain.RoutingPatternProject.code -> src ->com -> your -> company -> domain -> RoutingPatternProject -> code

    Double-click the MyRoutingJava.java file to open it in the Java Editor.



40. Replace the automatically generated Java Class skeleton with a pre-built version.

    Right-click on the MyRoutingJava.java file and select Import.

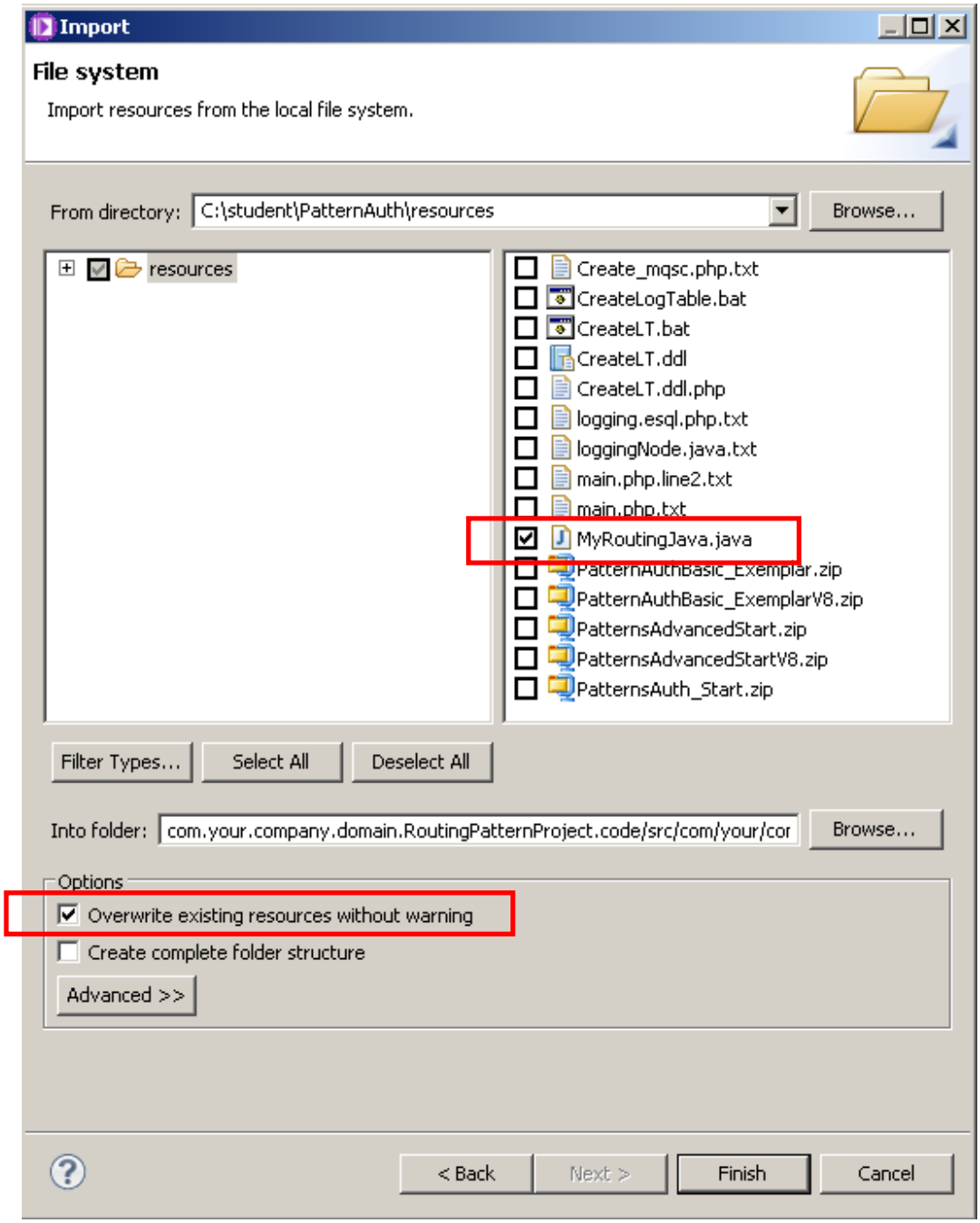


41. Select General -> File System and click Next.

42. Click on the Browse button and navigate to "C:\student\PatternAuth\resources".

    Select the MyRoutingJava.java file.

    Check "Overwrite existing resources without warning" and click Finish.



43. Take a few moments to review the java code and read the comments.

    In the first part of the code, the message flow, the Route node, the Routing Table, the OutputsAmount parameter, the Router node Filter table are loaded into memory.

    In the last line of the screenshot, the row of the Route node Filter table (the one required by the compiler) is removed.

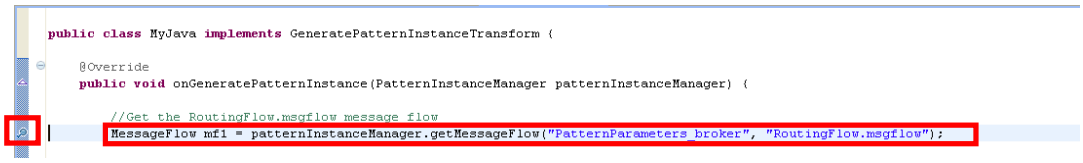
```
public void onGeneratePatternInstance(PatternInstanceManager
patternInstanceManager) {

       //Get the RoutingFlow.msgflow message flow
       MessageFlow mf1 =
       patternInstanceManager.getMessageFlow("PatternParameters",
       "RoutingFlow.msgflow");

       //Get the Route Node object
       RouteNode routeNode = (RouteNode) mf1.getNodeByName("Route");

       //Get the Parameter Table defined in the Pattern
       PatternParameterTable paramtable =
patternInstanceManager.getParameterTable("RoutingTable");


       Integer outNo;

       //Get the "outAmount" parameter (amount of desired outputs
       outNo =
Integer.valueOf(patternInstanceManager.getParameterValue("outAmount"));

       PatternParameterRow paramTableRow;

       //Get the Filter table from the route nodes
       RouteNode.FilterTable filterTable = (RouteNode.FilterTable)
routeNode.getFilterTable();

       //delete default row in the routeNode filter table
       filterTable.removeRow(filterTable.getRows().get(0));
```

44. Then in the second portion of the code, there's a cycle that associates each row of the parameter table (RoutingTable) to the Router node Filter Table.

    This portion also contains the code that creates the MQOutput nodes (with its properties taken from the pattern parameters) and connects them to the Router node.

```
for(int i=1;i<=outNo;i++){

//Add a row to the Filter Table with the Filter Pattern and Terminal for
the current output.
paramTableRow = paramtable.getRow(i-1);
RouteNode.FilterTableRow newRow = filterTable.createRow();

newRow.setFilterPattern(paramTableRow.getValue("filterPattern"));
newRow.setRoutingOutputTerminal(paramTableRow.getValue("terminal"));
filterTable.addRow(newRow);

//Create an MQOutput
MQOutputNode mqOutNode = new MQOutputNode();

//Set MQOutput node name
mqOutNode.setNodeName("MQOutput " + i);

//Set MQOutput node Queue Name
mqOutNode.setQueueName(patternInstanceManager.getParameterValue("qn"+i));

//Set MQOutput node Queue Manager

mqOutNode.setQueueManagerName(patternInstanceManager.getParameterValue("q
m"+i));
//Set location of new node on flow editor
mqOutNode.setLocation(400,50*i);
mf1.addNode(mqOutNode);

//Add terminal to the Route Node
OutputTerminal term =
routeNode.getOutputTerminal(paramTableRow.getValue("terminal"));

//Connect Route node to MQOutput node
mf1.connect(term, mqOutNode.INPUT_TERMINAL_IN);
              }
        }
```

45. Insert a Breakpoint in the Java code by double-clicking on the blue column, in the first line of the class:

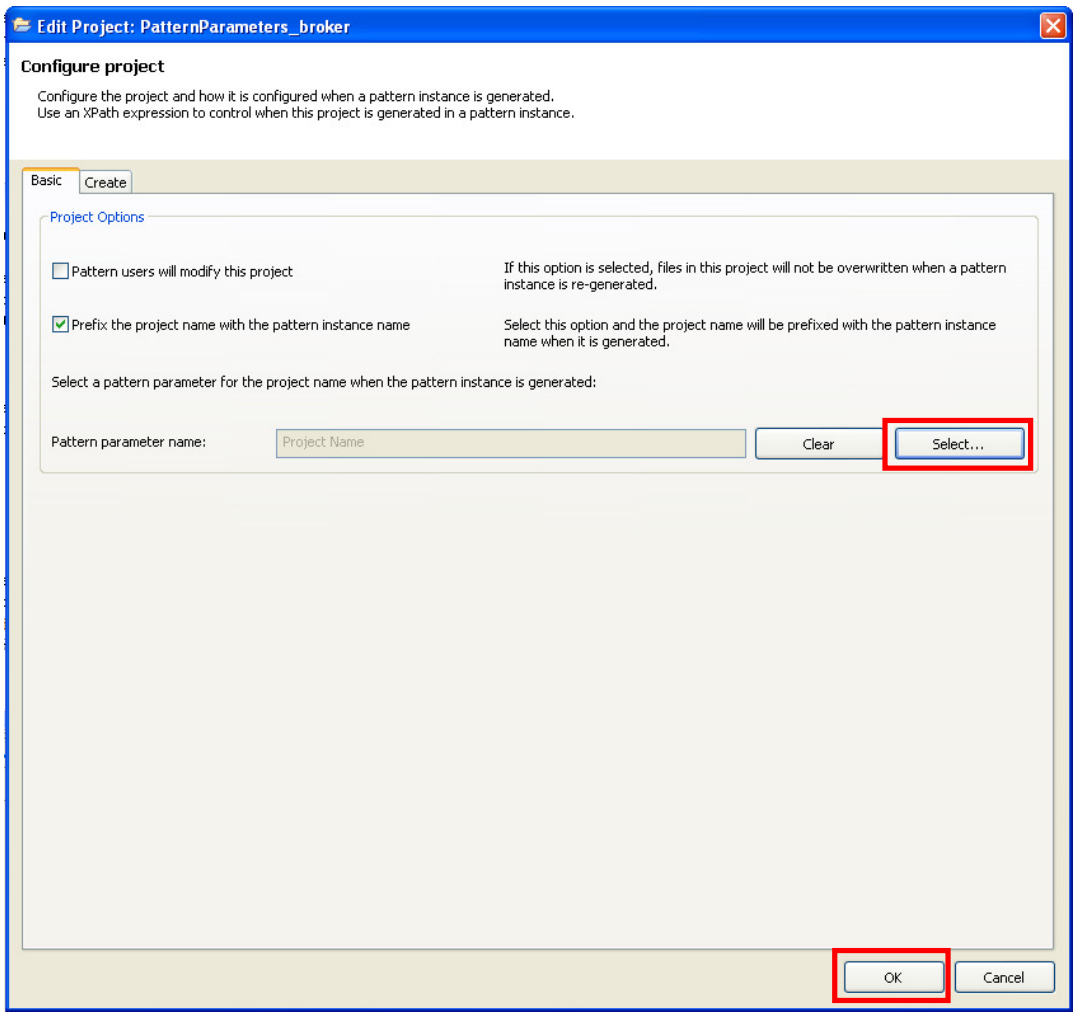


46. Close the MyRoutingJava.java file and return to the RoutingPattern.pattern in the Pattern editor.

    Click on the "Source Files" tab, and select the"PatternParameters" project.

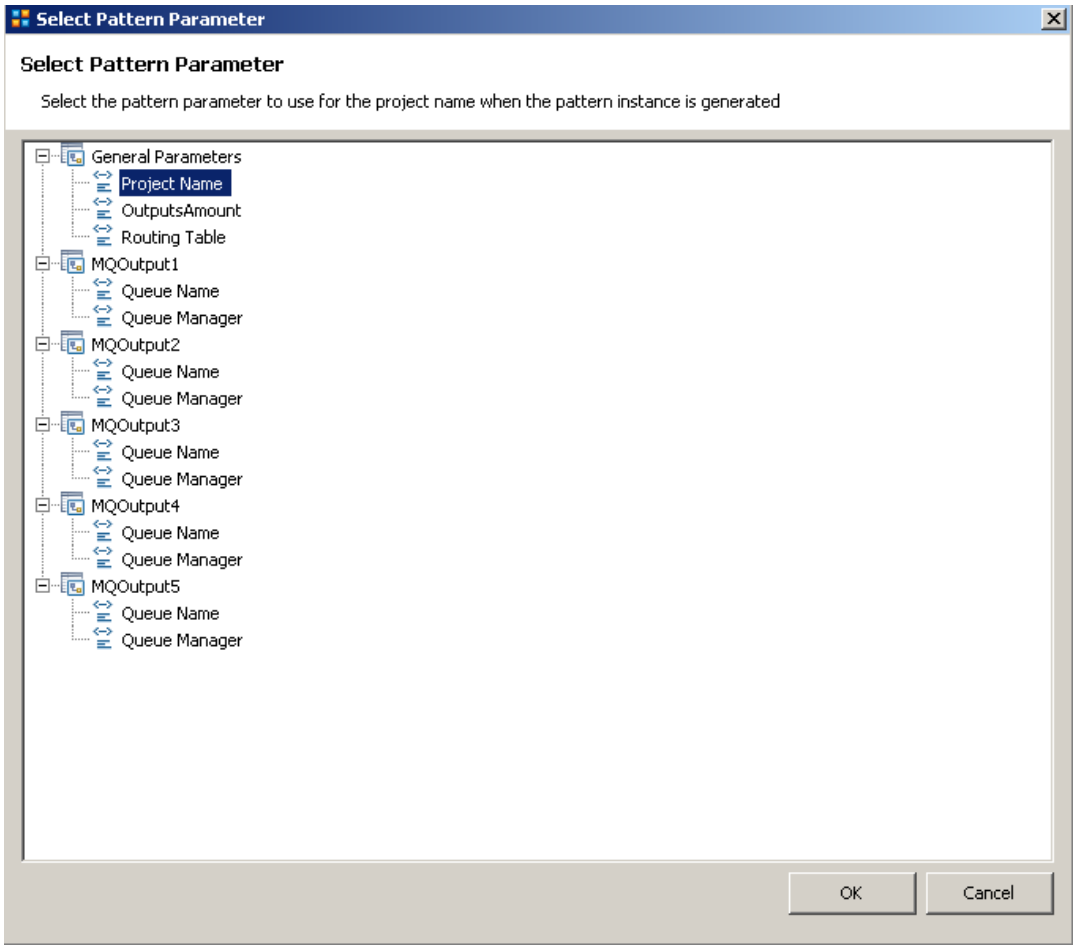
    Click on the Edit button.

47. In the "Edit Project" window, click on the Select button next to the "Pattern parameter name" field.



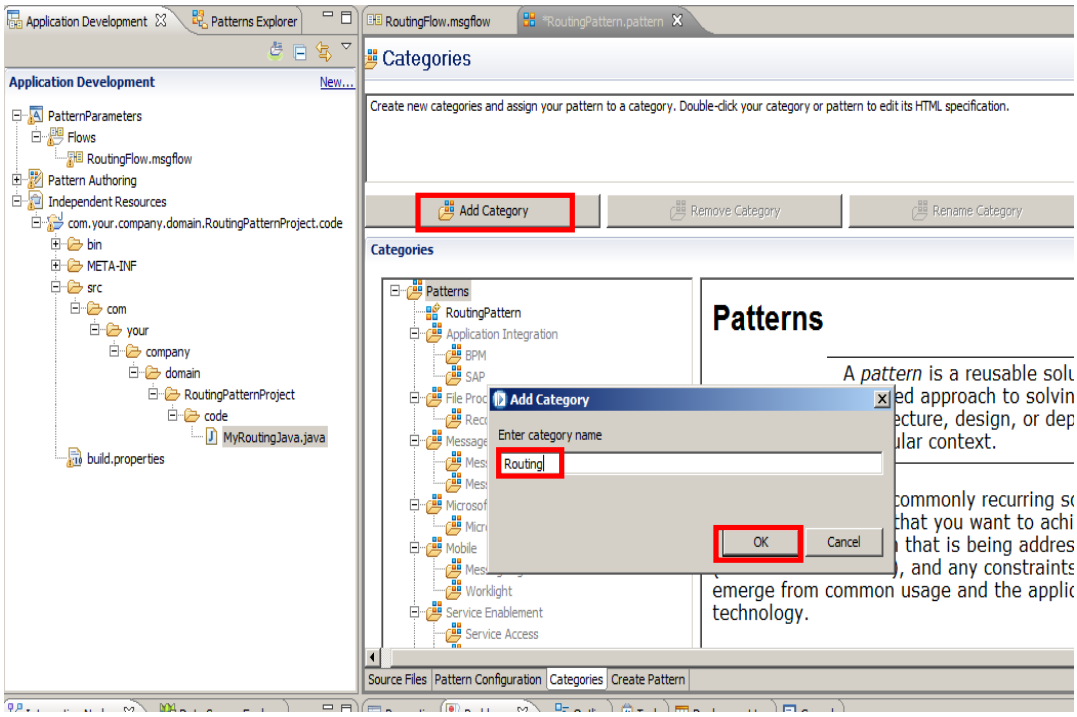
Click OK.

48. In the "Select Pattern Parameter" window , select the "Project Name" parameter under the "General Parameters" group and click OK, and OK again.



49. Select the Categories tab and click on the "Add Category" button.

    Enter "Routing" as the category name and click OK.



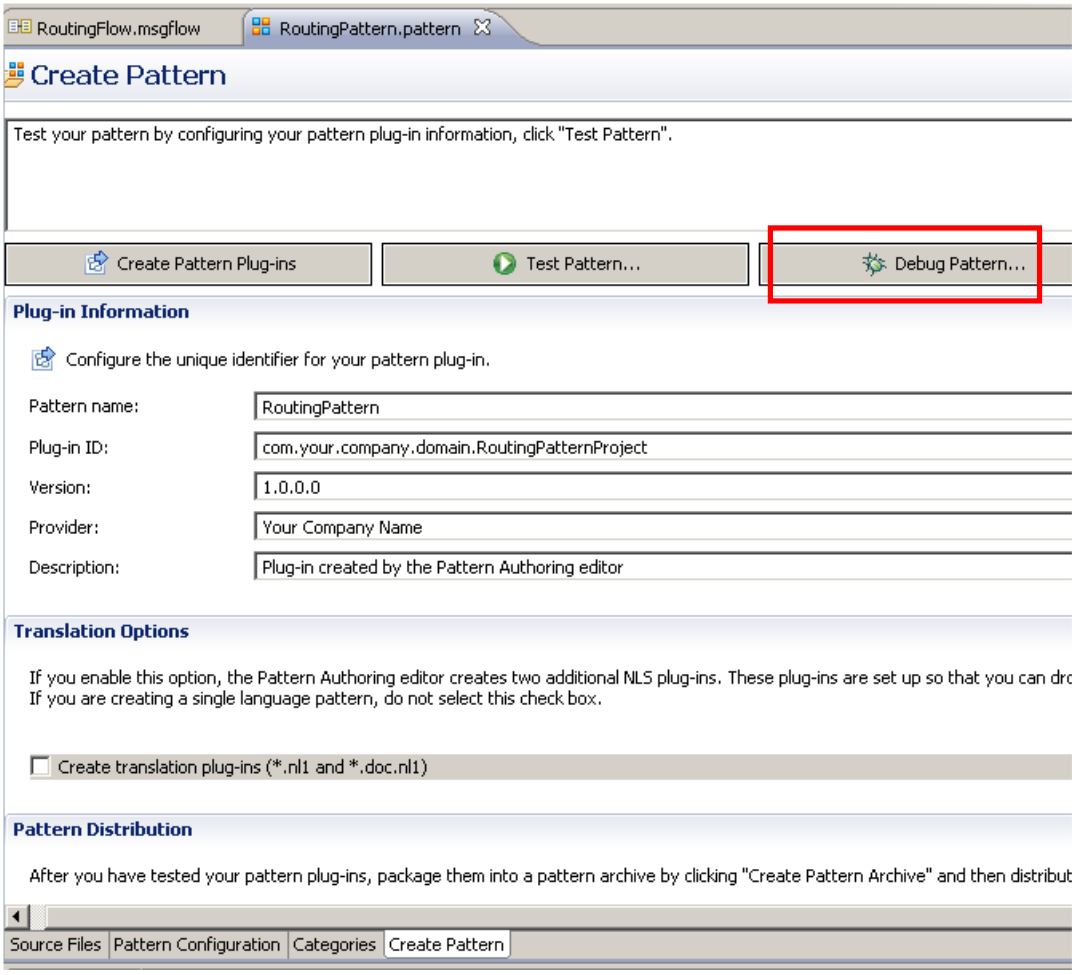
    Click Yes on the popup window to save your pattern.

50. Drag and Drop RoutingPattern under the newly created "Routing" Pattern category.

    Save the pattern.

3. Testing and Debugging the Pattern

1. Select the "Create Pattern" tab and click "Debug Pattern".

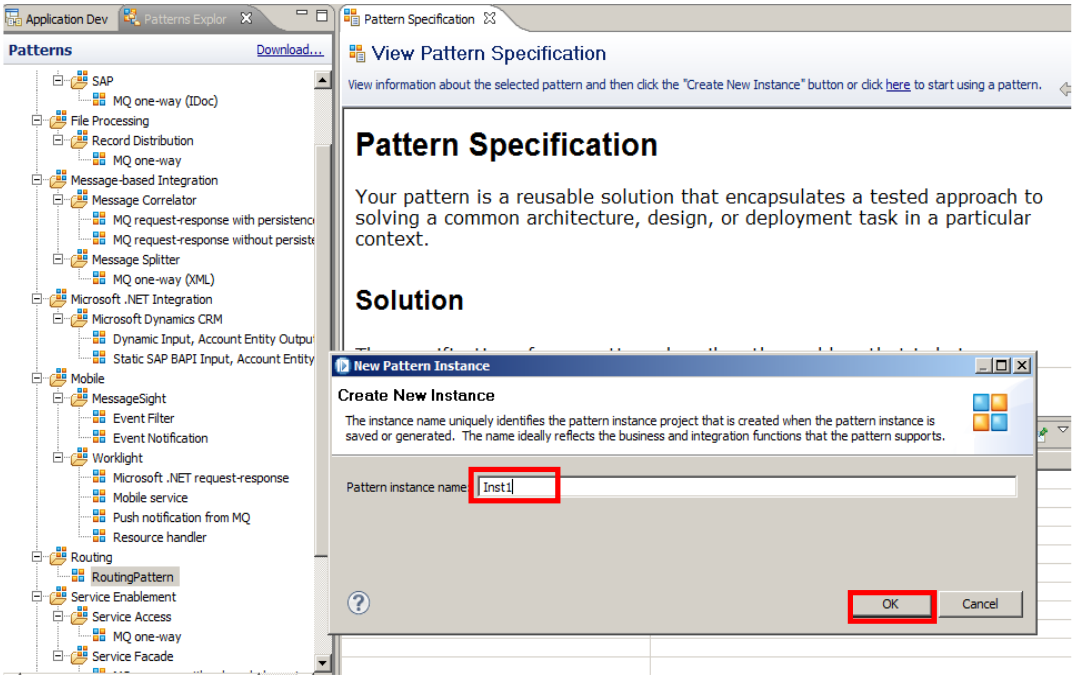


2. Accept the default workspace and click OK.

3. In the new Toolkit instance, click on the "Patterns Explorer" tab.

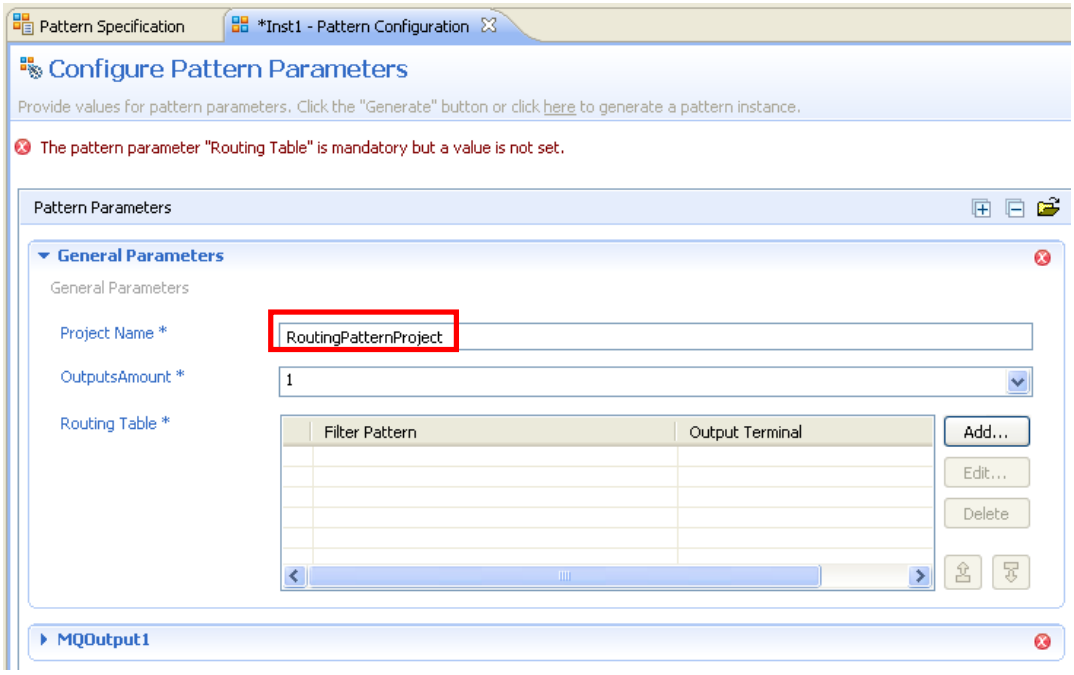
   Double-click on "RoutingPattern" to create a new instance.

   Enter "Inst1" as the Pattern instance name and click OK.



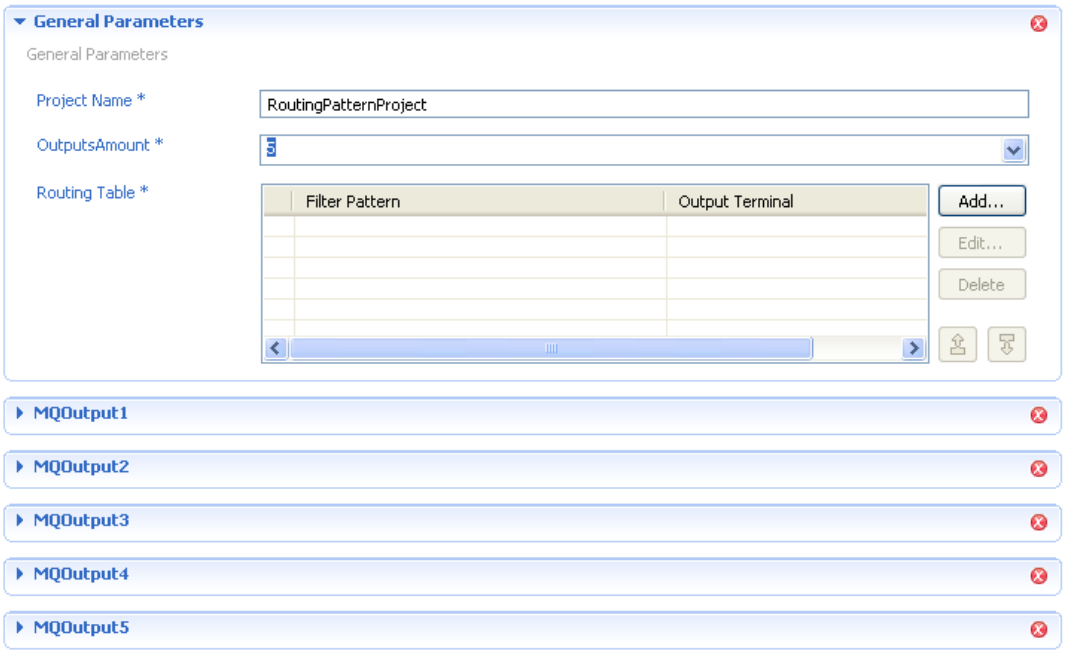
4. In the Pattern Configuration, expand the "General Parameters" group.

   Enter "RoutingPatternProject" as the Project Name.

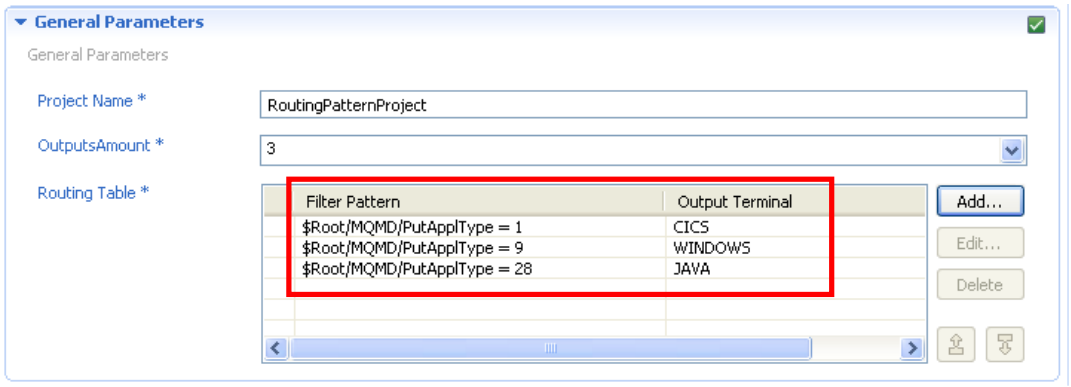


Note that since OutputsAmount is set to "1", there is only one MQOutput group (MQOutput1)

5. Change OutputsAmount to "5" and check that 5 MQOutput groups are shown.

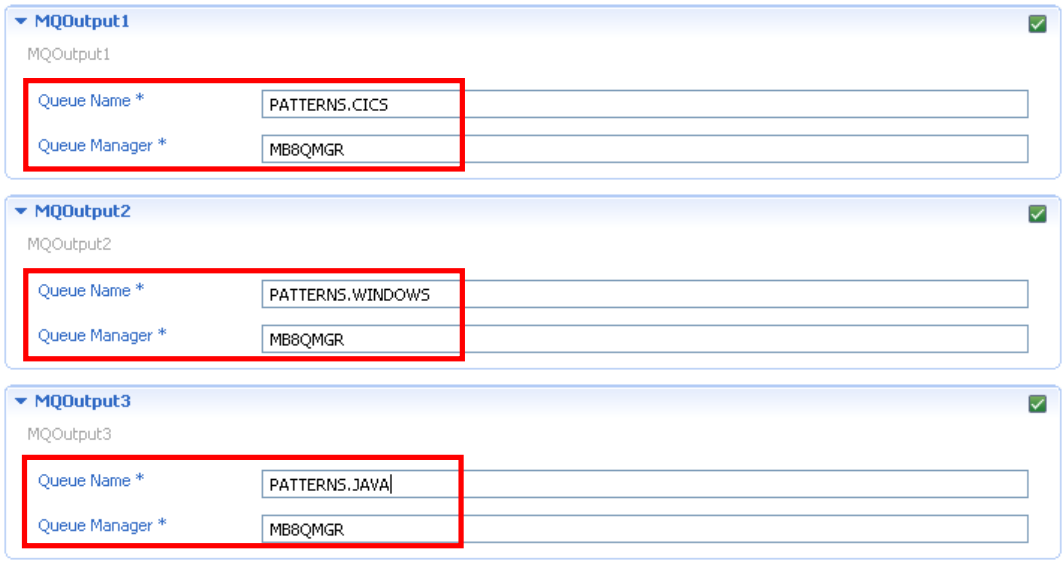


6. Change OutputsAmount to "3" and insert the following 3 rows in the Routing Table (using the Add button):



Note – if you specify OutputsAmount to a larger number, make sure you create the corresponding number of Routing Tables entries ….. the java code is not production-ready….

7. Now complete the Queue Name and Queue Manager parameters for the 3 MQOutput groups:

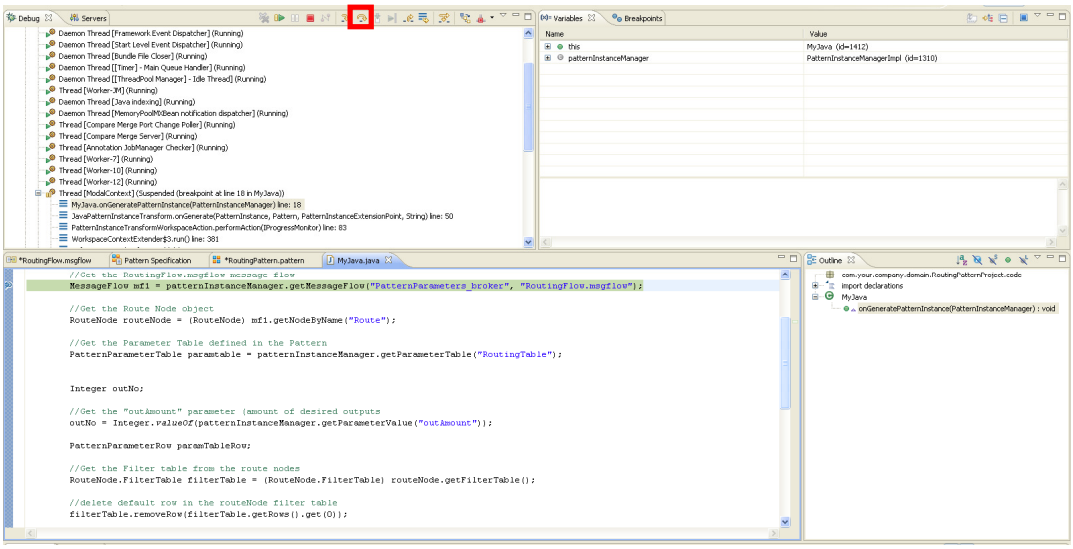


8. Click the Generate button.

9. Since you put a breakpoint in MyRoutingJava.java, the debugger will pause the execution of the java code and allow you to run it step by step.

   Go back to the original Toolkit instance, and click Yes in the "Confirm Perspective Switch" popup to switch to the Debug perspective.

10. Click on the "Step Over" icon ( ) to move one step forward.



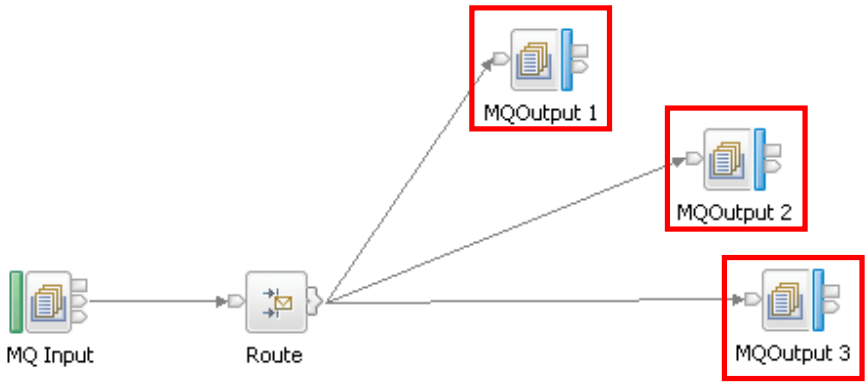
You can go through the code step by step, or click on the "Resume" icon ( ) and let it run until the end.

11. When finished, go back to the second (Test) Toolkit instance, and expand the generated Application Inst1_RoutingPatternProject.

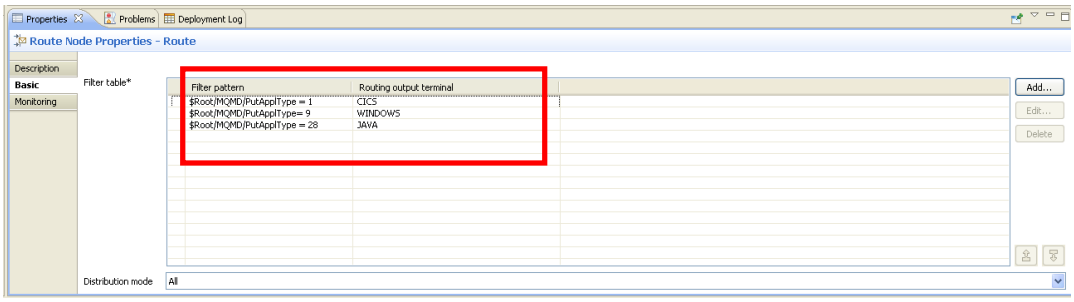
    Note that the Application name is composed by the Instance name as a prefix (Inst1), and the Project Name parameter you entered in the Pattern Configuration Editor (RoutingPatternProject).

    Double-click on the RoutingFlow.msgflow to open it.

12. Check that the message flow has 3 MQOutput Nodes, each with its defined Queue Name and Queue Manager properties.



13. Double-click on the Route node and verify that its Filter table was completed using the Routing Table parameter:



This concludes the "Groups and Table Parameters" Pattern Authoring lab.