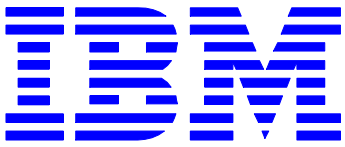


IBM Integration Bus

Pattern Authoring Lab 4

Adding and tailoring nodes using PHP and Java

June, 2013

Hands-on lab built at product code level version 9.0.0.0

1. Lab Objectives

In this lab, you will see how to create a pattern which enables a pattern user to dynamically change the message flow logic, based on the value of pattern parameters. The pattern that you create will include a new Compute node and associated ESQL which will be added to the generated message flow if the pattern selects the “database logging” option. The new node will take the entire contents of the message tree and insert it into a database row.

Two Pattern Authoring primary functions are used:

1. PHP scripts are used to create to the ESQL file for a new Compute node, and to create the related MQSC and DDL scripts to create MQ queues and database definitions.
2. A Java application, which creates the new Compute itself, and adds it to an existing message flow, depending on the value of the pattern parameters.

This hands-on lab is the fourth in a series of labs demonstrating the Pattern Authoring tools in IBM Integration Bus V9.0. Whilst this lab does not require any of the artefacts created in the first three labs, it is strongly recommended that you perform these earlier labs, before performing this lab. The reader is assumed to be familiar with the basic pattern authoring techniques explained in the earlier labs, and they are not covered here.

This lab uses a new message flow (exemplar) as its starting point; this is provided in the c:\student\PatternAuth directory, and is also packaged as a project interchange file for installation on the reader’s own installation.

Specific Tasks
The lab comprises the following key steps:

2.1 Import the message flow (exemplar) and create the Pattern project.
2.2 Configure the pattern to add a new Compute node.
- Create a PHP script to create the associated ESQL module
- Create a Java application which creates the Compute node and adds it to the flow

2.3 Create a new pattern category
2.4 Test the pattern
2.5 Extend the pattern to add further PHP scripts which will create MQSC and DDL scripts.
2.6 Test the pattern again

Source files
This lab provides a number of source files. These are located in the folder c:\student\PatternAuth\resources.

2. Lab Guide

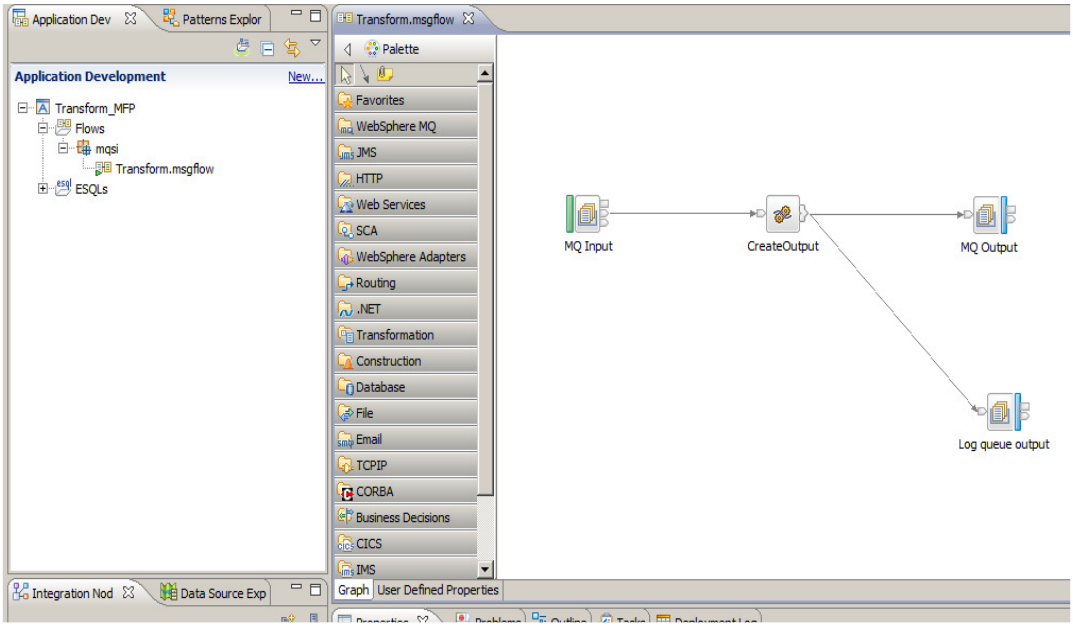
2.1 Create the pattern and add new pattern parameters

1. To avoid any naming conflicts, remove all projects from the earlier Pattern Authoring labs, including pattern projects, from your workspace (or create a new workspace).

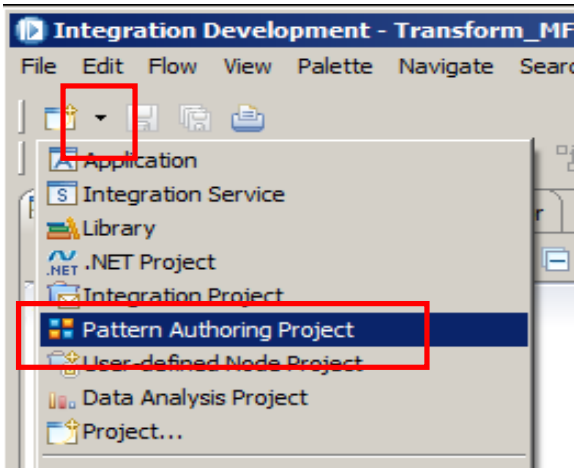
   Ensure you remove any associated pattern projects under “Independent Resources”.

2. Import the PI file c:\student\PatternAuth\resources\PatternsAdvancedStartV8.zip. This will create a project named Transform_MFP.

   Open the message flow, named Transform.msgflow.



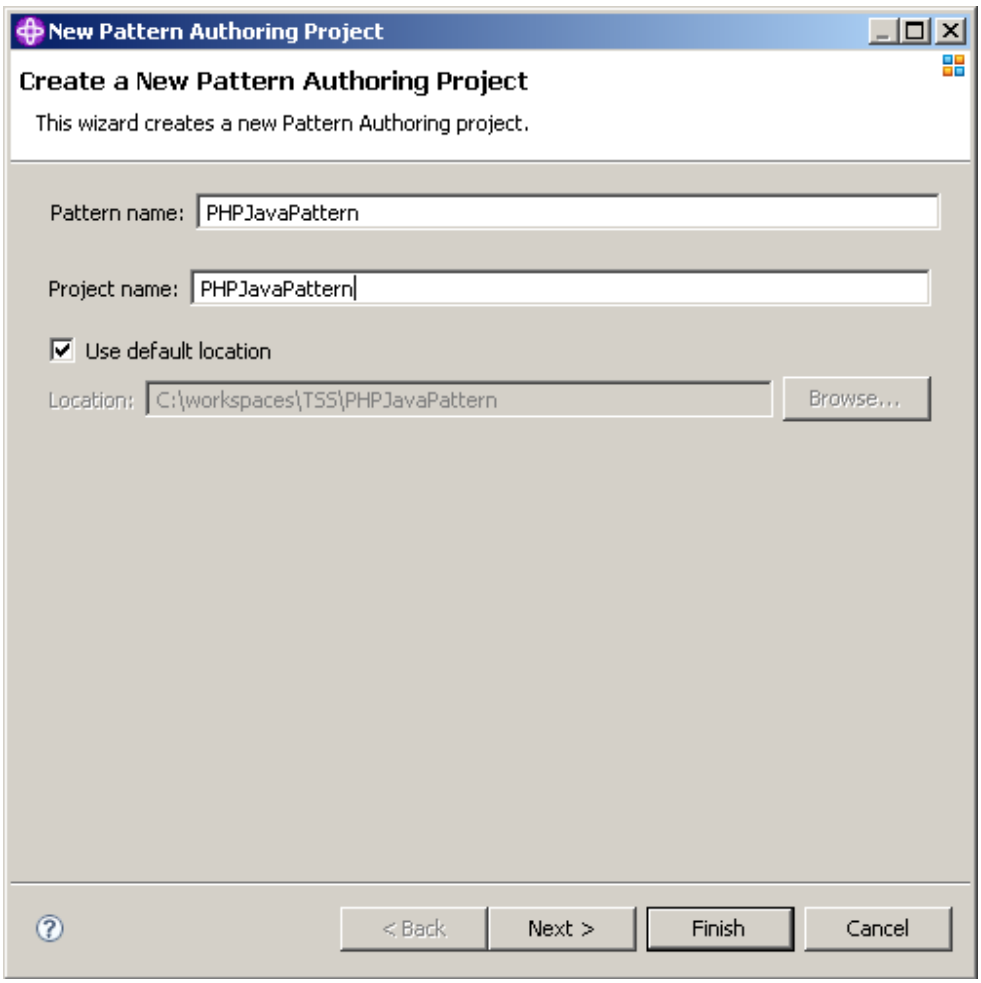
3. Create a new Pattern Authoring project.



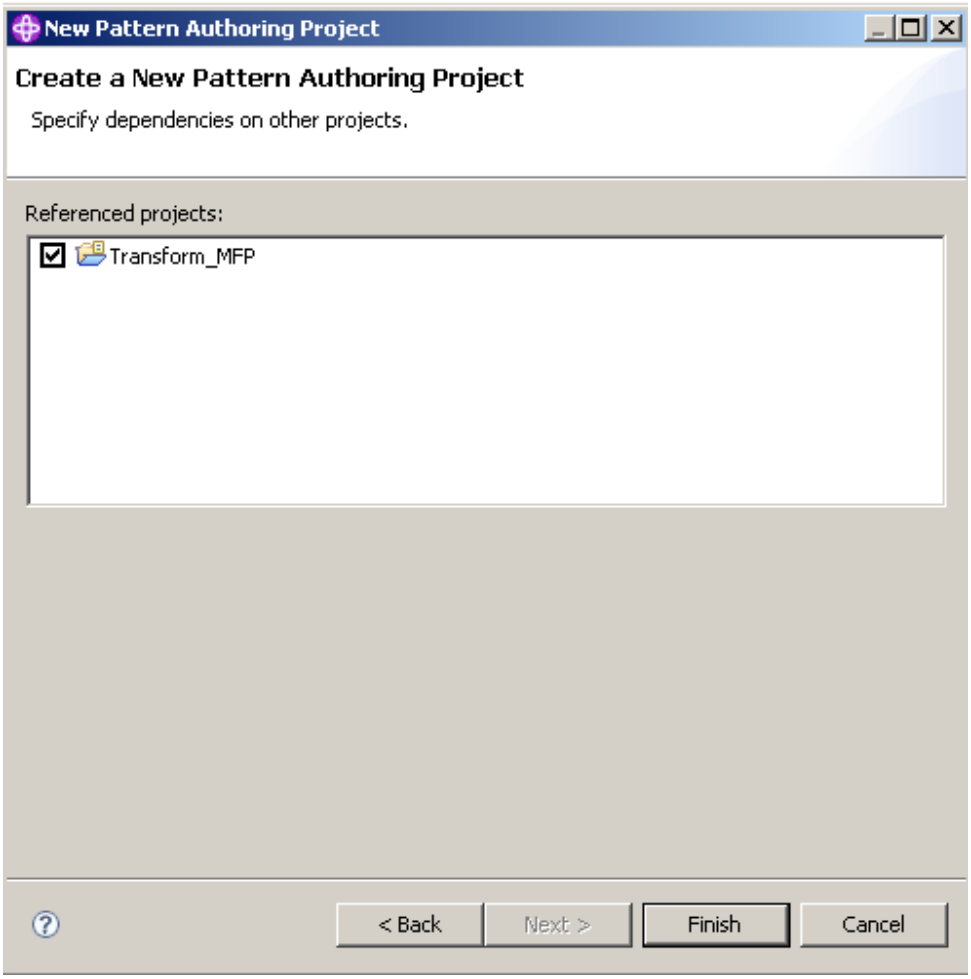
4. Name the pattern PHPJavaPattern (this name will match code later in the lab).

   Name the project PHPJavaPattern.

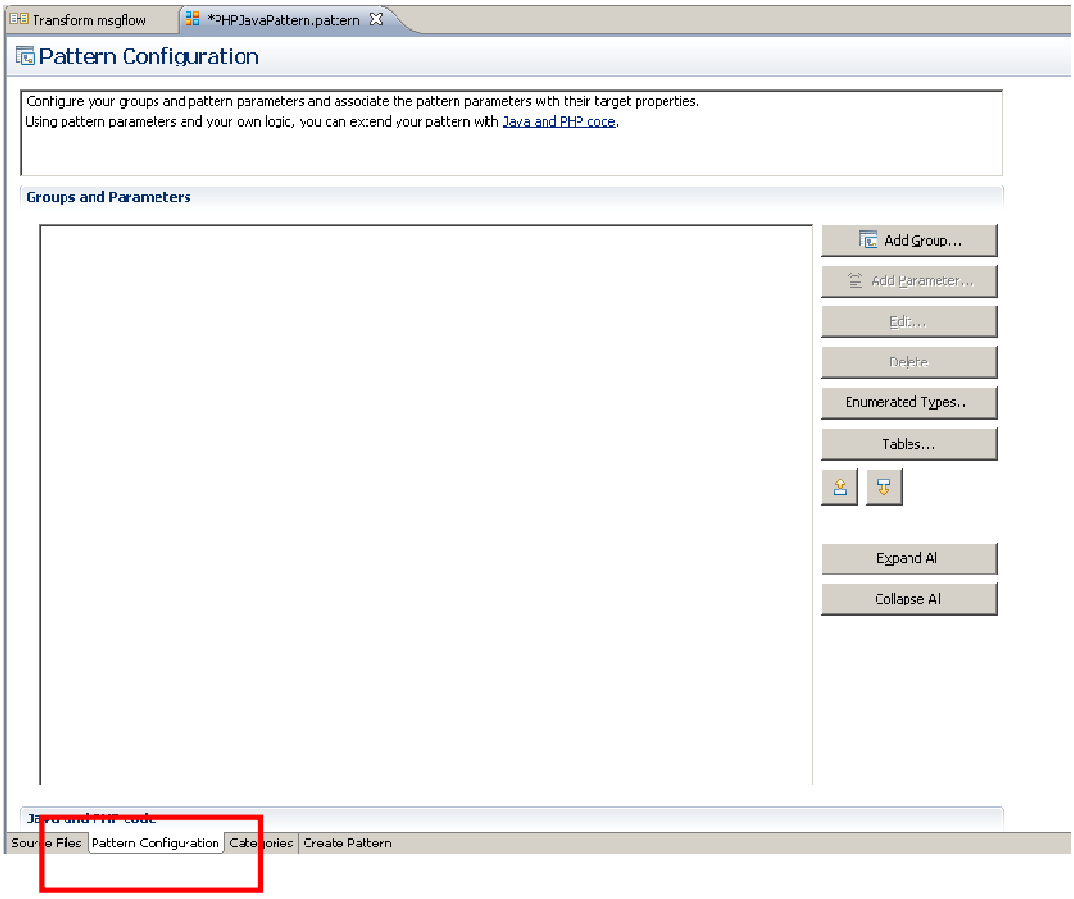
   Click Next.



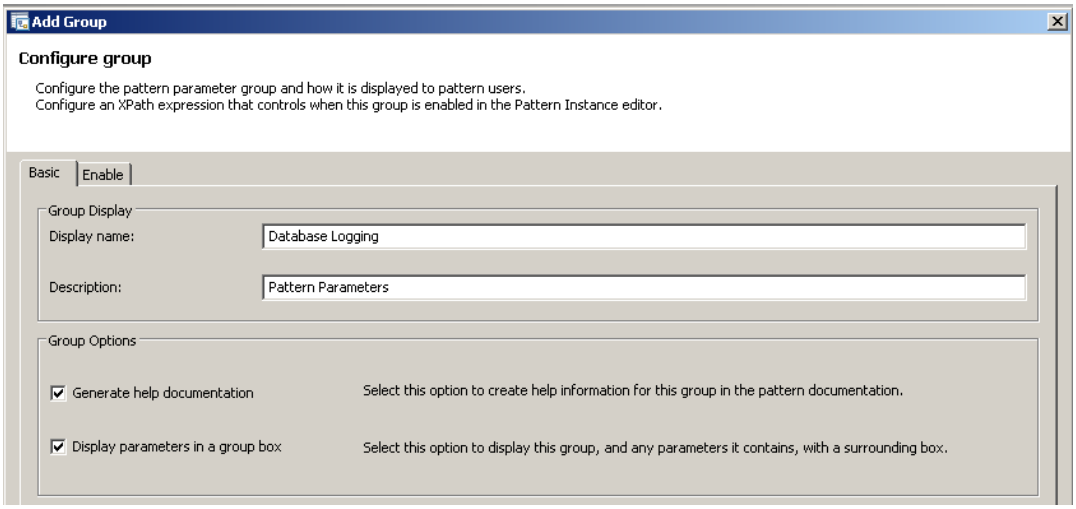
5. Select the Transform_MFP referenced project, and click Finish.



6. In the new pattern project, select the “Pattern Configuration” tab.



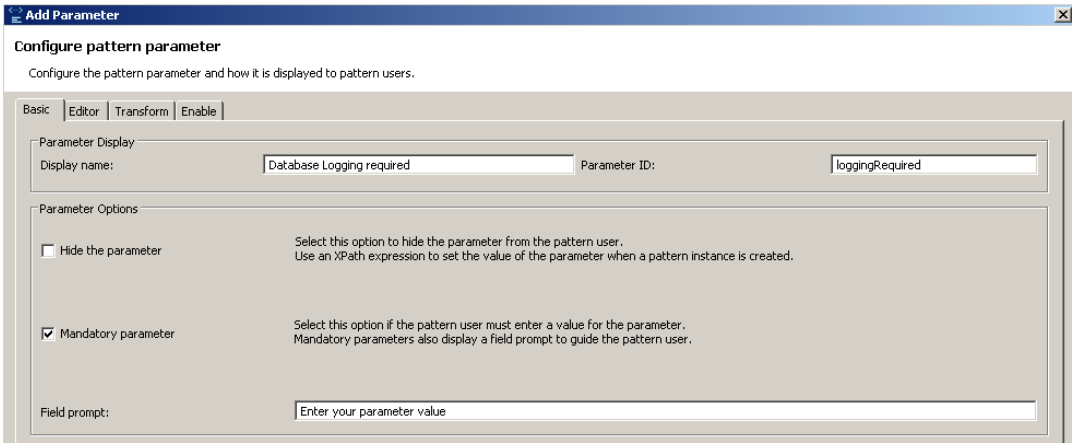
7. Create a new pattern group (click Add Group), and name it “Database Logging”.



Click OK.

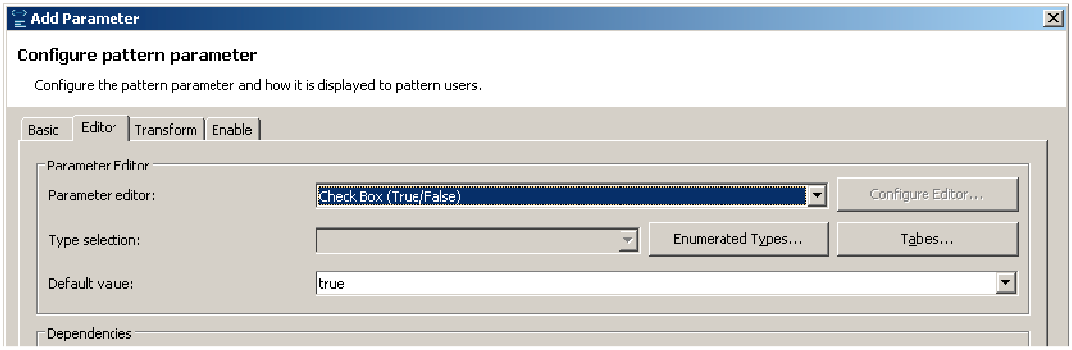
8. Add a new pattern parameter (click Add Parameter). Set the following values:

   Display name: Database Logging required
   Parameter ID: loggingRequired (case sensitive)



9. On the Editor tab, set the Parameter Editor to “Check Box”, and the default value to true.

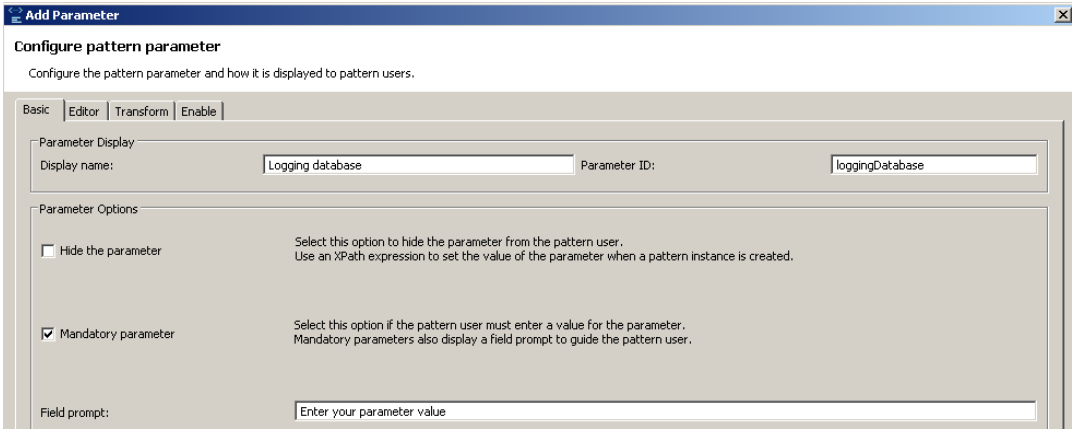
   Click OK.



10. Now add another pattern parameter, as follows:
    On the Basic tab:
    Display name: Logging database
    Parameter ID: loggingDatabase

    On the Editor tab, set default value to SAMPLE

    Click OK.



11. Similarly, add pattern parameters for database schema and table name:

    Display name = Logging database schema
    Parameter ID = loggingSchema
    Default value = wmbadmin

    Display name = Logging table
    Parameter ID = loggingTable
    Default value = LOG_TABLE

2.2 Configure the pattern to add a new Compute node

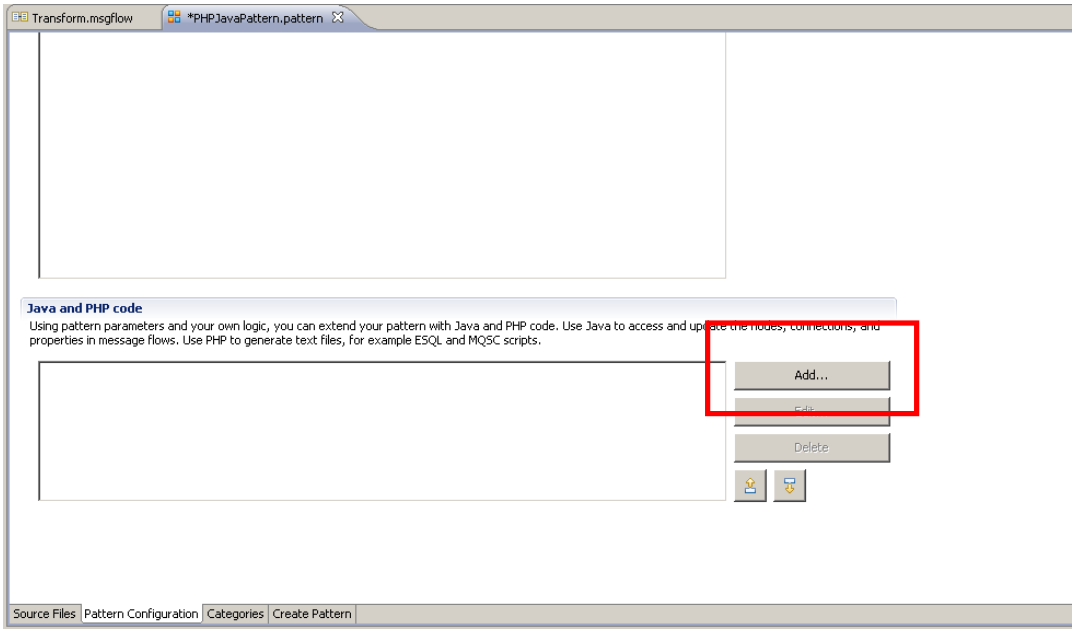
This section shows you how to add a new node with the pattern authoring tools.

We will add a Compute node to the flow (setting a couple of node properties), and create the associated ESQL file. This node will write the contents of the message tree to a single column in a database table. It will therefore need to have an associated ESQL file, and have the Database source node property set.

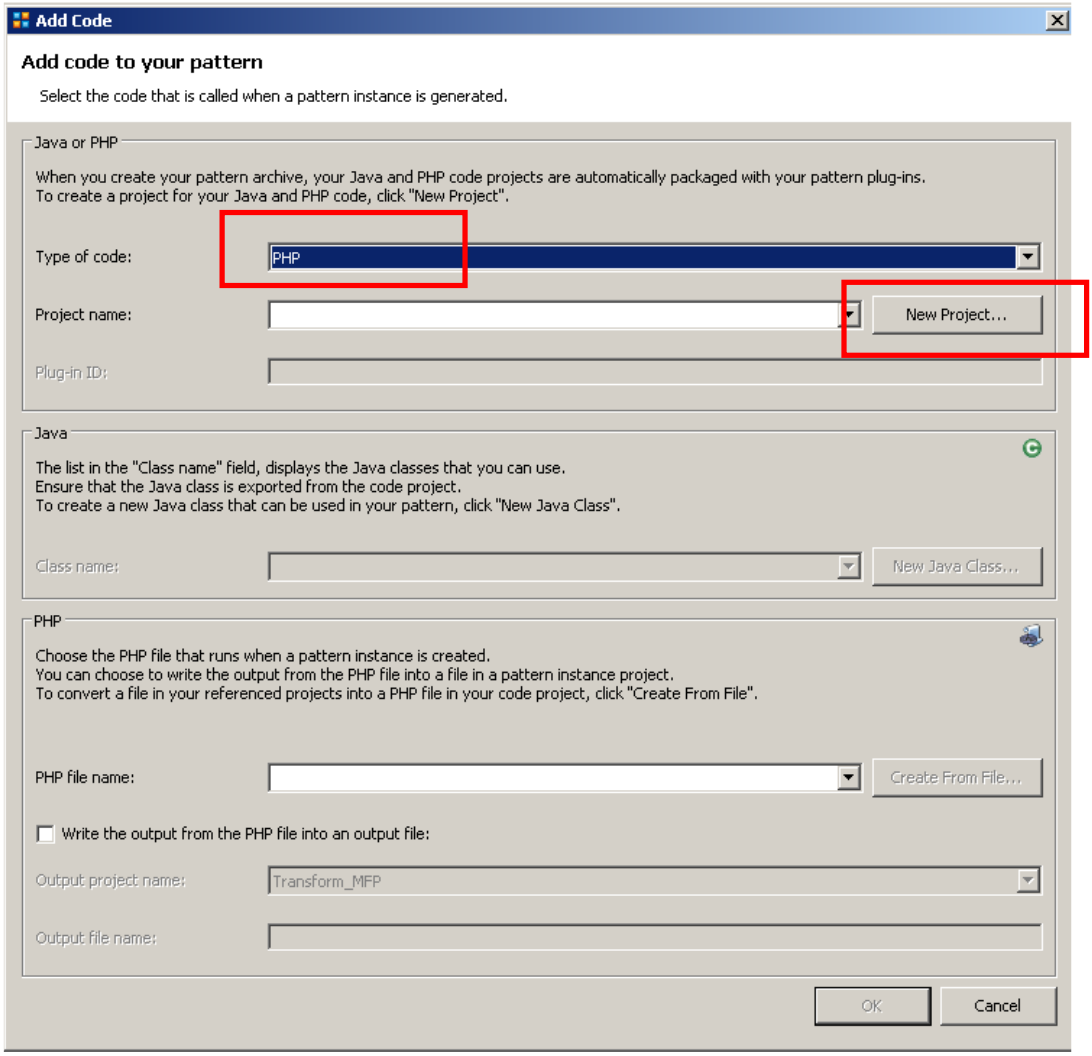
1. We will first create the ESQL module that the Compute node will reference. This is done with a PHP script.

   On the Pattern Configuration tab, move down to the bottom of the editor, to the “Java and PHP code” section.

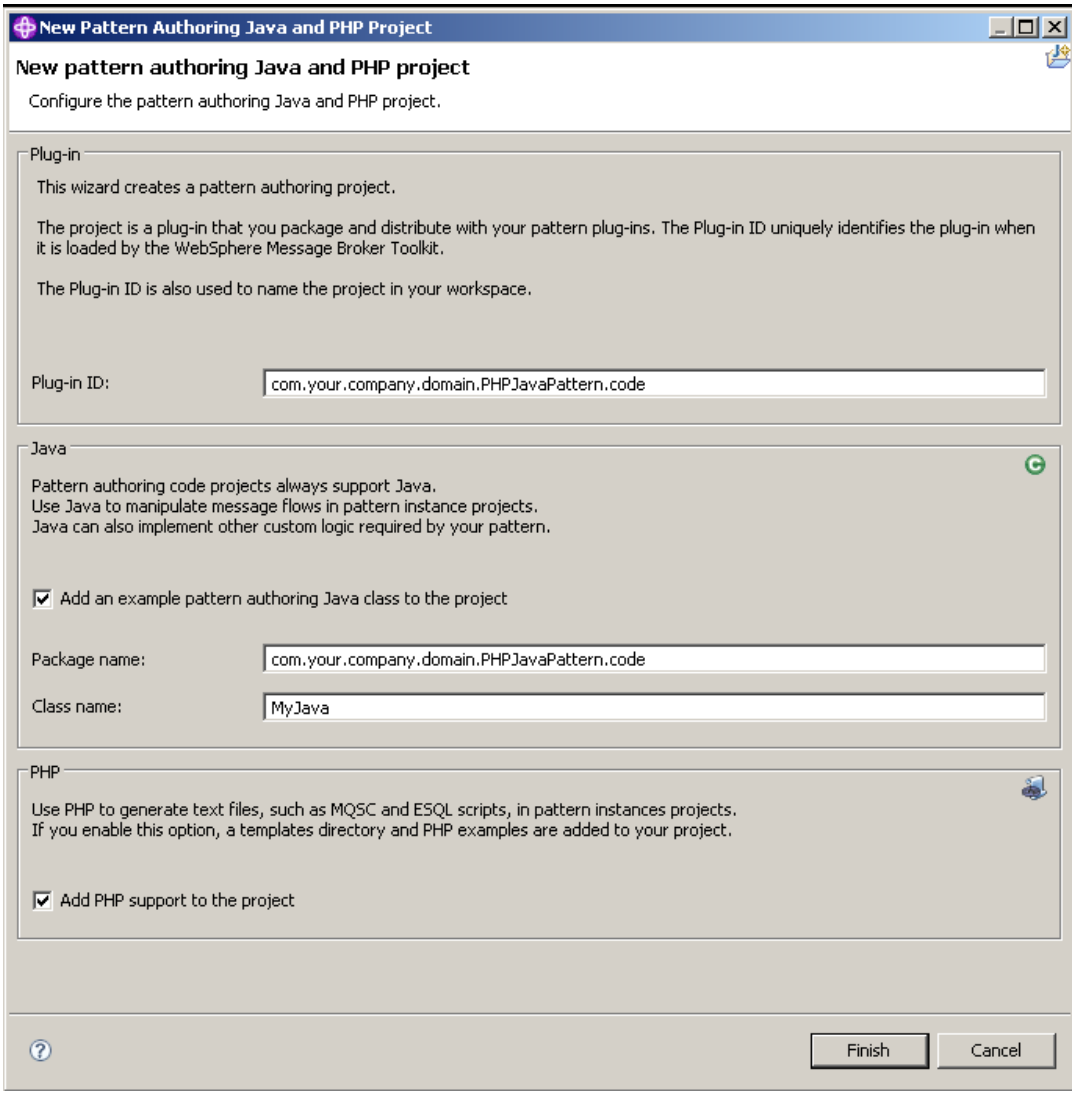
   Click Add to add some PHP or Java code.



2. Set the “Type of code” to PHP, and click “New Project”.

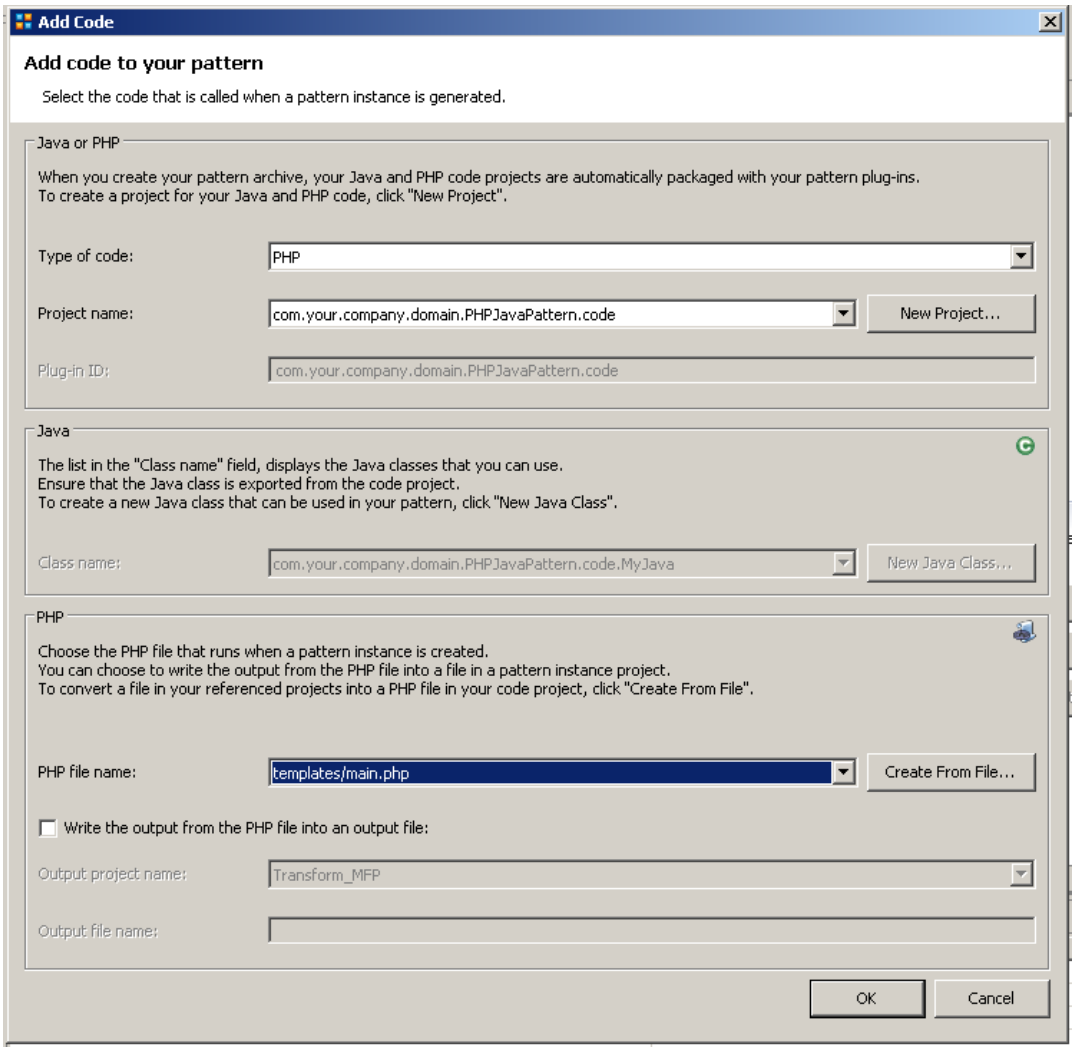


3. Accept the defaults and click Finish.

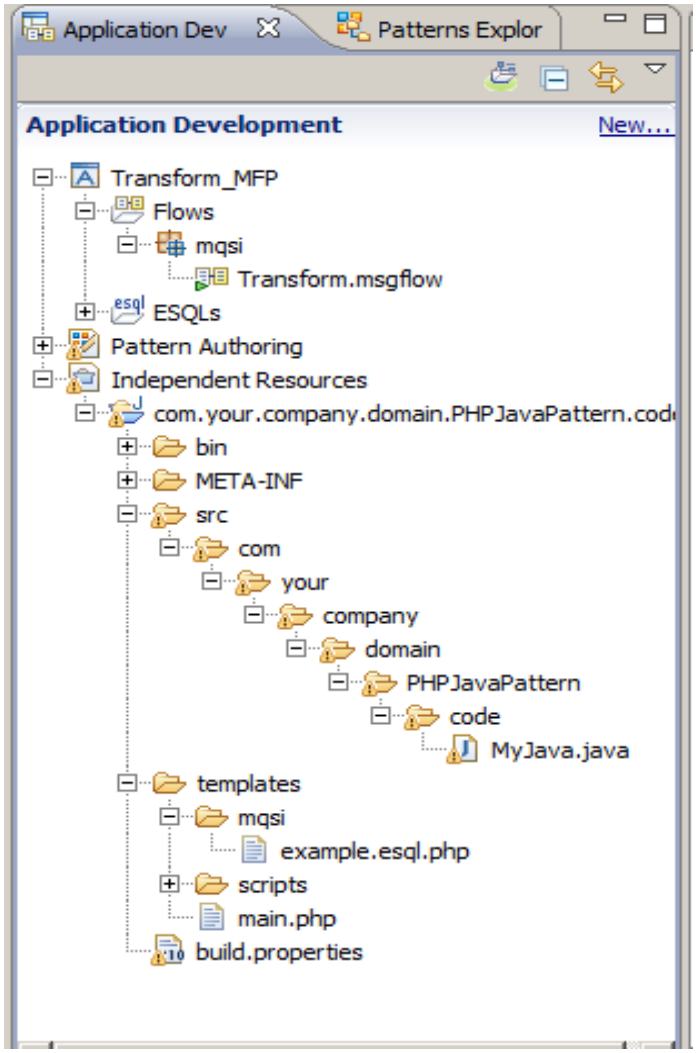


4. Note that the PHP file name has been set to templates/main.php. We will leave it for this lab, but you can change this as required.

   Click OK.



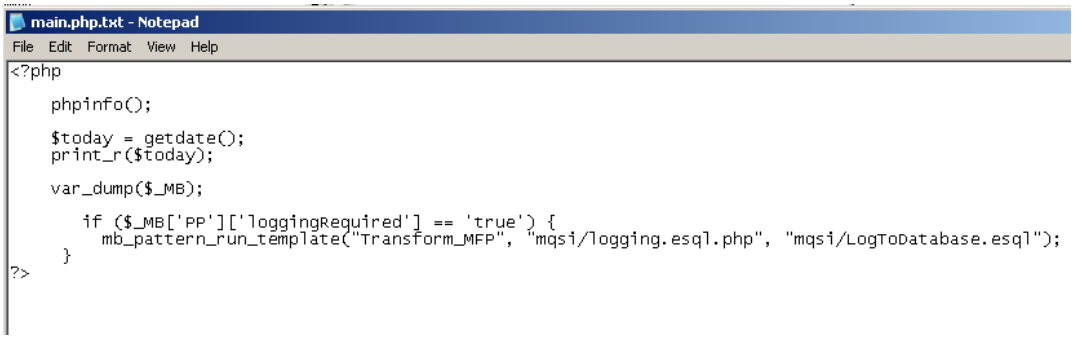
5. In the project navigator, under Independent Resources, expand the src and templates directories, as shown.



6. Open the main.php file.

   Delete the entire contents of this file.

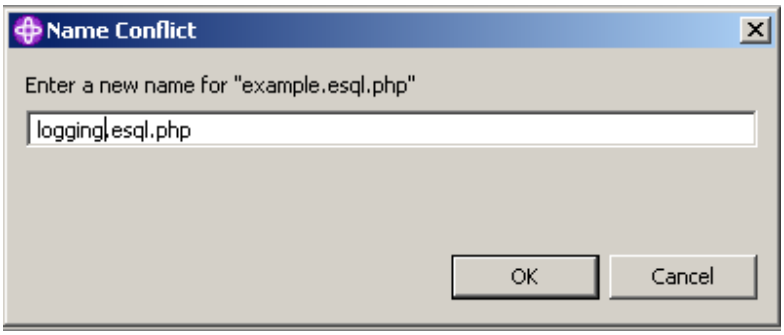
   Replace the contents with the entire contents of the file c:\student\PatternAuth\resources\main.php.txt.



   Things to observe:
   a) The “if” statement checks the value of the loggingRequired pattern parameter.
   b) The mb_pattern_run_template statement references the “Transform_MFP” project.
   c) It runs the mqsi/Logging.esql.php script (we’ll create this shortly)
   d) The output is written to the file mqsi/LogToDatabase.esql.

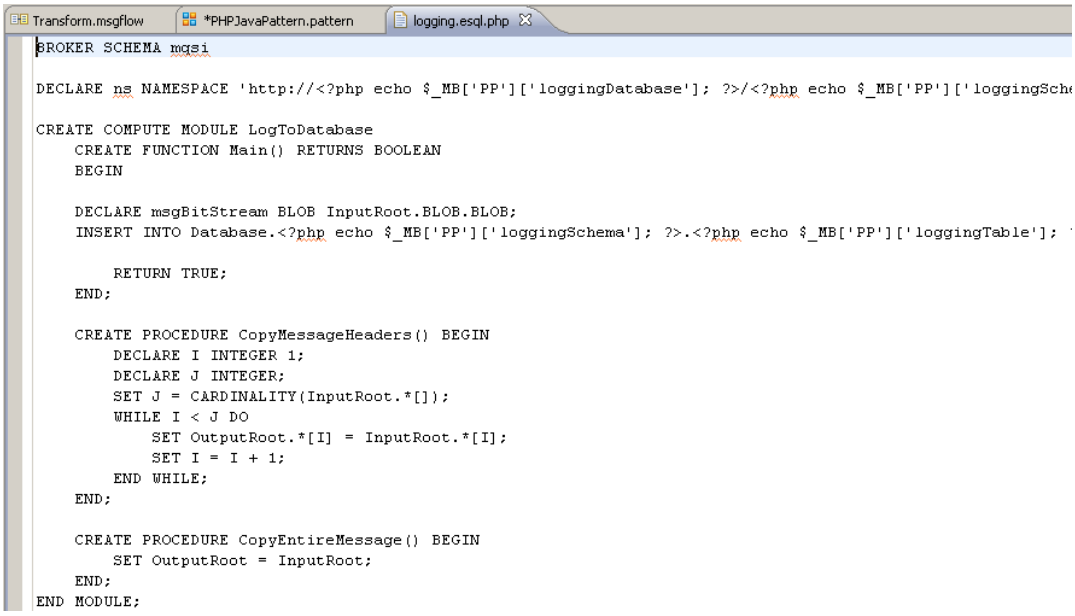
   Save and close main.php.

7. Rename the file example.esql.php to logging.esql.php (under templates\mqsi).
   Note: make sure you re-name the correct file



8. Open the file logging.esql.php, and replace the entire contents of this file with the contents of c:\student\PatternAuth\resources\logging.esql.php.txt.

   Note that the Broker Schema, mqsi, has been hard-coded into the PHP script. This can be replaced with a variable value if required, of omitted completely if you are going to use the default schema.

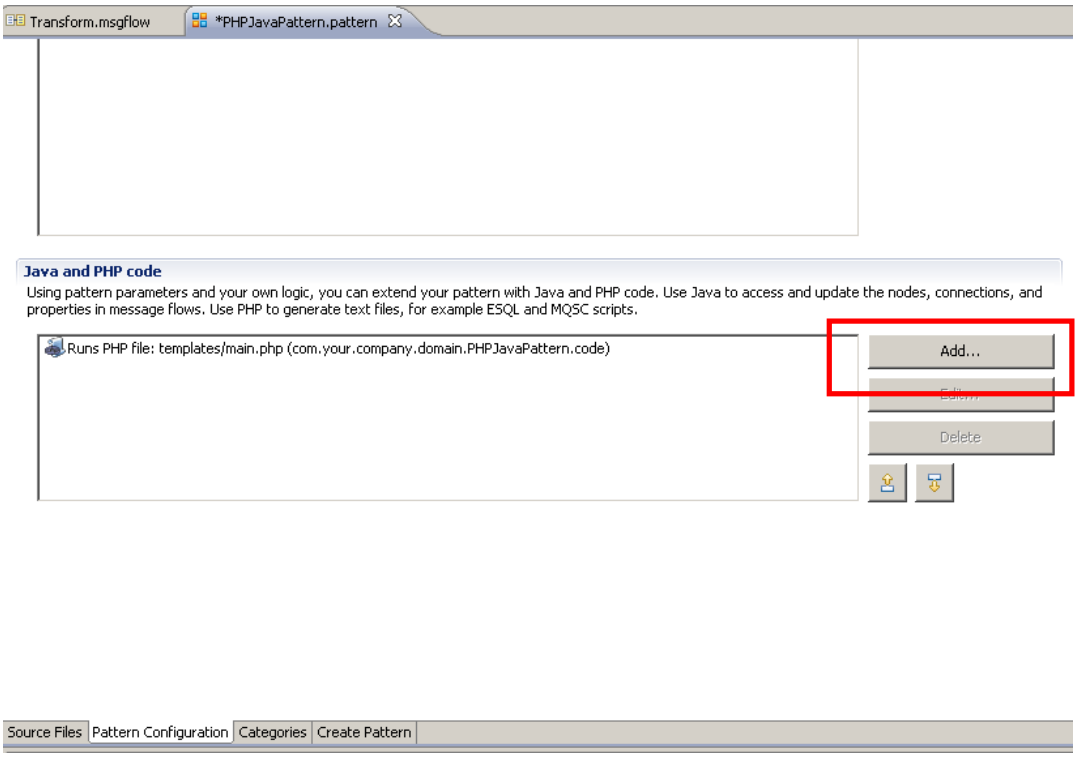


   A number of the static values have been replaced with variables based on PHP strings:
   a) <?php echo $_MB[MB['PP']['loggingDatabase']; ?>
   b) <?php echo $_MB[MB['PP']['loggingSchema']; ?>
   c) <?php echo $_MB[MB['PP']['loggingTable']; ?>

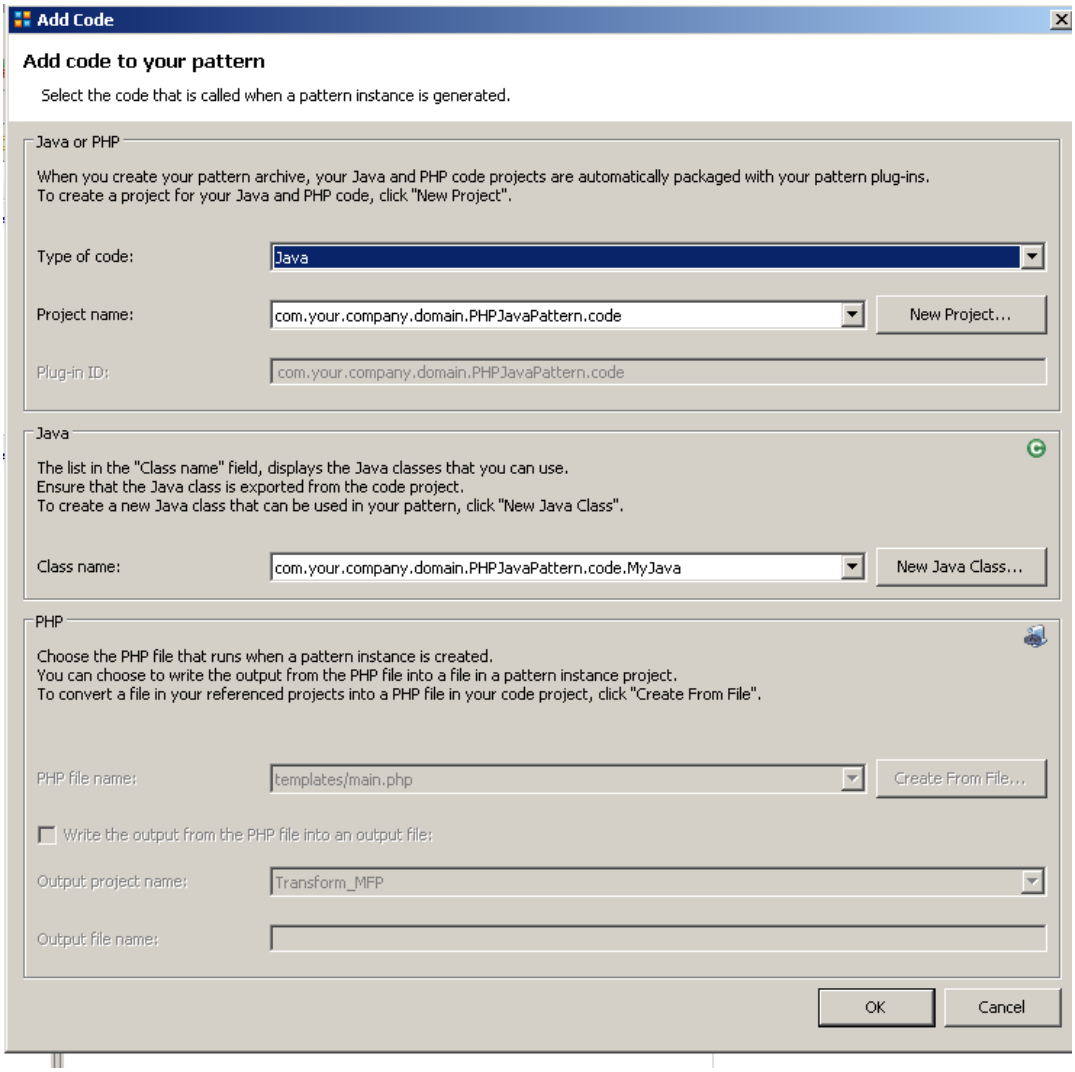
   Save and close the PHP file.

9. Now we're going to add a new Compute node into the flow. This node will use the ESQL file we just created. To add a new node, we will use a small amount of java code.

   In the Pattern Configuration, click Add to add a new Java item.



10. We will use the same project that we created earlier, so just click OK.



11. In the code project, expand all items under the src directory, and open MyJava.java.

    Replace the entire contents of this file with the entire contents of c:\student\PatternAuth\resources\loggingNode.java.txt.

    Take a few minutes to look at the key code statements, which is well commented.

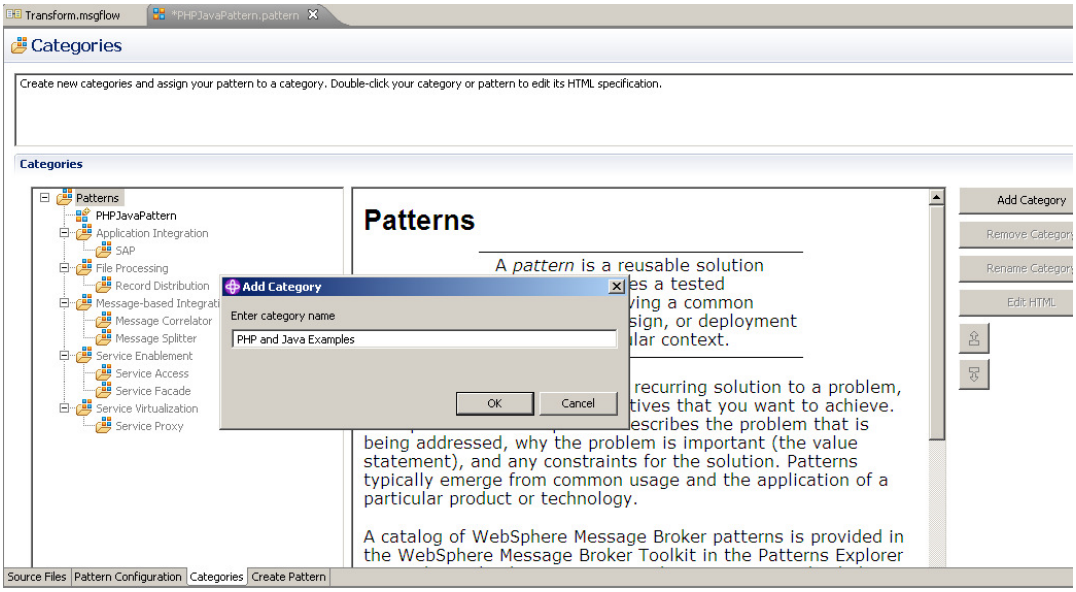
    This java code uses the new Message Flow API, which allows you to manipulate a message flow, add/remove nodes and connectors, and set node properties.

    Save and close Myjava.java.

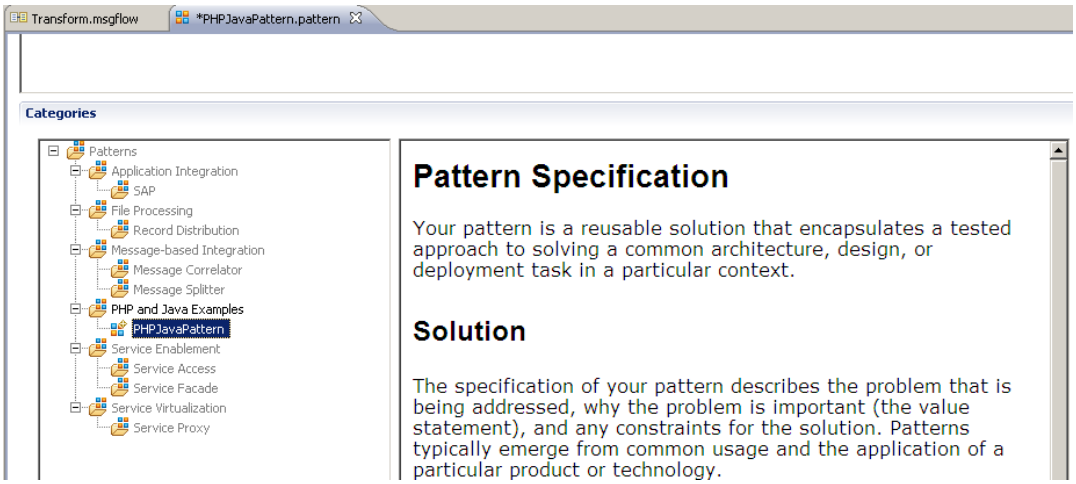
```
package com.your.company.domain.PHPJavaPattern.code;

import com.ibm.broker.config.appdev.ESQLModule;
import com.ibm.broker.config.appdev.MessageFlow;
import com.ibm.broker.config.appdev.nodes.ComputeNode;
import com.ibm.broker.config.appdev.patterns.GeneratePatternInstanceTransform;
import com.ibm.broker.config.appdev.patterns.PatternInstanceManager;

public class MyJava implements GeneratePatternInstanceTransform {

	@Override
	public void onGeneratePatternInstance(PatternInstanceManager patternInstanceManager) {
	// Get the name of the message flow
		MessageFlow mf1 = patternInstanceManager.getMessageFlow("Transform_MFP",
					"mqsi/Transform.msgflow");

	// Get the value of the loggingrequired pattern parameter
		String loggingNeeded = patternInstanceManager.getParameterValue("loggingRequired");

	// Has database logging been selected by the pattern user?
		if (loggingNeeded.equals("true") == true) {

		// Specify the schema and file name for the associated ESQL file (created in our PHP script)

			ESQLModule myEsqlModule = new ESQLModule();
			myEsqlModule.setBrokerSchema("mqsi");
			myEsqlModule.setEsqlMain("LogToDatabase");

		// Create a new Compute node called LogToDatabase, and specify its position on the flow editor
			ComputeNode loggingNode = new ComputeNode();
			loggingNode.setNodeName("LogToDatabase");
			loggingNode.setLocation(400,50);

		// Obtain the value of the pattern parameter for the logging database, and use this to
		// set the database source property on the new Logging node.
			String loggingDB = patternInstanceManager.getParameterValue("loggingDatabase");
			loggingNode.setDataSource(loggingDB);

		// Set the ESQL file name property on the new Logging node
			loggingNode.setComputeExpression(myEsqlModule);

		// Add the new node to the flow
			mf1.addNode(loggingNode);

		// Get a handle on the existing node in the flow, called "CreateOutput", and
		// create a connector from this node to our new Logging node.
				ComputeNode createoutputNode = (ComputeNode)
					mf1.getNodeByName("CreateOutput");
			mf1.connect(createoutputNode.OUTPUT_TERMINAL_OUT,
					loggingNode.INPUT_TERMINAL_IN);

		}
	}
}
```

2.3 Create a new Pattern Category

1. Now switch to the Categories tab. Create a new category (Add Category), and name it “PHP and Java Examples”.

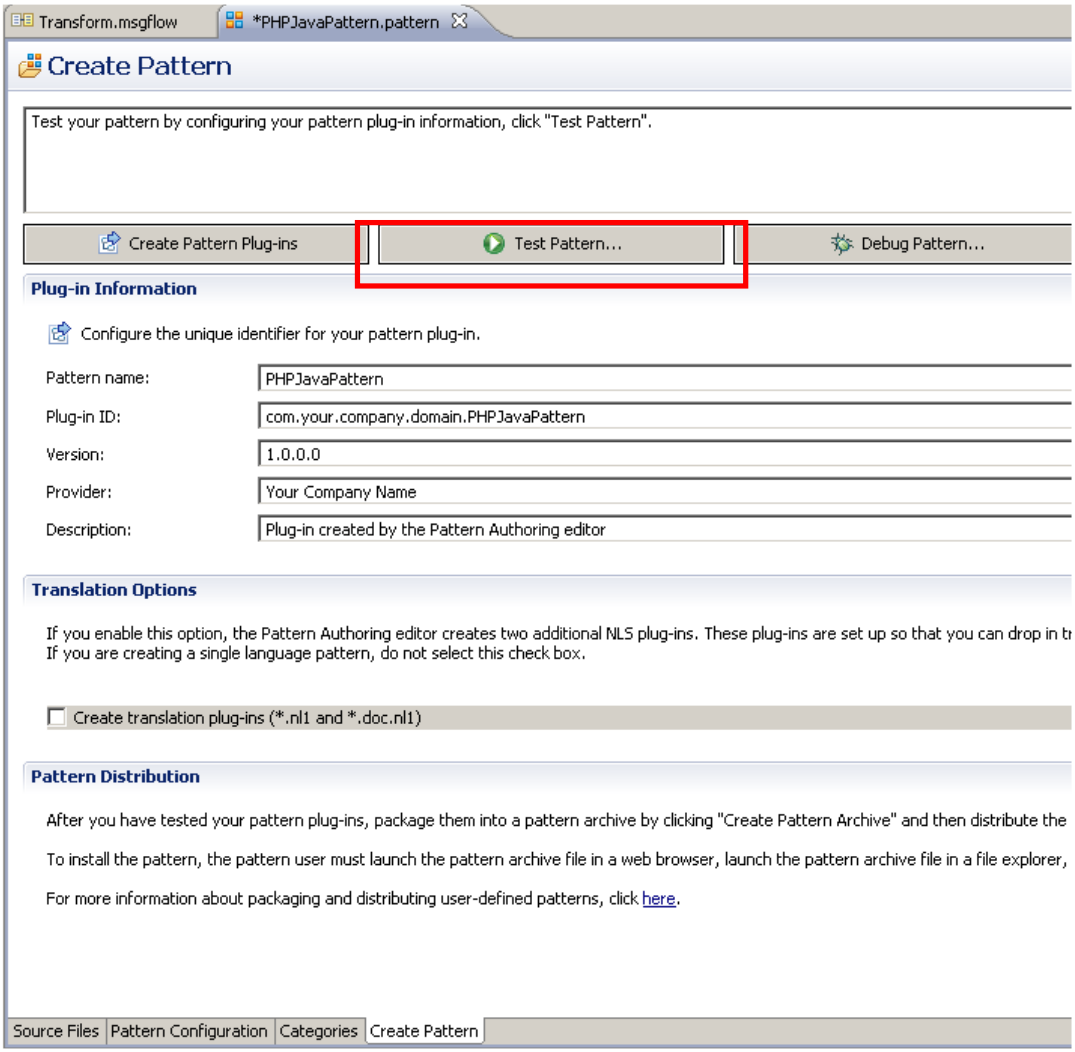


2. Drag the new pattern onto the new category.

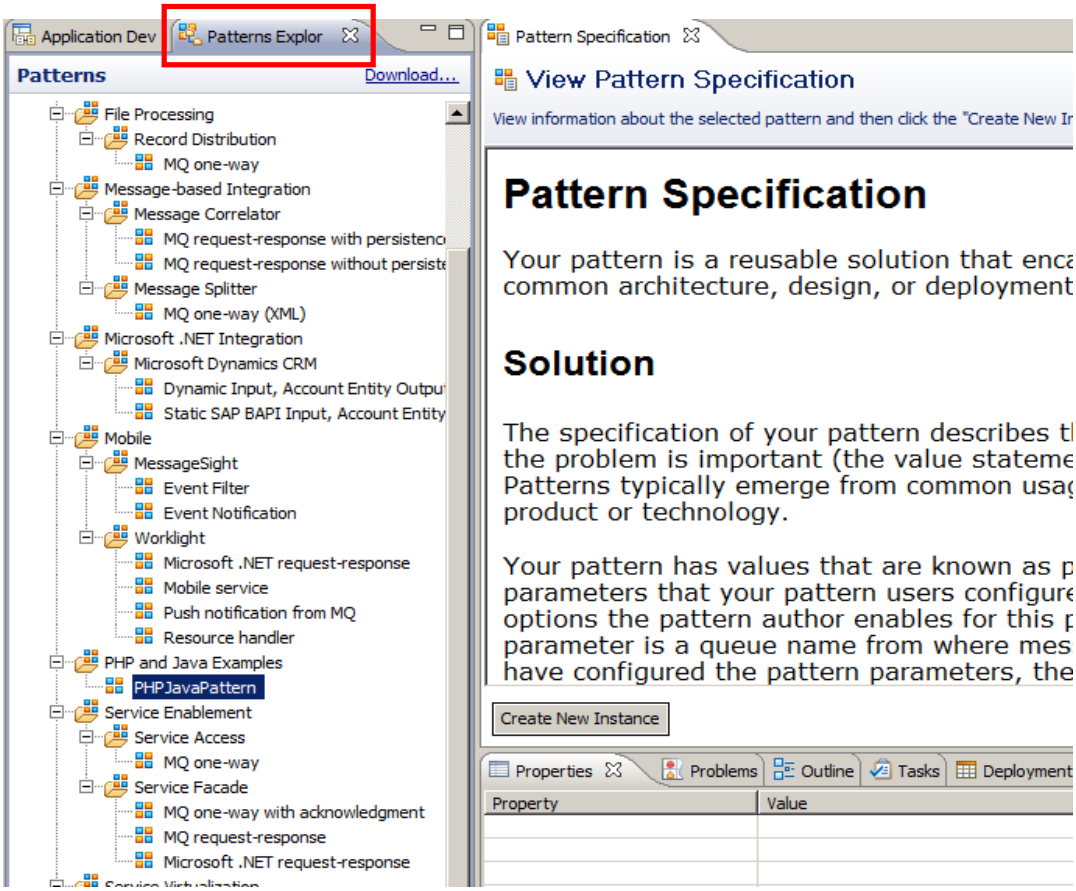


2.4 Test the Pattern

1. Switch to the “Create Pattern” tab, and click Test Pattern.



2. When the new workbench starts, select the Patterns Explorer, and click the PHPJavaPattern entry.

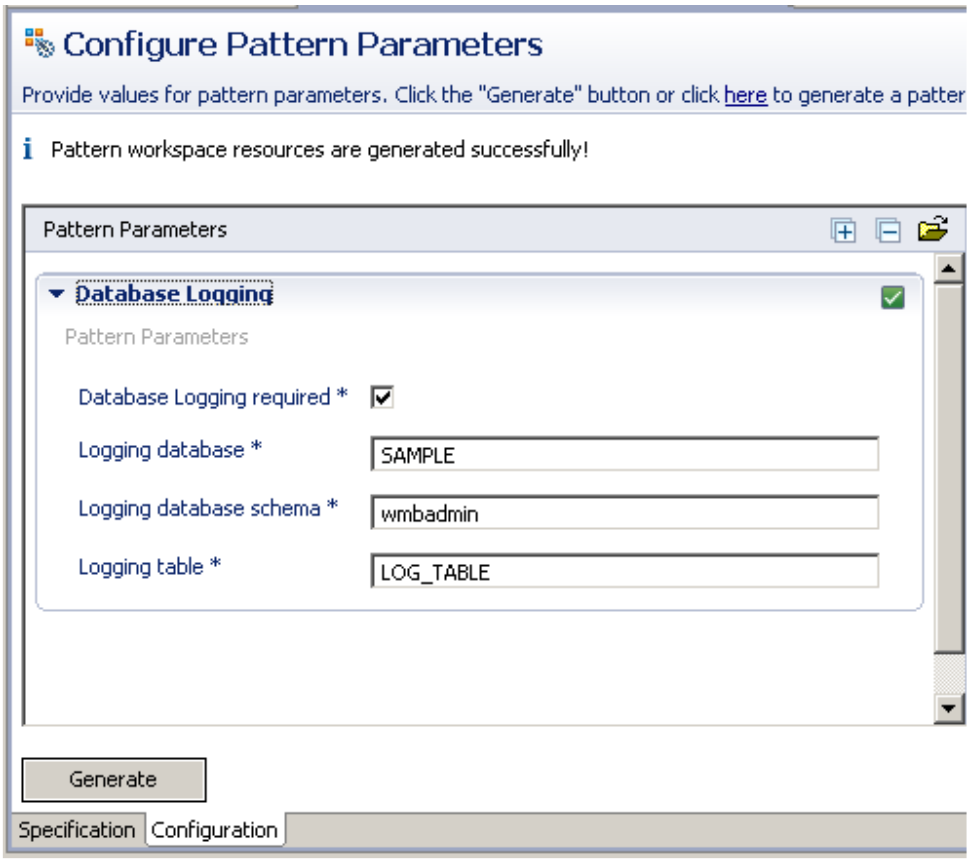


3. Click Create New Instance, provide a pattern instance name, and click OK.

4. Expand the “Database Logging” group, and observe the parameters and default values that you have just created.

   You can change these values if you want (but leave the Database Logging Required box checked).

   Click Generate.

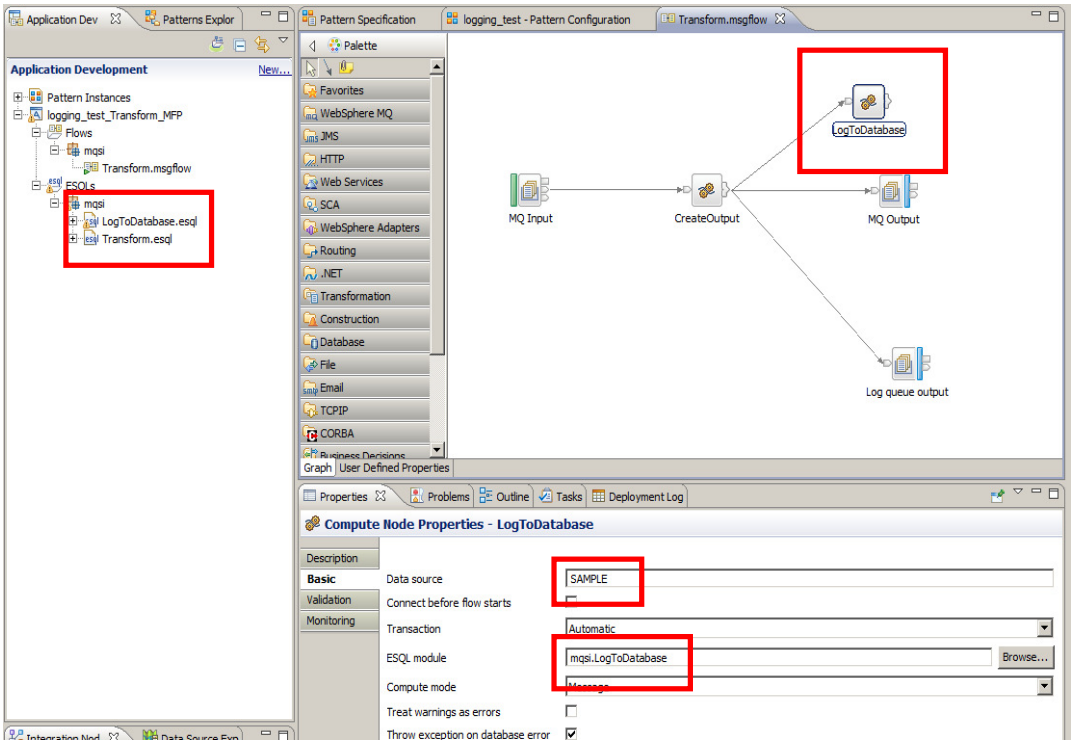


5. A new message flow will be generated (this example was generated with a pattern instance name of Test1).

   Observe that a new node, LogToDatabase, has been added to the flow. Click this node and see that the corresponding ESQL module has been set to mqsi.LogToDatabase.

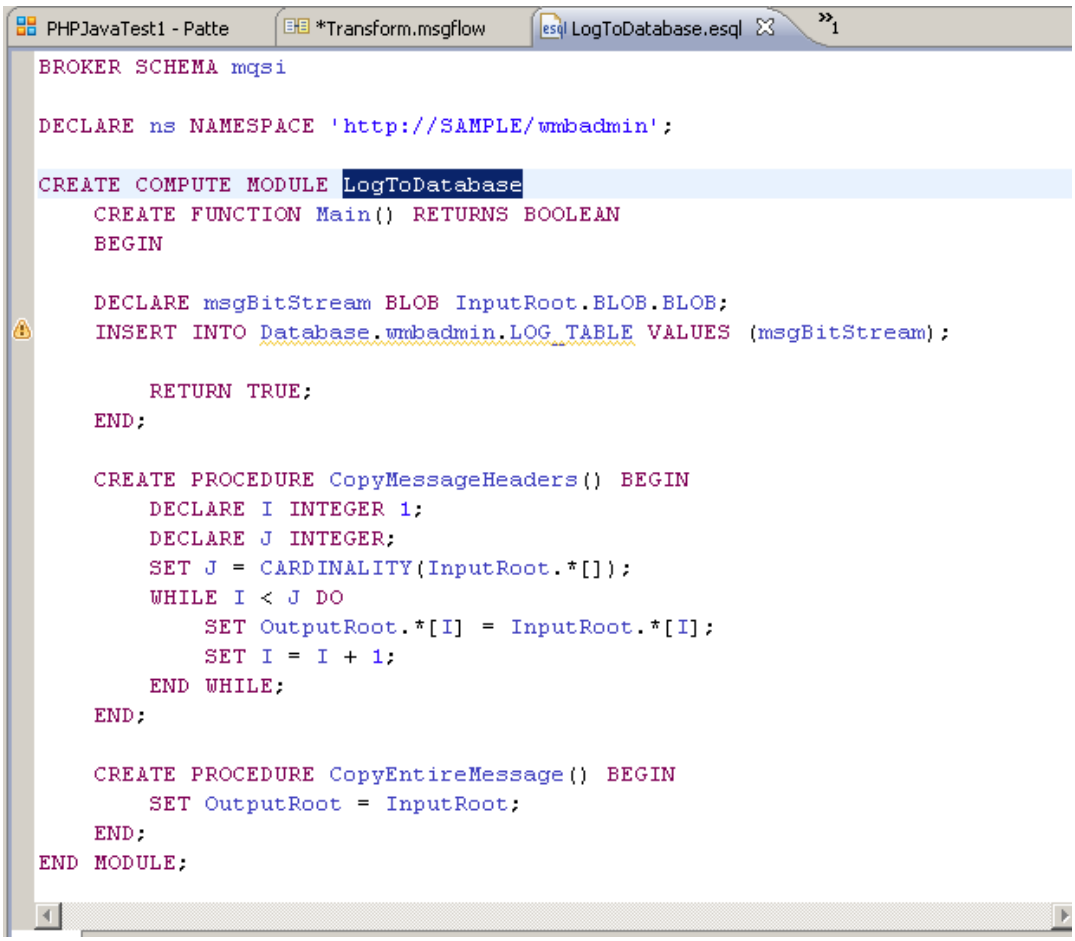
   Observe that a new ESQL file, LogToDatabase.esql, has been created in the mqsi schema, in the project navigator.

   Observe the new node Data source name = SAMPLE.



6. Open the ESQL file (double-click either the new Compute node, or the ESQL file in the project navigator).

   Observe that several variables have been set to the values specified by the pattern instance.



   Close the ESQL file, and the message flow.

7. Now we will create a new message flow, where we do not require database logging.

   In the pattern instance, uncheck the “Database Logging Required” tickbox.

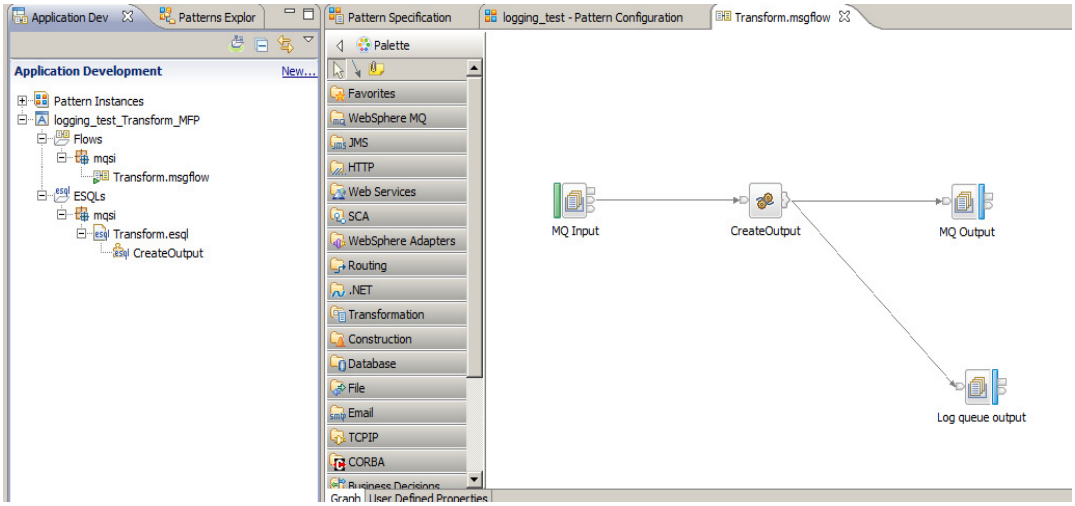
   Note – we could have built the pattern to disable (grey out) the database fields if this option is deselected. We will leave that as an exercise for the reader – see the earlier patterns labs for details.

   Click Generate.

8. Open the new message flow.

   Observe that the flow does not contain a database logging node.

   Observe that the ESQL section only contains ESQL for the existing Compute node.



2.5 Using PHP to generate MQSC and DDL scripts

Now we'll go and add a couple of other types of scripts to the pattern. In this section, we'll create an MQSC script.

Close the test workspace, and return to the Pattern Editor.

1. First, we need to add some MQ queue names as pattern parameters.

   You should know how to do this by now, so no screen captures.

   In the Message Flow editor for Transform.msgflow, right-click each of the three MQ nodes in turn, and in each case select the queue name as a pattern parameter (don't select the queue manager name on the MQ output nodes by accident).

   Save the flow.

   In the PHPJavaPattern, select Pattern Configuration. You will see the new pattern parameters added to the list.

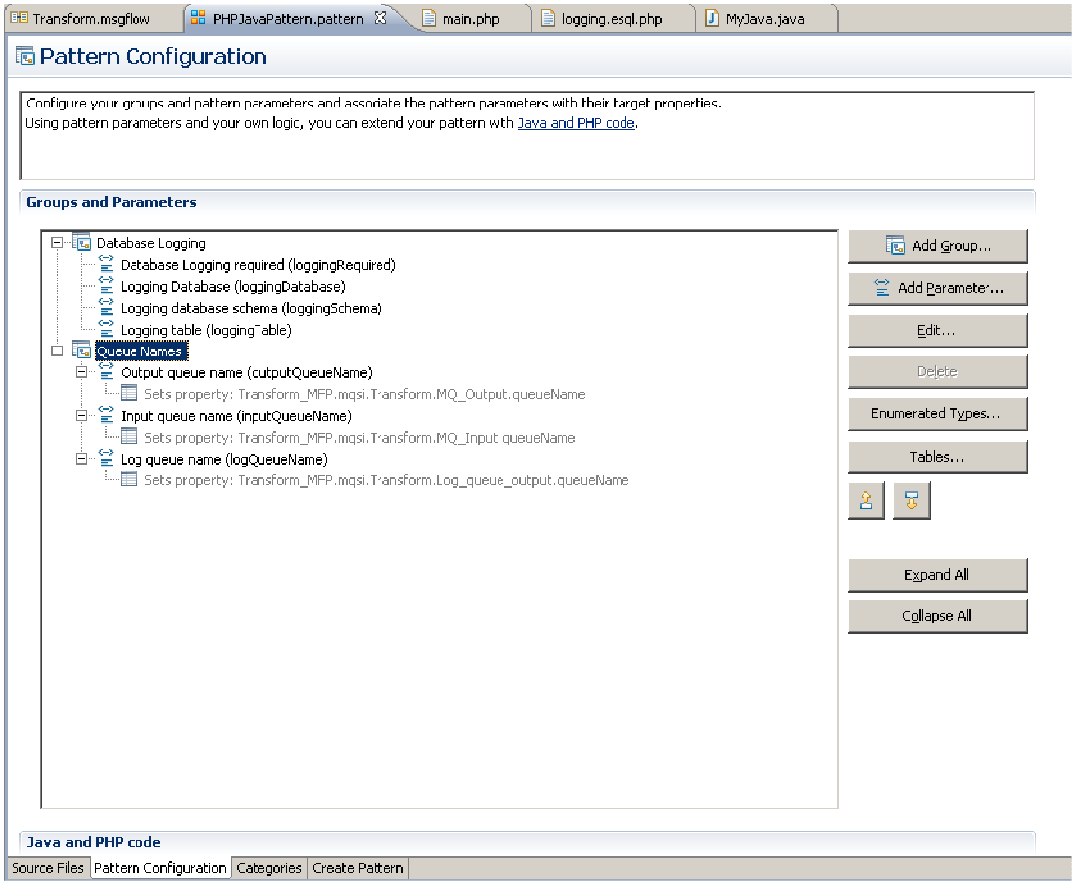
   Add a new group, and give it a title "Queue Names". Drag each of the MQ queue name parameters into this group (drag the items named "Queue Name"). Note that the pattern editor has automatically created new individual groups for each of the MQ nodes (named MQ_Input, etc). You can now safely discard these (highlight each, click Delete).

   For each of the queue name parameters, click Edit, and set the following values:
   MQ_Input node
   Display name = Input queue name
   Parameter ID = inputQueueName (case sensitive)
   Default value (on the Editor tab) = INPUT.QUEUE

   MQ_Output node
   Display name = Output queue name
   Parameter ID = outputQueueName
   Default value = OUTPUT.QUEUE

   MQ_Log_output
   Display name = Log queue name
   Parameter ID = logQueueName
   Default value = LOG.QUEUE

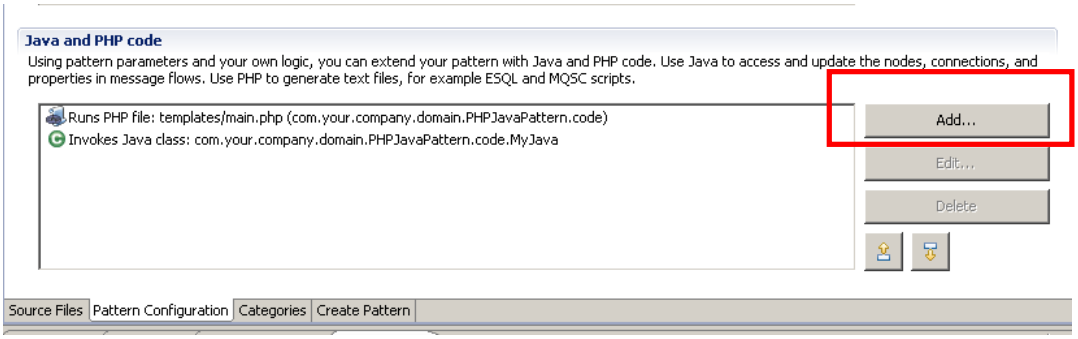
2. At this point, the pattern configuration should look like this:



Save the pattern configuration.

3. Still in Pattern Configuration, move the window down to the “Java and PHP” section.

Click Add to add a new PHP script.



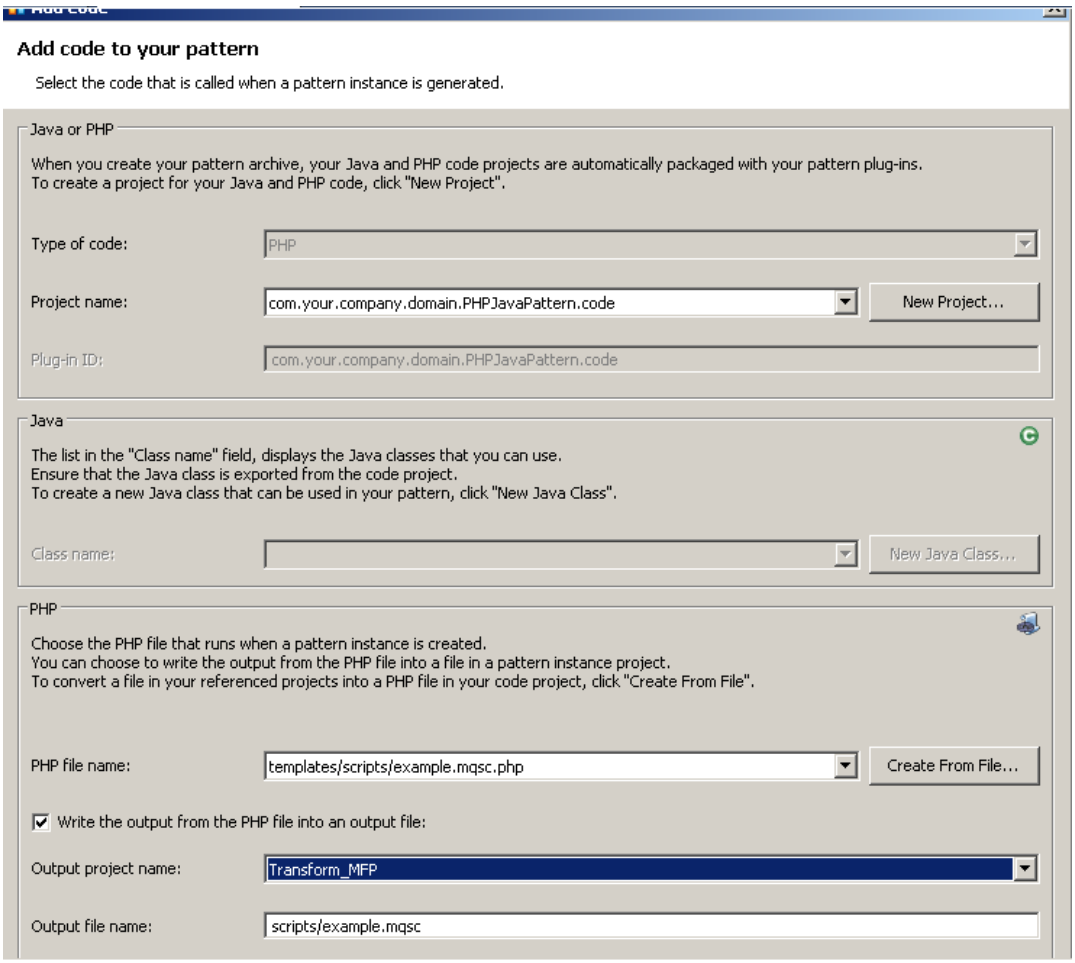
4. Set “Type of code” = PHP.

   Set “PHP file name” = templates/scripts/example.mqsc.php (one of the drop-down options).

   Select “Write output from PHP to an output file”.

   Set “Output project name” = Transform_MFP.

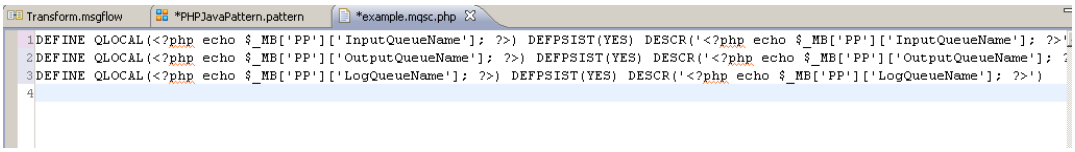
   Click OK.



5. Open the file templates/scripts/example.mqsc.php.

   Replace the entire contents of the file with the contents of c:\student\PatternAuth\resources\Create_mqsc.php.txt.

   Note that the queue names in this script have been set to the values of the queue name pattern parameters. (It's important that you typed the parameter IDs correctly when you added the queue parameters earlier).



   Save and close.

6. Now we will add a PHP script to create the database DDL statements to create the database logging table. We only want to run this script if "Database Logging" is selected by the pattern user.

   Open the file main.php.

   Inside the "if" clause, add a second line, which will invoke the CreateLT.ddl.php script file (we'll create this in the next step).

```
mb_pattern_run_template("Transform_MFP", "mqsi/CreateLT.ddl.php",
"mqsi/CreateLT.ddl");
```

   (To avoid mistakes, you can copy from the file c:\student\PatternAuth\resources\main.php.line2.txt).

7. Under the templates/mqsi directory, import the file CreateLT.ddl.php. (Right-click, Import, General, File System, C:\student\PatternAuth\resources\CreateLT.ddl.php.

   Open the newly imported file and observe that several fields have been set to PHP variables, matching the pattern parameters you created earlier.

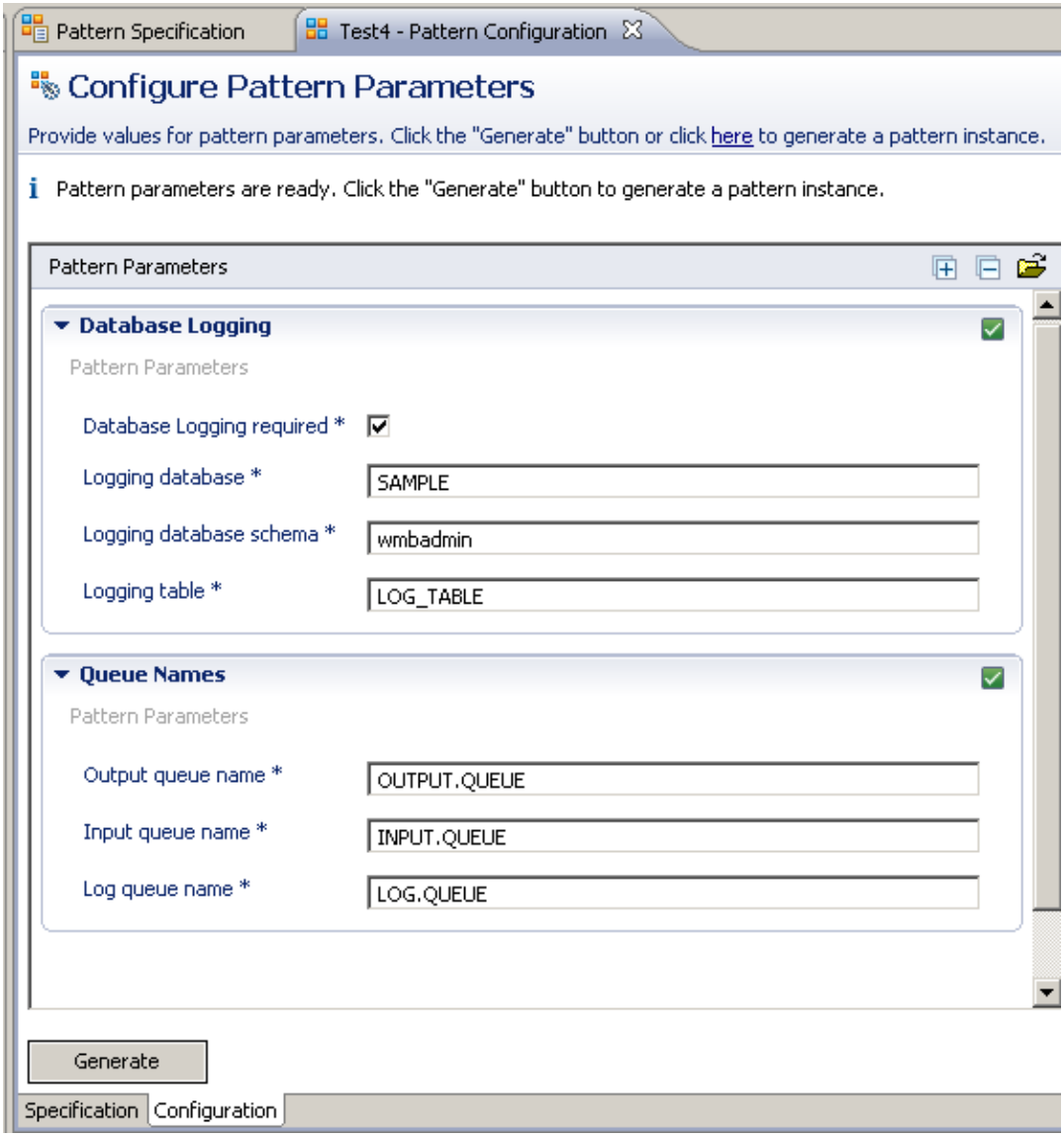
8. Now we're done.

   Save the pattern, then click Create Pattern, and Test Pattern to test the updated pattern in the usual way.

9. Create a new pattern instance. This example is named Test4.

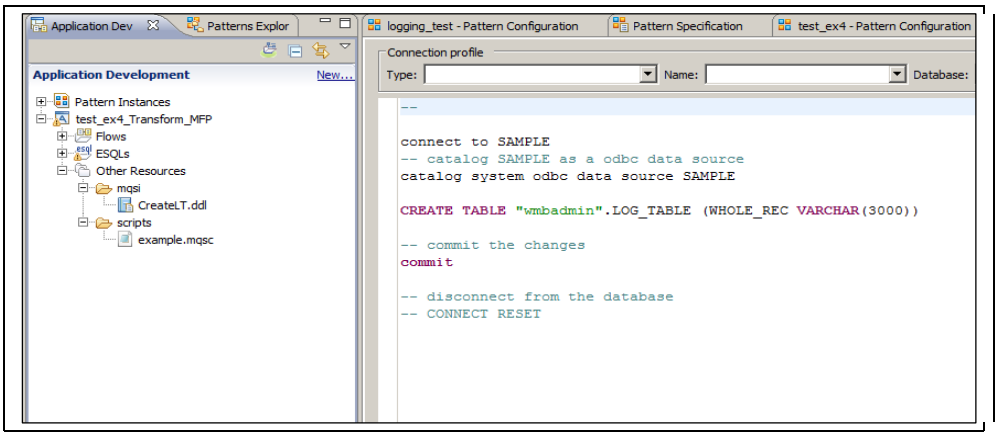
   You can override the default values if you want.

   Click Generate.



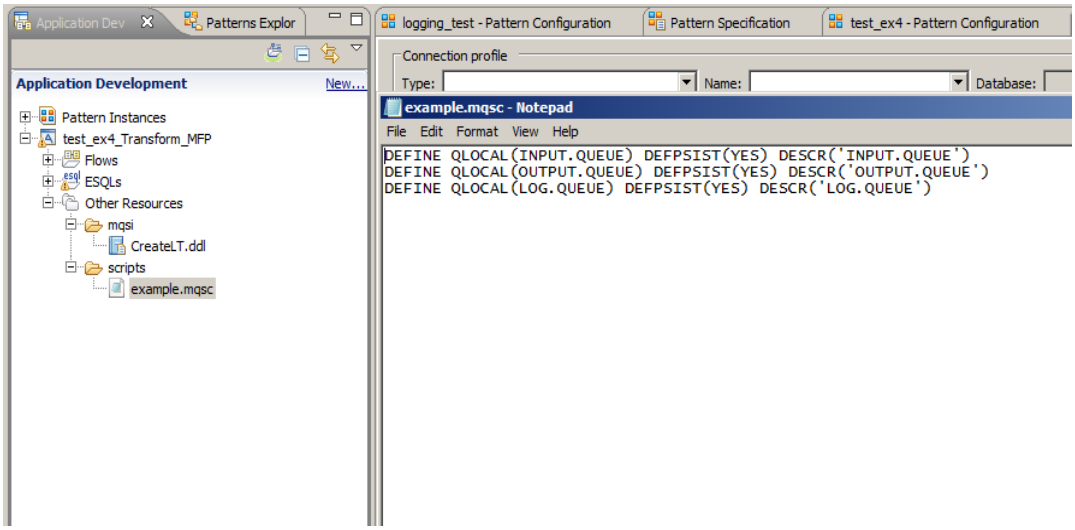
10. Open the scripts/CreateLT.ddl file.

    Observe the database, schema and table names have been set to those specified in the pattern instance.



11. Open the scripts/example.mqsc file.

    Observe the queue names (and descriptions) have been set to those specified in the pattern instance.



This concludes the Pattern Authoring PHP and Java lab.