

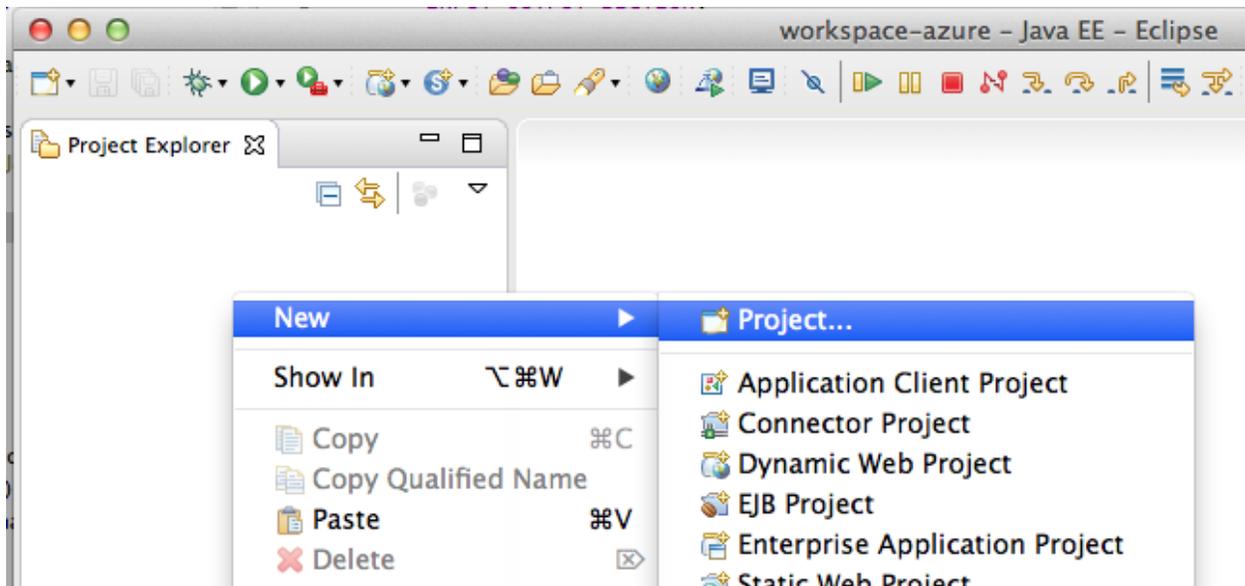
2. Building the sample EBP project for Azure

You can view an example of the steps required in video form here <https://youtu.be/ouCPWu4QkBs> and here <https://youtu.be/RmbbWhSpDZY>

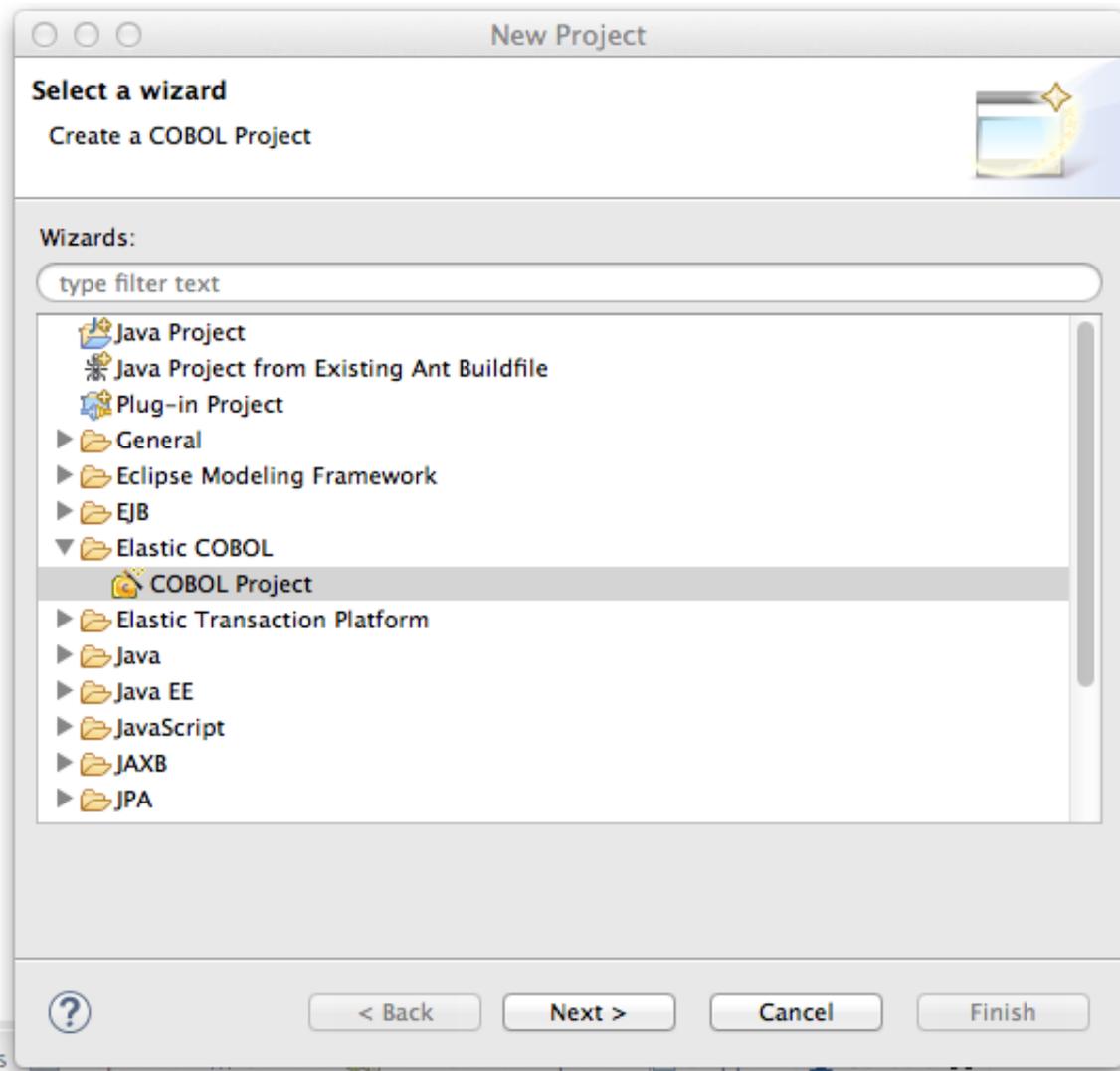
Download and install Eclipse for J2EE from here : <http://www.eclipse.org/downloads/>
Log into paas.heirloomcomputing.com and download our plugin, following instructions there for installation and license placement

Download Sample Source bundle from here:
Open Eclipse

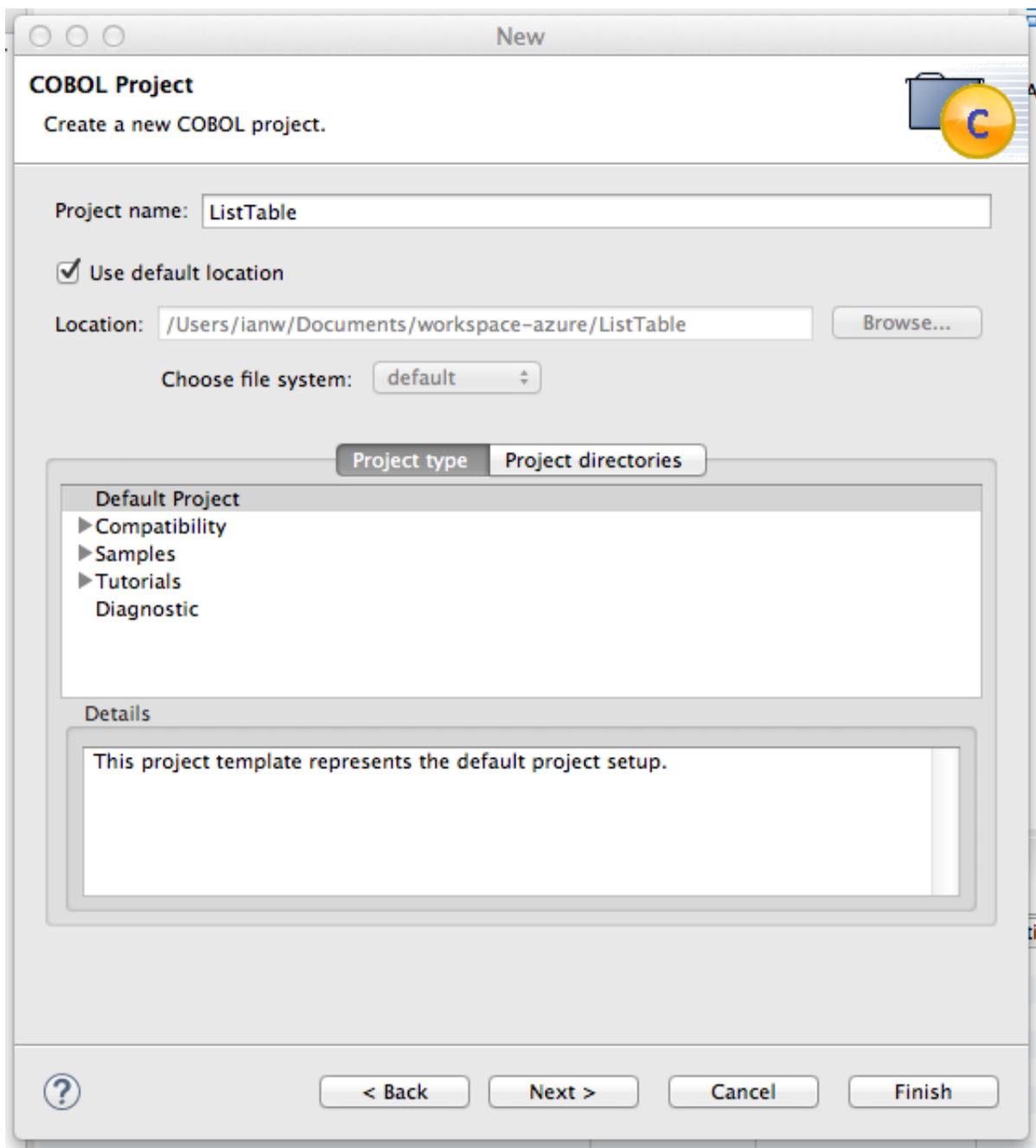
Right click the Project Explorer and select New->Project... :



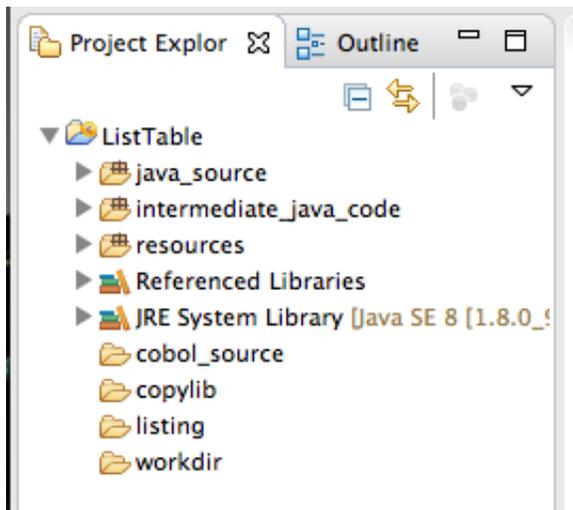
Choose COBOL Project and click Next:



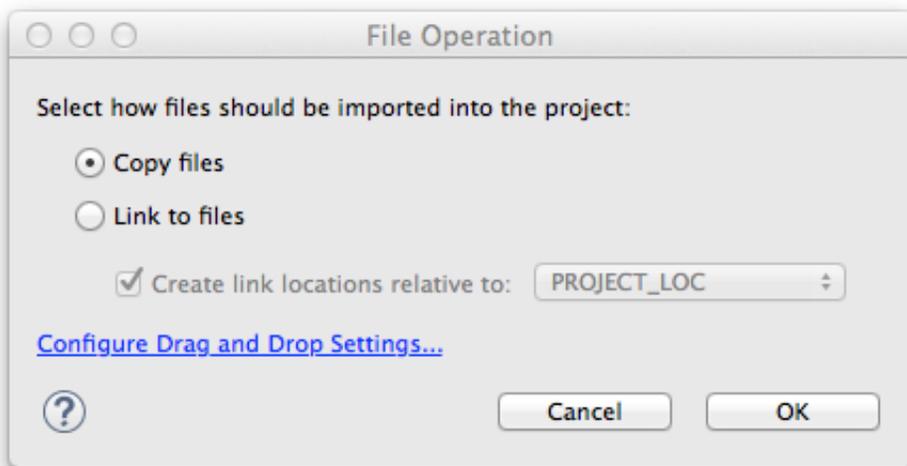
Name the project 'ListTable' , select 'Default Project' under Project Type and click Finish:



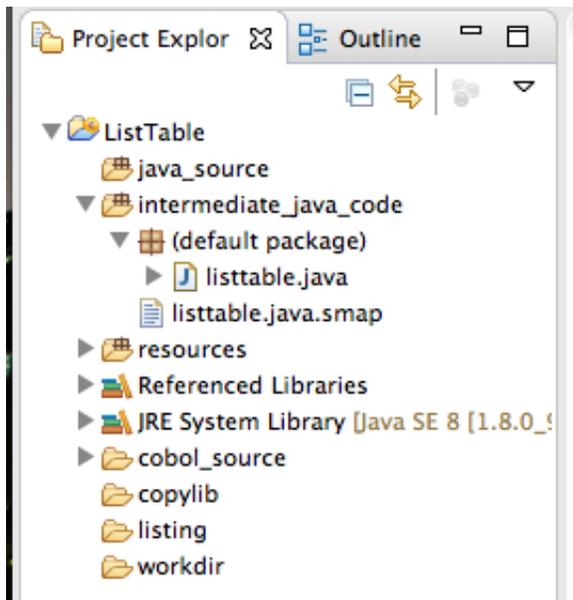
Expand the ListTable project:



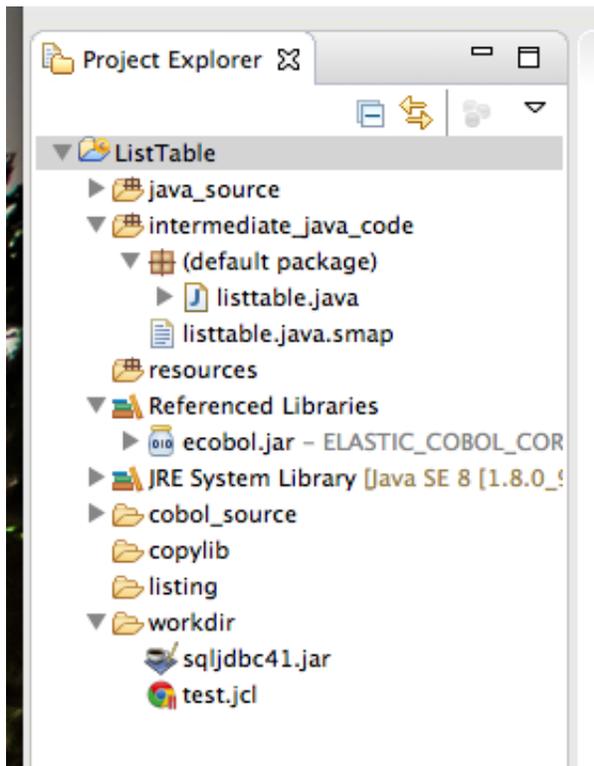
From your source bundle drag the ListTable.cbl file into the cobol_source folder and choose to 'Copy files' and click OK:



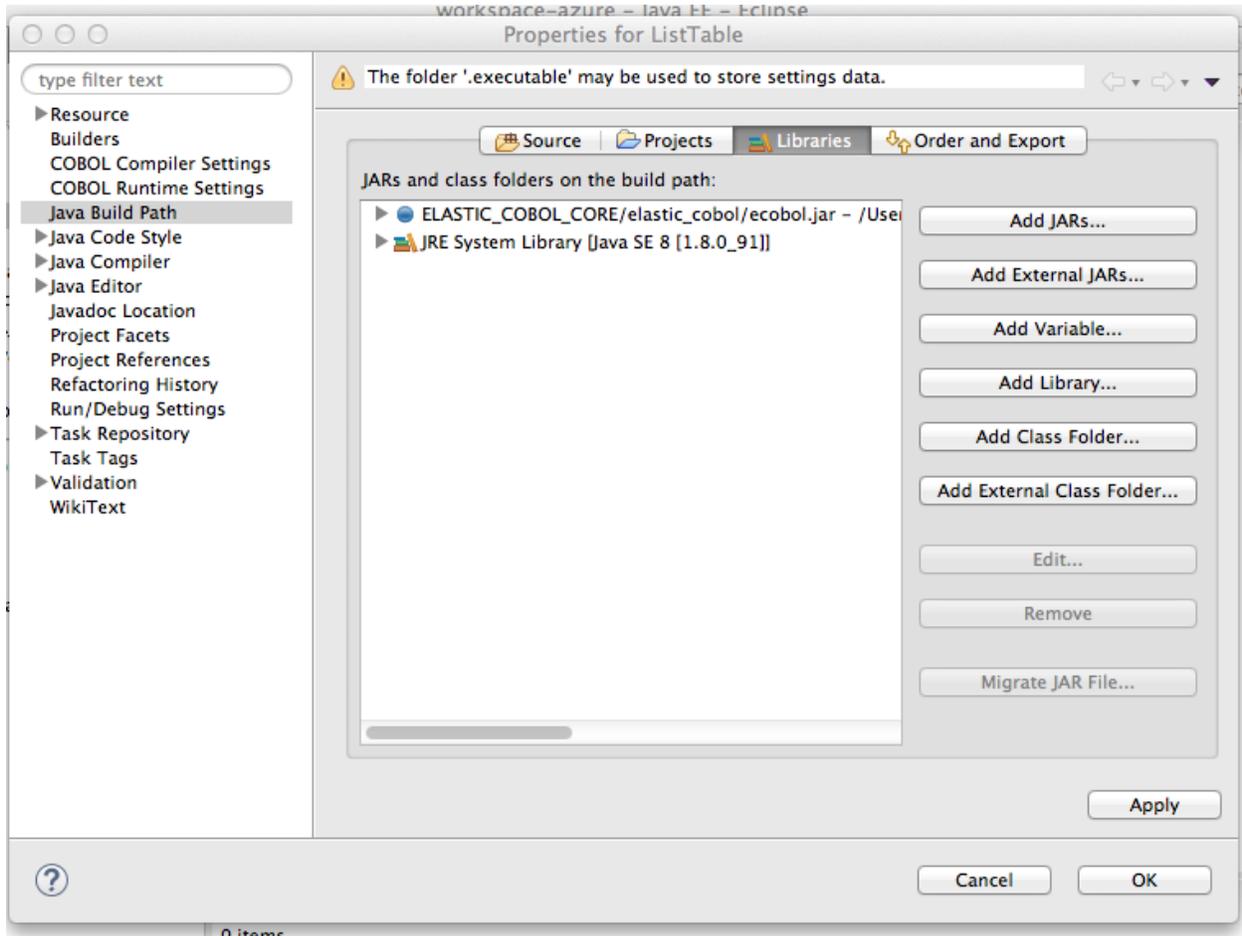
Expand the intermediate_java_code folder and note that Java has been generated from the COBOL:



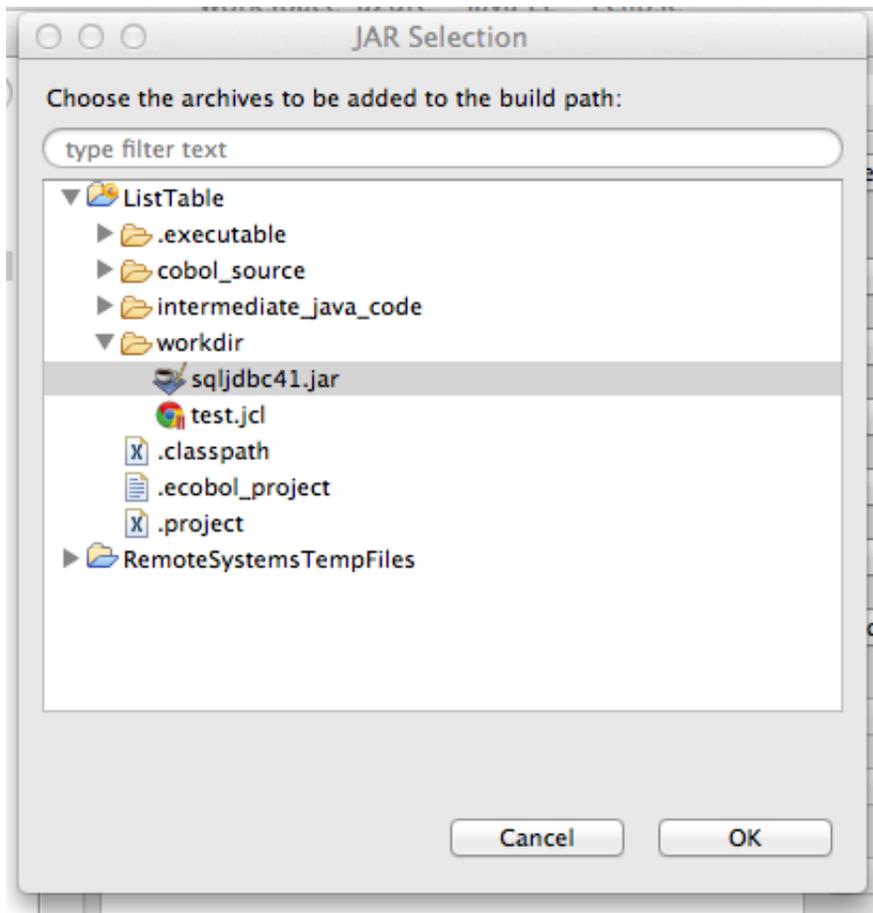
Drag the file test.jcl into the workdir folder and sqljdbc41.jar into the workdir folder, again choosing to 'Copy files' :



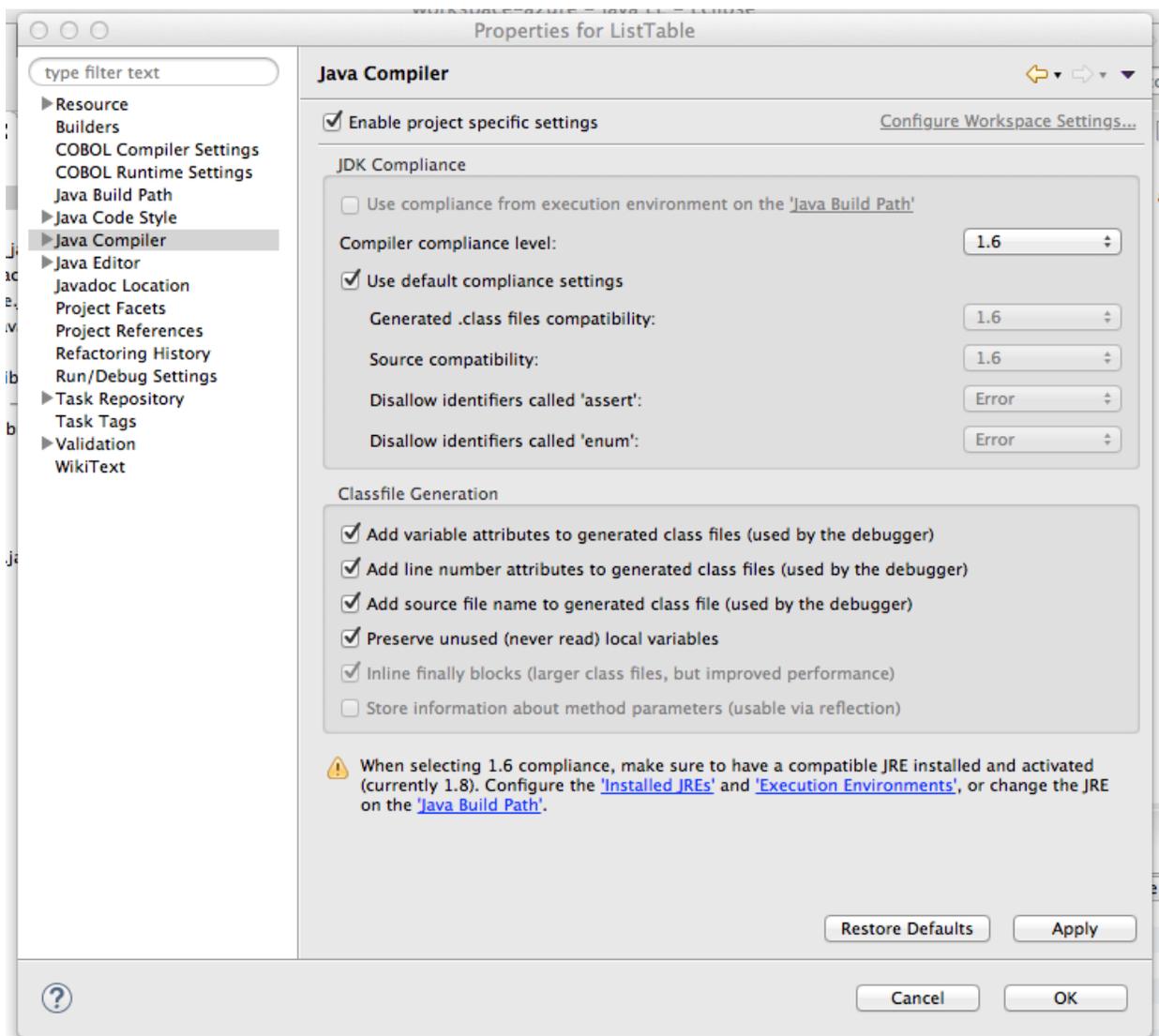
Right click the ListTable Project and choose properties, then click 'Java Build Path':



Click 'Add JARS' and expand the ListTable project to the workdir, select sqljdbc41.jar and click OK:



Click 'Java Compiler', check 'Enable project specific settings' and choose 1.6 and the Compiler compliance level:



Click OK, then click Yes to perform a project rebuild.

Open the cobol_source folder , double click ListTable.cbl to bring up the source code. Edit the jdbc driver information with the jdbc connection string you copied from the Azure SQL database properties. Remember to add in the user and password:

The screenshot shows the Eclipse IDE interface. The Project Explorer on the left displays a project named 'ListTable' with the following structure:

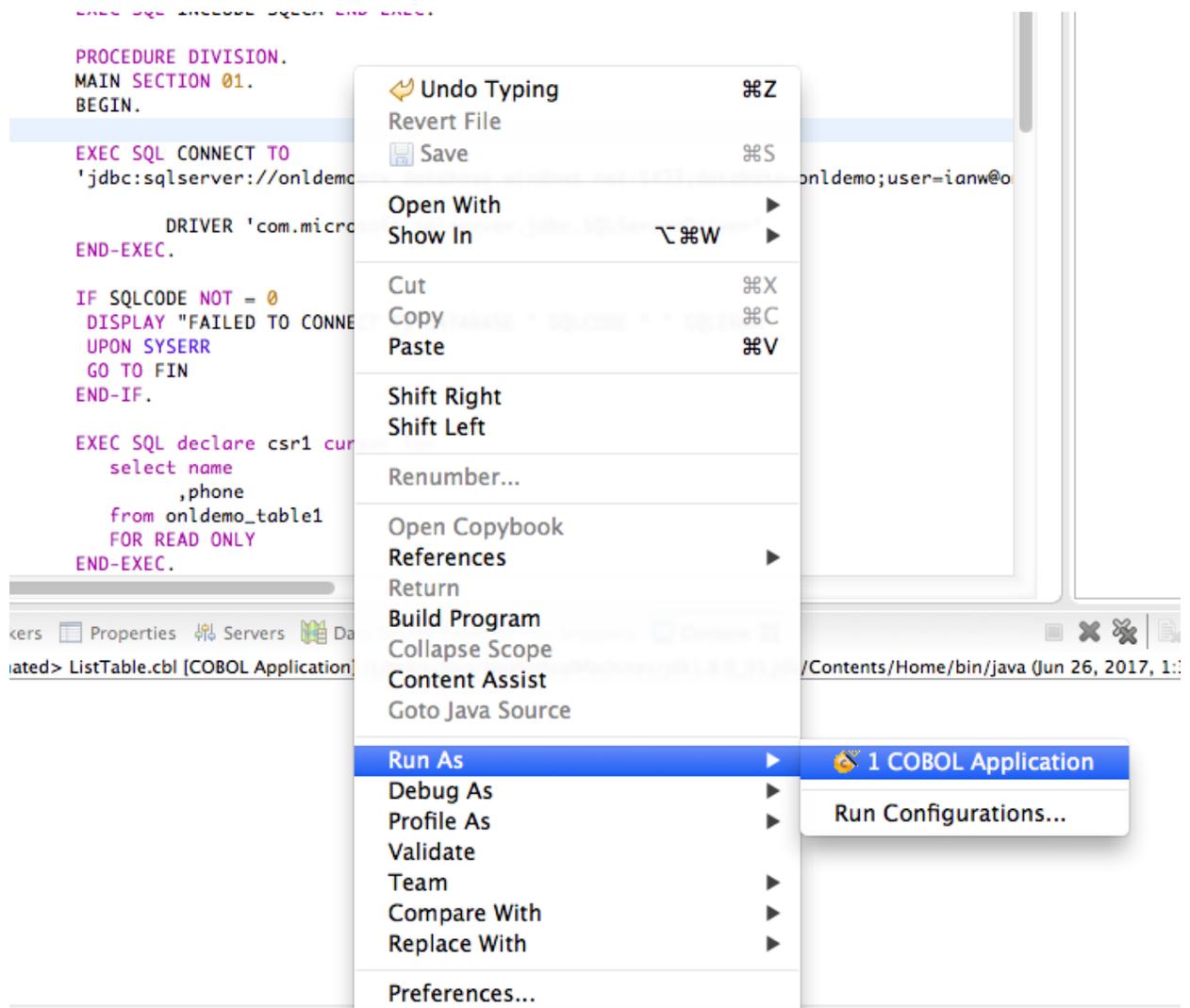
- java_source
- intermediate_java_code
 - (default package)
 - listtable.java
 - listtable.java.smap
- resources
- Referenced Libraries
 - ecobol.jar - ELASTIC_COBOL_CORE/elastic_cobol
 - sqljdbc41.jar
- JRE System Library [Java SE 8 [1.8.0_91]]
- cobol_source
 - ListTable.cbl
 - copylib
 - listing
- workdir
 - sqljdbc41.jar
 - test.jcl

The main editor window shows the source code for 'ListTable.cbl' with the following content:

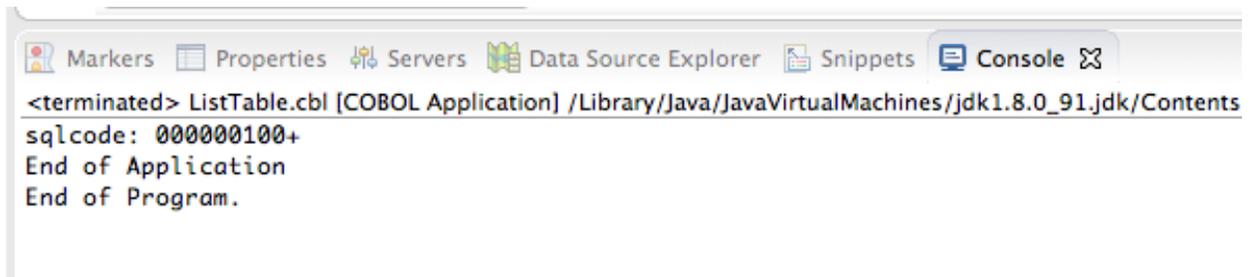
```
1  $set sourceformat "free"
2  IDENTIFICATION DIVISION.
3  PROGRAM-ID. ListTable.
4  ENVIRONMENT DIVISION.
5  INPUT-OUTPUT SECTION.
6  DATA DIVISION.
7  WORKING-STORAGE SECTION.
8  01 emp-group-item.
9  05 uname          pic x(16).
10 05 phone          PIC 9(9).
11 01 cursor-switch pic x value 'n'.
12 08 end-of-cursor value 'y'.
13
14 EXEC SQL INCLUDE SQLCA END-EXEC.
15
16 PROCEDURE DIVISION.
17 MAIN SECTION 01.
18 BEGIN.
19
20 EXEC SQL CONNECT TO
21 'jdbc:sqlserver://ondemosrv.database.windows.net:1433;database=onldemo;user=ian@o
22
23         DRIVER 'com.microsoft.sqlserver.jdbc.SQLServerDriver'
24 END-EXEC.
25
26 IF SQLCODE NOT = 0
27 DISPLAY "FAILED TO CONNECT TO DATABASE " SQLCODE " " SQLERRM
28 UPON SYSERR
29 GO TO FIN
30 END-IF.
31
32 EXEC SQL declare csr1 cursor for
```

Save ListTable.cbl.

Right click the source code and choose 'Run As->1 COBOL Application' :

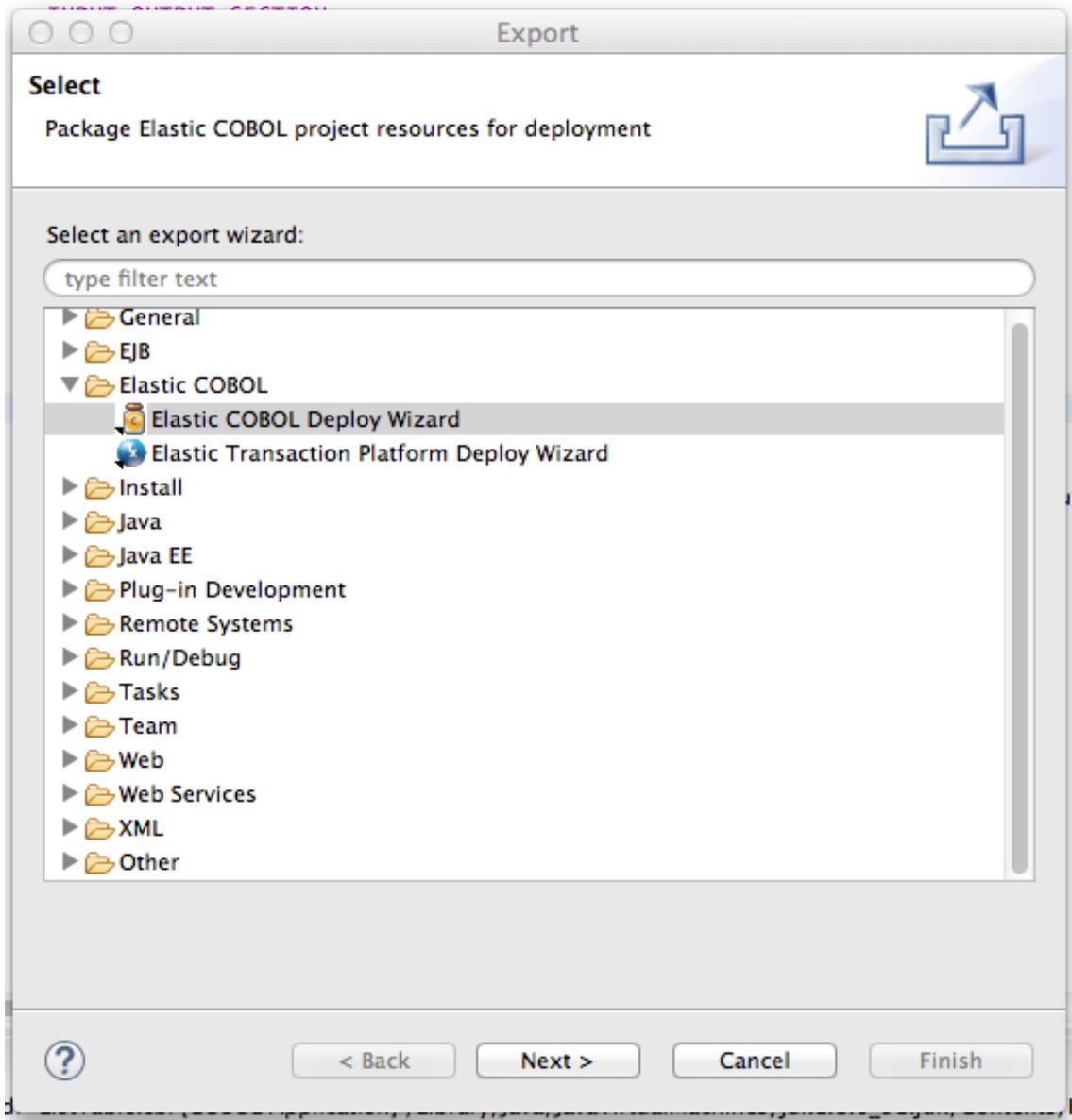


If you have set up the database and table you should see a SQL Code 100 in the console window:



This confirms the program will work correctly after it is exported and run under EBP.

To export this application so it is ready to deploy to EBP later on, right click the ListTable project and choose 'Export':



Choose Elastic COBOL Deploy Wizard and click Next:

COBOL Deploy Wizard

Archive Package Specification

Define what resources to package

Resources to export:

- [ListTable] listtable\$Wrk.class
- [ListTable] listtable.class
- [ListTable] listtable.java.smap

Add...
Remove
▲
▼

Deployment Options

Traditional Cloud

Select the export destination:

JAR or WAR file: Browse...

Specify the deployment license location:

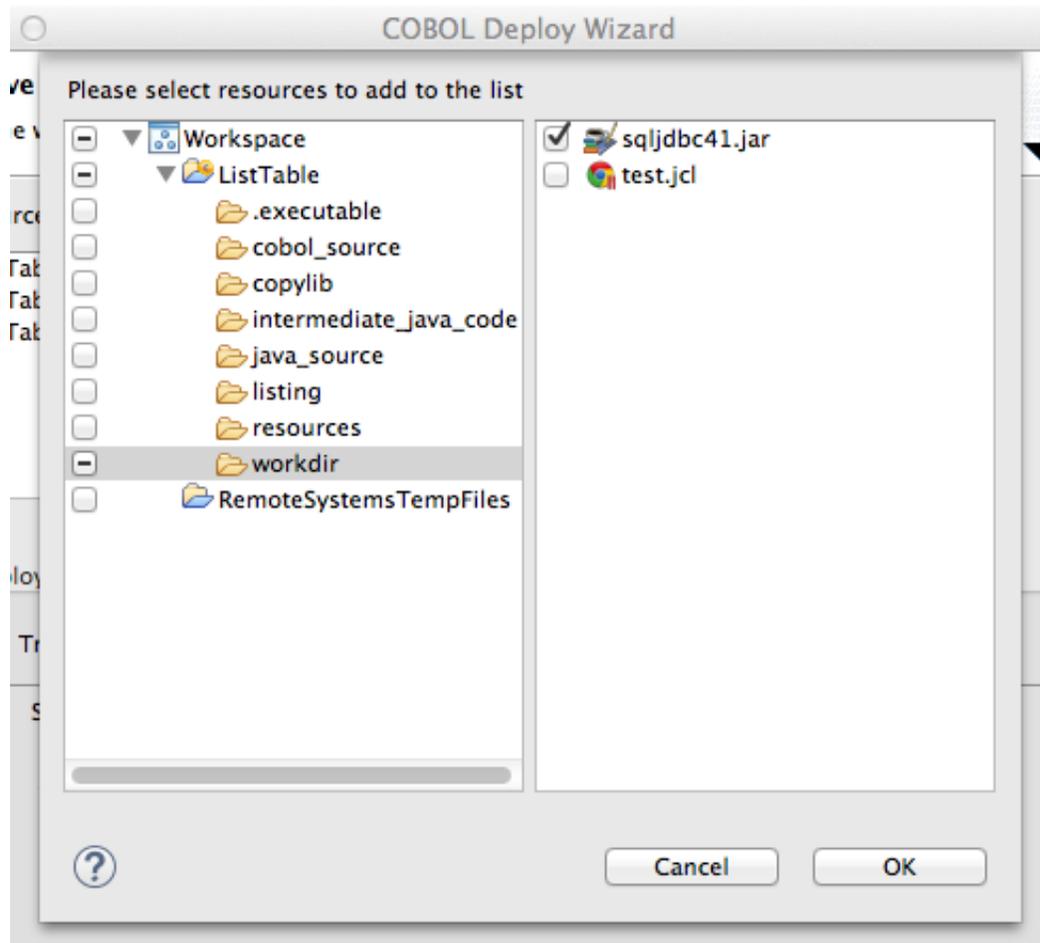
Browse...

Options:

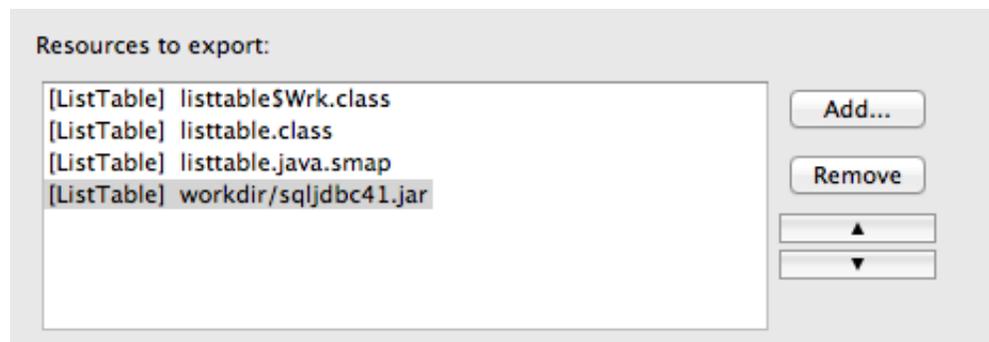
- Compress the contents of the archive
- Overwrite existing files without warning

? < Back Next > Cancel Finish

File in the name of the file to export (seen here is /Users/ianw/ListTable.jar) and click 'Add..' at the top to add the SQL driver to the export.
Open the workspace/project and workdir folder and select the sqljdbc41.jar, then click OK:



This will add the driver to the export list:



Click Next:

COBOL Deploy Wizard

Archive Packaging Options

Define required runtime elements and other options

Select runtime elements required by application:

<input checked="" type="checkbox"/> Elastic COBOL Runtime Elements	<input checked="" type="checkbox"/> Core Runtime Elements (Required)
	<input checked="" type="checkbox"/> Printing API
	<input checked="" type="checkbox"/> File Protocol Support
	<input checked="" type="checkbox"/> Graphical Screen Section Support
	<input checked="" type="checkbox"/> JCurse Support (Platform Depend
	<input checked="" type="checkbox"/> XML Support
	<input checked="" type="checkbox"/> Environment Support

Include the selected runtime elements within the export archive

Select the settings to use for launching the application:

Run Configuration:

Select the class of the application entry point:

Main Class:

Other Options (JAR only):

- Copy JAR dependencies from the Elastic COBOL directory to the export directory
- Generate script-based application launcher stub
- Generate HTML-based application launcher stub
 - Generate WAR package from JAR and HTML stub
 - Use Java plugin

Click Finish, the jar file will be created.

Minimize the ListTable project and close the ListTable.cbl source file:

