Hanseatic Mainframe Summit

z/OS DB2 access using JAVA

Lab Version V2.00

Tuesday, 14 September, 2010

Overview

This lab describes how to connect to a DB2 running on z/OS and how to run SQL statements with Java.

Lab Requirements

Eclipse, Network access to a DB2 z/OS

Lab Steps

Summary of the steps

Part 1: JDBC program accessing DB2 for z/OS data
How to connect to a DB2 database on a z/OS server with java
Part 2: Explanation of the code
How to execute SQL statements on the DB2
Part 3: Optional To-Dos
If you have time, choose additional Labs
Part 4: Additional Resources
Where to get more information

Part 1: JDBC program accessing DB2 for z/OS data

This part shows a very simple approach to connect to DB2. It demonstrates the basic concepts of a JDBC connection. The created sample Java program connects from your workstation to the Host DB2 database and uses the DB2 Universal Java Driver to establish the connection.

Part 2 describes the necessary steps in more detail.

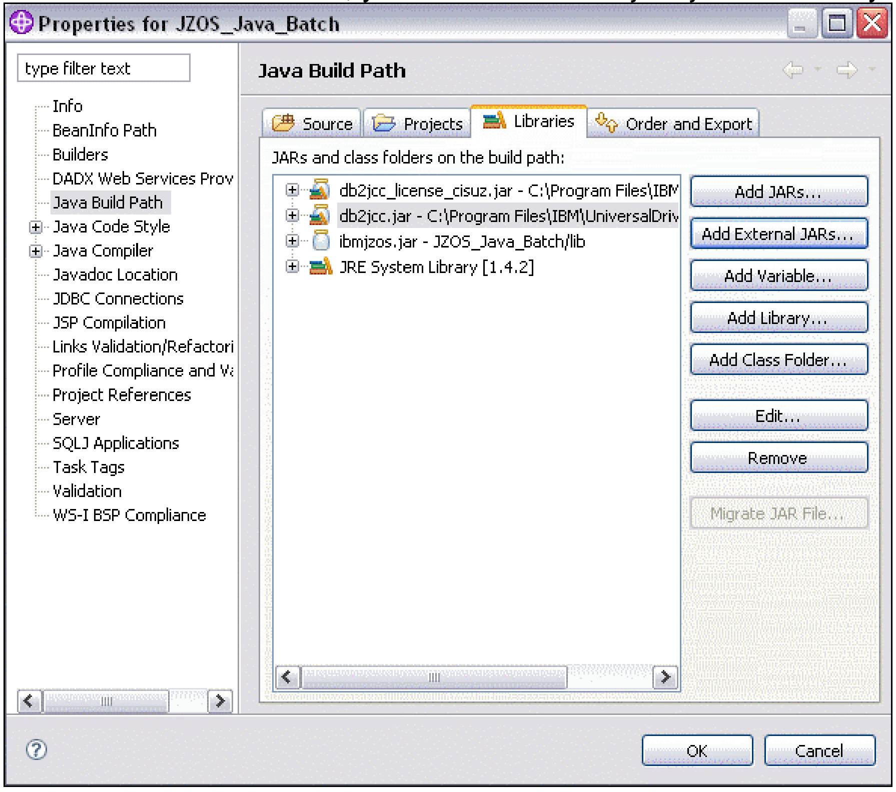
⏭ In Eclipse or WDz, create a new Java project.(File→NewProject→JavaProject) ☐
Enter *JavaDB2* as name and click on **Finish**

⏭ **Right-click** on the newly created project and select **Properties**

⏭ Go to **Java Build Path** ☐

⏭ In the **Libraries** tab, select **Add External Jars** to import the DB2 Universal Driver classes (db2jcc.jar and db2_license_cisuz.jar) into your Java classpath.

db2jcc.jar contains the driver classes and db2_license_cisuz.jar is the required license file that allows you to connect to a DB2 for z/OS server. ☐
To use JDBC 4.0 Standard, you can also use db2jcc4.jar alternatively.



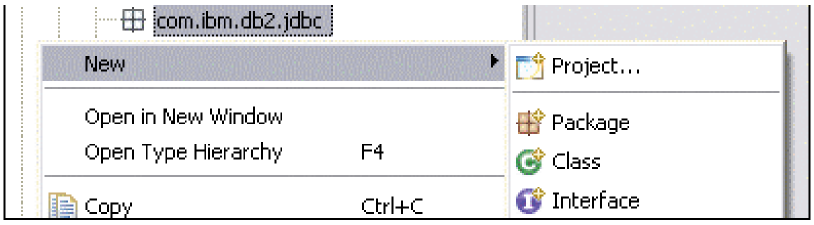
db2jcc_license_cisuz.jar - C:\Program Files\IBM
db2jcc.jar - C:\Program Files\IBM\UniversalDriv
ibmjzos.jar - JZOS_Java_Batch/lib
JRE System Library [1.4.2]

⏭ Click **OK**. ☐

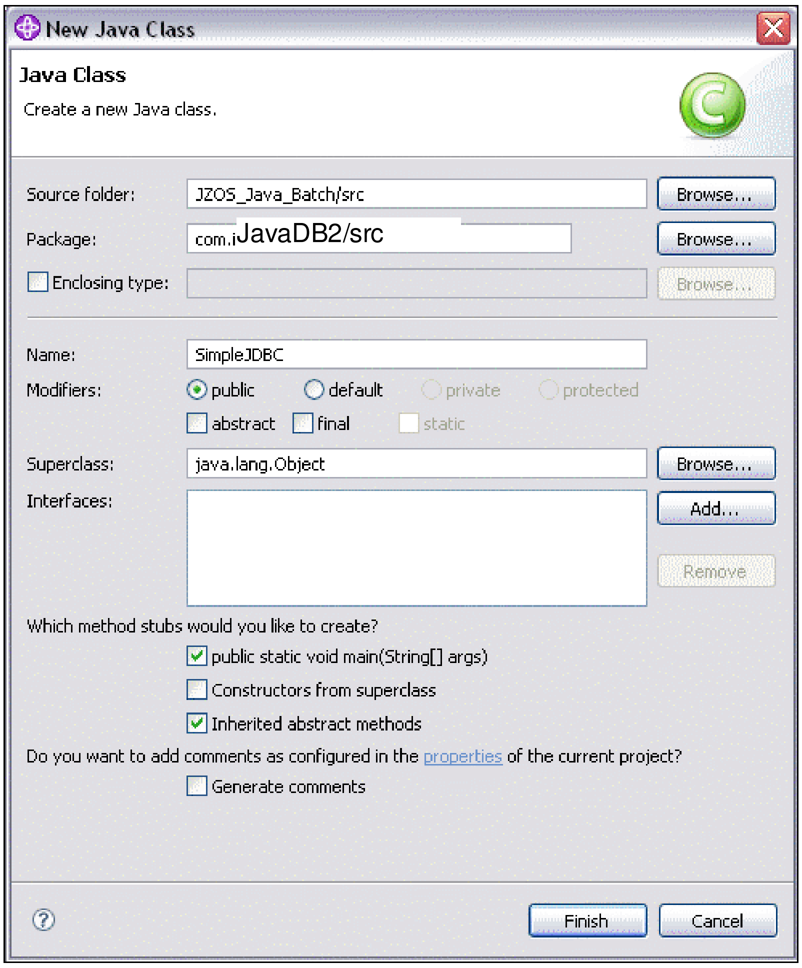
⏭ Right-click on the **JavaDB2** project and select **New → Package**. Type *com.ibm.db2.jdbc* as Name☐

⏭ Click **Finish**.

⏭ Right-click on the newly created package and select **New → Class** ☐



Fill in the following settings to create the Class **SimpleJDBC**



New Java Class

Java Class

Create a new Java class.

Source folder: JZOS_Java_Batch/src

Package: com.iJavaDB2/src

Enclosing type: ☐

Name: SimpleJDBC

Modifiers: [✓ public] default private protected

☐ abstract ☐ final ☐ static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

☑ public static void main(String[] args)

☐ Constructors from superclass

☑ Inherited abstract methods

Do you want to add comments as configured in the properties of the current project?

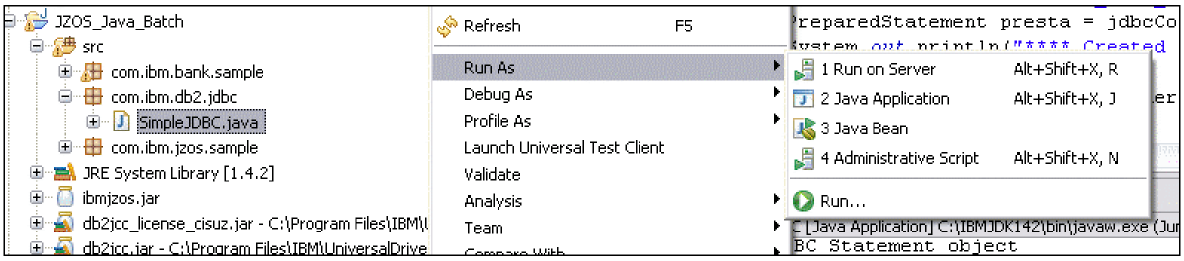
☐ Generate comments

Click **Finish** to start editing the class.

⏭ You have to type in your code into the created class. (Alternatively, copy the code from the PDF file). The following part 2 explains the inserted code. Ensure to enter your correct username and password. ☐

```
package com.ibm.db2.jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class SimpleJDBC {

	public static void main(String[] args) {

	  try {
	Class.forName("com.ibm.db2.jcc.DB2Driver");
	System.out.println("**** Loaded the JDBC driver");
	String url = "jdbc:db2://192.168.7.253:9446/ZOS10101";
	    Connection jdbcConn = DriverManager.
	             getConnection(url, "TEAM##", "PASSWORD");
	jdbcConn.setAutoCommit(false);
	System.out.println("**** Created JDBC connection");
	    String select = "SELECT OWNER, NAME FROM SYSIBM.SYSTABLES WHERE
OWNER = ?";
	    PreparedStatement presta = jdbcConn.prepareStatement(select);

	    System.out.println("**** Created JDBC Statement object");
	presta.setString(1, "TEAM##");
	    ResultSet rs = presta.executeQuery();
	    while (rs.next()) {
	      System.out.println("Table: "+rs.getString(1)+"."
+rs.getString(2));
	    }
	    System.out.println("**** Fetched all rows from JDBC ResultSet");
	    rs.close();
	    System.out.println("**** Closed JDBC ResultSet");
	    presta.close();
	    System.out.println("**** Closed JDBC Statement");

	    jdbcConn.commit();
	    System.out.println("**** Transaction committed");

	    jdbcConn.close();
	    System.out.println("**** Disconnected from data source");
	} catch (ClassNotFoundException e) {
	      System.err.println("Could not load JDBC driver");
	      System.out.println("Exception: " + e);
	      e.printStackTrace();
	} catch (SQLException ex) {
	      System.err.println("SQLException information");
	      while (ex != null) {
		     System.err.println("Error msg: " + ex.getMessage());
		     System.err.println("SQLSTATE: " + ex.getSQLState());
		     System.err.println("Error code: " + ex.getErrorCode());
		     ex.printStackTrace();
		     ex = ex.getNextException();

	      }
	}
	} // End main
} // End SimpleJDBC
```

⏭ Right-click the **SimpleJDBC.java** file and select **Run as→ Java Application**.
This will run the application on your local workstation and connect to DB2z via a Type4 connection.


JZOS_Java_Batch
src
com.ibm.bank.sample
com.ibm.db2.jdbc
SimpleJDBC.java
com.ibm.jzos.sample
JRE System Library [1.4.2]
ibmjzos.jar
db2jcc_license_cisuz.jar - C:\Program Files\IBM\
db2jcc.jar - C:\Program Files\IBM\UniversalDrive
Refresh F5
Run As
Debug As
Profile As
Launch Universal Test Client
Validate
Analysis
Team
1 Run on Server Alt+Shift+X, R
2 Java Application Alt+Shift+X, J
3 Java Bean
4 Administrative Script Alt+Shift+X, N
Run...

```
reparedStatement presta = jdbcCo
ystem.out.println("**** Created
C [Java Application] C:\IBMJDK142\bin\javaw.exe (Ju
BC Statement object
```

This should produce the following output: ☐

```
**** Loaded the JDBC driver
**** Created JDBC connection
**** Created JDBC Statement object
Table: TEAM11.AIRPORT
Table: TEAM11.AIRLINE
Table: TEAM11.SEATCLAS
Table: TEAM11.SEATCLASS
Table: TEAM11.FLIGHTS
**** Fetched all rows from JDBC ResultSet
**** Closed JDBC ResultSet
**** Closed JDBC Statement
**** Transaction committed
**** Disconnected from data source
```

Part 2: Explanation of the code

This chapter explains the code sample in part1 in more detail.

In order to work with SQL in a java program you have to import the java.sql classes.
Absolutely necessary are

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
```

Those imports are required to establish a connection to DB2 and to execute SQL statements.
In order to use Prepared Statements and Queries you also need the imports

```
import java.sql.PreparedStatement;
import java.sql.ResultSet;
```

You need PreparedStatements to make use of DB2´s dynamic statement cache.

The first statement in the main method loads the correct JDBC driver with the Class.forName command. The correct driver class for connecting to DB2 is “com.ibm.db2.jcc.DB2Driver”.

```
Class.forName("com.ibm.db2.jcc.DB2Driver");
```

Then you need to connect to the DB2 subsystem. The syntax for the database URL is: “Protocol//IP:Port/DB2 Location Name”

In our case this looks like:

```
String url = "jdbc:db2://IP-ADDRESS:Error! Reference source not
found./Locationname";
```

Now you can establish the connection. You need to create a connection object which serves as a handle for the connection and then attach it to an actual connection which you can get from the Drive Manager. The DriverManager needs the URL of the database, the username and password, either by hard coding them into the source code or by user input.

```
Connection jdbcConn = DriverManager.getConnection(url, user, password);
```

If nothing went wrong, you are now connected to the DB2. Before you quit the java program, don’t forget to close the connection. This is done by

```
jdbcConn.close();
```

Working with DB2

Prepared Statements should be used for SQL statements which have many variables like INSERT or SELECT … WHERE statements. Prepared Statements allow you to replace the values of the SQL datatypes in your statement by "?" and then to fill them later by using the set<Datatype>() Methods. Using PreparedStatements helps to use DB2s dynamic statement cache more efficiently.

Assuming this SQL statement:

```
String select = " SELECT OWNER, NAME FROM SYSIBM.SYSTABLES WHERE OWNER = ?
```

Then you need to create a PreparedStatement object and to use the prepareStatement method of the Connection to convert the SQL string into a prepared Statement.

```
PreparedStatement presta = jdbcConn.prepareStatement(select);
```

Then you can use the set<Datatype> methods of the PreparedStatements to set the value of the "?" in the Statement. As arguments you need the position of the "?" (beginning with 1) and the value for the variable.
For example to fill the statement with the integer value of "TEAM##" as its first operand.

```
presta.setInt(1, "TEAM##");
```

Finally you need to execute the PreparedStatement itself. The result of the query is given back to the program as resultset.

```
ResultSet rs = presta.executeQuery();
```

You can now print out the contents of the resultset to the console:

```
while (rs.next()) {
      System.out.println(("Table: "+rs.getString(1)+"." +rs.getString(2));
}
```

It is very important to clean up your program and the DB2 connection after the access. Therefore issue the following commands afterwards:

```
rs.close();
stmt.close();
con.commit();
jdbcConn.close();
```

End of lab ☺

Part 3: Optional ToDos

- Insert Data via JDBC
- Update Data via JDBC
- Call a Stored Procedure via JDBC
- Use the JPA Standard to Persist Java Objects in DB2

Part 4: Additional resources

DB2 for z/OS Application Programming Guide and Reference for Java (2.58 MB)
http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.db29.doc.java/dsnjvk15.pdf?noframes=true

DB2 Information Center
http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.db29.doc/db2prodhome.htm

JDBC official WebSite
http://java.sun.com/products/jdbc/

Wikipedia
http://en.wikipedia.org/wiki/Java_Database_Connectivity

OpenJPA
http://openjpa.apache.org/