How to connect to a DB2 database via JDBC toolkit

This document describes a step by step instruction to connect to a DB2 database via JDBC toolkit and get the SQL code and SQL message in case of any error.

1 - Download the DB2 jdbc driver (db2jcc4.jar) from:

http://www-01.ibm.com/support/docview.wss?uid=swg21363866

2 - Create a database and a test table in your DB2 database

login as db2inst1 on your DB2 server

create a database for example TESTDB2

create a test table and insert some data into table.

```
db2 create database TESTDB2
DB20000I  The CREATE DATABASE command completed successfully.

db2 connect to TESTDB2 user db2inst1 using db2passwd
Database Connection Information
Database server        = DB2/LINUXX8664 11.1.1
SQL authorization ID   = DB2INST1
Local database alias   = TESTDB2

db2 "create table test (name varchar(30), id int)";
DB20000I  The SQL command completed successfully.

db2 "insert into test values ('jim', 1)";
DB20000I  The SQL command completed successfully.
db2 "insert into test values ('kati', 2)";
DB20000I  The SQL command completed successfully.

db2 "select * from  test";
NAME                 ID
-------------------- -----------
jim                            1
kati                           2
2 record(s) selected.
```

3- Create a SPL project in your Streams server

JDBCDb2/Makefile

JDBCDb2/opt/db2jcc4.jar

JDBCDb2/application/JDBCDb2.spl

```
#########################################################################
# Copyright (C)2014, 2017 International Business Machines Corporation and
# others. All Rights Reserved.
#########################################################################

// *************************************************************************
// The sample SPL application JDBCDb2 demonstrates how to connect to a DB2 database
// and select data from a table using JDBCRun operator.
// It demonstrates also how to get the SQL message in case of any error.
//
// Required Streams Version = 4.1.x.x
// Required JDBC Toolkit Version = 1.2.2
// https://github.com/IBMStreams/streamsx.jdbc/releases/tag/v1.2.2
// DB2 jdbc driver (db2jcc4.jar)
// http://www-01.ibm.com/support/docview.wss?uid=swg21363866
//
// To connect to database, the following parameters need to be specified:
// jdbcDriverLib   : the jdbc driver libraries (download the jdbc driver file from DB2 site
// and store it in the opt folder, e.g. opt/db2jcc4.jar)
// jdbcClassName   : the class name for db2 jdbc driver (com.ibm.db2.jcc.DB2Driver)
// jdbcUrl         : the database URL. (e.g. "jdbc:db2://<your db2 databsae server>:50000/<y
// If you want to get more information in case of any SQL error add the following DB2
// JDBC parameter to your jdbcUrl:
// retrieveMessagesFromServerOnGetMessage=true;
// more detail in:
// https://www.ibm.com/support/knowledgecenter/en/SSEPGG_11.1.0/com.ibm.db2.luw.apdv.java.do
// dbcUser         :  the database user on whose behalf the connection is being made.
// jdbcPassword    : the user's password.
// sqlStatusAttr   : "error" ;
 // In this SPL sample:
// "select" operator demonstrates how to run SQL statement from stream attribute via stateme
// In this sample the JDBCRun operator connect to the database and read all rows from test t
// write them into data/output.csv
// The second output port "error" provide SQL code SQL Status and SQL message in case  of an
// *************************************************************************************/


namespace application ;

use com.ibm.streamsx.jdbc::* ;
use com.ibm.streamsx.jdbc.types::* ;
/*************************************************************************************
 * JDBCRunErrorPort demonstrates how to Error Port with JDBCRun operator.
 *************************************************************************************/
composite JDBCDb2
{
param
expression<rstring> $jdbcDriverLib : "opt/db2jcc4.jar" ;
expression<rstring> $jdbcClassName : "com.ibm.db2.jcc.DB2Driver" ;
expression<rstring> $jdbcUrl : "jdbc:db2://tank143.fyre.ibm.com:50000/TESTDB2:retrieveMessage
expression<rstring> $jdbcUser : "db2inst1" ;
expression<rstring> $jdbcPassword : "db2passwd" ;

type
insertSchema = int32 ID, rstring NAME ;
rsSchema = int32 ID, rstring NAME ;
selectSchema = rstring sql ;
graph
stream<insertSchema> pulse = Beacon()
{
param
iterations : 1000u ;
initDelay : 5.0 ;
}
```

```
(stream<rsSchema> runSql ; stream<tuple<insertSchema> inTuple, JdbcSqlStatus_T error> errors
JDBCRun(pulse)
{
logic
state :
{
mutable int32 count = 0  ;
}
}

//mutable int32 n=0
onTuple pulse : printStringLn((rstring) count++) ;

param
jdbcDriverLib  : $jdbcDriverLib ;
jdbcClassName  : $jdbcClassName ;
jdbcUrl        : $jdbcUrl ;
jdbcUser       : $jdbcUser ;
jdbcPassword   : $jdbcPassword ;
//statement      : "SELECT * FROM TEST" ;
statement      : "SELECT * FROM TEST2" ;
sqlStatusAttr  : "error" ;
}

() as errorprint = Custom(errors)
{
logic
onTuple errors : printStringLn("sqlCode: " +(rstring) error.sqlCode + ", sqlState: " +
error.sqlState + ", sqlMessage: " + error.sqlMessage) ;
}

() as runSqlprint = FileSink(runSql)
{
logic
onTuple runSql : printStringLn((rstring) ID + "," + NAME) ;
param
file : "output.csv" ;
}

}
```

4 - Make the SPL application

create a Makefile and run make

```
#################################################################################
# Copyright (C)2014, 2017 International Business Machines Corporation and
# others. All Rights Reserved.
#################################################################################

.PHONY: all clean

#SPLC_FLAGS = -t $(STREAMS_INSTALL)/toolkits/com.ibm.streamsx.jdbc  --data-directory data
SPLC_FLAGS = -t ../streamsx.jdbc/com.ibm.streamsx.jdbc  --data-directory data

SPLC = $(STREAMS_INSTALL)/bin/sc

SPL_CMD_ARGS ?=
SPL_COMP1NAME=JDBCDb2
SPL_MAIN_COMPOSITE1 = application::$(SPL_COMP1NAME)
BUILD_OUTPUT_DIR = output

all: data clean
```

```
	$(SPLC) $(SPLC_FLAGS) -M  $(SPL_MAIN_COMPOSITE1) --output-dir ./$(BUILD_OUTPUT_DIR)  $(SPL_CL

data:
	mkdir data
clean:
	$(SPLC) $(SPLC_FLAGS) -C -M $(SPL_MAIN_COMPOSITE1) --output-dir output
	-rm -rf toolkit.xml
	-rm -rf data/output.csv
```

5 - Run the SPL application

Change the database credentials in SPL file with your database credentials and run

```
$> make
```

Start the application with

```
$> output/bin/standalone
```

6 - check the SQL message

Change the statement in SPL file to

```
statement : "SELECT * FROM TEST2" ;
```

The table TEST2 doesn't exist in the database

```
$> make
$> output/bin/standalone
```

The JDCBRun operator delivers the following SQL code and SQL message, because the table TEST2 does not exist.

```
sqlCode: -204, sqlState: 42704, sqlMessage: "DB2INST1.TEST2" is an undefined name.. SQLCODE=
```