JDBC, NetRexx, Java

Robert John Wilson
robjohnwilson@hotmail.com

About Me

- Currently working as a middleware support consultant for IBM NL
- Providing support for DB2 / Java / PHP applications

- During my career I have developed applications for MVS/OS390/Windows and Unix platforms in a variety of languages including REXX :-)

Database Connectivity options

- Proprietary APIs
- Open DataBase Connectivity
- Java DataBase Connectivity

Language options

- Usual suspects, C, C++, .Net
- Also, REXX, PHP, PERL
- Only Java can make use of JDBC*

RDBMS

- Traditional players, IBM DB2, Oracle, Infomix (now an IBM product), Microsoft SQL Server
- Some 'new' contenders, MySQL (aquired by Oracle) ...

Challenge

- Support team supporting ever more complex Java applications
- Development team is offshore, communication is sometimes difficult.
- Difficult to pin-point source of problems due to bad application logging, often incidents are 'bounced' back and forth between support and development, lots of finger pointing.

Requirements

- Create a tool to assist with the debugging of Java application-database connectivity issues.
- Tool must provide proof that database is or is not source of any problem.
- Tool must be multiplatform and support both IBM DB2 and Oracle

The DB2 JDBC driver

- Different Types
  - Type 1 JDBC-ODBC Bridge
  - Type 2 Native API
  - Type 3 Pure Driver for Database Middleware
  - Type 4 Native Protocol Driver, Pure Java Driver

* our tool was built to support only type 2 and type 4

The DB2 JDBC driver

- A JDBC driver is supplied with the DB2 product, also available for download
- http://www-01.ibm.com/software/data/db2/express/download.html

Properties

- Host name
- Database name
- Port number
- SecurityMechanism
- TraceLevel
- TraceFileName

Why use NetRexx?

- Team skillset, everyone knows REXX.
- Resulting Java can be used on all Linux, Unix, Windows
- Gives opportunity for team to get exposure to Java

Getting started

Making a connection

```
con = java.sql.connection

props = Properties()

usr = “test”

pwd = “abcdef”

host = “localhost”

port = “50000”

db = “testdb”

props.setProperty(user, usr)

props.setProperty(password, pwd)

props.setProperty(“securityMechanism”,13)

url_ = "jdbc:db2://" || host || ":" || port || "/" || db_

con = java.sql.Connection java.sql.DriverManager.getConnection(url, props)
```

Executing SQL

```
sql = "SELECT CURRENT DATE FROM SYSIBM.SYSDUMMY1";
stmt = java.sql.Statement con.createStatement();
say "Executing the following query to test communication:" || sql

rs = ResultSet stmt.executeQuery(sql);
ts = String "";
loop while (rs.next())
  ts = String rs.getString(1);
  say "Query output : " ts
end
rs.close();
stmt.close();
catch e2 = java.sql.SQLException
  say "I think an error might possibly have happened ... error text follows:" e2
  say 'Exception (' e2 ') caught : \n' e2.getMessage()
  return
catch NullPointerException
  nop
end
```

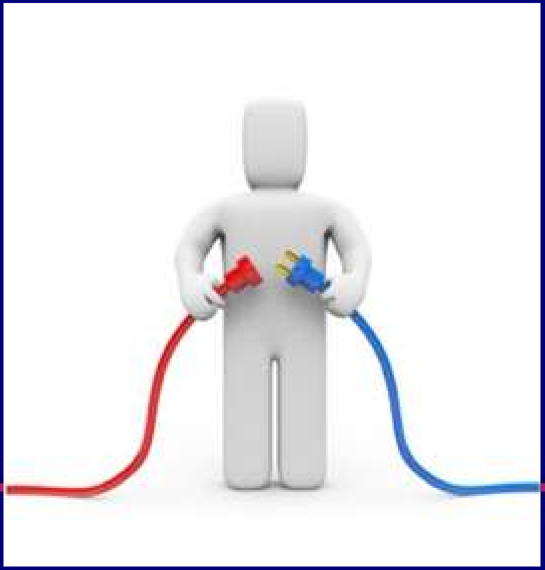
Gathering driver metadata

```
con = java.sql.Connection java.sql.DriverManager.getConnection(url_, props)
-- get some metadata regarding the connection
dm = con.getMetaData()
say "**************************************************************"
say "Connection OK, printing some meta data about the connection..."
say "Driver Name: " dm.getDriverName()
say "Driver Version:" dm.getDriverVersion()
say "Database Name:" dm.getDatabaseProductName()
say "Database Version:" dm.getDatabaseProductVersion()
say "**************************************************************"
```

JDBC Tracing

- Trace output can help find application errors or application configuration issues.
- Performance tuning.
- Understanding 3rd party software

Connectivity Issue????



JDBC Trace options

- *Legacy Type 2 driver offers two possibilities:
  - JDBC layer trace
  - CLI layer trace

  - Both options are configured via the db2cli.ini file

JDBC Trace options

- DB2 Universal driver trace options set via driver properties
- Can be set in code as in my example

JDBC Trace options

- Can also be set in external file, useful for 3rd party applications.

```
db2.jcc.traceFile=trace
db2.jcc.traceFileAppend=false
db2.jcc.traceDirectory=c:\\temp
db2.jcc.traceLevel=-1
```

- Use the -D switch to specify properties file:

```
java -Ddb2.jcc.propertiesFile=jcc.properties JccTraceExample2
```

JDBC Trace options

- It is possible to override trace options in the event they are 'hardcoded' in source code:

```
db2.jcc.override.traceDirectory=c:\\temp

db2.jcc.override.traceFile=trace

db2.jcc.override.traceFileAppend=false

db2.jcc.override.traceLevel=-1
```

Things to check

- Differing Java Runtime Environments (check versions, vendor)
- JDBC driver versions
- Security settings

More information

- http://www.ibm.com/developerworks/data/
- http://www.ibm.com/developerworks/data/library/techarticle/dm-0512kokkat/

Questions?

```
import java.lang.Runtime;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.util.Properties;

import com.ibm.db2.jcc.DB2Connection;

class DbConnectTest

  -- Global properties
  con = java.sql.connection
  connectionFile = String
  props = java.util.properties
  jdbcdriverp = Rexx
  url_ = Rexx

  -- Default constructor
  method DbConnectTest()

  method testConnect()
    readPropertiesFile(this.connectionFile)
    dbConnect()

  /**
   * Method readPropertiesFile reads a text file containing
   * JDBC properties to use for testing this connection
   *
  */
    method readPropertiesFile(name_)

    -- Initialize some required properties
    user_ = ""  -- user for jdbc connection
    password_ = "" -- password for jdbc connection
    -- following two lines are purely to get rid of annoying warning
    -- about variables being defined but not used
    user_ = user_
    password_ = password_
    props = Properties()
    driverType = "4"
```

```
do
  fi = BufferedReader(FileReader(name_))
  loop forever
    textLine = Rexx fi.readLine()
    if textLine = null then leave
    if textLine.substr(1,2) = "--" then iterate
    if textLine.pos("=") > 0 then
    do
      textLine = textLine.translate(" ","=")
      propname = textLine.word(1)
      propval  = textLine.word(2)
      props.setProperty(propname,propval)
      if propname.upper() = "HOST" then
        host_ = propval
      if propname.upper() = "PORT" then
        port_ = propval
      if propname.upper() = "DATABASE" then
         db_ = propval
      if propname.upper() = "DRIVER" then
         jdbcdriverp = propval
      if propname.upper() = "DRIVERTYPE" then
         driverType = propval
    end
  end
  if jdbcdriverp.length() > 0 then
    do
      if driverType = 2 then
        url_ = "jdbc:db2:" || db_
      if driverType = 4 then
        url_ = "jdbc:db2://" || host_ || ":" || port_ || "/" || db_
      return
    end
  else
    return
  catch IOException
    say "File" name_ || " could not be found."
    exit
 end
```

```
/**
  * Method dbConnect connects to a database using jdbc
  * @param jdbcdriverl is a Rexx for the jdbc driver name
  * @param url_ is a Rexx for the database url
  */
   method dbConnect()


   -- force loading of jdbc driver
   do
     Class.forName(jdbcdriverp).newInstance()
   catch e1 = Exception
     say 'JDBC driver could not be loaded.'
     say 'Exception (' e1 ') caught : \n' e1.getMessage()
     return
   end
   do
     -- make the connection
     say "***************************************************************"
     say "Testing the connection using this connect url:              "
     say url_
     say "***************************************************************"
     con = java.sql.Connection java.sql.DriverManager.getConnection(url_, props)
     -- get some metadata regarding the connection
     dm = con.getMetaData()
     say "***************************************************************"
     say "Connection OK, printing some meta data about the connection..."
     say "Driver Name: " dm.getDriverName()
     say "Driver Version:" dm.getDriverVersion()
     say "Database Name:" dm.getDatabaseProductName()
     say "Database Version:" dm.getDatabaseProductVersion()
     say "***************************************************************"


     -- execute the most basic of queries to check the db2 address space really does
     -- respond
     sql = "SELECT CURRENT DATE FROM SYSIBM.SYSDUMMY1";
     stmt = java.sql.Statement con.createStatement();
     say "Executing the following query to test communication:" || sql
```

```
rs = ResultSet stmt.executeQuery(sql);
    ts = String "";
    loop while (rs.next())
      ts = String rs.getString(1);
      say "Query output : " ts
    end
    say "***************************************************************"
    rs.close();
    stmt.close();
  catch e2 = java.sql.SQLException
    say "I think an error might possibly have happened ... error text follows:" e2
    say 'Exception (' e2 ') caught : \n' e2.getMessage()
    return
    catch NullPointerException
      nop
  end


method showArgs()
  do
    say "usage: java DbConnecTest propertiesfile"
  end


method main(args=String[]) static
  do
    a = DbConnectTest()
    uri = String args[0]
    if uri = "--help" then a.showArgs()
    if uri = null then a.showArgs()
    a.connectionFile = uri
    a.testConnect()
  catch java.lang.ArrayIndexOutOfBoundsException
    a.showArgs()
    exit
  end
```

```
host=test.nl.eu.happydayz.com
port=60961
database=test
driver=com.ibm.db2.jcc.DB2Driver
user=timothy
password=welcome01
#securityMechanism=3 --CLEAR_TEXT_PASSWORD_SECURITY
#securityMechanism=4 --USER_ONLY_SECURITY
#securityMechanism=7 --ENCRYPTED_PASSWORD_SECURITY
securityMechanism=9 --ENCRYPTED_USER_AND_DATA_SECURITY
#securityMechanism=11 --KERBEROS_SECURITY
#securityMechanism=12 --ENCRYPTED_USER_AND_DATA_SECURITY
#securityMechanism=13 --ENCRYPTED_USER_PASSWORD_AND_DATA_SECURITY
#securityMechanism=15 --PLUGIN_SECURITY
#securityMechanism=16 --ENCRYPTED_USER_ONLY_SECURITY
traceFile=jdbctrace.siecma1
driverType=4
--traceLevel=1024 -- TRACE_SQLJ
--traceLevel=1    --TRACE_CONNECTION_CALLS
--traceLevel=2    --TRACE_STATEMENT_CALLS
--traceLevel=4    --TRACE_RESULT_SET_CALLS
--traceLevel=16   --TRACE_DRIVER_CONFIGURATION
--traceLevel=32   --TRACE_CONNECTS
--traceLevel=64   --TRACE_DRDA_FLOWS
--traceLevel=128  --TRACE_RESULT_SET_META_DATA
--traceLevel=256  --TRACE_PARAMETER_META_DATA
--traceLevel=512  --TRACE_DIAGNOSTICS
--traceLevel=2048 --TRACE_XA_CALLS
traceLevel=-1   --TRACE_ALL
```

```
[ibm][db2][jcc] BEGIN TRACE_DRIVER_CONFIGURATION
[ibm][db2][jcc] Driver: IBM DB2 JDBC Universal Driver Architecture 1.2.117
[ibm][db2][jcc] Compatible JRE versions: { 1.3, 1.4 }
[ibm][db2][jcc] Target server licensing restrictions: { z/OS: enabled; SQLDS: enabled; iSeries:
enabled; DB2 for Unix/Windows: enabled; Cloudscape: enabled }
[ibm][db2][jcc] Range checking enabled: true
[ibm][db2][jcc] Bug check level: 0xff
[ibm][db2][jcc] Trace level: 0xffffffff
[ibm][db2][jcc] Default fetch size: 64
[ibm][db2][jcc] Default isolation: 2
[ibm][db2][jcc] Collect performance statistics: false
[ibm][db2][jcc] No security manager detected.
[ibm][db2][jcc] Detected local client host: n210l03/10.124.19.221
[ibm][db2][jcc] Access to package sun.io is permitted by security manager.
[ibm][db2][jcc] JDBC 1 system property jdbc.drivers = null
[ibm][db2][jcc] Java Runtime Environment version 1.5.0
[ibm][db2][jcc] Java Runtime Environment vendor = IBM Corporation
[ibm][db2][jcc] Java vendor URL = http://www.ibm.com/
[ibm][db2][jcc] Java installation directory = /usr/java5/jre
[ibm][db2][jcc] Java Virtual Machine specification version = 1.0
[ibm][db2][jcc] Java Virtual Machine specification vendor = Sun Microsystems Inc.
[ibm][db2][jcc] Java Virtual Machine specification name = Java Virtual Machine Specification
[ibm][db2][jcc] Java Virtual Machine implementation version = 2.3
[ibm][db2][jcc] Java Virtual Machine implementation vendor = IBM Corporation
[ibm][db2][jcc] Java Virtual Machine implementation name = IBM J9 VM
[ibm][db2][jcc] Java Runtime Environment specification version = 1.5
[ibm][db2][jcc] Java Runtime Environment specification vendor = Sun Microsystems Inc.
[ibm][db2][jcc] Java Runtime Environment specification name = Java Platform API Specification
[ibm][db2][jcc] Java class format version number = 49.0
[ibm][db2][jcc] Java class path = :/usr/java5/lib/tools.jar:/appl/mwp/lib/java/NetRexxC.jar:/appl/
mwp/lib/java/jpl.jar:/appl/mwp/lib/java/postgres.jar:/home/wi4911/mwp/mq/src:.:/
home/wi4911/dcs/mq/lib/java/NetRexxR.jar:/home/wi4911/dcs/mq/lib/java/MQDocument.jar:/appl/mwp/
lib/java/mwp.jar:/home/wi4911/dcs/mq/lib/java/mysql-connector-java-5.0.4-bin.jar:/u
sr/opt/db2_08_01/java/db2jcc.jar:/usr/opt/db2_08_01/java/db2java.zip:/usr/opt/db2_08_01/java/
db2jcc_license_cu.jar:/home/wi4911/dcs/mq/lib/java/mail.jar:/home/wi4911/dcs/mq/lib/j
ava/activation.jar:/home/wi4911/db2jcc_license_cisuz.jar
[ibm][db2][jcc] Java native library path = /usr/java5/jre/bin:/usr/java5/jre/bin:/usr/java5/jre/
bin/classic:/usr/java5/jre/bin:/home/wi4911/Quest_Software/qcdb2/bin:/usr/java5/jr
e/bin/j9vm:/appl/mwp/lib:/usr/lib
[ibm][db2][jcc] Path of extension directory or directories = /usr/java5/jre/lib/ext
[ibm][db2][jcc] Operating system name = AIX
```

```
[ibm][db2][jcc] Operating system architecture = ppc
[ibm][db2][jcc] Operating system version = 5.2
[ibm][db2][jcc] File separator ("/" on UNIX) = /
[ibm][db2][jcc] Path separator (":" on UNIX) = :
[ibm][db2][jcc] User's account name = wi4911
[ibm][db2][jcc] User's home directory = /home/wi4911
[ibm][db2][jcc] User's current working directory = /home/wi4911/dbms/utilities/build
[ibm][db2][jcc] END TRACE_DRIVER_CONFIGURATION
[ibm][db2][jcc] BEGIN TRACE_CONNECTS
[ibm][db2][jcc] Attempting connection to zb131l03.nl.eu.abnamro.com:60960/SDECMA1
[ibm][db2][jcc] Using properties: { securityMechanism=9, traceLevel=-1, port=60960,
#securityMechanism=16, user=ssecma1, traceFile=jdbctrace.siecma1, driverType=4, database=SDECM
A1, password=<escaped>, driver=com.ibm.db2.jcc.DB2Driver, host=zb131l03.nl.eu.abnamro.com }
[ibm][db2][jcc] END TRACE_CONNECTS
[ibm][db2][jcc][t4] Request.flush() called at 2010-11-5 10:57:19 Thread: main Tracepoint: 1
[ibm][db2][jcc][t4]        SEND BUFFER: EXCSAT                    (ASCII)           (EBCDIC)
[ibm][db2][jcc][t4]        0 1 2 3 4 5 6 7   8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
[ibm][db2][jcc][t4] 0000   0055D0010001004F  1041000E115E8482  .U.....O.A...^..  ..}....|.....;db
[ibm][db2][jcc][t4] 0010   F291838394818995  000B116D95F2F1F0  ............m....  2jccmain..._n210
[ibm][db2][jcc][t4] 0020   93F0F3000E115AC4  C2F2D1C3C340F14B  ......Z......@.K  l03...!DB2JCC 1.
[ibm][db2][jcc][t4] 0030   F000181404140300  0724070007240F00  .........$...$..  0...............
[ibm][db2][jcc][t4] 0040   0714400006147400  05000C1147D8C4C2  ..@...t.....G...  .. ..........QDB
[ibm][db2][jcc][t4] 0050   F261D1E5D4                          .a...             2/JVM
[ibm][db2][jcc][t4]
[ibm][db2][jcc][t4] Reply.fill() called at 2010-11-5 10:57:19 Thread: main Tracepoint: 2
[ibm][db2][jcc][t4]        RECEIVE BUFFER: EXCSATRD               (ASCII)           (EBCDIC)
[ibm][db2][jcc][t4]        0 1 2 3 4 5 6 7   8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
[ibm][db2][jcc][t4] 0000   006BD00300010065  14430024115EA289  .k.....e.C.$.^..  .,}..........;si
[ibm][db2][jcc][t4] 0010   85839481F1408482  F281878595A3F0F0  .....@..........  ecma1 db2agent00
[ibm][db2][jcc][t4] 0020   F0F4C3F3C5C56CC6  C5C46CE8F0F00018  ......l...l.....  04C3EE%FED%Y00..
[ibm][db2][jcc][t4] 0030   1404140300072407  0007240F00071440  ......$...$....@  ...............
[ibm][db2][jcc][t4] 0040   000614740005000E  1147D8C4C2F261C1  ...t.....G....a.  ..........QDB2/A
[ibm][db2][jcc][t4] 0050   C9E7F6F4000B116D  A28985839481F100  .......m........  IX64..._siecma1.
[ibm][db2][jcc][t4] 0060   0C115AE2D8D3F0F9  F0F5F6            ..Z........       ..!SQL09056
[ibm][db2][jcc][t4]
```