<https://docs.oracle.com/cd/A91034_01/DOC/java.901/a90211/samapp.htm>

/\*

\* This sample shows how to call PL/SQL blocks from JDBC.

\*/

import java.sql.\*;

class PLSQL

{

public static void main (String args [])

throws SQLException, ClassNotFoundException

{

// Load the driver

Class.forName ("oracle.jdbc.OracleDriver");

// Connect to the database

// You can put a database name after the @ sign in the connection URL.

Connection conn =

DriverManager.getConnection ("jdbc:oracle:oci8:@", "scott", "tiger");

// Create the stored procedures

init (conn);

// Cleanup the plsqltest database

Statement stmt = conn.createStatement ();

stmt.execute ("delete from plsqltest");

// Close the statement

stmt.close();

// Call a procedure with no parameters

{

CallableStatement procnone = conn.prepareCall ("begin procnone; end;");

procnone.execute ();

dumpTestTable (conn);

procnone.close();

}

// Call a procedure with an IN parameter

{

CallableStatement procin = conn.prepareCall ("begin procin (?); end;");

procin.setString (1, "testing");

procin.execute ();

dumpTestTable (conn);

procin.close();

}

// Call a procedure with an OUT parameter

{

CallableStatement procout = conn.prepareCall ("begin procout (?); end;");

procout.registerOutParameter (1, Types.CHAR);

procout.execute ();

System.out.println ("Out argument is: " + procout.getString (1));

procout.close();

}

// Call a procedure with an IN/OUT prameter

{

CallableStatement procinout = conn.prepareCall

("begin procinout (?); end;");

procinout.registerOutParameter (1, Types.VARCHAR);

procinout.setString (1, "testing");

procinout.execute ();

dumpTestTable (conn);

System.out.println ("Out argument is: " + procinout.getString (1));

procinout.close();

}

// Call a function with no parameters

{

CallableStatement funcnone = conn.prepareCall

("begin ? := funcnone; end;");

funcnone.registerOutParameter (1, Types.CHAR);

funcnone.execute ();

System.out.println ("Return value is: " + funcnone.getString (1));

funcnone.close();

}

// Call a function with an IN parameter

{

CallableStatement funcin = conn.prepareCall

("begin ? := funcin (?); end;");

funcin.registerOutParameter (1, Types.CHAR);

funcin.setString (2, "testing");

funcin.execute ();

System.out.println ("Return value is: " + funcin.getString (1));

funcin.close();

}

// Call a function with an OUT parameter

{

CallableStatement funcout = conn.prepareCall

("begin ? := funcout (?); end;");

funcout.registerOutParameter (1, Types.CHAR);

funcout.registerOutParameter (2, Types.CHAR);

funcout.execute ();

System.out.println ("Return value is: " + funcout.getString (1));

System.out.println ("Out argument is: " + funcout.getString (2));

funcout.close();

}

// Close the connection

conn.close();

}

// Utility function to dump the contents of the PLSQLTEST table and

// clear it

static void dumpTestTable (Connection conn)

throws SQLException

{

Statement stmt = conn.createStatement ();

ResultSet rset = stmt.executeQuery ("select \* from plsqltest");

while (rset.next ())

System.out.println (rset.getString (1));

stmt.execute ("delete from plsqltest");

rset.close();

stmt.close();

}

// Utility function to create the stored procedures

static void init (Connection conn)

throws SQLException

{

Statement stmt = conn.createStatement ();

try { stmt.execute ("drop table plsqltest"); } catch (SQLException e) { }

stmt.execute ("create table plsqltest (x char(20))");

stmt.execute ("create or replace procedure procnone

is begin insert into plsqltest values ('testing'); end;");

stmt.execute ("create or replace procedure procin (y char)

is begin insert into plsqltest values (y); end;");

stmt.execute ("create or replace procedure procout (y out char)

is begin y := 'tested'; end;");

stmt.execute ("create or replace procedure procinout (y in out varchar)

is begin insert into plsqltest values (y);

y := 'tested'; end;");

stmt.execute ("create or replace function funcnone return char

is begin return 'tested'; end;");

stmt.execute ("create or replace function funcin (y char) return char

is begin return y || y; end;");

stmt.execute ("create or replace function funcout (y out char) return char

is begin y := 'tested'; return 'returned'; end;");

stmt.close();

}

}