Appendix B. First JDBC Client Example

There is a copy of `Testdb.java` in the directory `src/org/hsqldb/sample` of your HSQLDB distribution.

Example B.1. JDBC Client source code example

```
/* Copyright (c) 2001-2005, The HSQL Development Group
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * Redistributions of source code must retain the above copyright notice, this
 * list of conditions and the following disclaimer.
 *
 * Redistributions in binary form must reproduce the above copyright notice,
 * this list of conditions and the following disclaimer in the documentation
 * and/or other materials provided with the distribution.
 *
 * Neither the name of the HSQL Development Group nor the names of its
 * contributors may be used to endorse or promote products derived from this
 * software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED. IN NO EVENT SHALL HSQL DEVELOPMENT GROUP, HSQLDB.ORG,
 * OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */


package org.hsqldb.sample;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;

/**
 * Title:        Testdb
 * Description:  simple hello world db example of a
 *               standalone persistent db application
 *
 *               every time it runs it adds four more rows to sample_table
 *               it does a query and prints the results to standard out
 *
 * Author: Karl Meissner karl@meissnersd.com
 */
public class Testdb {

    Connection conn;                                                //our connnection to the db - presist for life of program

    // we dont want this garbage collected until we are done
    public Testdb(String db_file_name_prefix) throws Exception {    // note more general exception

        // Load the HSQL Database Engine JDBC driver
        // hsqldb.jar should be in the class path or made part of the current jar
        Class.forName("org.hsqldb.jdbcDriver");

        // connect to the database.   This will load the db files and start the
        // database if it is not alread running.
        // db_file_name_prefix is used to open or create files that hold the state
        // of the db.
        // It can contain directory names relative to the
        // current working directory
        conn = DriverManager.getConnection("jdbc:hsqldb:"
                                           + db_file_name_prefix,    // filenames
                                           "sa",                     // username
                                           "");                      // password
    }

    public void shutdown() throws SQLException {

        Statement st = conn.createStatement();

        // db writes out to files and performs clean shuts down
        // otherwise there will be an unclean shutdown
        // when program ends
        st.execute("SHUTDOWN");
        conn.close();    // if there are no other open connection
    }

//use for SQL command SELECT
    public synchronized void query(String expression) throws SQLException {

        Statement st = null;
        ResultSet rs = null;

        st = conn.createStatement();         // statement objects can be reused with

        // repeated calls to execute but we
        // choose to make a new one each time
        rs = st.executeQuery(expression);    // run the query

        // do something with the result set.
        dump(rs);
        st.close();    // NOTE!! if you close a statement the associated ResultSet is

        // closed too
        // so you should copy the contents to some other object.
        // the result set is invalidated also  if you recycle an Statement
        // and try to execute some other query before the result set has been
        // completely examined.
    }

//use for SQL commands CREATE, DROP, INSERT and UPDATE
    public synchronized void update(String expression) throws SQLException {

        Statement st = null;

        st = conn.createStatement();    // statements

        int i = st.executeUpdate(expression);    // run the query

        if (i == -1) {
            System.out.println("db error : " + expression);
        }

        st.close();
    }    // void update()

    public static void dump(ResultSet rs) throws SQLException {

        // the order of the rows in a cursor
        // are implementation dependent unless you use the SQL ORDER statement
        ResultSetMetaData meta   = rs.getMetaData();
        int               colmax = meta.getColumnCount();
        int               i;
        Object            o = null;

        // the result set is a cursor into the data.  You can only
        // point to one row at a time
        // assume we are pointing to BEFORE the first row
        // rs.next() points to next row and returns true
        // or false if there is no next row, which breaks the loop
        for (; rs.next(); ) {
            for (i = 0; i < colmax; ++i) {
                o = rs.getObject(i + 1);    // Is SQL the first column is indexed

                // with 1 not 0
                System.out.print(o.toString() + " ");
            }
            System.out.println(" ");
        }
    }                                       //void dump( ResultSet rs )

    public static void main(String[] args) {

        Testdb db = null;

        try {
            db = new Testdb("db_file");
        } catch (Exception ex1) {
            ex1.printStackTrace();    // could not start db

            return;                   // bye bye
        }

        try {

            //make an empty table
            //
            // by declaring the id column IDENTITY, the db will automatically
            // generate unique values for new rows- useful for row keys
            db.update(
                "CREATE TABLE sample_table ( id INTEGER IDENTITY, str_col VARCHAR(256), num_col INTEGER)");
        } catch (SQLException ex2) {

            //ignore
            //ex2.printStackTrace();  // second time we run program
            //  should throw execption since table
            // already there
            //
            // this will have no effect on the db
        }

        try {

            // add some rows - will create duplicates if run more then once
            // the id column is automatically generated
            db.update(
                "INSERT INTO sample_table(str_col,num_col) VALUES('Ford', 100)");
            db.update(
                "INSERT INTO sample_table(str_col,num_col) VALUES('Toyota', 200)");
            db.update(
                "INSERT INTO sample_table(str_col,num_col) VALUES('Honda', 300)");
            db.update(
                "INSERT INTO sample_table(str_col,num_col) VALUES('GM', 400)");

            // do a query
            db.query("SELECT * FROM sample_table WHERE num_col < 250");

            // at end of program
            db.shutdown();
        } catch (SQLException ex3) {
            ex3.printStackTrace();
        }
    }    // main()
}    // class Testdb
```

Appendix A. Building HSQLDB

Appendix C. Hsqldb Database Files and Recovery