Embedded database in Java – use of HSQLDB.

September 15, 2009 - Author: Miron Sadziak

If you ever wanted to have a small, portable database that you could launch directly from the Java code, now you can do it – with HSQLDB! HSQLDB is written entirely in Java. To try it out, you will need just one file – hsqldb.jar which you will find in lib folder of the archive downloaded from here. Nothing else, just put the file on your class path and you are ready to roll!

The code belows shows a very simple use of HSQLDB API:

```
 1: import java.sql.Connection;
 2: import java.sql.DriverManager;
 3: import java.sql.ResultSet;
 4: import java.sql.SQLException;
 5:
 6: import org.hsqldb.Server;
 7:
 8: public class HSQLDBTest {
 9:
10:     public static void main(String[] args) throws
11:         ClassNotFoundException, SQLException {
12:
13:         // 'Server' is a class of HSQLDB representing
14:         // the database server
15:         Server hsqlServer = null;
16:         try {
17:             hsqlServer = new Server();
18:
19:             // HSQLDB prints out a lot of informations when
20:             // starting and closing, which we don't need now.
31:             // Normally you should point the setLogWriter
32:             // to some Writer object that could store the logs.
33:             hsqlServer.setLogWriter(null);
34:             hsqlServer.setSilent(true);
35:
36:             // The actual database will be named 'xdb' and its
37:             // settings and data will be stored in files
38:             // testdb.properties and testdb.script
39:             hsqlServer.setDatabaseName(0, "xdb");
40:             hsqlServer.setDatabasePath(0, "file:testdb");
41:
42:             // Start the database!
43:             hsqlServer.start();
44:
45:             Connection connection = null;
46:             // We have here two 'try' blocks and two 'finally'
47:             // blocks because we have two things to close
48:             // after all - HSQLDB server and connection
49:             try {
50:                 // Getting a connection to the newly started database
61:                 Class.forName("org.hsqldb.jdbcDriver");
62:                 // Default user of the HSQLDB is 'sa'
63:                 // with an empty password
64:                 connection = DriverManager.getConnection(
65:                     "jdbc:hsqldb:hsql://localhost/xdb", "sa", "");
66:
67:                 // Here we run a few SQL statements to see if
68:                 // everything is working.
69:                 // We first drop an existing 'testtable' (supposing
70:                 // it was there from the previous run), create it
71:                 // once again, insert some data and then read it
```

```
72:                 // with SELECT query.
73:                 connection.prepareStatement("drop table testtable;")
74:                     .execute();
75:                 connection.prepareStatement(
76:                     "create table testtable ( id INTEGER, "+
77:                     "name VARCHAR);")
78:                     .execute();
79:                 connection.prepareStatement(
80:                     "insert into testtable(id, name) "+
91:                     "values (1, 'testvalue');")
92:                     .execute();
93:                 ResultSet rs = connection.prepareStatement(
94:                     "select * from testtable;").executeQuery();
95:
96:                 // Checking if the data is correct
97:                 rs.next();
98:                 System.out.println("Id: " + rs.getInt(1) + " Name: "
99:                     + rs.getString(2));
90:             } finally {
101:                // Closing the connection
102:                if (connection != null) {
103:                    connection.close();
104:                }
105:
106:            }
107:        } finally {
108:            // Closing the server
109:            if (hsqlServer != null) {
110:                hsqlServer.stop();
111:            }
112:        }
113:    }
114:}
```

When you run this code, you should receive following output:

```
Id: 1 Name: testvalue
```

As you can see in the code, we first start the HSQLDB with the method Server.start(). Since that time, HSQLDB server runs as a separate thread. In this example, tables and data that we put in the database will be all stored in file ‘testdb.script’. Let’s take a look at the file created after the run of our code:

```
CREATE SCHEMA PUBLIC AUTHORIZATION DBA
CREATE MEMORY TABLE TESTTABLE(ID INTEGER,NAME VARCHAR)
CREATE USER SA PASSWORD ""
GRANT DBA TO SA
SET WRITE_DELAY 10
SET SCHEMA PUBLIC
INSERT INTO TESTTABLE VALUES(1,'testvalue')
```

As you see it is just a bunch of SQL statements creating the schema, tables, setting up user accounts and putting the data into tables. You can manually modify this file and every change you make to it will visible in the actual database after the next run of the server.

There is also ‘testdb.properties’ file which stores all the settings of the database. In our example it looks like that:

```
#HSQL Database Engine 1.8.0.10
#Sun Sep 06 18:11:59 JST 2009
hsqldb.script_format=0
runtime.gc_interval=0
sql.enforce_strict_size=false
hsqldb.cache_size_scale=8
readonly=false
hsqldb.nio_data_file=true
hsqldb.cache_scale=14
version=1.8.0
hsqldb.default_table_type=memory
hsqldb.cache_file_scale=1
hsqldb.log_size=200
modified=yes
hsqldb.cache_version=1.7.0
hsqldb.original_version=1.8.0
hsqldb.compatible_version=1.8.0
```

Besides using HSQLDB in your normal programs, it is also a perfect database for integration tests of your software. You can configure your testing suite to start the HSQLDB every time before a test with exactly specified tables and data. No more relying on external databases and no more fear that somebody could have changed your test data without a notice (which, by the way, you could achieve also with DbUnit).

Categories: Uncategorized 13 Comments

13 Comments until now

Carol Adams September 15th, 2009 (#):

Thanks for the tip – but you can still do the same with SQLite. I have developed a few standalone apps for clients, and they all used SQLite Database. It is small fast and efficient.

Thanks anyway, I will link to your post.

kovica September 15th, 2009 (#):

I'll go with H2 (http://www.h2database.com/html/main.html). It is much faster than HSQLDB. Look at http://www.h2database.com/html/performance.html

dbleyl September 16th, 2009 (#):

Another vote for H2.

Nicholas Bayborodin September 23rd, 2009 (#):

Thanks for article. Can you write about using DerbyDB?

ellis eghan July 21st, 2010 (#):

Thanks for the article. Is it possible to use the
JPA "entity" concept with HSQLDB? If yes, how can it be done?

sri September 29th, 2010 (#):

Hi,

I am a beginner to HSQL DB.
I have a doubt.
Like in postgres once we specify the IP address and port in the connection string of the database server, then any system can be able to access the database provided, the IP address of the client system needs to be entered in the configuration file.

Do we have any facility in HSQLDB , like any system can access the HSQL DB with the Server IP , Port no, user name and password to the database , without adding the
IP address of the Client System ,in the configuration file.

like once an IP address, port no , user name and password is known , is there any possibility to access the HSQL DB without restricting to Specific IP address ???

Can you pleas help….

Thank in advance,
Sri

HSQLDB | ingegno.it/eng February 3rd, 2011 (#):

[...] http://www.javablogging.com/embedded-database-in-java-use-of-hsqldb/ [...]

HSQLDB » ingegno.it - Just another developer weblog March 2nd, 2011 (#):

[...] http://www.javablogging.com/embedded-database-in-java-use-of-hsqldb/ [...]

HSQLDB « ingegno.it January 19th, 2012 (#):

[...] http://www.javablogging.com/embedded-database-in-java-use-of-hsqldb/ Share January 19th, 2012 | Category: Senza categoria [...]

HSQLDB Intro « ingegno.it January 20th, 2012 (#):

[...] http://www.javablogging.com/embedded-database-in-java-use-of-hsqldb/ [...]

تاپ لینک هایی در مورد برنامه نویسی February 5th, 2012 (#):

[...] Project manager و سوییچ کنیم؟فرق بین Git به Subversion چرا باید از HSQLDB با استفاده از Java شده در Embed کارهای جالبی می شود با عکس ها انجام داددیتابیس jQuery این 7 پلاگین [...]

Adrian March 11th, 2014 (#):

In oder to work you have to :

/* connection.prepareStatement(”drop table testtable;”).execute();*/
connection.prepareStatement(”create table testtable (id INTEGER, ” + “name VARCHAR(255));”).execute();

Sam September 7th, 2014 (#):

ResultSet should also be closed

Add your Comment!

Name*: Mail Adress*: Blog: Comment*:

Archives

- May 2010 (2)
- April 2010 (1)
- March 2010 (1)
- December 2009 (1)
- November 2009 (8)
- October 2009 (5)
- September 2009 (6)
- August 2009 (9)
- July 2009 (7)
- June 2009 (4)

Recent Posts

- Java ClassLoader (4) – Loading a custom ClassLoader on JVM start
- Java ClassLoader (3) – Namespaces
- Java ClassLoader (2) – Write your own ClassLoader
- Java ClassLoader (1) – What is a ClassLoader?
- Internationalization with MessageFormat
- Scripting in Java
- Simple guide to Java Message Service (JMS) using ActiveMQ
- A guide to Java SimpleDateFormat in examples
- Using Java SecurityManager to grant/deny access to system functions
- How to check if String is parseable to Integer or Double?

Tags

Atomic AtomicInteger autoboxing Bug Catalyst ClassLoader Close Code Quality Collection Comparable CompareTo Eclipse equals method File Final Finally FindBugs Generics Global Variable Hash Code Internationalization Java Compiler Lazy Initialization Locale Map Memory Leaks MessageFormat Micro Benchmarks Pattern Performance Regular Expression Remainder serialization Singleton Split String Substring Synchronization toArray Try/Catch Type Erasure Type safety Welcome wrapper classes xml

Contact us:

javablogging@javablogging.com