Java H2 tutorial

Java H2 tutorial shows how to do database programming in H2 with Java.

H2 is a relational database management system written in Java. It can be embedded in Java applications or run in the client-server mode. It can be used also in a memory mode.

H2 has a very small footprint. It somes with a browser based management application called H2 Console.

Downloading H2

From the H2's home page, we download the database in a ZIP file.

```
$ unzip h2-2017-06-10.zip
```

We unzip the archive.

```
$ mv h2 ~/bin/
```

We move the installation directory to a destination of our choice.

Starting H2 server

Now we are going to start H2 server.

```
$ java -jar bin/h2-1.4.196.jar -baseDir ~/tmp/h2dbs
Web Console server running at http://127.0.1.1:8082 (only local connections)
TCP server running at tcp://127.0.1.1:9092 (only local connections)
PG server running at pg://127.0.1.1:5435 (only local connections)
```

We move to the installation directory and run H2 in server mode. The command starts a web console application and two local connections; the PG server is a PostgreSQL compatibility mode with PostgreSQL protocol. The directory where the database files are generated is set to `~/tmp/h2dbs`, wher ~ stands for home directory.

We go to the web browser connect to the `testdb` database with `jdbc:h2:tcp://localhost:9092/testdb` URL. The database is generated in `~/tmp/h2dbs` directory. The default user is `sa` with no password set.

```
ALTER USER sa SET PASSWORD 's$cret'
```

While in the console, we set a password for user `sa` with the `ALTER USER` statement.

cars_h2.sql

```
CREATE TABLE cars(id INT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(150), price INT);
INSERT INTO cars(name, price) VALUES('Audi', 52642);
INSERT INTO cars(name, price) VALUES('Mercedes', 57127);
INSERT INTO cars(name, price) VALUES('Skoda', 9000);
INSERT INTO cars(name, price) VALUES('Volvo', 29000);
INSERT INTO cars(name, price) VALUES('Bentley', 350000);
INSERT INTO cars(name, price) VALUES('Citroen', 21000);
INSERT INTO cars(name, price) VALUES('Hummer', 41400);
INSERT INTO cars(name, price) VALUES('Volkswagen', 21600);
```

This is the SQL to create the `cars` table. We use this table in one example.

H2 Maven dependency

```
<dependencies>

    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <version>1.4.196</version>
    </dependency>

</dependencies>
```

This is the Maven dependency for H2.

Java H2 memory example

In the first example, we connect to an in-memory H2 database. The H2 server does not need to run for this example.

JavaSeH2Memory.java

```
package com.zetcode;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;

public class JavaSeH2Memory {

    public static void main(String[] args) {

        String url = "jdbc:h2:mem:";

        try (Connection con = DriverManager.getConnection(url);
                Statement st = con.createStatement();
                ResultSet rs = st.executeQuery("SELECT 1+1")) {

            if (rs.next()) {

                System.out.println(rs.getInt(1));
            }

        } catch (SQLException ex) {

            Logger lgr = Logger.getLogger(JavaSeH2Memory.class.getName());
            lgr.log(Level.SEVERE, ex.getMessage(), ex);
        }
    }
}
```

The example connects to an H2 in-memory database and executes a query. An in-memory private database for one connection only is created. The database is closed when the connection to the database is closed.

```
String url = "jdbc:h2:mem:";
```

This URL is for H2 database in memory mode.

Java H2 Server example

For this example, we start the H2 server with:

```
$ java -jar bin/h2-1.4.196.jar -baseDir ~/tmp/h2dbs
```

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>
    <groupId>com.zetcode</groupId>
    <artifactId>JavaSeH2Server</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>

    <dependencies>

        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <version>1.4.196</version>
        </dependency>

    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>exec-maven-plugin</artifactId>
                <version>1.6.0</version>
                <configuration>
                    <mainClass>com.zetcode.JavaSeH2Server</mainClass>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

The POM file contains the H2 Database Engine and the `exec-maven-plugin` for executing Java classes with Maven.

JavaSeH2Server.java

```
package com.zetcode;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;

public class JavaSeH2Server {

    public static void main(String[] args) {

        String url = "jdbc:h2:tcp://localhost:9092/testdb";
        String user = "sa";
        String passwd = "s$cret";

        String query = "SELECT * FROM cars";

        try (Connection con = DriverManager.getConnection(url, user, passwd);
                Statement st = con.createStatement();
                ResultSet rs = st.executeQuery(query)) {

            while (rs.next()) {

                System.out.printf("%d %s %d%n", rs.getInt(1),
                        rs.getString(2), rs.getInt(3));
            }

        } catch (SQLException ex) {

            Logger lgr = Logger.getLogger(JavaSeH2Server.class.getName());
            lgr.log(Level.SEVERE, ex.getMessage(), ex);
        }
    }
}
```

The example connects to the H2 server and executes a query. It returns all rows from the `cars` table.

```
String url = "jdbc:h2:tcp://localhost:9092/testdb";
```

This is the URL for connecting to the H2 Server's `testdb` database.

```
$ mvn compile
```

```
$ mvn -q exec:java
```

```
1 Audi 52642
2 Mercedes 57127
3 Skoda 9000
4 Volvo 29000
5 Bentley 350000
6 Citroen 21000
7 Hummer 41400
8 Volkswagen 21600
```

We compile and run the program.

Java H2 tutorial showed how to program H2 database in Java. You might be also interested in Derby tutorial, MySQL Java tutorial, RESTEasy H2 tutorial, and PostgreSQL Java tutorial.