Accessing a database from Java

Using JDBC

We've got a fuzzbox and we're gonna use it

Now we know a little about databases and SQL. So how do we access a database from a Java application?

There is an API for that, JDBC:

http://www.oracle.com/technetwork/java/javase/jdbc/index.html

What do we need in order to use SQLite3?

If we want to use our SQLite3 database from a Java application, we need a few things. First, we need a driver as a JAR file, and we need to make sure that it is on our CLASSPATH.

https://bitbucket.org/xerial/sqlite-jdbc/downloads

Putting the driver on the classpath

Let’s say we have the following directory structure:

```
.
|-- db
|   |-- app
|   |   `-- DatabaseTest.java
|   `-- main
|       `-- Main.java
|-- driver
|   `-- sqlite-jdbc-3.8.11.2.jar    <----- The driver
|-- my_municipalities
|-- my_municipalities2
`-- textfiles
    |-- HTTP-kommuner.txt
    |-- ... and more files
5 directories, 18 files
```

Putting the driver on the classpath

In order to use the driver in our Java application, we need to include it in the classpath:

```
java -cp .:driver/sqlite-jdbc-3.8.11.2.jar db.main.Main
```

Note: **sqlite-jdbc-3.8.11.2.jar** is the exact filename of the driver you downloaded

Loading the driver inside Java

```
static{
     try{
         Class.forName("org.sqlite.JDBC"); // Loads the class into the JVM
     }catch(ClassNotFoundException cnfe){
         System.err.println("Could not load driver: " + cnfe.getMessage());
     }
 }
```

Classes in the JDBC API

```
private static Connection con;
private final static String DB_CONN_STR="jdbc:sqlite:my_municipalities";

/* When the connection is created, the appropriate driver is chosen */
try{
       con = DriverManager.getConnection(DB_CONN_STR);
}catch(Exception e){
       System.err.println("Error getting connection to " +
                           DB_CONN_STR);
}
```

Interfaces in the JDBC API

```
/* import java.sql.*;  First in this class we import java.sql.*          */
Statement stm = null;
ResultSet rs  = null;
try{
    String query = "SELECT Name, HTTPS FROM municipalities LIMIT 5";
    stm          = con.createStatement();
    rs           = stm.executeQuery(query);
    while(rs.next()){
        System.out.println(rs.getString("Name") + " " +
                           (rs.getBoolean("HTTPS")?"HTTPS support":"HTTP only"));
    }
}catch(SQLException sqle){
    System.err.println(sqle.getMessage());
}finally{
    try{
        rs.close();
        stm.close();
    }catch(Exception e){}
}
```

Putting it all together

We've provided a small test class for the basics of JDBC:

- Loading the driver class
- Creating a `Connection`
- Creating a `Statement` using the `Connection`
- Executing a query with the `Statement` and getting a...
- `ResultSet` back
- Looping through the rows of the `ResultSet`
- Getting the values from each row

```
package db.app;
import java.sql.*;

public class DatabaseTest{
  private final static String DB_CONN_STR="jdbc:sqlite:my_municipalities";
    static{
      try{
        Class.forName("org.sqlite.JDBC");
      }catch(ClassNotFoundException cnfe){
        System.err.println("Could not load driver: "+cnfe.getMessage());
      }
    }

    private static Connection con;
    public DatabaseTest(){
      getConnection();
    }
    private void getConnection(){
      try{
        con = DriverManager.getConnection(DB_CONN_STR);
      }catch(Exception e){
        System.err.println("Error getting connection to " +
                           DB_CONN_STR);
      }
    }
```

Loading the driver class

Getting a connection

```
  public boolean hasConnection(){
    return con != null;
  }
  public void testQuery(){
    if(hasConnection()){
      Statement stm = null;
      ResultSet rs  = null;
      try{
        String query="SELECT Name, HTTPS FROM municipalities LIMIT 5";
        stm = con.createStatement();
        rs = stm.executeQuery(query);
        while(rs.next()){
          System.out.println(rs.getString("Name") + " " +
                             (rs.getBoolean("HTTPS")?"HTTPS support":"HTTP only"));
        }
      }catch(SQLException sqle){
        System.err.println(sqle.getMessage());
      }finally{
        try{
        rs.close();
        stm.close();
      }catch(Exception e){}
    }
  }
 }
}
```

Creating a Statement

Executing a query

Looping through the rows of the resultset and getting values from each row

The Main class

```
package db.main;
import db.app.DatabaseTest;

public class Main{
    public static void main(String[] args){
        System.out.println("Running db.Main...");
        DatabaseTest dt = new DatabaseTest();
        if(dt.hasConnection()){
            System.out.println("We have a connection");
        }else{
            System.out.println("There was a problem getting a connection.");
        }
        dt.testQuery();
    }
}
```

Running the code

```
$ javac db/main/Main.java && java -cp .:driver/sqlite-jdbc-3.8.11.2.jar db.main.Main
Running db.Main...
We have a connection
Ale kommun HTTP only
Alingsås kommun HTTPS support
Alvesta kommun HTTP only
Aneby kommun HTTP only
Arboga kommun HTTP only
```

How can this work?

How did Java know how to use the correct class?

The key lies in the Connection and the use of interfaces.

```
Class.forName("org.sqlite.JDBC"); // Will register with the DriverManager
...
con = DriverManager.getConnection(DB_CONN_STR);
// Returns a java.sql.Connection
// What is that?
public interface Connection
```

http://docs.oracle.com/javase/7/docs/api/index.html?java/sql/Connection.html

What interfaces exist in java.sql?

[...]
Connection
[...]
PreparedStatement (We'll look into that one later)
[...]
ResultSet
[...]
Statement
[...]

The driver implements all required interfaces

When we get a Connection object from the DriverManager, it actually asks the driver class which returns an instance of its implementation of the Connection interface.

Read

https://docs.oracle.com/javase/tutorial/jdbc/index.html

JDBC Introduction, and,

Lesson: JDBC Basics:

- **Getting Started**
- **Processing SQL Statements with JDBC**
- **Retrieving and Modifying Values from Result Sets**