Java and Database

YAO ZHAO

What is JDBC？

- JDBC stands for **J**ava **D**ata**b**ase **C**onnectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases.
- A database should implement the JDBC defined interfaces, then can be accessed through the JDBC interface.
- JDBC driver packages, usually are written by database vendors.
- Basically, common database, all can support Java langrage: Oracle、MySQL、SQLite……

Installation Guide

- SQLite
- Installing SQLite

Reference link: http://www.runoob.com/sqlite/sqlite-installation.html(Chinese)

Reference link: https://www.sqlitetutorial.net/download-install-sqlite/(English)

- Download sqlite jdbc

https://repo1.maven.org/maven2/org/xerial/sqlite-jdbc/3.34.0/

Download the latest version: sqlite-jdbc-3.34.0.jar

- MySQL
- Installing MySQL

Reference link： http://www.runoob.com/mysql/mysql-install.html(Chinese)

Reference link: https://dev.mysql.com/doc/refman/8.0/en/installing.html(English)

- Download MySQL jdbc: https://dev.mysql.com/downloads/connector/j/

SQLite download and installation

- https://www.sqlite.org/download.html
- For windows, download sqlite-tools-win32-x86-3350400.zip

How to import the jar

- Command Line：

1. CLASSPATH must be properly set, for instance on a Linux system or a Mac:

```
$ export CLASSPATH=.:sqlite-jdbc-version-number.jar
```

2. run the program with

```
$ java -cp  .:sqlite-jdbc-version-number.jar BasicJDBC
```

- eclipse：Java project->Properties->Java Build Path->Libraries->Add External Jars
- IntelliJ IDEA: File->Project Structure->dependencies->”+”->JARs or directories

JDBC Main tasks

- Making a connection to a database.
- Creating SQL or MySQL statements.
- Executing SQL or MySQL queries in the database.
- Viewing & Modifying the resulting records.

JDBC- Establishing a JDBC connection

- The programming involved to establish a JDBC connection is simple. Here are these simple four steps:
    1. **Import JDBC Packages:** Add **import** statements to your Java program to import required classes in your Java code.
    2. **Register JDBC Driver:** This step causes the JVM to load the desired driver implementation into memory so it can fulfill your JDBC requests.
    3. **Database URL Formulation:** This is to create a properly formatted address that points to the database to which you wish to connect.
    4. **Create Connection Object:** Finally, code a call to the *DriverManager* object's *getConnection( )* method to establish actual database connection.

1. Import JDBC Packages

```
import java.sql.* ; // for standard JDBC programs
```

2.Register JDBC Driver

```
try {
        Class.forName("org.sqlite.JDBC");
     } catch(Exception e) {
       System.err.println("Cannot find the driver.");
       System.exit(1);
     }
```

Other database drivers

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

Class.forName("com.mysql.jdbc.Driver");

Class.forName("org.postgresql.Driver");

Class.forName("org.sqlite.JDBC");

Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
```

3.Database URL Formulation

dbPath = “test.db” or “sampledb.sqlite”

String url = "jdbc:sqlite:" + dbPath

4.Create Connection Object

```
try {
        con = DriverManager.getConnection("jdbc:sqlite:" + dbPath);
        con.setAutoCommit(false);
        System.err.println("Successfully connected to the database.");
 } catch (Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
 }
```

Database URL Formulation

RDBMS	JDBC Diver Name	URL format
MySQL	com.mysql.jdbc.Driver	**jdbc:mysql:**//hostname/ databaseName
ORACLE	oracle.jdbc.driver.OracleDriver	**jdbc:oracle:thin:@**hostname:port Number:databaseName
DB2	COM.ibm.db2.jdbc.net.DB2Driver	**jdbc:db2:**hostname:port Number/databaseName
Sybase	com.sybase.jdbc.SybDriver	**jdbc:sybase:Tds:**hostname: port Number/databaseName

Creating SQL statements And Executing SQL queries in the database

```
          Statement stmt1;
          ResultSet rs1;
         stmt1 = con.createStatement();
          rs1 = stmt1.executeQuery(“select name from sqlite_master where
type='table'");
```

Viewing & Modifying the resulting records

```
while (rs1.next()) {
        System.out.println(rs1. getString(1));
}
rs1.close();
```

其他与数据库交互的方式

Interfaces	**Recommended Use**
Statement	Use this for general-purpose access to your database. Useful when you are using **static SQL** statements at runtime. The Statement interface **cannot accept parameters**.
PreparedStatement	Use this when you plan to use the SQL statements **many times**. The PreparedStatement interface **accepts input parameters at runtime**.
CallableStatement	CallableStatement Use this when you want to access the database **stored procedures**. The CallableStatement interface can also **accept runtime input parameters.**

Batch Processing

- Batch Processing allows you to group related SQL statements into a batch and submit them with one call to the database.When you send several SQL statements to the database at once, you reduce the amount of communication overhead, thereby improving performance.
  - JDBC drivers are not required to support this feature. You should use the *DatabaseMetaData.supportsBatchUpdates()* method to determine if the target database supports batch update processing. The method returns true if your JDBC driver supports this feature.
  - The **addBatch()** method of *Statement, PreparedStatement,* and *CallableStatement* is used to add individual statements to the batch. The **executeBatch()** is used to start the execution of all the statements grouped together.
  - The **executeBatch()** returns an array of integers, and each element of the array represents the update count for the respective update statement.
  - Just as you can add statements to a batch for processing, you can remove them with the **clearBatch()** method. This method removes all the statements you added with the addBatch() method. However, you cannot selectively choose which statement to remove.
- For more details, please see：https://www.tutorialspoint.com/jdbc/jdbc-batch-processing.htm

JDBC References

- https://www.tutorialspoint.com/jdbc/index.htm(English)
- http://wiki.jikexueyuan.com/project/jdbc/(Chinese)

SQLite References

- https://www.tutorialspoint.com/mysql/index.htm(English)
- http://www.runoob.com/sqlite/sqlite-tutorial.html(Chinese)