
MAHARISHI UNIVERSITY of MANAGEMENT
Engaging the Managing Intelligence of Nature
Computer Science Department
CS401 Modern Programming Practices (MPP)
Professor Paul Corazza

Lecture 10:Best Programming Practices with Java 8

Living Life in Accord with Natural Law

REVIEW: JDBC

JDBC

(Optional Module #3)

Executive Summary

Intro to JDBC:

1. JDBC is the mechanism for a Java application to communicate with a database
2. Description of JDBC and the architecture of a Java application that uses JDBC
3. Sample code showing how to connect to a database by way of JDBC

Review of JDBC
(Optional Module #3)

Another example of a resource that can be managed with `try-with-resources` is the `Connection` object, invoked in interacting with a database, using JDBC. Handling exceptions and closing the connection in the right way and in the right sequence has tended to be error-prone. Using `try-with-resources`, it is straightforward to write code in the correct way.

1. ***Review of JDBC.*** JDBC is a mechanism that makes it possible for a Java program to communicate with a database system (and other similar data sources) by way of SQL (structured query language) commands.
2. JDBC APIs facilitate
   - Making a connection to a database.
   - Creating SQL or MySQL statements.
   - Executing SQL or MySQL queries in the database.
   - Viewing and modifying the resulting records.

Review of JDBC (Optional Module #3) (cont.)

3. Simplified code sample (from Oracle tutorial)

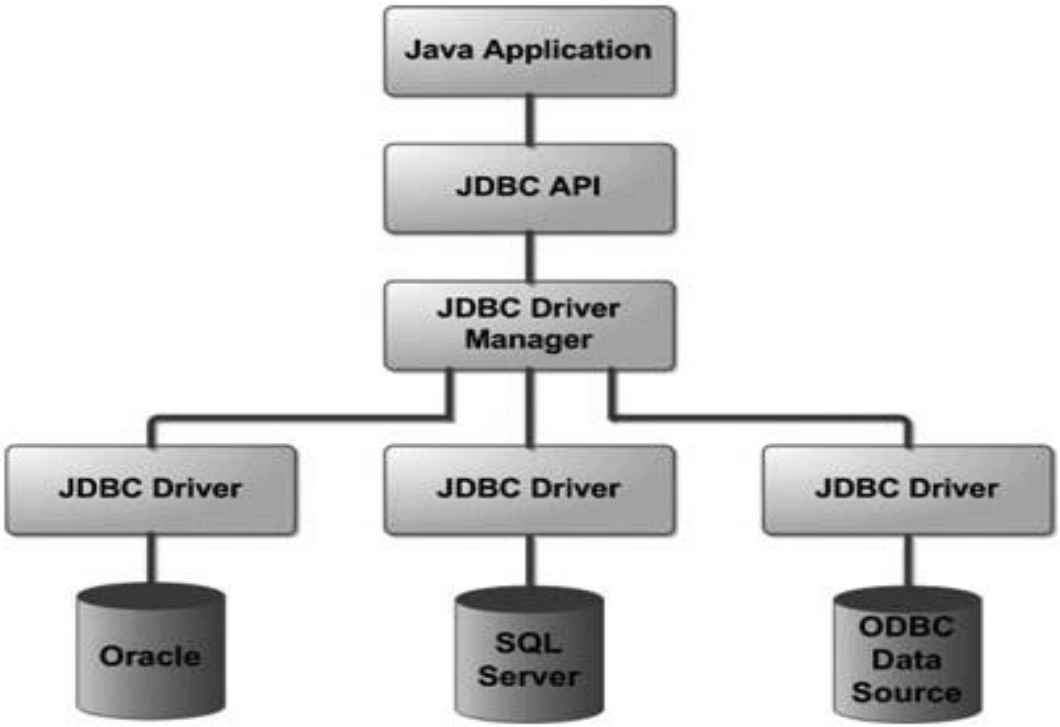
```
public void connectToAndQueryDatabase(String username, String password) {

    Connection con = DriverManager.getConnection(
                        "jdbc:myDriver:myDatabase",
                        username,
                        password);

    Statement stmt = con.createStatement();
    ResultSet rs = stmt.executeQuery("SELECT a, b, c FROM Table1");

    while (rs.next()) {
        int x = rs.getInt("a");
        String s = rs.getString("b");
        float f = rs.getFloat("c");
    }
}
```

Review of JDBC

(Optional Module #3) (cont.)

4. The JDBC Architecture

- **JDBC API:** This provides the application-to-JDBC Manager connection.
- **JDBC Driver API:** This supports the JDBC Manager-to-Driver Connection.



Advanced JDBC Programming

1. Begin with the example given in lesson10.lecture.jdbc.read_trywithres
2. As in the case of closing a Reader, if a SQLException is thrown in reading the database and another is thrown in attempting to close the Connection, the close exception is appended to the database exception as a suppressed exception.
3. The implementation style uses try-with-resources just to manage the Connection object; the steps of forming and executing a Statement are handled in an embedded try/catch block.

Handling Transactions and Auto-Generated Keys with JDBC

- Transactions enable you to control if, and when, changes are applied to the database. It treats a single SQL statement or a group of SQL statements as one logical unit, and if any statement fails, the whole transaction fails.
- To transaction support set the Connection object's autoCommit flag to false**.** For example, if you have a Connection object named conn, code the following to turn off auto-commit –

```
conn.setAutoCommit(false);
```

- Once you have executed your SQL code (for insertions, deletions,etc), you *commit* the changes with a call to the Connection object's **commit()** method like this:

```
conn.commit( );
```

- If an exception is thrown, you can rollback your changes to the database with a call to **rollback**:

```
conn.rollback( );
```

- Exception-handling for transaction management can be tricky, but try-with-resources simiplifies the steps.

DEMO: lesson10.lecture.jdbc.transact

Handling Transactions and Auto-Generated Keys with JDBC (cont.)

NOTES:

1. The demo illustrates the use of PreparedStatements. In a nutshell (for security reasons) whenever an SQL statement has parameters that need to be filled at runtime, the statement should be written using a PreparedStatement.
2. The Customer unique key field id is *auto-generated*. After you do an insert, you will often want to know what the value of the id that was generated by the database. The demo illustrates the technique for retrieving this value

RESOURCES:

JDBC and working with transactions are big topics. Here are two excellent follow-up resources:

I. http://docs.oracle.com/javase/tutorial/jdbc/basics/transactions.html
II. http://www.tutorialspoint.com/jdbc/index.htm

For practice with SQL, try:

I. http://www.w3schools.com/sql/sql_intro.asp