

JDBC

Important Packages:

javax.sql
java.sql

Key classes :

DriverManager
Connection
Statement
ResultSet
DataSource

Development Process:

1. Get a connection to database
2. Create a Statement object
3. Execute SQL query
4. Process Result Set

1. Get a connection to a database

- **jdbc:**<driver protocol>:<driver connection details>

ex. **jdbc:**mysql://localhost:3306/demodb

```
import java.sql.*  
  
...  
String dbUrl = "jdbc:mysql://localhost:3306/demo";  
String user = "student";  
String password = "student";  
  
Connection myConn =  
    DriverManager.getConnection(dbUrl,user,password);
```

2. Create a Statement object

The Statement object depends on connection

- It will be used later to execute SQL query

Statement myStmt = myConn.createStatement;

3. Execute SQL query

ResultSet myRs = myStmt.executeQuery('select * from employees');

4. Process Result Set

- Result Set is initially places before first row
- Method: ResultSet.next()
 - moves forward one row
 - returns true if there are more rows to process
- Looping through a result set:
- Collection of methods for reading data

```

ResultSet myRs = myStmt.executeQuery("select * from
employees");

while(myRs.next()){
    // read data from each row
}

```

- getXXX(columnName)
- getXXX(columnIndex) one-based

```

...
ResultSet myRs = myStamt.executeQuery("select * from
employess");

while (myRs.next()) {
    System.out.println(myRs.getString("last_name"));
    System.out.println(myRs.getStrig("first_name"));
}

```

Database Setup

Before run any java code must follow two steps:

- Step1: install MySQL
- Step2: Create New User: student

Step 1: Install MySQL

1. Goto: <http://dev.mysql.com/downloads>
2. Click on: [MySQL Community Server](#)
3. Download version for your Operating_system like mac or win(choose web installer or offline - i chose web)
4. no need to sigh up - no thanks, just start my download
5. in installer - choose developer default - next
6. choose MySql Workbench - Execute
7. default selection - execution - download the default required software - Next
8. In Type and Network - remain default settings -Next
9. In Accounts and Roles - make MySQL Root password - 12 char -strong - Next
10. In Windows Service - remain default - Next
11. In Apply Server Configuration - Execute
12. Finish- check - next - execute - installation complete - finish
13. Start mysql workbench for check - give root password - access to default tables and schemas

14. right click on tabel - Select Rows -Limit 1000 - Congratulation you have installed successfully!!

Step 2: Create New Database User

1. Open mysql workbench - double click on root user - type root password
2. should see Schemas tab - goto Management/Administration tab
3. click on Users and Privileges - click on Add Account
4. Fill details - Login Name: student; Limit to Hosts Matching: localhost; Password: student; Confirm Password: student;
5. click on the Administrative Roles tab - make sure DBA is selected - Apply
6. now we have new user created student
7. to verify - close Local instance 3306 near home icon
8. click on MySQL Connection + sign - fill the details - Connection Name: student; Username: student - Test Connection - enter the password student
9. if succeed - click on OK. - Congratulation you successfully added new user student!!!!

Download source Code:

<http://www.luv2code.com/downloads/udemy-jdbc/jdbc-source-code-v2.zip>

Setting Up Your Development Environment

- **System Requirements** - need to have MySQL Database installed
 - SQL GUI tool - MySQL Workbench
- **Loading sample database table** -
 - open mysql workbench - goto File -> Open SQL Script... -> downloaded code :sql folder->table-setup.sql - select all - run -> refresh database schema - see schema name : demo
- **installing JDBC Driver** -
 - Download jdbc driver: google - download mysql jdbc driver -> goto first link -> select platform independent - > download zip - no thanks, just start my download -> extract zip - mysql-connector-java-8.0.18.jar
- **Testing Database Connection**
 - Setup eclipse project : open eclipse - > create java project: jdbc-test -> paste mysql-connector-java-8.0.18.jar to root folder - > add jar in build path -> paste class JdbcTest.java - Run - See the message "Database connection successful!"

Getting rid of MySQL SSL Warning Message

JDBC and MySQL: Getting rid of the dreaded message

WARN: Establishing SSL connection without server's identity verification is not recommended

When connecting to a MySQL database, you may encounter this scary warning message.

WARN: Establishing SSL connection without server's identity verification is not recommended

Your app will continue to work fine ... it is just the MySQL database yelling at you.

Solution

To get rid of the warning message, append **?useSSL=false** to the end of your database connection string.

For example,

Replace – jdbc:mysql://localhost:3306/demo

With – jdbc:mysql://localhost:3306/demo**?useSSL=false**

Note that I appended **?useSSL=false** to the end.

That will get rid of the pesky message ... whew!

Insert Data

```
String dbUrl = "jdbc:mysql://localhost:3306/demo";
String user = "student";
String pass = "student";

Connection myConn = DriverManager.getConnection(dbUrl, user, pass);

Statement myStmt = myConn.createStatement();

int rowAffected = myStmt.executeUpdate(
    "insert into employees "+
    "(last_name, first_name, email, department, salary) "+
    "values "+
    "(`Khare`, `Raj`, `1.raj.khare@gmail.com`, `IT`, `52000.00`)");
```

Note: we can use executeUpdate() for - insert , update, delete

Update Data

```
String dbUrl = "jdbc:mysql://localhost:3306/demo";
String user = "student";
String pass = "student";

Connection myConn = DriverManager.getConnection(dbUrl, user, pass);

Statement myStmt = myConn.createStatement();

int rowAffected = myStmt.executeUpdate(
    "update employees "+
    "set email='raj.khare.jave@gmail.com' "+
    "where last_name='Khare' and first_name='Raj'");
```

Delete Data

```
String dbUrl = "jdbc:mysql://localhost:3306/demo";
String user = "student";
String pass = "student";

Connection myConn = DriverManager.getConnection(dbUrl, user, pass);

Statement myStmt = myConn.createStatement();

int rowAffected = myStmt.executeUpdate(
    "delete from employees "+
    "where last_name='Khare' and first_name='Raj'");
```

Prepared Statements

- **What are Prepared Statements**
- **Create a Prepared Statements**
- **Setting Parameter Values**
- **Execute a Prepared Statement**
- **Reusing a Prepared Statement**

What are Prepared Statements:

- A Prepared Statement is precompiled SQL statement
- Prepared Statement provides the following benefits
 - Makes it easier to set SQL parameter values
 - Prevent against SQL injection attack
 - May improve application performance
- SQL statement is precompiled

Using Prepared Statements

- Instead of hard coding your SQL values

```
select * from employees
where salary > 8000 and department='Legal'
```

- Set parameter placeholders
 - Use a question mark for placeholder: ?

```
select * from employees
where salary > ? and department=?
```

Create the prepared statement:

```
PreparedStatement myStmt =
    myConn.prepareStatement("select * from employees"+
        " where salary > ? and department=?");
```

```
myStmt.setDouble(1, 8000);
myStmt.setString(2, "Legal");

//now execute the query
ResultSet myRs = myStmt.executeQuery();
```


- Can also use prepared statement for
 - Insert, update and deletes

```
PreparedStatement myStmt =  
    myConn.prepareStatement("delete from employees" +  
        " where salary > ? and department=?");  
  
//set params  
myStmt.setDouble(1, 8000);  
myStmt.setString(2, "Legal");  
  
//execute statement  
int rowAffected = myStmt.executeUpdate();
```