Introducing to JDBC

Summary: in this tutorial, we will give you a very brief overview of JDBC so that you can use it for interacting with MySQL databases.

JDBC API provides a standard interface for interacting with any relational database management systems (RDBMS). JDBC API consists of the following main components:

1. JDBC Driver
2. Connection
3. Statement
4. ResultSet

Let's take a look at each component in more detail.

JDBC Driver

A JDBC driver is set of Java classes that implement JDBC interfaces for interacting with a specific database. Almost all database vendors such as MySQL, Oracle, Microsoft SQL Server, provide JDBC drivers. For example, MySQL provides a JDBC driver called MySQL Connection/J that allows you to work with MySQL database through a standard JDBC API.

There are three types of JDBC drivers including JDBC-native API Driver, JDBC-net Driver, and JDBC Driver.

> We will discuss about the JDBC driver, for more detailed information on the other driver type, you can check it out the JDBC driver.

JDBC Driver is written in pure Java. It translates JDBC calls into MySQL specific calls and sends the calls directly to a specific database. To use a JDBC driver, you need to include the driver JAR file with your application. MySQL Connector/J is the JDBC driver.

Connection

The first and most important component of JDBC is the Connection object. In a Java application, you first load a JDBC driver and then establish a connection to the database. Through the Connection object, you can interact with the database e.g., creating a Statement to execute SQL queries against tables. You can open more than one connection to a database at a time.

Statement

To execute a SQL query e.g., SELECT, INSERT, UPDATE, DELETE, etc., you use a `Statement` object. You create the Statement object through the `Connection` object. JDBC provides several types of statements for different purposes such as `PreparedStatement`, `CallableStatement`. We will cover the details of each object in the next tutorials.

ResultSet

After querying data from the database, you get a `ResultSet` object. The `ResultSet` object provides a set of API that allows you to traverse result of the query.
The typical flow of using JDBC is as follows:

The typical flow of using JDBC is as follows:

1. First, load the JDBC driver and create a connection to the database.
2. Then, create a `Statement` and execute the query to get a ResultSet.
3. Next, traverse and process the `ResultSet`.
4. Close the `ResultSet`, `Statement`, and `Connection`.

In this tutorial, we have introduced you to some basic information on JDBC and its main components: JDBC Driver, Connection, Statement, and ResultSet.

EXEMPLO QUERY:

```
package org.mysqltutorial;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

/**
*
* @author mysqltutorial.org
*/
public class Main {

  public static void main(String[] args) {
    //
    String sql = "SELECT first_name, last_name, email " +
            "FROM candidates";
```

```
    try (Connection conn = MySQLJDBCUtil.getConnection();
         Statement stmt  = conn.createStatement();
         ResultSet rs    = stmt.executeQuery(sql)) {

        // loop through the result set
        while (rs.next()) {
          System.out.println(rs.getString("first_name") + "\t" +
                             rs.getString("last_name")  + "\t" +
                             rs.getString("email"));

        }
    } catch (SQLException ex) {
        System.out.println(ex.getMessage());
    }
  }
}
```

.