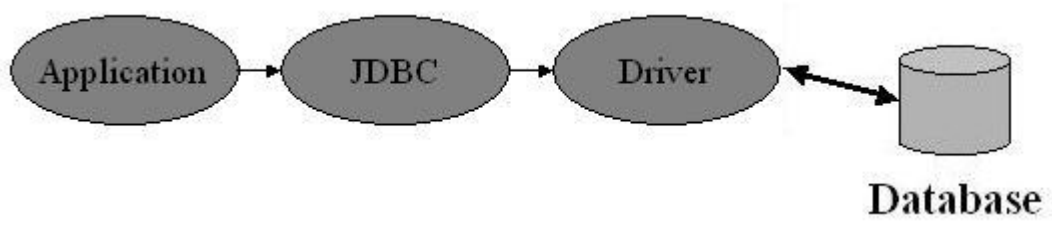
Simple tutorial for using JDBC

The JDBC (Java Database Connectivity) API defines interfaces and classes for writing database applications in Java by making database connections. Using JDBC you can send SQL, PL/SQL statements to almost any relational database. JDBC is a Java API for executing SQL statements and supports basic SQL functionality. It provides RDBMS access by allowing you to embed SQL inside Java code. Because Java can run on a thin client, applets embedded in Web pages can contain downloadable JDBC code to enable remote database access. You will learn how to create a table, insert values into it, query the table, retrieve results, and update the table with the help of a JDBC Program example.

Although JDBC was designed specifically to provide a Java interface to relational databases, you may find that you need to write Java code to access non-relational databases as well.

JDBC Architecture



Java application calls the JDBC library. JDBC loads a driver which talks to the database

In general, to process any SQL statement with JDBC, you follow these steps:

① Establishing a connection.
② Create a statement.
③ Execute the query.
④ Process the ***ResultSet*** object.
⑤ Close the connection.

1. Configuring a JDBC development environment

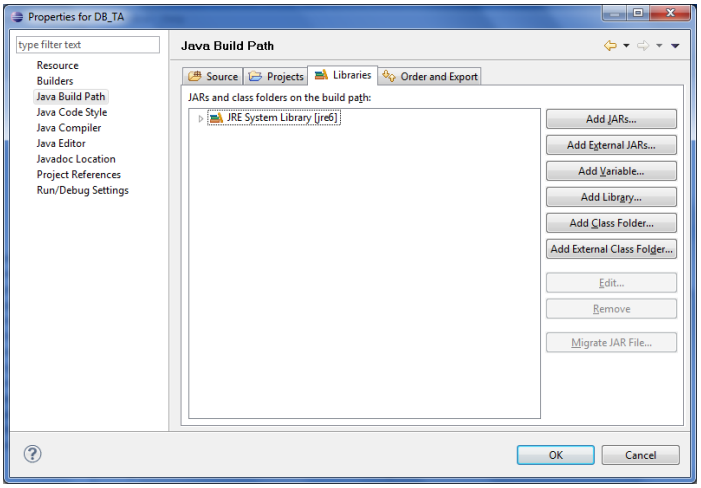
First of all, you should download the JDBC Driver for your DBMS. For this class, you can use "ojdbc6.jar" provided on class web-site. The ojdbc6.jar is a JDBC driver for Oracle Database 11g.

The below web sites are official pages for downloading official version of JDBC drivers.
Oracle : http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html
Mysql : http://www.mysql.com/downloads/connector/j/

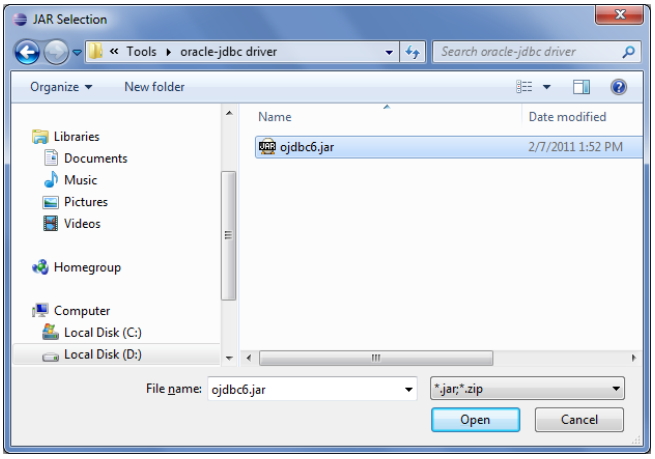
A. Install the latest version of the Java SE SDK on your computer.
B. Set up the classpath for a JDBC driver.
   The JDBC driver is not needed for compiling the java source but it is needed to execute the class. There are so many ways to set up this classpath according to the OS, programming tools and your preferences. This tutorial provides the case to use Eclipse.

If you use the eclipse, you can use following description.

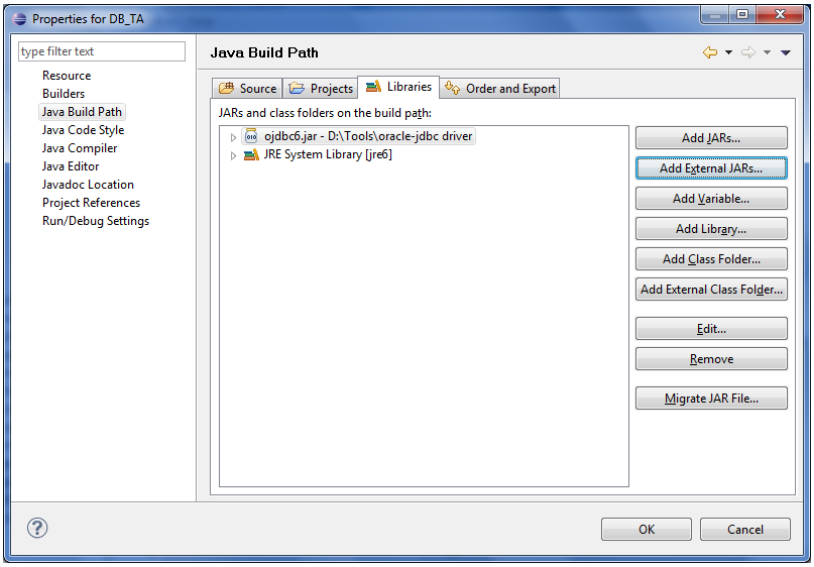
- Eclipse main menu-> Project -> Properties -> Java Build Path -> Librarie



- Click Add External JARs -> Select the JDBC driver. -> Open



- Click Add External JARs -> Select the JDBC driver. -> Open



- You can confirm the jar file is added then click OK.

2. Establishing a Connection

In this step of the jdbc connection process, we load the driver class by calling Class.forName() with the Driver class name as an argument. Once loaded, the Driver class creates an instance of itself.

Example for loading JDBC driver

```
String driverName = "oracle.jdbc.driver.OracleDriver"; // for Oracle
        // String driverName = "com.mysql.jdbc.Driver"; //for MySql
try {
        // Load the JDBC driver
        Class.forName(driverName);
} catch (ClassNotFoundException e) {
        // Could not find the database driver
        System.out.println("ClassNotFoundException : "+e.getMessage());
}
```

The JDBC DriverManager class defines objects which can connect Java applications to a JDBC driver. DriverManager is considered the backbone of JDBC architecture. DriverManager class manages the JDBC drivers that are installed on the system. Its getConnection() method is used to establish a connection to a database. It uses a username, password, and a jdbc url to establish a connection to the database and returns a connection object. A jdbc Connection represents a session/connection with a specific database. Within the context of a Connection, SQL, PL/SQL statements are executed and results are returned. An application can have one or more connections with a single database, or it can have many connections with different databases.

Example for JDBC connection

```
String serverName = "linux.grace.umd.edu";
String portNumber = "1521";
String sid = "dbclass1";

String url = "jdbc:oracle:thin:@" + serverName + ":" +
                        portNumber + ":" + sid; // for Oracle
//uri ="jdbc:mysql://server ip or address:port/database name"; //for Mysql
try {
        // Create a connection to the database
        connection = DriverManager.getConnection(url, username, password);

catch (SQLException e) {
        // Could not connect to the database
        System.out.println(e.getMessage());
}
```

Example for loading and connection

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;


public class TestCon{

Connection connection = null;

String driverName ="oracle.jdbc.driver.OracleDriver"; // for Oracle
        // String driverName = "com.mysql.jdbc.Driver"; //for MySql

String serverName = "ginger.umd.edu";    // Use this server.
String portNumber = "1521";
String sid = "dbclass1";

String url="jdbc:oracle:thin:@"+serverName+":"+ portNumber+":"+sid; // for Oracle
//uri ="jdbc:mysql://server ip or address:port/database name"; //for Mysql

String username = "your user name"; // You should modify this.
String password = "your password";    // You should modify this.

public TestCon() {}

public boolean doConnection(){
        try {
        // Load the JDBC driver
        Class.forName(driverName);
        // Create a connection to the database
        connection = DriverManager.getConnection(url, username, password);

        } catch (ClassNotFoundException e) {
        // Could not find the database driver
        System.out.println("ClassNotFoundException : "+e.getMessage());
                return false;
        } catch (SQLException e) {
        // Could not connect to the database
        System.out.println(e.getMessage());
                return false;
        }
        return true;
}

public static void main(String arg[]){
        TestCon con =new TestCon();
        System.out.println("Connection : " +con.doConnection());
}

}
```

If the output of the previous example is “Connection : true”, your environment setting and connection is correct. If not, try to check the jdbc driver classpath, your username and your password

3. Creating statements & executing queries

A Statement is an interface that represents a SQL statement. You execute Statement objects, and they generate ResultSet objects, which is a table of data representing a database result set. You need a Connection object to create a Statement object.

There are three different kinds of statements.

- Statement: Used to implement simple SQL statements with no parameters.
- PreparedStatement: (Extends Statement.) Used for precompiling SQL statements that might contain input parameters. See Using Prepared Statements for more information.
- CallableStatement: (Extends PreparedStatement.) Used to execute stored procedures that may contain both input and output parameters. See Stored Procedures for more information.

Example for Statement

```
Statement stmt = null;
try {
        stmt = connection.createStatement();
} catch (SQLException e ) {
        System.out.println(e.getMessage());
}
```

To execute a query, call an execute method from Statement such as the following:

- execute: Returns true if the first object that the query returns is a ResultSet object. Use this method if the query could return one or moreResultSet objects. Retrieve the ResultSet objects returned from the query by repeatedly calling Statement.getResutSet.
- executeQuery: Returns one ResultSet object.
- executeUpdate: Returns an integer representing the number of rows affected by the SQL statement. Use this method if you are using INSERT,DELETE, or UPDATE SQL statements.

Example for executeQuery

```
String country=”D”;
Statement stmt = null;
String query = "  SELECT * FROM CITY WHERE country=”’+country+”’”;
try {
        stmt = connection.createStatement();
        ResultSet rs = stmt.executeQuery(query);
        while (rs.next()) {
                String name = rs.getString(1); // or rs.getString("NAME");
                String coun= rs.getString(2);
                String province = rs.getString(3);
                int population = rs.getInt(4);
        }
        stmt.close();
} catch (SQLException e ) {
```

```
        System.out.println(e.getMessage());
}
```

The re.next() return ‘true’ until there is no more result.

Example for executeQuery

```
String population=”1705000”;
String cityName=”Hamburg”;
String province=”Hamburg”;
Statement stmt = null;
try {
          stmt = connection.createStatement();
        String sql = "UPDATE CITY SET population=”'+ population +”' WHERE NAME=”'+
        cityName +”' AND PROVINCE=”’+ province +”’";
        stmt.executeUpdate(sql);s
          stmt.close();
} catch (SQLException e ) {
System.out.println(e.getMessage());
}
```

Exmaple of simple JDBC program

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


public class TestCon{

        Connection connection = null;

        String driverName ="oracle.jdbc.driver.OracleDriver"; // for Oracle
        // String driverName = “com.mysql.jdbc.Driver”; //for MySql

        String serverName = "ginger.umd.edu";
        String portNumber = "1521";
        String sid = "dbclass1";

        String url = "jdbc:oracle:thin:@" + serverName + ":" + portNumber + ":" + sid; // for
Oracle
        //uri =”jdbc:mysql://server ip or address:port/database name”; //for Mysql

        String username = "XXXXXXXX"; //your username
        String password = "XXXXXXXX"; //your password
        public TestCon()      {
```

```
	}
	public boolean doConnection(){
		try {
		// Load the JDBC driver
		Class.forName(driverName);
		// Create a connection to the database
		connection = DriverManager.getConnection(url, username, password);

		} catch (ClassNotFoundException e) {
		// Could not find the database driver
			System.out.println("ClassNotFoundException : "+e.getMessage());
			return false;
		} catch (SQLException e) {
		// Could not connect to the database
			System.out.println(e.getMessage());
			return false;
		}
		return true;
	}
	public void printCounryByCapital(String capital) throws SQLException{

		Statement stmt = null;
		String query = "SELECT * FROM COUNTRY WHERE
CAPITAL="'+capital+'"";

		stmt = connection.createStatement();
		ResultSet rs = stmt.executeQuery(query);
		while (rs.next()) {
			String name = rs.getString(1); // or rs.getString("NAME");
			String code= rs.getString(2);
			String cap = rs.getString(3);
			String province = rs.getString(4);
			int area = rs.getInt(5);
			int population = rs.getInt(6);

			System.out.println(" Name : "+name);
			System.out.println(" Code : "+code);
			System.out.println(" Capital : "+cap);
			System.out.println(" Province : "+province);
			System.out.println(" Area : "+area);
			System.out.println(" Population : "+population);
		}
	}

	public void printCityByCountry(String country) throws SQLException{

	Statement stmt = null;
	String query = " SELECT * FROM CITY WHERE country="'+country+'"";
	stmt = connection.createStatement();
		ResultSet rs = stmt.executeQuery(query);
		while (rs.next()) {
			String name = rs.getString(1); // or rs.getString("NAME");
			String coun= rs.getString(2);
			String province = rs.getString(3);
```

```
                    int population = rs.getInt(4);

                    System.out.println(" Name : "+name);
                System.out.println(" Country : "+coun);
                System.out.println(" Province : "+province);
                System.out.println(" Population : "+population);
            }
            stmt.close();

    }
    public void updateCityPopulation(String cityName,String province,
                    String population)throws SQLException
    {
            Statement stmt = null;
            stmt = connection.createStatement();
            String sql = "UPDATE CITY SET population="'+ population
                                    +"' WHERE NAME="'+cityName +"' AND
PROVINCE="'+ province +"'";
            stmt.executeUpdate(sql);
            stmt.close();
    }

    public static void main(String arg[]){
            TestCon con =new TestCon();
            System.out.println("Connection : " +con.doConnection());
            try{
            con.printCounryByCapital("Paris");
            con.printCityByCountry("D");
            con.updateCityPopulation("Munich","Bayern","3000");
            }catch(SQLException ex){System.out.println(ex.getMessage());}
    }
}
```