

Java Database Connectivity (JDBC) is a Java-based data access technology that defines how a client may access a database. It provides methods for querying and updating data in a database. The JDBC classes are contained in the Java package `java.sql` and `javax.sql`.

In this JDBC Oracle connectivity example, we will see how to setup a JDBC development environment and create a simple Java database application to connect to Oracle Database Express Edition using JDBC API. We will also see the following important things, which are required for connecting to any database using JDBC.

1. Oracle JDBC Connector jar
2. JDBC Oracle Driver class
3. JDBC Oracle Connection String URL

SETUP A DATABASE CREATE DATABASE USER

To create database objects, we must create at least one database user. A user is associated with a database schema, you connect to the database as a database user, and the database user is the owner of any database objects (tables, views etc) that you create in the schema associated with the user.

For example, to create a database user named 'testuser'. Follow these steps, using the command line:

1. Open the SQL command prompt window. For example, on Windows, click Start, then Programs (or All Programs), then Oracle Database Express Edition, and then "Run SQL Command Line".

2. Connect as the SYSTEM user:

Type:

connect

Enter user-name: system

Enter password:

The password is the one you entered during installation.

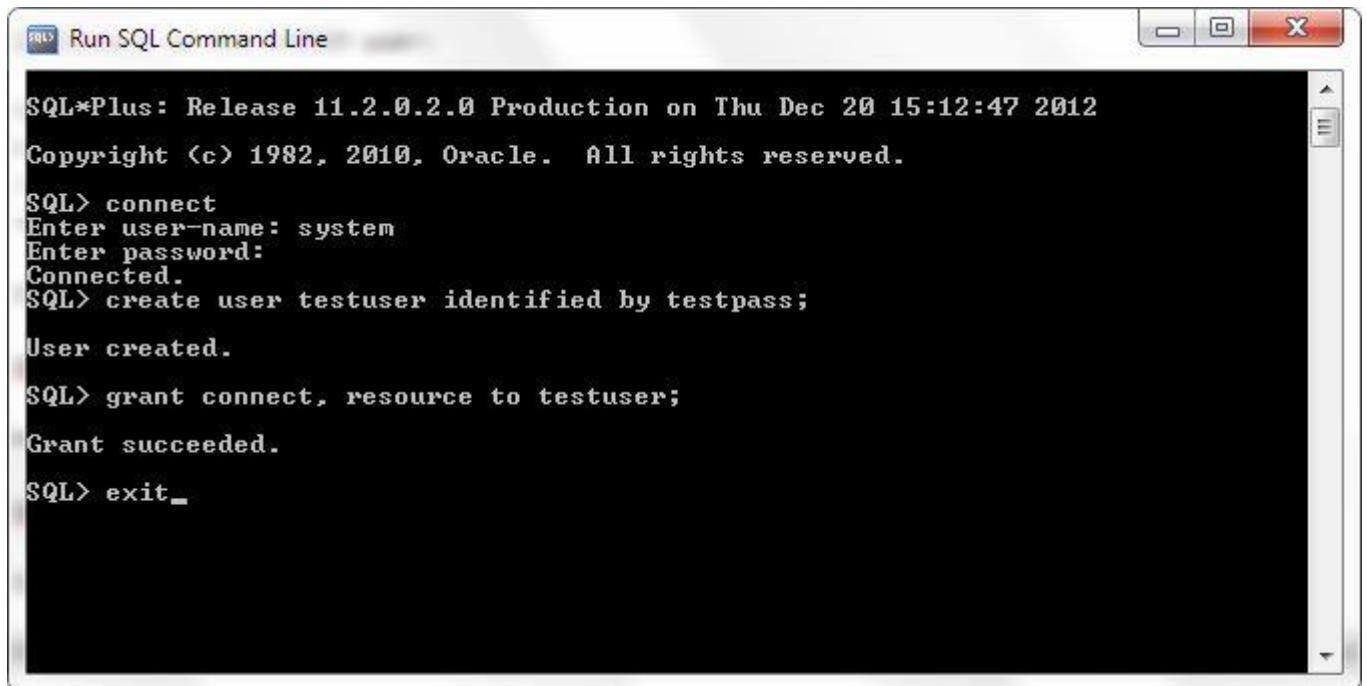
3. Create the user. For example, enter a statement in the following form:

```
SQL> create user testuser identified by <password-for-testuser>;
```

4. Grant the user the necessary privileges. For example:

```
SQL> grant connect, resource to testuser;
```

5. exit

A screenshot of a Windows-style window titled "Run SQL Command Line". The window has a black background with white text. The text inside the window shows the following sequence of commands and responses:
SQL*Plus: Release 11.2.0.2.0 Production on Thu Dec 20 15:12:47 2012
Copyright (c) 1982, 2010, Oracle. All rights reserved.
SQL> connect
Enter user-name: system
Enter password:
Connected.
SQL> create user testuser identified by testpass;
User created.
SQL> grant connect, resource to testuser;
Grant succeeded.
SQL> exit_
The window has standard Windows window controls (minimize, maximize, close) in the top right corner and a vertical scrollbar on the right side.

CREATE TABLE

Now let us login to the database with the newly created user ‘testuser’ and create a simple ‘Person’ table.

1. Open SQL Command Line.

2. Type “connect”

3. Enter username as “testuser”

4. Enter password as “testpass” (or the password you entered in the previous step while creating the user)

5. Create a person table.

```
SQL> create table person(pid integer primary key, name varchar2(50));
```

6. Enter a few data into the table.

```
SQL> insert into person values(1, 'Ram');
```

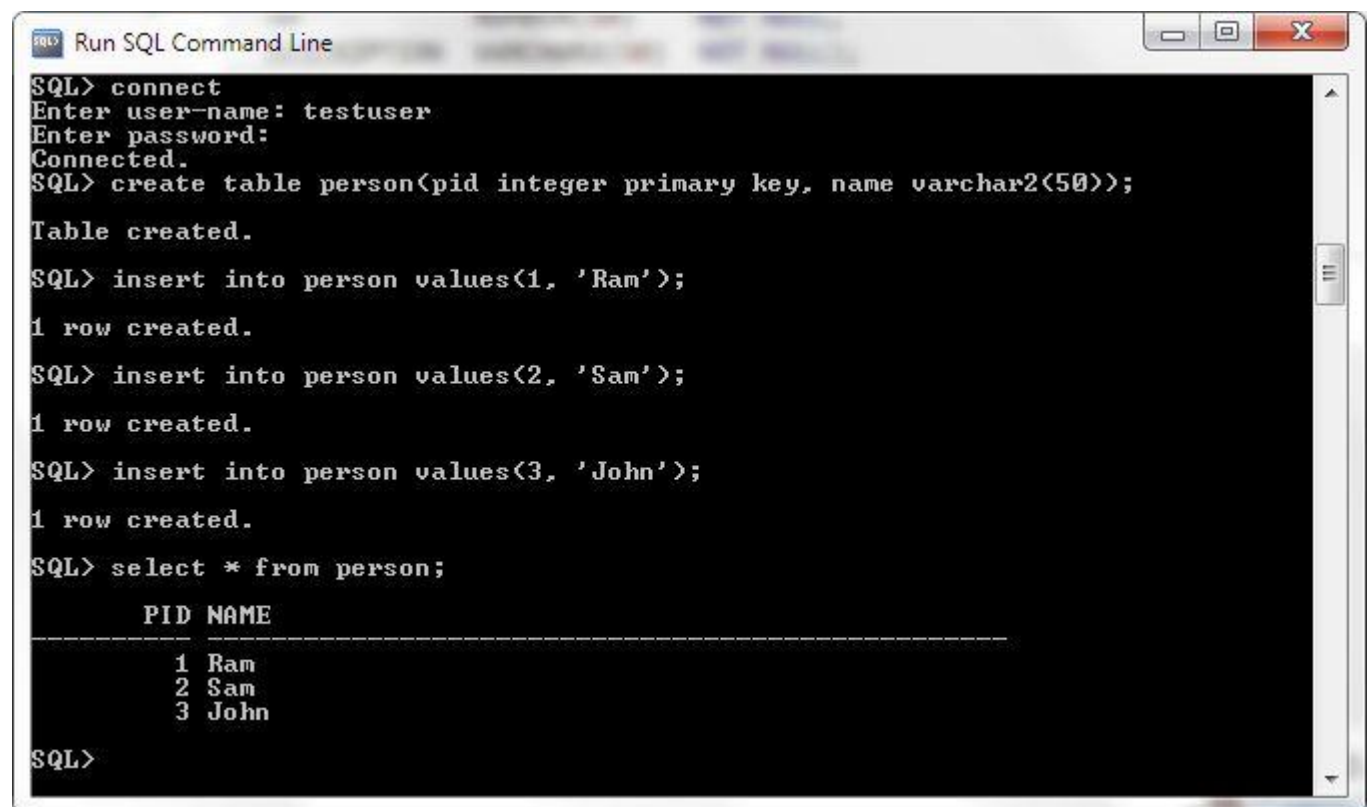
```
SQL> insert into person values(2, 'Sam');
```

```
SQL> insert into person values(3, 'John');
```

```
SQL> select * from person;
```

```
SQL> commit;
```

```
SQL> exit;
```

A screenshot of a Windows-style window titled "Run SQL Command Line". The window has a black background with white text. The text shows a sequence of SQL commands and their outputs. The commands include connecting as 'testuser', creating a table 'person' with columns 'pid' (integer primary key) and 'name' (varchar2(50)), inserting three rows of data (1, 'Ram'), (2, 'Sam'), and (3, 'John'), and finally selecting all data from the 'person' table. The output of the select statement is displayed in a table format with columns 'PID' and 'NAME'.

```
SQL> connect
Enter user-name: testuser
Enter password:
Connected.
SQL> create table person(pid integer primary key, name varchar2(50));
Table created.
SQL> insert into person values(1, 'Ram');
1 row created.
SQL> insert into person values(2, 'Sam');
1 row created.
SQL> insert into person values(3, 'John');
1 row created.
SQL> select * from person;

  PID NAME
-----
    1 Ram
    2 Sam
    3 John

SQL>
```

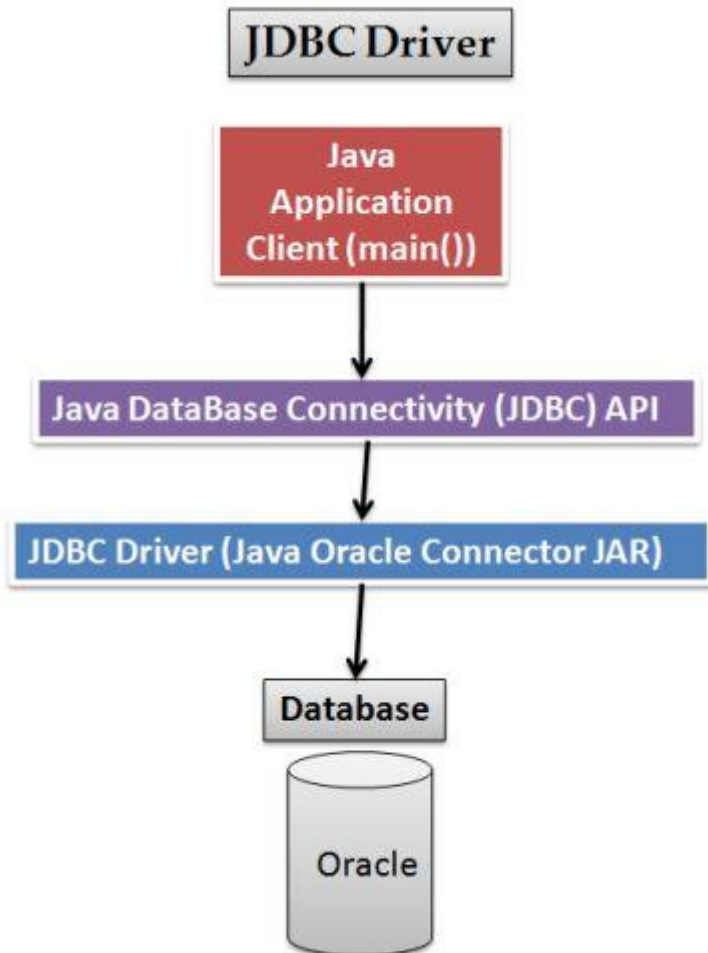
Now the database and table setup is done. Let us connect to this database table and retrieve the data using JDBC API.

JAVA ORACLE CONNECTOR

JDBC API mostly consists of interfaces which work independently of any database. A database specific driver is required for each database which implements the JDBC API.

The JDBC database Connector provides access to the database. To reach the database using JDBC we need a JDBC driver from the database provider in our case – Oracle. This connector is typically delivered with the product in a jar or zip

file or available in the provider's website. These files must be in our classpath (which is explained later under Configure JDBC Driver in Eclipse) otherwise we will get some class-not-found-exceptions indicating that the driver was not found on the classpath.



Oracle provides the JDBC connector jar with the product and is available in the following location in Windows,

C:\oraclexe\app\oracle\product\11.2.0\server\jdbc\lib

(if you followed the default installation procedure)

There will be multiple jars like,

ojdbc5.jar – Classes for use with JDK 1.5.

ojdbc6.jar – Classes for use with JDK 1.6.

We will be using “ojdbc6.jar”

CREATE A JAVA PROJECT IN ECLIPSE IDE

- Open Eclipse IDE.
- Create a new Java Project and name it as **JDBCOracle**. If you are a newbie, [refer this link](#) on getting started with Java and Eclipse.

JDBC ORACLE CONNECTIVITY

In order to establish a connection to the database using JDBC we need to perform the following steps,

1. Import the required interfaces/classes from java.sql package.
2. Load the JDBC Oracle Driver class
3. Establish the connection by providing the jdbc oracle connection string url

LOAD ORACLE JAVA DRIVER

We need to know and specify which of the classes in the connector jar implements the JDBC driver so as to load the class in memory. For Oracle the class oracle.jdbc.driver.OracleDriver is the jdbc driver class. The statement `Class.forName (“oracle.jdbc.driver.OracleDriver”)`

loads the driver class in memory.

JDBC ORACLE CONNECTION URL

We connect to Oracle database from Java using DriverManager class by calling `DriverManager.getConnection()` method. This method requires JDBC Oracle connection URL string, Oracle database username and password. The Java database connection string URL is of the following format:

```
jdbc:oracle::[username/password]@[//]host_name[:port]/[XE]
```

In this URL:

// is optional.

:port is optional.Specify this only if the default Oracle Net listener port (1521) is not used.

/XE, or the service name, is not required.

The connection adapter for the Oracle Database XE Client connects to the default service on the host.

Default service is a new feature of Oracle Database XE. If you use any other Oracle Database client to connect to Oracle Database XE, then you must specify the service name.

For example, if you connect to a local database using default port number, then the Oracle database connection URL is:

`jdbc:oracle:thin:testuser/testpass@localhost`

JAVA APPLICATION CODE

An application involving Java with database to process any SQL statement must follow these steps:

1. Establish a connection. (This is done by DriverManager class)
2. Create a Statement object. (Line 17)
3. Execute the query. (Line 18)
4. Process the ResultSet object. This is required only for SELECT SQL query. (Line 19-22)
5. Close the connection. (Line 29-31)

To do the above steps create a package **“com.theopentutorials.jdbc.oracle”**. Then create a class **“TestOracleJDBC”** with main method and copy the following code.

```
1 package com.theopentutorials.jdbc.oracle;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.ResultSet;
6 import java.sql.SQLException;
7 import java.sql.Statement;
8
9 public class TestOracleJDBC {
10     public static void main(String[] args) {
11         Connection con = null;
12         Statement stmt = null;
13         ResultSet rs = null;
14         try {
15             Class.forName("oracle.jdbc.driver.OracleDriver");
16             con =
17 DriverManager.getConnection("jdbc:oracle:thin:testuser/testpass@localhost");
18             stmt = con.createStatement();
19             rs = stmt.executeQuery("SELECT * FROM person");
20             while(rs.next()) {
21                 System.out.print(rs.getInt(1) + "\t");
22                 System.out.println(rs.getString(2));
23             }
24         } catch (ClassNotFoundException e) {
25             e.printStackTrace();
26         } catch (SQLException e) {
27             e.printStackTrace();
28         } finally {
29             try {
```

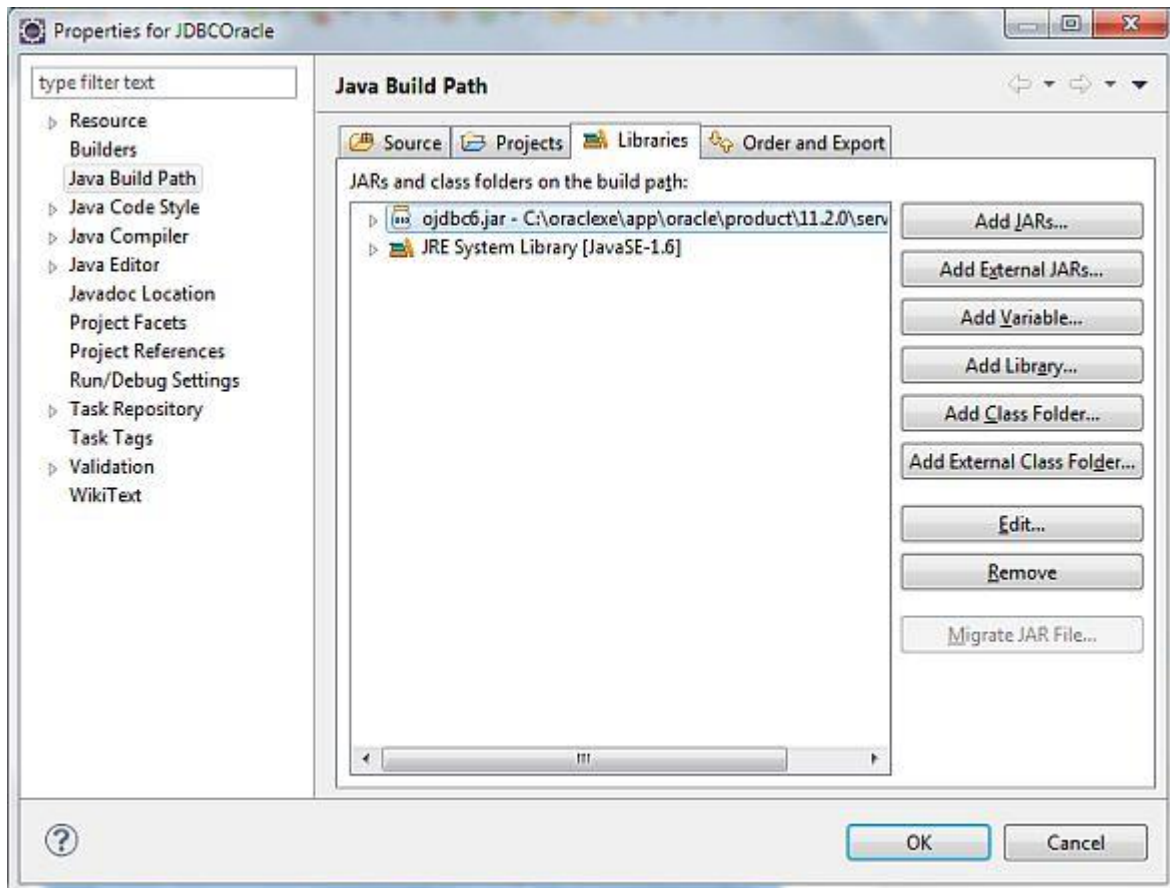
```
25         rs.close();
26         stmt.close();
27         con.close();
28     } catch (SQLException e) {
29         e.printStackTrace();
30     }
31 }
32
33
34
35
36
37
```

CONFIGURE JDBC DRIVER IN ECLIPSE IDE

If you run the above class you will get a runtime exception mentioning Driver class not found as shown below

```
java.lang.ClassNotFoundException: oracle.jdbc.driver.OracleDriver
```

Because we need to add the Java Oracle Connector JAR in project's classpath. To do this, right click on your Java Project -> Properties -> Buildpath -> Libraries -> Add External JAR and select the odbc6.jar file.



OUTPUT

Run the above program to get the following output.

```

Console  @ Javadoc  Declaration
<terminated> TestOracleJDBC [Java Application] C:\Program File
1      Ram
2      Sam
3      John
|

```

JDBC ORACLE APPLICATION FOLDER STRUCTURE

