建立JDBC连接所涉及的编程相当简单。 以下是基本的四个步骤 -

- **导入JDBC包：使用Java语言的import语句在Java代码开头位置导入所需的类。**
- **注册JDBC驱动程序：使JVM将所需的驱动程序实现加载到内存中，从而可以满足JDBC请求。**
- **数据库URL配置：创建一个正确格式化的地址，指向要连接到的数据库(如：MySQL,Oracle和MSSQL等等)。**
- **创建连接对象：最后，调用DriverManager对象的getConnection()方法来建立实际的数据库连接。**

1. 导入JDBC包

import语句告诉Java编译器在哪里找到在代码中引用的类，import语句一般放置在源代码的开头。

要使用标准的JDBC包，它允许在数据库表中执行选择，插入，更新和删除数据，请将以下导入添加到源代码中 -

```
import java.sql.* ;  // for standard JDBC programs
import java.math.* ; // for BigDecimal and BigInteger support
```

2. 注册JDBC驱动程序

在使用程序之前，必须先注册该驱动程序。 注册驱动程序是将Oracle驱动程序的类文件加载到内存中的过程，因此可以将其用作JDBC接口的实现。

只需在程序中一次注册就可以。可以通过两种方式之一来注册驱动程序。

2.1 方法I - Class.forName()

注册驱动程序最常见的方法是使用Java的Class.forName()方法，将驱动程序的类文件动态加载到内存中，并将其自动注册。这个方法是推荐使用的方法，因为它使驱动程序注册可配置和便携。

以下示例使用Class.forName()注册Oracle驱动程序 -

```
try {
   Class.forName("oracle.jdbc.driver.OracleDriver");
}
catch(ClassNotFoundException ex) {
   System.out.println("Error: unable to load driver class!");
   System.exit(1);
}
```

使用JDBC驱动程序连接MySQL数据库的示例代码片段 -

```
Class.forName("com.mysql.jdbc.Driver");
Connection conn = null;
conn =
DriverManager.getConnection("jdbc:mysql://hostname:port/db_name","db_username",
"db_password");
conn.close();
```

使用getInstance()方法来解决不合规的JVM，但是必须编写两个额外的异常，如下所示：

```
try {
   Class.forName("oracle.jdbc.driver.OracleDriver").newInstance();
}
catch(ClassNotFoundException ex) {
   System.out.println("Error: unable to load driver class!");
   System.exit(1);
catch(IllegalAccessException ex) {
   System.out.println("Error: access problem while loading!");
   System.exit(2);
catch(InstantiationException ex) {
   System.out.println("Error: unable to instantiate driver!");
   System.exit(3);
}
```

2.2 方法II - DriverManager.registerDriver()

第二种方法是使用静态DriverManager.registerDriver()方法来注册驱动程序。

如果使用的是非JDK兼容的JVM(如Microsoft提供的)，则应使用registerDriver()方法。

以下示例使用registerDriver()注册Oracle驱动程序 -

```
try {
   Driver myDriver = new oracle.jdbc.driver.OracleDriver();
   DriverManager.registerDriver( myDriver );
}
catch(ClassNotFoundException ex) {
   System.out.println("Error: unable to load driver class!");
   System.exit(1);
}
```

数据库URL配置

加载驱动程序后，可以使用DriverManager.getConnection()方法建立连接。 为了方便参考，这里列出三个重载的DriverManager.getConnection()方法 -

- getConnection(String url)
- getConnection(String url, Properties prop)
- getConnection(String url, String user, String password)

这里每个格式都需要一个数据库URL。 数据库URL是指向数据库的地址。

制定数据库URL是建立连接相关联的大多数错误问题发生的地方。

下表列出了常用的JDBC驱动程序名称和数据库URL。

RDBMS	JDBC驱动程序名称	URL格式
MySQL	com.mysql.jdbc.Driver	jdbc:mysql://hostname/databaseName

ORACLE	oracle.jdbc.driver.OracleDriver	jdbc:oracle:thin:@hostname:portNumber:databaseName
PostgreSQL	org.postgresql.Driver	jdbc:postgresql://hostname:port/dbname
DB2	com.ibm.db2.jdbc.net.DB2Driver	jdbc:db2:hostname:port Number/databaseName
Sybase	com.sybase.jdbc.SybDriver	jdbc:sybase:Tds:hostname: portNumber/databaseName

URL格式的所有突出部分都是静态的，只需要根据数据库设置更改对应的部分。

创建连接对象

上面列出了三种形式的DriverManager.getConnection()方法来创建一个连接对象。

使用具有用户名和密码的数据库URL

getConnection()最常用的形式要求传递数据库URL，用户名和密码：

假设使用Oracle thin驱动程序，那么需要为URL的数据库部分指定：

host:port:databaseName值。

如果主机名为amrood的TCP/IP地址为*192.0.0.10*，并且Oracle侦听器配置为侦听端口1521，并且要连接的数据库名称是EMP，则完整的数据库URL将是 -

```
jdbc:oracle:thin:@amrood:1521:EMP
// 或者
jdbc:oracle:thin:@192.0.0.10:1521:EMP
```

现在必须使用适当的用户名和密码调用getConnection()方法获取一个Connection对象，如下所示：

```
String URL = "jdbc:oracle:thin:@amrood:1521:EMP";
// String URL = "jdbc:oracle:thin:@192.0.0.10:1521:EMP";
String USER = "username";
String PASS = "password"
Connection conn = DriverManager.getConnection(URL, USER, PASS);
```

仅使用数据库URL

DriverManager.getConnection()方法的第二种形式只需要数据库URL -

```
DriverManager.getConnection(String url);
```

但是，在本示例中，数据库URL包括用户名和密码，并具有以下一般形式 -

```
jdbc:oracle:driver:username/password@database
```

所以，上述连接可以使用如下方式创建 -

```
String URL = "jdbc:oracle:thin:username/password@192.168.0.10:1521:EMP";
Connection conn = DriverManager.getConnection(URL);
```

使用数据库URL和Properties对象

DriverManager.getConnection()方法的第三种形式需要一个数据库URL和一个Properties对象 -
DriverManager.getConnection(String url, Properties info);

Properties对象包含一组键-值对。 在调用getConnection()方法时，它用于将驱动程序属性传递给驱动程序。
要进行与上述示例相同的连接，请使用以下代码 -

```
import java.util.*;

String URL = "jdbc:oracle:thin:@amrood:1521:EMP";
Properties info = new Properties( );
info.put( "user", "root" );
info.put( "password", "password12321" );

Connection conn = DriverManager.getConnection(URL, info);
```

关闭JDBC连接

在JDBC程序结束之后，显式地需要关闭与数据库的所有连接以结束每个数据库会话。 但是，如果在编写程序中忘记了关闭也没有关系，Java的垃圾收集器将在清除过时的对象时也会关闭这些连接。
依靠垃圾收集，特别是数据库编程，是一个非常差的编程实践。所以应该要使用与连接对象关联的close()方法关闭连接。
要确保连接已关闭，可以将关闭连接的代码中编写在“finally”块中。 一个finally块总是会被执行，不管是否发生异常。
要关闭上面打开的连接，应该调用close()方法如下 -

```
conn.close();
```

SQL

参考Oracle+JDBC示例代码：

```
package com.yiibai;

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

public class OracleJDBCExample {
    public static void main(String[] argv) {
        System.out.println("-------- Oracle JDBC Connection Testing ------");
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
```

```
        } catch (ClassNotFoundException e) {
            System.out.println("Where is your Oracle JDBC Driver?");
            e.printStackTrace();
            return;
        }
        System.out.println("Oracle JDBC Driver Registered!");
        Connection connection = null;
        try {
            connection = DriverManager.getConnection(
                    "jdbc:oracle:thin:@localhost:1521:xe", "system",
"password");

        } catch (SQLException e) {
            System.out.println("Connection Failed! Check output console");
            e.printStackTrace();
            return;
        }
        if (connection != null) {
            System.out.println("You made it, take control your database now!");
        } else {
            System.out.println("Failed to make connection!");
        }
    }
}
```