1、加载JDBC驱动程序：

 在连接数据库之前，首先要加载想要连接的数据库的驱动到JVM（Java虚拟机），

  这通过java.lang.Class类的静态方法forName(String  className)实现。

例如：

**try**{

    //加载MySql的驱动类

    Class.forName("com.mysql.jdbc.Driver") ;

    }**catch**(ClassNotFoundException e){

    System.out.println("找不到驱动程序类 ，加载驱动失败！");

    e.printStackTrace() ;

    }

   成功加载后，会将Driver类的实例注册到DriverManager类中。

2、提供JDBC连接的URL

   •连接URL定义了连接数据库时的协议、子协议、数据源标识。

    •书写形式：协议：子协议：数据源标识

    协议：在JDBC中总是以jdbc开始

    子协议：是桥连接的驱动程序或是数据库管理系统名称。

    数据源标识：标记找到数据库来源的地址与连接端口。

    例如：（MySql的连接URL）

    jdbc:mysql:

        //localhost:3306/test?useUnicode=true&characterEncoding=gbk ;

   useUnicode=**true**：表示使用Unicode字符集。如果characterEncoding设置为

   gb2312或GBK，本参数必须设置为**true** 。characterEncoding=gbk：字符编码方式。

3、创建数据库的连接

    •要连接数据库，需要向java.sql.DriverManager请求并获得Connection对象，

     该对象就代表一个数据库的连接。

    •使用DriverManager的getConnectin(String url , String username ,

    String password )方法传入指定的欲连接的数据库的路径、数据库的用户名和

     密码来获得。

     例如：

     //连接MySql数据库，用户名和密码都是root

     String url = "jdbc:mysql://localhost:3306/test" ;

     String username = "root" ;

     String password = "root" ;

**try**{

    Connection con =

             DriverManager.getConnection(url , username , password ) ;

     }**catch**(SQLException se){

    System.out.println("数据库连接失败！");

    se.printStackTrace() ;

     }

 4、创建一个Statement

    •要执行SQL语句，必须获得java.sql.Statement实例，Statement实例分为以下3

     种类型：

      1、执行静态SQL语句。通常通过Statement实例实现。

      2、执行动态SQL语句。通常通过PreparedStatement实例实现。

      3、执行数据库存储过程。通常通过CallableStatement实例实现。

    具体的实现方式：

        Statement stmt = con.createStatement() ;

       PreparedStatement pstmt = con.prepareStatement(sql) ;

       CallableStatement cstmt =

                            con.prepareCall("{CALL demoSp(? , ?)}") ;

5、执行SQL语句

    Statement接口提供了三种执行SQL语句的方法：executeQuery 、executeUpdate

   和execute

    1、ResultSet executeQuery(String sqlString)：执行查询数据库的SQL语句

        ，返回一个结果集（ResultSet）对象。

     2、**int** executeUpdate(String sqlString)：用于执行INSERT、UPDATE或

        DELETE语句以及SQL DDL语句，如：CREATE TABLE和DROP TABLE等

     3、execute(sqlString):用于执行返回多个结果集、多个更新计数或二者组合的

        语句。

   具体实现的代码：

          ResultSet rs = stmt.executeQuery("SELECT \* FROM ...") ;

**int** rows = stmt.executeUpdate("INSERT INTO ...") ;

**boolean** flag = stmt.execute(String sql) ;

6、处理结果

    两种情况：

     1、执行更新返回的是本次操作影响到的记录数。

     2、执行查询返回的结果是一个ResultSet对象。

    • ResultSet包含符合SQL语句中条件的所有行，并且它通过一套get方法提供了对这些

      行中数据的访问。

    • 使用结果集（ResultSet）对象的访问方法获取数据：

**while**(rs.next()){

         String name = rs.getString("name") ;

    String pass = rs.getString(1) ; // 此方法比较高效

     }

    （列是从左到右编号的，并且从列1开始）

7、关闭JDBC对象

     操作完成以后要把所有使用的JDBC对象全都关闭，以释放JDBC资源，关闭顺序和声

     明顺序相反：

     1、关闭记录集

     2、关闭声明

     3、关闭连接对象

**if**(rs != **null**){   // 关闭记录集

**try**{

            rs.close() ;

        }**catch**(SQLException e){

            e.printStackTrace() ;

        }

          }

**if**(stmt != **null**){   // 关闭声明

**try**{

            stmt.close() ;

        }**catch**(SQLException e){

            e.printStackTrace() ;

        }

          }

**if**(conn != **null**){  // 关闭连接对象

**try**{

            conn.close() ;

         }**catch**(SQLException e){

            e.printStackTrace() ;

         }

          }