**3.1、JDBC的简介（理解）**

JDBC是Java数据库连接服务，主要是一个由Java程序操作数据库的执行标准，所以在这个JDBC的开发包java.sql包之中，所有的组成几乎都是接口，那么不同的数据库如果要想使用java开发，就必须实现这些接口的标准。不过JDBC不是技术，而是一种服务：所有的操作完全固定。

在JDBC之中的核心组成部分：DriverManager类、Connection接口、Statement接口、PreparedStatement接口、ResultSet接口，即：五个接口一个类，就构成了整个JDBC的组成。

但是如果要进行JDBC的开发，程序必须首先要连接到数据库，而在Java之中，对于程序连接数据库的操作，有以下的三种方式支持：

· JDBC-ODBC：使用微软的ODBC技术进行数据库的连接；

|- 流程：程序 🡺 JDBC 🡺 ODBC 🡺 数据库，性能很差，数据的实时性特别差，没人用，这个操作由SUN自己提供，所有的JDBC技术都是最新版本；

· JDBC连接：由各个数据库生产商提供数据库驱动程序的实现，用户需要单独配置开发包（\*.jar）；

|- 流程：程序 🡺 JDBC 🡺 数据库，直接操作，性能高；

· JDBC网络连接：由程序根据网络协议连接到指定的数据库上，这个驱动程序依然由数据库生产商提供；

在以后的开发之中主要使用JDBC网络连接。

在Java之中，由于JDBC属于固定的服务，所以其代码的操作流程完全一样：

· **第一步：**加载数据库驱动程序；

· **第二步：**通过DriverManager类根据指定的属性内容连接数据库；

· **第三步：**进行数据表的CRUD操作，执行SQL；

· **第四步：**关闭数据库连接。

**3.2、连接Oracle数据库（重点）**

如果现在要使用Oracle进行连接，那么对于Oracle数据库就必须启动以下两个服务：

· 监听服务：OracleOraDb 10g \_home1TNSListener；

· 数据库实例服务：OracleServiceMLDN。

如果监听服务有问题，那么程序根本就无法连接数据库。当这两个服务已经正常启动之后，下面就需要配置Oracle的驱动程序了。对于Oracle或DB2这样的大型数据库，一般都会在安装目录下存在了数据库的驱动程序开发包，Oracle的驱动程序路径：D:\oracle\product\ 10.1.0 \db\_1\jdbc\lib\classes12.jar。

对于这个jar包的配置需要分两种情况：

**情况一：**如果现在使用命令行方式开发（不用开发工具），则这个jar包要在CLASSPATH属性之中进行配置。

**情况二：**如果现在使用的是Eclipse进行开发，这个时候CLASSPATH是不起作用的，必须在项目属性之中配置指定库文件的开发包。

配置流程：项目点右键 🡺 属性 🡺 Java Build Path 🡺 Add External Jars。

如果要想进行连接，那么首先需要知道如下的几个信息（背）：

· 数据库的驱动程序路径（配置的\*.jar）：oracle.jdbc.driver.OracleDriver；

· 数据库的连接地址：jdbc:oracle:thin:@主机名称:端口号:数据库的实例名称；

|- 要连接本机的mldn数据库：jdbc:oracle:thin:@localhost:1521:mldn；

· 数据库的用户名：scott；

· 数据库的密码：tiger。

不同的数据库以上的信息是不一样的，但是至少背一个，应付面试去。清楚了以上的操作之后，下面来逐步分析如何实现之前所说的JDBC的每一步操作：

**第一步：**加载数据库驱动程序；

数据库驱动程序给出的是“包.类”，只要看见写出了完整的类名称都应该想到反射，所以驱动程序的加载是通过Class类完成的：Class.forName(驱动程序)；

**第二步：**通过DriverManager类根据指定的属性内容连接数据库；

java.sql.DriverManager是一个负责取得数据库连接接口（Connection）对象的操作类，在这个类之中定义了如下一个可以取得连接的方法：public static **Connection** getConnection(String url, String user, String password) throws SQLException；

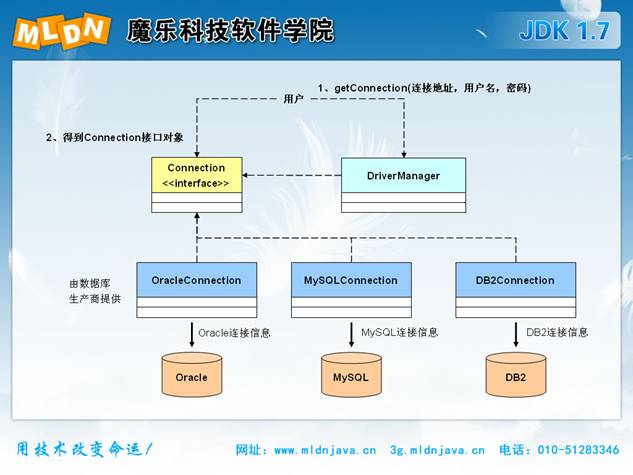
**第四步：**关闭数据库连接。

在Connection接口中定义了close()方法：public void close() throws SQLException。

**范例：**连接mldn数据库

|  |
| --- |
| **package** cn.mldn.demo;  **import** java.sql.Connection;  **import** java.sql.DriverManager;  **public** **class** TestDemo {  **public** **static** **final** String *DBDRIVER* = "oracle.jdbc.driver.OracleDriver";  **public** **static** **final** String *DBURL* = "jdbc:oracle:thin:@localhost:1521:mldn";  **public** **static** **final** String *DBUSER* = "scott";  **public** **static** **final** String *PASSWORD* = "tiger";  **public** **static** **void** main(String[] args) **throws** Exception {  Connection conn = **null**; // 每一个Connection对象表示一个数据库连接  Class.*forName*(*DBDRIVER*); // 加载驱动程序  conn = DriverManager.*getConnection*(*DBURL*, *DBUSER*, *PASSWORD*);  System.*out*.println(conn); // 如果可以输出对象则表示连接成功  conn.close() ; // 关闭连接  }  } |

对于日后不同的数据而言，除了驱动程序、连接地址、用户名和密码不同之外，其余的所有代码都一样。



JDBC的设计就属于一个工厂设计模式，其中的DriverManager类负责取得Connection接口的实例化对象。

在正常情况下以上的程序肯定是可以连接数据库，但是也会有一些个别份子由于自己电脑上经常安装一些不正当的软件会造成连接失败，主要的常见问题有二：

**问题一：**监听问题无法启动

那么这种情况下，如果要想解决此问题，首先要确定一下监听程序的路径是否正确。

现在已经配置的路径：D:\oracle\product\ 10.1.0 \db\_1\BIN\TNSLSNR，如果执行程序的路径正确，那么下面可以使用一个Oracle的工具通过命令行来控制监听启动：lsnrctl，现在就有可能是网络的配置环境出现了问题。

为了解决这个问题，现在进入到网络配置路径：D:\oracle\product\ 10.1.0 \db\_1\NETWORK\ADMIN路径下有两个文件：tnsnames.ora、listener.ora，如果这两个文件的网络配置出错，那么基本上监听服务就Over了。将网络名称修改为主机名称一样的数值。

**问题二：**缺少SID，从而导致无法连接

|  |
| --- |
| Exception in thread "main" java.sql.SQLException: Io 异常: The Network Adapter could not establish the connection |

这个是因为有些电脑安装了Oracle之后，有可能安装过程之中出现了一些问题，导致SID没有配置成功。

这个时候，可以启动Net Manager程序，通过这个程序注册一个与当前数据库实例名称相同的SID，保存后即可连接。

**3.3、使用Statement操作数据库（重点）**

如果现在要进行数据库的操作，那么一定需要数据表，所以首先编写一个如下的数据库创建脚本：

|  |
| --- |
| DROP SEQUENCE myseq ;  DROP TABLE member PURGE ;  CREATE SEQUENCE myseq ;  CREATE TABLE member(  mid NUMBER ,  name VARCHAR2(20) NOT NULL ,  age NUMBER(3) ,  birthday DATE ,  note CLOB ,  CONSTRAINT pk\_mid PRIMARY KEY (mid) ,  CONSTRAINT ck\_age CHECK (age BETWEEN 0 AND 200)  ) ; |

如果要想执行数据表操作在Java之中可以使用Statement接口完成，但是如果要想取得Statement接口对象的话，必须依靠Connection接口完成，在Connection接口中定义了如下的一个操作方法：

· 取得Statement接口对象：public Statement createStatement() throws SQLException；

当取得了Statement接口对象之后就可以进行数据表的CRUD操作了，而操作的方法：

· 执行数据更新操作：public int executeUpdate(String sql) throws SQLException，返回更新的行数；

· 执行数据查询操作：public ResultSet executeQuery(String sql) throws SQLException；

· 关闭操作：public void close() throws SQLException。

**3.3.1、使用Statement接口进行数据表的更新操作**

数据更新就是三种操作语法：增加、修改、删除。

**范例：**执行数据的增加操作

· 增加的SQL语法：INSERT INTO 表名称 (字段,字段,..) VALUES (值,值,…)；

|  |
| --- |
| **package** cn.mldn.demo;  **import** java.sql.Connection;  **import** java.sql.DriverManager;  **import** java.sql.Statement;  **public** **class** TestDemo {  **public** **static** **final** String *DBDRIVER* = "oracle.jdbc.driver.OracleDriver";  **public** **static** **final** String *DBURL* = "jdbc:oracle:thin:@localhost:1521:mldn";  **public** **static** **final** String *DBUSER* = "scott";  **public** **static** **final** String *PASSWORD* = "tiger";  **public** **static** **void** main(String[] args) **throws** Exception {  Connection conn = **null**; // 每一个Connection对象表示一个数据库连接  Statement stmt = **null**; // 数据库操作对象  Class.*forName*(*DBDRIVER*); // 加载驱动程序  conn = DriverManager.*getConnection*(*DBURL*, *DBUSER*, *PASSWORD*);  stmt = conn.createStatement(); // 找到Statement接口对象  String sql = "INSERT INTO member (mid,name,age,birthday,note) VALUES (myseq.nextval,'张三',20,SYSDATE,'是世界末日的一个人')";  **int** len = stmt.executeUpdate(sql);  System.*out*.println("更新行数：" + len);  stmt.close(); // 关闭操作，如果不关闭操作，关闭连接也是一样的  conn.close(); // 关闭连接  }  } |

**范例：**执行修改操作

· 修改的SQL语法：UPDATE 表名称 SET 字段=值,字段=值,.. [WHERE 更新条件(s)]；

|  |
| --- |
| **package** cn.mldn.demo;  **import** java.sql.Connection;  **import** java.sql.DriverManager;  **import** java.sql.Statement;  **public** **class** TestDemo {  **public** **static** **final** String *DBDRIVER* = "oracle.jdbc.driver.OracleDriver";  **public** **static** **final** String *DBURL* = "jdbc:oracle:thin:@localhost:1521:mldn";  **public** **static** **final** String *DBUSER* = "scott";  **public** **static** **final** String *PASSWORD* = "tiger";  **public** **static** **void** main(String[] args) **throws** Exception {  Connection conn = **null**; // 每一个Connection对象表示一个数据库连接  Statement stmt = **null**; // 数据库操作对象  Class.*forName*(*DBDRIVER*); // 加载驱动程序  conn = DriverManager.*getConnection*(*DBURL*, *DBUSER*, *PASSWORD*);  stmt = conn.createStatement(); // 找到Statement接口对象  String sql = "UPDATE member SET name='李四',age=19,note='扫除达人' WHERE mid BETWEEN 10 AND 20";  **int** len = stmt.executeUpdate(sql);  System.*out*.println("更新行数：" + len);  stmt.close(); // 关闭操作，如果不关闭操作，关闭连接也是一样的  conn.close(); // 关闭连接  }  } |

**范例：**执行删除操作

· 删除的SQL语法：DELETE FROM 表名称 WHERE 删除条件 ;

|  |
| --- |
| **package** cn.mldn.demo;  **import** java.sql.Connection;  **import** java.sql.DriverManager;  **import** java.sql.Statement;  **public** **class** TestDemo {  **public** **static** **final** String *DBDRIVER* = "oracle.jdbc.driver.OracleDriver";  **public** **static** **final** String *DBURL* = "jdbc:oracle:thin:@localhost:1521:mldn";  **public** **static** **final** String *DBUSER* = "scott";  **public** **static** **final** String *PASSWORD* = "tiger";  **public** **static** **void** main(String[] args) **throws** Exception {  Connection conn = **null**; // 每一个Connection对象表示一个数据库连接  Statement stmt = **null**; // 数据库操作对象  Class.*forName*(*DBDRIVER*); // 加载驱动程序  conn = DriverManager.*getConnection*(*DBURL*, *DBUSER*, *PASSWORD*);  stmt = conn.createStatement(); // 找到Statement接口对象  String sql = "DELETE FROM member WHERE mid IN (1,11,21)";  **int** len = stmt.executeUpdate(sql);  System.*out*.println("更新行数：" + len);  stmt.close(); // 关闭操作，如果不关闭操作，关闭连接也是一样的  conn.close(); // 关闭连接  }  } |

在数据的更新操作之中，如果已经成功的执行了数据的更新，一定会返回更新的行数的。

**3.3.2、使用Statement接口进行查询**

更新操作会返回给用户更新的行数，那么查询更应该返回数据给用户浏览，所以在整个进行查询的操作之中，就必须有一种结构，可以装下整个的查询结果，而这个结构就使用ResultSet表示。

当所有的记录返回到ResultSet的时候，所有的内容都是按照数据类型存放的，所以用户只需要按照数据类型一行行的取出数据即可，在ResultSet接口之中定义了这样的几个方法：

· 移动指针并判断是否有数据：public boolean next() throws SQLException；

· 取得指定类型的数据：public 数据 getXxx(列的标记) throws SQLException

· 关闭：public void close() throws SQLException。

**范例：**使用ResultSet取出数据

|  |
| --- |
| **package** cn.mldn.demo;  **import** java.sql.Connection;  **import** java.sql.DriverManager;  **import** java.sql.ResultSet;  **import** java.sql.Statement;  **import** java.util.Date;  **public** **class** TestDemo {  **public** **static** **final** String *DBDRIVER* = "oracle.jdbc.driver.OracleDriver";  **public** **static** **final** String *DBURL* = "jdbc:oracle:thin:@localhost:1521:mldn";  **public** **static** **final** String *DBUSER* = "scott";  **public** **static** **final** String *PASSWORD* = "tiger";  **public** **static** **void** main(String[] args) **throws** Exception {  Connection conn = **null**; // 每一个Connection对象表示一个数据库连接  Statement stmt = **null**; // 数据库操作对象  Class.*forName*(*DBDRIVER*); // 加载驱动程序  conn = DriverManager.*getConnection*(*DBURL*, *DBUSER*, *PASSWORD*);  stmt = conn.createStatement(); // 找到Statement接口对象  String sql = "SELECT mid,name,age,birthday,note FROM member";  ResultSet rs = stmt.executeQuery(sql) ; // 查询  **while** (rs.next()) {  **int** mid = rs.getInt("mid") ;  String name = rs.getString("name") ;  **int** age = rs.getInt("age") ;  Date birthday = rs.getDate("birthday") ;  String note = rs.getString("note") ;  System.*out*.println(mid + "，" + name + "，" + age + "，" + birthday  + "，" + note);  }  rs.close() ;  stmt.close(); // 关闭操作，如果不关闭操作，关闭连接也是一样的  conn.close(); // 关闭连接  }  } |

从今天开始，**所有的SQL语句的查询之中不允许再出现“SELECT \*…”，必须明确的写出要显示字段名称**。但是，在编写SQL语句的时候，既然已经明确的知道了要显示列的顺序，所以也可以采用序号的形式在ResultSet中使用。

|  |
| --- |
| **while** (rs.next()) {  **int** mid = rs.getInt(1) ;  String name = rs.getString(2) ;  **int** age = rs.getInt(3) ;  Date birthday = rs.getDate(4) ;  String note = rs.getString(5) ;  System.*out*.println(mid + "，" + name + "，" + age + "，" + birthday  + "，" + note);  } |

这种方式更加的简单，所以以后都采用此方式完成。

**3.4、预处理：PreparedStatement（核心）**

之前所讲解的Statement基本上不用，即：可以不会，下面首先来解释一下之所以不去使用Statement操作的原因，假设：现在要增加的数据为用户自己输入的。

|  |
| --- |
| **package** cn.mldn.demo;  **import** java.sql.Connection;  **import** java.sql.DriverManager;  **import** java.sql.Statement;  **public** **class** TestDemo {  **public** **static** **final** String *DBDRIVER* = "oracle.jdbc.driver.OracleDriver";  **public** **static** **final** String *DBURL* = "jdbc:oracle:thin:@localhost:1521:mldn";  **public** **static** **final** String *DBUSER* = "scott";  **public** **static** **final** String *PASSWORD* = "tiger";  **public** **static** **void** main(String[] args) **throws** Exception {  String name = "Mr'Smith" ;  String birthday = " 1989-01-02 " ;  **int** age = 30 ;  String note = "终于来了个人" ;  Connection conn = **null**; // 每一个Connection对象表示一个数据库连接  Statement stmt = **null**; // 数据库操作对象  Class.*forName*(*DBDRIVER*); // 加载驱动程序  conn = DriverManager.*getConnection*(*DBURL*, *DBUSER*, *PASSWORD*);  stmt = conn.createStatement(); // 找到Statement接口对象  String sql = "INSERT INTO member (mid,name,age,birthday,note) VALUES (myseq.nextval,'" + name + "'," + age + ",TO\_DATE('" + birthday  + "','yyyy-mm-dd'),'" + note + "')";  System.out.println(sql);  **int** len = stmt.executeUpdate(sql);  System.*out*.println("更新行数：" + len);  stmt.close(); // 关闭操作，如果不关闭操作，关闭连接也是一样的  conn.close(); // 关闭连接  }  } |

本程序一执行之后，会出现如下的错误提示信息：

|  |
| --- |
| INSERT INTO member (mid,name,age,birthday,note) VALUES (myseq.nextval,'**Mr'Smith**',30,TO\_DATE(' 1989-01-02 ','yyyy-mm-dd'),'终于来了个人')  Exception in thread "main" java.sql.**SQLException**: ORA-00917: missing comma |

在日后的所有开发之中，如果出现了SQLException不用去检查代码，就检查SQL语句是不是有问题，在SQL语句之中所有的字符串要求使用“'”定义，但是现在名字上有了“'”，那么自然SQL语句就出现了错误，所以导致程序无法进行数据的保存。所以通过这一代码就可以发现，在开发之中，这种拼凑SQL语句的形式根本就不可能被使用，即：Statement不应该被使用，所有的开发都会使用Statement的子接口：PreparedStatement完成。

PreparedStatement采用的是一种预处理的方式来进行程序编写的，所谓的预处理指的是先在数据库之中执行好要操作的SQL语句，但是其对应的内容暂时不插入，通过程序再依次设置。

如果要想取得PreparedStatement接口的对象依然需要通过Connection接口完成，在Connection接口定义的方法：

· 取得PreparedStatement接口对象：public PreparedStatement prepareStatement(String **sql**) throws SQLException

当取得了PreparedStatement接口对象之后，就可以通过PreparedStatement定义的方法进行数据库操作：

· **更新操作：**public int executeUpdate() throws SQLException；

· **查询操作：**public ResultSet executeQuery() throws SQLException。

如果现在要为操作的SQL设置内容，使用的方法：

· 根据索引设置各种类型：public void setBigDecimal(int parameterIndex, 数据类型 x) throws SQLException

在使用PreparedStatement接口中的setDate()方法的时候，其对应的类型是java.sql.Date，而不是平常使用的java.util.Date，那么就必须执行转换，在java.sql.Date类中有一个构造：public Date(long date)。

**范例：**通过PreparedStatement执行增加操作

|  |
| --- |
| **package** cn.mldn.demo;  **import** java.sql.Connection;  **import** java.sql.DriverManager;  **import** java.sql.PreparedStatement;  **import** java.util.Date;  **public** **class** TestDemo {  **public** **static** **final** String *DBDRIVER* = "oracle.jdbc.driver.OracleDriver";  **public** **static** **final** String *DBURL* = "jdbc:oracle:thin:@localhost:1521:mldn";  **public** **static** **final** String *DBUSER* = "scott";  **public** **static** **final** String *PASSWORD* = "tiger";  **public** **static** **void** main(String[] args) **throws** Exception {  String name = "Mr'Smith" ;  Date birthday = **new** Date() ; // 表示的是日期  **int** age = 30 ;  String note = "终于来了个人" ;  String sql = "INSERT INTO member (mid,name,age,birthday,note) VALUES (myseq.nextval,?,?,?,?)";  Connection conn = **null**; // 每一个Connection对象表示一个数据库连接  PreparedStatement pstmt = **null**; // 数据库操作对象  Class.*forName*(*DBDRIVER*); // 加载驱动程序  conn = DriverManager.*getConnection*(*DBURL*, *DBUSER*, *PASSWORD*);  pstmt = conn.prepareStatement(sql) ; // 执行了SQL  pstmt.setString(1, name) ;  pstmt.setInt(2,age) ;  pstmt.setDate(3,**new** java.sql.Date(birthday.getTime())) ;  pstmt.setString(4, note) ;  **int** len = pstmt.executeUpdate() ; // 执行更新操作  System.*out*.println("更新行数：" + len);  pstmt.close(); // 关闭操作，如果不关闭操作，关闭连接也是一样的  conn.close(); // 关闭连接  }  } |

正因为PreparedStatement使用起来要比Statement方便，所以在日后的开发之中，全部都使用PreparedStatement，而不要使用Statement接口操作。

更新的操作都和增加操作类似，而最麻烦的就在于数据的查询操作，下面编写几个程序，这几个程序是必须会的操作，直接和后续的代码开发有联系。

**范例：**查询全部操作

|  |
| --- |
| **package** cn.mldn.demo;  **import** java.sql.Connection;  **import** java.sql.DriverManager;  **import** java.sql.PreparedStatement;  **import** java.sql.ResultSet;  **import** java.util.Date;  **public** **class** TestDemo {  **public** **static** **final** String *DBDRIVER* = "oracle.jdbc.driver.OracleDriver";  **public** **static** **final** String *DBURL* = "jdbc:oracle:thin:@localhost:1521:mldn";  **public** **static** **final** String *DBUSER* = "scott";  **public** **static** **final** String *PASSWORD* = "tiger";  **public** **static** **void** main(String[] args) **throws** Exception {  String sql = "SELECT mid,name,age,birthday,note FROM member";  Connection conn = **null**; // 每一个Connection对象表示一个数据库连接  PreparedStatement pstmt = **null**; // 数据库操作对象  Class.*forName*(*DBDRIVER*); // 加载驱动程序  conn = DriverManager.*getConnection*(*DBURL*, *DBUSER*, *PASSWORD*);  pstmt = conn.prepareStatement(sql) ; // 执行了SQL  ResultSet rs = pstmt.executeQuery() ; // 查询操作  **while** (rs.next()) {  **int** mid = rs.getInt(1) ;  String name = rs.getString(2) ;  **int** age = rs.getInt(3) ;  Date birthday = rs.getDate(4) ;  String note = rs.getString(5) ;  System.*out*.println(mid + "，" + name + "，" + age + "，" + birthday  + "，" + note);  }  pstmt.close(); // 关闭操作，如果不关闭操作，关闭连接也是一样的  conn.close(); // 关闭连接  }  } |

**范例：**设置按id查询，如果按照id查询只会返回一条数据，要么就不返回。

|  |
| --- |
| **package** cn.mldn.demo;  **import** java.sql.Connection;  **import** java.sql.DriverManager;  **import** java.sql.PreparedStatement;  **import** java.sql.ResultSet;  **import** java.util.Date;  **public** **class** TestDemo {  **public** **static** **final** String *DBDRIVER* = "oracle.jdbc.driver.OracleDriver";  **public** **static** **final** String *DBURL* = "jdbc:oracle:thin:@localhost:1521:mldn";  **public** **static** **final** String *DBUSER* = "scott";  **public** **static** **final** String *PASSWORD* = "tiger";  **public** **static** **void** main(String[] args) **throws** Exception {  String sql = "SELECT mid,name,age,birthday,note FROM member WHERE mid=?";  Connection conn = **null**; // 每一个Connection对象表示一个数据库连接  PreparedStatement pstmt = **null**; // 数据库操作对象  Class.*forName*(*DBDRIVER*); // 加载驱动程序  conn = DriverManager.*getConnection*(*DBURL*, *DBUSER*, *PASSWORD*);  pstmt = conn.prepareStatement(sql) ; // 执行了SQL  pstmt.setInt(1, 300) ; // 在执行之前设置内容  ResultSet rs = pstmt.executeQuery() ; // 查询操作  **if** (rs.next()) { // 查询出来就显示  **int** mid = rs.getInt(1) ;  String name = rs.getString(2) ;  **int** age = rs.getInt(3) ;  Date birthday = rs.getDate(4) ;  String note = rs.getString(5) ;  System.*out*.println(mid + "，" + name + "，" + age + "，" + birthday  + "，" + note);  } **else** {  System.*out*.println("没有查询结果！");  }  pstmt.close(); // 关闭操作，如果不关闭操作，关闭连接也是一样的  conn.close(); // 关闭连接  }  } |

在使用ResultSet接口操作的时候，如果有内容则next()才可以执行，如果没有查询结果，next()直接返回false。

**范例：**设置模糊查询

|  |
| --- |
| **package** cn.mldn.demo;  **import** java.sql.Connection;  **import** java.sql.DriverManager;  **import** java.sql.PreparedStatement;  **import** java.sql.ResultSet;  **import** java.util.Date;  **public** **class** TestDemo {  **public** **static** **final** String *DBDRIVER* = "oracle.jdbc.driver.OracleDriver";  **public** **static** **final** String *DBURL* = "jdbc:oracle:thin:@localhost:1521:mldn";  **public** **static** **final** String *DBUSER* = "scott";  **public** **static** **final** String *PASSWORD* = "tiger";  **public** **static** **void** main(String[] args) **throws** Exception {  String column = "name" ;  String keyWord = "三" ;  String sql = "SELECT mid,name,age,birthday,note FROM member WHERE " + column + " LIKE ?";  Connection conn = **null**; // 每一个Connection对象表示一个数据库连接  PreparedStatement pstmt = **null**; // 数据库操作对象  Class.*forName*(*DBDRIVER*); // 加载驱动程序  conn = DriverManager.*getConnection*(*DBURL*, *DBUSER*, *PASSWORD*);  pstmt = conn.prepareStatement(sql) ; // 执行了SQL  **pstmt.setString(1, "%" + keyWord + "%");** // 在执行之前设置内容  ResultSet rs = pstmt.executeQuery() ; // 查询操作  **while** (rs.next()) { // 查询出来就显示  **int** mid = rs.getInt(1) ;  String name = rs.getString(2) ;  **int** age = rs.getInt(3) ;  Date birthday = rs.getDate(4) ;  String note = rs.getString(5) ;  System.*out*.println(mid + "，" + name + "，" + age + "，" + birthday  + "，" + note);  }  pstmt.close(); // 关闭操作，如果不关闭操作，关闭连接也是一样的  conn.close(); // 关闭连接  }  } |

**范例：**分页显示，Oracle使用ROWNUM完成

|  |
| --- |
| **package** cn.mldn.demo;  **import** java.sql.Connection;  **import** java.sql.DriverManager;  **import** java.sql.PreparedStatement;  **import** java.sql.ResultSet;  **import** java.util.Date;  **public** **class** TestDemo {  **public** **static** **final** String *DBDRIVER* = "oracle.jdbc.driver.OracleDriver";  **public** **static** **final** String *DBURL* = "jdbc:oracle:thin:@localhost:1521:mldn";  **public** **static** **final** String *DBUSER* = "scott";  **public** **static** **final** String *PASSWORD* = "tiger";  **public** **static** **void** main(String[] args) **throws** Exception {  String column = "name" ;  String keyWord = "三" ;  **int** currentPage = 2 ; // 当前所在页  **int** lineSize = 5 ; // 每页显示的长度  String sql = " SELECT \* FROM ( "  + " SELECT mid,name,age,birthday,note,ROWNUM rn FROM member WHERE "  + column + " LIKE ? AND ROWNUM<=?) temp " + " WHERE temp.rn>? ";  Connection conn = **null**; // 每一个Connection对象表示一个数据库连接  PreparedStatement pstmt = **null**; // 数据库操作对象  Class.*forName*(*DBDRIVER*); // 加载驱动程序  conn = DriverManager.*getConnection*(*DBURL*, *DBUSER*, *PASSWORD*);  pstmt = conn.prepareStatement(sql) ; // 执行了SQL  pstmt.setString(1, "%" + keyWord + "%"); // 在执行之前设置内容  pstmt.setInt(2, currentPage \* lineSize) ;  pstmt.setInt(3, (currentPage - 1) \* lineSize);  ResultSet rs = pstmt.executeQuery() ; // 查询操作  **while** (rs.next()) { // 查询出来就显示  **int** mid = rs.getInt(1) ;  String name = rs.getString(2) ;  **int** age = rs.getInt(3) ;  Date birthday = rs.getDate(4) ;  String note = rs.getString(5) ;  System.*out*.println(mid + "，" + name + "，" + age + "，" + birthday  + "，" + note);  }  pstmt.close(); // 关闭操作，如果不关闭操作，关闭连接也是一样的  conn.close(); // 关闭连接  }  } |

如果这个时候设置的关键字为空字符串（String keyWord = "" ;）的话呢，则表示查询全部。

**范例：**统计查询，统计数据量，统计函数使用COUNT()完成。

|  |
| --- |
| **package** cn.mldn.demo;  **import** java.sql.Connection;  **import** java.sql.DriverManager;  **import** java.sql.PreparedStatement;  **import** java.sql.ResultSet;  **public** **class** TestDemo {  **public** **static** **final** String *DBDRIVER* = "oracle.jdbc.driver.OracleDriver";  **public** **static** **final** String *DBURL* = "jdbc:oracle:thin:@localhost:1521:mldn";  **public** **static** **final** String *DBUSER* = "scott";  **public** **static** **final** String *PASSWORD* = "tiger";  **public** **static** **void** main(String[] args) **throws** Exception {  String column = "name" ;  String keyWord = "" ;  String sql = " SELECT COUNT(mid) FROM member WHERE " + column  + " LIKE ? ";  Connection conn = **null**; // 每一个Connection对象表示一个数据库连接  PreparedStatement pstmt = **null**; // 数据库操作对象  Class.*forName*(*DBDRIVER*); // 加载驱动程序  conn = DriverManager.*getConnection*(*DBURL*, *DBUSER*, *PASSWORD*);  pstmt = conn.prepareStatement(sql) ; // 执行了SQL  pstmt.setString(1, "%" + keyWord + "%"); // 在执行之前设置内容  ResultSet rs = pstmt.executeQuery() ; // 查询操作  **if** (rs.next()) { // 一定会有内容返回  **int** count = rs.getInt(1) ;  System.*out*.println(count);  }  pstmt.close(); // 关闭操作，如果不关闭操作，关闭连接也是一样的  conn.close(); // 关闭连接  }  } |

以上的几种就是JDBC在实际开发之中的基本操作，也是必须掌握的操作。

**3.5、批处理及事务处理（理解）**

以上的一些代码是属于JDBC 1.0的时候就推出的功能，但是这些功能被新版本在一些bug进行了纠正。而最新的JDBC版本为4.0（没人使），因为从JDBC 2.0之后增加了一些新功能：可滚动结果集、使用ResultSet更新数据、批处理，这之中批处理操作是唯一有点用的技术：所谓的批处理指的是数据库可以一次性的执行多条SQL语句。所以在JDBC 2.0之后，对于Statement和PreparedStatement接口都有了一些新的方法：

· Statement接口增加了两个方法：

|- 增加一个执行的SQL：public void addBatch(String sql) throws SQLException，没有执行；

|- 一次性执行多条SQL：public int[] executeBatch() throws SQLException；

· PreparedStatement接口增加了一个方法：

|- 增加执行的SQL：public void addBatch() throws SQLException。

**范例：**为了方便起见，使用Statement执行一次批处理

|  |
| --- |
| **package** cn.mldn.demo;  **import** java.sql.Connection;  **import** java.sql.DriverManager;  **import** java.sql.Statement;  **import** java.util.Arrays;  **public** **class** TestDemo {  **public** **static** **final** String *DBDRIVER* = "oracle.jdbc.driver.OracleDriver";  **public** **static** **final** String *DBURL* = "jdbc:oracle:thin:@localhost:1521:mldn";  **public** **static** **final** String *DBUSER* = "scott";  **public** **static** **final** String *PASSWORD* = "tiger";  **public** **static** **void** main(String[] args) **throws** Exception {  Connection conn = **null**; // 每一个Connection对象表示一个数据库连接  Statement stmt = **null** ;  Class.*forName*(*DBDRIVER*); // 加载驱动程序  conn = DriverManager.*getConnection*(*DBURL*, *DBUSER*, *PASSWORD*);  stmt = conn.createStatement() ;  stmt.addBatch("INSERT INTO member(mid,name) VALUES (myseq.nextval,'张三A')") ;  stmt.addBatch("INSERT INTO member(mid,name) VALUES (myseq.nextval,'张三B')") ;  stmt.addBatch("INSERT INTO member(mid,name) VALUES (myseq.nextval,'张三C')") ;  stmt.addBatch("INSERT INTO member(mid,name) VALUES (myseq.nextval,'张三D')") ;  stmt.addBatch("INSERT INTO member(mid,name) VALUES (myseq.nextval,'张三E')") ;  **int** data [] = stmt.executeBatch() ; // 执行批处理  System.*out*.println(Arrays.*toString*(data));  conn.close(); // 关闭连接  }  } |

但是在这里面有一个小问题了，如果假设以上的五条更新SQL都属于同一个业务：

· A、从罗先生帐户上转出5000W；

· B、在我的帐户上增加5000W； 🡺 本步出错了

· C、交付转帐手续费50；

· D、交付个人所得税：10。

在JDBC下，所有的事务处理都是自动提交的，这就意味着只要执行了SQL语句，那么都会自动的提交数据库的事务，而如果要想解决之前的问题，就必须手工处理事务，所有的事务处理操作命令都在Connection接口之中定义了，有如下的三个支持事务操作的方法：

· 设置事务是否自动提交：public void setAutoCommit(boolean autoCommit) throws SQLException；

· 提交事务：public void commit() throws SQLException；

· 回滚事务：public void rollback() throws SQLException。

**范例：**手工控制事务

|  |
| --- |
| **package** cn.mldn.demo;  **import** java.sql.Connection;  **import** java.sql.DriverManager;  **import** java.sql.Statement;  **import** java.util.Arrays;  **public** **class** TestDemo {  **public** **static** **final** String *DBDRIVER* = "oracle.jdbc.driver.OracleDriver";  **public** **static** **final** String *DBURL* = "jdbc:oracle:thin:@localhost:1521:mldn";  **public** **static** **final** String *DBUSER* = "scott";  **public** **static** **final** String *PASSWORD* = "tiger";  **public** **static** **void** main(String[] args) **throws** Exception {  Connection conn = **null**; // 每一个Connection对象表示一个数据库连接  Statement stmt = **null** ;  Class.*forName*(*DBDRIVER*); // 加载驱动程序  conn = DriverManager.*getConnection*(*DBURL*, *DBUSER*, *PASSWORD*);  conn.setAutoCommit(**false**) ; // 取消自动提交  stmt = conn.createStatement() ;  **try** {  stmt.addBatch("INSERT INTO member(mid,name) VALUES (myseq.nextval,'张三A')") ;  stmt.addBatch("INSERT INTO member(mid,name) VALUES (myseq.nextval,'张三B')") ;  stmt.addBatch("INSERT INTO member(mid,name) VALUES (myseq.nextval,'张三'C')") ;  stmt.addBatch("INSERT INTO member(mid,name) VALUES (myseq.nextval,'张三D')") ;  stmt.addBatch("INSERT INTO member(mid,name) VALUES (myseq.nextval,'张三E')") ;  **int** data [] = stmt.executeBatch() ; // 执行批处理  System.*out*.println(Arrays.*toString*(data));  conn.commit() ; // 不出错提交  } **catch** (Exception e) {  e.printStackTrace() ;  conn.rollback() ;// 回滚事务  }  conn.close(); // 关闭连接  }  } |

但是这些手工的事务处理，日后实际上也不需要用户自己完成，因为有容器帮助。

**4、总结**

1、 JDBC连接Oracle数据库；

2、 PreparedStatement接口实现数据的更新和查询操作。

**5、作业**

1、 巩固一下Java的基本概念，尤其是面向对象；

2、 代码：

· 简单Java类和数据表映射；

· 动物园程序，换成List接口；

· 各个设计模式的组成：单例、工厂、代理；

· 异常处理的标准格式；

· 日期、基本类型、String的互相转换；

· 正则表达式；

· 文件拷贝、打印流、扫描流；

· JDBC的PreparedStatement操作。