

知识点列表

| 编号 | 名称 | 描述 | 级别 |
|----|-----------------|------------------------------|----|
| 1 | JDBC 原理 | 了解 JDBC API 的概念和原理 | * |
| 2 | JDBC API 的使用(1) | 开发 java 程序，通过 JDBC 遍历数据表 | ** |
| 3 | JDBC API 的使用(2) | 开发 java 程序，通过 JDBC 实现登录功能 | ** |
| 4 | JDBC API 的使用(3) | 开发 java 程序，通过 JDBC 实现 DML 操作 | ** |

注： "*"理解级别 "**"掌握级别 "***"应用级别

目录

| | |
|---|----|
| 1. JDBC 原理* | 4 |
| 2. JDBC 使用 | 4 |
| 2.1. 连接 Oracle 数据库必要数据 | 4 |
| 2.2. Oracle 连接字符串 | 5 |
| 3. 【案例 1】Eclipse 工具开发第一个 JDBC 程序 : MyJDBCDemo ** | 5 |
| 3.1. 步骤 1 | 5 |
| 3.2. 步骤 2 | 5 |
| 3.3. 步骤 3 | 6 |
| 3.4. 步骤 4 | 9 |
| 3.5. 小结 : JDBC 使用过程 | 12 |
| 3.6. 常见错误 | 13 |
| 4. 【案例 2】利用 JDBC 访问数据库 , 实现登录功能 ** | 19 |
| 步骤 1 : 准备数据 , 建表 users_xxx(或者使用原有的 users 表) | 19 |
| 步骤 2 : 在 MyJDBCDemo 中定义并实现方法 login(int id , String pwd) | 20 |
| 步骤 3 : 在 MyJDBCDemo 的 main 方法中调用 login 方法 , 得到登录成功或失败的结果。 | 21 |
| 5. 【案例 3】DML 操作_insert ** | 21 |
| 5.1. 插入数据_Statement(较麻烦) | 21 |
| 5.2. 插入数据_PreparedStatement(简单 , 多次执行时效率高) | 23 |
| 6. 【案例 4】DML 操作_update ** | 25 |
| 6.1. 伪代码 | 25 |
| 6.2. 代码 | 26 |

| | |
|---|----|
| 7. 【案例 5】DML 操作_delete ** | 27 |
| 7.1. 伪代码 | 27 |
| 7.2. 代码 | 28 |
| 8. 总结 | 29 |
| 8.1. Select 与 DML(insert/update/delete)的区别 | 29 |
| 8.2. Select 与 DML(insert/update/delete)的相同点 | 30 |

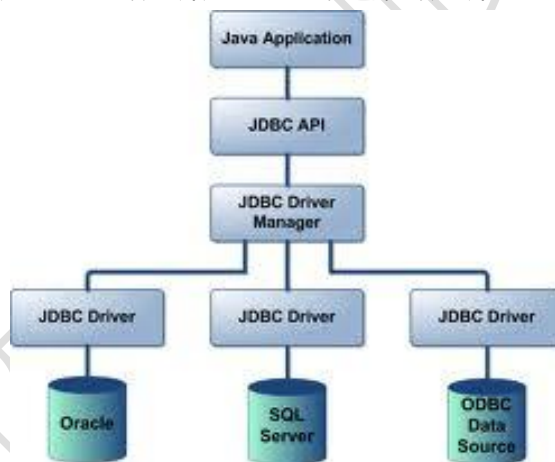
1. JDBC 原理*

JDBC(Java DataBase Connectivity,java 数据库连接)是一种用于执行 SQL 语句的 Java API , 可以为多种关系数据库提供统一访问 , 它由一组用 Java 语言编写的类和接口组成。

JDBC API 是 SUN 公司提出的访问数据库的接口标准。有了 JDBC API , 就不必为访问不同的数据库编写不同的程序 , 程序员可以使用相同的一套 API 访问不同的数据库 , 同时 , 将 Java 语言和 JDBC 结合起来使程序员不必为不同的平台编写不同的应用程序 , 只须写一遍程序就可以让它在任何平台上运行 , 这也是 Java 语言 “编写一次 , 处处运行” 的优势。

JDBC 对 Java 程序员而言是 API , 对数据库厂商而言是接口模型。作为 API , JDBC 为程序开发提供标准的接口 , 并为数据库厂商及第三方中间件厂商实现与数据库的连接提供了标准方法。

简单地说 , JDBC 可做三件事 : 与数据库建立连接、发送操作数据库的语句、返回处理结果。



2. JDBC 使用

2.1. 连接 Oracle 数据库必要数据

- 1) **Ip**(Oracle 数据库所在服务器的 ip 地址) 192.168.0.26(中关村校区)
- 2) **sid**(Oracle 数据库的唯一标识号) tarena
- 3) **port**(Oracle 数据库的默认端口号) 1521
- 4) **dbUser/dbPassword**(数据库的访问帐号) openlab/open123

2.2. Oracle 连接字符串

jdbc : oracle : thin : @192.168.0.26 : 1521 : tarena(中关村校区)

3. 【案例 1】Eclipse 工具开发第一个 JDBC 程序 : MyJDBCDemo **

3.1. 步骤 1

在 Oracle 数据库中新建表 emp_xxx

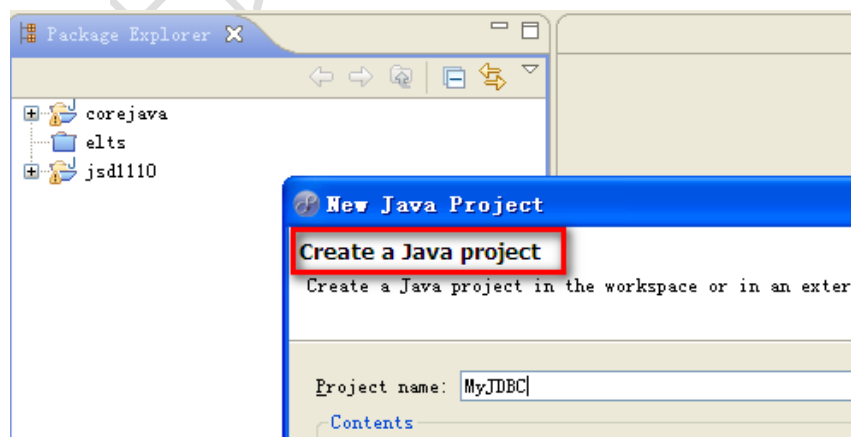
```
SQL> create table emp_xxx(
      id number primary key,
      name varchar2(20),
      age varchar2(20)
    );

SQL> insert into emp_xxx values(1001 , 'sala' , 18);
SQL> insert into emp_xxx values(1002 , 'liucs' , 28);
SQL> insert into emp_xxx values(1003 , 'lala' , 38);

SQL> commit; --注意一定要提交!
```

3.2. 步骤 2

新建 java project



3.3. 步骤 3

把 jar 包(驱动)导入到项目中

3.3.1. 查看 JavaDoc 中 java.sql.*

- 常用接口

| | |
|----------------------------|--------|
| java.sql.Connection | 连接 |
| java.sql.Statement | 语句 |
| java.sql.PreparedStatement | 预编译的语句 |
| java.sql.ResultSet | 结果集 |

- 常用类

| | |
|---------------|-------|
| DriverManager | 驱动管理器 |
|---------------|-------|

The screenshot shows the 'JDK API 1.6.0 中文版' window. On the left, a tree view lists various Java packages, with 'java.sql' selected. Below the tree, a list of interfaces is shown: 'java.sql.Connection', 'java.sql.Statement', 'java.sql.PreparedStatement', 'java.sql.ResultSet', 'java.sql.Driver', 'java.sql.DatabaseMetaData', 'java.sql.CallableStatement', 'java.sql.Array', 'java.sql.Blob', 'java.sql.Clob', and 'java.sql.NClob'. On the right, a table titled '接口摘要' (Interface Summary) provides a brief description for each interface.

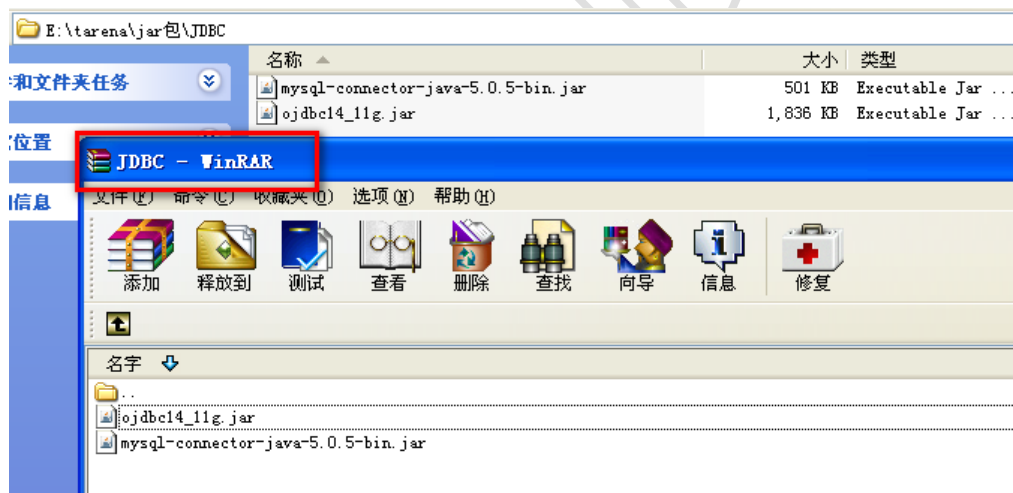
| 接口摘要 | |
|-----------------------------------|------------|
| Array | SQL 类型 |
| Blob | SQL BLOB |
| CallableStatement | 用于执行 |
| Clob | SQL 类型 |
| Connection | 与特定数据库建立连接 |
| DatabaseMetaData | 关于数据库元数据 |
| Driver | 每个驱动程序实现 |
| NClob | SQL NCLOB |

[Array](#)
[Blob](#)
[CallableStatement](#)
[Clob](#)
[Connection](#)
[DatabaseMetaData](#)
[Driver](#)
[NClob](#)
[ParameterMetaData](#)
[PreparedStatement](#)
[Ref](#)
[ResultSet](#)
[ResultSetMetaData](#)
[RowId](#)
[Savepoint](#)

| | |
|-----------------------------------|----------------|
| ParameterMetaData | 可用于获取属性信息 |
| PreparedStatement | 表示预编译 |
| Ref | Java 编程 SQL 结构 |
| ResultSet | 表示数据结果集。 |
| ResultSetMetaData | 可用于获取 |
| RowId | SQL ROWID |
| Savepoint | 保存点的当前 |

3.3.2. 接口的实现类 ojdbc14_11g.jar

ojdbc14_11g.jar 是 Oracle 公司为 Oracle 数据库提供的接口实现类，相应的还有 MySQL、SqlServer 等数据库的接口实现类。

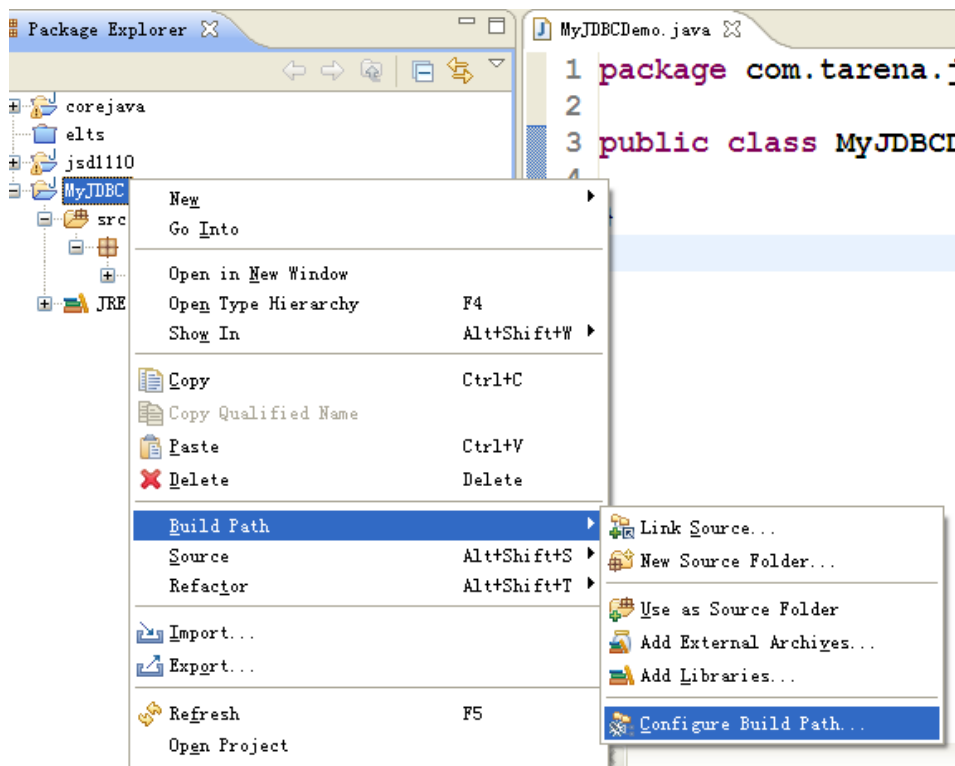


- ✓ Jar 包可以用解压缩软件打开。
- ✓ 导入到 MyEclipse 时，不需要解压，直接把 jar 包导入。
- ✓ 连接不同的数据库，需要下载不同的包，这些包也被称作驱动包。

3.3.3. 把 jar 包(驱动)导入到项目中

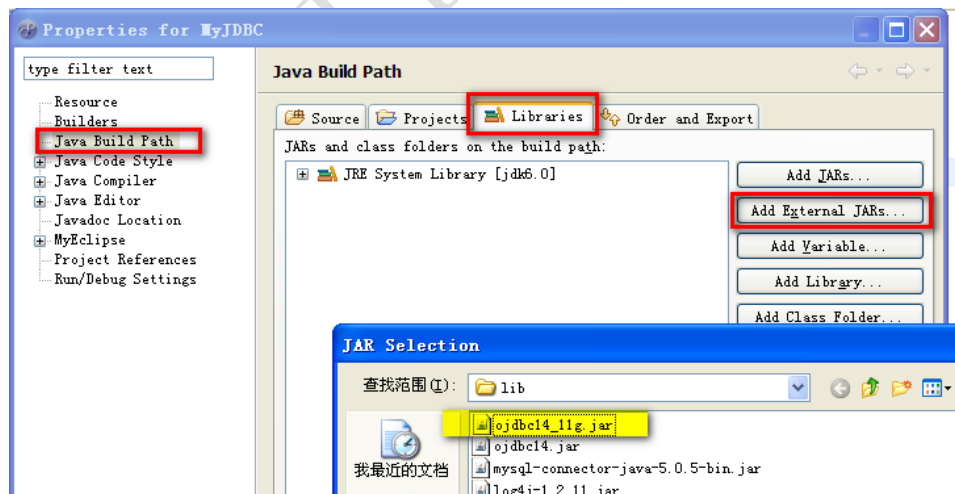
● 步骤 1

项目右键->build path(构建路径)->configure build path(配置构建路径)

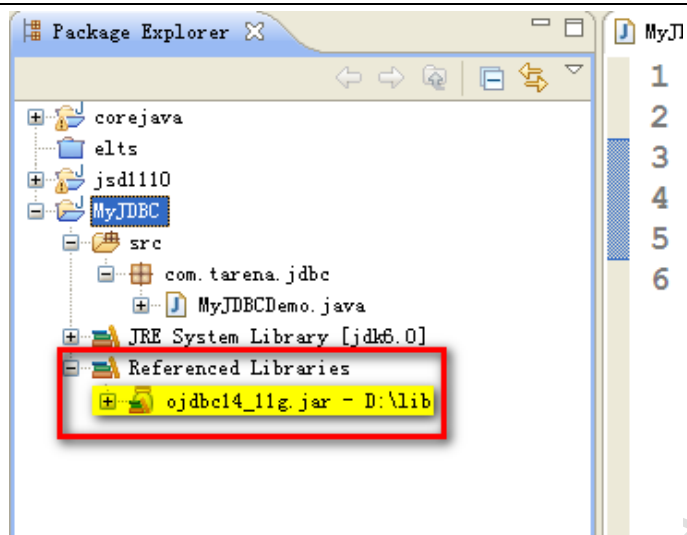


- 步骤 2

Java Build Path --> libraries(库) --> Add External JARs --> 选择 ojdbc14_11g.jar



- 在项目中出现 Referenced Libraries，则完成



3.4. 步骤 4

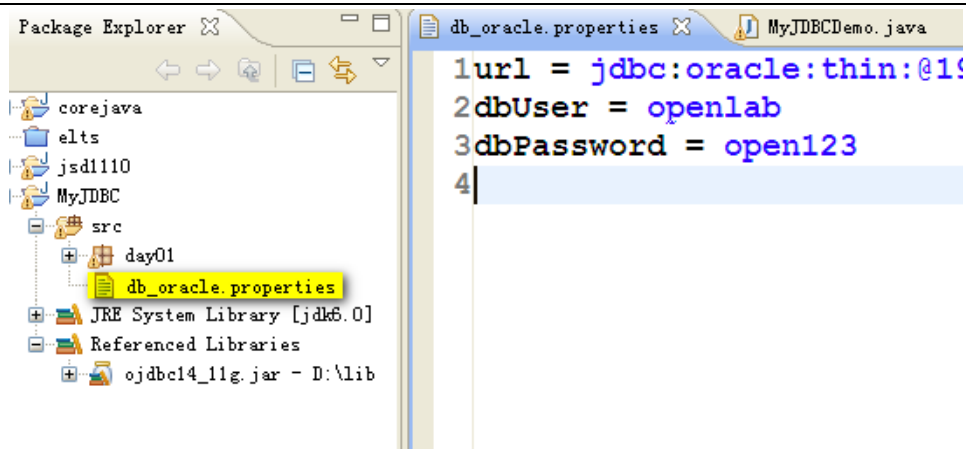
开发访问数据库的程序

测试 Java 程序利用 JDBC API 访问 Oracle 数据库

3.4.1. 新建 db_oracle.properties，存放项目必须使用到的参数

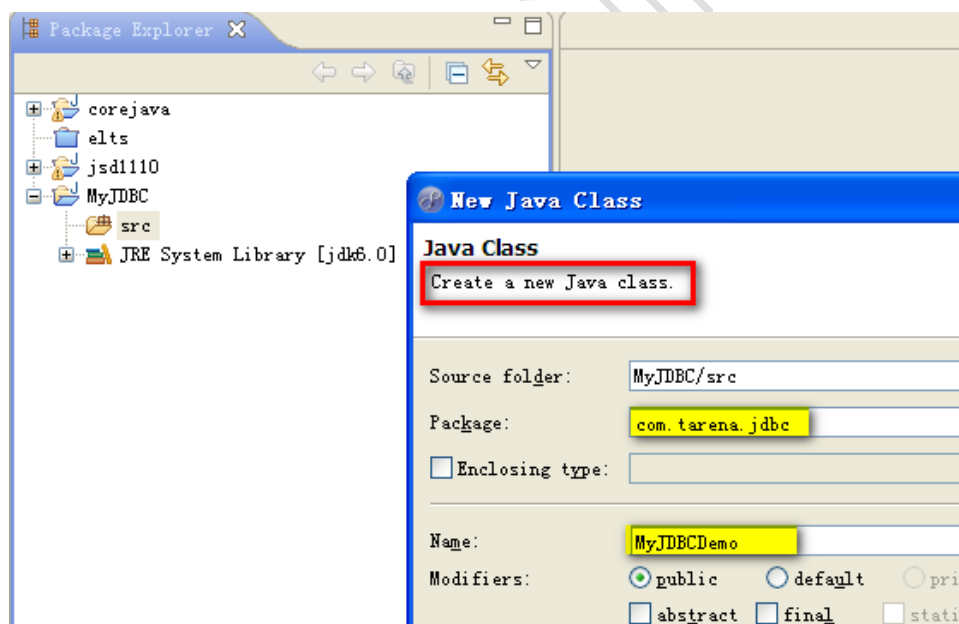
- **db_oracle.properties**

```
url = jdbc:oracle:thin:@192.168.0.26:1521:tarena
dbUser = openlab
dbPassword = open123
```



注意：.properties 为扩展名的文本文件被称作属性文件，内容是键值对的格式，在项目中经常用作 参数配置文件

3.4.2. 开发 java 程序：MyJDBCDemo.java



- **此程序的功能：**

- 1) 读取 db_oracle.properties 中的属性，获取属性文件中的参数(这一步和 JDBC 没有关系)
- 2) 利用第一步获取的参数连接数据库
- 3) 传递语句 sql = "select * from emp"进入数据库，并且返回结果
- 4) 关闭资源

- **参考代码**

```

1 package day01;
2 import java.io.*;
3 import java.sql.*;
4 import java.util.*;
5 /**
6  * 测试Java程序利用JDBC API访问oracle数据库
7  * @author teacher
8  *
9  */
10 public class MyJDBCDemo {
11     private static String url ;
12     private static String dbUser ;
13     private static String dbPassword ;
14
15     /**
16      * 读入filename指定的文件并解析，取出键值对中的数据
17      * 给全局变量url, dbUser, dbPassword赋值
18      *
19      * @param filename : 文件名
20      */
21     public static void getParam(String filename){
22         Properties propes = new Properties();
23
24         File file = new File(filename);
25         try {
26             FileInputStream fis = new FileInputStream(file);
27             //加载输入流指定的文件
28             propes.load(fis);
29             //获取文件中url(key)对应的value，并赋值给全局变量
30             url = propes.getProperty("url");
31             dbUser = propes.getProperty("dbUser");
32             dbPassword = propes.getProperty("dbPassword");
33
34         } catch (FileNotFoundException e) {
35             e.printStackTrace();
36         } catch (IOException e) {
37             e.printStackTrace();
38         }
39     }
40
41     /**
42      * 连接数据库，获取emp表的数据并输出到控制台
43      */
44     public static void getEmpData(){
45         Connection conn = null;//定义连接对象
46         Statement stmt = null;
47         ResultSet rs = null;
48         //sql语句，注意没有分号
49         String sql = "select * from emp";

```

```

50         try {
51             //利用DriverManager给连接对象赋值,
52             //需要三个参数: 连接字符串、数据库的用户名、密码
53             conn = DriverManager
54                 .getConnection(url,dbUser,dbPassword);
55
56             stmt = conn.createStatement();
57
58             //传递sql语句到数据库中,并返回结果集
59             rs = stmt.executeQuery(sql);
60
61
62             //next()方法的功能:
63             //     指针下移一行,并返回boolean值
64             //     指针的初始位置是第一行数据之前(BeforeFirst)
65             //下移一行后,如果指针指向的这行有数据,返回true,
66             //     否则返回false
67             while(rs.next()){//每个while循环体,指针指向一行
68
69                 //id、name、age是列名
70                 //数据库表中number类型的数据用getInt()取值
71                 // varchar2/char类型的数据用getString()取值
72                 int id      = rs.getInt("id");
73                 String name = rs.getString("name");
74                 int age     = rs.getInt("age");
75                 System.out.println(id+","+name+","+age);
76             }
77
78             //关闭资源
79             rs.close();
80             stmt.close();
81             conn.close();
82
83         } catch (SQLException e) {
84             e.printStackTrace();
85         }
86     }
87
88     /**
89     * @param args
90     */
91     public static void main(String[] args) {
92         getParam("src/db_oracle.properties");
93         getEmpData();
94     }
95 }

```

3.5. 小结：JDBC 使用过程

- 1) 在项目加入数据库的驱动(jar 包)

连接不同的数据库，需要加入不同的数据库的 jar 包

2) 必须有要连接的数据库的参数

不同的数据库，连接字符串不一样。

3) 连接字符串

- ✓ **oracle** 的连接字符串 : jdbc: oracle: thin: @ip: port: sid
- ✓ **mysql** 的连接字符串(假设 mysql 安装在本地主机，其中有个 database 叫 test，账户和密码分别是 openlab/open123) :
jdbc: mysql: //localhost: 3306/openlab?user=openlab&password=open123

4) 构造对数据库的连接并操作

- 构造 Connection(会话,连接)对象
DriverManager.getConnection(url , dbUser , dbPassword) ;
- 构造 Statement(语句)对象: 传递 sql 语句的载体
stmt = conn.createStatement() ;
- 如果执行的是查询语句，使用 executeQuery 方法，返回结果集，标识查询结果：
ResultSet rs = stmt.executeQuery("select * from emp"); //执行查询
While(rs.next()){
 int id = rs.getInt("id") ;
}
■ 如果执行的是 DML 操作(insert/update/delete)，执行 executeUpdate 方法，返回的是数字，代表影响的记录数
int n = stmt.executeUpdate("update emp set salary = salary + 1000");

5) 关闭资源

```
if (rs != null) rs.close();
if (stmt != null) stmt.close();
if (conn != null) stmt.close();
```

3.6. 常见错误

注意：一般的 SQLException 都是因为操作数据库时出错了，要么 Sql 语句写错了，要么数据库中的表或数据出错了。

3.6.1. 文件找不到

● 错误演示

```

91     public static void main(String[] args) {
92         getParam("src/db_oracle.properties");
93         getEmpData();
94     }

```

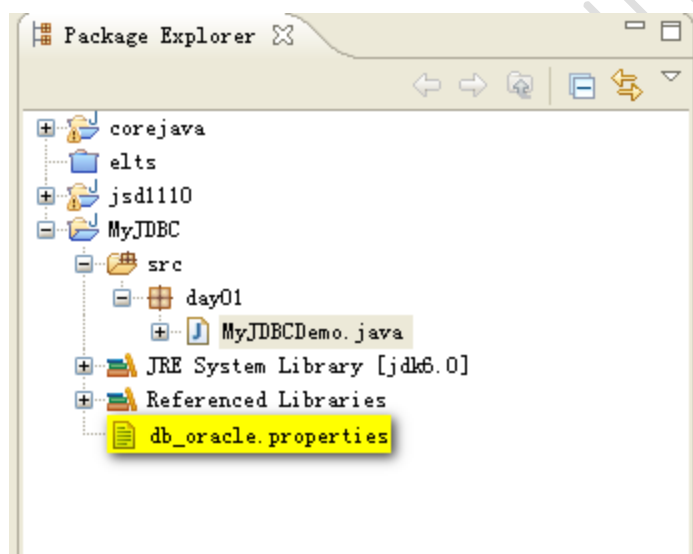
Console

```

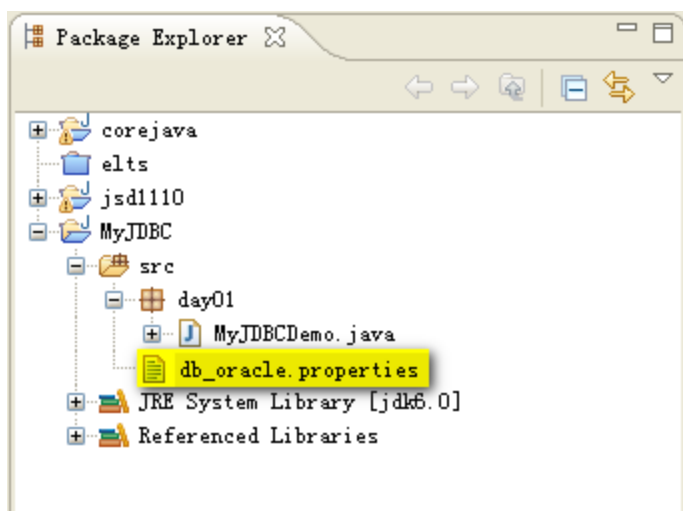
<terminated> MyJDBCDemo [Java Application] C:\Program Files\Java\jdk1.6.0_06\bin\javaw.exe (Dec 13, 2011 10:59:28 AM)
java.io.FileNotFoundException: src\db_oracle.properties (系统找不到指定的文件)
    at java.io.FileInputStream.open(Native Method)
    at java.io.FileInputStream.<init>(FileInputStream.java:106)
    at day01.MyJDBCDemo.getParam(MyJDBCDemo.java:26)
    at day01.MyJDBCDemo.main(MyJDBCDemo.java:92)
java.sql.SQLException: The url cannot be null
    at java.sql.DriverManager.getConnection(DriverManager.java:554)
    at java.sql.DriverManager.getConnection(DriverManager.java:185)
    at day01.MyJDBCDemo.getEmpData(MyJDBCDemo.java:54)
    at day01.MyJDBCDemo.main(MyJDBCDemo.java:93)

```

- 错误原因：db_oracle.properties 不在 src 目录下，可以在 db_oracle.properties 上右键查看文件的 Properties，检查路径是否正确。

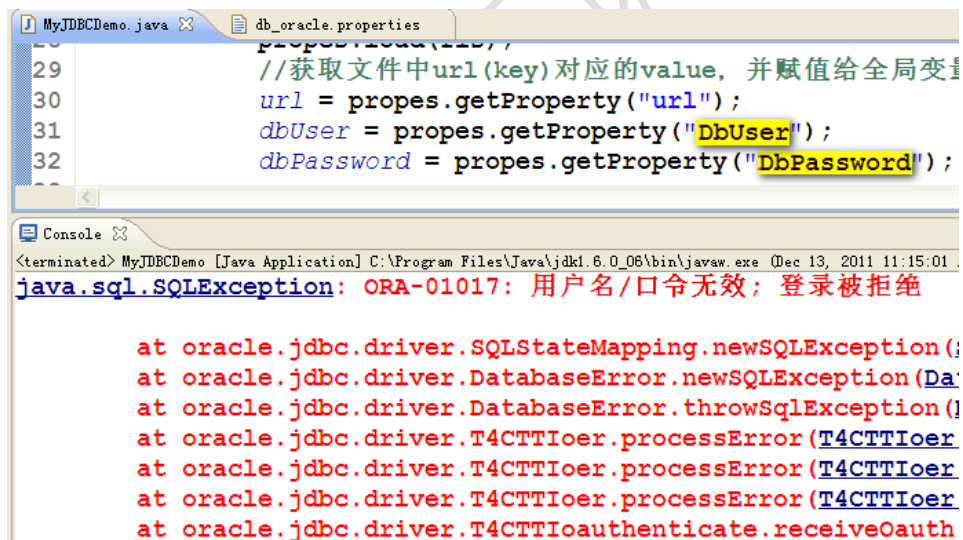


- 正确结果

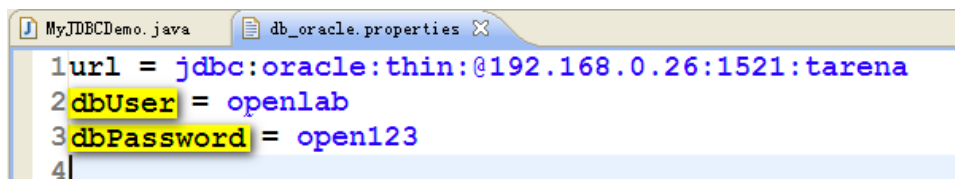


3.6.2. 登录被拒绝

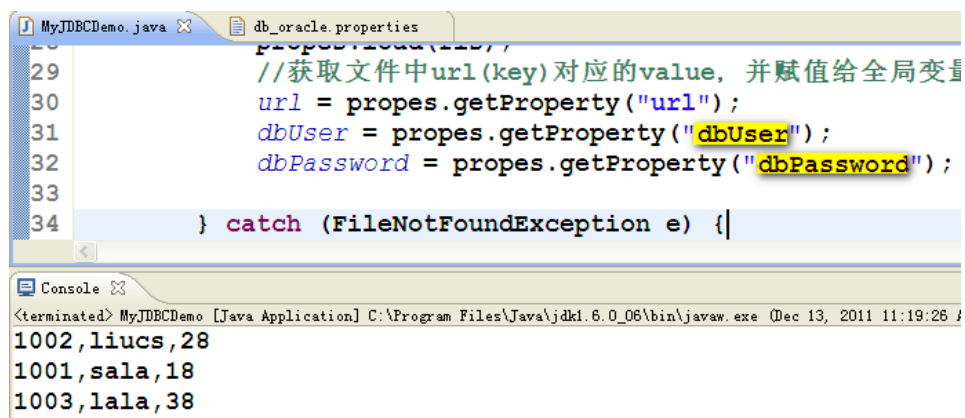
- 错误演示



- 错误原因：程序里取键值对信息时的大小写和属性文件中不匹配



- 正确结果



The screenshot shows an IDE with two tabs: 'MyJDBCDemo.java' and 'db_oracle.properties'. The Java code in 'MyJDBCDemo.java' is as follows:

```

29 //获取文件中url(key)对应的value, 并赋值给全局变量
30 url = propes.getProperty("url");
31 dbUser = propes.getProperty("dbUser");
32 dbPassword = propes.getProperty("dbPassword");
33
34 } catch (FileNotFoundException e) {

```

The console output shows the following data:

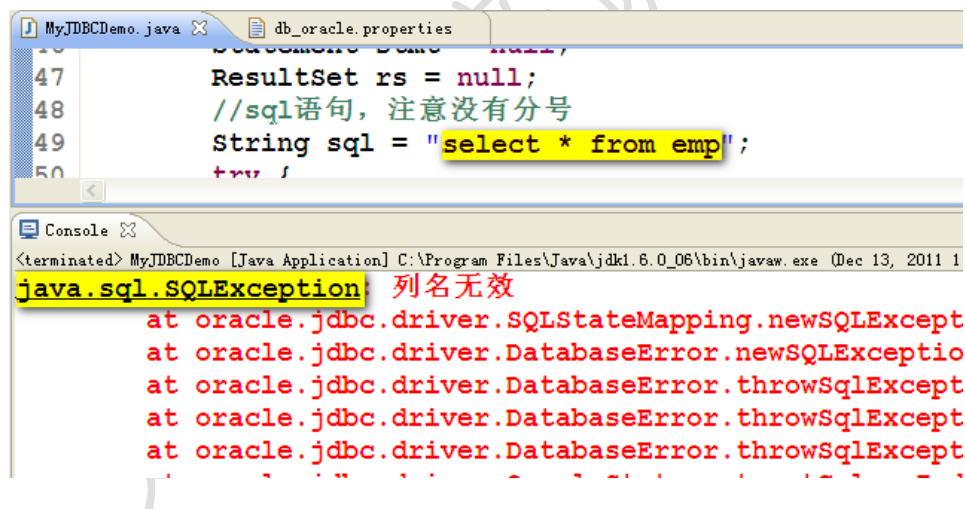
```

1002,liucs,28
1001,sala,18
1003,lala,38

```

3.6.3. 列名无效

- 错误演示



The screenshot shows an IDE with two tabs: 'MyJDBCDemo.java' and 'db_oracle.properties'. The Java code in 'MyJDBCDemo.java' is as follows:

```

47 ResultSet rs = null;
48 //sql语句, 注意没有分号
49 String sql = "select * from emp";
50

```

The console output shows the following error message:

```

java.sql.SQLException: 列名无效
    at oracle.jdbc.driver.SQLStateMapping.newSQLException
    at oracle.jdbc.driver.DatabaseError.newSQLException
    at oracle.jdbc.driver.DatabaseError.throwSQLException
    at oracle.jdbc.driver.DatabaseError.throwSQLException
    at oracle.jdbc.driver.DatabaseError.throwSQLException

```

- 错误原因：查找的表和查找的列不匹配


```

Telnet 192.168.0.26
SQL> desc emp;
Name                                     Null?      Type
-----
EMPNO                                     NUMBER(6)
ENAME                                     VARCHAR2
JOB                                       VARCHAR2
MGR                                       NUMBER(6)
HIREDATE                                  DATE
COMM                                      NUMBER(7)
DEPTNO                                   NUMBER(6)
SAL                                       NUMBER(7)

SQL> desc emp_xxx;
Name                                     Null?      Type
-----
ID                                       NOT NULL   NUMBER
NAME                                     VARCHAR2
AGE                                     VARCHAR2

SQL>

```

● 正确结果

```

MyJDBCDemo.java  db_oracle.properties
47      ResultSet rs = null;
48      //sql语句，注意没有分号
49      String sql = "select * from emp_xxx";
50      try {
51          //利用...
52          //连接服务器时...

Console
<terminated> MyJDBCDemo [Java Application] C:\Program Files\Java\jdk1.6.0_06\bin\javaw.exe (Dec 13, 2011 1
1002,liucs,28
1001,sala,18
1003,lala,38

```

3.6.4. 无效字符

● 错误演示

```

MyJDBCdemo.java db_oracle.properties
47      ResultSet rs = null;
48      //sql语句, 注意没有分号
49      String sql = "select * from emp_xxx;";
50

Console
<terminated> MyJDBCdemo [Java Application] C:\Program Files\Java\jdk1.6.0_06\bin\javaw.exe (Dec 13, 2011 1
java.sql.SQLException: ORA-00911: 无效字符

    at oracle.jdbc.driver.SQLStateMapping.newSQLException(
    at oracle.jdbc.driver.DatabaseError.newSQLException(
    at oracle.jdbc.driver.DatabaseError.throwSQLException(
    at oracle.jdbc.driver.T4CTTIoer.processError(T4CTTI
    at oracle.jdbc.driver.T4CTTIoer.processError(T4CTTI
    at oracle.jdbc.driver.T4COCall.receive(T4COCall.c
  
```

- 错误原因：sql 语句语法有错，语句结尾不能有分号

- 正确结果

```

MyJDBCdemo.java db_oracle.properties
47      ResultSet rs = null;
48      //sql语句, 注意没有分号
49      String sql = "select * from emp_xxx";
50      try {
51          //利用DriverManager给连接对象赋值,
52          //需要三个参数：连接字符串、数据库的用户名、
53          conn = DriverManager

Console
<terminated> MyJDBCdemo [Java Application] C:\Program Files\Java\jdk1.6.0_06\bin\javaw.exe (Dec 13, 2011 1
1002,liucs,28
1001,sala,18
1003,lala,38
  
```

3.6.5. 提示：无法转换为内部表示

```

40         while(rs.next()){
41             System.out.println(rs.getInt("empno") + ", "
42                                 //当出现" 无法转换为内部表示 "异常时
43                                 //检查rs获取数据的方式, 数据类型
44                                 + rs.getInt("ename") + ", "
45                                 + rs.getDouble("salary"));
46         }
47     }catch(Exception e){

```

Console SQL Results

<terminated> SelectPageDemo [Java Application] C:\Program Files\Java\jdk1.6.0_06\bin\javaw.exe (Dec 19, 2011 4:10:26)

java.sql.SQLException: 无法转换为内部表示

at oracle.jdbc.driver.SQLStateMapping.newSQLException(S
at oracle.jdbc.driver.DatabaseError.newSQLException(Dat
at oracle.jdbc.driver.DatabaseError.throwSQLException(D

- 错误原因：结果集取数据时注意数据类型。

3.6.6. 新增数据后务必要 commit, 否则查不到数据。

4. 【案例 2】利用 JDBC 访问数据库，实现登录功能 **

步骤 1：准备数据，建表 users_xxx(或者使用原有的 users 表)

```

SQL> create table users_ning(
    id number(4) primary key ,
    password varchar2(4) ,
    name varchar2(10) ,
    phone varchar2(20) ,
    email varchar2(30)
);
SQL> insert into users_ning values( 1001,'1234','liucs','13500000000',
    'liucs@tarena.com.cn') ;
SQL> commit ;
SQL> select name from users_ning
    where id = 1001 and password = '1234';

```

-- 注意一定要提交

```
SQL> select name from users_ning
      2 where id = 1001 and password = '1234';
```

```
NAME
-----
liucs
```

步骤 2：在 MyJDBCDemo 中定义并实现方法 login(int id, String pwd)

伪代码如下所示：

```
public boolean login(int id, String pwd){
    //填充方法，实现判断 id 和 pwd 是否是合法用户
    //如果是则返回 true，不是返回 false
}
```

```
209= /**
210  * 判断是否有userId/userPwd对应的考生
211  * @param userId 考生id
212  * @param userPwd 考生密码
213  * @return 有,返回true, 没有,返回false
214  */
215= public static boolean login(int userId, String userPwd) {
216     boolean flag = false;
217
218     try {
219         // 1. 获得连接,需要三个参数:url/dbUser/dbPassword
220         Connection conn
221             = DriverManager.getConnection(url, dbUser, dbPassword);
222         // 2. 发送查询语句到数据库中,并接收结果
223         String sql = "select name from users_ning"
224             + " where id = " + userId
225             + " and password = '" + userPwd + "'";
226         System.out.println(sql);
227         Statement stmt = conn.createStatement();
228         ResultSet rs = stmt.executeQuery(sql); // 执行sql语句
229         if (rs.next()) {
230             flag = true;
231             System.out.println("结果集中有数据!");
232             System.out.println(rs.getString("name")); // 从结果集中取数据.
233         }
234     }
```

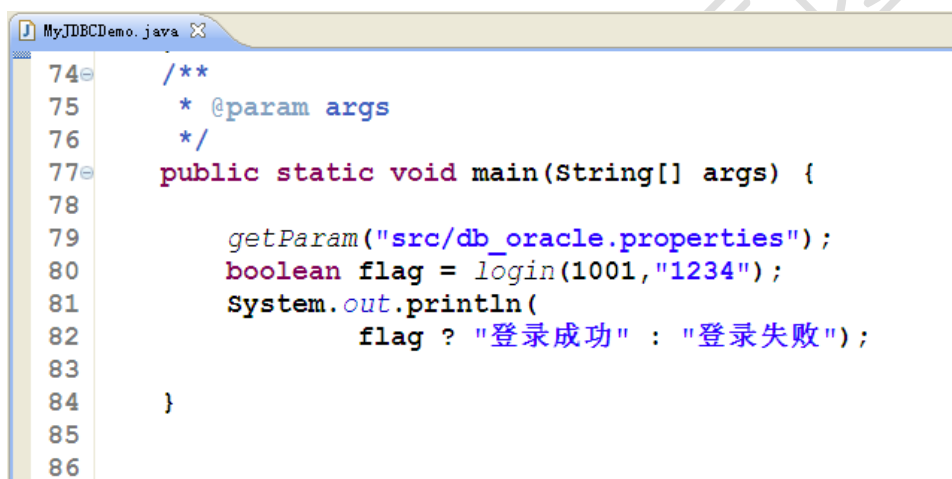
```

234      // 3.释放资源
235      rs.close();
236      stmt.close();
237      conn.close();
238  } catch (SQLException e) {
239      e.printStackTrace();
240  }
241
242      return flag;
243  }
244

```

✓ **注意**代码 225 行，单引号"'" 是 SQL 语句的一部分，不要丢掉了

步骤 3：在 MyJDBCdemo 的 main 方法中调用 login 方法，得到登录成功或失败的结果。



```

74  /**
75   * @param args
76   */
77  public static void main(String[] args) {
78
79      getParam("src/db_oracle.properties");
80      boolean flag = login(1001,"1234");
81      System.out.println(
82          flag ? "登录成功" : "登录失败");
83
84  }
85
86

```

5. 【案例 3】DML 操作_insert **

5.1. 插入数据_Statement(较麻烦)

步骤 1：准备数据，向 users_ning 表插入数据

SQL> desc users_ning

| Name | Null? | Type |
|----------|----------|--------------|
| ID | NOT NULL | NUMBER(4) |
| PASSWORD | | VARCHAR2(4) |
| NAME | | VARCHAR2(10) |
| PHONE | | VARCHAR2(20) |
| EMAIL | | VARCHAR2(30) |

步骤 2：实现方法 insertUser(int id, String pwd, String name, String phone, String email)

```

389= public static boolean addUser(int id, String name,
390     String password, String phone, String email) {
391     boolean flag = false;
392     Connection conn = null;
393     Statement stmt = null;
394     try {
395         conn = DriverManager
396             .getConnection(url, dbUser, dbPassword);
397         stmt = conn.createStatement();
398
399         String sql = "insert into users_ning" +
400             "(id, name, password, phone, email) " +
401             " values(" + id + ", '" + name + "', '" +
402             password + "', '" + phone + "', '" + email + "')";
403
404         int n = stmt.executeUpdate(sql);
405         if (n == 1)
406             flag = true;
407     } catch (Exception e) {
408         e.printStackTrace();
409     } finally {
410         try {
411             if (stmt != null)
412                 stmt.close();
413         } catch (SQLException e) {
414             e.printStackTrace();
415         }
416         try {
417             if (conn != null)
418                 conn.close();
419         } catch (SQLException e) {
420             e.printStackTrace();
421         }
422     }

```

✓ 这种方式非常容易出错，比如，email 数据前后没有加单引号。

```
399     String sql = "insert into users_ning" +
400               "(id, name, password, phone, email) " +
401               " values (" + id + ", '" + name + "', '" +
402               password + "', '" + phone + "', " + email + ")";
```

- ✓ 在 insert 语句中，非数字要用单引号引起来。否则数据会被当作列，报列在此处不允许的错误

5.2. 插入数据_PreparedStatement(简单，多次执行时效率高)

```
158  /**
159   * 增加考生记录，SQL语句:
160   * insert into users_ning(id, name, password, phone, email)
161   * values(1010,'rose','abcd','5678','rose@tarena')
162   *
163   * @param id 考生id
164   * @param name 考生姓名
165   * @param password 考生密码
166   * @param phone 考生电话
167   * @param email 考生email
168   * @return 增加成功: true, 增加失败: false.
169   */
170  public static boolean add(int id, String name,
171    String password, String phone, String email) {
172    boolean flag = false;
173    Connection conn = null;
174    PreparedStatement stmt = null;
175    try {
176      conn = DriverManager
177        .getConnection(url, dbUser, dbPassword);
178
179      String sql = "insert into users_ning" +
180                "(id, name, password, phone, email) " +
181                " values(?,?,?,?)" ;// 占位符
182
183      stmt = conn.prepareStatement(sql);
```

```

184
185         // 给占位符赋值
186         stmt.setInt(1, id);
187         stmt.setString(2, name);
188         stmt.setString(3, password);
189         stmt.setString(4, phone);
190         stmt.setString(5, email);
191
192         // 执行.
193         int n = stmt.executeUpdate();
194         if (n == 1)
195             flag = true;
196
197     } catch (Exception e) {
198         e.printStackTrace();
199     } finally {
200         try {
201             if (stmt != null)
202                 stmt.close();
203         } catch (SQLException e) {
204             e.printStackTrace();
205         }
206         try {
207             if (conn != null)
208                 conn.close();
209         } catch (SQLException e) {
210             e.printStackTrace();
211         }
212     }
213
214     return flag;
215 }

```

● 在 main 方法中调用：

```

43 public static void main(String[] args) {
44     getParam("src/db_oracle.properties");
45
46     System.out.println(
47         add(1011, "rose", "abcd", "5678", "rose@tarena")
48         ?
49         "增加成功"
50         :
51         "增加失败"
52     );
53
54 }

```


- 使用 PreparedStatement 时可能出现的错误：索引中丢失 in 或 out 参数：2

```

34     ResultSet rs = null;
35     try{
36         stmt = conn.prepareStatement(sql);
37         stmt.setInt(1, begin);
38         //当提示" 索引中丢失 IN 或 OUT 参数:: 2 " 异常时
39         //检查是否所有的问号都关联了数据
40         //stmt.setInt(2, end);
41         rs = stmt.executeQuery();
42         while(rs.next()){
43             System.out.println(rs.getInt("empno") + ", "
44                               + rs.getString("ename") + ", "
45                               + rs.getDouble("salary"));
46         }
47     }catch(Exception e){
48         e.printStackTrace();
49     }

```

Console X SQL Results

<terminated> SelectPageDemo [Java Application] C:\Program Files\Java\jdk1.6.0_06\bin\javaw.exe (Dec 19, 2011 4:03:18)

java.sql.SQLException: 索引中丢失 IN 或 OUT 参数:: 2

at oracle.jdbc.driver.SQLStateMapping.newSQLException(SQ
at oracle.jdbc.driver.DatabaseError.newSQLException(Data
at oracle.jdbc.driver.DatabaseError.throwSQLException(Da
at oracle.jdbc.driver.DatabaseError.throwSQLException(Da

- Statement 和 PreparedStatement

| | |
|----------------------------|---|
| java.sql.Statement | 用于执行静态 SQL 语句并返回其生成结果的对象。 |
| java.sql.PreparedStatement | 继承 Statement 接口，表示预编译的 SQL 语句的对象，SQL 语句被预编译并且存储在 PreparedStatement 对象中。然后可以使用此对象高效地多次执行该语句。 |

6. 【案例 4】DML 操作_update **

6.1. 伪代码

```

/*修改考生密码
update user_xxx set passwd = ? where id = ?*/
public static boolean changePassword (int id, String newPassword){
}

```

6.2. 代码

步骤 1：准备数据，user_xxx，确保其中有 1001 号考生的数据

```
SQL> select name from users_ning
2  where id = 1001 and password = '1234';

NAME
-----
liucs
```

步骤 2：实现方法 changePassword(int id, String password)

```
40=  /**
41     * @param args
42     */
43=  public static void main(String[] args) {
44      getParam("src/db_oracle.properties");
45      System.out.println(
46          changePassword(1011, "abcd")
47          ?
48          "修改成功"
49          :
50          "修改失败"
51          );
52=
109= /**
110     * 修改考生密码
111     *
112     * @param id 要修改密码的考生id
113     * @param newPassword 新密码
114     * @return 修改成功: true, 修改失败: false
115     */
```

```

116 public static boolean changePassword
117     (int id, String newPassword) {
118     String sql = "update users_ning " +
119         "set passwd = ? " + " where id = ?";
120     boolean flag = false;
121
122     Connection conn = null;
123     PreparedStatement stmt = null;
124
125     try {
126         conn = DriverManager
127             .getConnection(url, dbUser, dbPassword);
128         stmt = conn.prepareStatement(sql);
129         stmt.setString(1, newPassword);
130         stmt.setInt(2, id);
131
132         int n = stmt.executeUpdate();
133         // id是主键
134         if (n == 1) {
135             flag = true;
136         }
137     } catch (Exception e) {
138         e.printStackTrace();
139     } finally {
140         if (stmt != null)
141             try {
142                 stmt.close();
143             } catch (SQLException e) {
144                 e.printStackTrace();
145             }
146         if (conn != null)
147             try {
148                 conn.close();
149             } catch (SQLException e) {
150                 e.printStackTrace();
151             }
152     }
153 }

```

7. 【案例 5】DML 操作_delete **

7.1. 伪代码

```

/*删除指定 id 的用户*/
public static boolean deleteUser(int id){

```

7.2.代码

```

64  /**
65   * 删除考生
66   *
67   * @param id 要删除的考生信息
68   * @return 删除成功: true; 失败: false
69   */
70  private static boolean deleteUser(int id) {
71      String sql = "delete users_ning where id = ?";
72      boolean flag = false;
73
74      Connection conn = null;
75      PreparedStatement stmt = null;
76
77      try {
78          conn = DriverManager
79              .getConnection(url, dbUser, dbPassword);
80          stmt = conn.prepareStatement(sql);
81          stmt.setInt(1, id);
82
83          int n = stmt.executeUpdate();
84          // 假设id是主键
85          if (n == 1) {
86              flag = true;
87          }
88

```

```

89     } catch (Exception e) {
90         e.printStackTrace();
91     } finally {
92         if (stmt != null)
93             try {
94                 stmt.close();
95             } catch (SQLException e) {
96                 e.printStackTrace();
97             }
98         if (conn != null)
99             try {
100                 conn.close();
101             } catch (SQLException e) {
102                 e.printStackTrace();
103             }
104     }
105
106     return flag;
107 }

```

- **Main 方法中调用：**

```

43 public static void main(String[] args) {
44     getParam("src/db_oracle.properties");
45     System.out.println(
46         deleteUser(1011)
47         ?
48         "删除成功"
49         :
50         "删除失败"
51     );

```

8. 总结

8.1. Select 与 DML(insert/update/delete)的区别

- 1) sql 语句不一样
- 2) Select 通过 Statement 对象的 executeQuery()方法获得结果集：
ResultSet rs = stmt.executeQuery()
- 3) DML 语句通过 Statement 对象的 executeUpdate()方法获得操作数：
int n = stmt.executeUpdate();

8.2. Select 与 DML(insert/update/delete)的相同点

- 1) 获得连接(Connection)的方式相同
- 2) 数据库连接状态对象(Statement / PreparedStatement) 两种都可以
需要多次执行的 sql 语句 , 使用 PreparedStatement 性能更好
- 3) 如果使用 PreparedStatement , sql 语句在生成语句对象的时候传递 ,
并绑定占位符(问号)和数据的关系

```
stmt = conn.prepareStatement(sql);  
stmt.setString(1, password);  
stmt.setInt(2, id);  
stmt.executeQuery() 或 executeUpdate();
```

 //参数不写 sql
- 4) 如果使用 Statement , sql 语句在执行时传递

```
stmt = conn.createStatement();  
stmt.executeQuery(sql) 或 executeUpdate(sql)
```
- 5) PreparedStatement 和 Statement 对象 , sql 语句的传递时机不同。