

一、实验介绍

本实验将学习 Statement、PreparedStatement、CallableStatement 等 JDBC 接口的相关知识。

- Statement
- PreparedStatement
- CallableStatement

- JDK1.7
- Eclipse
- Mysql 5.5.50

二、实验步骤

2.1 JDBC 接口概述



我们上节课学习了如何连接数据库，当我们连接上了数据库后，就需要通过 sql 语句对数据库进行操作。随着Java语言应用面的逐步拓宽,Sun 公司开发了一个标准的 SQL 数据库访问接口——JDBC API。它可以使 Java 编程人员通过一个一致的接口,访问多种关系数据库。而今天我们就来学习一下，如何利用 JDBC 的一些核心 API 与数据库进行交互。

通过使用 JDBC Statement, CallableStatement 和 PreparedStatement 接口定义的方法和属性，使可以使用 SQL 或 PL/SQL 命令和从数据库接收数据。它们还定义了许多方法，帮助消除Java 和数据库之间数据类型的差异。

接口	应用场景
Statement	当在运行时使用静态 SQL 语句时（Statement接口不能接受的参数）
CallableStatement	当要访问数据库中的存储过程时 （CallableStatement对象的接口还可以接受运行时输入参数）
PreparedStatement	当计划多次使用 SQL 语句时（PreparedStatement 接口接收在运行时输入参数）


2.2 Statement

我们要使用 Statement 接口，第一步肯定是创建一个 Statement 对象了。我们需要使用 Connection 对象的 createStatement() 方法进行创建。

```
Statement stmt = null;
try {
    stmt = conn.createStatement( );
    . . .
}
catch (SQLException e) {
    . . .
}
finally {
    . . .
}
```

一旦创建了一个Statement对象，我们就可以用它来执行SQL语句了，首先我们先来看看 Statement 里面有哪些方法吧！

方法	说明
boolean execute(String SQL)	如果 ResultSet 对象可以被检索返回布尔值 true，否则返回 false。使用这个方法执行 SQL DDL 语句，或当需要使用真正的动态 SQL
int executeUpdate(String SQL)	用于执行 INSERT、UPDATE 或 DELETE 语句以及 SQLDDL（数据定义语言）语句。返回值是一个整数，指示受影响的行数（即更新计数）

方法	说明
 JDBC 入门教程 (/courses/110)	
ResultSet executeQuery(String SQL)	返回 ResultSet 对象。用于产生单个结果集的语句，例如 SELECT 语句

正如关闭一个 Connection 对象来释放数据库连接资源，出于同样的原因，也应该关闭 Statement 对象。

```
Statement stmt = null;
try {
    stmt = conn.createStatement( );
    . . .
}
catch (SQLException e) {
    . . .
}
finally {
    stmt.close();
}
```

注：如果关闭了 Connection 对象首先它会关闭 Statement 对象，然而应该始终明确关闭 Statement 对象，以确保正确的清除。

2.3 PreparedStatement

PreparedStatement 接口扩展了 Statement 接口，有利于高效地执行多次使用的 SQL 语句。我们先来创建一个 PreparedStatement 对象。Statement 为一条 SQL 语句生成执行计划。如果要执行两条 SQL 语句，会生成两个执行计划。一万个查询就生成一万个执行计划！

```
select colume from table where colume=1;
select colume from table where colume=2;
```

PreparedStatement 用于使用绑定变量重用执行计划。

```
select colume from table where colume=:x;
```

通过 set 不同数据，只需要生成一次执行计划，并且可以重用。

🔍 JDBC 入门教程 (/courses/110)

```
PreparedStatement pstmt = null;
```

```
try {
```

```
/*
```

在JDBC中所有的参数都被代表? 符号, 这是已知的参数标记。在执行SQL语句之前, 必须提供值的每一个参数。

```
*/
```

```
String SQL = "Update Students SET age = ? WHERE id = ?";
```

```
pstmt = conn.prepareStatement(SQL);
```

```
...
```

```
}
```

```
/*
```

setXXX()方法将值绑定到参数, 其中XXX表示希望绑定到输入参数值的 Java 数据类型。如果忘了提供值, 将收到一个 SQLException。

```
*/
```

```
catch (SQLException e) {
```

```
...
```

```
}
```

```
finally {
```

```
//同理, 我们需要关闭 PreparedStatement 对象
```

```
pstmt.close();
```

```
}
```

说了这么多, 我们还是来看代码吧。即上一次我们创建的数据库 EXAMPLE:

🔗 <https://www.shiyanlou.com/courses/110>
JDBC 入门教程 (courses/110)
import java.sql.*;

```
public class JdbcTest {
    // JDBC 驱动器的名称和数据库地址
    static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://localhost/EXAMPLE";

    static final String USER = "root";
    static final String PASS = "";

    public static void main(String[] args) {
        Connection conn = null;
        PreparedStatement stmt = null;
        try{
            //注册 JDBC 驱动器
            Class.forName("com.mysql.jdbc.Driver");

            //打开连接
            System.out.println("Connecting to database...");
            conn = DriverManager.getConnection(DB_URL,USER,PASS);

            //执行查询
            System.out.println("Creating statement...");
            //这里我们要更改一个同学的年龄，参数待定
            String sql = "UPDATE Students set age=? WHERE id=?";
            stmt = conn.prepareStatement(sql);

            //将值绑定到参数，参数从左至右序号为1, 2...
            stmt.setInt(1, 22); // 绑定 age 的值(序号为1)
            stmt.setInt(2, 1); // 绑定 ID 的值

            // 更新 ID 为1的同学的年龄
            int rows = stmt.executeUpdate();
            System.out.println("被影响的行数 : " + rows );

            // 查询所有记录，并显示.
            sql = "SELECT id, name, age FROM Students";
            ResultSet rs = stmt.executeQuery(sql);

            //处理结果集
            while(rs.next()){
                //检索
                int id = rs.getInt("id");
                int age = rs.getInt("age");
                String name = rs.getString("name");

                //显示
                System.out.print("ID: " + id);
                System.out.print(", Age: " + age);
                System.out.print(", Name: " + name);
                System.out.println();
            }
            //清理
            rs.close();
```

🔗 JDBC 入门教程 (courses/110)

```
        stmt.close();
    }catch(SQLException se){
        se.printStackTrace();
    }catch(Exception e){
        e.printStackTrace();
    }finally{
        try{
            if(stmt!=null)
                stmt.close();
        }catch(SQLException se2){
        }
    }
    try{
        if(conn!=null)
            conn.close();
    }catch(SQLException se){
        se.printStackTrace();
    }
}

    System.out.println("Goodbye!");
}
}
```

运行结果：

```
Connecting to database...
Creating statement...
被影响的行数 : 1
ID: 1, Age: 22, Name: Tom
ID: 2, Age: 20, Name: Aby
ID: 3, Age: 22, Name: Bob
ID: 4, Age: 20, Name: Tomson
Goodbye!
```

2.4 CallableStatement

CallableStatement 对象为所有的 DBMS 提供了一种以标准形式调用存储过程的方法。存储过程储存在数据库中。对储存过程的调用是 CallableStatement 对象所含的内容。三种类型的参数有：IN, OUT和INOUT。PreparedStatement对象只使用IN参数。 CallableStatement对象可以使用所有三个

参数	描述
IN	它的值是在创建 SQL 语句时未知的参数，将 IN 参数传给 CallableStatement 对象是通过 setXXX() 方法完成的
OUT	其值由它返回的 SQL 语句提供的参数。从 OUT 参数的 getXXX() 方法检索值
INOUT	同时提供输入和输出值的参数，绑定的 setXXX() 方法的变量，并使用 getXXX() 方法检索值

在 JDBC 中调用存储过程的语法如下所示。注意，方括号表示其间的内容是可选项；方括号本身并不是语法的组成部分。

```
{call 存储过程名[(?, ?, ...)]}
```

返回结果参数的过程的语法为：

```
{? = call 存储过程名[(?, ?, ...)]}
```

不带参数的存储过程的语法类似：

```
{call 存储过程名}
```

CallableStatement 对象是用 Connection 方法 prepareCall 创建的。

```
CallableStatement cstmt = null;
try {
    String SQL = "{call getEXAMPLEName (?, ?)}";
    cstmt = conn.prepareCall (SQL);
    . . .
}
catch (SQLException e) {
    . . .
}
finally {
    cstmt.close();
}
```

好了，CallableStatement 接口就不再详细地讲解了，同学们可以自己查阅相关的资料，对 CallableStatement 进行深入学习。

三、实验总结

本次课程介绍了 JDBC 接口的相关知识，下一节我们将进入 JDBC 结果集的学习。

四、课后习题

请同学们将上面的代码在 Eclipse 上再打一遍，熟悉各个对象的用法。

*本课程内容，由作者授权实验楼发布，未经允许，禁止转载、下载及非法传播。

上一节：JDBC 基础 (/courses/110/labs/1193/document)

🔍 JDBC 入门教程 (/courses/110) 下一节: JDBC 结果集 (/courses/110/labs/1196/document)
