typora-copy-images-to: img

day19-MySQL函数和JDBC

今日内容

- Mysql常见的函数
- JDBC---CRUD----------重点掌握
- JDBC---事务操作----------重点掌握

第一章-MySql常见的函数

1.1 MySql函数的介绍

使用MySql函数的目的

为了简化操作，MySql提供了大量的函数给程序员使用（比如你想输入当前时间，可以调用now()函数）

函数可以出现的位置

插入语句的values()中，更新语句中，删除语句中，查询语句及其子句中。

环境准备

```
create database day19_1;
use day19_1;
-- 用户表
CREATE TABLE t_user (
  id int primary key AUTO_INCREMENT,
  uname varchar(40),
  age int,
  sex int
);

insert  into t_user values (null,'zs',18,1);
insert  into t_user values (null,'ls',20,0);
insert  into t_user values (null,'ww',23,1);
insert  into t_user values (null,'zl',24,1);
insert  into t_user values (null,'lq',15,0);
insert  into t_user values (null,'hh',12,0);
insert  into t_user values (null,'wzx',60,null);
insert  into t_user values (null,'lb',null,null);
```

1.2 if相关函数

- 1.if(expr1,expr2,expr3)
  如果 expr1 是TRUE，则 IF()的返回值为expr2; 否则返回值则为 expr3。if() 的返回值为数字值或字符串值，具体情况视其所在语境而定。
  - 需求: 查询姓名,年龄,性别,如果性别为1,就显示1,否则就显示0

```
select uname,age,if(sex=1,1,0) from t_user;
select uname,age,if(sex,1,0) from t_user;
```

- 2.ifnull(expr1,expr2)
  假如expr1 不为 NULL，则 IFNULL() 的返回值为 expr1; 否则其返回值为 expr2。ifnull()的返回值是数字或是字符串，具体情况取决于其所使用的语境。
  - 需求: 查询姓名,年龄,性别,如果性别为null,就显示2

```
select uname,age,ifnull(sex,2) from t_user;
```

1.3 字符串函数

- 函数:

```
1.concat(str1, str2, ...)   字符串连接函数，可以将多个字符串进行连接
2.concat_ws(separator, str1, str2, ...) 可以指定间隔符将多个字符串进行连接;
3.upper(str) 得到str的大写形式
4.lower(str)  得到str的小写形式
5.trim(str)  将str两边的空白符移除
6.substr(str,pos)、substring(str,pos)  ,substr(str,pos,len)、
substring(str,pos,len) 截取字符串
```

- 练习:

```
-- - 需求:使用concat查询t_user表中的uname,显示格式为: 你好,uname
select concat("你好,",uname) from t_user;

-- - 需求:使用concat_ws查询t_user表中的uname,显示格式为: 你好,uname
select concat_ws(",","你好",uname) from t_user;

-- - 需求:使用upper查询查询t_user表中的uname
select upper(uname) from t_user;

-- - 需求:使用lower查询查询t_user表中的uname
select lower(uname) from t_user;

-- - 需求:使用trim查询查询t_user表中的uname
select trim(uname) from t_user;

-- - 需求:获取helloworld从第二个字符开始的完整子串
select substr("helloworld",2);
select substring("helloworld",2);

-- - 需求:获取helloworld从第二个字符开始,长度为4的子串
select substr("helloworld",2,4);
select substring("helloworld",2,4);
```

1.4 时间日期函数

- 函数

```
current_date()  获取当前日期，如 2019-10-18
current_time()  获取当前时：分：秒，如：15:36:11
now()  获取当前的日期和时间，如：2019-10-18 15:37:17
```

- 练习

```
-- 1.获取当前的日期
select current_date();

-- 2.获取当前的时间
select current_time();

-- 3.获取当前的日期和时间
select now();
```

1.5 数值函数

- 函数:

```
abs(x) 获取数值x的绝对值
ceil(x)  向上取整，获取不小于x的整数值
floor(x) 向下取整，获取不大于x的整数值
pow(x，y) 获取x的y次幂
rand()  获取一个0-1之间的随机浮点数
```

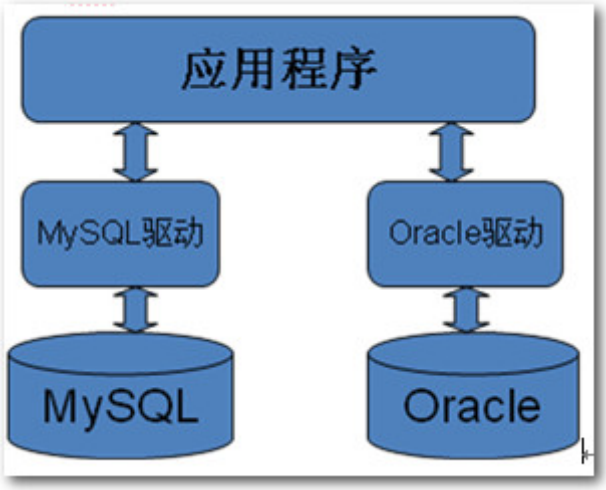
- 练习:

```
-- - 需求:  获取-11的绝对值
select abs(-11);

-- - 需求: 获取大于3.14的最小整数
select ceil(3.14);

-- - 需求: 获取小于3.14的最大整数
select floor(3.14);

-- - 需求: 获取2的5次幂
select pow(2,5);

-- - 需求: 获取一个0-100之间的随机数
select rand()*100;
```

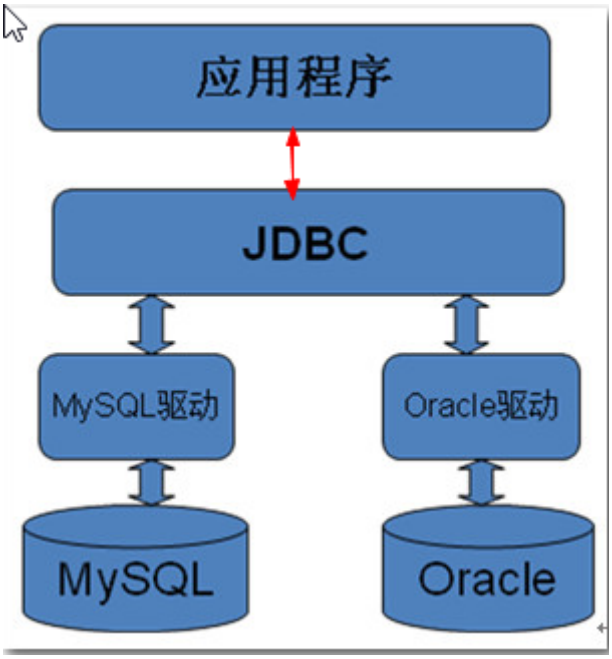
第二章-JDBC入门

2.1 JDBC概述

1. 没有JDBC



2. 有了JDBC后



什么是JDBC

JDBC(java database connectivity): sun公司为了简化和统一java连接数据库,定义的一套**规范(**类,接口).

JDBC和驱动的关系

接口(JDBC)与**实现**(驱动jar包)的关系

2.2 JDBC快速入门

准备工作

```sql
create database day18;
use day18;
create table user(
    id int primary key auto_increment,
    username varchar(20),
    password varchar(20),
    nickname varchar(20)
);

INSERT INTO `USER` VALUES(null,'zs','123456','老张');
INSERT INTO `USER` VALUES(null,'ls','123456','老李');
INSERT INTO `USER` VALUES(null,'wangwu','123','东方不败');
```

1.需求

查询所有的用户, 输出到控制台

2.步骤

- 在java项目模块下,导入mysql驱动包
- 注册驱动
- 获得连接
- 创建执行sql语句对象
- 执行sql语句,处理结果
- 释放资源

3.代码实现

```java
/**
 * @Author: pengzhilin
 * @Date: 2021/4/27 9:47
 */
public class Test {
    public static void main(String[] args) throws SQLException {
        // 1.注册驱动
        DriverManager.registerDriver(new Driver());

        // 2.获得连接
        String url = "jdbc:mysql://localhost:3306/day19_1";
        String user = "root";
        String password = "root";
        Connection connection = DriverManager.getConnection(url, user,
password);

        // 3.创建执行sql语句对象
        Statement statement = connection.createStatement();

        // 4.执行sql语句,处理结果
        String sql = "select * from user";
        ResultSet resultSet = statement.executeQuery(sql);

        // 处理结果
        while (resultSet.next()) {
            // 取数据
            System.out.println(resultSet.getObject("id"));
```

```
            System.out.println(resultSet.getObject("username"));
            System.out.println(resultSet.getObject("password"));
            System.out.println(resultSet.getObject("nickname"));
            System.out.println("--------------");
        }

        // 5.释放资源
        resultSet.close();
        statement.close();
        connection.close();

    }
}
```

第三章-JDBC API详解

API-Drivermanager类

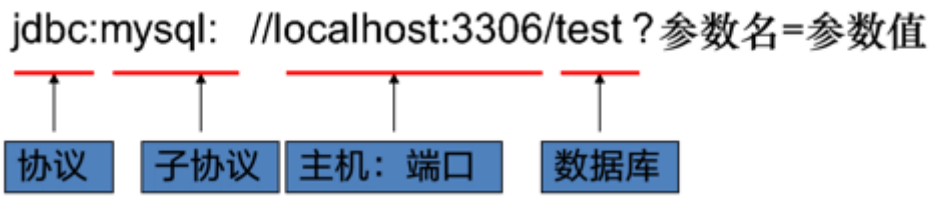
1. `public static void registerDriver(Driver driver) ;`注册驱动

```
com.mysql.jdbc.Driver类:
static {
    try {
        java.sql.DriverManager.registerDriver(new Driver());
    } catch (SQLException E) {
        throw new RuntimeException("Can't register driver!");
    }
}
```

翻阅源码发现,通过API的方式注册驱动,Driver会new两次,所有推荐这种写法:

```
Class.forName("com.mysql.jdbc.Driver");  // 当获取Driver类的字节码对象,就会加载Driver类,执行静态代码块,执行静态代码块就会注册驱动
```

2. `public static Connection getConnection(String url, String user, String password) ;`与数据库建立连接



API-Connection接口

1. 概述

接口的实现在数据库驱动中。所有与数据库交互都是==基于连接对象==的。

2. 常用方法

```
Statement createStatement();创建执行sql语句对象

PreparedStatement prepareStatement(String sql) 预编译sql语句,返回预编译sql语句对象
PreparedStatement继承statement接口

void setAutoCommit(boolean autoCommit) false--手动开启事务(start transaction)
void commit();提交事务
void rollback();回滚事务
```

API-Statement接口

1. 概述

   接口的实现在数据库驱动中. 用来==**操作sql语句(增删改查)==，并返回sql语句的结果**

2. 作用

   `ResultSet executeQuery(String sql)` 根据查询语句返回结果集。只能执行**select**语句。

   `int executeUpdate(String sql)` 根据执行的DML（insert update delete）语句，返回受影响的行数。

   `boolean execute(String sql)` 此方法可以执行任意sql语句。返回boolean值. 【了解】

   true: 执行select语句

   false: 执行insert, delete,update语句

API-ResultSet接口

1. **作用: 用来封装查询后的结果集,表示一个结果集对象;**

   提供一个游标，默认游标指向结果集第一行之前。

   调用一次next()，游标向下移动一行。

   提供一些get方法。

2. ResultSet接口常用API

   - **boolean next();将光标从当前位置向下移动一行**
   - int getInt(int colIndex)以int形式获取ResultSet结果集当前行指定列号值
   - **int getInt(String colLabel)以int形式获取ResultSet结果集当前行指定列名值**
   - float getFloat(int colIndex)以float形式获取ResultSet结果集当前行指定列号值
   - **float getFloat(String colLabel)以float形式获取ResultSet结果集当前行指定列名值**
   - String getString(int colIndex)以String 形式获取ResultSet结果集当前行指定列号值
   - **String getString(String colLabel)以String形式获取ResultSet结果集当前行指定列名值**
   - Date getDate(int columnIndex); 以Date 形式获取ResultSet结果集当前行指定列号值
   - **Date getDate(String columnName);以Date形式获取ResultSet结果集当前行指定列名值**
   - **Object getObject(String columnLabel)以Object形式获取ResultSet结果集当前行指定列名值**

- ...
- **void close()关闭ResultSet 对象**

```
1. next()
   *默认情况下, 光标在表头位置
   *每次调用一下next(), 光标向下移动一行;如果当前行有数据,返回
true, 如果当前行没有数据, 返回false
2. getxxx(String 列名);     xxx代表类型
   getxxx(int 列的索引);    xxx代表类型
   *建议通过列名来获得
   *如果为了封装, 建议这个列是什么类型 就通过get类型()来获得
```

id	username	password	nickname
1	zs	123456	老张
2	ls	123456	老李
3	wangwu	123	东方不败
4	zl	123	赵六

- 改写处理查询结果

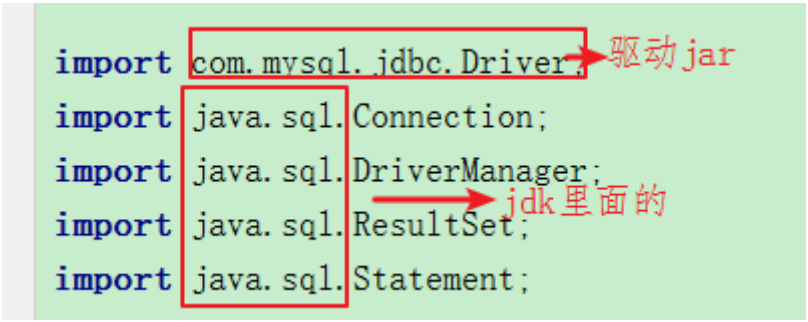
```
    // 处理结果
        while (resultSet.next()) {
            // 取数据
            /*System.out.println(resultSet.getObject("id"));
            System.out.println(resultSet.getObject("username"));
            System.out.println(resultSet.getObject("password"));
            System.out.println(resultSet.getObject("nickname"));*/

            int id = resultSet.getInt("id");
            String username = resultSet.getString("username");
            String pwd = resultSet.getString("password");
            String nickname = resultSet.getString("nickname");

            System.out.println("id:"+id);
            System.out.println("username:"+username);
            System.out.println("pwd:"+pwd);
            System.out.println("nickname:"+nickname);
            System.out.println("--------------");
        }
```

API-总结

1. DriverManager:驱动管理器
   - 注册驱动
   - 获得连接
2. Connection: 代表连接对象
   - 创建执行sql语句对象
   - 创建预编译sql语句对象
   - 事务操作
3. Statement: 执行sql语句对象
   - 执行查询 Result executeQuery(String sql) 返回结果集
   - 执行增删改 int excuteUpdate(String sql) 返回受影响的行数
4. ResultSet: 结果集
   - boolen next() 每调用一次, 光标就向下移动一行; 这个行有数据, 返回true; 没有数据, 返回false
   - get类型(String 列名); 根据列名 获得当前列的数据

注意事项

- 包名



- 结果封装
  - 概述: 结果集一般会封装成一个对象,要求该对象所属的类的所有属性必须和表中的列名一致
  - 设计数据库:
    - 一个项目对应一个数据库
    - 一个类对应一张表
    - 一个对象对应一条记录
  - 封装:
    - 创建一个User类--->基本类型的属性最好定义为对应的包装类类型----为了避免字段的值为null的问题

```
public class User {
    /**
     * id
     */
    private Integer id;
    /**
     * username
     */
    private String username;
    /**
     * password
     */
    private String password;
    /**
     * nickname
     */
    private String nickname;

    public User(Integer id, String username, String password,
String nickname) {
        this.id = id;
        this.username = username;
        this.password = password;
        this.nickname = nickname;
    }

    public User() {
    }
```

```
    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getNickname() {
        return nickname;
    }

    public void setNickname(String nickname) {
        this.nickname = nickname;
    }

    @Override
    public String toString() {
        return "User{" +
                "id=" + id +
                ", username='" + username + '\'' +
                ", password='" + password + '\'' +
                ", nickname='" + nickname + '\'' +
                '}';
    }
}
```

- 对结果集进行封装
  - 结果集如果是一条记录,就封装成一个对象
  - 结果集如果是多条记录,每条记录封装成一个对象,再把每个对象添加到集合中

```
public class Test1 {
    public static void main(String[] args) throws Exception {
        // 1.注册驱动
        //DriverManager.registerDriver(new Driver());
        // 改: 只要加载Driver类就会注册驱动(Driver类的静态代码块中会注册驱动)
```

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

        // 2.获得连接
        String url = "jdbc:mysql://localhost:3306/day19_1";
        String user = "root";
        String password = "root";
        Connection connection =
DriverManager.getConnection(url, user, password);

        // 3.创建执行sql语句对象
        Statement statement = connection.createStatement();

        // 4.执行sql语句,处理结果
        String sql = "select * from user";
        ResultSet resultSet = statement.executeQuery(sql);

        // 处理结果
        ArrayList<User> list = new ArrayList<>();
        while (resultSet.next()) {
            // 创建User对象
            User user1 = new User();

            // 取值
            int id = resultSet.getInt("id");
            String username = resultSet.getString("username");
            String pwd = resultSet.getString("password");
            String nickname = resultSet.getString("nickname");

            // 设置值
            user1.setId(id);
            user1.setUsername(username);
            user1.setPassword(pwd);
            user1.setNickname(nickname);

            // 添加到集合中
            list.add(user1);
        }

        // 5.释放资源
        resultSet.close();
        statement.close();
        connection.close();

        for (User user1 : list) {
            System.out.println(user1);
        }

    }
}
```

第四章-JDBC操作数据库练习

4.1 增删改查练习

- 需求一：查询所有用户信息

```
public class Test5_查多条记录 {
    public static void main(String[] args) throws Exception{
        // 1.注册驱动
        Class.forName("com.mysql.jdbc.Driver");

        // 2.获得连接
        String url = "jdbc:mysql://localhost:3306/day19_1";
        Connection connection = DriverManager.getConnection(url, "root",
"root");

        // 3.创建执行sql语句对象
        Statement statement = connection.createStatement();

        // 4.执行sql语句,处理结果
        String sql = "select * from user";
        ResultSet resultSet = statement.executeQuery(sql);
        // 创建User对象
        User user = null;

        // 创建ArrayList集合对象,集合元素的类型为User
        ArrayList<User> list = new ArrayList<>();

        while (resultSet.next()){
            // 创建User对象
            user = new User(); // 1.优化内存     2.可以作为判断条件

            // 取值,然后赋值
            user.setId( resultSet.getInt("id"));
            user.setUsername(resultSet.getString("username"));
            user.setPassword(resultSet.getString("password"));
            user.setNickname(resultSet.getString("nickname"));

            // 把user对象添加到集合中
            list.add(user);
        }


        // 5.释放资源
        resultSet.close();
        statement.close();
        connection.close();


        for (User user1 : list) {
            System.out.println(user1);
        }


    }
```

```
}
```

- 需求二：根据id查询用户信息

```
package com.itheima.demo2_增删改查练习;

import com.bean.User;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

/**
 * @Author: pengzhilin
 * @Date: 2021/4/27 11:00
 */
public class Test4_查一条记录 {
    // 查询id为1的记录
    public static void main(String[] args) throws Exception {
        // 1.注册驱动
        Class.forName("com.mysql.jdbc.Driver");

        // 2.获得连接
        String url = "jdbc:mysql://localhost:3306/day19_1";
        Connection connection = DriverManager.getConnection(url, "root",
"root");

        // 3.创建执行sql语句对象
        Statement statement = connection.createStatement();

        // 4.执行sql语句,处理结果
        String sql = "select * from user where id = 1";
        ResultSet resultSet = statement.executeQuery(sql);
        // 创建User对象
        User user = null;

        while (resultSet.next()){
            // 创建User对象
            user = new User(); // 1.优化内存     2.可以作为判断条件

            // 取值,然后赋值
            user.setId( resultSet.getInt("id"));
            user.setUsername(resultSet.getString("username"));
            user.setPassword(resultSet.getString("password"));
            user.setNickname(resultSet.getString("nickname"));
        }


        // 5.释放资源
        resultSet.close();
        statement.close();
        connection.close();
```

```
        if (user == null){
            System.out.println("没有查询到数据!");
        }else{
            System.out.println("查询到了数据: "+user);
        }

    }
}
```

- 需求三：新增用户信息

```
package com.itheima.demo2_增删改查练习;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

/**
 * @Author: pengzhilin
 * @Date: 2021/4/27 11:00
 */
public class Test1_增 {
    public static void main(String[] args) throws Exception{
        // 1.注册驱动
        Class.forName("com.mysql.jdbc.Driver");

        // 2.获取连接
        String url = "jdbc:mysql://localhost:3306/day19_1";
        Connection connection = DriverManager.getConnection(url, "root",
"root");

        // 3.创建执行sql语句对象
        Statement statement = connection.createStatement();

        // 4.执行sql语句,处理结果
        String sql = "insert into user values(null,'zl','123456','赵六')";
        int rows = statement.executeUpdate(sql);
        System.out.println("受影响的行数:"+rows);

        // 5.释放资源
        statement.close();
        connection.close();

    }
}
```

- 需求四：修改用户信息

```
package com.itheima.demo2_增删改查练习;

import java.sql.Connection;
```

```
import java.sql.DriverManager;
import java.sql.Statement;

/**
 * @Author: pengzhilin
 * @Date: 2021/4/27 11:00
 */
public class Test2_改 {
    // 修改wangwu的密码为abcdef
    public static void main(String[] args) throws Exception {
        // 1.注册驱动
        Class.forName("com.mysql.jdbc.Driver");

        // 2.获得连接
        String url = "jdbc:mysql://localhost:3306/day19_1";
        Connection connection = DriverManager.getConnection(url, "root",
"root");

        // 3.创建执行sql语句对象
        Statement statement = connection.createStatement();

        // 4.执行sql语句,处理结果
        String sql = "update user set password = 'abcdef' where username =
'wangwu'";
        int rows = statement.executeUpdate(sql);
        System.out.println("受影响的行数:"+rows);

        // 5.释放资源
        statement.close();
        connection.close();
    }
}
```

- 需求五：删除用户信息

```
package com.itheima.demo2_增删改查练习;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

/**
 * @Author: pengzhilin
 * @Date: 2021/4/27 11:00
 */
public class Test3_删 {
    // 删除id为3的记录
    public static void main(String[] args) throws Exception {
        // 1.注册驱动
        Class.forName("com.mysql.jdbc.Driver");

        // 2.获得连接
        String url = "jdbc:mysql://localhost:3306/day19_1";
```

```
        Connection connection = DriverManager.getConnection(url, "root",
"root");

        // 3.创建执行sql语句对象
        Statement statement = connection.createStatement();

        // 4.执行sql语句,处理结果
        String sql = "delete from user where id = 3";
        int rows = statement.executeUpdate(sql);
        System.out.println("受影响的行数:" + rows);

        // 5.释放资源
        statement.close();
        connection.close();
    }
}
```

4.2 JDBC工具类的抽取

- 配置文件

```
driverClass=com.mysql.jdbc.Driver
url=jdbc:mysql://localhost:3306/day19_1
username=root
password=root
```

- 工具类

```
package com.utils;

import java.io.IOException;
import java.io.InputStream;
import java.sql.*;
import java.util.Properties;

/**
 * @Author: pengzhilin
 * @Date: 2021/4/27 11:23
 */
public class JDBCUtils {
    private static String driverClass;
    private static String url;
    private static String username;
    private static String password;

    static {
        try {
            // 1.创建Properties对象
            Properties pro = new Properties();

            // 2.加载配置文件
            InputStream is =
JDBCUtils.class.getClassLoader().getResourceAsStream("db.properties");
```

```
            pro.load(is);

            // 3.取值
            driverClass = pro.getProperty("driverClass");
            url = pro.getProperty("url");
            username = pro.getProperty("username");
            password = pro.getProperty("password");

            // 1.注册驱动
            Class.forName(driverClass);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /**
     * 获得连接
     *
     * @return
     * @throws Exception
     */
    public static Connection getConnection() throws Exception {
        // 2.获得连接
        Connection connection = DriverManager.getConnection(url, username,
password);
        return connection;
    }

    /**
     * 释放资源
     *
     * @param resultSet
     * @param statement
     * @param connection
     */
    public static void release(ResultSet resultSet, Statement statement,
Connection connection) {
        if (resultSet != null) {
            try {
                resultSet.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }

        if (statement != null) {
            try {
                statement.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }

        if (connection != null) {
            try {
                connection.close();
            } catch (SQLException e) {
```

```
                e.printStackTrace();
            }
        }
    }

}
```

- 测试工具类:

```
package com.itheima.demo3_测试工具类;

import com.utils.JDBCUtils;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

/**
 * @Author：pengzhilin
 * @Date: 2021/4/27 11:00
 */
public class Test1_增 {
    public static void main(String[] args) throws Exception{
        // 1.注册驱动,获得连接
        Connection connection = JDBCUtils.getConnection();

        // 3.创建执行sql语句对象
        Statement statement = connection.createStatement();

        // 4.执行sql语句,处理结果
        String sql = "insert into user values(null,'wangwu','123456','王
五')";
        int rows = statement.executeUpdate(sql);
        System.out.println("受影响的行数:"+rows);

        // 5.释放资源
        JDBCUtils.release(null,statement,connection);

    }
}
```

```
package com.itheima.demo3_测试工具类;

import com.bean.User;
import com.utils.JDBCUtils;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;

/**
 * @Author：pengzhilin
```

```
 * @Date: 2021/4/27 11:00
 */
public class Test5_查多条记录 {
    public static void main(String[] args) throws Exception{
        // 1.注册驱动,获得连接
        Connection connection = JDBCUtils.getConnection();

        // 3.创建执行sql语句对象
        Statement statement = connection.createStatement();

        // 4.执行sql语句,处理结果
        String sql = "select * from user";
        ResultSet resultSet = statement.executeQuery(sql);
        // 创建User对象
        User user = null;

        // 创建ArrayList集合对象,集合元素的类型为User
        ArrayList<User> list = new ArrayList<>();

        while (resultSet.next()){
            // 创建User对象
            user = new User(); // 1.优化内存     2.可以作为判断条件

            // 取值,然后赋值
            user.setId( resultSet.getInt("id"));
            user.setUsername(resultSet.getString("username"));
            user.setPassword(resultSet.getString("password"));
            user.setNickname(resultSet.getString("nickname"));

            // 把user对象添加到集合中
            list.add(user);
        }


        // 5.释放资源
        JDBCUtils.release(resultSet,statement,connection);


        for (User user1 : list) {
            System.out.println(user1);
        }


    }
}
```

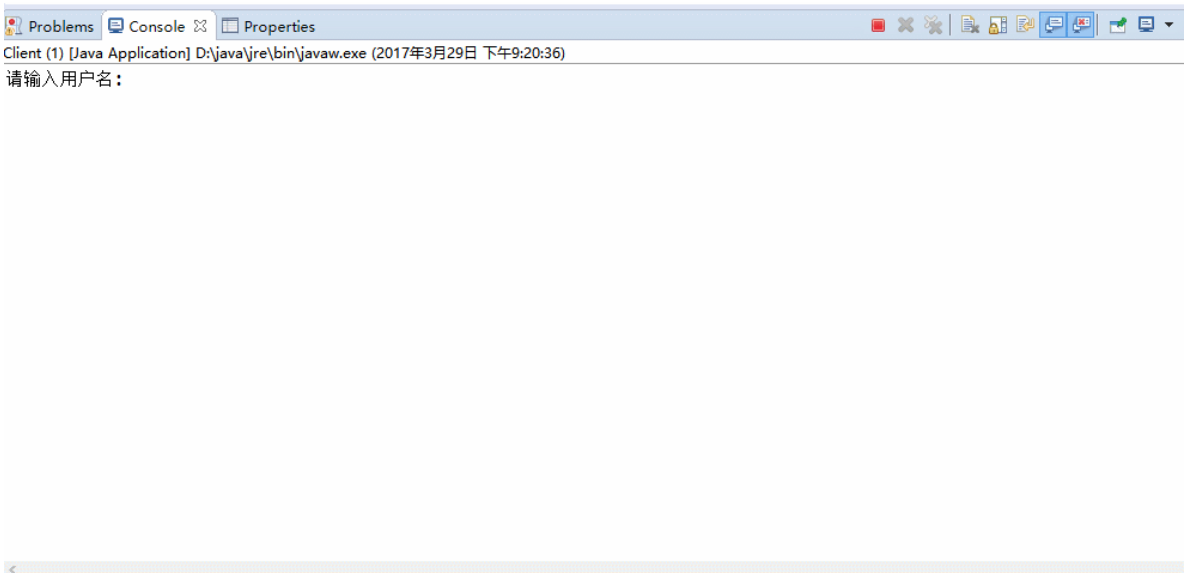
第五章-PreparedStatement

5.1 登录案例

1.需求

在控制台输入用户名和密码,查询数据库,如果数据库存在当前用户,显示登录成功!

如果数据库不存在当前用户,显示登录失败!



2. 分析

- 其实这个登录案例---就是根据用户输入的用户名和密码去数据库中查找记录,如果查询到了,就登录成功,查询不到就登录失败
- 实现步骤:
    - 用户输入用户名和密码
    - 注册驱动,获得连接
    - 创建执行sql语句对象
    - 执行sql语句,处理结果(判断是否登录成功,其实就是判断User对象是否为null)
    - 释放资源

3.代码实现

```
package com.itheima.demo4_登录案例;

import com.bean.User;
import com.utils.JDBCUtils;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Scanner;

/**
 * @Author: pengzhilin
 * @Date: 2021/4/27 11:49
 */
public class Test {
    public static void main(String[] args) throws Exception {
        //- 用户输入用户名和密码
        Scanner sc = new Scanner(System.in);
        System.out.println("请输入用户名:");
        String username = sc.nextLine();// zs

        System.out.println("请输入密码:");
```

```
        String password = sc.nextLine();// 123456

        //- 注册驱动,获得连接
        Connection connection = JDBCUtils.getConnection();

        //- 创建执行sql语句对象
        Statement statement = connection.createStatement();

        //- 执行sql语句,处理结果(判断是否登录成功,其实就是判断User对象是否为null)
        //String sql = "select * from user where username = 'zs' and password =
'123456'";
        String sql = "select * from user where username = '" + username + "' and
password = '" + password + "'";
        ResultSet resultSet = statement.executeQuery(sql);
        // 定义User变量
        User user = null;
        while (resultSet.next()) {
            // 创建User对象
            user = new User();
            // 封装数据
            user.setId(resultSet.getInt("id"));
            user.setUsername(resultSet.getString("username"));
            user.setPassword(resultSet.getString("password"));
            user.setNickname(resultSet.getString("nickname"));

        }

        //- 释放资源
        JDBCUtils.release(resultSet, statement, connection);

        // 判断是否登录成功
        if (user == null) {
            System.out.println("登录失败!");
        } else {
            System.out.println("登录成功!");
        }
    }
}
```

4 演示SQL注入攻击

- 输入的用户名是对的,但输入的密码为: **123' or '1=1** 发现直接登录成功---->sql注入漏洞
- 输入的用户名是不对的,但输入的密码为: **123' or '1=1** 发现直接登录成功---->sql注入漏洞

5.2 登录中SQL注入问题分析和解决

- 问题分析:

```
输入的用户名: zs
输入的密码: 123' or '1=1
sql语句:
```

```
String sql = "select * from user where username = '" + username + "' and
password = '" + password + "'";


sql拼接后分析:
select * from user where username = 'zs' and password = '123' or '1=1';
select * from user where username = 'zs' and password = '123' or true;
select * from user where true;
select * from user;

分析:
拼接之前sql语句的结构:  select * from user where username = '用户名' and
password = '密码'
拼接之后sql语句的结构:  select * from user where username = '用户名' and
password = '密码'  or true;

原因:
    sql语句是做一个简单的字符串拼接,所以会把用户输入的密码中的or关键字注入到sql语句中,从
而改变了sql语句的结构

解决方式:
    固定sql语句的格式---->使用预编译sql语句对象,对sql语句的格式进行固定,哪怕后期拼接的
内容里面含有关键字,也不会改变sql语句的格式
```

- PreparedStatement接口: 继承Statement接口
  - 作用: 在执行sql语句之前，将sql语句进行提前编译。编译后, 就明确了sql语句的格式，以后就不会再改变了。剩余的内容都会认为是参数!
  - SQL语句中的参数使用?作为占位符 `eg:select * from user where username = ? and password = ?`
  - 获得预编译sql语句对象: 使用Connection的方法
    - `PreparedStatement prepareStatement(String sql)` 预编译sql语句,得到预编译sql语句对象
  - 为?占位符赋值的方法：setXXX(参数1,参数2);
    - XXX代表：数据类型----->列的类型
    - 参数1：?的位置编号(编号从1开始)
    - 参数2: 具体的值
  - 执行sql语句:
    - 执行查询语句:
      - `ResultSet executeQuery()`根据查询语句返回结果集。只能执行**select**语句。
    - 执行增删改语句:
      - `int executeUpdate()` 根据执行的DML（insert update delete）语句，返回受影响的行数。
    - **注意: 预编译sql语句对象执行sql语句的方法不需要传入sql语句**
  - **使用PreparedStatement对象解决登录案例中的sql注入漏洞问题:**

```
package com.itheima.demo5_解决登录案例的sql注入问题;

import com.bean.User;
import com.utils.JDBCUtils;
```

```java
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.Scanner;

/**
 * @Author: pengzhilin
 * @Date: 2021/4/27 12:21
 */
public class Test {
    public static void main(String[] args) throws Exception{
        //- 用户输入用户名和密码
        Scanner sc = new Scanner(System.in);
        System.out.println("请输入用户名:");
        String username = sc.nextLine();// zs

        System.out.println("请输入密码:");
        String password = sc.nextLine();// 123456

        //- 注册驱动,获得连接
        Connection connection = JDBCUtils.getConnection();

        // 预编译sql语句,得到预编译对象
        String sql = "select * from user where username = ? and
password = ?";
        PreparedStatement ps = connection.prepareStatement(sql);

        // 给sql语句设置参数
        ps.setString(1,username);
        ps.setString(2,password);

        // 执行sql语句,处理结果
        ResultSet resultSet = ps.executeQuery();
        User user = null;
        while (resultSet.next()){
            // 创建User对象
            user = new User();
            // 封装数据
            user.setId(resultSet.getInt("id"));
            user.setUsername(resultSet.getString("username"));
            user.setPassword(resultSet.getString("password"));
            user.setNickname(resultSet.getString("nickname"));

        }

        JDBCUtils.release(resultSet, ps, connection);

        // 判断是否登录成功
        if (user == null) {
            System.out.println("登录失败!");
        } else {
            System.out.println("登录成功!");
        }

    }
}
```

5.3 使用preparedStatement完成CRUD

1.需求

通过PreparedStatement完成增、删、改、查

2.分析

- 1.注册驱动,获得连接
- 2.预编译sql语句,得到预编译对象
- 3.为sql语句设置参数
- 4.执行sql语句,处理结果
- 5.释放资源

3.实现

- 增

```
public class Test1_增 {
    public static void main(String[] args) throws Exception {
        //- 1.注册驱动,获得连接
        Connection connection = JDBCUtils.getConnection();

        //- 2.预编译sql语句,得到预编译对象
        String sql = "insert into user values(null,?,?,?)";
        PreparedStatement ps = connection.prepareStatement(sql);

        //- 3.为sql语句设置参数
        ps.setString(1, "tq");
        ps.setString(2, "123456");
        ps.setString(3, "田七");

        //- 4.执行sql语句,处理结果
        int rows = ps.executeUpdate();
        System.out.println("受影响的行数:" + rows);

        //- 5.释放资源
        JDBCUtils.release(null,ps,connection);
    }
}
```

- 删

```
public class Test3_删 {
    // 删除username=wangwu的记录
    public static void main(String[] args) throws Exception {
        // 1.注册驱动,获得连接
        Connection connection = JDBCUtils.getConnection();
```

```
        // 2.预编译sql语句,得到预编译对象
        String sql = "delete from user where username = ?";
        PreparedStatement ps = connection.prepareStatement(sql);

        // 3.为sql语句设置参数
        ps.setString(1,"wangwu");

        // 4.执行sql语句,处理结果
        int rows = ps.executeUpdate();
        System.out.println("受影响的行数:"+rows);

        // 5.释放资源
        JDBCUtils.release(null,ps,connection);


    }
}
```

- 改

```
public class Test2_改 {
    // 修改id为2的记录的密码为abcdef
    public static void main(String[] args) throws Exception {
        // 1.注册驱动,获得连接
        Connection connection = JDBCUtils.getConnection();

        // 2.预编译sql语句,得到预编译对象
        String sql = "update user set password = ? where id = ?";
        PreparedStatement ps = connection.prepareStatement(sql);

        // 3.为sql语句设置参数
        ps.setString(1,"abcdef");
        ps.setInt(2,2);

        // 4.执行sql语句,处理结果
        int rows = ps.executeUpdate();
        System.out.println("受影响的行数:"+rows);

        // 5.释放资源
        JDBCUtils.release(null,ps,connection);

    }
}
```

- 查
  - 查询所有记录

```
public class Test4_查询所有记录 {
    public static void main(String[] args) throws Exception{
        // 1.注册驱动,获得连接
        Connection connection = JDBCUtils.getConnection();
```

```
        // 2.预编译sql语句,得到预编译对象
        String sql = "select * from user";
        PreparedStatement ps = connection.prepareStatement(sql);

        // 3.设置参数---->sql语句没有参数
        // 4.执行sql语句,处理结果
        ResultSet resultSet = ps.executeQuery();

        // 创建集合对象
        ArrayList<User> list = new ArrayList<>();

        // 循环遍历
        while (resultSet.next()) {
            // 创建User对象
            User user = new User();

            // 获取数据,并封装到User对象中
            user.setId(resultSet.getInt("id"));
            user.setUsername(resultSet.getString("username"));
            user.setPassword(resultSet.getString("password"));
            user.setNickname(resultSet.getString("nickname"));

            // 把User对象添加到集合中
            list.add(user);

        }

        // 5.释放资源
        JDBCUtils.release(resultSet,ps,connection);

        for (User user : list) {
            System.out.println(user);
        }

    }
}
```

- 查询一条记录---->id为1的记录

```
public class Test5_查询单条记录 {
    // 查询id为1的记录
    public static void main(String[] args) throws Exception{
        // 1.注册驱动,获得连接
        Connection connection = JDBCUtils.getConnection();

        // 2.预编译sql语句,得到预编译对象
        String sql = "select * from user where id = ?";
        PreparedStatement ps = connection.prepareStatement(sql);

        // 3.设置参数
        ps.setInt(1,1);

        // 4.执行sql语句,处理结果
        ResultSet resultSet = ps.executeQuery();
```

```
        // 定义User遍历
        User user = null;

        // 循环遍历
        while (resultSet.next()) {
            // 创建User对象
            user = new User();

            // 获取数据,并封装到User对象中
            user.setId(resultSet.getInt("id"));
            user.setUsername(resultSet.getString("username"));
            user.setPassword(resultSet.getString("password"));
            user.setNickname(resultSet.getString("nickname"));
        }

        // 5.释放资源
        JDBCUtils.release(resultSet,ps,connection);

        System.out.println(user);

    }
}
```

第六章-JDBC事务的处理

6.1 JDBC事务介绍

- 管理事务的功能类：Connection接口
  - 开启事务：`setAutoCommit(boolean autoCommit);` 参数为false，则开启事务。
  - 提交事务：`commit();`
  - 回滚事务：`rollback();`
- 案例:

```
public class Test1_事务入门 {
    // 修改id为6的密码
    public static void main(String[] args) {
        Connection connection = null;
        PreparedStatement ps = null;
        try {
            // 1.注册驱动,获得连接
            connection = JDBCUtils.getConnection();

            // 2.开启事务
            connection.setAutoCommit(false);

            // 3.预编译sql语句,得到预编译对象
            String sql = "update user set password = ? where id = ?";
            ps = connection.prepareStatement(sql);

            // 4.设置参数
```

```
            ps.setString(1,"abcdef");
            ps.setInt(2,6);

            // 5.执行sql语句,处理结果
            int rows = ps.executeUpdate();
            System.out.println("受影响的行数:"+rows);

            // 发生异常
            int i = 1/0;

            // 6.没有异常提交事务
            connection.commit();

        } catch (Exception e) {
            // 7.有异常回滚事务
            try {
                connection.rollback();
            } catch (SQLException e1) {
                e1.printStackTrace();
            }

        } finally {
            // 8.释放资源
            JDBCUtils.release(null,ps,connection);
        }

    }
}
```

6.2 转账案例

- 案例的准备工作

```
create table account(
    id int primary key auto_increment,
    name varchar(20),
    money double
);

insert into account values (null,'zs',1000);
insert into account values (null,'ls',1000);
insert into account values (null,'ww',1000);
```

1.需求

zs给ls转100, 使用事务进行控制

2.分析

- 1.注册驱动,获得连接
- 2.开启事务

- 3.预编译sql语句,得到预编译对象
- 4.设置sql语句参数
- 5.执行sql语句,处理结果
- 6.提交事务或者回滚事务
- 7.释放资源

3.实现

```
public class Test2_事务转账案例 {
    public static void main(String[] args)  {
        Connection connection = null;
        PreparedStatement ps1 = null;
        PreparedStatement ps2 = null;
        try {
            //- 1.注册驱动,获得连接
            connection = JDBCUtils.getConnection();

            //- 2.开启事务
            connection.setAutoCommit(false);

            //- 3.预编译sql语句,得到预编译对象
            String sql1 = "update account set money = money - ? where name = ?";
            String sql2 = "update account set money = money + ? where name = ?";
            ps1 = connection.prepareStatement(sql1);
            ps2 = connection.prepareStatement(sql2);

            //- 4.设置sql语句参数
            ps1.setDouble(1,100);
            ps1.setString(2,"zs");

            ps2.setDouble(1,100);
            ps2.setString(2,"ls");

            //- 5.执行sql语句,处理结果
            int rows1 = ps1.executeUpdate();
            int rows2 = ps2.executeUpdate();
            System.out.println("rows1:"+rows1);
            System.out.println("rows2:"+rows2);

            int i = 1/0;

            if (rows1 > 0  && rows2 > 0){
                //- 6.提交事务
                connection.commit();
            } else{
                //- 6.回滚事务
                try {
                    connection.rollback();
                } catch (SQLException e1) {
                    e1.printStackTrace();
                }
            }

        } catch (Exception e) {
            //- 6.回滚事务
            try {
                connection.rollback();
```

```
            } catch (SQLException e1) {
                e1.printStackTrace();
            }

        } finally {
            //- 7.释放资源
            JDBCUtils.release(null,ps1,connection);
            JDBCUtils.release(null,ps2,null);
        }

    }
}
```

总结

```
必须练习:
    1.JDBC的增删查改操作(CRUD)----->Statement--->目的: 1.熟悉jdbc操作步骤;2.抽取工具类
  4.1
    2.JDBC工具类的抽取-----必须完成     4.2
    3.PreparedStatement完成增删查改操作(CRUD)-----重点   5.3
    4.JDBC事务操作转账案例-----重点    6.2

- 能够使用常见的函数
    if相关的函数:
        if(expr1,expr2,expr3);
        ifnull(expr1,expr2);
    字符串函数:
        concat(str1,str2,...);
        concat_ws(separator,str1,str2,...)
        upper(str);
        lower(str);
        trim(str);
        substr(str,pos),substring(str,pos);
        substr(str,pos,len),substring(str,pos,len);
    时间日期函数:
        current_date();
        current_time();
        now();
    数值函数:
        abs(x);
        ceil(x);
        floor(x);
        pow(x,y);
        rand();

- 能够理解JDBC的概念
    JDBC其实就是一套java代码操作数据库的规范(少量的类,大量的接口)
    所有数据库产商提供的驱动jar包都得遵守JDBC规范(实现接口)

- 能够使用DriverManager类
    DriverManager类: 驱动管理类
            加载驱动: public static void registerDriver(Driver driver)
```

获得连接: public static Connection getConnection(String url, String user, String password)

- 能够使用Connection接口
  Connection接口: 连接对象
    创建执行sql语句对象: Statement createStatement()
    创建预编译sql语句对象: PreparedStatement prepareStatement(String sql)
      参数sql:表示预编译的sql语句,如果sql语句有参数通过?来占位

- 能够使用Statement接口
  Statement接口: 执行sql语句对象
      执行查询语句:ResultSet executeQuery(String sql)
      执行增删改语句:int executeUpdate(String sql)

- 能够使用ResultSet接口
  ResultSet接口: 封装查询语句的结果集对象
      boolean next();
      类型 get类型("列名");

- 能够说出SQL注入原因和解决方案
  原因:因为sql语句是简单的字符串拼接,所以拼接后sql语句的格式可能会改变
  解决方案:使用预编译sql语句对象,对sql语句进行预编译,从而固定sql语句的格式
    setXXX(参数位置,参数的值)

- 能够通过PreparedStatement完成增、删、改、查
  设置sql语句参数的值:
    预编译对象.set类型(int i,类型 值);参数1i:指的就是问号的索引(指第几个问号,从1开始),参数2就是值
  执行sql语句:
    执行查询语句:ResultSet executeQuery()
    执行增删改语句:int executeUpdate()

- 能够完成PreparedStatement改造登录案例
  0.用户输入用户名和密码
  1.注册驱动,获得连接
  2.创建预编译sql语句对象
  3.设置参数
  4.执行sql语句,处理结果
  5.释放资源

- 能够使用JDBC操作事务
  Connection接口:
    connection.setAutoCommit(fasle); //开启事务
    connection.commit();   //提交事务
    connection.rollback();   //回滚事务