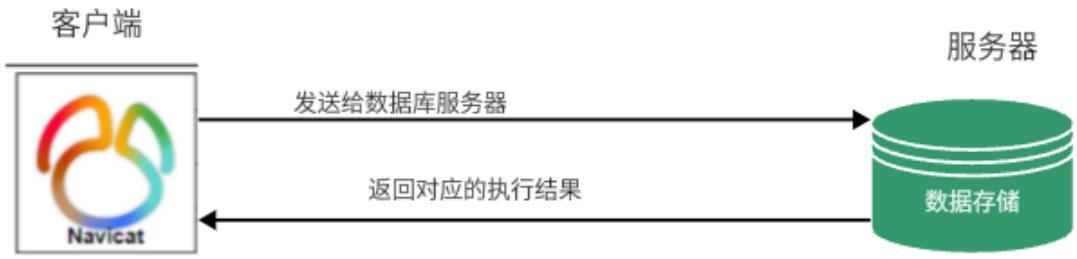
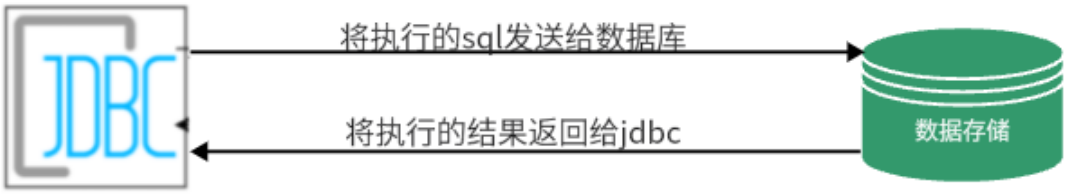
1.JDBC引言

1.1、JDBC简介

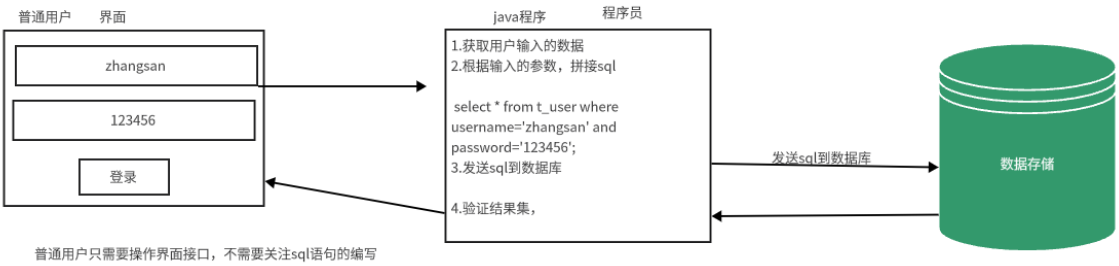
JDBC(java dataBase Connectivity):java数据库链接技术，通过java链接数据库，并对数据库中表的数据进行增删改查等操作。





数据可视化工具比JDBC简单，为什么还要使用JDBC?

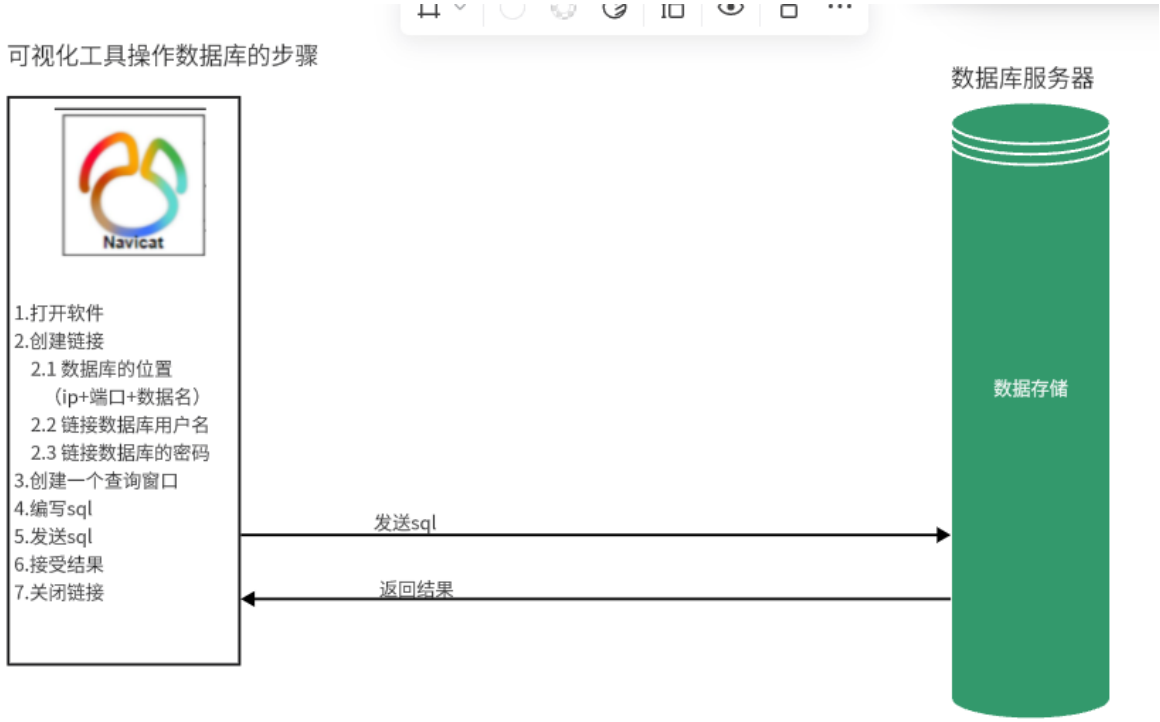
JDBC是一个可以定制化的数据库链接工具，可以降低普通用户操作数据的难度。



此前我们也学习使用navicat，SQLyog等图形化工具操作数据库，JDBC的作用和图形化工具相同的，都是发送sql到数据库，，差别图形化操作简单，而JDBC需要通过编码完成对应的操作

1.2 、操作数据的步骤

可视化工具操作数据库的步骤



JDBC操作数据库的步骤

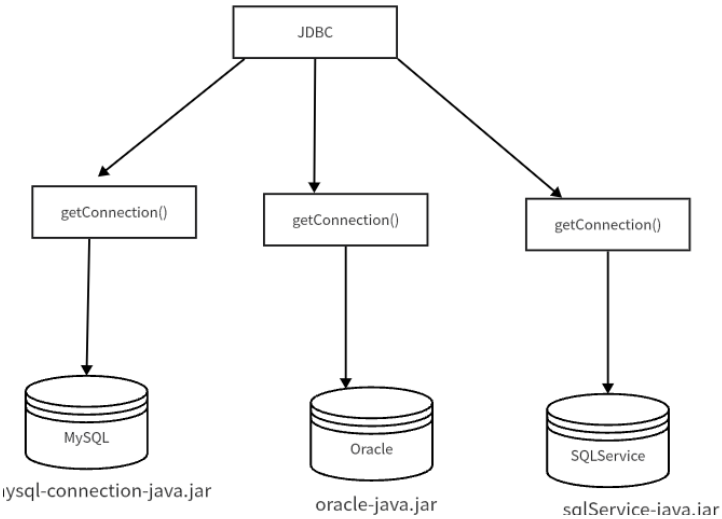
1.3、JDBC 常用API

JDBC要通过Java代码操作数据库，JDBC中定义了相关操作数据的各种接口和类型。

- Driver驱动接口， 定义了java如何和数据库获取链接
- DriverManager工具类，提供了关联驱动的便捷能力，可以获取和数据库的链接
- Connection 链接接口 代表Java和数据库之间的链接
- PreparedStatement 发送sql的工具接口，该类型的对象用于向数据库发送sql语句
- ResultSet 结果集接口，该类型的对象用于接收数据库返回结果集。

1.4、JDBC是标准不是实现

从根本上说，JDBC是一种规范，提供了一组完整的接口，允许便携式的访问底层数据库，因此可以轻松的使用JDBC提供的各种接口。



1.各个数据之间进行代码平移会很困难，
2.每个数据库中底层实现不一样，

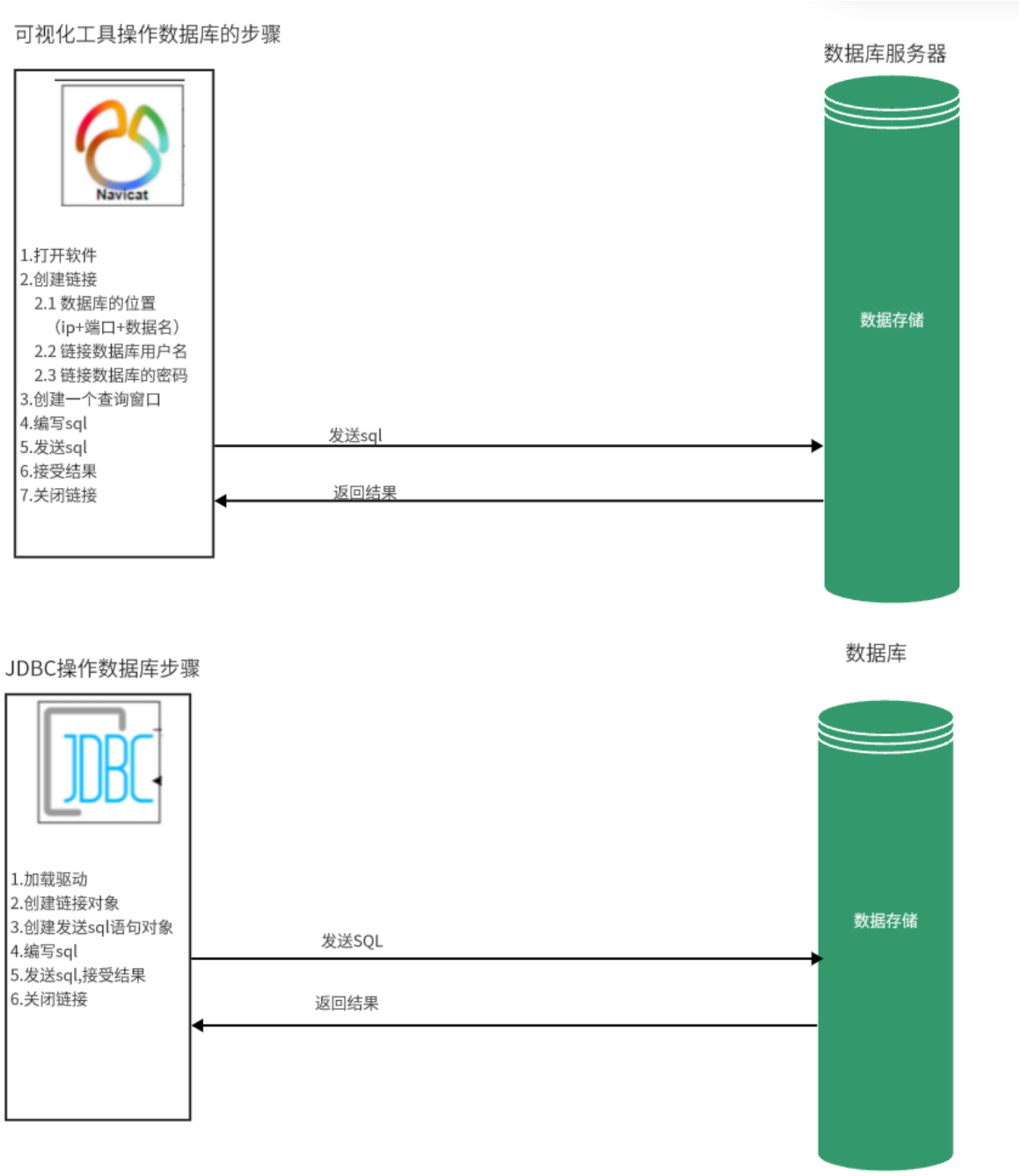
3.java制订标准，不同数据库具体来进行实现，以jar包的方式
mysql-connection-java.jar

155****9091

2.JDBC编程

2.1 、JDBC操作步骤

之前，我们已经反复说过JDBC和可视化工具操作数据库的作用类似，回顾navicat和JDBC的操作步骤



总结：JDBC开发步骤

```
1. 加载驱动
2.创建链接对象
3.准备sql以及创建发送sql的对象
4.发送sql
5.处理结果集
6.关闭链接
```

2.2 、第一个JDBC操作步骤

需求：使用JDBC向 t_person表中插入一条数据

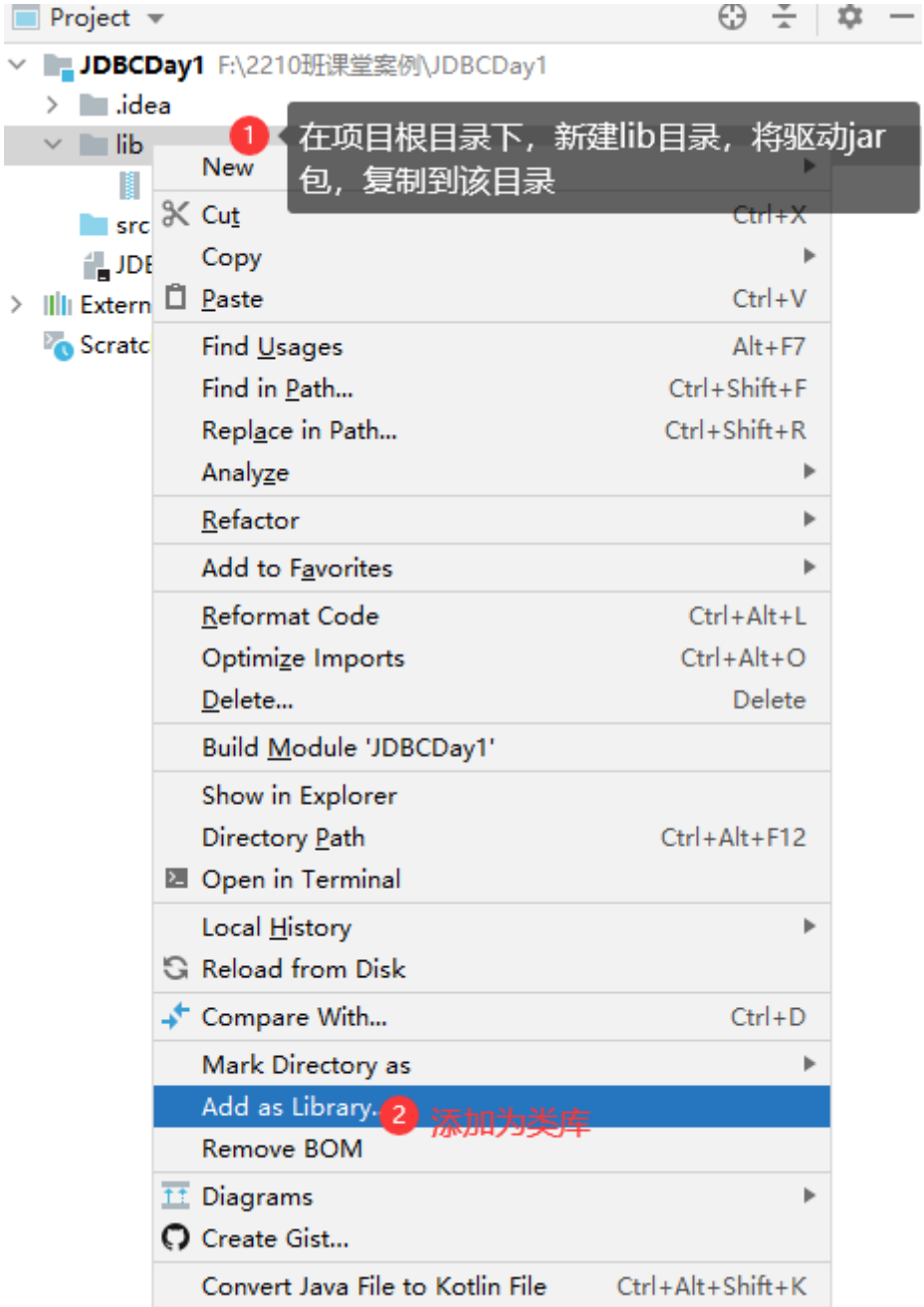
```
create table t_person(
    p_id int(10) PRIMARY key auto_increment,
    p_name VARCHAR(20) not null,
    age int(4) DEFAULT 1,
    gender VARCHAR(10) DEFAULT '男',
    mobile VARCHAR(20) UNIQUE,
    address VARCHAR(50)
)

-- 向t_person表中插入一条数据
insert into t_person values(10010,'zhangsan',21,'男','123456789','河南郑州');
```

1.准备工作（搭建开发环境）

需要在项目中引入数据库的驱动jar包，

```
idea
1.在项目的根目录新建一个lib目录，将jar包复制到该目录
2.选中lib目录，右键 add as library
```



2.编码

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class JDBCTest1 {
    public static void main(String[] args) throws ClassNotFoundException,
SQLException {
//        1.加载驱动
        /**
         * 驱动版本 5.1.XX：com.mysql.jdbc.Driver
         * 驱动版本 8.X ： com.mysql.cj.jdbc.Driver
         */
        Class.forName("com.mysql.jdbc.Driver");
//        2.获取链接对象
        /**
         * url:数据库对应的路径
         *      jdbc:mysql://ip:端口号/数据库名
         *      链接本机数据库：   jdbc:mysql://127.0.0.1:3306/java2210
         *                        jdbc:mysql://localhost:3306/java2210
         *      链接参数：useUnicode=true&characterEncoding=utf-
8&useSSL=false&serverTimeZone=Asia/Shanghai
         *      完整的链接参数
         *
         *      username:数据库的用户名
         *      password：数据库密码
         */
        String url ="jdbc:mysql://localhost:3306/java2210?
useUnicode=true&characterEncoding=utf-
8&useSSL=false&serverTimeZone=Asia/Shanghai";
        String username="root";
        String password="root";
        Connection conn =  DriverManager.getConnection(url,username,password);
//        3.编写sql,并创建执行sql语句对象
        String sql = "insert into t_person(p_name,age,gender,mobile,address)
values('安建龙',21,'男','1568526547','河南焦作')";
//        通过链接对象，创建发送sql语句的对象，并且将sql绑定到该对象
        PreparedStatement pstm = conn.prepareStatement(sql);
//        4.发送sql
        /**
         * executeUpdate():用来进行对数据库的增删改动作。该方法的返回结果为int类型，即受影
响的行数
         */
        int i = pstm.executeUpdate();
//        5.处理结果集
        if(i>0){
            System.out.println("添加成功");
        }else{
            System.out.println("添加失败");
        }
//        6.关闭链接
        pstm.close();
        conn.close();
    }
}
```

练习：完成对t_person表的修改和删除操作

3.JDBC结果集的处理

需求：查询t_person表中的所有数据

```
select * from t_person;
```

开发步骤回顾

```
1.加载驱动
2.创建链接对象
3.编写sql，并获取执行sql对象
4.发送sql
5.处理结果集
6.关闭链接
```

查询功能代码演示

```
import java.sql.*;

public class QueryAllPersonTest {
    public static void main(String[] args) throws ClassNotFoundException,
SQLException {
//        1.加载驱动
        Class.forName("com.mysql.jdbc.Driver");
//        2.创建链接对象
        String url = "jdbc:mysql://localhost:3306/java2210?
useUnicode=true&characterEncoding=utf-
8&useSSL=false&serverTimeZone=Asia/shanghai";
        String username="root";
        String password="root";
        Connection conn = DriverManager.getConnection(url,username,password);
//        3.编写sql,并获取执行sql语句对象
        String sql = "select * from t_person";
        PreparedStatement pstm = conn.prepareStatement(sql);
//        4.发送sql
       ResultSet rs = pstm.executeQuery();        // executeQuery():jdbc执行查询动
作的方法
//        5.处理结果集
//        rs.next() :让指向结果集的指针，执行下一行数据
        while (rs.next()){
//            获取结果集中的数据有两种方式
//            方式一：通过字段名进行获取          结果集对象.get字段类型（”字段名“）
            int pid =  rs.getInt("p_id");
            String pname = rs.getString("p_name");
            int age = rs.getInt("age");
            String gender = rs.getString("gender");
            String mobile = rs.getString("mobile");
            String address = rs.getString("address");
//            方式二：通过字段的下标进行获取，下标从1开始
//            int pid = rs.getInt(1);          //获取结果集中的第一个字段值
//            String pname = rs.getString(2);      //获取结果集中第二个字段值
//            建议：最好使用通过字段名的方式获取对应字段值
```

```
 System.out.println("pid="+pid+",pname="+pname+",age="+age+",gender="+gender+",m
obile="+mobile+",address="+address);
        }
//        6.关闭链接
        rs.close();
        pstm.close();
        conn.close();
    }
}
```

结果集处理整理

1.查询的结果集使用ResultSet对象接收

2.使用while循环进行处理

3.通过结果集对象.next()让执行结果集的指针执行下一行数据

4.获取数据行中数据的方式

1.通过字段名进行获取

语法：结果集对象.get字段类型（"字段名"）；

优点：稳定，当表中字段顺序进行调整时，不影响代码

2.通过字段下标顺序进行获取，下标从1开始

语法：结果集对象.get字段类型（下标）

优点：简单，

缺点：当表中字段顺序进行调整时，jdbc代码也要进行调整

建议：建议使用通过字段名的方式进行获取字段值

##

4.数据绑定

场景：要求将控制台输入的值保存到数据库中。

数据绑定：将用户输入的数据，绑定到要执行的sql中。

> JDBC执行的sql中的数据要根据用户的输入发生变化，比如，登录功能背后的查询sql要根据用户名不同，执行不同的条件，这就需要将输入的数据绑定到执行的sql中

数据绑定的方式：

1.字符串拼接（实际开发中很少用）

2.使用SQL占位符，？

环境准备

```
Create table t_user(
uid int(10) PRIMARY key auto_increment,
user_name VARCHAR(40) UNIQUE not null,
password VARCHAR(50) not null
)

-- 添加数据
insert into t_user VALUES(10010,'admin','123456');
```

4.1 字符串拼

字符串拼接方式，本质上就是通过java字符串拼接语法构造正确的sql语句，sql拼接的步骤如下

```
1.在需要数据的地方，使用变量名进行替换、
2.在变量名前需要添加”+ ，后面需要添加 +" ,完整语法： "insert into 表名 vaules("+变量
1+","+变量2+") "
3.如果拼接的值是字符串，则需要在“+前面添加‘，在+“后面也需要添加’，
举例：
String sql = "insert into t_user values("+id+",'"+username+"','"+password+"')";
```

案例：

```
 public static void addUser(String userName,String password) throws
ClassNotFoundException, SQLException {
        //0加载驱动
        Class.forName("com.mysql.jdbc.Driver");
//        1.获取链接
        String url = "jdbc:mysql://localhost:3306/java2210?
useUnicode=true&characterEncoding=utf-
8&useSSL=false&serverTimeZone=Asia/shanghai";
        String user = "root";
        String pwd = "root";
        Connection conn =  DriverManager.getConnection(url,user,pwd);
//        2.编写sql，并创建执行sql语句对象
        String sql = "insert into t_user
values(null,'"+userName+"','"+password+"')";
        PreparedStatement pstm = conn.prepareStatement(sql);
//        3.发送sql
        int i = pstm.executeUpdate();
//        4.处理结果集
        if(i>0){
            System.out.println("用户添加成功");
        }else{
            System.out.println("用户添加失败");
        }
//        5.关闭链接
        pstm.close();
        conn.close();
    }
```

缺点：1.字符串拼接过于繁琐，容易出错，

2.不安全，不能防止sql注入攻击。

sql注入攻击案例

```
    public static void login(String username,String password) throws
ClassNotFoundException, SQLException {
        //0加载驱动
        Class.forName("com.mysql.jdbc.Driver");
//        1.获取链接
        String url = "jdbc:mysql://localhost:3306/java2210?
useUnicode=true&characterEncoding=utf-
8&useSSL=false&serverTimeZone=Asia/shanghai";
        String user = "root";
        String pwd = "root";
        Connection conn =  DriverManager.getConnection(url,user,pwd);
//        2.编写sql，并创建执行sql语句对象
        String sql = "select * from t_user where user_name='"+username+"' and
password='"+password+"'";
        PreparedStatement pstm = conn.prepareStatement(sql);
//        3.发送sql
       ResultSet rs = pstm.executeQuery();
//        4.处理结果集
       if(rs != null){
           System.out.println("登录成功");
       }else{
           System.out.println("登录失败");
       }

//        5.关闭链接
        pstm.close();
        conn.close();
    }

    public static void main(String[] args) throws ClassNotFoundException,
SQLException {
        Scanner sc = new Scanner(System.in);
        System.out.println("请输入用户名");
        String userName = sc.nextLine();
        System.out.println("请输入密码");
        String password = sc.nextLine();
//        调用添加用户的方法
//        addUser(userName,password);
//        调用登录方法
        login(userName,password);
    }
    场景模拟：正常的输入用户名和密码，会在数据库中进行用户名和密码的检查，如果输入的用户名为 '
or '1=1
    和密码 为 ' or ' 1=1 此时，拼接的sql如下
    select * from t_user where user_name=' ' or '1=1' and password=' ' or '1=1';
    以上sql的逻辑就绕过了用户名和密码的检查，这就是sql注入攻击
```

4.2？占位符

？占位符是JDBC的一种特殊语法，专用于参数数据的绑定

使用步骤：

1.在需要使用数据的地方，使用？代替对应的数据(占位)

2.在发送sql之前，通过pstm.setXXX()方法，给对应的占位符赋值

登录案例：

```
 public static void login1(String userName,String password) throws
ClassNotFoundException, SQLException {
//        1.加载驱动
        Class.forName("com.mysql.jdbc.Driver");
//        2.创建链接
        String url = "jdbc:mysql://localhost:3306/java2210?
useUnicode=true&characterEncoding=utf-
8&useSSL=false&serverTimeZone=Asia/shanghai";
        String user = "root";
        String pwd = "root";
        Connection conn = DriverManager.getConnection(url,user,pwd);
//        3.定义sql和获取执行sql语句对象
//        使用? 占位符，替换具体的参数值
        String sql = "select * from t_user where user_name=? and password=?";
        PreparedStatement pstm = conn.prepareStatement(sql);
       //给占位符赋值
        pstm.setString(1,userName);     //通过pstm对象.set类型（占位符下标【从1开始】，
对应的值）；
        pstm.setString(2,password);
//        4.发送sql
        ResultSet rs = pstm.executeQuery();
//        5.处理结果集
        if(rs.next()){
            System.out.println("登录成功");
        }else{
            System.out.println("登录失败");
        }
//        6.关闭链接
        rs.close();
        pstm.close();
        conn.close();
    }

    public static void main(String[] args) throws ClassNotFoundException,
SQLException {
        Scanner sc = new Scanner(System.in);
        System.out.println("请输入用户名");
        String userName = sc.nextLine();
        System.out.println("请输入密码");
        String password = sc.nextLine();
//        调用添加用户的方法
//        addUser(userName,password);
//        调用登录方法
        login1(userName,password);
    }
```

4.3 字符串拼接和使用占位符的区别

方式	特点	使用场景	最佳实践
字符串拼接	可能会被SQL注入攻击	可以拼接表名列名，sql关键字	动态查询不同的表以及根据用户选中升序或降序查询数据时。
?占位符	可以防止SQL注入攻击	绑定sql中的数据，不能绑定sql中的关键字，比如order by asc|desc	通常情况下，对应数据值的绑定

5.异常

解决异常的步骤

1.定位问题（哪一行代码的问题）

①、根据异常信息定位 （通过异常信息栈找到错误代码的位置）

②、添加打印语句

2.分析问题

根据异常信息和所学的知识来分析问题的原因

3.解决异常

根据已有的异常原因，解决异常。

5.1 根据异常日志的分析

分析异常最高效的办法就是通过阅读异常信息，根据异常日志的信息可以快速定位异常的位置，问题的原因就可能就是定位代码的问题，

举例：

```
异常一：java.sql.SQLException: Access denied for user 'root'@'localhost' (using password: YES)
异常原因：通过用户名和密码进行访问被拒绝，
如何解决：检查用户名和密码是否正确

异常二：com.mysql.jdbc.exceptions.jdbc4.MySQLSyntaxErrorException: Unknown database 'java2211'
异常原因：未知的数据库名
如何解决：检查数据库链接路径中数据名是否正确

异常三：java.sql.SQLException: No value specified for parameter 1
异常原因：没有为参数1赋值，参数绑定问题
如何解决：检查sql中是否给占位符全部赋值


异常四：com.mysql.jdbc.exceptions.jdbc4.MySQLSyntaxErrorException: Unknown column 'username' in 'where clause'
```

```
异常原因：未知的sql字段，sql中的字段和数据库中的字段不匹配
如何解决：检查sql语法是否规范

异常五：java.lang.ClassNotFoundException: com.mysql.cj.jdbc.Driver
原因：驱动没有加载（驱动包没有导入）
如何解决：1.检查驱动包是否导入
         2.检查驱动包的版本是否和数据版本一致
         3.检查数据库驱动名是否正确
```

5.2 添加打印日志

笨办法，需要手动添加打印语句