day17-JDBC

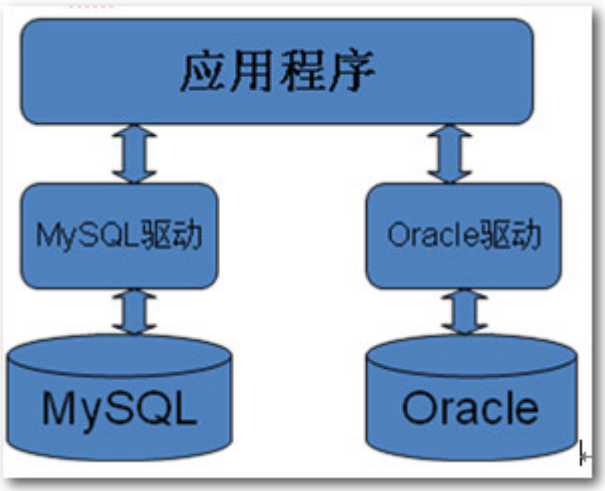
学习目标

1. 能够理解JDBC的概念
2. 能够使用DriverManager类
3. 能够使用Connection接口
4. 能够使用Statement接口
5. 能够使用ResultSet接口
6. 能够说出SQL注入原因和解决方案
7. 能够通过PreparedStatement完成增、删、改、查
8. 能够完成PreparedStatement改造登录案例

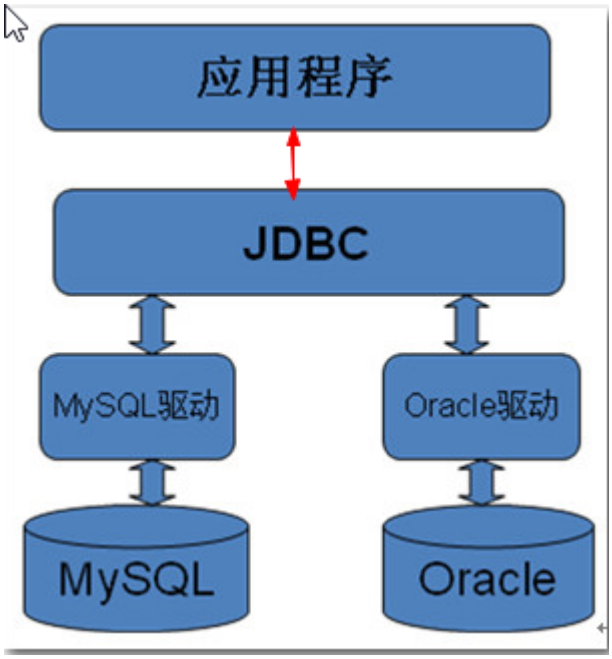
一,JDBC概述

1.为什么要使用JDBC

1. 没有JDBC



2. 有了JDBC后



2.JDBC的概念

2.1什么是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(API,接口).

2.2JDBC和数据库驱动的关系

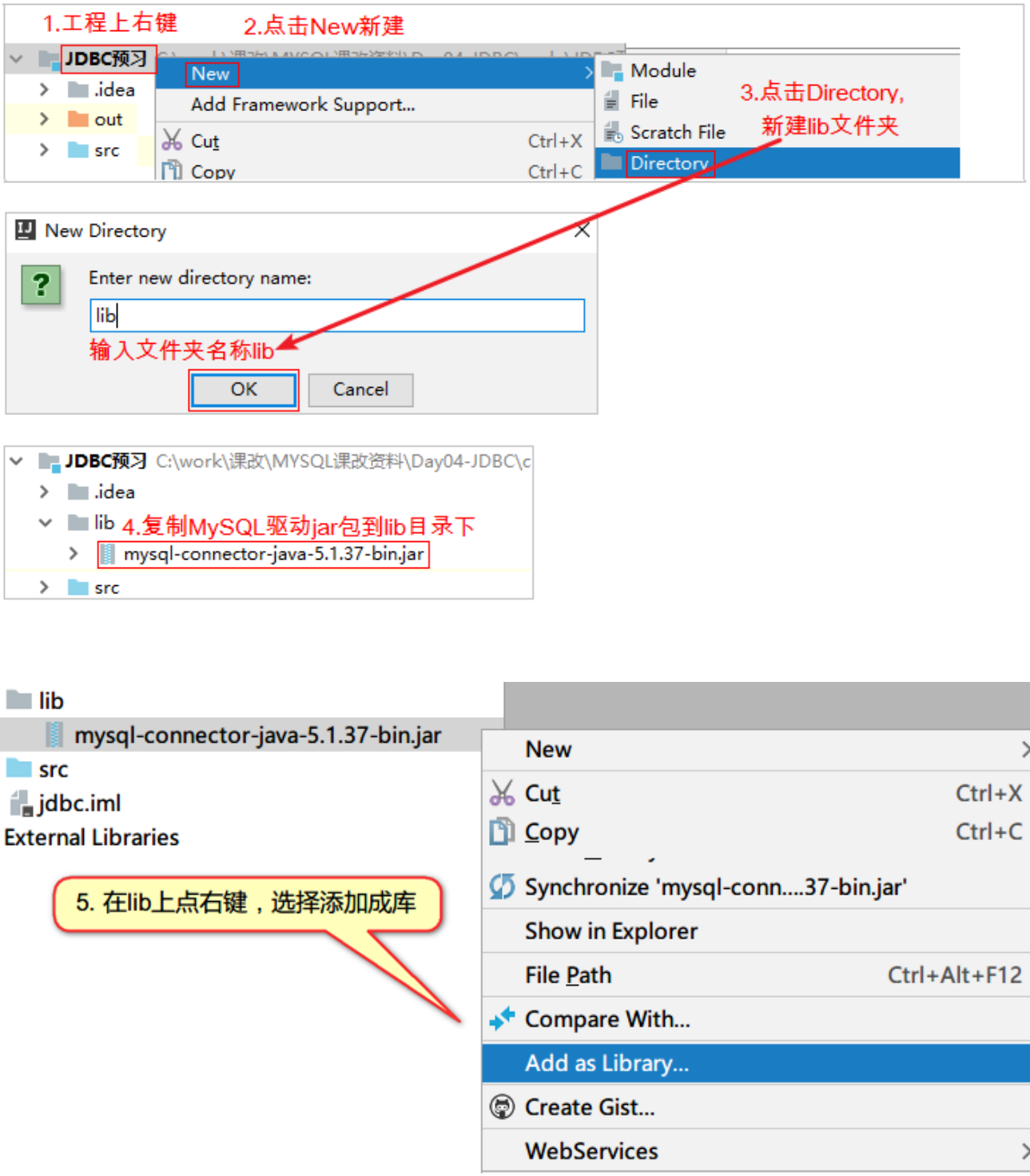
接口(JDBC)与实现(驱动jar包)的关系

二,开发第一个JDBC程序

1.开发第一个JDBC程序

1.1准备工作:

驱动jar包导入项目



1.2开发步骤

- 通过DriverManager的registerDriver()方法new jdbc.Driver()
- 通过DriverManager的getConnection()方法传入地址用户密码获取连接
- 通过Connection的createStatement()方法获取声明Statement对象
- 通过Statement的executeQurry方法得到数据库返回的ResultSet集合遍历集合用next()判断
- 关闭资源

1.3代码实现

```
    import com.mysql.jdbc.Driver;
```

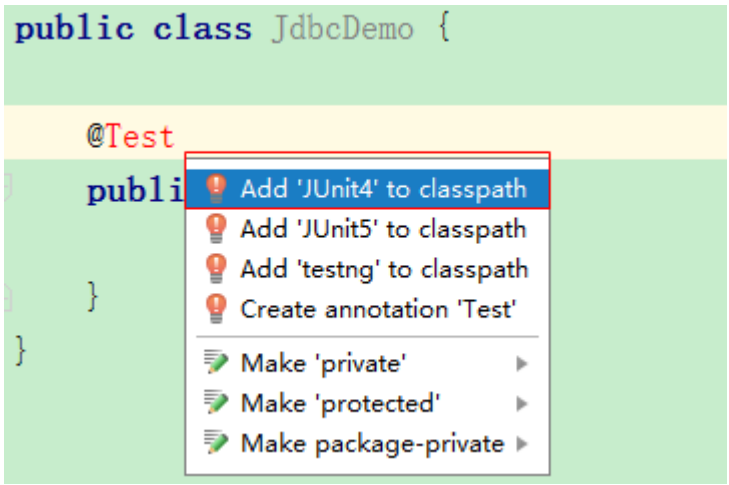
```
    public static void main(String[] args) throws SQLException {
        //注册驱动,尤其注意这里包不是导入的mysql内的Driver包，是jdbc
        //register注册 manager管理
        DriverManager.registerDriver(new Driver());
        String url = "jdbc:mysql://localhost:3306/day10";
        String user = "root";
        String password = "123456";
        //获得连接
        Connection connection = DriverManager.getConnection(url, user, password);
        //创建执行sql语句对象
        //statement声明
        Statement statement = connection.createStatement();
        //执行sql,处理结果  execute 执行
        String sql = "select *from user";
        ResultSet resultSet = statement.executeQuery(sql);
        while (resultSet.next()) {
            System.out.println(resultSet.getObject(1));
            System.out.println(resultSet.getObject(2));
            System.out.println(resultSet.getObject(3));
            System.out.println(resultSet.getObject(4));

        }
        //关闭资源
        if(resultSet != null){
            resultSet.close();

        }
        if(statement  != null){
            statement .close();
        }
        if(connection != null){
            connection.close();
        }
    }
```

2.单元测试介绍和使用

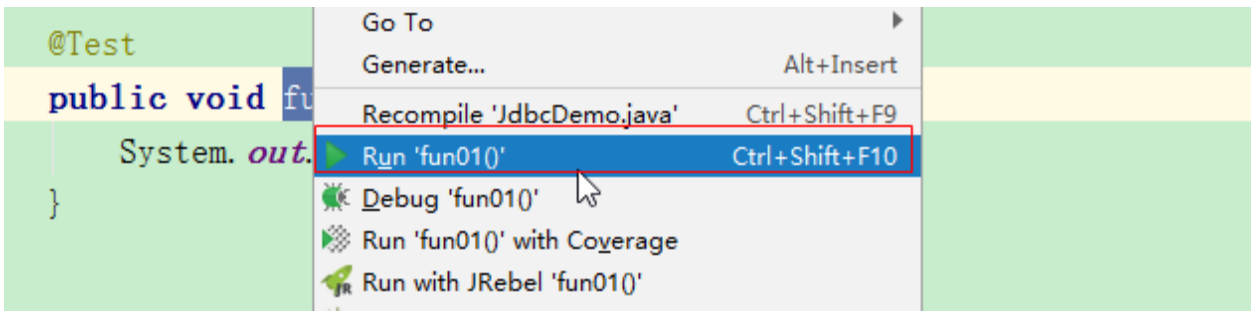
1. 介绍：JUnit是一个Java语言的单元测试框架。属于第三方工具，一般情况下需要导入jar包，不过，多数Java开发环境 已经集成了JUnit作为单元测试工具
2. 编写测试类，简单理解可以用于取代java的main方法
3. 在测试类方法上添加注解@Test
4. 注解修饰的方法要求：public void 方法名() {...} ，方法名自定义建议test开头，没有参数。

```java
public class JdbcDemo {

    @Test
    public void fun01(){
        System.out.println("测试");
    }
}
```

5. 添加IDEA中集成的Junit库，使用快捷键“Alt+Enter”，点击“Add Junit ...”



6. 使用：选中方法右键，执行当前方法或者选中类名右键，执行类中所有方法（方法必须标记@Test）



7. 常见使用错误，如果没有添加“@Test”，使用“Junit Test”进行运行，将抛异常

单元测试需要注意的地方:

```java
@Test
public void demo(){
    System.out.println("哈哈哈哈");
}
```

1.@Test一定要写,并且就是@Test
2.方法一定要是public
3.不要返回值

三,JDBC API详解

1.java.sql.Drivermanager

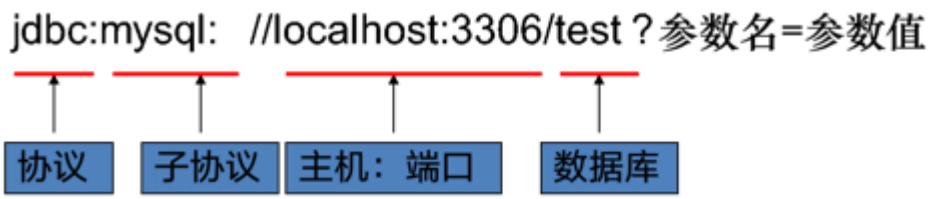
1.registerDriver(Driver driver) ;注册驱动

```
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");
```

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



2.java.sql.Connection接口

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。

1. createStatement() ;创建执行sql语句对象
2. prepareStatement(String sql) ;创建预编译执行sql语句的对象

3.java.sql.Statement接口

接口的实现在数据库驱动中. 用来操作sql语句，并返回相应结果对象

1. Statement; 执行sql语句对象

- 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，表示是否返回的是ResultSet结果集。仅当执行select语句，且有返回结果时返回true, 其它语句都返回false (了解)

4.java.sql.ResultSet接口

1. 封装结果集,查询结果表的对象

- 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
- 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
- void close()关闭ResultSet 对象

四.JDBC操作数据库练习

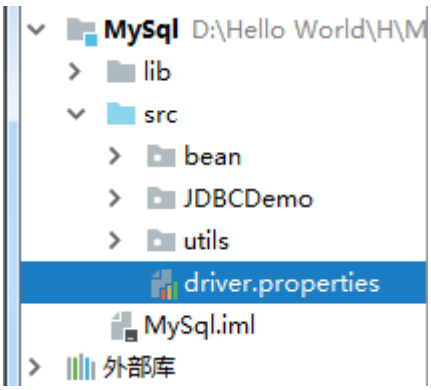
1.增删改查

```
    // 把id为4的用户密码改成666666
    @Test
    public void fun03() throws ClassNotFoundException, SQLException {

        // 1.注册驱动
        Class.forName("com.mysql.jdbc.Driver");
        // 2.创建连接
        String url = "jdbc:mysql://localhost:3306/web10";

        String user = "root";
```

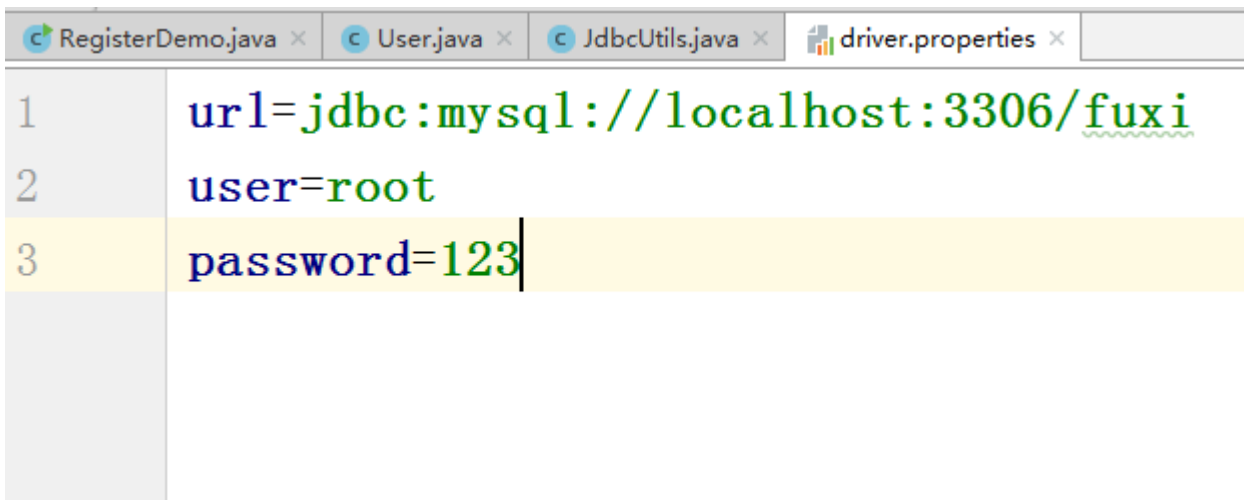
```
        String password = "123456";

        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 = "update user set password = '666666' where id = 4";
        statement.executeUpdate(sql);
        // 5.关闭资源

        if (statement != null) {
            statement.close();
        }

        if (connection != null) {
            connection.close();
        }

    }
```

2.JDBC工具类的抽取

- 创建配置文件,配置文件在src目录下,扩展名是properties



配置文件:



- 工具类

```java
package utils;
import java.io.InputStream;
import java.sql.*;
import java.util.Properties;

public class JdbcUtils {
    private JdbcUtils() {
    }
    private static String url;
    private static String name;
    private static String password;

    static {
        try {
            InputStream is=JdbcUtils.class
                .getClassLoader().getResourceAsStream("driver.properties");
            Properties p=new Properties();
            p.load(is);
            url=p.getProperty("url");
            name=p.getProperty("user");
            password=p.getProperty("password");
            Class.forName("com.mysql.jdbc.Driver");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(url,name,password);
    }
    public static void Jclose(Connection c,Statement s,ResultSet set){
        if(set!=null){
            try {
                set.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        if(s!=null){
            try {
                s.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        if(c!=null){
            try {
                c.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }

    }
```

```
}
```

- 主类

```
package JDBCDemo;
import bean.User;
import utils.JdbcUtils;
import java.sql.*;
import java.util.*;

public class RegisterDemo {
    public static void main(String[] args) throws SQLException {
            Connection c= JdbcUtils.getConnection();
            Statement s=c.createStatement();
            ResultSet set=s.executeQuery("select * from cend;");
            List<User> list=new ArrayList<>();

            while(set.next()){
                User u=new User(set.getString("id"),
                        set.getString("name"),
                        set.getString("style"),
                        set.getString("money")
                );
                list.add(u);
            }
        System.out.println(list);
            JdbcUtils.Jclose(c,s,set);
    }
}
```

- 用户类

```
package bean;

public class User {
    private String id;
    private String name;
    private String style;
    private String money;

    public User(String id, String name, String style, String money) {
        this.id = id;
        this.name = name;
        this.style = style;
        this.money = money;
    }
    public User() {
    }
    @Override
    public String toString() {

        return "User{" +
```

```
                "id='" + id + '\'' +
                ", name='" + name + '\'' +
                ", style='" + style + '\'' +
                ", money='" + money + '\'' +
                '}';
    }
}
```

五,实现一个用户登录功能

代码实现注意单引号

```
package JDBCDemo;
import utils.JdbcUtils;
import java.sql.*;
import java.util.*;

public class RegisterDemo {
    public static void main(String[] args) throws SQLException {
            Connection c= JdbcUtils.getConnection();
        Scanner scanner=new Scanner(System.in);
        while(true) {
            Statement s=c.createStatement();
                String user = scanner.nextLine();
                String ps = scanner.nextLine();
                ResultSet set = s.executeQuery("select * from login where name='" + user + "'
and psw='" + ps + "'");
                String bName=null;
                String bpsw=null;
                while(set.next()){
                   bName=set.getString("name");
                   bpsw=set.getString("psw");
                }
                if (bName!= null&&bpsw!=null) {
                    System.out.println("密码正确");
                } else {
                    System.out.println("密码错误");
                }
                JdbcUtils.Jclose(null,s,set);
            }
    }
}
```

六,SQL注入问题解决：preparedStatement

1.preparedStatement概述

预编译对象， 是Statement对象的子类。

特点：

- 性能要高
- 会把sql语句先编译,格式固定好,
- sql语句中的参数会发生变化，过滤掉用户输入的关键字(or)
- ?默认带有单引号''功能

2.用法

2.1通过connection对象创建

- connection.prepareStatement(String sql) ;创建prepareStatement对象
- sql表示预编译的sql语句,如果sql语句有参数通过?来占位

```
String sql = "SELECT *from `user` where username = ? and password = ?";
```

2.2过setxxx方法来指定参数

- setxxx(int i,Obj obj); i 指的就是问号的索引(指第几个问号,从1开始),xxx是类型(eg:int,String,Long)

2.3实例

```
package JDBCDemo;
import utils.JdbcUtils;
import java.sql.*;
import java.util.*;

public class RegisterDemo {
    public static void main(String[] args) throws SQLException {
            Connection c= JdbcUtils.getConnection();
        Scanner scanner=new Scanner(System.in);
        while(true) {
                String user = scanner.nextLine();
                String ps = scanner.nextLine();
            String sql="select * from login where name= ?  and psw= ? ";
            PreparedStatement s=c.prepareStatement(sql);
            s.setString(1,user);
            s.setString(2,ps);
                ResultSet set = s.executeQuery();
                String bName=null;
                String bpsw=null;
                while(set.next()){
                   bName=set.getString("name");
                   bpsw=set.getString("psw");
                }
                if (bName!= null&&bpsw!=null) {
                    System.out.println("密码正确");
                } else {
                    System.out.println("密码错误");
                }
```

```
                JdbcUtils.Jclose(null,s,set);
            }
    }
}
```

3.使用preparedStatement改写CURD练习

利用Java程序将数据库fuxi中product表内iphone7价格改为2500

```
url=jdbc:mysql://localhost:3306/fuxi
name=root
psw=123
```

```
package utils;

import java.io.InputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Properties;

public class JdbcUtils {
static{
    try {
        Class.forName("com.mysql.jdbc.Driver");
    } catch (Exception e) {
        e.printStackTrace();
    }
}
    public static Connection getcon() throws Exception {
        InputStream is=JdbcUtils.class.getClassLoader().getResourceAsStream("mysql.properties");
        Properties p=new Properties();
        p.load(is);
        String url=p.getProperty("url");
        String name=p.getProperty("name");
        String password=p.getProperty("psw");
        Connection c=DriverManager.getConnection(url,name,password);
        return c;
    }
    public static void cclose(Connection c, PreparedStatement ps){
       if(c!=null){
           try {
               c.close();
           } catch (SQLException e) {
               e.printStackTrace();
           }
       }
       if(ps!=null){
           try {

               ps.close();
```

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

```
package UpdateDemo;

import utils.JdbcUtils;

import java.sql.Connection;
import java.sql.PreparedStatement;

//利用Java程序将数据库fuxi中product表内iphone7价格改为2500
public class UpdateDemo {
    public static void main(String[] args) throws Exception {
        Connection c=JdbcUtils.getcon();
        String sql="update product set price=? where name=?";
        PreparedStatement ps=c.prepareStatement(sql);
        ps.setString(1,"2500");
        ps.setString(2,"iphone7");
        int result=ps.executeUpdate();
        System.out.println(result);
        JdbcUtils.cclose(c,ps);
    }
}
```

七,模块复制

1.复制，粘贴

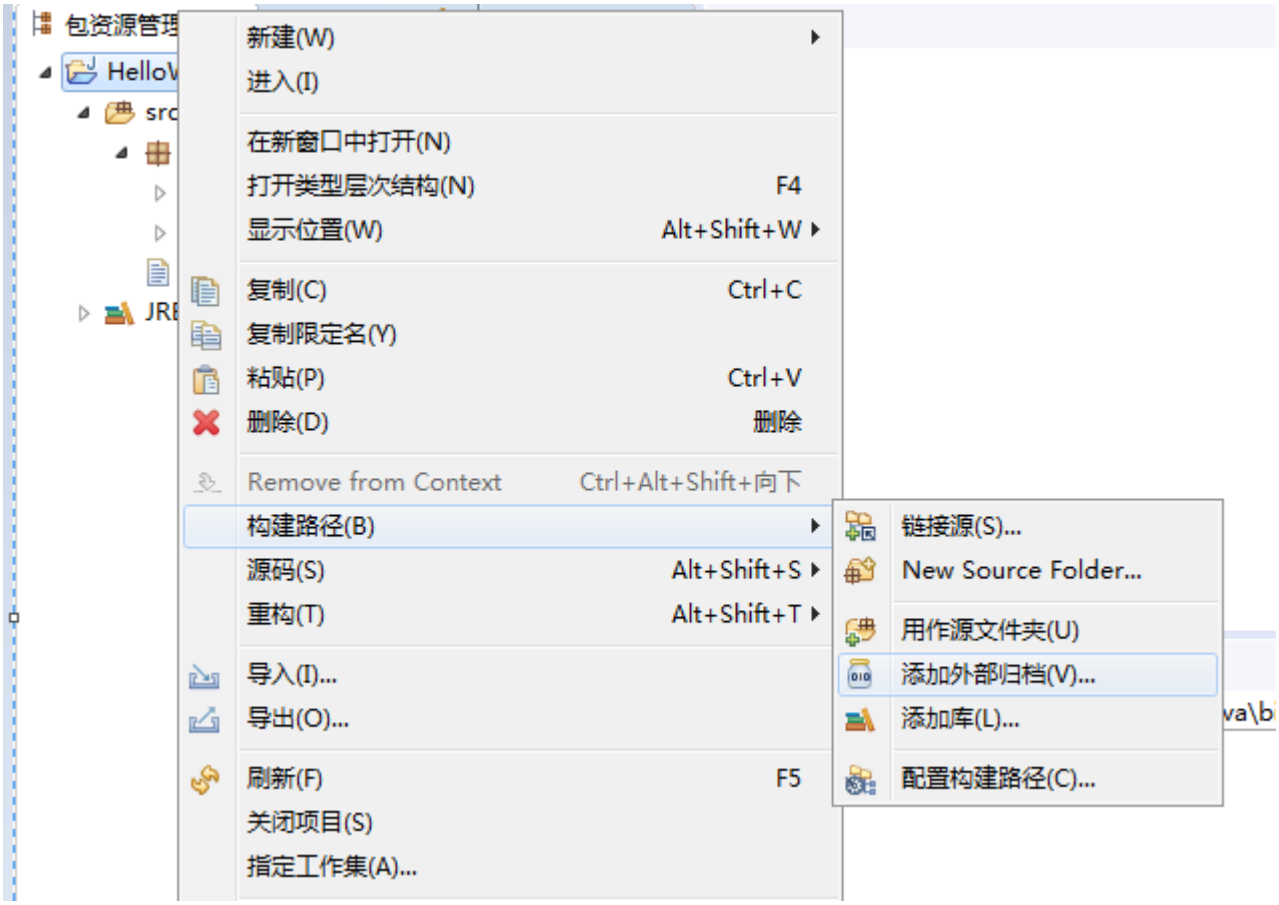
2.重命名模块

3.找到路径重命名.iml文件与模块名称一致

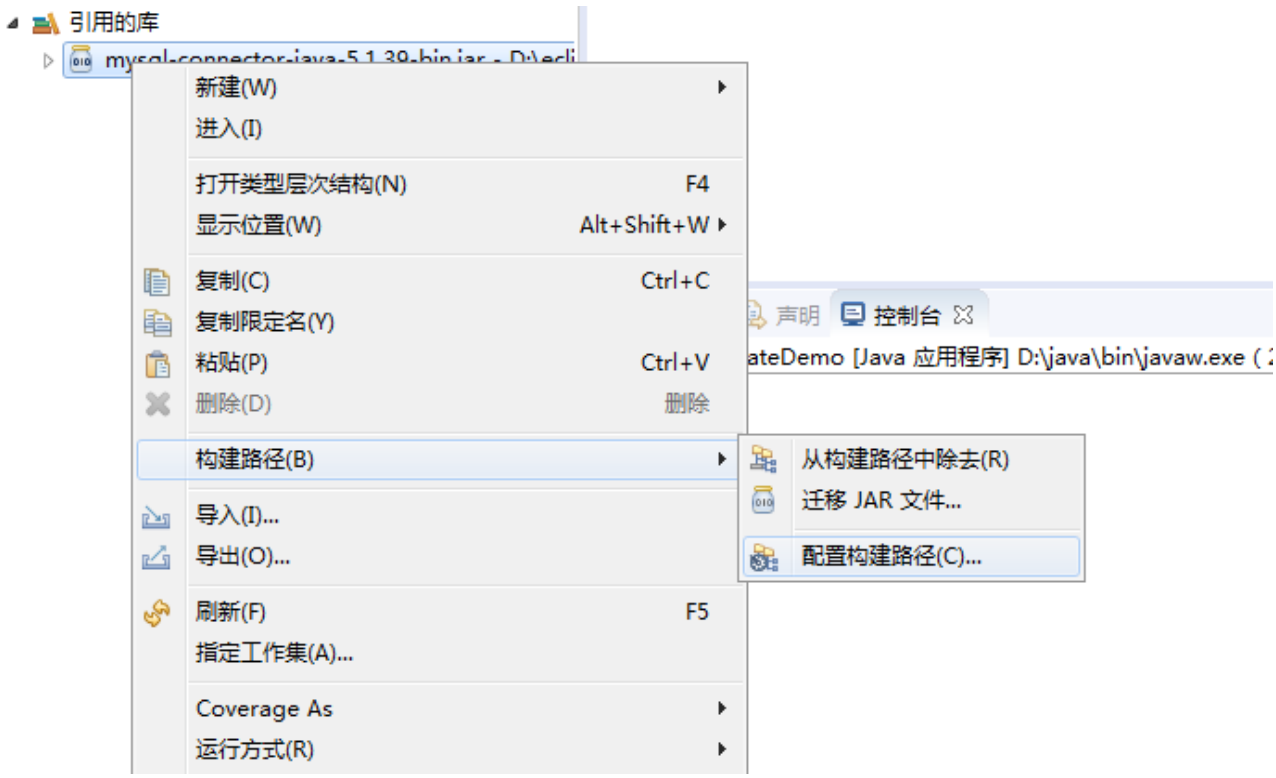
4.进入项目结构Module Import Module

八,Eclipse导包

导入jar包


包资源管理
新建(W)
进入(I)
在新窗口中打开(N)
打开类型层次结构(N) F4
显示位置(W) Alt+Shift+W
复制(C) Ctrl+C
复制限定名(Y)
粘贴(P) Ctrl+V
删除(D) 删除
Remove from Context Ctrl+Alt+Shift+向下
构建路径(B)
源码(S) Alt+Shift+S
重构(T) Alt+Shift+T
导入(I)...
导出(O)...
刷新(F) F5
关闭项目(S)
指定工作集(A)...
链接源(S)...
New Source Folder...
用作源文件夹(U)
添加外部归档(V)...
添加库(L)...
配置构建路径(C)...

导入源码


引用的库
新建(W)
进入(I)
打开类型层次结构(N) F4
显示位置(W) Alt+Shift+W
复制(C) Ctrl+C
复制限定名(Y)
粘贴(P) Ctrl+V
删除(D) 删除
构建路径(B)
导入(I)...
导出(O)...
刷新(F) F5
指定工作集(A)...
Coverage As
运行方式(R)
从构建路径中除去(R)
迁移 JAR 文件...
配置构建路径(C)...
声明
控制台


输入过滤器文本
资源
构建器
项目引用
运行 / 调试设置
重构历史
Coverage
Java 编辑器
Java 编译器
Java 代码样式
Java 构建路径
Javadoc 位置
Task Repository
Task Tags
Validation
WikiText
Java 构建路径
源码(S)
项目(P)
库(L)
排序和导出(O)
构建路径上的 JAR 和类文件夹(T)：
Modulepath
Classpath
mysql-connector-java-5.1.39-bin.jar - D:\eclipse
源代码连接：（无）
Javadoc 位置:（无）
External annotations:（无）
Is not modular
本机库位置:（无）
访问规则：（无限制）
添加 JAR...
添加外部 JAR(X)...
添加变量(V)...
Add Library...
添加类文件夹(C)...
Add External Class Folder...
编辑(E)...
移除(R)
迁移 JAR 文件...
应用(A)
Apply and Close
取消