

课程回顾

1. JDBC提供五个对象实现数据库单表的CRUD
- 2 `DriverManager.getConnection()`: 获取数据库连接
- 3 `Connection.prepareStatement()`: 执行sql语句的预编译执行对象
- 4 `PreparedStatement.executeQuery()`: `ResultSet`查询结果集对象

课程目标

1 JDBC的CRUD封装：执行sql的两个方法

2 反射机制 ===== 理解

3 DBUtils工具类

课程实施

1 案例实现：实现Dept的CRUD

- 1 分析：
 - 2 1. `Scanner`接收用户输入部门信息
 - 3 2. JDBC代码完成用户提交信息的添加
- 4 实现思路：
 - 5 · 获取要添加的数据
 - 6 · 定义String的sql: `insert`语句
 - 7 · `PreparedStatement`指定`insert`语句
 - 8 · 数据添加成功！
 - 9 · 显示添加成功后，数据库所有的dept信息
- 10
- 11 3. `Scanner`提示用户输入要修改或删除的部门编号
- 12 4. 根据部门编号查询对应的部门信息，部门信息如果存在，则可以修改或删除
- 13 实现删除的思路：
 - 14 · 提示用户输入要删除的部门编号 `Scanner 90`
 - 15 · 使用`select`查询用户输入的部门编号是否存在 `select * from dept where deptno=?`
 - 16
 - 17 · 如果编号存在的，就执行删除 `delete from dept where deptno=?`
- 18 实现修改的思路：
 - 19 · 提示用户输入要修改的部门编号 `Scanner 90`
 - 20 · 使用`select`查询用户输入的部门编号是否存在 `select * from dept where deptno=?`
 - 21
 - 22 · 如果编号存在的，就执行修改 `update dept set 列=新的值,列=值 where deptno=?`

1-1 新增代码实现

```
1 package cn.kgc.demo2;  
2  
3 import cn.kgc.util.JDBCUtil;  
4  
5 import java.sql.Connection;
```

```

6  import java.sql.PreparedStatement;
7  import java.sql.ResultSet;
8  import java.sql.SQLException;
9  import java.util.Scanner;
10
11  /**
12   * @Author: lc
13   * @Date: 2022/5/10
14   * @Description: 部门信息的新增
15   * @Version: 1.0
16   */
17  public class InsertDemo {
18      public static void main(String[] args) {
19          /*实现思路:
20           * 获取要添加的数据
21           * 定义String的sql: insert语句
22           * PreparedStatement指定insert语句
23           * 数据添加成功!
24           * 显示添加成功后, 数据库所有的dept信息
25           */
26          Scanner input = new Scanner(System.in);
27          //部门编号: 主键 (唯一、非空) 主键不是自增长
28          System.out.print("部门编号: ");
29          int deptNo=input.nextInt();
30          System.out.print("部门名称: ");
31          String deptName = input.next();
32          System.out.print("所在位置: ");
33          String location = input.next();
34
35          //定义sql, 插入数据库的dept表
36          String sql="insert dept values(?,?,?);";
37          //JDBC五大对象
38          Connection conn = JDBCUtil.getConnection();
39          PreparedStatement pstmt = null;//1.防范sql注入 2.预编译对象
40          try {
41              pstmt = conn.prepareStatement(sql);
42
43              //将?与实际接收用户输入的数据做对应
44              //setXXX(?的序号,?对应的实参变量名)
45              pstmt.setInt(1,deptNo);
46              pstmt.setString(2,deptName);
47              pstmt.setString(3,location);
48
49              //执行
50              int row = pstmt.executeUpdate();//程序就会报错!!!
51              System.out.println(row==1?"执行成功":"执行失败");
52          } catch (SQLException throwables) {
53              throwables.printStackTrace();
54          } finally {
55              //释放资源
56              JDBCUtil.release(conn,pstmt);
57          }
58      }
59  }
60

```

1-2 查询所有代码实现

```
1 package cn.kgc.demo2;
2
3 import cn.kgc.util.JDBCUtil;
4
5 import java.sql.Connection;
6 import java.sql.PreparedStatement;
7 import java.sql.ResultSet;
8 import java.sql.SQLException;
9 import java.util.Scanner;
10
11 /**
12  * @Author: lc
13  * @Date: 2022/5/10
14  * @Description: 部门信息的新增
15  * @Version: 1.0
16  */
17 public class InsertDemo {
18     public static void main(String[] args) {
19         //JDBC五大对象
20         Connection conn = JDBCUtil.getConnection();
21         PreparedStatement pstmt2=null;
22         ResultSet rs=null;
23         try {
24             //显示所有的dept信息
25             String querySql="select deptno,dname,loc from dept;";
26             pstmt2= conn.prepareStatement(querySql);
27             rs= pstmt2.executeQuery();
28             System.out.println("编号\t\t\t部门名称\t\t\t位置");
29             while(rs.next()){
30
31                 System.out.println(rs.getInt(1)+"\t\t\t"+rs.getString(2)+"\t\t\t"+rs.getString("loc"));
32             }
33         } catch (SQLException throwables) {
34             throwables.printStackTrace();
35         } finally {
36             //释放资源
37             JDBCUtil.release(conn,pstmt2,rs);
38         }
39     }
40 }
```

1-3 删除代码实现

```
1 package cn.kgc.demo2;
2
3 import cn.kgc.util.JDBCUtil;
4
5 import java.sql.Connection;
6 import java.sql.PreparedStatement;
7 import java.sql.ResultSet;
8 import java.util.Scanner;
9
```

```

10  /**
11   * @Author: lc
12   * @Date: 2022/5/10
13   * @Description: cn.kgc.demo2
14   * @Version: 1.0
15   */
16  public class DeleteDemo {
17      public static void main(String[] args) throws Exception{
18          //1.获取部门编号
19          Scanner input = new Scanner(System.in);
20          System.out.print("要删除的部门编号: ");
21          int deptno = input.nextInt();
22          //2.校验部门编号是否存在
23          String sql="select deptno,dname,loc from dept where deptno=?";
24          //3.执行sql, 获取结果
25          Connection conn = JDBCUtil.getConnection();
26          PreparedStatement pstmt = conn.prepareStatement(sql);
27          pstmt.setInt(1,deptno);
28          ResultSet rs = pstmt.executeQuery();
29          if(!rs.next()){
30              //用户输入的编号不存, 不删除
31              System.out.println("您要删除的部门不存在!");
32              return;
33          }
34          //执行删除
35          String sql2="delete from dept where deptno=?";
36          pstmt=conn.prepareStatement(sql2);
37          pstmt.setInt(1,deptno);
38          int row = pstmt.executeUpdate();
39          System.out.println(row==1?"删除成功":"删除失败");
40
41          JDBCUtil.release(conn,pstmt,rs);
42      }
43  }

```

1-4 修改代码实现

所有行 行的范围 自行: 0 100 行			
	deptno	dname	loc
<input type="checkbox"/>	10	教研部	北京
<input type="checkbox"/>	20	学工部	上海
<input type="checkbox"/>	30	销售部	广州
<input type="checkbox"/>	40	财务部	武汉
<input type="checkbox"/>	50	商务部	上海
<input type="checkbox"/>	55	市场部	北京
<input type="checkbox"/>	60	销售部	重庆
*	(NULL)	(NULL)	(NULL)

java程序修改:
通常都是根据主键修改非主键列的值。主键列的值不建议修改!!!

```

1  package cn.kgc.demo2;
2
3  import cn.kgc.util.JDBCUtil;
4
5  import java.sql.Connection;
6  import java.sql.PreparedStatement;

```

```

7  import java.sql.ResultSet;
8  import java.util.Scanner;
9
10 /**
11  * @Author: lc
12  * @Date: 2022/5/10
13  * @Description: cn.kgc.demo2
14  * @Version: 1.0
15  */
16 public class UpdateDemo {
17     public static void main(String[] args) throws Exception{
18         //1.输入要修改的部门编号
19         Scanner input = new Scanner(System.in);
20         System.out.print("要修改的部门编号: ");
21         int deptno = input.nextInt();
22
23         //2.基于存在才修改
24         //2.校验部门编号是存在
25         String sql="select deptno,dname,loc from dept where deptno=?";
26         //3.执行sql, 获取结果
27         Connection conn = JDBCUtil.getConnection();
28         PreparedStatement pstmt = conn.prepareStatement(sql);
29         pstmt.setInt(1,deptno);
30         ResultSet rs = pstmt.executeQuery();
31         if(!rs.next()){
32             //用户输入的编号不存, 不删除
33             System.out.println("您要删除的部门不存在!");
34             return;
35         }
36         //可以进行修改
37         String sql2="update dept set dname=?,loc=? where deptno=?";
38         System.out.print("修改后的部门名称: ");
39         String newDname = input.next();
40
41         System.out.print("修改后的部门所在位置: ");
42         String newLoc = input.next();
43
44         pstmt=conn.prepareStatement(sql2);
45         pstmt.setString(1,newDname);
46         pstmt.setString(2,newLoc);
47         pstmt.setInt(3,deptno);
48         int row = pstmt.executeUpdate();
49         System.out.println(row==1?"修改成功":"修改失败");
50
51         JDBCUtil.release(conn,pstmt,rs);
52     }
53 }

```

2 Detp进行JDBC执行代码的封装

2-1 DML操作封装成一个方法

```

1  package cn.kgc.demo2;
2
3  import cn.kgc.util.JDBCUtil;
4

```

```

5  import java.sql.Connection;
6  import java.sql.PreparedStatement;
7  import java.sql.SQLException;
8
9  /**
10   * @Author: lc
11   * @Date: 2022/5/10
12   * @Description: cn.kgc.demo2
13   * @Version: 1.0
14   */
15  public class ExecuteHandler {
16      /**
17       * 处理insert、update、delete的sql
18       * @param sql 要执行的DML语句
19       * @param params sql语句?对应的参数列表
20       * @return DML操作受影响的行数
21       */
22      public int executedML(String sql, Object... params) {
23          Connection conn = JDBCUtil.getConnection();
24          PreparedStatement pstmt = null;
25          try {
26              pstmt = conn.prepareStatement(sql);
27              //有几个实参，就要做几次? 的参数映射
28              for(int i=0; i<params.length; i++){
29                  pstmt.setObject(i+1, params[i]);
30              }
31              return pstmt.executeUpdate();
32          } catch (SQLException e) {
33              throw new RuntimeException(e);
34          } finally {
35              JDBCUtil.release(conn, pstmt);
36          }
37      }
38  }

```

2-2 DQL操作封装成一个方法

```

Exception in thread "main" java.sql.SQLException: Create breakpoint : Operation not allowed after ResultSet closed
    at com.mysql.jdbc.SQLException.createSQLException(SQLException.java:1075)
    at com.mysql.jdbc.SQLException.createSQLException(SQLException.java:989)
    at com.mysql.jdbc.SQLException.createSQLException(SQLException.java:984)
    at com.mysql.jdbc.SQLException.createSQLException(SQLException.java:929)
    at com.mysql.jdbc.ResultSetImpl.checkClosed(ResultSetImpl.java:794)
    at com.mysql.jdbc.ResultSetImpl.next(ResultSetImpl.java:7145)
    at cn.kgc.demo2.SelectDeptDemo.main(SelectDeptDemo.java:21)
Process finished with exit code 1

```

```

1  /**
2   * 处理SELECT的sql
3   * @param sql 要执行的DQL语句
4   * @param params sql语句?对应的参数列表
5   * @return
6   */
7  //sql:查询dept表, resultSet保存所有的部门信息 List<部门信息>
8  //sql:查询emp表, resultSet保存所有的员工信息 List<员工信息>
9  public List<T> executeDQLX(String sql, Class<T> clazz, Object... params) {
10      Connection conn = JDBCUtil.getConnection();
11      PreparedStatement pstmt = null;

```

```

12         try {
13             pstmt = conn.prepareStatement(sql);
14             //有几个实参，就要做几次？的参数映射
15             for(int i=0;i<params.length;i++){
16                 pstmt.setObject(i+1,params[i]);
17             }
18             ResultSet rs= pstmt.executeQuery();
19             //遍历rs
20             //rs==>List<>
21             List<T> list=new ArrayList<>();
22             while(rs.next()){
23                 //new对象
24                 T obj = clazz.newInstance();//底层基于无参构造方法
25                 //获取所有定义的属性对象 其实字段，设计与表列名同名、同类型
26                 Field[] fields = clazz.getDeclaredFields();
27                 for(Field f:fields){
28                     String colName = f.getName();//f.getName():获取属性对应的变
量名，变量名与表列同名、同类型
29                     Object value = rs.getObject(colName);
30                     if(!f.isAccessible()){
31                         f.setAccessible(true);
32                     }
33                     f.set(obj,value);//赋值
34                 }
35                 //obj对象设置属性值
36                 list.add(obj);
37             }
38             return list;
39         } catch (Exception e) {
40             throw new RuntimeException(e);
41         } finally {
42             //ResultSet必须基于连接才能数据遍历
43             JDBCUtil.release(conn,pstmt,rs);
44         }
45     }

```

3 反射

3-1 概念

java一种编程机制：可以通过java代码动态获取一个类中每个组成部分对应的对象类型。

3-2 作用

分析一个类的组成部分。很多高级框架的底层实现。理解了反射，就更容易理解很多框架底层实现思想

3-3 使用

获取一个class文件对应的字节码对象***

java将字节码文件对应的对象抽取成一个类，类名：Class

- 1 1. 对象.getClass():Class对象
- 2
- 3 2. 类名.class:Class对象
- 4
- 5 3. Class.forName():Class对象 推荐使用方式
- 6 优点: 对于未来产生的类做预见性的对象获取

课堂案例

```

1 package cn.kgc.demo3;
2
3 /**
4  * @Author: lc
5  * @Date: 2022/5/10
6  * @Description: cn.kgc.demo3
7  * @Version: 1.0
8  */
9 public class TestPerson {
10     public static void main(String[] args) throws Exception{
11         //获取Person这个字节码对象
12         //Class clazz = Person.class;
13         //Class.forName("类完整限定名: 包名.类名");
14         Class clazz = Class.forName("cn.kgc.demo3.Person");
15         //获取Class的方式三
16         /*Person p = new Person();
17         Class clazz = p.getClass();*/
18         //1.创建对象
19         Object obj=clazz.newInstance();//newInstance创建对象，底层基于无参构造方
20         法
21         System.out.println(obj);
22     }
23 }
24 //输出结果
25 cn.kgc.demo3.Person@45ee12a7

```

获取构造方法对象

获取constructor对象的方法

clazz.con

通过形参获取指定public修饰的构造方法

System.out.println(clazz.getConstructor(Class<?>... parameterTypes) Constructor<?>[])

//2. 获取所有public修饰的构造方法

//2. 通过形参列表获取声明的构造方法

File f = new File("...");

getDeclaredConstructors(Class<?>... parameterTypes) Constructor<?>[]

//对 所有声明的构造方法

//如 getEnclosingConstructor() Constructor<?>

Press Enter to insert, Tab to replace Next Tip

可以暴力破

- 1 构造方法: Constructor
- 2 方法: newInstance()

课堂案例

```

1 package cn.kgc.demo3;
2
3 import java.lang.reflect.Constructor;
4 import java.lang.reflect.Field;
5 import java.lang.reflect.Method;
6
7 /**
8  * @Author: lc
9  * @Date: 2022/5/10
10  * @Description: cn.kgc.demo3
11  * @Version: 1.0
12  */
13 public class TestPerson {
14     public static void main(String[] args) throws Exception{
15         //获取Class的方式三: Class.forName("类完整限定名: 包名.类名");
16         Class clazz = Class.forName("cn.kgc.demo3.Person");
17         //1.创建对象
18         //Object obj=clazz.newInstance();//newInstance创建对象, 底层基于无参构造
19         方法
20         //有参构造方法创建对象
21         Constructor constructor = clazz.getConstructor(String.class,
22             int.class);//构造方法可以重载, 指定构造方法的形参列表
23         Object obj=constructor.newInstance("李四",54);
24         System.out.println(obj);//obj.toString()
25     }
26 }

```

获取属性对象

常见异常

```
//1. 创建对象
Object obj = new Object();
System.out.println(obj.getClass().getFields());
//2. 根据属性名获取对应的public属性
System.out.println(obj.getClass().getField("name"));
//2. 根据属性名获取声明属性
System.out.println(obj.getClass().getDeclaredField("name"));
//2. 获取所有的声明的属性
System.out.println(obj.getClass().getDeclaredFields());
```

Ctrl+向下箭头 and Ctrl+向上箭头 will move caret down and up in the editor [Next Tip](#)

clazz.getField()

```
1 属性对象: Field
2 方法: set(对象, 值)  get(对象)
```

```
cn.kgc.demo3.Person@45ee12a7
Exception in thread "main" java.lang.IllegalAccessException Create breakpoint : Class cn.kgc.demo3
.TestPerson can not access a member of class cn.kgc.demo3.Person with modifiers "private"
<1 internal line>
    at java.lang.reflect.AccessibleObject.slowCheckMemberAccess(AccessibleObject.java:296)
    at java.lang.reflect.AccessibleObject.checkAccess(AccessibleObject.java:288)
    at java.lang.reflect.Field.set(Field.java:761)
    at cn.kgc.demo3.TestPerson.main(TestPerson.java:28)
```

name是private修饰的属性，java中对象私有化的属性？？？

课堂案例

```
1 package cn.kgc.demo3;
2
3 import java.lang.reflect.Field;
4
5 /**
6  * @Author: lc
7  * @Date: 2022/5/10
8  * @Description: cn.kgc.demo3
9  * @Version: 1.0
10 */
11 public class TestPerson {
12     public static void main(String[] args) throws Exception{
13         //获取Class的方式一：获取Person这个字节码对象
14         //Class clazz = Person.class;
15
16         //获取Class的方式二
17         /*Person p = new Person();
18         Class clazz = p.getClass();*/
19
20         //获取Class的方式三：Class.forName("类完整限定名：包名.类名");
21         Class clazz = Class.forName("cn.kgc.demo3.Person");
22         //1.创建对象
23         Object obj=clazz.newInstance();//newInstance创建对象，底层基于无参构造方
法
24         System.out.println(obj);
25         //2.对象属性赋值
26         //2-1 Class对象指定的属性对象
27         Field objName = clazz.getDeclaredField("name");
28         //对属性值进行设置或获取
29         //如果属性是非public修饰的，但是又希望对属性进行操作，可以暴力破解
30         if(!objName.isAccessible()) { //isAccessible()name属性是否可访问，
true: 可访问 false: 不可访问
31             //破解
32             objName.setAccessible(true);//设置name可访问
33         }
34         //2-1 给name赋值
35         //objName.set(要赋值的对象,属性值)
36         objName.set(obj,"张三");//传统代码：对象.name=值;
37         //2-2 name的值显示
38         //objName.get(要获取属性值的对象)
39         System.out.println("name="+objName.get(obj));//传统代码：对象.name
40         //3.对象调用方法
41     }
42 }
```

获取方法对象

获取方法的常用方式

```
Field[] fields = clazz.getDeclaredFields();
/*for (Field field : fields) {
    n   getMethod(String name, Class<?>... Method
    n   getMethods() public修饰的方法 Method[]
    n   getDeclaredMethod(String name, Cla... Method
    n   getDeclaredMethods() 所有的声明的方法 Method[]
    m   getEnclosingMethod() Method
}*/
//3.
//3. Ctrl+向下箭头 and Ctrl+向上箭头 will move caret down and up in the editor Next Tip
```

```
clazz.getme
}
```

- 1 方法对象: Method
- 2 方法: `invoke`(对象, 实参列表) 执行静态方法: `invoke(null, 实参列表)`

课堂案例

```
1 package cn.kgc.demo3;
2
3 import java.lang.reflect.Field;
4 import java.lang.reflect.Method;
5
6 /**
7  * @Author: lc
8  * @Date: 2022/5/10
9  * @Description: cn.kgc.demo3
10  * @Version: 1.0
11  */
12 public class TestPerson {
13     public static void main(String[] args) throws Exception{
14         //获取Class的方式一: 获取Person这个字节码对象
15         //获取Class的方式三: Class.forName("类完整限定名: 包名.类名");
16         Class clazz = Class.forName("cn.kgc.demo3.Person");
17         //1.创建对象
18         Object obj=clazz.newInstance();//newInstance创建对象, 底层基于无参构造方
19         法
20         System.out.println(obj);
21         //3.对象调用方法
22         //3-1 获取方法对象
23         //Method[] methods = clazz.getMethods();
24         //getMethod(方法名,形参的class类型)
25         Method objSayMethod = clazz.getMethod("say", int.class);
26         //调用方法
27         //传统方法, 调用静态方法: 返回值=类名.方法名
28         Object returnValue=objSayMethod.invoke(null,3);//invoke(指定对象, 实
29         参)执行当前方法对象
30         System.out.println(returnValue);
31     }
32 }
```

4 基于封装后ExecuteHandler类实现JDBC代码的步骤

将表转换为对应的java类

特点：类名与表名相同 类中的属性与表中的列名相同、属性的数据类型与列的数据类型相同

```
1 package cn.kgc.demo2;
2
3 /**
4  * @Author: lc
5  * @Date: 2022/5/10
6  * @Description: 将数据库查询的表转换为java对象的过渡结果
7  * 该类中只能定义与表列同名、同类型的属性名
8  * @Version: 1.0
9  */
10 public class Dept {
11     private Integer deptNo;
12     private String dname;
13     private String loc;
14
15     public Integer getDeptNo() {
16         return deptNo;
17     }
18
19     public void setDeptNo(Integer deptNo) {
20         this.deptNo = deptNo;
21     }
22
23     public String getDname() {
24         return dname;
25     }
26
27     public void setDname(String dname) {
28         this.dname = dname;
29     }
30
31     public String getLoc() {
32         return loc;
33     }
34
35     public void setLoc(String loc) {
36         this.loc = loc;
37     }
38
39     @Override
40     public String toString() {
41         final StringBuilder sb = new StringBuilder("Dept{");
42         sb.append("deptNo=").append(deptNo);
43         sb.append(", dname=").append(dname).append('\n');
44         sb.append(", loc=").append(loc).append('\n');
45         sb.append('}');
46         return sb.toString();
47     }
48 }
```

定义SQL执行即可

```
1 package cn.kgc.demo2;
2
3 import java.util.List;
4
5 /**
6  * @Author: lc
7  * @Date: 2022/5/10
8  * @Description: 查询dept表并显示结果
9  * @Version: 1.0
10 */
11 public class SelectDeptDemo {
12     public static void main(String[] args) throws Exception {
13         //1.定义sql
14         String sql="select * from dept";
15         //2.执行sql
16         ExecuteHandler<Dept> handler=new ExecuteHandler<Dept>();
17         List<Dept> depts = handler.executeDQLX(sql,Dept.class);
18         //循环集合
19         System.out.println(depts);
20     }
21 }
```

案例2：演示user表的查询

1 将User表转换为User类：属性与列同名、同类型

```
1 package cn.kgc.demo2;
2
3 /**
4  * @Author: lc
5  * @Date: 2022/5/10
6  * @Description: cn.kgc.demo2
7  * @Version: 1.0
8  */
9 public class User {
10     private String uid;
11     private String username;
12     private String password;
13
14     public String getUid() {
15         return uid;
16     }
17
18     public void setUid(String uid) {
19         this.uid = uid;
20     }
21
22     public String getUsername() {
23         return username;
24     }
25
26     public void setUsername(String username) {
27         this.username = username;
28     }
29 }
```

```

29
30     public String getPassword() {
31         return password;
32     }
33
34     public void setPassword(String password) {
35         this.password = password;
36     }
37
38     @Override
39     public String toString() {
40         final StringBuilder sb = new StringBuilder("User{");
41         sb.append("uid=").append(uid).append('\n');
42         sb.append(", username=").append(username).append('\n');
43         sb.append(", password=").append(password).append('\n');
44         sb.append('}');
45         return sb.toString();
46     }
47 }

```

2 定义sql执行并输出结果

```

1  package cn.kgc.demo2;
2
3  import java.util.List;
4
5  /**
6   * @Author: lc
7   * @Date: 2022/5/10
8   * @Description: cn.kgc.demo2
9   * @Version: 1.0
10  */
11  public class SelectDeptDemo {
12      public static void main(String[] args) throws Exception {
13          //查询emp表
14          String sql="select * from user";
15          ExecuteHandler<User> handler = new ExecuteHandler<>();
16          List<User> users = handler.executedQL(sql, User.class);
17          System.out.println(users);
18      }
19  }

```

常见异常

```

Exception in thread "main" java.lang.InstantiationException Create breakpoint : cn.kgc.demo3.Person
    at java.lang.Class.newInstance(Class.java:427)
    at cn.kgc.demo3.TestPerson.main(TestPerson.java:19)
Caused by: java.lang.NoSuchMethodException Create breakpoint : cn.kgc.demo3.Person.<init>()
    at java.lang.Class.getConstructor0(Class.java:3082)
    at java.lang.Class.newInstance(Class.java:412)
    ... 1 more
Person中没有无参构造方法

Process finished with exit code 1

```

```
java.sql.SQLException: Parameter index out of range (1 > number of parameters, which is 0).
    at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQLException.java:129) 原因: ?与实参的个数不一致
    at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQLException.java:97)
    at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQLException.java:89)
    at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQLException.java:63)
    at com.mysql.cj.jdbc.ClientPreparedStatement.checkBounds(ClientPreparedStatement.java:1396)
    at com.mysql.cj.jdbc.ClientPreparedStatement.getCoreParameterIndex(ClientPreparedStatement.java:1409)
    at com.mysql.cj.jdbc.ClientPreparedStatement.setInt(ClientPreparedStatement.java:1597)
```

课程总结

- 1 1. 掌握PreparedStatement使用方式 掌握
- 2 2. 封装ExecuteHandler执行类 基于提供代码，应用即可
- 3 3. 反射： 掌握
- 4 3-1 获取class对象
- 5 3-2 获取constructor对象
- 6 3-3 获取field对象
- 7 3-4 获取method对象

预习安排

三层架构

数据库连接池：C3P0

DBUtils工具类：