JDBC-01-授课笔记

一、JDBC快速入门

1.jdbc的概念

- JDBC（Java DataBase Connectivity,java数据库连接）是一种用于执行SQL语句的Java API，可以为多种关系型数据库提供统一访问，它是由一组用Java语言编写的类和接口组成的。

2.jdbc的本质

- 其实就是java官方提供的一套规范(接口)。用于帮助开发人员快速实现不同关系型数据库的连接!

3.jdbc的快速入门程序

1. 导入jar包
2. 注册驱动

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

3. 获取连接

```
Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/db2", "root",
"root");
```

4. 获取执行者对象

```
Statement stat = con.createStatement();
```

5. 执行sql语句，并接收返回结果

```
String sql = "SELECT * FROM user";
ResultSet rs = stat.executeQuery(sql);
```

6. 处理结果

```
while(rs.next()) {
    System.out.println(rs.getInt("id") + "\t" + rs.getString("name"));
}
```

7. 释放资源

```
con.close();
stat.close();
rs.close();
```

二、JDBC各个功能类详解

1.DriverManager

- DriverManager：驱动管理对象
  - 注册驱动(告诉程序该使用哪一个数据库驱动)
    - static void registerDriver(Driver driver)：注册与给定的驱动程序 DriverManager
    - 写代码使用：Class.forName("com.mysql.jdbc.Driver");
    - 通过查看源码发现：在com.mysql.jdbc.Driver类中存在静态代码块

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

    - 注意：mysql5之后的驱动jar包可以省略注册驱动的步骤。在jar包中，存在一个java.sql.Driver配置文件，文件中指定了com.mysql.jdbc.Driver
  - 获取数据库连接(获取到数据库的连接并返回连接对象)
    - static Connection getConnection(String url, String user, String password);
      - 返回值：Connection数据库连接对象
      - 参数
        - url：指定连接的路径。语法：jdbc:mysql://ip地址(域名):端口号/数据库名称
        - user：用户名
        - password：密码

2.Connection

- Connection：数据库连接对象
  - 获取执行者对象
    - 获取普通执行者对象：Statement createStatement();
    - 获取预编译执行者对象：PreparedStatement prepareStatement(String sql);
  - 管理事务
    - 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();
  - 释放资源
    - 立即将数据库连接对象释放：void close();

3.Statement

- Statement：执行sql语句的对象
  - 执行DML语句：int executeUpdate(String sql);
    - 返回值int：返回影响的行数。
    - 参数sql：可以执行insert、update、delete语句。
  - 执行DQL语句：ResultSet executeQuery(String sql);
    - 返回值ResultSet：封装查询的结果。
    - 参数sql：可以执行select语句。
  - 释放资源

        - 立即将执行者对象释放：void close();

4.ResultSet

- ResultSet：结果集对象
    - 判断结果集中是否还有数据：boolean next();
        - 有数据返回true，并将索引向下移动一行
        - 没有数据返回false
    - 获取结果集中的数据：XXX getXxx("列名");
        - XXX代表数据类型(要获取某列数据，这一列的数据类型)
        - 例如：String getString("name");　　int getInt("age");
    - 释放资源
        - 立即将结果集对象释放：void close();

三、JDBC案例student学生表的CRUD

1.数据准备

- 数据库和数据表

```
-- 创建db14数据库
CREATE DATABASE db14;

-- 使用db14数据库
USE db14;

-- 创建student表
CREATE TABLE student(
    sid INT PRIMARY KEY AUTO_INCREMENT, -- 学生id
    NAME VARCHAR(20),                   -- 学生姓名
    age INT,                            -- 学生年龄
    birthday DATE                       -- 学生生日
);

-- 添加数据
INSERT INTO student VALUES (NULL,'张三',23,'1999-09-23'),(NULL,'李四',24,'1998-
08-10'),(NULL,'王五',25,'1996-06-06'),(NULL,'赵六',26,'1994-10-20');
```

- 实体类
    - Student类，成员变量对应表中的列
    - 注意：所有的基本数据类型需要使用包装类，以防null值无法赋值

```
package com.itheima02.domain;

import java.util.Date;

public class Student {
    private Integer sid;
    private String name;
    private Integer age;
    private Date birthday;

    public Student() {
    }
```

```
    public Student(Integer sid, String name, Integer age, Date birthday) {
        this.sid = sid;
        this.name = name;
        this.age = age;
        this.birthday = birthday;
    }

    public Integer getSid() {
        return sid;
    }

    public void setSid(Integer sid) {
        this.sid = sid;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Integer getAge() {
        return age;
    }

    public void setAge(Integer age) {
        this.age = age;
    }

    public Date getBirthday() {
        return birthday;
    }

    public void setBirthday(Date birthday) {
        this.birthday = birthday;
    }

    @Override
    public String toString() {
        return "Student{" +
                "sid=" + sid +
                ", name='" + name + '\'' +
                ", age=" + age +
                ", birthday=" + birthday +
                '}';
    }
}
```

2.需求一：查询全部

- 持久层

```
/*
      查询所有学生信息
```

```
*/
@Override
public ArrayList<Student> findAll() {
    ArrayList<Student> list = new ArrayList<>();
    Connection con = null;
    Statement stat = null;
    ResultSet rs = null;
    try{
        //1.注册驱动
        Class.forName("com.mysql.jdbc.Driver");

        //2.获取数据库连接
        con =
DriverManager.getConnection("jdbc:mysql://192.168.59.129:3306/db14", "root",
"itheima");

        //3.获取执行者对象
        stat = con.createStatement();

        //4.执行sql语句，并且接收返回的结果集
        String sql = "SELECT * FROM student";
        rs = stat.executeQuery(sql);

        //5.处理结果集
        while(rs.next()) {
            Integer sid = rs.getInt("sid");
            String name = rs.getString("name");
            Integer age = rs.getInt("age");
            Date birthday = rs.getDate("birthday");

            //封装Student对象
            Student stu = new Student(sid,name,age,birthday);

            //将student对象保存到集合中
            list.add(stu);
        }

    } catch(Exception e) {
        e.printStackTrace();
    } finally {
        //6.释放资源
        if(con != null) {
            try {
                con.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }

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

        if(rs != null) {
```

```
            try {
                rs.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
    //将集合对象返回
    return list;
}
```

- 业务层

```
/*
    查询所有学生信息
*/
@Override
public ArrayList<Student> findAll() {
    return dao.findAll();
}
```

- 控制层

```
/*
    查询所有学生信息
*/
@Test
public void findAll() {
    ArrayList<Student> list = service.findAll();
    for(Student stu : list) {
        System.out.println(stu);
    }
}
```

3.需求二：条件查询

- 持久层

```
/*
    条件查询，根据id查询学生信息
*/
@Override
public Student findById(Integer id) {
    Student stu = new Student();
    Connection con = null;
    Statement stat = null;
    ResultSet rs = null;
    try{
        //1.注册驱动
        Class.forName("com.mysql.jdbc.Driver");

        //2.获取数据库连接
        con =
DriverManager.getConnection("jdbc:mysql://192.168.59.129:3306/db14", "root",
"itheima");
```

```
        //3.获取执行者对象
        stat = con.createStatement();

        //4.执行sql语句，并且接收返回的结果集
        String sql = "SELECT * FROM student WHERE sid='"+id+"'";
        rs = stat.executeQuery(sql);

        //5.处理结果集
        while(rs.next()) {
            Integer sid = rs.getInt("sid");
            String name = rs.getString("name");
            Integer age = rs.getInt("age");
            Date birthday = rs.getDate("birthday");

            //封装Student对象
            stu.setSid(sid);
            stu.setName(name);
            stu.setAge(age);
            stu.setBirthday(birthday);
        }

    } catch(Exception e) {
        e.printStackTrace();
    } finally {
        //6.释放资源
        if(con != null) {
            try {
                con.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }

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

        if(rs != null) {
            try {
                rs.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
    //将对象返回
    return stu;
}
```

- 业务层

```
/*
    条件查询，根据id查询学生信息
*/
@Override
public Student findById(Integer id) {
    return dao.findById(id);
}
```

- 控制层

```
/*
    条件查询，根据id查询学生信息
*/
@Test
public void findById() {
    Student stu = service.findById(3);
    System.out.println(stu);
}
```

4.需求三：新增数据

- 持久层

```
/*
      添加学生信息
*/
@Override
public int insert(Student stu) {
    Connection con = null;
    Statement stat = null;
    int result = 0;
    try{
        //1.注册驱动
        Class.forName("com.mysql.jdbc.Driver");

        //2.获取数据库连接
        con =
DriverManager.getConnection("jdbc:mysql://192.168.59.129:3306/db14", "root",
"itheima");

        //3.获取执行者对象
        stat = con.createStatement();

        //4.执行sql语句，并且接收返回的结果集
        Date d = stu.getBirthday();
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        String birthday = sdf.format(d);
        String sql = "INSERT INTO student VALUES
('"+stu.getSid()+"','"+stu.getName()+"','"+stu.getAge()+"','"+birthday+"')";
        result = stat.executeUpdate(sql);

    } catch(Exception e) {
        e.printStackTrace();
    } finally {
        //6.释放资源
        if(con != null) {
```

```
            try {
                con.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }

        if(stat != null) {
            try {
                stat.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
    //将结果返回
    return result;
}
```

- 业务层

```
/*
    新增学生信息
*/
@Override
public int insert(Student stu) {
    return dao.insert(stu);
}
```

- 控制层

```
/*
    新增学生信息
*/
@Test
public void insert() {
    Student stu = new Student(5,"周七",27,new Date());
    int result = service.insert(stu);
    if(result != 0) {
        System.out.println("新增成功");
    }else {
        System.out.println("新增失败");
    }
}
```

5.需求四：修改数据

- 持久层

```
/*
    修改学生信息
*/
@Override
public int update(Student stu) {
    Connection con = null;
    Statement stat = null;
```

```
    int result = 0;
    try{
        //1.注册驱动
        Class.forName("com.mysql.jdbc.Driver");

        //2.获取数据库连接
        con =
DriverManager.getConnection("jdbc:mysql://192.168.59.129:3306/db14", "root",
"itheima");

        //3.获取执行者对象
        stat = con.createStatement();

        //4.执行sql语句，并且接收返回的结果集
        Date d = stu.getBirthday();
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        String birthday = sdf.format(d);
        String sql = "UPDATE student SET
sid='"+stu.getSid()+"',name='"+stu.getName()+"',age='"+stu.getAge()+"',birthday=
'"+birthday+"' WHERE sid='"+stu.getSid()+"'";
        result = stat.executeUpdate(sql);

    } catch(Exception e) {
        e.printStackTrace();
    } finally {
        //6.释放资源
        if(con != null) {
            try {
                con.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }

        if(stat != null) {
            try {
                stat.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
    //将结果返回
    return result;
}
```

- 业务层

```
/*
    修改学生信息
*/
@Override
public int update(Student stu) {
    return dao.update(stu);
}
```

- 控制层

```
/*
    修改学生信息
*/
@Test
public void update() {
    Student stu = service.findById(5);
    stu.setName("周七七");

    int result = service.update(stu);
    if(result != 0) {
        System.out.println("修改成功");
    }else {
        System.out.println("修改失败");
    }
}
```

6.需求五：删除数据

- 持久层

```
/*
    删除学生信息
*/
@Override
public int delete(Integer id) {
    Connection con = null;
    Statement stat = null;
    int result = 0;
    try{
        //1.注册驱动
        Class.forName("com.mysql.jdbc.Driver");

        //2.获取数据库连接
        con =
DriverManager.getConnection("jdbc:mysql://192.168.59.129:3306/db14", "root",
"itheima");

        //3.获取执行者对象
        stat = con.createStatement();

        //4.执行sql语句，并且接收返回的结果集
        String sql = "DELETE FROM student WHERE sid='"+id+"'";
        result = stat.executeUpdate(sql);

    } catch(Exception e) {
        e.printStackTrace();
    } finally {
        //6.释放资源
        if(con != null) {
            try {
                con.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }

        if(stat != null) {
```

```
            try {
                stat.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
    //将结果返回
    return result;
}
```

- 业务层

```
/*
    删除学生信息
*/
@Override
public int delete(Integer id) {
    return dao.delete(id);
}
```

- 控制层

```
/*
    删除学生信息
*/
@Test
public void delete() {
    int result = service.delete(5);

    if(result != 0) {
        System.out.println("删除成功");
    }else {
        System.out.println("删除失败");
    }
}
```

四、JDBC工具类

1.工具类的抽取

- 配置文件(在src下创建config.properties)

```
driverClass=com.mysql.jdbc.Driver
url=jdbc:mysql://localhost:3306/db14
username=root
password=itheima
```

- 工具类

```
/*
    JDBC工具类
 */
public class JDBCUtils {
    //1.私有构造方法
```

```
    private JDBCUtils(){};

    //2.声明配置信息变量
    private static String driverClass;
    private static String url;
    private static String username;
    private static String password;
    private static Connection con;

    //3.静态代码块中实现加载配置文件和注册驱动
    static{
        try{
            //通过类加载器返回配置文件的字节流
            InputStream is =
JDBCUtils.class.getClassLoader().getResourceAsStream("config.properties");

            //创建Properties集合，加载流对象的信息
            Properties prop = new Properties();
            prop.load(is);

            //获取信息为变量赋值
            driverClass = prop.getProperty("driverClass");
            url = prop.getProperty("url");
            username = prop.getProperty("username");
            password = prop.getProperty("password");

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

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

    //4.获取数据库连接的方法
    public static Connection getConnection() {
        try {
            con = DriverManager.getConnection(url,username,password);
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return con;
    }

    //5.释放资源的方法
    public static void close(Connection con, Statement stat, ResultSet rs) {
        if(con != null) {
            try {
                con.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }

        if(stat != null) {
            try {
                stat.close();
```

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

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

    public static void close(Connection con, Statement stat) {
        close(con,stat,null);
    }
}
```

2.使用工具类优化student表的CRUD

- 查询全部

```
/*
    查询所有学生信息
*/
@Override
public ArrayList<Student> findAll() {
    ArrayList<Student> list = new ArrayList<>();
    Connection con = null;
    Statement stat = null;
    ResultSet rs = null;
    try{

        con = JDBCUtils.getConnection();

        //3.获取执行者对象
        stat = con.createStatement();

        //4.执行sql语句，并且接收返回的结果集
        String sql = "SELECT * FROM student";
        rs = stat.executeQuery(sql);

        //5.处理结果集
        while(rs.next()) {
            Integer sid = rs.getInt("sid");
            String name = rs.getString("name");
            Integer age = rs.getInt("age");
            Date birthday = rs.getDate("birthday");

            //封装Student对象
            Student stu = new Student(sid,name,age,birthday);

            //将student对象保存到集合中
            list.add(stu);
        }
```

```
    } catch(Exception e) {
        e.printStackTrace();
    } finally {
        //6.释放资源
        JDBCUtils.close(con,stat,rs);
    }
    //将集合对象返回
    return list;
}
```

- 条件查询

```
/*
    条件查询，根据id查询学生信息
*/
@Override
public Student findById(Integer id) {
    Student stu = new Student();
    Connection con = null;
    Statement stat = null;
    ResultSet rs = null;
    try{

        con = JDBCUtils.getConnection();

        //3.获取执行者对象
        stat = con.createStatement();

        //4.执行sql语句，并且接收返回的结果集
        String sql = "SELECT * FROM student WHERE sid='"+id+"'";
        rs = stat.executeQuery(sql);

        //5.处理结果集
        while(rs.next()) {
            Integer sid = rs.getInt("sid");
            String name = rs.getString("name");
            Integer age = rs.getInt("age");
            Date birthday = rs.getDate("birthday");

            //封装Student对象
            stu.setSid(sid);
            stu.setName(name);
            stu.setAge(age);
            stu.setBirthday(birthday);
        }

    } catch(Exception e) {
        e.printStackTrace();
    } finally {
        //6.释放资源
        JDBCUtils.close(con,stat,rs);
    }
    //将对象返回
    return stu;
}
```

- 新增数据

```
/*
     添加学生信息
*/
@Override
public int insert(Student stu) {
    Connection con = null;
    Statement stat = null;
    int result = 0;
    try{
        con = JDBCUtils.getConnection();

        //3.获取执行者对象
        stat = con.createStatement();

        //4.执行sql语句，并且接收返回的结果集
        Date d = stu.getBirthday();
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        String birthday = sdf.format(d);
        String sql = "INSERT INTO student VALUES
('"+stu.getSid()+"','"+stu.getName()+"','"+stu.getAge()+"','"+birthday+"')";
        result = stat.executeUpdate(sql);

    } catch(Exception e) {
        e.printStackTrace();
    } finally {
        //6.释放资源
        JDBCUtils.close(con,stat);
    }
    //将结果返回
    return result;
}
```

- 修改数据

```
/*
     修改学生信息
*/
@Override
public int update(Student stu) {
    Connection con = null;
    Statement stat = null;
    int result = 0;
    try{
        con = JDBCUtils.getConnection();

        //3.获取执行者对象
        stat = con.createStatement();

        //4.执行sql语句，并且接收返回的结果集
        Date d = stu.getBirthday();
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        String birthday = sdf.format(d);
        String sql = "UPDATE student SET
sid='"+stu.getSid()+"',name='"+stu.getName()+"',age='"+stu.getAge()+"',birthday=
'"+birthday+"' WHERE sid='"+stu.getSid()+"'";
        result = stat.executeUpdate(sql);
```

```
    } catch(Exception e) {
        e.printStackTrace();
    } finally {
        //6.释放资源
        JDBCUtils.close(con,stat);
    }
    //将结果返回
    return result;
}
```

- 删除数据

```
/*
   删除学生信息
*/
@Override
public int delete(Integer id) {
    Connection con = null;
    Statement stat = null;
    int result = 0;
    try{
        con = JDBCUtils.getConnection();

        //3.获取执行者对象
        stat = con.createStatement();

        //4.执行sql语句，并且接收返回的结果集
        String sql = "DELETE FROM student WHERE sid='"+id+"'";
        result = stat.executeUpdate(sql);

    } catch(Exception e) {
        e.printStackTrace();
    } finally {
        //6.释放资源
        JDBCUtils.close(con,stat);
    }
    //将结果返回
    return result;
}
```

3.student表的CRUD整合页面

- 用户表的数据准备

```
-- 创建用户表
CREATE TABLE USER(
    uid VARCHAR(50) PRIMARY KEY,    -- 用户id
    ucode VARCHAR(50),              -- 用户标识
    loginname VARCHAR(100),         -- 登录用户名
    PASSWORD VARCHAR(100),          -- 登录密码
    username VARCHAR(100),          -- 用户名
    gender VARCHAR(10),             -- 用户性别
    birthday DATE,                  -- 出生日期
    dutydate DATE                   -- 入职日期
);
```

```
-- 添加一条测试数据
INSERT INTO `user` VALUES ('11111111', 'zhangsan001', 'zhangsan', '1234', '张三',
'男', '2008-10-28', '2018-10-28');
```

- 将student表的dao层操作复制到项目中的dao层即可

```
public class StudentDaoImpl implements StudentDao {

    /*
        查询所有学生信息
     */
    @Override
    public ArrayList<Student> findAll() {
        Connection con = null;
        Statement stat = null;
        ResultSet rs = null;
        ArrayList<Student> list = new ArrayList<>();
        try {
            //1.获取连接
            con = JDBCUtils.getConnection();

            //2.获取执行者对象
            stat = con.createStatement();

            //3.执行sql语句，并接收结果
            String sql = "SELECT * FROM student";
            rs = stat.executeQuery(sql);

            //4.处理结果，将每条记录封装成一个Student对象。将多个Student对象保存到集合中
            while(rs.next()) {
                Integer sid = rs.getInt("sid");
                String name = rs.getString("name");
                Integer age = rs.getInt("age");
                Date birthday = rs.getDate("birthday");

                Student stu = new Student(sid,name,age,birthday);

                list.add(stu);
            }

        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            //5.释放资源
            JDBCUtils.close(con,stat,rs);
        }

        return list;
    }

    /*
        条件查询，根据id查询学生信息
     */
    @Override
    public Student findById(Integer id) {
        Connection con = null;
```

```
        Statement stat = null;
        ResultSet rs = null;
        Student stu = new Student();
        try {
            //1.获取连接
            con = JDBCUtils.getConnection();

            //2.获取执行者对象
            stat = con.createStatement();

            //3.执行sql语句，并接收结果
            String sql = "SELECT * FROM student WHERE sid='"+id+"'";
            rs = stat.executeQuery(sql);

            //4.处理结果，将记录封装成一个Student对象。
            if(rs.next()) {
                Integer sid = rs.getInt("sid");
                String name = rs.getString("name");
                Integer age = rs.getInt("age");
                Date birthday = rs.getDate("birthday");

                stu.setSid(sid);
                stu.setName(name);
                stu.setAge(age);
                stu.setBirthday(birthday);
            }

        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            //5.释放资源
            JDBCUtils.close(con,stat,rs);
        }

        return stu;
    }

    /*
        新增学生信息
     */
    @Override
    public int insert(Student stu) {
        Connection con = null;
        Statement stat = null;
        int result = 0;
        try{
            //1.获取连接
            con = JDBCUtils.getConnection();

            //2.获取执行者对象
            stat = con.createStatement();

            //3.执行sql语句，并接收结果
            Date date = stu.getBirthday();
            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
            String birthday = sdf.format(date);
            String sql = "INSERT INTO student VALUES
(null,'"+stu.getName()+"','"+stu.getAge()+"','"+birthday+"')";
```

```
            result = stat.executeUpdate(sql);

        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            //4.释放资源
            JDBCUtils.close(con,stat);
        }

        return result;
    }

    /*
        修改学生信息
     */
    @Override
    public int update(Student stu) {
        Connection con = null;
        Statement stat = null;
        int result = 0;
        try{
            //1.获取连接
            con = JDBCUtils.getConnection();

            //2.获取执行者对象
            stat = con.createStatement();

            //3.执行sql语句，并接收结果
            Date date = stu.getBirthday();
            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
            String birthday = sdf.format(date);
            String sql = "UPDATE student SET
sid='"+stu.getSid()+"',name='"+stu.getName()+"',age='"+stu.getAge()+"',birthday=
'"+birthday+"' WHERE sid='"+stu.getSid()+"'";
            result = stat.executeUpdate(sql);

        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            //4.释放资源
            JDBCUtils.close(con,stat);
        }

        return result;
    }

    /*
        删除学生信息
     */
    @Override
    public int delete(Integer id) {
        Connection con = null;
        Statement stat = null;
        int result = 0;
        try{
            //1.获取连接
            con = JDBCUtils.getConnection();
```

```
            //2.获取执行者对象
            stat = con.createStatement();

            //3.执行sql语句，并接收结果
            String sql = "DELETE FROM student WHERE sid='"+id+"'";
            result = stat.executeUpdate(sql);

        } catch (SQLException e) {
            e.printStackTrace();
        }  finally {
            //4.释放资源
            JDBCUtils.close(con,stat);
        }

        return result;
    }
}
```

五、SQL注入攻击

1.sql注入攻击的演示

- 在登录界面，输入一个错误的用户名或密码，也可以登录成功

2.sql注入攻击的原理

- 按照正常道理来说，我们在密码处输入的所有内容，都应该认为是密码的组成
- 但是现在Statement对象在执行sql语句时，将一部分内容当做查询条件来执行了

3.PreparedStatement的介绍

- 预编译sql语句的执行者对象。在执行sql语句之前，将sql语句进行提前编译。明确sql语句的格式后，就不会改变了。剩余的内容都会认为是参数！参数使用?作为占位符
- 为参数赋值的方法：setXxx(参数1,参数2);
  - 参数1：?的位置编号(编号从1开始)
  - 参数2：?的实际参数
- 执行sql语句的方法

- 执行insert、update、delete语句：int executeUpdate();
- 执行select语句：ResultSet executeQuery();

4.PreparedStatement的使用

```
/*
     使用PreparedStatement的登录方法，解决注入攻击
*/
@Override
public User findByLoginNameAndPassword(String loginName, String password) {
    //定义必要信息
    Connection conn = null;
    PreparedStatement pstm = null;
    ResultSet rs = null;
    User user = null;
    try {
        //1.获取连接
        conn = JDBCUtils.getConnection();
        //2.创建操作SQL对象
        String sql = "SELECT * FROM user WHERE loginname=? AND password=?";
        pstm = conn.prepareStatement(sql);
        //3.设置参数
        pstm.setString(1,loginName);
        pstm.setString(2,password);
        System.out.println(sql);
        //4.执行sql语句，获取结果集
        rs = pstm.executeQuery();
        //5.获取结果集
        if (rs.next()) {
            //6.封装
            user = new User();
            user.setUid(rs.getString("uid"));
            user.setUcode(rs.getString("ucode"));
            user.setUsername(rs.getString("username"));
            user.setPassword(rs.getString("password"));
            user.setGender(rs.getString("gender"));
            user.setDutydate(rs.getDate("dutydate"));
            user.setBirthday(rs.getDate("birthday"));
            user.setLoginname(rs.getString("loginname"));
        }
        //7.返回
        return user;
    }catch (Exception e){
        throw new RuntimeException(e);
    }finally {
        JDBCUtils.close(conn,pstm,rs);
    }
}
```

5.使用PreparedStatement优化student表的CRUD（作业）

```
public class StudentDaoImpl implements StudentDao {

    @Override
    public ArrayList<Student> findAll() {
        //定义必要信息
```

```
        Connection conn = null;
        PreparedStatement pstm = null;
        ResultSet rs = null;
        ArrayList<Student> students = null;
        try {
            //1.获取连接
            conn = JDBCUtils.getConnection();
            //2.获取操作对象
            pstm = conn.prepareStatement("select * from student");
            //3.执行sql语句，获取结果集
            rs = pstm.executeQuery();
            //4.遍历结果集
            students = new ArrayList<Student>();
            while (rs.next()) {
                //5.封装
                Student student = new Student();
                student.setSid(rs.getInt("sid"));
                student.setName(rs.getString("name"));
                student.setAge(rs.getInt("age"));
                student.setBirthday(rs.getDate("birthday"));
                //加入到集合中
                students.add(student);
            }
            //6.返回
            return students;
        }catch (Exception e){
            throw new RuntimeException(e);
        }finally {
            JDBCUtils.close(conn,pstm,rs);
        }
    }

    @Override
    public Student findById(Integer sid) {
        //定义必要信息
        Connection conn = null;
        PreparedStatement pstm = null;
        ResultSet rs = null;
        Student student = null;
        try {
            //1.获取连接
            conn = JDBCUtils.getConnection();
            //2.获取操作对象
            pstm = conn.prepareStatement("select * from student where sid = ?
");
            pstm.setInt(1,sid);
            //3.执行sql语句，获取结果集
            rs = pstm.executeQuery();
            //4.遍历结果集
            if (rs.next()) {
                //5.封装
                student = new Student();
                student.setSid(rs.getInt("sid"));
                student.setName(rs.getString("name"));
                student.setAge(rs.getInt("age"));
                student.setBirthday(rs.getDate("birthday"));
            }
            //6.返回
```

```
            return student;
        }catch (Exception e){
            throw new RuntimeException(e);
        }finally {
            JDBCUtils.close(conn,pstm,rs);
        }
    }

    @Override
    public int insert(Student student) {
        //定义必要信息
        Connection conn = null;
        PreparedStatement pstm = null;
        int result = 0;
        try {
            //1.获取连接
            conn = JDBCUtils.getConnection();
            //2.获取操作对象
            pstm = conn.prepareStatement("insert into
student(sid,name,age,birthday)values(null,?,?,?)");
            //3.设置参数
            //pstm.setInt(1,null);
            pstm.setString(1,student.getName());
            pstm.setInt(2,student.getAge());
            pstm.setDate(3,new Date(student.getBirthday().getTime()));
            //4.执行sql语句
            result = pstm.executeUpdate();
        }catch (Exception e){
            throw new RuntimeException(e);
        }finally {
            JDBCUtils.close(conn,pstm);
        }
        return result;
    }

    @Override
    public int update(Student student) {
        //定义必要信息
        Connection conn = null;
        PreparedStatement pstm = null;
        int result = 0;
        try {
            //1.获取连接
            conn = JDBCUtils.getConnection();
            //2.获取操作对象
            pstm = conn.prepareStatement("update student set
name=?,age=?,birthday=? where sid=? ");
            //3.设置参数
            pstm.setString(1,student.getName());
            pstm.setInt(2,student.getAge());
            pstm.setDate(3,new Date(student.getBirthday().getTime()));
            pstm.setInt(4,student.getSid());
            //4.执行sql语句
            result = pstm.executeUpdate();
        }catch (Exception e){
            throw new RuntimeException(e);
        }finally {
            JDBCUtils.close(conn,pstm);
```

```
        }
        return result;
    }

    @Override
    public int delete(Integer sid) {
        //定义必要信息
        Connection conn = null;
        PreparedStatement pstm = null;
        int result = 0;
        try {
            //1.获取连接
            conn = JDBCUtils.getConnection();
            //2.获取操作对象
            pstm = conn.prepareStatement("delete from student where sid=? ");
            //3.设置参数
            pstm.setInt(1,sid);
            //4.执行sql语句
            result = pstm.executeUpdate();
        }catch (Exception e){
            throw new RuntimeException(e);
        }finally {
            JDBCUtils.close(conn,pstm);
        }
        return result;
    }
}
```

六、综合案例-课程表批量新增加事务管理

1.service层

- 接口

```
/*
     批量添加
*/
void batchAdd(List<User> users);
```

- 实现类

```
/*
      事务要控制在此处
*/
@Override
public void batchAdd(List<User> users) {
    //获取数据库连接
    Connection connection = JDBCUtils.getConnection();
    try {
        //开启事务
        connection.setAutoCommit(false);
        for (User user : users) {
            //1.创建ID,并把UUID中的-替换
            String uid = UUID.randomUUID().toString().replace("-",
"").toUpperCase();
            //2.给user的uid赋值
```

```
            user.setUid(uid);
            //3.生成员工编号
            user.setUcode(uid);

            //模拟异常
            //int n = 1 / 0;

            //4.保存
            userDao.save(connection,user);
        }
        //提交事务
        connection.commit();
    }catch (Exception e){
        try {
            //回滚事务
            connection.rollback();
        }catch (Exception ex){
            ex.printStackTrace();
        }
        e.printStackTrace();
    }finally {
        JDBCUtils.close(connection,null,null);
    }
}
```

2.dao层

- 接口

```
/**
    支持事务的添加
*/
void save(Connection connection,User user);
```

- 实现类

```
/*
       支持事务的添加
*/
@Override
public void save(Connection connection, User user) {
    //定义必要信息
    PreparedStatement pstm = null;
    try {
        //1.获取连接
        connection = JDBCUtils.getConnection();
        //2.获取操作对象
        pstm = connection.prepareStatement("insert into
user(uid,ucode,loginname,password,username,gender,birthday,dutydate)values(?,?,?
,?,?,?,?,?)");
        //3.设置参数
        pstm.setString(1,user.getUid());
        pstm.setString(2,user.getUcode());
        pstm.setString(3,user.getLoginname());
        pstm.setString(4,user.getPassword());
        pstm.setString(5,user.getUsername());
```

```
            pstm.setString(6,user.getGender());
            pstm.setDate(7,new Date(user.getBirthday().getTime()));
            pstm.setDate(8,new Date(user.getDutydate().getTime()));
            //4.执行sql语句，获取结果集
            pstm.executeUpdate();
        }catch (Exception e){
            throw new RuntimeException(e);
        }finally {
            JDBCUtils.close(null,pstm,null);
        }
}
```