# 实验10 JDBC连接MySQL

一、实验目的：

1、掌握JDBC连接数据库的一般操作。

2、理解JavaBean的基本作用。

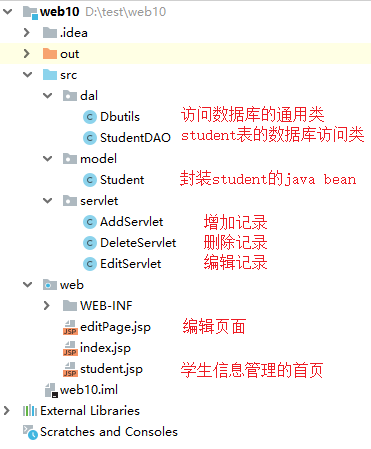
3、理解分层设计的思想。

1. 实验内容：

实现学生信息的增删改查操作。

|  |
| --- |
|  |
| 图1 学生信息管理的首页 |

注：通过以下提供的实验步骤，最终得到的项目结构如下所示：



【实验步骤】

（1）将jar包“mysql-connector-java-5.0.8-bin.jar”拷贝到tomcat的lib文件夹中。

（2）在MySQL中，创建名为“my\_db”的数据库，并在数据库中创建student表。

**create database my\_db;**

**use my\_db;**

**create table student**

**(sno char(5) primary key,**

**sname varchar(50),**

**sage int);**

**insert into student**

**values('99001','张三',20),('99002','李四',21);**

（3）新建一个名为”web10”的项目（Web应用程序）。在项目中新建“Java包”，命名为“model”。

（4）在model包中新建“Java类”，命名为“Student”。Student.java为封装student表的JavaBean。

/\*\*

\* 与学生表对应的实体类（封装数据的JavaBean），目的是方便数据传输以及代码维护。

\* JavaBean的要求：

\* 1、是一个Java 类

\* 2、必须存在无参数构造方法

\* 3、所有成员变量都是private

\* 4、私有化的属性必须通过public类型的方法暴露给其它程序， 并且方法的命名也必须遵守一定的命名规范。

\* 即如果想private成员变量被外界访问，提供public的get和set方法。

\*/

package model;

public class Student {

private String sno;

private String sname;

private int sage;

public Student(String sno, String sname, int sage) {

this.sno = sno;

this.sname = sname;

this.sage = sage;

}

public Student() {

}

public String getSno() {

return sno;

}

public void setSno(String sno) {

this.sno = sno;

}

public String getSname() {

return sname;

}

public void setSname(String sname) {

this.sname = sname;

}

public int getSage() {

return sage;

}

public void setSage(int sage) {

this.sage = sage;

}

}

（5）在项目中新建“Java包”，命名为“dal”（Data Access Layer）。该包中存储的文件主要用于数据库访问。

（6）在dal包中创建“Java类”，命名为“Dbutils”。该文件中定义了访问数据库的通用读写方法。

package dal;

import java.sql.\*;

public class Dbutils {

protected Connection conn = null;

protected PreparedStatement pstmt = null;

protected ResultSet rs = null;

/\*\*

\* 获取连接对象

\* @return 连接对象

\*/

public Connection getConnection() {

try {

//1、加载驱动类

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

//2、创建连接对象

String url = "jdbc:mysql://localhost:3306/my\_db?useUnicode=true&characterEncoding=utf-8";

String user = "root";

String password = "";

conn = DriverManager.getConnection(url,user,password);

} catch (Exception e) {

e.printStackTrace();

}

return conn;

}

/\*\*

\* 关闭资源

\*

\* @param conn

\* @param pstmt

\* @param rs

\*/

public void closeAll(Connection conn, PreparedStatement pstmt, ResultSet rs) {

try {

if (rs != null) {

rs.close();

}

if (pstmt != null) {

pstmt.close();

}

if (conn != null) {

conn.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

/\*\*

\* 增删改操作

\*

\* @param sql

\* @param params

\* @return

\*/

public int executeUpdate(String sql, Object[] params) {

this.getConnection();

int result = 0;

try {

//3、创建prepareStatement对象

pstmt = conn.prepareStatement(sql);

//4、为占位符赋值

if (null != params) {

for (int i = 0; i < params.length; i++) {

pstmt.setObject(i + 1, params[i]);

}

}

//5、调用方法：执行sql语句

result = pstmt.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

} finally {

this.closeAll(conn, pstmt, null);

}

return result;

}

/\*\*

\* 查询方法

\*

\* @param sql

\* @param params

\* @return

\*/

public ResultSet executeQuery (String sql, Object[] params) {

this.getConnection();

try {

//3、创建prepareStatement对象

pstmt = conn.prepareStatement(sql);

//4、为占位符赋值

if (null != params) {

for (int i = 0; i < params.length; i++) {

pstmt.setObject(i + 1, params[i]);

}

}

//5、调用方法：执行sql语句

rs = pstmt.executeQuery();

} catch (SQLException e) {

e.printStackTrace();

}

//后面具体的查询方法还需要用到rs ,所以此处最后不能关闭数据流

return rs;

}

}

（7）在dal包中创建“Java类”，命名为“StudentDAO”。该文件中定义了对Student表进行增删改查操作的方法。该类中，需使用model包中的Student类。

package dal;

import model.Student;

import java.util.List;

import java.util.ArrayList;

import java.sql.\*;

/\*\*

\* 对Student表进行增删改查的方法

\*/

public class StudentDAO {

Dbutils dbutils = new Dbutils();

//返回所有的学生信息

public List<Student> findAllStudents() {

String sql = "select sno,sname,sage from student";

Object[] params = null;

List<Student> list = new ArrayList<>();

ResultSet rs = dbutils.executeQuery (sql, params);

try {

if (null != rs) {

while (rs.next()) {

Student student = new Student();

student.setSno(rs.getString("sno"));

student.setSname(rs.getString("sname"));

student.setSage(rs.getInt("sage"));

list.add(student);

}

}

} catch (SQLException ex) {

ex.printStackTrace();

} finally {

dbutils.closeAll(dbutils.conn, dbutils.pstmt, dbutils.rs);

}

return list;

}

//根据学号查找学生信息

public Student findBySno(String sno) {

Student student = new Student();

String sql = "select sno,sname,sage from student where sno=?";

Object[] params = {sno};

ResultSet rs = dbutils.executeQuery (sql, params);

try {

if (null != rs) {

if (rs.next()) {

student.setSno(rs.getString("sno"));

student.setSname(rs.getString("sname"));

student.setSage(rs.getInt("sage"));

}

}

} catch (SQLException ex) {

ex.printStackTrace();

} finally {

dbutils.closeAll(dbutils.conn, dbutils.pstmt, dbutils.rs);

}

return student;

}

//判断学号在student表中是否存在

public boolean isSnoExist(String sno) {

boolean flag = false;

String sql = "select sno,sname,sage from student where sno=?";

Object[] params = {sno};

ResultSet rs = dbutils.executeQuery (sql, params);

try {

if (rs != null && rs.next()) {

flag = true;

}

} catch (SQLException ex) {

ex.printStackTrace();

}

return flag;

}

//插入一条学生记录

public boolean insert(Student student) {

boolean flag = false;

String sql = "insert into student(sno,sname,sage) values(?,?,?)";

Object[] params = {student.getSno(), student.getSname(), student.getSage()};

return (dbutils.executeUpdate(sql, params) == 1);

}

//根据学号，更新该生的其他属性

public boolean update(Student student) {

String sql = "update student set sname=?,sage=? WHERE sno=?";

Object[] params = {student.getSname(), student.getSage(), student.getSno()};

return dbutils.executeUpdate(sql, params) == 1;

}

//根据学号删除学生信息

public boolean deleteBySno(String sno) {

String sql = "delete from student where sno=?";

Object[] params = {sno};

return dbutils.executeUpdate(sql, params) == 1;

}

}

（8）在项目中新建“JSP文件”，命名为“student”。该文件为“学生信息管理”的首页。

在该文件中，使用了model包中的Student类，以及dal包中的StudentDAO类。其中，通过dal.StudentDAO类的findAllStudents()方法，返回了student表中的全部记录。

<%@page import="dal.StudentDAO"%>

<%@page import="java.util.List"%>

<%@page import="model.Student"%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>学生信息管理</title>

</head>

<body>

<div align="center">

<h1>学生信息管理</h1>

<%

StudentDAO studentDAO = new StudentDAO();

//获取所有学生信息

List<Student> students = studentDAO.findAllStudents();

%>

<table border='1' width="80%" >

<tr><th width="20%">编号</th>

<th width="20%">学生学号</th>

<th width="20%">姓名</th>

<th width="20%">年龄</th>

<th width="20%">操作</th>

</tr>

<%

int rowNum = 0;//行编号

for (int i = 0; i < students.size(); i++) {

rowNum++;

String sno = students.get(i).getSno();

String sname = students.get(i).getSname();

int sage = students.get(i).getSage();

String deleteLink = "delete?sno=" + sno;

String editLink = "editPage.jsp?sno=" + sno;

%>

<tr>

<td><%= rowNum%></td>

<td><%= sno%></td>

<td><%= sname%></td>

<td><%= sage%></td>

<!--点击“删除”链接，使用javascript脚本弹出确认对话框，单击“确定”后，才执行删除操作-->

<td><a href="<%=deleteLink%>" onClick="return confirm('您确定要删除吗？');">删除</a>

<a href='<%=editLink%>'>修改</a></td>

</tr>

<%

}

%>

</table>

<br/>

<form action="add" method = "get">

学号 <input name="sno">

姓名 <input name="sname">

年龄 <input name="sage">

<input type="submit" value="增加"></br>

</form>

</div>

</body>

</html>

（9）在项目中新建“JSP文件”，命名为“editPage”。该文件为编辑文件的页面。

该文件中，同样使用了model包的Student类以及dal包的StudentDAO类。其中，通过调用dal.studentDAO类的findBySno(Sring sno)方法，返回被选中的学生的基本信息。

<%@page import="model.Student"%>

<%@page import="dal.StudentDAO"%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>JSP Page</title>

</head>

<body>

<h1>编辑学生信息</h1>

<%

StudentDAO studentDAO = new StudentDAO();

String sno = request.getParameter("sno");

Student student = studentDAO.findBySno(sno);

String sname = student.getSname();

int sage = student.getSage();

%>

<form action="edit" method = "get">

<!--通过readonly="true"，设置学号文本框不能修改-->

学号 <input name="sno" readonly="true" value=<%=sno%>><br/>

姓名 <input name="sname" value=<%=sname%>><br/>

年龄 <input name="sage" value=<%=sage%>><br/><br/>

<input type="submit" value="提交"></br>

</form>

</body>

</html>

（10）在项目中新建“Java包”，命名为“servlet”。该包中存储项目的servlet文件。

（11）在“servlet”包中新建“Java类”，命名为“AddServlet”，实现学生记录的添加。

如果用户输入的学号在学生表存在，则提示“学号重复，添加失败”；否则将学生记录插入到学生表中，并提示“添加成功”。最后，跳转到首页“student.jsp”中。

该文件中调用了dal.StudentDAO类的insert(Student student)方法，实现学生记录的添加。

package servlet;

import dal.StudentDAO;

import java.io.IOException;

import javax.servlet.\*;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.\*;

import model.Student;

@WebServlet("/add")

public class AddServlet extends HttpServlet {

@Override

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

//设置请求对象的编码方案为utf-8

req.setCharacterEncoding("utf-8");

//设置响应对象以及浏览器的字符方案为utf-8

resp.setContentType("text/html;charset=utf-8");

StudentDAO studentDAO = new StudentDAO();

//获取用户输入的学号,判断学号在学生表中是否存在

String sno = req.getParameter("sno");

if (studentDAO.isSnoExist(sno)) {

resp.getWriter().println("<script>alert('学号重复，添加失败');window.location.href='student.jsp'</script>");

} else {

//获取用户输入的姓名、年龄

String sname = req.getParameter("sname");

String sageStr = req.getParameter("sage");

Integer sage = Integer.parseInt(sageStr);

Student student = new Student(sno, sname, sage);

studentDAO.insert(student);

//使用javascript脚本输出提示对话框，并跳转到student.jsp页面中

resp.getWriter().println("<script>alert('添加成功');window.location.href='student.jsp'</script>");

}

//不能使用resp.sendRedirect("student.jsp");否则，javascript对话框无法显示。

}

}

（12）在“serlvet”包中新建“Java类”，命名为“DeleteServlet”。

该文件通过调用dal.StudentDAO类的deleteBySno(String sno)，实现根据学号删除学生记录的操作。

package servlet;

import dal.StudentDAO;

import java.io.IOException;

import javax.servlet.\*;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.\*;

@WebServlet("/delete")

public class DeleteServlet extends HttpServlet {

@Override

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

StudentDAO studentDAO = new StudentDAO();

String sno = req.getParameter("sno");

//根据用户输入的学号，删除学生记录

studentDAO.deleteBySno(sno);

//重定向到首页student.jsp

resp.sendRedirect("student.jsp");

}

}

（13）在“serlvet”包中新建“Java类”，命名为“EditServlet”。

该文件通过调用dal.StudentDAO类的update(Student student)，实现了根据学号修改学生记录的操作。

package servlet;

import dal.StudentDAO;

import java.io.IOException;

import javax.servlet.\*;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.\*;

import model.Student;

@WebServlet("/edit")

public class EditServlet extends HttpServlet {

@Override

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

req.setCharacterEncoding("utf-8");

resp.setContentType("text/html;charset=utf-8");

StudentDAO studentDAO = new StudentDAO();

String sno = req.getParameter("sno");

String sname = req.getParameter("sname");

String sageStr = req.getParameter("sage");

int sage = Integer.parseInt(sageStr);

Student student = new Student(sno, sname, sage);

if (studentDAO.update(student)) {

resp.getWriter().println("<script>alert('修改成功');window.location.href='student.jsp'</script>");

} else {

resp.getWriter().println("<script>alert('程序发生了未知的错误，修改失败');window.location.href='student.jsp'</script>");

}

}

}