Web Applications

18302098 易辉轩

由于篇幅原因，无法在实验报告中提交全部源代码，源代码可以在github上找到：

https://github.com/Yile1/WebApp

实验目的

This lab introduces you to construction of Web applications

实验内容

```
1. Create a web page with separate forms, and corresponding servlets, to
perform each of the following actions:
a. Accept a roll number, and display the following data about the student:
name, department, and all courses taken, in tabular form: list the course
id, title, credits and grade (show a blank if the grade is null).
b. Accept a word, and find all courses whose title contains that word
c. Show all students with two or more fail grades; no need to show the
courses they have failed, and I don't care if they passed the course
subsequently.
d. Take a roll number, name and department name, and insert a new student
record with tot_creds set to 0. Make sure you catch exceptions and report
them (and test it with an incorrect department name, to see what happens
when a foreign key constraint is violated, and similarly with a duplicate
roll number, to see what happens when a primary key constraint is
violated). Make sure also to close connections, even if there is an
exception.
2. This part of the assignment illustrates how it is useful to create
functions that generate HTML, instead of writing it directly.
a. Write a function createDropDown(ResultSet) that takes a ResultSet, and
creates a drop-down menu from the ResultSet; the first attribute is used
as the result value, and all attributes are displayed in the drop-down
menu, concatenated together.
b. Also write a function createDropDown(String) which does the same thing,
but instead of a ResultSet it takes an SQL query as a string; the second
version can be used safely as long as the query does not involve any value
that the user inputs, otherwise it would be vulnerable to SQL injection
attacks.
c. Create form interfaces to insert records for the student and instructor
tables, with a drop-down menu for the department name, showing all valid
department names. As before, exceptions should be caught and reported.
```

实验步骤

框架

数据库：

- 利用Druid连接数据库并维护数据库连接池

- 为了简化JDBC的代码，使用DBUtils连接并操作数据库（DBUtils是JDBC的一个工具类库）

后端：Java Servlet

前端：vue + axios + bootstrap（添加了一些简单样式）

服务器：tomcat（运行在http://localhost:8080/WebApp）

代码结构

主要代码结构如下：

```
|- WebApp_Lab
|
|   |- src
|       |- pers
|           |- pojo
|               |- Student.class: 构建Student对象的JavaBean
|               |- Course.class: 构建Course对象的JavaBean
|               |- Instructor.class: 构建Instructor对象的JavaBean
|           |- utils
|               |- JdbcUtils.class: 提供数据库连接的工具类
|           |- dao
|               |- impl
|                   |- BaseDao.class: 数据库操作的基础类
|                   |- UniversityDaoImpl.class: 查找操作university数据库的接
口实现
|               |- UniversityDao.interface: 查找操作university数据库的接口
|           |- service
|               |- impl
|                   |- UniversityServiceImpl.class: web功能接口实现
|               |- UniversityService.interface: web功能接口
|           |- test
|               |- JdbcUtilsTest.class: JdbcUtils类测试
|               |- UniversityDaoImplTest.class: UniversityDaoImpl类测试
|           |- web
|               |- StudentServlet.class: 查找学生信息的Servlet程序
|               |- CourseServlet.class: 查找课程信息的Servlet程序
|               |- InsertServlet: 插入学生、教师信息的Servlet程序
|               |- DeptServlet: 查找院系的Servlet程序
|               |- SqlServlet: 利用sql查询的Servlet程序
```

```
|
|       |- jdbc.properties: 数据库配置信息
|
|   |- webapp
|       |- static
|           |- main.js: vue程序
|           |- jquery-1.7.2.js
|       |- WEB-INF
|           |- lib: 项目所需jar包
|           |- web.xml: servlet配置
|       |- index.html: web主页面
|
|   |- README.md: 项目说明文档
```

功能实现

展示部分关键代码

1.1

1.1 Accept a roll number, and display the following data about the student: name, department, and all courses taken, in tabular form: list the course id, title, credits and grade (show a blank if the grade is null).

pers.dao.impl.UniversityDaoImpl

```
@Override
public List<Course> queryCourseByStudentID(String ID) {
    String sql = "select course.course_id as courseID, course.title,
course.dept_name as deptName, course.credits, takes.grade from takes,
course where takes.ID = ? and takes.course_id = course.course_id";
    return queryForList(Course.class, sql, ID);
}
```

pers.service.impl.UniversityServiceImpl

```
@Override
public List<Course> searchCourseByStudentID(String ID) {
    return universityDao.queryCourseByStudentID(ID);
}
```

pers.web.CourseServlet

```
protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        PrintWriter out = resp.getWriter();
        Gson gson = new Gson();
        String solve = req.getParameter("solve");
        if (solve.equals("1")){
            String ID = req.getParameter("ID");
            List<Course> lists =
universityService.searchCourseByStudentID(ID);
            String info = gson.toJson(lists);
            System.out.println(info);
            out.write(info);
        }
```

1.2

1.2.Accept a word, and find all courses whose title contains that word

pers.dao.impl.UniversityDaoImpl

```
@Override
public List<Course> queryCourseLikeTitle(String subTitle) {
    StringBuilder sql = new StringBuilder();
    sql.append("select course_id as courseID, title, dept_name as
deptName, credits from course where title like '%");
    sql.append(subTitle + "%'");
    return queryForList(Course.class, sql.toString());
}
```

pers.service.impl.UniversityServiceImpl

```
 @Override
public List<Course> searchCourseByTitle(String subTitle) {
    return universityDao.queryCourseLikeTitle(subTitle);
}
```

pers.web.CourseServlet

```
 protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        PrintWriter out = resp.getWriter();
        Gson gson = new Gson();
        String solve = req.getParameter("solve");
        if(solve.equals("2")){
            String keyWord = req.getParameter("keyWord");
            List<Course> lists =
universityService.searchCourseByTitle(keyWord);
            String info = gson.toJson(lists);
            System.out.println(info);
            out.write(info);
        }
```

1.3

1.3.Show all students with two or more fail grades; no need to show the courses they have failed, and I don't care if they passed the course subsequently.

由于数据库中不存在成绩为F的学生，故把成绩为C-定义为failed

pers.dao.impl.UniversityDaoImpl

```
@Override
public List<Student> queryStudentFail(Integer failCnt) {
    String sql = "with fail_grade(ID, name, deptName, totCred) as " +
        "(select ID, name, student.dept_name, tot_cred " +
        "from student natural join takes " +
        "where grade = 'F' or grade = 'C-') " +
        "select * " +
        "from fail_grade " +
        "group by ID, name, deptName, totCred " +
        "having count(ID) >= ?";
    return queryForList(Student.class, sql, failCnt);
}
```

pers.service.impl.UniversityServiceImpl

```
@Override
public List<Student> findStudentFail(int failCourseNum) {
    return universityDao.queryStudentFail(failCourseNum);
}
```

pers.web.StudentServlet

```
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
    PrintWriter out = resp.getWriter();
    Gson gson = new Gson();
    String solve = req.getParameter("solve");
    if(solve.equals("3")){
        String parameter = req.getParameter("failNum");
        Integer failNum = Integer.valueOf(parameter);
        List<Student> lists = universityService.findStudentFail(failNum);
        String info = gson.toJson(lists);
        System.out.println(info);
        out.write(info);
    }
    out.flush();
    out.close();
}
```

1.4&&2.3

1.4.Take a roll number, name and department name, and insert a new student record with tot_creds set to 0. Make sure you catch exceptions and report them (and test it with an incorrect department name, to see what happens when a foreign key constraint is violated, and similarly with a duplicate roll number, to see what happens when a primary key constraint is violated). Make sure also to close connections, even if there is an exception.

2.3.Create form interfaces to insert records for the student and instructor tables, with a drop-down menu for the department name, showing all valid department names. As before, exceptions should be caught and reported.

由于添加学生和添加教师两个功能要求相似，故把其放在一起实现（展示添加学生的代码，添加教师类似）

pers.dao.impl.UniversityDaoImpl

```
@Override
public int saveStudent(Student student) {
    String sql = "insert into student (`ID`, `name`, `dept_name`,
`tot_cred`) values (?, ?, ?, ?)";
    return update(sql, student.getID(), student.getName(),
student.getDeptName(), student.getTotCred());
}
```

pers.service.impl.UniversityServiceImpl

```
//返回-1则添加失败
@Override
public int insertStudent(Student student) {
    return universityDao.saveStudent(student);
}
```

pers.web.InsertServlet

```
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
    PrintWriter out = resp.getWriter();
    Gson gson = new Gson();
    String feedBack = "";
    String personType = req.getParameter("personType");
    String ID = req.getParameter("ID");
    String name = req.getParameter("name");
    String deptName = req.getParameter("deptName");
    if (personType.equals("student")){
        if (universityService.existStudentID(ID)){
            feedBack = "student ID already exists";
        }
        if (!universityService.existDeptName(deptName)){
            if(feedBack != ""){
                feedBack += " and dept_name doesn't exist";
            }else {
                feedBack += "dept_name doesn't exist";
            }
        }
        if(feedBack == "" || feedBack.length() == 0){
            Integer idx = universityService.insertStudent(new Student(ID,
name, deptName, 0));
            feedBack = idx.toString();
        }
    }
```

```
        out.write(feedBack);
        out.flush();
        out.close();
    }
```

检测主键（ID是否已经存在）以及外键（院系是否存在）

pers.dao.impl.UniversityDaoImpl

```
public Student queryStudentByID(String ID) {
        String sql = "select ID, name, dept_name as deptName, tot_cred as
totCred from student where ID = ?";
        return queryForOne(Student.class, sql, ID);
    }

    //查找院系
    @Override
    public List<String> queryDeptName() {
        String sql = "select distinct(dept_name) from department";
        List<String> resLists = new ArrayList<String>();
        List<Object[]> objLists = queryForArray(sql);
        for(Object[] objList : objLists){
            resLists.add(objList[0].toString());
        }
        return resLists;
    }
```

pers.service.impl.UniversityServiceImpl

```
    //判断ID是否唯一
    @Override
    public boolean existStudentID(String ID) {
        if (universityDao.queryStudentByID(ID) == null){
            return false;
        }else{
            return true;
        }
    }

    //判断院系是否存在
    @Override
    public boolean existDeptName(String deptName) {
        List<String> lists = universityDao.queryDeptName();
        if(lists.contains(deptName)){
            return true;
```

```
    }else{
        return false;
    }
}
```

院系下拉菜单栏：后台Servlet返回包含dept_name的List

pers.web.DeptServlet

```
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
    PrintWriter out = resp.getWriter();
    Gson gson = new Gson();
    List<String> lists = universityService.getAllDeptName();
    String info = gson.toJson(lists);
    System.out.println(info);
    out.write(info);
    out.flush();
    out.close();
}
```

2.1

2.1.Write a function createDropDown(ResultSet) that takes a ResultSet, and creates a drop-down menu from the ResultSet; the first attribute is used as the result value, and all attributes are displayed in the drop-down menu, concatenated together.

实验采用的DBUtils对JDBC的ResultSet封装了一个ResultSetHandler结果集处理类，接收ResultSet并进行相对应的处理，故前述步骤基本上都是接收一个ResultSet并形成下拉表格。

2.2

2.2.Also write a function createDropDown(String) which does the same thing, but instead of a ResultSet it takes an SQL query as a string; the second version can be used safely as long as the query does not involve any value that the user inputs, otherwise it would be vulnerable to SQL injection attacks.

前端接收一个sql语句并传递给后端进行查询（为了防止injection attacks，该功能只实现DQL语句）

pers.dao.impl.UniversityDaoImpl

```
@Override
public List<Object[]> queryBySql(String sql) {
    return queryForArray(sql);
}
```

pers.service.impl.UniversityServiceImpl

```
@Override
public List<Object[]> createDropDown(String sql) {
    return universityDao.queryBySql(sql);
}
```

pers.web.SqlServlet

```
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
    PrintWriter out = resp.getWriter();
    Gson gson = new Gson();
    String sql = req.getParameter("sql");
    List<Object[]> lists = universityService.createDropDown(sql);
    String info = gson.toJson(lists);
    System.out.println(lists);
    out.write(info);
    out.flush();
    out.close();
}
```

前端

采用vue实现，利用axios进行前后端数据传递

其中一个vue示例代码：

```
var app2 = new Vue({
    el: '#app2',
```

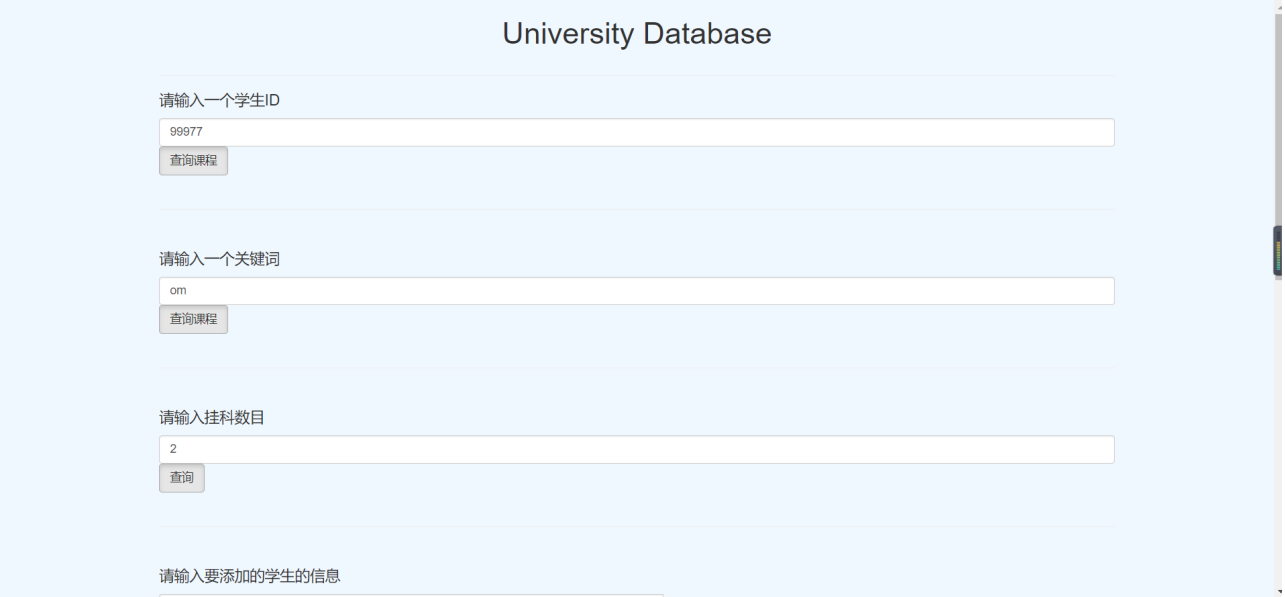
```
    data:{
        courses: [],
        show : false
    },

    methods: {
        //改为ES6写法
        courseSearch(){
            axios.get("http://localhost:8080/WebApp/courseServlet", {
                params:{
                    solve: 2,
                    keyWord: this.$refs.inputVal2.value
                }
            }).then(resp =>{
                //var myData = JSON.parse(resp.data);
                if(resp.data != null) {
                    this.courses = resp.data;
                    this.show = true;
                }else{
                    this.show = true;
                }
            }).catch(function (error){
                console.log(error);
            })
        }
    }
})
```

页面上的表格采用v-for指令生成HTML

```
<table v-if = "show" class="table table-condensed">
    <thead>
    </thead>
    <tbody>
        <tr v-for = "list in lists">
            <td v-for = "(column, index) in list">{{list[index]}}</td>
        </tr>
    </tbody>
</table>
```

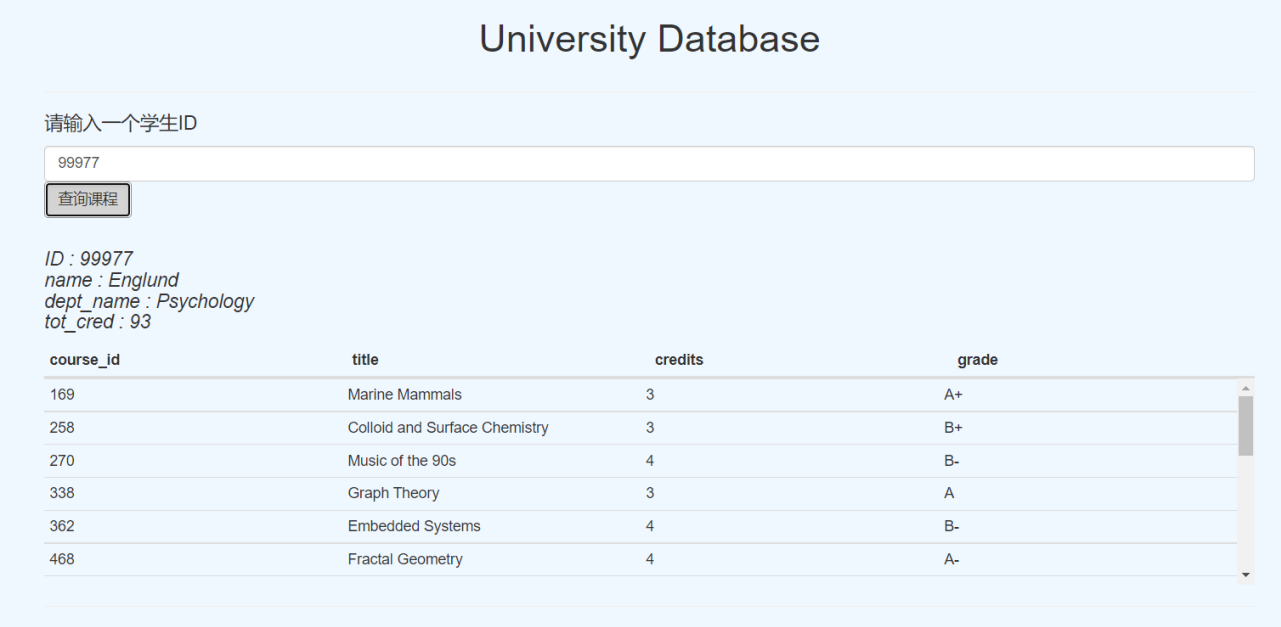
实验结果

主页面



1.1

传入一个已存在的ID



传入一个不存在的ID

请输入一个学生ID

99998

查询课程

ID :
name :
dept_name :
tot_cred :

course_id	title	credits	grade

1.2

请输入一个关键词

om

查询课程

course_id	title	credits	grade
130	Differential Geometry	Physics	3
190	Romantic Literature	Civil Eng.	3
275	Romantic Literature	Languages	3
328	Composition and Literature	Cybernetics	3
337	Differential Geometry	Statistics	3
348	Compiler Design	Elec. Eng.	3

请输入一个关键词

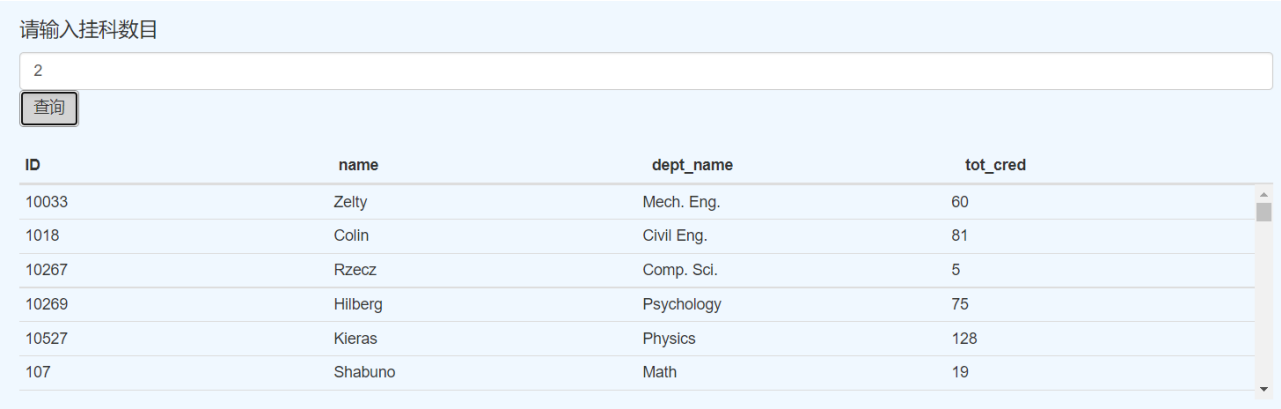
axios

查询课程

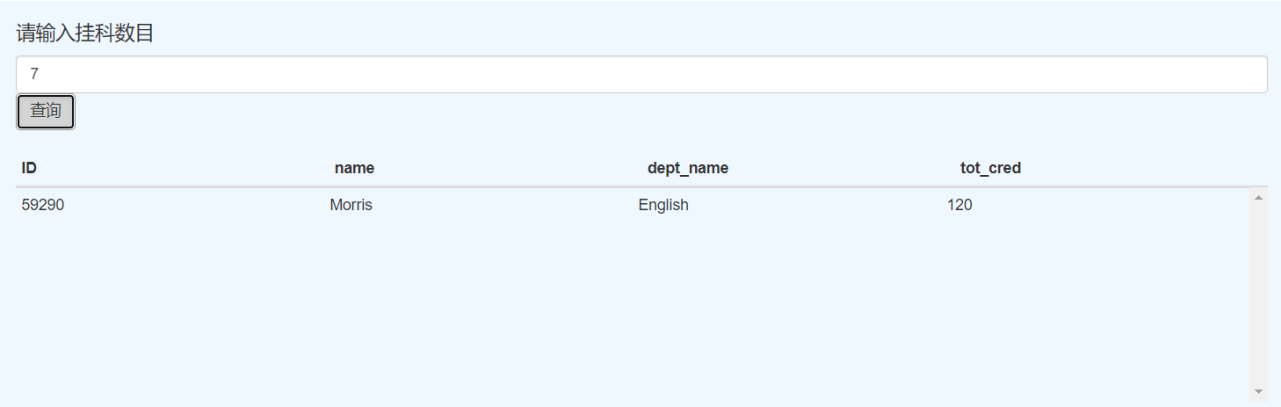
course_id	title	credits	grade

1.3

查看挂科数超过2门的学生



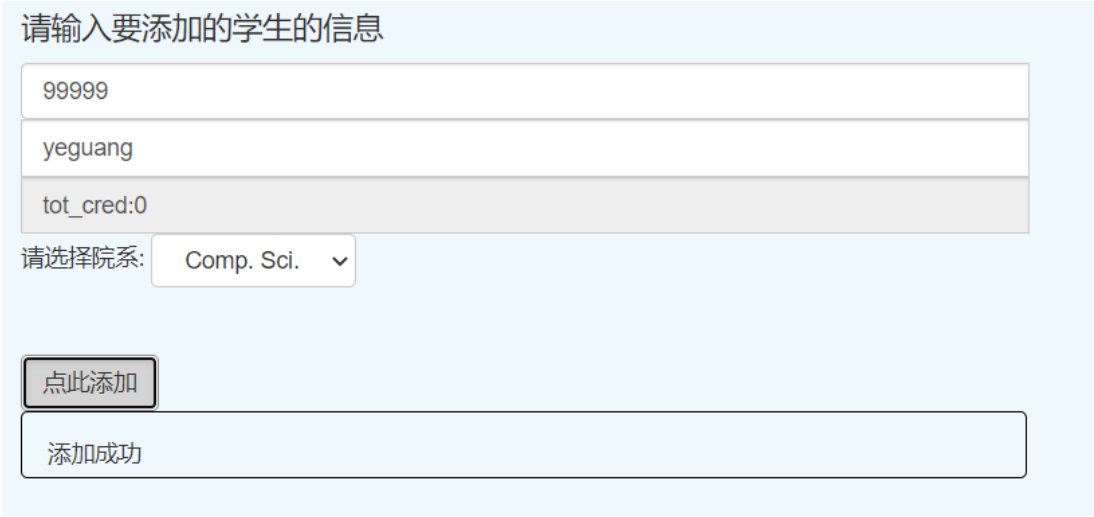
查看挂科数超过7门的学生



1.4&&2.3

院系下拉菜单栏

正常添加一个学生信息（不违反主键以及外键约束），显示添加成功



从数据库中查询确实添加成功

```
mysql> select * from student where ID = 99999;
+-------+---------+------------+----------+
| ID    | name    | dept_name  | tot_cred |
+-------+---------+------------+----------+
| 99999 | yeguang | Comp. Sci. |        0 |
+-------+---------+------------+----------+
1 row in set (0.00 sec)
```

添加一个已存在的ID，报错并显示"student ID already exists"

由于是从下拉菜单里选择，故不会出现院系不存在的情况

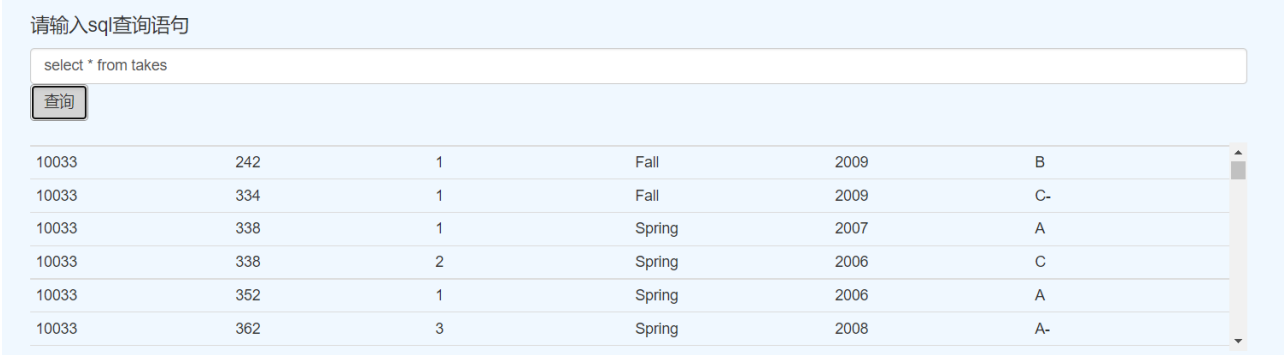
同样测试添加教师的功能如下（教师工资需大于29000，故此处统一添加为29001元）：

正常添加：

添加一个已有ID

2.2

输入DQL查询语句，查询返回



实验体会

这个实验还是花了比较多时间的，之前对servlet以及前端的一些开发工具都没有系统学习过，由于时间原因，只能从头学习一小部分用法便开始构建web application，许多功能都存在细节上的一些问题没有做好，还有很多可以改进或者优化的地方，编码的过程中也出现了许多问题没有弄清楚原理，比如vue的深度响应式原理等，导致出现了一些问题而没有办法解决，前端上也只是简单的加了一些bootstrap样式做了一点排版。总的来说，虽然还存在许多问题，但大体功能还是实现了，三周的时间从零开始也学习到许多数据库相关的开发工具，为以后的学习积累了许多知识和经验。