**实 验 报 告**

课程名称： web应用开发

实验名称： 基于 ASP.Net Core 的学生选课系统开发

院 （系）： 信控学院

专业班级： 计算机2104

2024年 6月 30 日

# 实验目的

了解ASP.NET MVC 开发框架，学习使用视图、控制器、模型实现学生选课系统，掌握HTML和CSS的前端编程语法，掌握C#与前端的混合编程，学会使用网页前端模板进行开发以及使用布局简化前端代码开发。

# 二．实验基本要求和内容

## （一）管理员课程信息维护

a)管理员登录：管理员可以通过输入自己的管理员账号和密码登录到管理员界面中。

b)可以操作（新增，修改，删除）课程信息 ：在所有课程页面中，管理员可以新增、修改、删除课程信息。但是如果有学生选择这门课程，将不允许删除。

c)当在“课程信息明细”页面，输入数据并点击“新增”按钮时，输入的课程信息将被录入系统。

d)选择已有课程，删除，修改

## （二）管理员维护学生信息

a)管理员登录：管理员可以通过输入自己的管理员账号和密码登录到管理员界面中。

b)增删改学生信息：管理员通过点击所有学生信息按钮，可以显示所有学生信息，在该页面中可以对学生信息进行增加修改和删除等。注意：学号不能修改，学号不能重复，删除学生需要连带已选课程一起删除。

## （三）学生选课

a) 学生登录：学生能够通过自己的学号以及初始密码登录到学生选课系统。

b) 课程查询：学生登录到学生选课系统中，通过点击所有课程选项进入到课程查询页面，在输入框中输入课程编号或者课程名称，即可实现查询课程的功能。

c) 选课：在所有可选课程页面中，可以通过点击每一项课程信息右边的选课按钮为自己选课。

d) 学生已选的课程查看：学生可以查看自己的所有已选课程。

e) 取消已选课程：在所有已选课程页面，学生可以点击课程信息右边的退课按钮。

# 三．实验环境

开发环境：Visio Studio 2022 + .NET 8.0 + Mircrosoft.EntityFrameworkCore

浏览器：Microsoft Edge

数据库可视化软件：Datagrip

关系型数据库软件：MySQL 8.0

# 软件设计

## 数据库设计

1. 学生信息表设计
2. StudentID 字段:varchar(20) 类型，主键，并且限制不为空，表示学生学号。
3. StudentName 字段：varchar（10）类型，表示学生姓名。
4. StudentClass 字段：varchar（10）类型，表示学生班级。
5. InitPassword 字段：varchar（10）类型，表示学生初始密码。
6. 学生选课信息表设计
7. CourseID 字段：varchar（20）类型，表示课程号。
8. StudentID 字段：varchar（20）类型，表示学号，两个字段一起作为学生选课信息表的主键。
9. 课程信息表设计
10. CourseID 字段：varchar（20）类型，主键并且限制不为空，表示课程号。
11. CourseaName 字段：varchar（10）类型，表示课程名称。
12. CourseCredit 字段：int 类型，表示课程的学分。

## 软件功能设计

1. 登录

public IActionResult LoginUser(LoginDemo loginUser)：该方法根据传入的参数的登录用户名，判断需要登录哪个角色，返回哪个页面，如果是学生还需要在数据库中检查学生编号是否存在，密码是否正确登，学生登陆成功以后返回欢迎界面；管理员登录成功返回显示所有课程的界面，否则都返回登录界面。

1. 管理员管理学生信息

public IActionResult ShowStudentInfo()：该方法将数据库中所有学生表信息都返回出来，通过模型向视图传递数据。

public IActionResult StudentInfoEdit(String id)：学生信息编辑页面，参数用于在数据库中找到对应学生的信息并显示出来。

public IActionResult SaveStudentInfo(Student student)：保存编辑学生信息的结果，并将编辑后的结果保存到数据库中。

public IActionResult DeleteStudentInfo(Student student)：删除学生信息，根据传入的参数，判断其是否选课，如果选课了，需要将所选课程一并删除，然后再将该学生删除，最后将删除结果提交数据库，方法返回到显示所有学生信息的页面中。

public IActionResult StudentInfoDelete(string Id) ：用于在删除学生页面中显示要删除学生的信息。

public IActionResult StudentInfoAdd(Student student)：添加学生信息，首先需要判断该学生是否已经在学生信息表中存在，如果不存在，才允许添加，然后将添加结果保存到数据库，方法返回到现实所有学生视图。

1. 管理员管理课程信息

public IActionResult ShowAllCourse()：显示所有课程信息的方法，将数据库中课程表的所有课程信息都作为模型返回。

public IActionResult CourseEdit(String id)：该方法用于在编辑课程信息时，显示编辑前的课程信息，该方法返回id对应的课程model的视图。

public IActionResult SaveCourse(Course course)：该方法将编辑后的信息保存到数据库中，将会返回所有课程信息页面。

public IActionResult DelateCourse(Course course)：删除课程方法，方法中首先要判断是否有学生选课，如果有学生选课，则不允许管理员删除，该方法返回课程信息页面。

public IActionResult ShowAvailableCourse(Course course)：显示可用课程的方法，该方法返回所有未被学生选修的课程信息视图。

public IActionResult AddNewCourse(Course course)：显示添加新课程页面的方法。

public IActionResult AddCourse(Course course)：添加课程的方法，该方法将新课程信息保存到数据库中，如果课程已存在或信息不完整，则返回错误信息。

1. 学生课程管理

public IActionResult Index()：显示学生主页面。

public IActionResult ShowSelectedCourse()：返回已经登录的学生选过的课程。

public IActionResult StudentWelcome()：显示学生欢迎页面 。

public IActionResult ShowAvailableCourse(）：展示某个学生所有可以选择的课程。

public IActionResult SelectCourse(Course course)：学生选课的方法，假如某个学生重复选课，该方法也会返回错误信息，成功选课以后将会返回显示所有已选课程的信息。

public IActionResult UnSelectCourse(Course course)：学生退课的方法，假如该学生没有选择这门课，该方法将会返回提示信息，成功退课以后，方法返回显示所有已选课程的信息。

# 关键源代码与说明

## （一）学生控制器

namespace LogInDemo.Controllers

{

public class StudentController : Controller

{

public IActionResult Index()

{

return View();

}

//返回的应该是登录学生已经选过的课程,而不是全部的课程

public IActionResult ShowSelectedCourse()

{

ViewData["UserName"] = HttpContext.Session.GetString("UserName");

//获取传入的学生编号

string userid = ViewData["Username"]?.ToString();

using (var db = new \_2114010313DbContext())

{

var student = db.Students.Find(userid);

//返回所有包含该学生编号的选课信息,即是该学生已经选的课程

var selectedCourses = db.Stucourse1s

.Where(sc=>sc.StudentId.Contains(userid))

.ToList();

return View(selectedCourses);

}

}

public IActionResult StudentWelcome()

{

ViewData["UserName"] = HttpContext.Session.GetString("UserName");

return View();

}

//展示某个学生所有可以选择的课程

public IActionResult ShowAvailableCourse()

{

using (var db = new \_2114010313DbContext())

{

var courses = db.Courses.ToList();

return View(courses);

}

}

public IActionResult SelectCourse(Course course)

{

String stuId = HttpContext.Session.GetString("UserName");

using(var db = new \_2114010313DbContext())

{

//如果该学生已经选过该课程,返回到查看所有可选课程的页面中

if(db.Stucourse1s.Find(stuId,course.CourseId) != null)

{

TempData["SelectCourseFailed"] = "你已经选过该课程了,不可重复选课";

return RedirectToAction("ShowAvaiableCourse","Student");

}

else

{

Stucourse1 stucourse = new Stucourse1();

stucourse.StudentId = stuId;

stucourse.CourseCredit = course.CourseCredit;

stucourse.CourseName = course.CourseName;

stucourse.CourseId = course.CourseId;

db.Stucourse1s.Add(stucourse);

db.SaveChanges();

}

return RedirectToAction("ShowSelectedCourse", "Student");

}

}

public IActionResult UnSelectCourse(Course course)

{

String stuId = HttpContext.Session.GetString("UserName");

using(var db = new \_2114010313DbContext())

{

//如果该学生已经选过该课程,才允许学生退课

if (db.Stucourse1s.Find(stuId,course.CourseId) == null)

{

TempData["UnselectCourseFailed"] = "你没有选过这门课程,不能退课";

return RedirectToAction("ShowSelectedCourse", "Student");

}

else

{

Stucourse1 stucourse = db.Stucourse1s.Find(stuId,course.CourseId);

db.Stucourse1s.Remove(stucourse);

db.SaveChanges();

return RedirectToAction("ShowSelectedCourse", "Student");

}

}

}

}

}

## （二）主页控制器

namespace LogInDemo.Controllers

{

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

public HomeController(ILogger<HomeController> logger)

{

\_logger = logger;

}

public IActionResult ShowLogin()

{

return View();

}

public IActionResult LoginUser(LoginDemo loginUser)

{

if(loginUser.username == "admin")

{

HttpContext.Session.SetString("UserName", loginUser.username);

//使用TempData记录登录的用户名,显示在登录后的界面中

TempData["UserNamemessage"] = loginUser.username;

//跳转到Course控制器的ShowAllCourse方法下面

return RedirectToAction("ShowAllCourse","Course");

}

//如果输入的username以S开头,表明现在是学生正在登陆,跳转到学生相关的页面

else if(loginUser.username.StartsWith("S") || loginUser.username.StartsWith("s"))

{

HttpContext.Session.SetString("UserName", loginUser.username);

TempData["message"] = loginUser.username;

//直接重定向到学生控制器下的显示该学生所有已选课程下.

string studentId = loginUser.username;

//使用pwd接收用户登录时候传入的密码

string pwd = loginUser.pwd;

using(var db = new \_2114010313DbContext())

{

var student = db.Students.Find(studentId);

if(student == null)

{

TempData["UserNameInvalid"] = "用户不存在,请重新输入";

return RedirectToAction("ShowLogin", "Home");

}

//判断学生用户的密码是否正确

else if(pwd != student.InitPassword)

{

TempData["PasswordInvalid"] = "用户密码错误,请重新输入";

return RedirectToAction("ShowLogin", "Home");

}

else

{

return RedirectToAction("StudentWelcome", "Student");

}

}

}

else

{

TempData["Message"] = "无效的用户名或者密码错误";

return RedirectToAction("ShowLogin","Home");

}

}

public IActionResult Index()

{

return View();

}

public IActionResult Privacy()

{

return View();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

}

## 学生信息控制器

namespace LogInDemo.Controllers

{

public class StudentInfoController : Controller

{

public IActionResult Index()

{

return View();

}

public IActionResult ShowStudentInfo()

{

using(var db = new \_2114010313DbContext())

{

List<Student> student = db.Students.ToList();

return View(student);

}

}

public IActionResult StudentInfoEdit(String id)

{

if(string.IsNullOrWhiteSpace(id))

{

return View();

}

using(var db = new \_2114010313DbContext())

{

var student = db.Students.Find(id);

return View(student);

}

}

public IActionResult SaveStudentInfo(Student student)

{

using(var db = new \_2114010313DbContext())

{

var model = db.Students.Find(student.StudentId);

if (model == null)

{

model = new Student();

model.StudentId = student.StudentId;

model.StudentName = student.StudentName;

model.StudentClass = student.StudentClass;

db.Students.Add(model);

}

else

{

model.StudentName = student.StudentName;

model.StudentClass = student.StudentClass;

}

db.SaveChanges();

}

return RedirectToAction("ShowStudentInfo", "StudentInfo");

}

public IActionResult DeleteStudentInfo(Student student)

{

using(var db = new \_2114010313DbContext())

{

//删除课程需要连带所有已选课程一起删除

var model = db.Students.Find(student.StudentId);

var allCourses = db.Courses.ToList();

//遍历所有课程,只要能找到要删除的学生有选课记录,就删除

foreach(var course in allCourses)

{

var selectedCourses = db.Stucourse1s.Find(student.StudentId, course.CourseId);

// 该变量不为空,说明学生选了这门课

if(selectedCourses != null)

{

db.Stucourse1s.Remove(selectedCourses);

}

}

if(model == null)

{

db.SaveChanges();

return RedirectToAction("ShowStudentInfo", "StudentInfo");

}

else

{

db.Students.Remove(model);

}

db.SaveChanges();

}

return RedirectToAction("ShowStudentInfo", "StudentInfo");

}

public IActionResult StudentInfoDelete(string Id)

{

if(string.IsNullOrEmpty(Id))

{

return View();

}

using(var db = new \_2114010313DbContext())

{

var student = db.Students.Find(Id);

return View(student);

}

}

public IActionResult AddStudentInfo(Student student)

{

return View(student);

}

public IActionResult StudentInfoAdd(Student student)

{

using(var db = new \_2114010313DbContext())

{

if(db.Students.Find(student.StudentId) != null)

{

TempData["StudentInfoAddFailed"] = "该学生已经存在,请重试";

return RedirectToAction("StudentInfoAdd", "StudentInfo");

}

else

{

db.Students.Add(student);

db.SaveChanges();

return RedirectToAction("ShowStudentInfo", "StudentInfo");

}

}

}

}

}

## 课程信息控制器

using LogInDemo.Models;

using Microsoft.AspNetCore.Mvc;

using NuGet.Protocol.Plugins;

namespace LogInDemo.Controllers

{

public class StudentInfoController : Controller

{

public IActionResult Index()

{

return View();

}

public IActionResult ShowStudentInfo()

{

using(var db = new \_2114010313DbContext())

{

List<Student> student = db.Students.ToList();

return View(student);

}

}

public IActionResult StudentInfoEdit(String id)

{

if(string.IsNullOrWhiteSpace(id))

{

return View();

}

using(var db = new \_2114010313DbContext())

{

var student = db.Students.Find(id);

return View(student);

}

}

public IActionResult SaveStudentInfo(Student student)

{

using(var db = new \_2114010313DbContext())

{

var model = db.Students.Find(student.StudentId);

if (model == null)

{

model = new Student();

model.StudentId = student.StudentId;

model.StudentName = student.StudentName;

model.StudentClass = student.StudentClass;

db.Students.Add(model);

}

else

{

model.StudentName = student.StudentName;

model.StudentClass = student.StudentClass;

}

db.SaveChanges();

}

return RedirectToAction("ShowStudentInfo", "StudentInfo");

}

public IActionResult DeleteStudentInfo(Student student)

{

using(var db = new \_2114010313DbContext())

{

//删除课程需要连带所有已选课程一起删除

var model = db.Students.Find(student.StudentId);

var allCourses = db.Courses.ToList();

//遍历所有课程,只要能找到要删除的学生有选课记录,就删除

foreach(var course in allCourses)

{

var selectedCourses = db.Stucourse1s.Find(student.StudentId, course.CourseId);

// 该变量不为空,说明学生选了这门课

if(selectedCourses != null)

{

db.Stucourse1s.Remove(selectedCourses);

}

}

if(model == null)

{

db.SaveChanges();

return RedirectToAction("ShowStudentInfo", "StudentInfo");

}

else

{

db.Students.Remove(model);

}

db.SaveChanges();

}

return RedirectToAction("ShowStudentInfo", "StudentInfo");

}

public IActionResult StudentInfoDelete(string Id)

{

if(string.IsNullOrEmpty(Id))

{

return View();

}

using(var db = new \_2114010313DbContext())

{

var student = db.Students.Find(Id);

return View(student);

}

}

public IActionResult AddStudentInfo(Student student)

{

return View(student);

}

public IActionResult StudentInfoAdd(Student student)

{

using(var db = new \_2114010313DbContext())

{

if(db.Students.Find(student.StudentId) != null)

{

TempData["StudentInfoAddFailed"] = "该学生已经存在,请重试";

return RedirectToAction("StudentInfoAdd", "StudentInfo");

}

else

{

db.Students.Add(student);

db.SaveChanges();

return RedirectToAction("ShowStudentInfo", "StudentInfo");

}

}

}

}

}

## 前端页面设计

@using LogInDemo.Models;

@model Student

@{

Layout = "\_Layout2";

}

<header class="panel-heading">

<div class="panel-actions">

</div>

<h2 class="panel-title">学生添加详细页面</h2>

</header>

<div class="panel-body">

<form class="form-horizontal form-bordered" method="get" action="@Url.Action("StudentInfoAdd","StudentInfo")">

<div class="form-group">

<label class="col-md-3 control-label">学生ID</label>

<div class="col-md-6">

<input type="text" value="@Model?.StudentId" name="StudentID" class="form-control">

</div>

</div>

<div class="form-group">

<label class="col-md-3 control-label" for="inputDefault">学生名称</label>

<div class="col-md-6">

<input type="text" class="form-control" value="@Model?.StudentName" name="StudentName" id="inputDefault">

</div>

</div>

<div class="form-group">

<label class="col-md-3 control-label" for="inputDefault">学生班级</label>

<div class="col-md-6">

<input type="text" class="form-control" value="@Model?.StudentClass" name="StudentClass" id="inputDefault">

</div>

</div>

<div class="form-group">

<label class="col-md-3 control-label" for="inputDefault">学生初始密码</label>

<div class="col-md-6">

<input type="text" class="form-control" value="@Model?.InitPassword" name="InitPassword" id="inputDefault">

</div>

</div>

<div class="form-group">

<label class="col-md-3 control-label"></label>

<div class="col-sm-8">

<div class="row">

<div class="col-sm-2">

<button type="submit" class="mb-xs mt-xs mr-xs btn btn-primary">保存</button>

</div>

<div class="visible-xs mb-md"></div>

<div class="col-sm-3">

<button type="button" onclick="location.href='/Course/ShowAllCourse'" class="mb-xs mt-xs mr-xs btn btn-default">返回</button>

</div>

</div>

</div>

</div>

</form>

</div>

# 实验结果（包含产生结果用到的测试用例和过程说明）

## 测试登录模块

通过在Username框中输入admin字符串，系统将登入管理员界面，如图所示：登入以后将会显示所有课程信息。

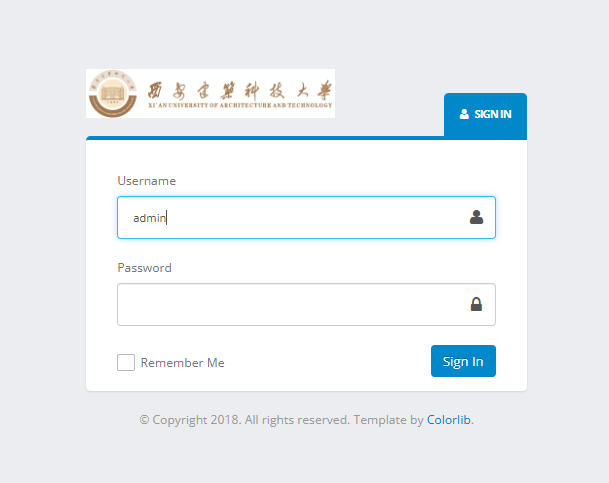


图 1 登录界面

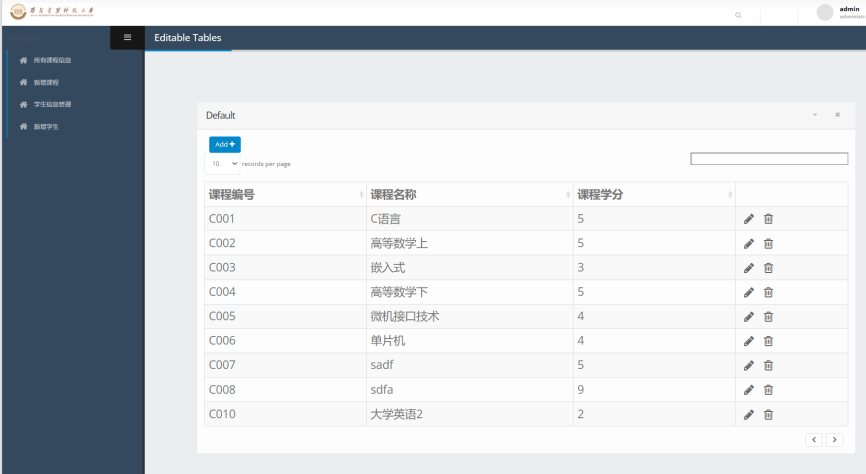


图 2 管理员登录成功界面

学生登入需要特定的学号以及学生的初始密码，如果登录的账号不存在或者已经删除，登录将会提示用户不存在的报错信息，然后浏览器重定向到登陆入口处，如图所示：

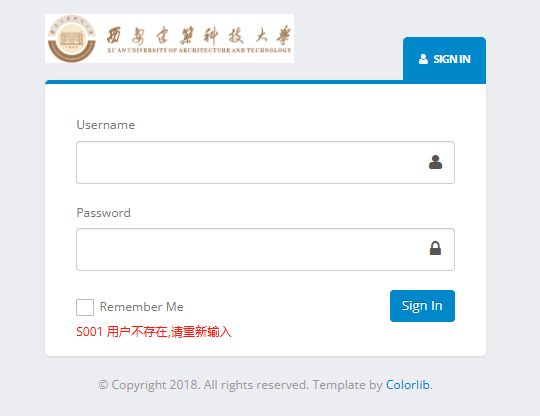


图 3 测试不存在的学生账号登录

如果登录学生输入的密码不正确，将会提示密码错误信息，浏览器重定向到登陆入口，如图所示：

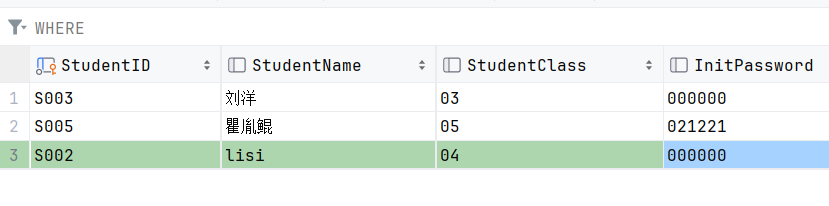


图 4 测试错误密码登录



图 5 密码错误提示信息



图 6 学生成功登录

## 测试管理员功能

1. 测试增加课程信息

点击显示所有课程信息页面左上角的添加按钮，将会跳转到添加课程信息的页面，在其中输入想要添加的课程信息，点击确认以后，将会把课程信息添加到数据库中。

假如此时添加的课程编号已经被其他课程占用，将会输出该课程编号已经被占用的提示信息，如图所示：

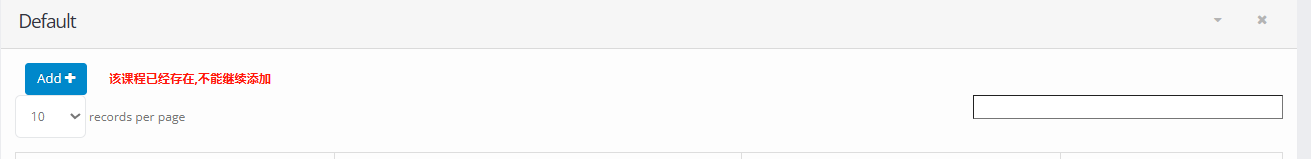


图 7 测试添加课程信息



图 8 所有课程信息界面点击添加课程

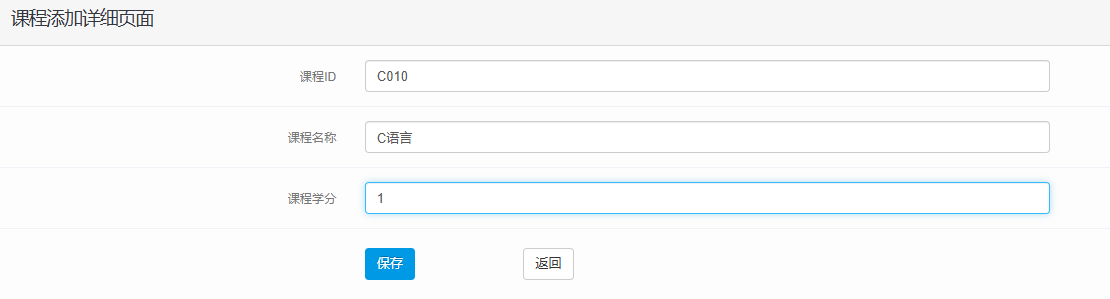


图 9 输入课程详细信息

如果添加课程信息的时候输入的信息不完整，也会报错，如图所示：

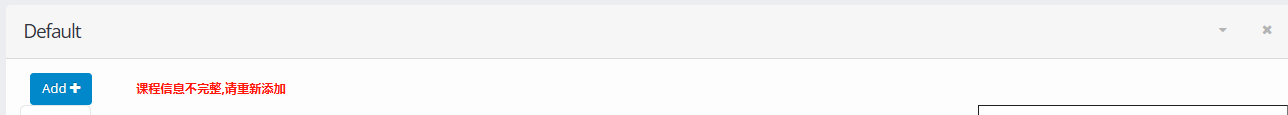


图 10 测试输入课程信息不完整提示信息

添加课程成功，如图所示，在显示所有课程页面可以看到多了一项课程信息：

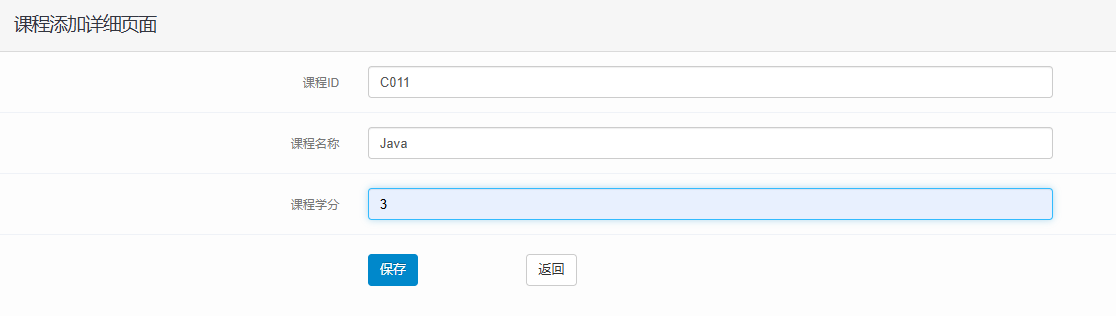


图 11 正确输入课程信息



图 12 添加课程以后查看所有课程

1. 测试删除课程信息

如果管理员想要删除一个有学生选过的课程信息，系统将会提示不允许删除，返回课程管理界面，如图所示：



当管理员删除一个没有学生选课的课程信息时，才会成功删除，删除后将会重新跳转到课程信息管理界面，这时可以看到已经不存在删除的课程，在数据库中同样不存在该课程信息，如下图所示：



图 13 测试删除课程信息



图 14 删除成功以后显示所有课程页面



图 15 删除成功以后数据库课程表信息

1. 测试增加学生信息

在学生信息管理模块中，点击页面左上角增加按钮，即可跳转到新增学生页面中，如图所示，在该页面中输入学生基本信息，注意学生ID同样不允许重复，并且所有学生信息必须完整，否则将无法添加到数据库中并会提示报错信息。

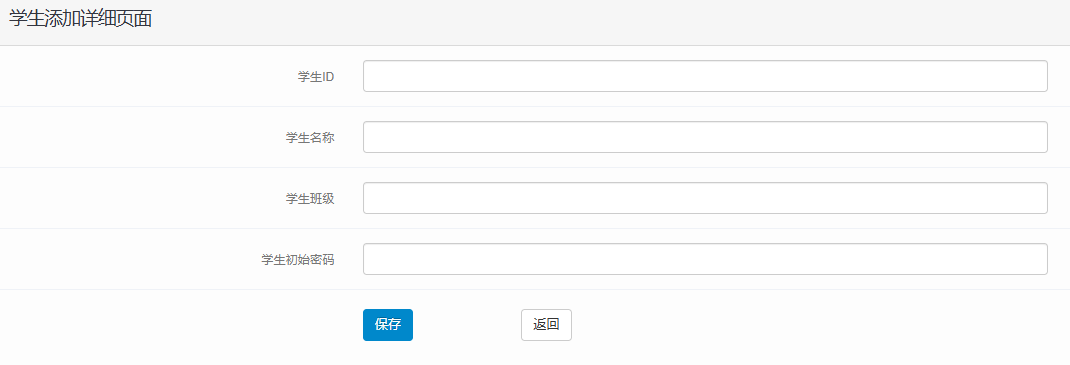


图 16 添加学生信息页面

成功添加学生信息以后，系统将会跳转到显示所有学生信息页面，在该页面中可以看到新添加的学生信息，如图所示：



图 17 成功添加学生信息

1. 测试修改学生信息

点击每个学生旁边铅笔形状的图标，将会跳转到该学生的编辑界面，在该界面中可以对学生信息进行修改，但是不能修改学生学号。修改S001学生信息如下图所示：

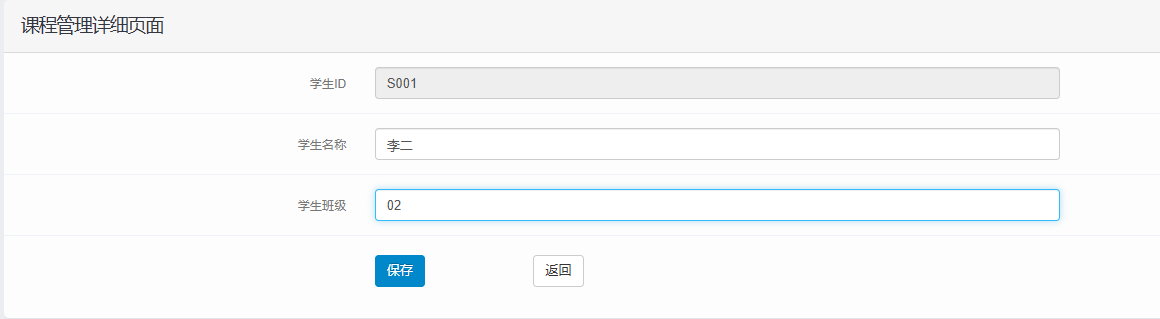


图 18 测试修改学生信息

点击确认以后，系统将跳转到显示所有学生信息方法里面，在这里，能够看到学生编号为S001的学生信息已经被修改，如图所示：

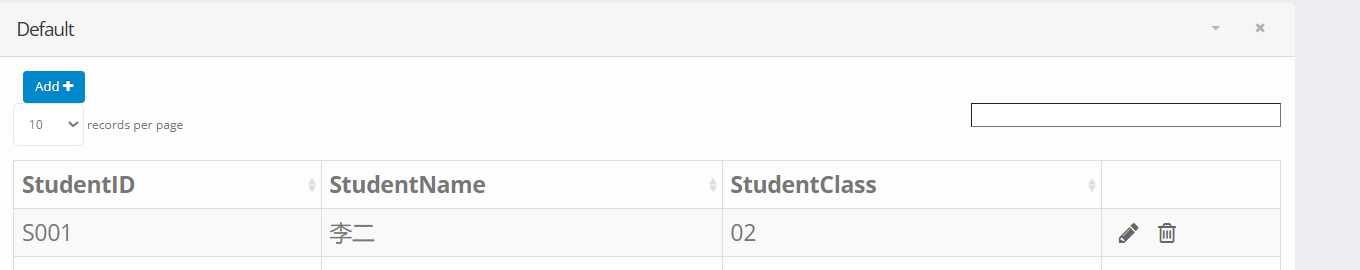


图 19 测试修改学生信息

1. 测试删除学生信息

在学生信息管理页面，点击每个学生项右边的删除图标，将会跳转到删除页面，如图所示：

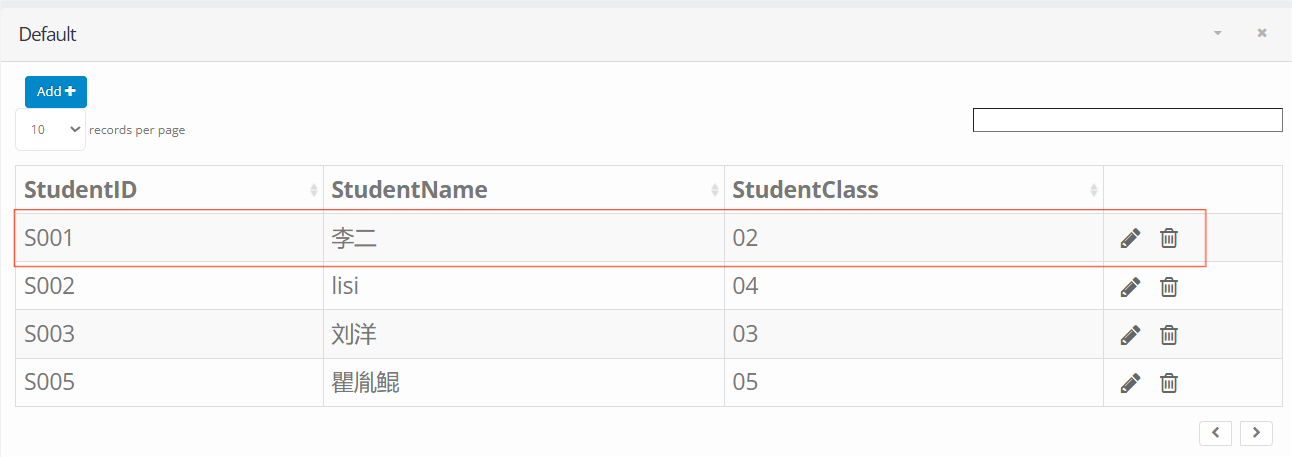


图 20 删除学生信息



图 21 删除学生信息详细界面

当点击确认删除以后，系统会检查该学生是否有选课记录，如果存在选课记录，则需要将学生所有选课记录都删除，然后再将该学生删除，如图所示，此时所有学生信息中已经没有编号为S001的学生信息，说明删除成功。

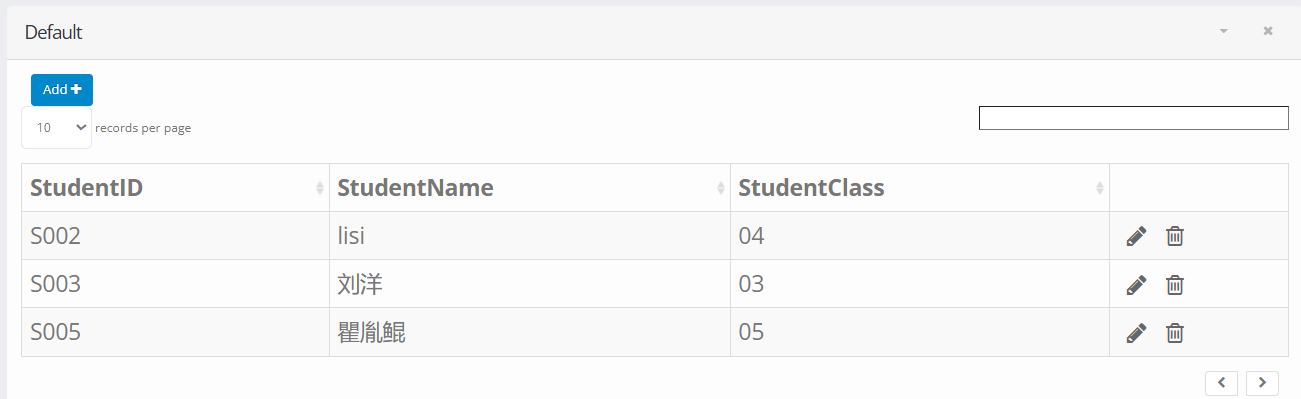


图 22 学生删除成功

## 测试学生功能

1. 测试学生查看所有可选课程

在学生主页面点击左侧Sidebar中的所有可选课程按钮，将会出现此时数据库中的全部课程，如图所示：



图 23 测试查看所有课程信息

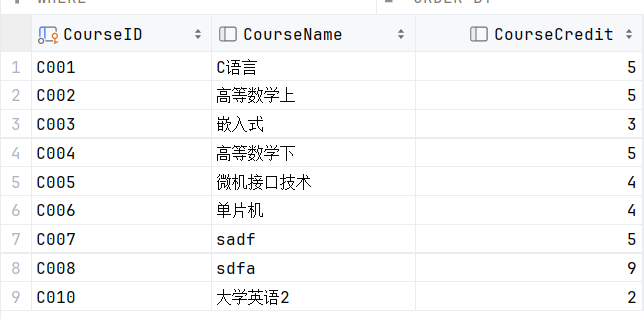


图 24 数据库中的课程信息

1. 测试学生查找课程

在选课页面中，学生可以直接通过页面右上方的搜索框进行课程信息搜索，同样该搜索框也支持模糊搜索，如图所示：



图 25 测试精确搜索功能



图 26 测试模糊搜索功能

1. 测试学生选课功能

在所有课程信息页面中，通过点击每个课程项右边的选课按钮，可以给自己选课，如图所示，这里选择嵌入式课程进行测试，点击以后将会跳转到学生所有已选课程页面，如图所示：



图 27 测试学生选课功能



图 28 选课成功后跳转页面

这是，通过数据库软件也可以看到该学生有一条选课记录，如图所示：



图 29 选课成功以后数据库信息

1. 测试学生取消选课功能

在学生的所有已选课程列表中，点击每一个课程选项旁边的退课按钮，就可以给每个学生退课，如图所示：



图 30 测试退课功能

点击退课以后，再查看该学生的所有已选课程界面，可以看到该学生的选课信息已经不存在嵌入式这门课程，如图所示，同时数据库中也不存在该学生的选课记录：

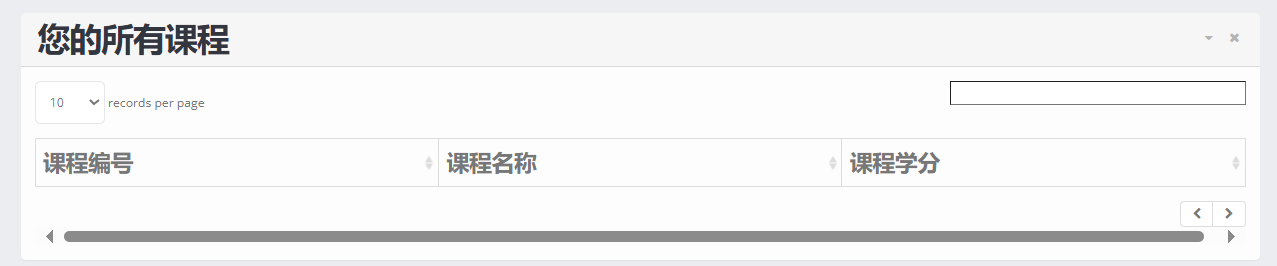


图 31 成功退课以后该学生的所有已选课程信息

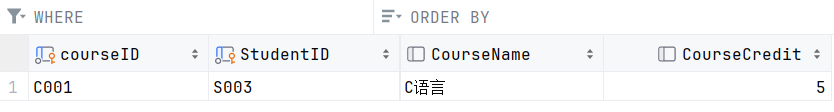


图 32 数据库学生选课表

# 存在的问题与收获

## （一）存在的问题与解决方案

在答辩过程中，我学习到自己的学生选课管理系统还有一些小问题，比如在学生选课过程中，没有处理同一个学生重复选课的异常情况，在发现问题以后，立马在课程控制器选课方法中增加了对于多次选课的判断，判断逻辑为假如学生选过课，也就说明db.Stucourse1s.Find(stuId,course.CourseId) != null 成立，这时页面将会重定向到显示学生所有可选课程的页面中，并且显示一个提示信息，如图所示：

public IActionResult SelectCourse(Course course)

{

String stuId = HttpContext.Session.GetString("UserName");

using(var db = new \_2114010313DbContext())

{

//如果该学生已经选过该课程,返回到查看所有可选课程的页面中

if(db.Stucourse1s.Find(stuId,course.CourseId) != null)

{

TempData["SelectCourseFailed"] = "你已经选过该课程了,不可重复选课";

return RedirectToAction("ShowAvailableCourse","Student");

}

else

{

Stucourse1 stucourse = new Stucourse1();

stucourse.StudentId = stuId;

stucourse.CourseCredit = course.CourseCredit;

stucourse.CourseName = course.CourseName;

stucourse.CourseId = course.CourseId;

db.Stucourse1s.Add(stucourse);

db.SaveChanges();

}

return RedirectToAction("ShowSelectedCourse", "Student");

}

}

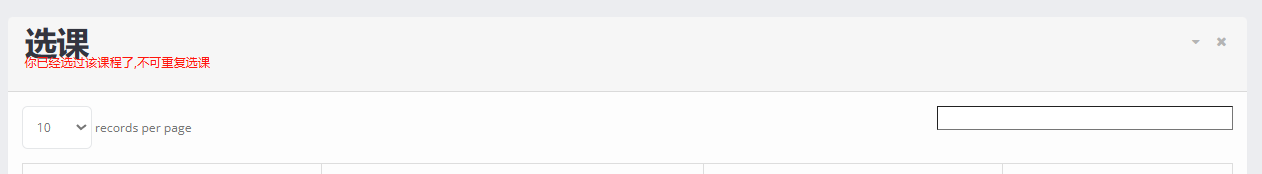


图 33 解决重复选课BUG

## （二）收获

通过这次实验，我掌握了ASP.NET Core MVC编程框架，了解了如何导入数据库，如何将前端代码与后端数据库相连接，掌握了前后端通信的几种方式。这次实验从数据库设计开始，然后自己设计页面，导入模型，编写控制器代码，深刻的理解了MVC框架的实现原理，复习了C#编程语言。同时提高了自己源代码阅读理解能力以及代码调试能力。

当然，在实验过程中，我也意识到自己存在很多不足的地方，比如只能够实现实验的基本功能，没有在此基础上进行功能完善和代码优化，目前看来，项目还能够优化登录功能，实现多次密码错误账号锁定，登录时输入验证码；学生信息或者课程信息可以从表格文件中直接读取，省去每一项添加时间；批量删除学生或者课程信息等功能，希望在后续的软件版本迭代中改进。