IT2030 Assignment 14 Matt Brown

**Startup.cs**

namespace ClassSchedule

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddControllersWithViews();

services.AddMemoryCache();

services.AddSession();

services.AddTransient<IClassScheduleUnitOfWork, ClassScheduleUnitOfWork>();

services.AddTransient(typeof(IRepository<>), typeof(Repository<>));

services.AddHttpContextAccessor();

services.AddDbContext<ClassScheduleContext>(options =>

options.UseSqlServer(Configuration.GetConnectionString("ClassScheduleContext")));

}

// Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app)

{

app.UseDeveloperExceptionPage();

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseSession();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

});

}

}

}

**IClassScheduleUnitOfWork.cs**

namespace ClassSchedule.Models

{

public interface IClassScheduleUnitOfWork

{

public IRepository<Day> Days { get; }

public IRepository<Teacher> Teachers { get; }

public IRepository<Class> Classes { get; }

public void Save();

}

}

**ClassScheduleUnitOfWork.cs**

namespace ClassSchedule.Models

{

public class ClassScheduleUnitOfWork : IClassScheduleUnitOfWork

{

private ClassScheduleContext context { get; set; }

public ClassScheduleUnitOfWork(ClassScheduleContext ctx) => context = ctx;

private IRepository<Day> dayData;

public IRepository<Day> Days {

get {

if (dayData == null)

dayData = new Repository<Day>(context);

return dayData;

}

}

private IRepository<Teacher> teacherData;

public IRepository<Teacher> Teachers {

get {

if (teacherData == null)

teacherData = new Repository<Teacher>(context);

return teacherData;

}

}

private IRepository<Class> classData;

public IRepository<Class> Classes {

get {

if (classData == null)

classData = new Repository<Class>(context);

return classData;

}

}

public void Save() => context.SaveChanges();

}

}

**ClassController.cs**

namespace ClassSchedule.Controllers

{

public class ClassController : Controller

{

private IHttpContextAccessor http { get; set; }

private IClassScheduleUnitOfWork data { get; set; }

public ClassController(IClassScheduleUnitOfWork rep, IHttpContextAccessor ctx)

{

data = rep;

http = ctx;

}

public RedirectToActionResult Index()

{

// clear session and navigate to list of classes

http.HttpContext.Session.Remove("dayid");

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

}

[HttpGet]

public ViewResult Add()

{

this.LoadViewBag("Add");

return View();

}

[HttpGet]

public ViewResult Edit(int id)

{

this.LoadViewBag("Edit");

var c = this.GetClass(id);

return View("Add", c);

}

[HttpPost]

public IActionResult Add(Class c)

{

string operation = (c.ClassId == 0) ? "Add" : "Edit";

if (ModelState.IsValid) {

if (c.ClassId == 0)

data.Classes.Insert(c);

else

data.Classes.Update(c);

data.Classes.Save();

string verb = (operation == "Add") ? "added" : "updated";

TempData["msg"] = $"{c.Title} {verb}";

return this.GoToClasses();

}

else {

this.LoadViewBag(operation);

return View();

}

}

[HttpGet]

public ViewResult Delete(int id)

{

var c = this.GetClass(id);

ViewBag.DayId = http.HttpContext.Session.GetInt32("dayid");

return View(c);

}

[HttpPost]

public RedirectToActionResult Delete(Class c)

{

c = data.Classes.Get(c.ClassId); // so can get class title for notification message

data.Classes.Delete(c);

data.Classes.Save();

TempData["msg"] = $"{c.Title} deleted";

return this.GoToClasses();

}

// private helper methods

private Class GetClass(int id)

{

var classOptions = new QueryOptions<Class> {

Includes = "Teacher, Day",

Where = c => c.ClassId == id

};

return data.Classes.Get(classOptions);

}

private void LoadViewBag(string operation)

{

ViewBag.Days = data.Days.List(new QueryOptions<Day> {

OrderBy = d => d.DayId

});

ViewBag.Teachers = data.Teachers.List(new QueryOptions<Teacher> {

OrderBy = t => t.LastName

});

ViewBag.Operation = operation;

ViewBag.DayId = http.HttpContext.Session.GetInt32("dayid");

}

private RedirectToActionResult GoToClasses()

{

// if session has a value for day id, add to id route segment when redirecting

if (http.HttpContext.Session.GetInt32("dayid").HasValue)

return RedirectToAction("Index", "Home", new { id = http.HttpContext.Session.GetInt32("dayid") });

else

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

}

}

}

**HomeController.cs**

namespace ClassSchedule.Controllers

{

public class HomeController : Controller

{

private IHttpContextAccessor http { get; set; }

private IClassScheduleUnitOfWork data { get; set; }

public HomeController(IClassScheduleUnitOfWork unit, IHttpContextAccessor ctx)

{

data = unit;

http = ctx;

}

public ViewResult Index(int id)

{

// if day id passed to action method, store in session

if (id > 0) {

http.HttpContext.Session.SetInt32("dayid", id);

}

// options for Days query

var dayOptions = new QueryOptions<Day> {

OrderBy = d => d.DayId

};

// options for Classes query

var classOptions = new QueryOptions<Class> {

Includes = "Teacher, Day"

};

// order by day if no day id. Otherwise, filter by day and order by time.

if (id == 0) {

classOptions.OrderBy = c => c.DayId;

}

else {

classOptions.Where = c => c.DayId == id;

classOptions.OrderBy = c => c.MilitaryTime;

}

// execute queries

ViewBag.Days = data.Days.List(dayOptions);

return View(data.Classes.List(classOptions));

}

}

}

**TeacherController.cs**

namespace ClassSchedule.Controllers

{

public class TeacherController : Controller

{

private IRepository<Teacher> teachers { get; set; }

public TeacherController(IRepository<Teacher> rep) => teachers = rep;

public ViewResult Index()

{

var options = new QueryOptions<Teacher> {

OrderBy = t => t.LastName

};

return View(teachers.List(options));

}

[HttpGet]

public ViewResult Add() => View();

[HttpPost]

public IActionResult Add(Teacher teacher)

{

if (ModelState.IsValid) {

teachers.Insert(teacher);

teachers.Save();

TempData["msg"] = $"{teacher.FullName} added to list of teachers";

return RedirectToAction("Index");

}

else{

return View(teacher);

}

}

[HttpGet]

public IActionResult Delete(int id)

{

return View(teachers.Get(id));

}

[HttpPost]

public RedirectToActionResult Delete(Teacher teacher)

{

teacher = teachers.Get(teacher.TeacherId); // so can get teacher name for notification message

teachers.Delete(teacher);

teachers.Save();

TempData["msg"] = $"{teacher.FullName} removed from list of teachers";

return RedirectToAction("Index");

}

}

}

**HomeControllerTests.cs**

namespace ClassScheduleTests

{

public class HomeControllerTests

{

[Fact]

public void IndexActionMethod\_ReturnsAViewResult()

{

var unit = new Mock<IClassScheduleUnitOfWork>();

var classes = new Mock<IRepository<Class>>();

var days = new Mock<IRepository<Day>>();

unit.Setup(r => r.Classes).Returns(classes.Object);

unit.Setup(r => r.Days).Returns(days.Object);

var http = new Mock<IHttpContextAccessor>();

var controller = new HomeController(unit.Object, http.Object);

var result = controller.Index(0);

Assert.IsType<ViewResult>(result);

}

}

}

**TeacherControllerTests.cs**

namespace ClassScheduleTests

{

public class TeacherControllerTests

{

[Fact]

public void IndexActionMethod\_ReturnsAViewResult()

{

var rep = new Mock<IRepository<Teacher>>();

var controller = new TeacherController(rep.Object);

var result = controller.Index();

Assert.IsType<ViewResult>(result);

}

}

}