using AutoMapper;

using Barton.WebPortal.Controllers;

using Barton.WebPortal.Controllers.Entities;

using Barton.WebPortal.Controllers.Utility;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using Moq;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Threading;

using System.Threading.Tasks;

using Xunit;

namespace Barton.UnitTest

{

public class ClassesControllerTest

{

[Fact]

public void Get\_Return\_OK()

{

// mock entities

var userTypeMock = new Mock<DbSet<UserType>>();

UserType usertype1 = new UserType { Id = 1, Title = "Student" };

UserType usertype2 = new UserType { Id = 2, Title = "Parent" };

UserType usertype3 = new UserType { Id = 3, Title = "Teacher" };

UserType usertype4 = new UserType { Id = 4, Title = "TeacherAssistant" };

userTypeMock.AddList(usertype1, usertype2, usertype3, usertype4);

var userMock = new Mock<DbSet<User>>();

User user1 = new User { Id = 1, FirstName = "Student", LastName = "LastName", RoleId = 1, Status = 1, Deleted = 0 };

User user2 = new User { Id = 2, FirstName = "Parent", LastName = "LastName", RoleId = 2, Status = 1, Deleted = 0 };

User user3 = new User { Id = 3, FirstName = "Teacher", LastName = "LastName", RoleId = 3, Status = 1, Deleted = 0 };

User user4 = new User { Id = 4, FirstName = "TA", LastName = "LastName", RoleId = 4, Status = 1, Deleted = 0 };

userMock.AddList(user1, user2, user3, user4);

var courseMock = new Mock<DbSet<Course>>();

Course course1 = new Course { Id = 1, Title = "Test Course", TaId = 4, TeacherId = 3 };

courseMock.AddList(course1);

var classMock = new Mock<DbSet<Classes>>();

var class1 = new Classes

{

Id = 11,

Title = "Test Class",

Honor = "Gold",

Score = 86,

Level = "Basic",

Time = DateTime.Now.UTCSeconds(),

CourseId = 1,

TaId = 4,

TeacherId = 3,

Status = 1,

Deleted = 0,

Course = course1,

Ta = user4,

Teacher = user3

};

classMock.AddList(class1);

var classStudentMock = new Mock<DbSet<ClassStudent>>();

var classstudent1 = new ClassStudent { Id = 111, ClassId = 11, StudentId = 1 };

classStudentMock.AddList(classstudent1);

// mock DbContext

var dbMock = new Mock<MyDBContext>();

dbMock.Setup(x => x.UserTypes).Returns(userTypeMock.Object);

dbMock.Setup(x => x.Users).Returns(userMock.Object);

dbMock.Setup(x => x.Courses).Returns(courseMock.Object);

dbMock.Setup(x => x.Classes).Returns(classMock.Object);

dbMock.Setup(x => x.ClassStudents).Returns(classStudentMock.Object);

ClassesController classController = new ClassesController(dbMock.Object);

classController.ControllerContext = new ControllerContext();

classController.ControllerContext.HttpContext = new DefaultHttpContext();

classController.ControllerContext.HttpContext.Request.Headers.Add("userid", "1");

IActionResult result = classController.Get();

Assert.IsType<OkObjectResult>(result);

}

[Fact]

public void Get\_Return\_UserIdMissing()

{

// mock entities

var userTypeMock = new Mock<DbSet<UserType>>();

userTypeMock.AsQuerable<UserType>();

var userMock = new Mock<DbSet<User>>();

userMock.AsQuerable<UserType>();

var courseMock = new Mock<DbSet<Course>>();

courseMock.AsQuerable<Course>();

var classMock = new Mock<DbSet<Classes>>();

classMock.AsQuerable<Classes>();

var classStudentMock = new Mock<DbSet<ClassStudent>>();

classStudentMock.AsQuerable<ClassStudent>();

// mock DbContext

var dbMock = new Mock<MyDBContext>();

dbMock.Setup(x => x.UserTypes).Returns(userTypeMock.Object);

dbMock.Setup(x => x.Users).Returns(userMock.Object);

dbMock.Setup(x => x.Courses).Returns(courseMock.Object);

dbMock.Setup(x => x.Classes).Returns(classMock.Object);

dbMock.Setup(x => x.ClassStudents).Returns(classStudentMock.Object);

ClassesController classController = new ClassesController(dbMock.Object);

classController.ControllerContext = new ControllerContext();

classController.ControllerContext.HttpContext = new DefaultHttpContext();

// do not setup userid in header

//classController.ControllerContext.HttpContext.Request.Headers.Add("userid", "1");

IActionResult result = classController.Get();

Assert.IsType<BadRequestObjectResult>(result);

}

[Fact]

public void Get\_Return\_Failed()

{

// mock entities

var userTypeMock = new Mock<DbSet<UserType>>();

userTypeMock.AsQuerable<UserType>();

var userMock = new Mock<DbSet<User>>();

userMock.AsQuerable<UserType>();

var courseMock = new Mock<DbSet<Course>>();

courseMock.AsQuerable<Course>();

var classMock = new Mock<DbSet<Classes>>();

classMock.AsQuerable<Classes>();

var classStudentMock = new Mock<DbSet<ClassStudent>>();

classStudentMock.AsQuerable<ClassStudent>();

// mock DbContext

var dbMock = new Mock<MyDBContext>();

dbMock.Setup(x => x.UserTypes).Returns(userTypeMock.Object);

dbMock.Setup(x => x.Users).Returns(userMock.Object);

dbMock.Setup(x => x.Courses).Returns(courseMock.Object);

dbMock.Setup(x => x.Classes).Returns(classMock.Object);

dbMock.Setup(x => x.ClassStudents).Returns(classStudentMock.Object);

ClassesController classController = new ClassesController(dbMock.Object);

// no userid setting

IActionResult result = classController.Get();

Assert.IsType<BadRequestObjectResult>(result);

}

[Fact]

public void CreateClass\_Return\_OK()

{

// mock entities

var userTypeMock = new Mock<DbSet<UserType>>();

userTypeMock.AsQuerable<UserType>();

var userMock = new Mock<DbSet<User>>();

userMock.AsQuerable<UserType>();

var courseMock = new Mock<DbSet<Course>>();

courseMock.AsQuerable<Course>();

var classMock = new Mock<DbSet<Classes>>();

classMock.AsQuerable<Classes>();

var classStudentMock = new Mock<DbSet<ClassStudent>>();

classStudentMock.AsQuerable<ClassStudent>();

// mock DbContext

var dbMock = new Mock<MyDBContext>();

dbMock.Setup(x => x.UserTypes).Returns(userTypeMock.Object);

dbMock.Setup(x => x.Users).Returns(userMock.Object);

dbMock.Setup(x => x.Courses).Returns(courseMock.Object);

dbMock.Setup(x => x.Classes).Returns(classMock.Object);

dbMock.Setup(x => x.ClassStudents).Returns(classStudentMock.Object);

ClassCreateModel classModel = new ClassCreateModel

{

Id = 0,

CourseId = 1,

Description = "Test",

Honor = "Test",

Level = "Test",

Status = "Left",

TaId = 4,

TeacherId = 3,

Time = DateTime.Now.UTCSeconds(),

Title = "New Class",

StudentList = new List<int> { 1, 2 }

};

ClassesController classController = new ClassesController(dbMock.Object);

IActionResult result = classController.CreateClass(classModel);

Assert.IsType<OkResult>(result);

}

[Fact]

public void CreateClass\_Return\_FieldInvalid()

{

// mock entities

var userTypeMock = new Mock<DbSet<UserType>>();

userTypeMock.AsQuerable<UserType>();

var userMock = new Mock<DbSet<User>>();

userMock.AsQuerable<UserType>();

var courseMock = new Mock<DbSet<Course>>();

courseMock.AsQuerable<Course>();

var classMock = new Mock<DbSet<Classes>>();

classMock.AsQuerable<Classes>();

var classStudentMock = new Mock<DbSet<ClassStudent>>();

classStudentMock.AsQuerable<ClassStudent>();

// mock DbContext

var dbMock = new Mock<MyDBContext>();

dbMock.Setup(x => x.UserTypes).Returns(userTypeMock.Object);

dbMock.Setup(x => x.Users).Returns(userMock.Object);

dbMock.Setup(x => x.Courses).Returns(courseMock.Object);

dbMock.Setup(x => x.Classes).Returns(classMock.Object);

dbMock.Setup(x => x.ClassStudents).Returns(classStudentMock.Object);

ClassCreateModel classModel = new ClassCreateModel

{

Id = 0,

//CourseId = 1, //Course is required

Description = "Test",

Honor = "Test",

Level = "Test",

Status = "Left",

TaId = 4,

TeacherId = 3,

Time = DateTime.Now.UTCSeconds(),

Title = "New Class",

StudentList = new List<int> { 1, 2 }

};

ClassesController classController = new ClassesController(dbMock.Object);

IActionResult result = classController.CreateClass(classModel);

Assert.IsType<BadRequestObjectResult>(result);

}

[Fact]

public void CreateClass\_Return\_Error()

{

// mock entities

var userTypeMock = new Mock<DbSet<UserType>>();

userTypeMock.AsQuerable<UserType>();

var userMock = new Mock<DbSet<User>>();

userMock.AsQuerable<UserType>();

var courseMock = new Mock<DbSet<Course>>();

courseMock.AsQuerable<Course>();

var classMock = new Mock<DbSet<Classes>>();

classMock.AsQuerable<Classes>();

var classStudentMock = new Mock<DbSet<ClassStudent>>();

classStudentMock.AsQuerable<ClassStudent>();

// mock DbContext

var dbMock = new Mock<MyDBContext>();

dbMock.Setup(x => x.UserTypes).Returns(userTypeMock.Object);

dbMock.Setup(x => x.Users).Returns(userMock.Object);

dbMock.Setup(x => x.Courses).Returns(courseMock.Object);

dbMock.Setup(x => x.Classes).Returns(classMock.Object);

dbMock.Setup(x => x.ClassStudents).Returns(classStudentMock.Object);

ClassesController classController = new ClassesController(dbMock.Object);

// New Class Object is null, it will throw error

IActionResult result = classController.CreateClass(null);

Assert.IsType<BadRequestObjectResult>(result);

}

[Fact]

public void GetUsers\_Return\_OK()

{

// mock entities

var userTypeMock = new Mock<DbSet<UserType>>();

UserType usertype1 = new UserType { Id = 1, Title = "Student" };

UserType usertype2 = new UserType { Id = 2, Title = "Parent" };

UserType usertype3 = new UserType { Id = 3, Title = "Teacher" };

UserType usertype4 = new UserType { Id = 4, Title = "TeacherAssistant" };

userTypeMock.AddList(usertype1, usertype2, usertype3, usertype4);

var userMock = new Mock<DbSet<User>>();

User user1 = new User { Id = 1, FirstName = "Student", LastName = "LastName", RoleId = 1, Status = 1, Deleted = 0 };

User user2 = new User { Id = 2, FirstName = "Parent", LastName = "LastName", RoleId = 2, Status = 1, Deleted = 0 };

User user3 = new User { Id = 3, FirstName = "Teacher", LastName = "LastName", RoleId = 3, Status = 1, Deleted = 0 };

User user4 = new User { Id = 4, FirstName = "TA", LastName = "LastName", RoleId = 4, Status = 1, Deleted = 0 };

userMock.AddList(user1, user2, user3, user4);

var courseMock = new Mock<DbSet<Course>>();

courseMock.AsQuerable<Course>();

var classMock = new Mock<DbSet<Classes>>();

classMock.AsQuerable<Classes>();

var classStudentMock = new Mock<DbSet<ClassStudent>>();

classStudentMock.AsQuerable<ClassStudent>();

// mock DbContext

var dbMock = new Mock<MyDBContext>();

dbMock.Setup(x => x.UserTypes).Returns(userTypeMock.Object);

dbMock.Setup(x => x.Users).Returns(userMock.Object);

dbMock.Setup(x => x.Courses).Returns(courseMock.Object);

dbMock.Setup(x => x.Classes).Returns(classMock.Object);

dbMock.Setup(x => x.ClassStudents).Returns(classStudentMock.Object);

ClassesController classController = new ClassesController(dbMock.Object);

IActionResult result = classController.GetUsers();

Assert.IsType<OkObjectResult>(result);

}

[Fact]

public void GetUsers\_Return\_Fail()

{

// mock entities

var userTypeMock = new Mock<DbSet<UserType>>();

userTypeMock.AsQuerable<UserType>();

var userMock = new Mock<DbSet<User>>();

userMock.AsQuerable<UserType>();

var courseMock = new Mock<DbSet<Course>>();

courseMock.AsQuerable<Course>();

var classMock = new Mock<DbSet<Classes>>();

classMock.AsQuerable<Classes>();

var classStudentMock = new Mock<DbSet<ClassStudent>>();

classStudentMock.AsQuerable<ClassStudent>();

// mock DbContext

var dbMock = new Mock<MyDBContext>();

dbMock.Setup(x => x.UserTypes).Returns(userTypeMock.Object);

dbMock.Setup(x => x.Users).Returns(userMock.Object);

dbMock.Setup(x => x.Courses).Returns(courseMock.Object);

dbMock.Setup(x => x.Classes).Returns(classMock.Object);

dbMock.Setup(x => x.ClassStudents).Returns(classStudentMock.Object);

// Users is empty in DbContext, it will throw error

ClassesController classController = new ClassesController(dbMock.Object);

IActionResult result = classController.GetUsers();

Assert.IsType<BadRequestObjectResult>(result);

}

}

}