

Bài giảng tích hợp:

ENTITY FRAMEWORK CORE

Faculty of IT

Email: smdat@hueic.edu.vn

ENTITY FRAMEWORK CORE

Chương 1. Tổng quan về Entity Framework Core

Chương 2. Truy vấn trong Entity Framework Core

Chương 3. Razor Page với Entity Framework Core

Chương 4. Asp.Net Core MVC với Entity Framework Core

Chương 5: API

1. Khởi tạo project web API

- **Tạo project**

- Tạo project mới, chọn **ASP.NET Core Web App (Model-View-Controller) (.NET 5.0)**
- Thiết lập style cho site: *Views/Shared/_Layout.cshtml* để tạo các menu

```
<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-
action="Index">Home</a> </li>
<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-
action="About">About</a> </li>
<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-controller="Students"
asp-action="Index">Students</a> </li>
<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-controller="Courses"
asp-action="Index">Courses</a> </li>
<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-controller="Instructors"
asp-action="Index">Instructors</a> </li>
<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-controller="Departments"
asp-action="Index">Departments</a> </li>
```

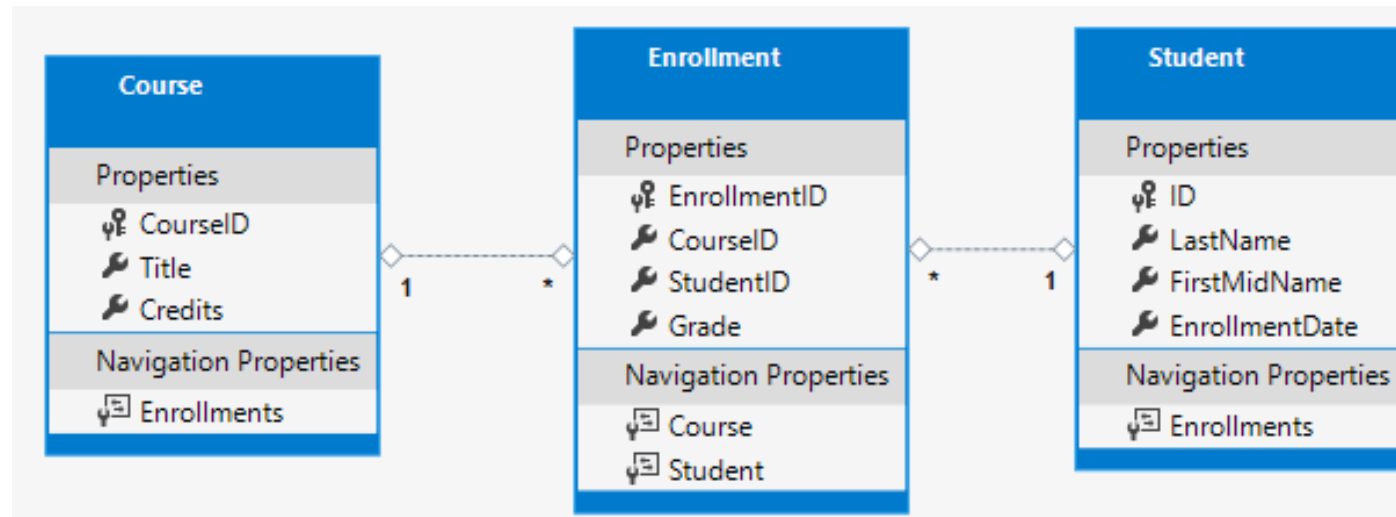
1. Khởi tạo project

- **Cài đặt các gói EF Core NuGet**

- Microsoft.EntityFrameworkCore
- Microsoft.EntityFrameworkCore.Tools
- Microsoft.EntityFrameworkCore.SqlServer
- Microsoft.AspNetCore.Diagnostics.EntityFrameworkCore: cung cấp middleware cho các trang lỗi EF Core. Middleware này giúp phát hiện và chặn đoán các lỗi của migration trong EF Core.

1. Khởi tạo project

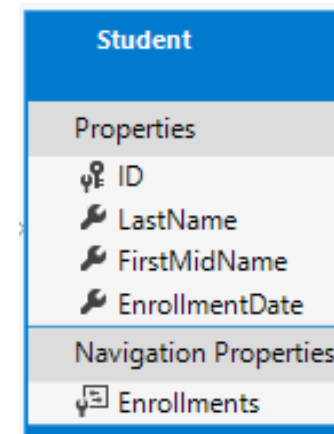
• Tạo các model dữ liệu



• Thực thể Student

- Trong thư mục Model, tạo Student

```
public class Student
{
    public int ID { get; set; }
    public string LastName { get; set; }
    public string FirstMidName { get; set; }
    public DateTime EnrollmentDate { get; set; }
    public ICollection<Enrollment> Enrollments { get; set; }
}
```



1. Khởi tạo project

• Thực thể Course

- Trong thư mục Model, tạo Course

```
public class Course  
{
```

```
    [DatabaseGenerated(DatabaseGeneratedOption.None)]
```

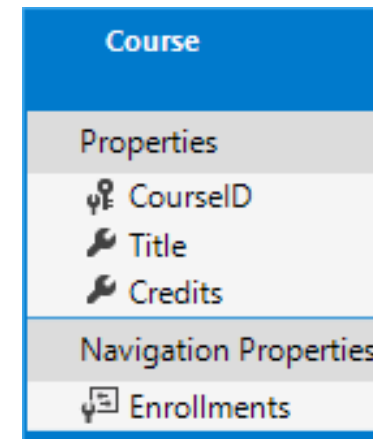
```
    public int CourseID { get; set; }
```

```
    public string Title { get; set; }
```

```
    public int Credits { get; set; }
```

```
    public ICollection<Enrollment> Enrollments { get; set; }
```

```
}
```



1. Khởi tạo project

• Thực thể Enrollment

- Trong thư mục Model, tạo Enrollment

```
public enum Grade
{
    A, B, C, D, F
}

public class Enrollment
{
    public int EnrollmentID { get; set; }
    public int CourseID { get; set; }
    public int StudentID { get; set; }
    public Grade? Grade { get; set; }
    public Course Course { get; set; }
    public Student Student { get; set; }
}
```

Enrollment
Properties
EnrollmentID
CourseID
StudentID
Grade
Navigation Properties
Course
Student

1. Khởi tạo project

- **Tạo database context**

- Tạo thư mục Data, tạo class SchoolContext

```
public class SchoolContext : DbContext
{
    public SchoolContext(DbContextOptions<SchoolContext>
options) : base(options)
    {
    }

    public DbSet<Course> Courses { get; set; }
    public DbSet<Enrollment> Enrollments { get; set; }
    public DbSet<Student> Students { get; set; }

    protected override void OnModelCreating(ModelBuilder
modelBuilder)
    {
        modelBuilder.Entity<Course>().ToTable("Course");
        modelBuilder.Entity<Enrollment>().ToTable("Enrollment");
        modelBuilder.Entity<Student>().ToTable("Student");
    }
}
```


1. Khởi tạo project

- Đăng ký **SchoolContext**

- Mở file *Startup.cs*, cập nhật lại

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<SchoolContext>(options =>
        options.UseSqlServer(Configuration
            .GetConnectionString("DefaultConnection")));
    services.AddDatabaseDeveloperPageExceptionFilter();
    services.AddControllersWithViews();
}
```

- Mở file *appsetting.json*, tạo chuỗi kết nối

```
"ConnectionStrings": {
    "DefaultConnection":
    "Server=(localdb)\\mssqllocaldb;Database=SchoolManager;Trusted_Connection=True;MultipleActiveResultSets=true" },
```

1. Khởi tạo project

- Tạo dữ liệu thử

- Ở thư mục Data, tạo class DbInitializer.cs

```
public static class DbInitializer {  
    public static void Initialize(SchoolContext context) {  
        context.Database.EnsureCreated();  
        // Look for any students.  
        if (context.Students.Any()) {  
            return; // DB has been seeded  
        }  
        var students = new Student[] {  
            new Student{FirstMidName="Carson", LastName="Alexander",  
EnrollmentDate=DateTime.Parse("2005-09-01")},  
            new Student{FirstMidName="Meredith", LastName="Alonso",  
EnrollmentDate=DateTime.Parse("2002-09-01")},  
            new Student{FirstMidName="Arturo", LastName="Anand",  
EnrollmentDate=DateTime.Parse("2003-09-01")},  
            new Student{FirstMidName="Gytis", LastName="Barzdukas",  
EnrollmentDate=DateTime.Parse("2002-09-01")},  
            new Student{FirstMidName="Yan", LastName="Li",  
EnrollmentDate=DateTime.Parse("2002-09-01")},  
            new Student{FirstMidName="Peggy", LastName="Justice",  
EnrollmentDate=DateTime.Parse("2001-09-01")},  
            new Student{FirstMidName="Laura", LastName="Norman",  
EnrollmentDate=DateTime.Parse("2003-09-01")},  
            new Student{FirstMidName="Nino", LastName="Olivetto",  
EnrollmentDate=DateTime.Parse("2005-09-01")}  
        };  
    }  
};
```

1. Khởi tạo project

- Tạo dữ liệu thử

- Ở thư mục Data, tạo class DbInitializer.cs

```
foreach (Student s in students)
{
    context.Students.Add(s);
}
context.SaveChanges();
var courses = new Course[] {
    new Course{CourseID=1050,Title="Chemistry",Credits=3},
    new Course{CourseID=4022,Title="Microeconomics",Credits=3},
    new Course{CourseID=4041,Title="Macroeconomics",Credits=3},
    new Course{CourseID=1045,Title="Calculus",Credits=4},
    new Course{CourseID=3141,Title="Trigonometry",Credits=4},
    new Course{CourseID=2021,Title="Composition",Credits=3},
    new Course{CourseID=2042,Title="Literature",Credits=4}
};
foreach (Course c in courses)
{
    context.Courses.Add(c);
}
context.SaveChanges();
```

1. Khởi tạo project

- Tạo dữ liệu thử

- Ở thư mục Data , tạo class DbInitializer.cs

```
var enrollments = new Enrollment[]
{
    new Enrollment{StudentID=1, CourseID=1050, Grade=Grade.A},
    new Enrollment{StudentID=1, CourseID=4022, Grade=Grade.C},
    new Enrollment{StudentID=1, CourseID=4041, Grade=Grade.B},
    new Enrollment{StudentID=2, CourseID=1045, Grade=Grade.B},
    new Enrollment{StudentID=2, CourseID=3141, Grade=Grade.F},
    new Enrollment{StudentID=2, CourseID=2021, Grade=Grade.F},
    new Enrollment{StudentID=3, CourseID=1050},
    new Enrollment{StudentID=4, CourseID=1050},
    new Enrollment{StudentID=4, CourseID=4022, Grade=Grade.F},
    new Enrollment{StudentID=5, CourseID=4041, Grade=Grade.C},
    new Enrollment{StudentID=6, CourseID=1045},
    new Enrollment{StudentID=7, CourseID=3141, Grade=Grade.A},
};
foreach (Enrollment e in enrollments)
{
    context.Enrollments.Add(e);
}
context.SaveChanges();
}
```

1. Khởi tạo project

- **Tạo dữ liệu thử**

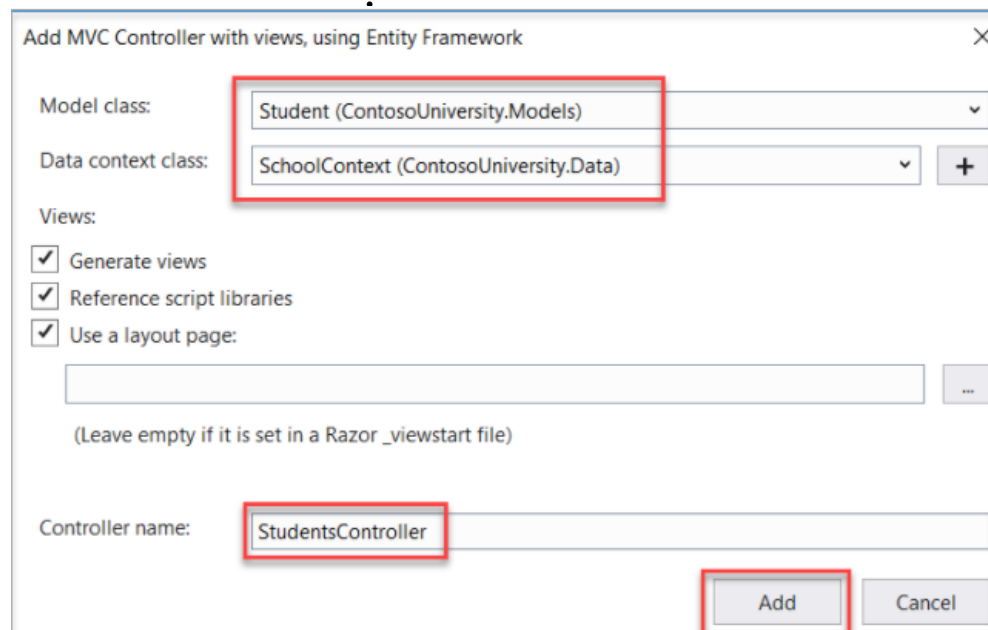
- **Cập nhật lại file Program.cs**

```
public static void Main(string[] args) {  
    var host = CreateHostBuilder(args).Build();  
    CreateDbIfNotExists(host);  
    host.Run();  
}  
  
private static void CreateDbIfNotExists(IHost host) {  
    using (var scope = host.Services.CreateScope()) {  
        var services = scope.ServiceProvider;  
        try {  
            var context = services.GetRequiredService<SchoolContext>();  
            DbInitializer.Initialize(context);  
        }  
        catch (Exception ex) {  
            var logger = services.GetRequiredService<ILogger<Program>>();  
            logger.LogError(ex, "An error occurred creating the DB.");  
        }  
    }  
}  
  
public static IHostBuilder CreateHostBuilder(string[] args) =>  
    Host.CreateDefaultBuilder(args)  
        .ConfigureWebHostDefaults(webBuilder =>  
        {  
            webBuilder.UseStartup<Startup>();  
        })  
    ;  
}
```

1. Khởi tạo project

- **Tạo controller và các view**

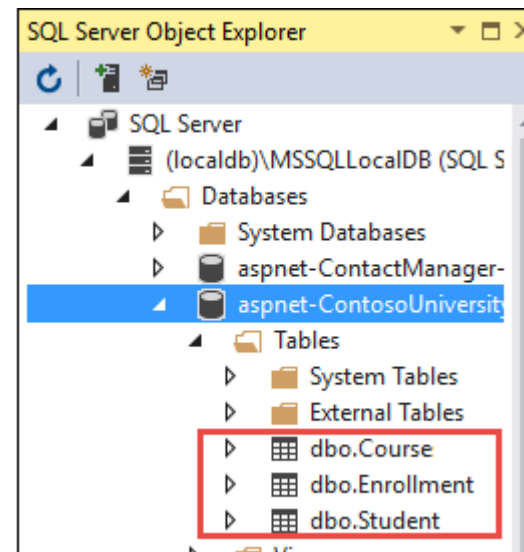
- Trong **Solution Explorer**, click phải vào thư mục **Controllers**, chọn **Add -> New Scaffolded Item**
- Trong hộp thoại Add Scaffold:
 - Chọn **MVC controller with views, using Entity Framework**.
 - Chọn Add, hộp thoại **Add MVC Controller with views, using Entity Framework** xuất hiện



- Mục Model class: chọn Student
- Mục Data context class: chọn SchoolContext
- Chọn nút Add

1. Khởi tạo project

- Xem database:
 - Vào menu View, chọn **SQL Server Object Explorer**



The screenshot shows the SQL Server Data Viewer window displaying the data in the 'dbo.Students' table. The table has five columns: 'ID', 'EnrollmentDate', 'FirstMidName', and 'LastName'. The data is as follows:

ID	EnrollmentDate	FirstMidName	LastName
1	9/1/2005 12:00:...	Carson	Alexander
2	9/1/2002 12:00:...	Meredith	Alonso
3	9/1/2003 12:00:...	Arturo	Anand
4	9/1/2002 12:00:...	Gytis	Barzdukas

- Chạy thử

1. Khởi tạo project

- **SQL Logging trong EF Core:**

- Cấu hình Logging thường được cung cấp trong phần Logging của appsettings.json

```
{ "ConnectionStrings": {  
  "DefaultConnection":  
    "Server=(localdb)\\mssqllocaldb;Database=MyDB-  
2;Trusted_Connection=True;MultipleActiveResultSets=true"  
},  
  "Logging": {  
    "LogLevel": {  
      "Default": "Information",  
      "Microsoft": "Warning",  
      "Microsoft.Hosting.Lifetime": "Information",  
      "Microsoft.EntityFrameworkCore.Database.Command":  
        "Information"  
    }  
  },  
  "AllowedHosts": "*" }  
}
```

- Chạy thử

1. Khởi tạo project
2. CRUD

• Tùy chỉnh trang Details:

- Code đã được scaffold cho trang Index của Students bỏ qua thuộc tính Enrollments, bởi vì thuộc tính này là kiểu danh sách. Trong trang Details, chúng ta cần hiển thị nội dung của danh sách này trong một bảng HTML.
- Trong *Controllers/StudentsController.cs*, action cho view của Details sử dụng phương thức `FirstOrDefaultAsync` để truy xuất đến một thực thể đơn Student. Vì vậy cần thêm phương thức `Include`, `ThenInclude` và `AsNoTracking` để truy xuất từ nhiều thực thể.

```
public async Task<IActionResult> Details(int? id) {  
    if (id == null) {  
        return NotFound();  
    }  
    var student = await _context.Students  
        .Include(s => s.Enrollments)  
        .ThenInclude(e => e.Course)  
        .AsNoTracking()  
        .FirstOrDefaultAsync(m => m.ID == id);  
    if (student == null) {  
        return NotFound();  
    }  
    return View(student);  
}
```

1. Khởi tạo project
2. CRUD

- Tùy chỉnh trang Details:

Student		
LastName	Alexander	
FirstMidName	Carson	
EnrollmentDate	9/1/2005 12:00:00 AM	
Enrollments	Course Title	Grade
	Chemistry	A
	Microeconomics	C
	Macroeconomics	B

1. Khởi tạo project**2. CRUD****• Tùy chỉnh trang Details:****• Thêm Enrollments cho view Details****• Mở *Views/Students/Details.cshtml***

```
<dt class="col-sm-2">
    @Html.DisplayNameFor(model => model.Enrollments)
</dt>
<dd class="col-sm-10">
    <table class="table">
        <tr>
            <th>Course Title</th>
            <th>Grade</th>
        </tr>
        @foreach (var item in Model.Enrollments) {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.Course.Title)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Grade)
                </td>
            </tr>
        }
    </table>
</dd>
```

1. Khởi tạo project**2. CRUD****• Cập nhật trang Create:**

- Trong *StudentsController.cs*, chỉnh lại phương thức Create HttpPost

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Create(
    [Bind("EnrollmentDate, FirstMidName, LastName")] Student student)
{
    try {
        if (ModelState.IsValid) {
            _context.Add(student);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }
    }
    catch (DbUpdateException /* ex */) {
        //Log the error (uncomment ex variable name and write a log.
        ModelState.AddModelError("", "Unable to save changes. " +
            "Try again, and if the problem persists " +
            "see your system administrator.");
    }
    return View(student);
}
```

1. Khởi tạo project

2. CRUD

• Cập nhật trang Update:

- Trong *StudentsController.cs*, chỉnh lại phương thức *Edit HttpPost*

```
[[HttpPost, ActionName("Edit")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> EditPost(int? id) {
    if (id == null) {
        return NotFound();
    }
    var studentToUpdate = await _context.Students.FirstOrDefault(s
=> s.ID == id);
    if (await TryUpdateModelAsync<Student>( studentToUpdate, "",
        s => s.FirstMidName, s => s.LastName, s => s.EnrollmentDate))
    {
        try {
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }
        catch (DbUpdateException /* ex */)
        {
            //Log the error (uncomment ex variable name and write a log.)
            ModelState.AddModelError("", "Unable to save changes. "
+ "Try again, and if the problem persists, "
+ "see your system administrator.");
        }
    }
    return View(studentToUpdate);
}
```

1. Khởi tạo project

2. CRUD

- Cập nhật trang **Delete**:

- Trong *StudentsController.cs*, chỉnh lại phương thức *Delete*

```
public async Task<IActionResult> Delete(int? id, bool? saveChangesError =
false)
{
    if (id == null) {
        return NotFound(); }
    var student = await _context.Students
        .AsNoTracking()
        .FirstOrDefaultAsync(m => m.ID == id);
    if (student == null) {
        return NotFound(); }
    if (saveChangesError.GetValueOrDefault())
    {
        ViewData["ErrorMessage"] =
            "Delete failed. Try again, and if the problem persists " +
            "see your system administrator.";
    }
    return View(student);
}
```

1. Khởi tạo project

2. CRUD

- **Cập nhật trang Delete:**

- Trong *StudentsController.cs*, chỉnh lại phương thức *Delete*

```
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public async Task DeleteConfirmed(int id)
{
    var student = await _context.Students.FindAsync(id);
    if (student == null) {
        return RedirectToAction(nameof(Index)); }
    try
    {
        _context.Students.Remove(student);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    catch (DbUpdateException /* ex */)
    {
        //Log the error (uncomment ex variable name and write a log.)
        return RedirectToAction(nameof>Delete),
            new { id = id, saveChangesError = true });
    }
}
```

- **Cập nhật view Delete**

```
<h2>Delete</h2>
<p class="text-danger">@ViewData["ErrorMessage"]</p>
<h3>Are you sure you want to delete this?</h3>
```

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang

- Sắp xếp:

- Trong *StudentsController.cs* sửa lại phương thức Index

```
public async Task<IActionResult> Index(string sortOrder)
{
    ViewData["NameSortParm"] = String.IsNullOrEmpty(sortOrder) ?
    "name_desc" : "";
    ViewData["DateSortParm"] = sortOrder == "Date" ? "date_desc" : "Date";
    var students = from s in _context.Students select s;
    switch (sortOrder)
    {
        case "name_desc":
            students = students.OrderByDescending(s => s.LastName);
            break;
        case "Date":
            students = students.OrderBy(s => s.EnrollmentDate);
            break;
        case "date_desc":
            students = students.OrderByDescending(s => s.EnrollmentDate);
            break;
        default:
            students = students.OrderBy(s => s.LastName);
            break;
    }
    return View(await students.AsNoTracking().ToListAsync());
}
```


1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang

- **Sắp xếp:**

- Trong *Views/Students/Index.cshtml*, chỉnh sửa lại

```
<table class="table">
  <thead>
    <tr>
      <th>
        <a asp-action="Index" asp-route-sortOrder =
"@ViewData["NameSortParm"]">@Html.DisplayNameFor(model =>
model.LastName)</a>
      </th>
      <th>
        @Html.DisplayNameFor(model => model.FirstMidName)
      </th>
      <th>
        <a asp-action="Index" asp-route-sortOrder =
"@ViewData["DateSortParm"]">@Html.DisplayNameFor(model =>
model.EnrollmentDate)</a>
      </th>
    </tr>
  </thead>
  <tbody>
    ...
```

- 1. Khởi tạo project**
- 2. CRUD**
- 3. Sắp xếp, lọc, phân trang**

- **Tìm kiếm:**

- Trong *StudentsController.cs* sửa lại phương thức Index

```
public async Task<IActionResult> Index(string sortOrder, string
searchString)
{
    ViewData["NameSortParm"] = String.IsNullOrEmpty(sortOrder) ?
"name_desc" : "";
    ViewData["DateSortParm"] = sortOrder == "Date" ? "date_desc" :
"Date";
    ViewData["CurrentFilter"] = searchString;
    var students = from s in _context.Students select s;
    if (!String.IsNullOrEmpty(searchString))
    {
        students = students.Where(s =>
            s.LastName.Contains(searchString)
            || s.FirstMidName.Contains(searchString));
    }
    switch (sortOrder) {
        .....
```

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang

- **Tìm kiếm:**

- Trong *Views/Students/Index.cshtml*, chỉnh sửa lại

```
<p>
    <a asp-action="Create">Create New</a>
</p>
<form asp-action="Index" method="get">
    <div class="form-actions no-color">
        <p>
            Find by name: <input type="text" name="SearchString"
value="@ViewData["CurrentFilter"]" /> <input type="submit" value="Search"
class="btn btn-default" /> |
            <a asp-action="Index">Back to Full List</a>
        </p>
    </div>
</form>
<table class="table">
....
```

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang

- **Phân trang:**

- Trong thư mục project, tạo class *PaginatedList.cs*

```
public class PaginatedList<T> : List<T> {
    public int PageIndex { get; private set; }
    public int TotalPages { get; private set; }
    public PaginatedList(List<T> items, int count, int pageIndex, int
pageIndex) {
        PageIndex = pageIndex;
        TotalPages = (int)Math.Ceiling(count / (double)pageSize);
        this.AddRange(items);
    }
    public bool HasPreviousPage {
        get { return (PageIndex > 1); }
    }
    public bool HasNextPage {
        get { return (PageIndex < TotalPages); }
    }
    public static async Task<PaginatedList<T>> CreateAsync(IQueryable<T>
source, int pageIndex, int pageSize) {
        var count = await source.CountAsync();
        var items = await source.Skip((pageIndex - 1) *
pageSize).Take(pageSize).ToListAsync();
        return new PaginatedList<T>(items, count, pageIndex,
pageSize);
    }
}
```

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang

- Phân trang:

- Trong *StudentsController.cs*, chỉnh sửa lại phương thức Index

```
public async Task<IActionResult> Index(
    string sortOrder, string currentFilter,
    string searchString, int? pageNumber)
{
    ViewData["CurrentSort"] = sortOrder;
    ViewData["NameSortParm"] = String.IsNullOrEmpty(sortOrder) ?
"name_desc" : "";
    ViewData["DateSortParm"] = sortOrder == "Date" ? "date_desc" :
"Date";
    if (searchString != null) {
        pageNumber = 1; }
    else {
        searchString = currentFilter; }
    ViewData["CurrentFilter"] = searchString;
    var students = from s in _context.Students select s;
    if (!String.IsNullOrEmpty(searchString)) {
        students = students.Where(s => s.LastName.Contains(searchString)
|| s.FirstMidName.Contains(searchString)); }
    ....
    int pageSize = 3;
    return View(await PaginatedList<Student>.CreateAsync(students
.AsNoTracking(), pageNumber ?? 1, pageSize));
}
```

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang

- **Phân trang:**

- Trong *Views/Students/Index.cshtml*, chỉnh sửa lại

```
@model PaginatedList<MVC_EFCore.Models.Student>
@{ ViewData["Title"] = "Index"; }
.....
<table class="table">
    <thead>
        <tr>
            <th>
                <a asp-action="Index" asp-route-sortOrder =
"@ViewData["NameSortParm]" asp-route-currentFilter =
"@ViewData["CurrentFilter"]">Last Name</a>
            </th>
            <th> First Name </th>
            <th> <a asp-action="Index" asp-route-sortOrder =
"@ViewData["DateSortParm]" asp-route-currentFilter =
"@ViewData["CurrentFilter"]">Enrollment Date</a>
            </th>
        </tr>
    </thead>
    <tbody>
        .....
    </tbody>
</table>
```

- 1. Khởi tạo project**
- 2. CRUD**
- 3. Sắp xếp, lọc, phân trang**

- **Phân trang:**

- Trong *Views/Students/Index.cshtml*, chỉnh sửa lại

```
.....
</table>
@{
    var prevDisabled = !Model.HasPreviousPage ? "disabled" : "";
    var nextDisabled = !Model.HasNextPage ? "disabled" : "";
}
<a asp-action="Index"
    asp-route-sortOrder="@ViewData["CurrentSort"]"
    asp-route-pageNumber="@ (Model.PageIndex - 1) "
    asp-route-currentFilter="@ViewData["CurrentFilter"]"
    class="btn btn-default @prevDisabled"> Previous
</a>
<a asp-action="Index"
    asp-route-sortOrder="@ViewData["CurrentSort"]"
    asp-route-pageNumber="@ (Model.PageIndex + 1) "
    asp-route-currentFilter="@ViewData["CurrentFilter"]"
    class="btn btn-default @nextDisabled"> Next
</a>
```

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang

- **Group:**

- **Tạo view model**

- Trong thư Models, tạo thư mục *SchoolViewModels*
 - Tạo class *EnrollmentDateGroup.cs*

```
public class EnrollmentDateGroup
{
    [DataType(DataType.Date)]
    public DateTime? EnrollmentDate { get; set; }

    public int StudentCount { get; set; }
}
```

- **Chỉnh sửa Home Controller**

```
public class HomeController : Controller
{
    private readonly ILogger<HomeController> _logger;
    private readonly SchoolContext _context;
    public HomeController(ILogger<HomeController> logger,
        SchoolContext context)
    {
        _logger = logger;
        _context = context;
    }
}
```


- 1. Khởi tạo project**
- 2. CRUD**
- 3. Sắp xếp, lọc, phân trang**

- **Group:**

- **Chỉnh sửa Home Controller**

- Thêm phương thức About:

```
public async Task<ActionResult> About() {  
    IQueryable<EnrollmentDateGroup> data =  
        from student in _context.Students  
        group student by student.EnrollmentDate into dateGroup  
        select new EnrollmentDateGroup()  
        {  
            EnrollmentDate = dateGroup.Key,  
            StudentCount = dateGroup.Count()  
        };  
    return View(await data.AsNoTracking().ToListAsync());  
}
```

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang

- **Group:**

- Tạo View cho About

```
@model
IEnumerable<MVC_EFCore.Models.SchoolViewModels.EnrollmentDateGroup>
@{
    ViewData["Title"] = "Student Body Statistics";
}
<h2>Student Body Statistics</h2>
<table>
    <tr>
        <th> Enrollment Date </th>
        <th> Students </th>
    </tr>
    @foreach (var item in Model) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.EnrollmentDate)
            </td>
            <td> @item.StudentCount </td>
        </tr>
    }
</table>
```

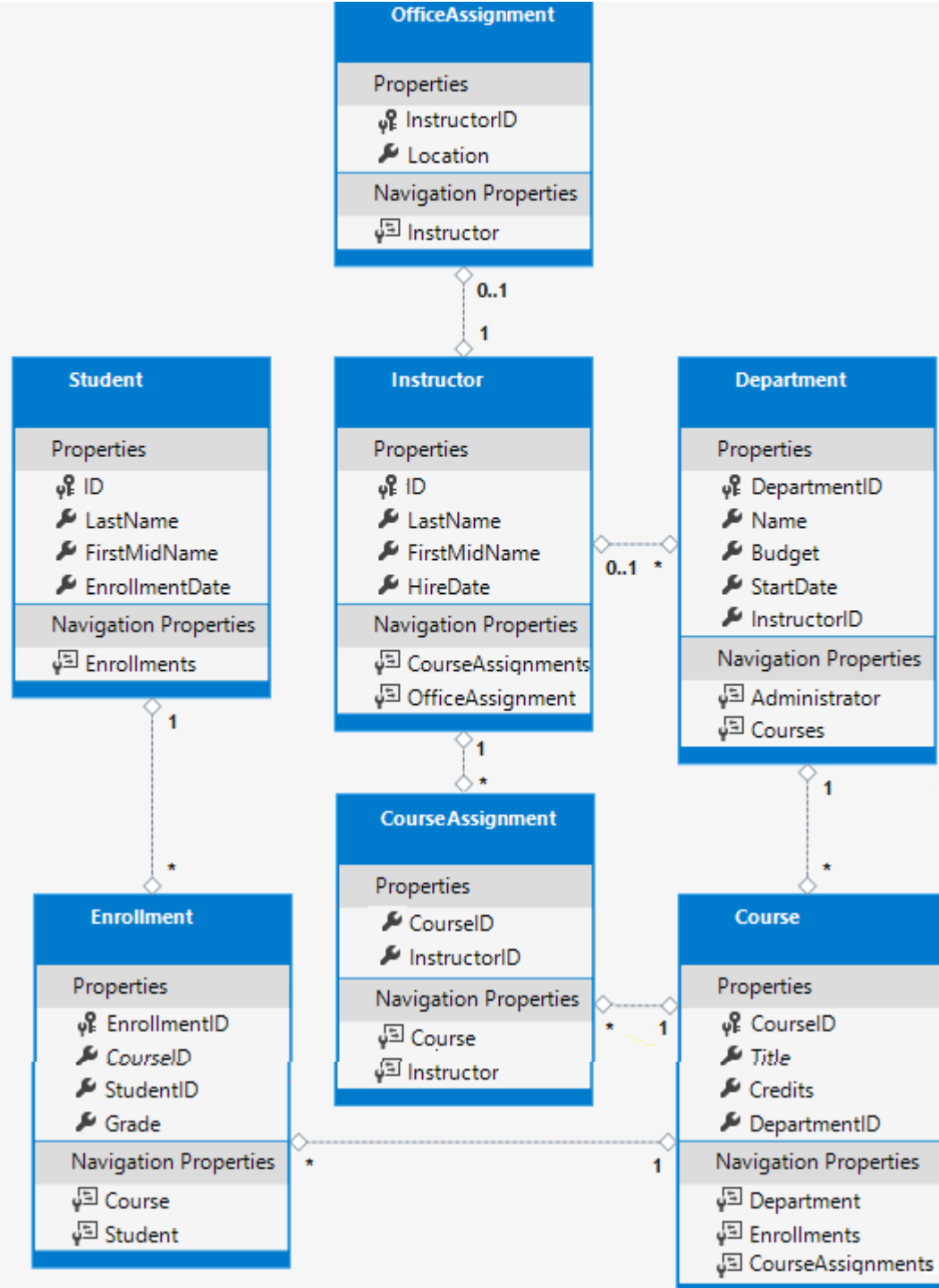
- Chạy thử

- Add migration

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration

Chương 4. MVC với Entity Framework Core






1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng



1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng

• Thay đổi thực thể Student







```
public class Student {  
    public int ID { get; set; }  
    [Required]  
    [StringLength(50)]  
    [Display(Name = "Last Name")]  
    public string LastName { get; set; }  
    [Required]  
    [StringLength(50)]  
    [Column("FirstName")]  
    [Display(Name = "First Name")]  
    public string FirstMidName { get; set; }  
    [DataType(DataType.Date)]  
    [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}",  
        ApplyFormatInEditMode = true)]  
    [Display(Name = "Enrollment Date")]  
    public DateTime EnrollmentDate { get; set; }  
    [Display(Name = "Full Name")]  
    public string FullName {  
        get { return LastName + ", " + FirstMidName; }  
    }  
    public ICollection<Enrollment> Enrollments { get; set; }  
}
```

Student	
Properties	
	ID
	LastName
	FirstMidName
	EnrollmentDate
Navigation Properties	
	Enrollments

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng

• Tạo thực thể Instructor

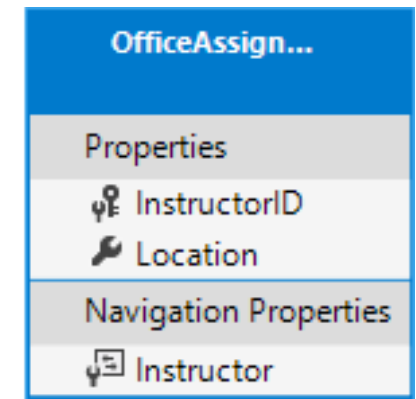
```
public class Instructor {
    public int ID { get; set; }
    [Required]
    [Display(Name = "Last Name")]
    [StringLength(50)]
    public string LastName { get; set; }
    [Required]
    [Column("FirstName")]
    [Display(Name = "First Name")]
    [StringLength(50)]
    public string FirstMidName { get; set; }
    [DataType(DataType.Date)]
    [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}",
        ApplyFormatInEditMode = true)]
    [Display(Name = "Hire Date")]
    public DateTime HireDate { get; set; }
    [Display(Name = "Full Name")]
    public string FullName {
        get {
            return LastName + ", " + FirstMidName;
        }
    }
    public ICollection<CourseAssignment> CourseAssignments { get; set; }
    public OfficeAssignment OfficeAssignment { get; set; }
}
```

Instructor	
Properties	
	ID
	LastName
	FirstMidName
	HireDate
Navigation Properties	
	CourseAssignments
	OfficeAssignment

• Tạo thực thể OfficeAssignment

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng








```
public class OfficeAssignment
{
    [Key]
    public int InstructorID { get; set; }
    [StringLength(50)]
    [Display(Name = "Office Location")]
    public string Location { get; set; }
    public Instructor Instructor { get; set; }
}
```



• Cập nhật thực thể Course

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng








```
public class Course
{
    [DatabaseGenerated(DatabaseGeneratedOption.None)]
    [Display(Name = "Number")]
    public int CourseID { get; set; }
    [StringLength(50, MinimumLength = 3)]
    public string Title { get; set; }
    [Range(0, 5)]
    public int Credits { get; set; }
    public int DepartmentID { get; set; }
    public Department Department { get; set; }
    public ICollection<Enrollment> Enrollments { get; set; }
    public ICollection<CourseAssignment> CourseAssignments { get;
set; }
}
```

Course
Properties
 CourseID
 Title
 Credits
 DepartmentID
Navigation Properties
 Department
 Enrollments
 CourseAssignments

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng

• Tạo thực thể Department

```
public class Department
{
    public int DepartmentID { get; set; }
    [StringLength(50, MinimumLength = 3)]
    public string Name { get; set; }
    [DataType(DataType.Currency)]
    [Column(TypeName = "money")]
    public decimal Budget { get; set; }
    [DataType(DataType.Date)]
    [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}",
        ApplyFormatInEditMode = true)]
    [Display(Name = "Start Date")]
    public DateTime StartDate { get; set; }
    public int? InstructorID { get; set; }
    public Instructor Administrator { get; set; }
    public ICollection<Course> Courses { get; set; }
}
```







Department	
Properties	
	DepartmentID
	Name
	Budget
	StartDate
	InstructorID
Navigation Properties	
	Administrator
	Courses

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng

• Cập nhật thực thể Enrollment

```
public enum Grade
{
    A, B, C, D, F
}





public class Enrollment
{
    public int EnrollmentID { get; set; }
    public int CourseID { get; set; }
    public int StudentID { get; set; }
    [DisplayFormat(NullDisplayText = "No grade")]
    public Grade? Grade { get; set; }
    public Course Course { get; set; }
    public Student Student { get; set; }
}
```

Enrollment
Properties
 EnrollmentID  CourseID  StudentID  Grade
Navigation Properties
 Course  Student

• Thực thể CourseAssignment

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng

```
public class CourseAssignment
{
    public int InstructorID { get; set; }
    public int CourseID { get; set; }
    public Instructor Instructor { get; set; }
    public Course Course { get; set; }
}
```

CourseAssignment	
Properties	
	CourseID
	InstructorID
Navigation Properties	
	Course
	Instructor

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng

• Cập nhật database context

```
public class SchoolContext : DbContext {
    public SchoolContext(DbContextOptions<SchoolContext> options) :
base(options)
    {
        public DbSet<Course> Courses { get; set; }
        public DbSet<Enrollment> Enrollments { get; set; }
        public DbSet<Student> Students { get; set; }
        public DbSet<Department> Departments { get; set; }
        public DbSet<Instructor> Instructors { get; set; }
        public DbSet<OfficeAssignment> OfficeAssignments { get; set; }
        public DbSet<CourseAssignment> CourseAssignments { get; set; }
        protected override void OnModelCreating(ModelBuilder modelBuilder) {
            modelBuilder.Entity<Course>().ToTable("Course");
            modelBuilder.Entity<Enrollment>().ToTable("Enrollment");
            modelBuilder.Entity<Student>().ToTable("Student");
            modelBuilder.Entity<Department>().ToTable("Department");
            modelBuilder.Entity<Instructor>().ToTable("Instructor");
            modelBuilder.Entity<OfficeAssignment>()
                .ToTable("OfficeAssignment");
            modelBuilder.Entity<CourseAssignment>()
                .ToTable("CourseAssignment");
            modelBuilder.Entity<CourseAssignment>()
                .HasKey(c => new { c.CourseID, c.InstructorID });
        }
    }
}
```

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng

- Cập nhật dữ liệu

- Tạo class *Data/DbInitializer.cs*

```
public static class DbInitializer {  
    public static void Initialize(SchoolContext context) {  
        //context.Database.EnsureCreated();  
        // Look for any students.  
        if (context.Students.Any()) {  
            return; // DB has been seeded  
        }  
        var students = new Student[] {  
            new Student { FirstMidName = "Carson", LastName = "Alexander",  
                EnrollmentDate = DateTime.Parse("2010-09-01") },  
            new Student { FirstMidName = "Meredith", LastName = "Alonso",  
                EnrollmentDate = DateTime.Parse("2012-09-01") },  
            new Student { FirstMidName = "Arturo", LastName = "Anand",  
                EnrollmentDate = DateTime.Parse("2013-09-01") },  
            new Student { FirstMidName = "Gytis", LastName = "Barzdukas",  
                EnrollmentDate = DateTime.Parse("2012-09-01") },  
            new Student { FirstMidName = "Yan", LastName = "Li",  
                EnrollmentDate = DateTime.Parse("2012-09-01") },  
            new Student { FirstMidName = "Peggy", LastName = "Justice",  
                EnrollmentDate = DateTime.Parse("2011-09-01") },  
            new Student { FirstMidName = "Laura", LastName = "Norman",  
                EnrollmentDate = DateTime.Parse("2013-09-01") },  
            new Student { FirstMidName = "Nino", LastName = "Olivetto",  
                EnrollmentDate = DateTime.Parse("2005-09-01") }  
        };  
    }  
};
```

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng

- **Cập nhật dữ liệu**

- Tạo class *Data/DbInitializer.cs*

```
foreach (Student s in students) {  
    context.Students.Add(s);  
}  
context.SaveChanges();  
var instructors = new Instructor[] {  
    new Instructor { FirstMidName = "Kim", LastName = "Abercrombie",  
        HireDate = DateTime.Parse("1995-03-11") },  
    new Instructor { FirstMidName = "Fadi", LastName = "Fakhouri",  
        HireDate = DateTime.Parse("2002-07-06") },  
    new Instructor { FirstMidName = "Roger", LastName = "Harui",  
        HireDate = DateTime.Parse("1998-07-01") },  
    new Instructor { FirstMidName = "Candace", LastName = "Kapoor",  
        HireDate = DateTime.Parse("2001-01-15") },  
    new Instructor { FirstMidName = "Roger", LastName = "Zheng",  
        HireDate = DateTime.Parse("2004-02-12") } };  
foreach (Instructor i in instructors) {  
    context.Instructors.Add(i);  
}  
context.SaveChanges();
```

- 1. Khởi tạo project**
- 2. CRUD**
- 3. Sắp xếp, lọc, phân trang**
- 4. Migration**
- 5. Tạo model dữ liệu liên quan đến nhiều bảng**

- **Cập nhật dữ liệu**

- Tạo class *Data/DbInitializer.cs*

```
var departments = new Department[] {  
    new Department { Name = "English", Budget = 350000,  
        StartDate = DateTime.Parse("2007-09-01"), InstructorID =  
            instructors.Single( i => i.LastName == "Abercrombie").ID },  
    new Department { Name = "Mathematics", Budget = 100000,  
        StartDate = DateTime.Parse("2007-09-01"), InstructorID =  
            instructors.Single( i => i.LastName == "Fakhouri").ID },  
    new Department { Name = "Engineering", Budget = 350000,  
        StartDate = DateTime.Parse("2007-09-01"), InstructorID =  
            instructors.Single( i => i.LastName == "Harui").ID },  
    new Department { Name = "Economics", Budget = 100000,  
        StartDate = DateTime.Parse("2007-09-01"), InstructorID =  
            instructors.Single( i => i.LastName == "Kapoor").ID }  
};  
foreach (Department d in departments) {  
    context.Departments.Add(d);  
}  
context.SaveChanges();
```


1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng

- Cập nhật dữ liệu

- Tạo class *Data/DbInitializer.cs*

```
var courses = new Course[] {  
    new Course {CourseID = 1050, Title = "Chemistry", Credits = 3,  
        DepartmentID = departments.Single( s => s.Name ==  
"Engineering").DepartmentID },  
    new Course {CourseID = 4022, Title = "Microeconomics",Credits= 3,  
        DepartmentID = departments.Single( s => s.Name ==  
"Economics").DepartmentID },  
    new Course {CourseID = 4041, Title = "Macroeconomics",Credits= 3,  
        DepartmentID = departments.Single( s => s.Name ==  
"Economics").DepartmentID },  
    new Course {CourseID = 1045, Title = "Calculus", Credits = 4,  
        DepartmentID = departments.Single( s => s.Name ==  
"Mathematics").DepartmentID },  
    new Course {CourseID = 3141, Title = "Trigonometry", Credits = 4,  
        DepartmentID = departments.Single( s => s.Name ==  
"Mathematics").DepartmentID },  
    new Course {CourseID = 2021, Title = "Composition", Credits = 3,  
        DepartmentID = departments.Single( s => s.Name ==  
"English").DepartmentID },  
    new Course {CourseID = 2042, Title = "Literature", Credits = 4,  
        DepartmentID = departments.Single( s => s.Name ==  
"English").DepartmentID },  
};  
foreach (Course c in courses) { context.Courses.Add(c); }  
context.SaveChanges();
```


1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng

- **Cập nhật dữ liệu**

- Tạo class *Data/DbInitializer.cs*

```
var officeAssignments = new OfficeAssignment[]
{
    new OfficeAssignment {
        InstructorID = instructors.Single( i => i.LastName ==
            "Fakhouri").ID, Location = "Smith 17" },
    new OfficeAssignment {
        InstructorID = instructors.Single( i => i.LastName ==
            "Harui").ID, Location = "Gowan 27" },
    new OfficeAssignment {
        InstructorID = instructors.Single( i => i.LastName ==
            "Kapoor").ID, Location = "Thompson 304" },
};
foreach (OfficeAssignment o in officeAssignments)
{
    context.OfficeAssignments.Add(o);
}
context.SaveChanges();
```

Chương 4. MVC với Entity Framework Core

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng

```
var courseInstructors = new CourseAssignment[] {
    new CourseAssignment { CourseID = courses.Single(c => c.Title ==
"Chemistry").CourseID, InstructorID = instructors.Single(i => i.LastName ==
"Kapoor").ID },
    new CourseAssignment { CourseID = courses.Single(c => c.Title ==
"Chemistry").CourseID, InstructorID = instructors.Single(i => i.LastName ==
"Harui").ID },
    new CourseAssignment { CourseID = courses.Single(c => c.Title ==
"Microeconomics").CourseID, InstructorID = instructors.Single(i =>
i.LastName == "Zheng").ID },
    new CourseAssignment { CourseID = courses.Single(c => c.Title ==
"Macroeconomics").CourseID, InstructorID = instructors.Single(i =>
i.LastName == "Zheng").ID },
    new CourseAssignment { CourseID = courses.Single(c => c.Title ==
"Calculus").CourseID, InstructorID = instructors.Single(i => i.LastName ==
"Fakhouri").ID },
    new CourseAssignment { CourseID = courses.Single(c => c.Title ==
"Trigonometry").CourseID, InstructorID = instructors.Single(i => i.LastName
== "Harui").ID },
    new CourseAssignment { CourseID = courses.Single(c => c.Title ==
"Composition").CourseID, InstructorID = instructors.Single(i => i.LastName
== "Abercrombie").ID },
    new CourseAssignment { CourseID = courses.Single(c => c.Title ==
"Literature").CourseID, InstructorID = instructors.Single(i => i.LastName
== "Abercrombie").ID }, };

foreach (CourseAssignment ci in courseInstructors) {
    context.CourseAssignments.Add(ci); }

context.SaveChanges();
```

Chương 4. MVC với Entity Framework Core

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng

```
var enrollments = new Enrollment[] {  
    new Enrollment { StudentID = students.Single(s => s.LastName ==  
        "Alexander").ID, CourseID = courses.Single(c => c.Title == "Chemistry").CourseID, Grade  
        = Grade.A },  
    new Enrollment { StudentID = students.Single(s => s.LastName ==  
        "Alexander").ID, CourseID = courses.Single(c => c.Title == "Microeconomics").CourseID,  
        Grade = Grade.C },  
    new Enrollment { StudentID = students.Single(s => s.LastName ==  
        "Alexander").ID, CourseID = courses.Single(c => c.Title == "Macroeconomics").CourseID,  
        Grade = Grade.B },  
    new Enrollment { StudentID = students.Single(s => s.LastName ==  
        "Alonso").ID, CourseID = courses.Single(c => c.Title == "Calculus").CourseID, Grade =  
        Grade.B },  
    new Enrollment { StudentID = students.Single(s => s.LastName ==  
        "Alonso").ID, CourseID = courses.Single(c => c.Title == "Trigonometry").CourseID, Grade  
        = Grade.B },  
    new Enrollment { StudentID = students.Single(s => s.LastName ==  
        "Alonso").ID, CourseID = courses.Single(c => c.Title == "Composition").CourseID, Grade  
        = Grade.B },  
    new Enrollment { StudentID = students.Single(s => s.LastName ==  
        "Anand").ID, CourseID = courses.Single(c => c.Title == "Chemistry").CourseID },  
    new Enrollment { StudentID = students.Single(s => s.LastName ==  
        "Anand").ID, CourseID = courses.Single(c => c.Title == "Microeconomics").CourseID, Grade  
        = Grade.B },  
    new Enrollment { StudentID = students.Single(s => s.LastName ==  
        "Barzdukas").ID, CourseID = courses.Single(c => c.Title == "Chemistry").CourseID, Grade  
        = Grade.B },  
    new Enrollment { StudentID = students.Single(s => s.LastName == "Li").ID,  
        CourseID = courses.Single(c => c.Title == "Composition").CourseID, Grade = Grade.B },  
    new Enrollment { StudentID = students.Single(s => s.LastName ==  
        "Justice").ID, CourseID = courses.Single(c => c.Title == "Literature").CourseID, Grade =  
        Grade.B } };
```

- 1. Khởi tạo project**
- 2. CRUD**
- 3. Sắp xếp, lọc, phân trang**
- 4. Migration**
- 5. Tạo model dữ liệu liên quan đến nhiều bảng**

- **Cập nhật dữ liệu**

- Tạo class *Data/DbInitializer.cs*

```
foreach (Enrollment e in enrollments)
{
    var enrollmentInDataBase = context.Enrollments.Where(
        s => s.Student.ID == e.StudentID &&
            s.Course.CourseID == e.CourseID).SingleOrDefault();
    if (enrollmentInDataBase == null)
    {
        context.Enrollments.Add(e);
    }
}
context.SaveChanges();
}
```

- Tiến hành Add migration
- Cập nhật lại dữ liệu
- Chạy thử

- 1. Khởi tạo project
- 2. CRUD
- 3. Sắp xếp, lọc, phân trang
- 4. Migration
- 5. Tạo model dữ liệu liên quan đến nhiều bảng
- 6. Đọc dữ liệu liên quan

Courses

Create New

Number	Title	Credits	Department	
1045	Calculus	4	Mathematics	Edit Details Delete
1050	Calculus	4	Mathematics	
2021	Composition	3	English	

Instructors

Create New

Last Name	First Name	Hire Date	Office	Courses
Abercrombie	Karen	1989-08-05	300	
Fakhouri	Rami	1995-07-16	201	
Harui	Patricia	1990-09-17	202	

Courses Taught by Selected Instructor

	Number	Title	Department
Select	2021	Composition	English
Select	2042	Literature	English

Students Enrolled in Selected Course

Name	Grade
Alonso, Meredith	B
Li, Yan	B

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng
6. Đọc dữ liệu liên quan

• Các cách load dữ liệu liên quan

- Eager loading: sử dụng *Include*, *ThenInclude*

```
var departments = _context.Departments.Include(d => d.Courses);  
foreach (Department d in departments)  
{  
    foreach (Course c in d.Courses)  
    {  
        courseList.Add(d.Name + c.Title);  
    }  
}
```

Query: all Department entities and related Course entities

- Hoặc

```
var departments = _context.Departments;  
foreach (Department d in departments)  
{  
    _context.Courses.Where(c => c.DepartmentID == d.DepartmentID).Load();  
    foreach (Course c in d.Courses)  
    {  
        courseList.Add(d.Name + c.Title);  
    }  
}
```

Query: all Department rows

Query: Course rows related to Department d

- Explicit loading:

```
var departments = _context.Departments;  
foreach (Department d in departments)  
{  
    _context.Entry(d).Collection(p => p.Courses).Load();  
    foreach (Course c in d.Courses)  
    {  
        courseList.Add(d.Name + c.Title);  
    }  
}
```

Query: all Department rows

Query: Course rows related to Department d

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng
6. Đọc dữ liệu liên quan

• Tạo trang Courses

- Trong **Solution Explorer**, click phải vào thư mục **Controllers**, chọn **Add -> New Scaffolded Item**
- Trong hộp thoại Add Scaffold:
 - Chọn **MVC controller with views, using Entity Framework**.
 - Chọn Add, hộp thoại **Add MVC Controller with views, using Entity Framework** xuất hiện

The screenshot shows the 'Add Controller' dialog box. The 'Model class' dropdown is set to 'Course (ContosoUniversity.Models)'. The 'Data context class' dropdown is set to 'SchoolContext (ContosoUniversity.Data)'. The 'Views' section has three checked options: 'Generate views', 'Reference script libraries', and 'Use a layout page'. The 'Controller name' field is set to 'CoursesController'. The 'Add' button is highlighted.

- Mục Model class: chọn Course
- Mục Data context class: chọn SchoolContext
- Chọn nút Add

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng
6. Đọc dữ liệu liên quan

• Tạo trang Courses

- Mở *CoursesController.cs*, sửa lại phương thức *Index*

```
public async Task<IActionResult> Index()
{
    var courses = _context.Courses
        .Include(c => c.Department)
        .AsNoTracking();
    return View(await courses.ToListAsync());
}
```

- Mở *Views/Courses/Index.cshtml*, chỉnh sửa lại

```
<table class="table">
<thead>
    <tr>
    <th> @Html.DisplayNameFor(model => model.CourseID) </th>
    .....
<tbody>
    @foreach (var item in Model) {
    <tr>
        <td> @Html.DisplayFor(modelItem => item.CourseID) </td>
        .....
        <td> @Html.DisplayFor(modelItem => item.Department.Name) </td>
        <td>
            <a asp-action="Edit" asp-route-id="@item.CourseID">Edit
            </a> |
```


1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng
6. Đọc dữ liệu liên quan

- **Tạo trang Instructors**

- Tương tự tạo trang Instructor bằng Scaffold
- Trong thư mục *SchoolViewModel*, tạo class *InstructorIndexData.cs*

```
public class InstructorIndexData
{
    public IEnumerable<Instructor> Instructors { get; set; }
    public IEnumerable<Course> Courses { get; set; }
    public IEnumerable<Enrollment> Enrollments { get; set; }
}
```

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng
6. Đọc dữ liệu liên quan

• Tạo trang Instructors

- Mở *InstructorsController.cs*, chỉnh sửa lại

```
public async Task<IActionResult> Index(int? id, int? courseID) {  
    var viewModel = new InstructorIndexData();  
    viewModel.Instructors = await _context.Instructors  
        .Include(i => i.OfficeAssignment) .Include(i => i.CourseAssignments)  
        .ThenInclude(i => i.Course) .ThenInclude(i => i.Enrollments)  
        .ThenInclude(i => i.Student)  
        .Include(i => i.CourseAssignments)  
        .ThenInclude(i => i.Course) .ThenInclude(i => i.Department)  
        .AsNoTracking()  
        .OrderBy(i => i.LastName)  
        .ToListAsync();  
    if (id != null) {  
        ViewData["InstructorID"] = id.Value;  
        Instructor instructor = viewModel.Instructors.Where(  
            i => i.ID == id.Value).Single();  
        viewModel.Courses = instructor.CourseAssignments  
            .Select(s => s.Course);  
    }  
    if (courseID != null) {  
        ViewData["CourseID"] = courseID.Value;  
        viewModel.Enrollments = viewModel.Courses.Where(  
            x => x.CourseID == courseID).Single().Enrollments;  
    }  
    return View(viewModel);  
}
```

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng
6. Đọc dữ liệu liên quan

• **Chỉnh sửa view của Instructor Index**

- Mở *Views/Instructors/Index.cshtml*, chỉnh sửa lại

```
@model PXU_MVC_EFCore.Models.SchoolViewModels.InstructorIndexData
@{
    ViewData["Title"] = "Instructors";
}
<h2>Instructors</h2>
<p>
    <a asp-action="Create">Create New</a>
</p>
<table class="table">
    <thead>
        <tr>
            <th>Last Name</th>
            <th>First Name</th>
            <th>Hire Date</th>
            <th>Office</th>
            <th>Courses</th>
            <th></th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model.Instructors) {
            string selectedRow = "";
            if (item.ID == (int?)ViewData["InstructorID"]) {
                selectedRow = "table-success";
            }
            <tr class="@selectedRow">
                <td> @Html.DisplayFor(modelItem => item.LastName) </td>
```

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng
6. Đọc dữ liệu liên quan

• **Chỉnh sửa view của Instructor Index**

- Mở *Views/Instructors/Index.cshtml*, chỉnh sửa lại

```
<td> @Html.DisplayFor(modelItem => item.FirstMidName) </td>
<td> @Html.DisplayFor(modelItem => item.HireDate) </td>
<td>
    @if (item.OfficeAssignment != null) {
        @item.OfficeAssignment.Location }
</td>
<td>
    @foreach (var course in item.CourseAssignments) {
        @course.Course.CourseID @course.Course.Title <br />
    }
</td>
<td>
    <a asp-action="Index" asp-route-id="@item.ID">Select</a>
    | <a asp-action="Edit" asp-route-id="@item.ID">Edit</a>
    | <a asp-action="Details" asp-route-id="@item.ID">Details
</a>
    | <a asp-action="Delete" asp-route-id="@item.ID">Delete</a>
</td>
</tr>
}
</tbody>
</table>
```

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng
6. Đọc dữ liệu liên quan

• Chỉnh sửa view của Instructor Index

- Tiếp tục chỉnh sửa trong *Views/Instructors/Index.cshtml*

```
@if (Model.Courses != null) {  
    <h3>Courses Taught by Selected Instructor</h3>  
    <table class="table">  
        <tr>  
            <th></th>  
            <th>Number</th>  
            <th>Title</th>  
            <th>Department</th>  
        </tr>  
        @foreach (var item in Model.Courses) {  
            string selectedRow = "";  
            if (item.CourseID == (int?)ViewData["CourseID"]) {  
                selectedRow = "success";  
            }  
            <tr class="@selectedRow">  
                <td> @Html.ActionLink("Select", "Index", new {courseID=  
item.CourseID })  
                </td>  
                <td> @item.CourseID </td>  
                <td> @item.Title </td>  
                <td> @item.Department.Name </td>  
            </tr>  
        }  
    </table>  
}
```

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng
6. Đọc dữ liệu liên quan

• **Chỉnh sửa view của Instructor Index**

- Tiếp tục chỉnh sửa trong *Views/Instructors/Index.cshtml*

```
@if (Model.Enrollments != null) {  
    <h3> Students Enrolled in Selected Course </h3>  
    <table class="table">  
        <tr>  
            <th>Name</th>  
            <th>Grade</th>  
        </tr>  
        @foreach (var item in Model.Enrollments) {  
            <tr>  
                <td> @item.Student.FullName </td>  
                <td> @Html.DisplayFor(modelItem => item.Grade) </td>  
            </tr>  
        }  
    </table>  
}
```

Chương 4. MVC với Entity Framework Core

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng
6. Đọc dữ liệu liên quan
7. Cập nhật dữ liệu có liên quan

Edit

Course

Number
1000

Title

Credits

Department

Edit

Instructor

Last Name

First Name

Hire Date

Office Location

☐ 1000 Algebra 2 ☐ 1045 Calculus ☐ 1050 Chemistry
☒ 2021 Composition ☒ 2042 Literature ☐ 3141 Trigonometry
☐ 4022 Microeconomics ☐ 4041 Macroeconomics

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng
6. Đọc dữ liệu liên quan
7. Cập nhật dữ liệu có liên quan

- **Tùy chỉnh trang Courses**

- Trong *CoursesController.cs*, xóa phương thức Create và Edit và thay thế bằng đoạn code sau:

```
private void PopulateDepartmentsDropDownList(object
selectedDepartment = null)
{
    var departmentsQuery = from d in _context.Departments
                           orderby d.Name
                           select d;
    ViewBag.DepartmentID = new SelectList(departmentsQuery
                                           .AsNoTracking(),
                                           "DepartmentID", "Name",
                                           selectedDepartment);
}
```


1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng
6. Đọc dữ liệu liên quan
7. Cập nhật dữ liệu có liên quan

• Tùy chỉnh trang Courses

- Trong *CoursesController.cs*, xóa phương thức Create và Edit và thay thế bằng đoạn code sau:

```
public IActionResult Create()
{
    PopulateDepartmentsDropDownList();
    return View();
}

[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Create([Bind("CourseID,Credits,
DepartmentID,Title")] Course course)
{
    if (ModelState.IsValid)
    {
        _context.Add(course);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    PopulateDepartmentsDropDownList(course.DepartmentID);
    return View(course);
}
```

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng
6. Đọc dữ liệu liên quan
7. Cập nhật dữ liệu có liên quan

- **Tùy chỉnh trang Courses**

- Trong *CoursesController.cs*, xóa phương thức Create và Edit và thay thế bằng đoạn code sau:

```
public async Task<IActionResult> Edit(int? id)
{
    if (id == null) {
        return NotFound(); }
    var course = await _context.Courses
        .AsNoTracking()
        .FirstOrDefaultAsync(m => m.CourseID == id);
    if (course == null) {
        return NotFound(); }
    PopulateDepartmentsDropDownList(course.DepartmentID);
    return View(course);
}
```

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng
6. Đọc dữ liệu liên quan
7. Cập nhật dữ liệu có liên quan

• Tùy chỉnh trang Courses

- Trong *CoursesController.cs*, xóa phương thức Create và Edit và thay thế bằng đoạn code sau:

```
[HttpPost, ActionName("Edit")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> EditPost(int? id) {
    if (id == null) { return NotFound(); }
    var courseToUpdate = await _context.Courses
        .FirstOrDefaultAsync(c => c.CourseID == id);
    if (await TryUpdateModelAsync<Course>(courseToUpdate,
        "", c => c.Credits, c => c.DepartmentID, c => c.Title))
    {
        try {
            await _context.SaveChangesAsync(); }
        catch (DbUpdateException /* ex */) {
            //Log the error (uncomment ex variable name and write
a log.)
            ModelState.AddModelError("", "Unable to save changes.
" + "Try again, and if the problem persists, " + "see your system
administrator.");
        }
        return RedirectToAction(nameof(Index));
    }
    PopulateDepartmentsDropDownList(courseToUpdate.DepartmentID);
    return View(courseToUpdate);
}
```

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng
6. Đọc dữ liệu liên quan
7. Cập nhật dữ liệu có liên quan

- **Cập nhật lại các views của Course:**

- *Views/Courses/Create.cshtml*: thay thế DepartmentID bằng thẻ sau

```
<div class="form-group">
  <label asp-for="Department" class="control-label"></label>
  <select asp-for="DepartmentID" class="form-control" asp-
items="ViewBag.DepartmentID">
    <option value="">-- Select Department --</option>
  </select>
  <span asp-validation-for="DepartmentID" class="text-danger" />
</div>
```

- Làm tương tự với *Views/Courses/Edit.cshtml*
- Cũng trong *Views/Courses/Edit.cshtml*, trước thẻ Title, thêm thẻ

```
<div class="form-group">
  <label asp-for="CourseID" class="control-label"></label>
  <div>@Html.DisplayFor(model => model.CourseID)</div>
</div>
```

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng
6. Đọc dữ liệu liên quan
7. Cập nhật dữ liệu có liên quan

- **Cập nhật lại các views của Course:**

- *Views/Courses/Delete.cshtml*: thêm CourseID trước thẻ Title và thay thẻ DepartmentID bằng thẻ DepartmentName

```
<dt class="col-sm-2">
    @Html.DisplayNameFor(model => model.CourseID)
</dt>
<dd class="col-sm-10">
    @Html.DisplayFor(model => model.CourseID)
</dd>

....
<dd class="col-sm-10">
    @Html.DisplayFor(model => model.Department.Name)
</dd>
```

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng
6. Đọc dữ liệu liên quan
7. Cập nhật dữ liệu có liên quan

- **Tùy chỉnh trang Instructor:**

- *InstructorsController.cs*: thay đổi phương thức `HttpGet` `Edit` để load thể thực thể `OfficeAssignment`

```
public async Task<IActionResult> Edit(int? id)
{
    if (id == null) {
        return NotFound(); }
    var instructor = await _context.Instructors
        .Include(i => i.OfficeAssignment)
        .AsNoTracking()
        .FirstOrDefaultAsync(m => m.ID == id);
    if (instructor == null) {
        return NotFound(); }
    return View(instructor);
}
```

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng
6. Đọc dữ liệu liên quan
7. Cập nhật dữ liệu có liên quan

• Tùy chỉnh trang Instructor:

- *InstructorsController.cs*: Chỉnh sửa lại HttpPost Edit

```
[HttpPost, ActionName("Edit")] [ValidateAntiForgeryToken]
public async Task EditPost(int? id) {
    if (id == null) { return NotFound(); }
    var instructorToUpdate = await _context.Instructors
        .Include(i => i.OfficeAssignment) .FirstOrDefaultAsync(s => s.ID == id);
    if (await TryUpdateModelAsync<Instructor>(
        instructorToUpdate, "", i => i.FirstMidName,
        i => i.LastName, i => i.HireDate, i => i.OfficeAssignment))
    {
        if (String.IsNullOrEmpty(instructorToUpdate
            .OfficeAssignment?.Location))
        {
            instructorToUpdate.OfficeAssignment = null;
        }
        try {
            await _context.SaveChangesAsync();
        } catch (DbUpdateException /* ex */) {
            //Log the error (uncomment ex variable name and write a log.)
            ModelState.AddModelError("", "Unable to save changes. " + "Try
again, and if the problem persists, " + "see your system administrator.");
        }

        return RedirectToAction(nameof(Index));
    }
    return View(instructorToUpdate);
}
```


1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng
6. Đọc dữ liệu liên quan
7. Cập nhật dữ liệu có liên quan

- **Tùy chỉnh trang Instructor:**

- *Views/Instructors/Edit.cshtml*: thêm một trường để cập nhật Office Location vào phần cuối, trước nút Save

```
<div class="form-group">
    <label asp-for="OfficeAssignment.Location" class="control-label"></label>
    <input asp-for="OfficeAssignment.Location" class="form-control" />
    <span asp-validation-for="OfficeAssignment.Location" class="text-danger" />
</div>
```


1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng
6. Đọc dữ liệu liên quan
7. Cập nhật dữ liệu có liên quan

• Thêm các khóa học vào trang Edit:

- Tạo class *AssignedCourseData.cs* trong thư mục *SchoolViewModels*

```
public class AssignedCourseData
{
    public int CourseID { get; set; }
    public string Title { get; set; }
    public bool Assigned { get; set; }
}
```

- Trong *InstructorsController.cs*, chỉnh lại phần Edit

```
public async Task<IActionResult> Edit(int? id) {
    if (id == null) {
        return NotFound(); }
    var instructor = await _context.Instructors
        .Include(i => i.OfficeAssignment)
        .Include(i => i.CourseAssignments)
        .ThenInclude(i => i.Course)
        .AsNoTracking()
        .FirstOrDefaultAsync(m => m.ID == id);
    if (instructor == null) {
        return NotFound(); }
    PopulateAssignedCourseData(instructor);
    return View(instructor);
}
```

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng
6. Đọc dữ liệu liên quan
7. Cập nhật dữ liệu có liên quan

- **Thêm các khóa học vào trang Edit:**

- Trong *InstructorsController.cs*, chỉnh lại phần Edit

```
private void PopulateAssignedCourseData(Instructor instructor)
{
    var allCourses = _context.Courses;
    var instructorCourses = new HashSet<int>(
        instructor.CourseAssignments.Select(c => c.CourseID));
    var viewModel = new List<AssignedCourseData>();
    foreach (var course in allCourses)
    {
        viewModel.Add(new AssignedCourseData
        {
            CourseID = course.CourseID,
            Title = course.Title,
            Assigned = instructorCourses.Contains(course.CourseID)
        });
    }
    ViewData["Courses"] = viewModel;
}
```

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng
6. Đọc dữ liệu liên quan
7. Cập nhật dữ liệu có liên quan

- **Thêm các khóa học vào trang Edit:**

- Trong *InstructorsController.cs*, cập nhật lại EditPost

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int? id, string[] selectedCourses) {
    if (id == null) { return NotFound(); }
    var instructorToUpdate = await _context.Instructors
        .Include(i => i.OfficeAssignment)
        .Include(i => i.CourseAssignments) .ThenInclude(i => i.Course)
        .FirstOrDefaultAsync(m => m.ID == id);
    if (await TryUpdateModelAsync<Instructor>(instructorToUpdate, "",
        i => i.FirstMidName, i => i.LastName, i => i.HireDate, i => i.OfficeAssignment)) {
        if (String.IsNullOrEmpty( instructorToUpdate.OfficeAssignment?.Location)) {
            instructorToUpdate.OfficeAssignment = null; }
        UpdateInstructorCourses(selectedCourses, instructorToUpdate);
        try {
            await _context.SaveChangesAsync();
        } catch (DbUpdateException /* ex */) {
            //Log the error (uncomment ex variable name and write a log.)
            ModelState.AddModelError("", "Unable to save changes. " + "Try again, and if
the problem persists, " + "see your system administrator.");
        }
        return RedirectToAction(nameof(Index));
    }
    UpdateInstructorCourses(selectedCourses, instructorToUpdate);
    PopulateAssignedCourseData(instructorToUpdate);
    return View(instructorToUpdate);
}
```

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng
6. Đọc dữ liệu liên quan
7. Cập nhật dữ liệu có liên quan

- **Thêm các khóa học vào trang Edit:**

- Trong *InstructorsController.cs*, cập nhật lại EditPost

```
private void UpdateInstructorCourses(string[] selectedCourses, Instructor
instructorToUpdate) {
    if (selectedCourses == null) {
        instructorToUpdate.CourseAssignments = new List<CourseAssignment>();
        return;
    }
    var selectedCoursesHS = new HashSet<string>(selectedCourses);
    var instructorCourses = new HashSet<int>
        (instructorToUpdate.CourseAssignments.Select(c=> c.Course.CourseID));
    foreach (var course in _context.Courses) {
        if (selectedCoursesHS.Contains(course.CourseID.ToString())) {
            if (!instructorCourses.Contains(course.CourseID)) {
                instructorToUpdate.CourseAssignments.Add(new CourseAssignment{
                    InstructorID = instructorToUpdate.ID,
                    CourseID = course.CourseID });
            }
        }
        else {
            if (instructorCourses.Contains(course.CourseID)) {
                CourseAssignment courseToRemove = instructorToUpdate
                    .CourseAssignments.FirstOrDefault(i => i.CourseID == course.CourseID);
                _context.Remove(courseToRemove);
            }
        }
    }
}
```

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng
6. Đọc dữ liệu liên quan
7. Cập nhật dữ liệu có liên quan

• Thêm các khóa học vào trang Edit:

- Trong *Views/Instructors/Edit.cshtml*, thêm thẻ Course vào sau thẻ Office và trước thẻ Save

```
<div class="form-group">
  <div class="col-md-offset-2 col-md-10">
    <table>
      <tr>
        @{
          int cnt = 0;
          List<PXU_MVC_EFCore.Models.SchoolViewModels.AssignedCourseData> courses
= ViewBag.Courses;
          foreach (var course in courses) {
            if (cnt++ % 3 == 0) {
              @:</tr><tr>
            }
            @:<td>
              <input type="checkbox" name="selectedCourses"
                value="@course.CourseID"
                @(Html.Raw(course.Assigned ? "checked=\"checked\" : \"\") ) />
              @course.CourseID @: @course.Title
            @:</td>
          }
        @:</tr>
      }
    </table>
  </div>
</div>
```

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng
6. Đọc dữ liệu liên quan
7. Cập nhật dữ liệu có liên quan

- **Cập nhật trang Delete:**

- Trong *InstructorsController.cs*, chỉnh lại DeleteConfirmed

```
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> DeleteConfirmed(int id)
{
    Instructor instructor = await _context.Instructors
        .Include(i => i.CourseAssignments)
        .SingleOrDefault(i => i.ID == id);
    var departments = await _context.Departments
        .Where(d => d.InstructorID == id)
        .ToListAsync();
    departments.ForEach(d => d.InstructorID = null);
    _context.Instructors.Remove(instructor);
    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}
```

1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng
6. Đọc dữ liệu liên quan
7. Cập nhật dữ liệu có liên quan

- **Thêm office location và courses vào trang Create:**

- Trong *InstructorsController.cs*, cập nhật lại `HttpGet` và `HttpPost` **Create**

```
public IActionResult Create() {  
    var instructor = new Instructor();  
    instructor.CourseAssignments = new List<CourseAssignment>();  
    PopulateAssignedCourseData(instructor);  
    return View();  
}  
  
// POST: Instructors/Create  
[HttpPost] [ValidateAntiForgeryToken]  
public async Task<IActionResult> Create([Bind("FirstMidName,HireDate,LastName,OfficeAssignment")] Instructor instructor, string[] selectedCourses)  
{  
    if (selectedCourses != null) {  
        instructor.CourseAssignments = new List<CourseAssignment>();  
        foreach (var course in selectedCourses) {  
            var courseToAdd = new CourseAssignment {  
                InstructorID = instructor.ID, CourseID = int.Parse(course) };  
            instructor.CourseAssignments.Add(courseToAdd);  
        }  
    }  
    if (ModelState.IsValid) {  
        _context.Add(instructor);  
        await _context.SaveChangesAsync();  
        return RedirectToAction(nameof(Index));  
    }  
    PopulateAssignedCourseData(instructor);  
    return View(instructor);  
}
```


1. Khởi tạo project
2. CRUD
3. Sắp xếp, lọc, phân trang
4. Migration
5. Tạo model dữ liệu liên quan đến nhiều bảng
6. Đọc dữ liệu liên quan
7. Cập nhật dữ liệu có liên quan

• Thêm office location và courses vào trang Create:

- Trong *Views/Instructor/Create.cshtml*, thêm ô để gõ office location và các checkbox cho các khóa học trước nút Submit

```
<div class="form-group">
    <label asp-for="OfficeAssignment.Location" class="control-label"></label>
    <input asp-for="OfficeAssignment.Location" class="form-control" />
    <span asp-validation-for="OfficeAssignment.Location" class="text-danger" />
</div>
<div class="form-group">
    <div class="col-md-offset-2 col-md-10">
        <table>
            <tr>
                @{
                    int cnt = 0;
                    List<PXU_MVC_EFCore.Models.SchoolViewModels.AssignedCourseData>
courses = ViewBag.Courses;
                    foreach (var course in courses) {
                        if (cnt++ % 3 == 0) { @:</tr><tr> }
                        @:<td>
                            <input type="checkbox" name="selectedCourses"
                                value="@course.CourseID"
                                @(Html.Raw(course.Assigned ? "checked=\"checked\"" :
"")) />
                            @course.CourseID @: @course.Title
                        @:</td>
                    }
                @:</tr>
            }
        </table>
    </div> </div>
```


Thank you!