

*Bài giảng tích hợp:*

# ENTITY FRAMEWORK CORE

Faculty of IT

Email: [smdat@hueic.edu.vn](mailto: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

- **Tạo project**

- Tạo project mới, chọn **ASP.NET Core Web Application (.NET 5.0)**
- Thiết lập style cho site: *Pages/Shared/\_Layout.cshtml*

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - Contoso University</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.css" />
    <link rel="stylesheet" href="~/css/site.css" />
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">
            <div class="container">
                <a class="navbar-brand" asp-area="" asp-page="/Index">Contoso
University</a>
                <button class="navbar-toggler" type="button" data-toggle="collapse" data-target=".navbar-collapse" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
            </div>
        </nav>
    </header>
</body>
</html>
```

### 1. Khởi tạo project

- Thiết lập style cho site

```
<div class="navbar-collapse collapse d-sm-inline-flex flex-sm-row-reverse">
  <ul class="navbar-nav flex-grow-1">
    <li class="nav-item">
      <a class="nav-link text-dark" asp-area="" asp-
page="/About">About</a>
    </li>
    <li class="nav-item">
      <a class="nav-link text-dark" asp-area="" asp-
page="/Students/Index">Students</a>
    </li>
    <li class="nav-item">
      <a class="nav-link text-dark" asp-area="" asp-
page="/Courses/Index">Courses</a>
    </li>
    <li class="nav-item">
      <a class="nav-link text-dark" asp-area="" asp-
page="/Instructors/Index">Instructors</a>
    </li>
    <li class="nav-item">
      <a class="nav-link text-dark" asp-area="" asp-
page="/Departments/Index">Departments</a>
    </li>
  </ul>
</div>
</div>
</nav>
```

### 1. Khởi tạo project

- Thiết lập style cho site

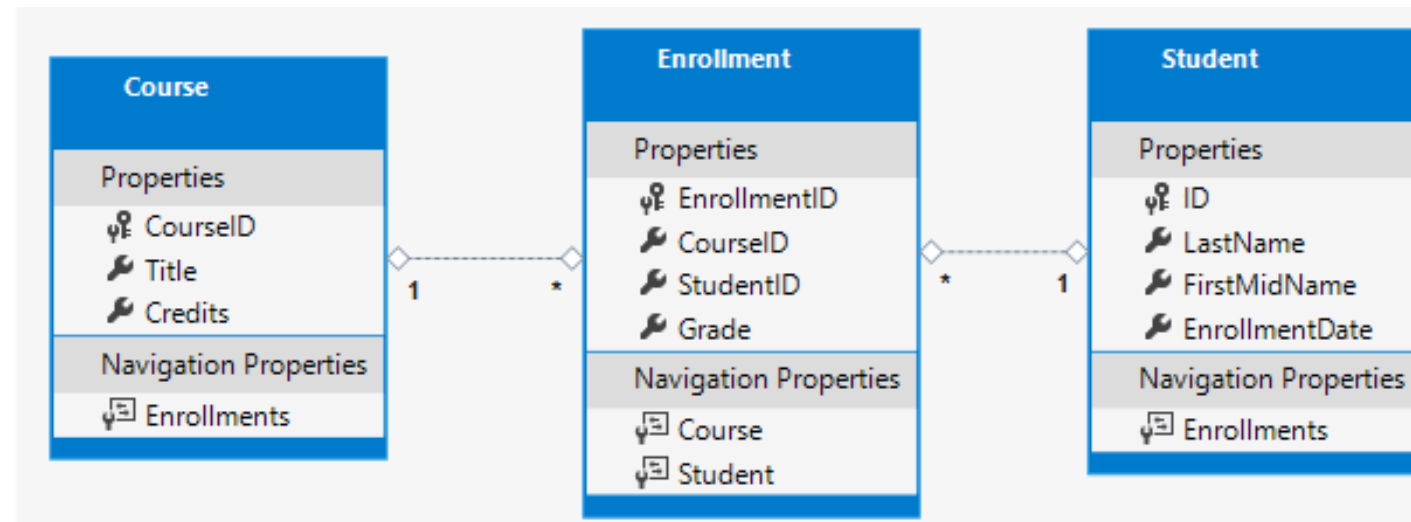
```
</header>
<div class="container">
  <main role="main" class="pb-3">
    @RenderBody()
  </main>
</div>
<footer class="border-top footer text-muted">
  <div class="container">
    &copy; 2021 - PXU University - <a asp-area="" asp-
page="/Privacy">Privacy</a>
  </div>
</footer>

<script src="~/lib/jquery/dist/jquery.js"></script>
<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.js"></script>
<script src="~/js/site.js" asp-append-version="true"></script>

@RenderSection("Scripts", required: false)
</body>
</html>
```

### 1. Khởi tạo project

- Tạo model:

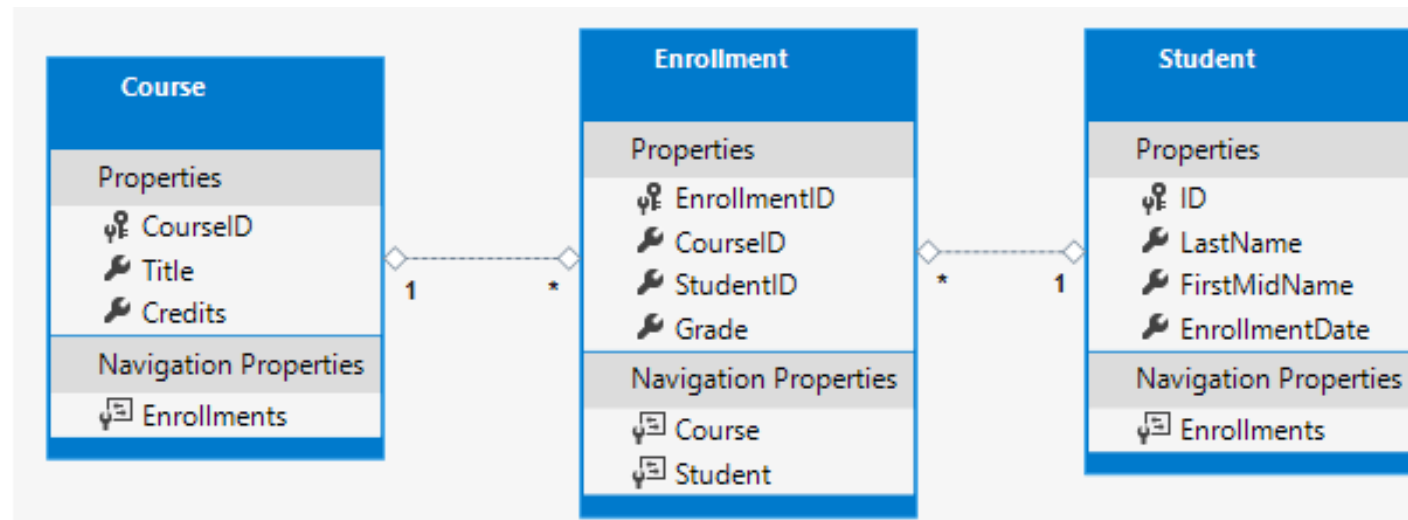


- Tạo thư mục Models
- Tạo class Student

```
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

- Tạo model:

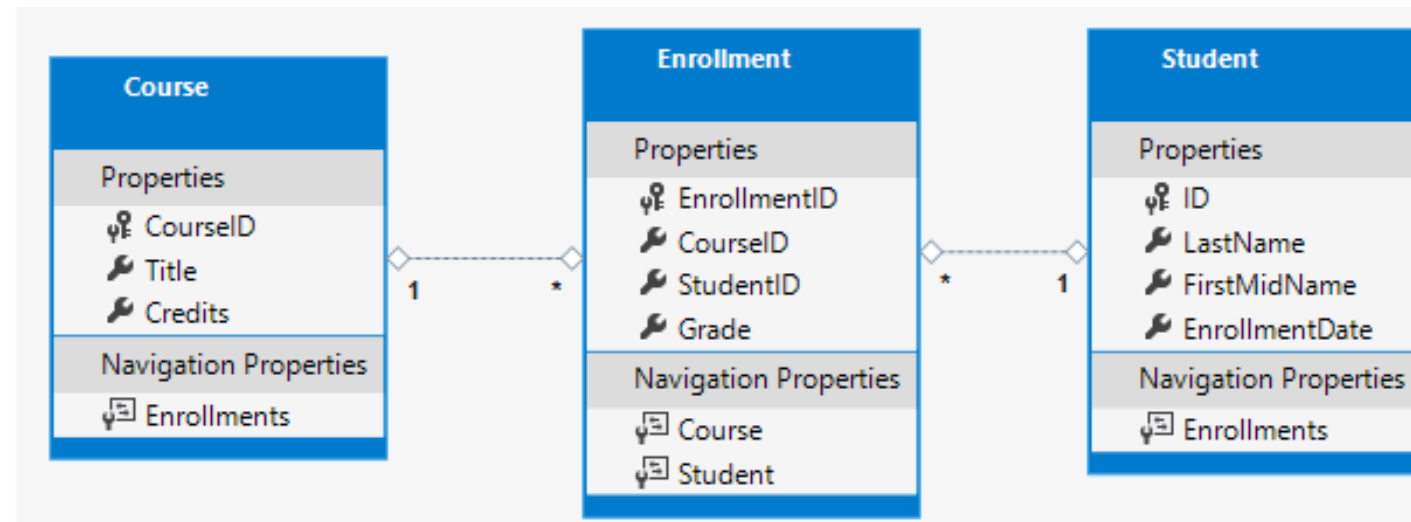


- Tạo class Course

```
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

- Tạo model:



- Tạo class 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; }
}
```



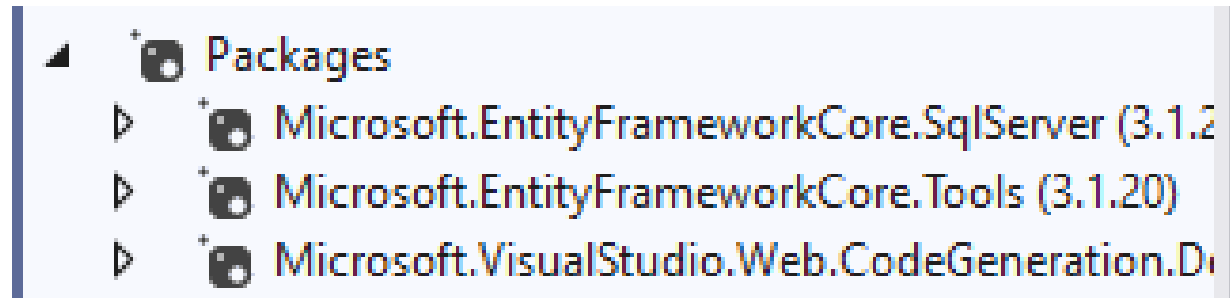
### 1. Khởi tạo project

- **Tạo trang Student bằng Scaffold:**
  - Tạo thư mục *Pages/Students*
  - Click phải vào thư mục *Pages/Students*, chọn **Add > New Scaffolded Item**.
    - Ở thư mục bên trái, chọn **Installed > Common > Razor Pages**
    - Chọn **Razor Pages using Entity Framework (CRUD) > ADD**.
  - Ở hộp hội thoại **Razor Pages using Entity Framework (CRUD)**
    - Ở **Model class**: chọn Student
    - Ở **Data context class**, chọn nút +, đổi tên *RazorPageEFCore.Data.SchoolContext*

### 1. Khởi tạo project

- **Tạo trang Student bằng Scaffold:**

- Sau khi cài xong, được các gói



- Tạo các trang Razor trong thư mục Pages/Students
  - *Create.cshtml* và *Create.cshtml.cs*
  - *Delete.cshtml* và *Delete.cshtml.cs*
  - *Details.cshtml* và *Details.cshtml.cs*
  - *Edit.cshtml* và *Edit.cshtml.cs*
  - *Index.cshtml* và *Index.cshtml.cs*
- Tạo file *SchoolContext.cs* trong thư mục *Data*
- Chỉ định context trong *Startup.cs*
- Thêm chuỗi kết nối vào trong *appsettings.json*

### 1. Khởi tạo project

- Cập nhật lại lớp *SchoolContext.cs*:

```
public class SchoolContext : DbContext
{
    public SchoolContext (DbContextOptions<SchoolContext> options)
        : base(options)
    {
    }
    public DbSet<Student> Students { get; set; }
    public DbSet<Enrollment> Enrollments { get; set; }
    public DbSet<Course> Courses { 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

- **Thêm bộ lọc ngoại lệ database:**
  - Trong **Package Manager Console**, gõ *Install-Package Microsoft.AspNetCore.Diagnostics.EntityFrameworkCore*
  - Cập nhật lại các phương thức *ConfigureServices*, *Configure*

```
public void ConfigureServices(IServiceCollection services)
{
    .....
    services.AddDatabaseDeveloperPageExceptionFilter();
}
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
        app.UseMigrationsEndPoint();
    }
    else
    {
        app.UseExceptionHandler("/Error");
        app.UseHsts();
    }
    .....
}
```

### 1. Khởi tạo project

- **Tạo database:**

- Cập nhật Program.cs để tạo database nếu nó không tồn tại

```
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>();
            context.Database.EnsureCreated();
            // DbInitializer.Initialize(context);
        }
        catch (Exception ex)
        {
            var logger = services.GetRequiredService<ILogger<Program>>();
            logger.LogError(ex, "An error occurred creating the DB.");
        }
    }
}
```

- Chạy thử

### 1. Khởi tạo project

- **Tạo database:**
  - Thêm dữ liệu
    - Tạo file *DbInitializer.cs* trong thư mục *Data*:

```
public static class DbInitializer
{
    public static void Initialize(SchoolContext context)
    {
        // Look for any students.
        if (context.Students.Any()) {
            return; // DB has been seeded
        }
        var students = new Student[] {
            new Student{FirstMidName="Carson", LastName="Alexander"},
            new Student{FirstMidName="Meredith", LastName="Alonso"},
            new Student{FirstMidName="Arturo", LastName="Anand"},
            new Student{FirstMidName="Gytis", LastName="Barzdulis"},
            new Student{FirstMidName="Yan", LastName="Li", EnrollmentDate=DateTime.Now},
            new Student{FirstMidName="Peggy", LastName="Justice"},
            new Student{FirstMidName="Laura", LastName="Norman"},
            new Student{FirstMidName="Nino", LastName="Olivetto"}
        };
        context.Students.AddRange(students);
        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}
        };
        context.Courses.AddRange(courses);
        context.SaveChanges();
        var enrollments = new Enrollment[]
        {
            new Enrollment{StudentID=1, CourseID=1050, EnrollmentDate=DateTime.Now},
            new Enrollment{StudentID=1, CourseID=4022, EnrollmentDate=DateTime.Now},
            new Enrollment{StudentID=1, CourseID=4041, EnrollmentDate=DateTime.Now},
            new Enrollment{StudentID=2, CourseID=1050, EnrollmentDate=DateTime.Now},
            new Enrollment{StudentID=2, CourseID=4022, EnrollmentDate=DateTime.Now},
            new Enrollment{StudentID=3, CourseID=1050, EnrollmentDate=DateTime.Now},
            new Enrollment{StudentID=4, CourseID=1050, EnrollmentDate=DateTime.Now},
            new Enrollment{StudentID=5, CourseID=1050, EnrollmentDate=DateTime.Now},
            new Enrollment{StudentID=6, CourseID=1050, EnrollmentDate=DateTime.Now},
            new Enrollment{StudentID=7, CourseID=1050, EnrollmentDate=DateTime.Now}
        };
        context.Enrollments.AddRange(enrollments);
        context.SaveChanges();
    }
}
```

```
context.Courses.AddRange(courses);
context.SaveChanges();
var enrollments = new Enrollment[]
{
    new Enrollment{StudentID=1, CourseID=1050, EnrollmentDate=DateTime.Now},
    new Enrollment{StudentID=1, CourseID=4022, EnrollmentDate=DateTime.Now},
    new Enrollment{StudentID=1, CourseID=4041, EnrollmentDate=DateTime.Now},
    new Enrollment{StudentID=2, CourseID=1050, EnrollmentDate=DateTime.Now},
    new Enrollment{StudentID=2, CourseID=4022, EnrollmentDate=DateTime.Now},
    new Enrollment{StudentID=3, CourseID=1050, EnrollmentDate=DateTime.Now},
    new Enrollment{StudentID=4, CourseID=1050, EnrollmentDate=DateTime.Now},
    new Enrollment{StudentID=5, CourseID=1050, EnrollmentDate=DateTime.Now},
    new Enrollment{StudentID=6, CourseID=1050, EnrollmentDate=DateTime.Now},
    new Enrollment{StudentID=7, CourseID=1050, EnrollmentDate=DateTime.Now}
};
context.Enrollments.AddRange(enrollments);
context.SaveChanges();
```

### 1. Khởi tạo project

- **Tạo database:**

- Trong *Program.cs*, xóa // ở dòng *DbInitializer.Initialize(context);*
- Tiến hành xóa database hiện tại

### *Drop-Database -Confirm*

- Chạy lại ứng dụng.

localhost:44374/Students

PXU University About Students Courses Instructors Departments

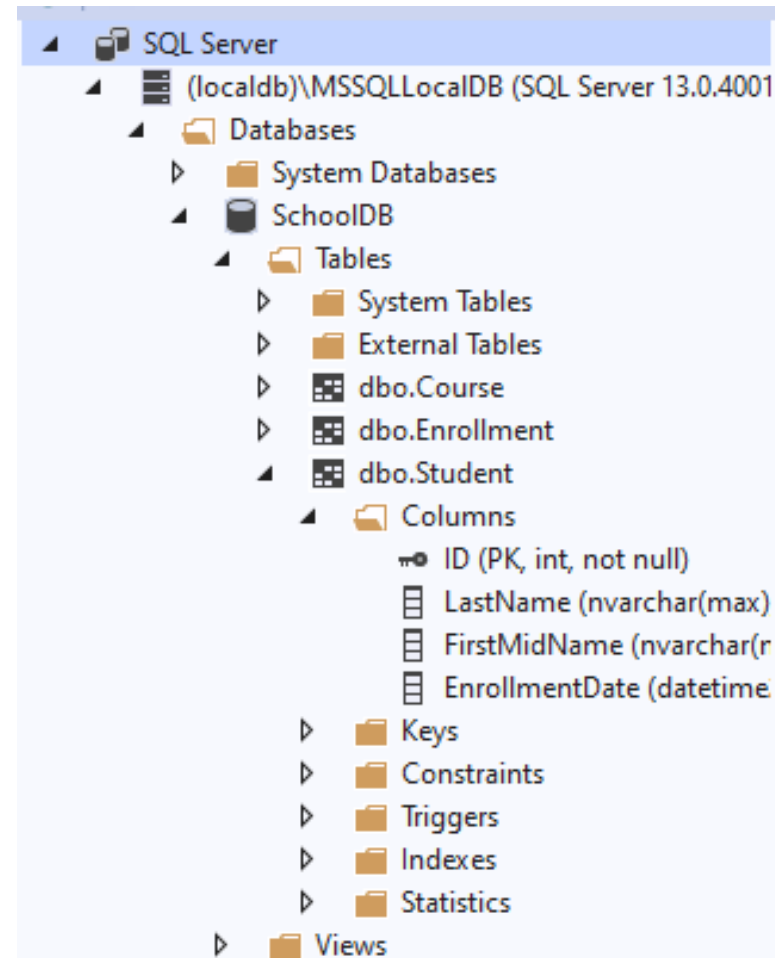
## Index

[Create New](#)

LastName	FirstMidName	EnrollmentDate	
Alexander	Carson	01/09/2019 12:00:00 SA	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Alonso	Meredith	01/09/2017 12:00:00 SA	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Anand	Arturo	01/09/2018 12:00:00 SA	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Barzdukas	Gytis	01/09/2017 12:00:00 SA	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Li	Yan	01/09/2017 12:00:00 SA	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>

### 1. Khởi tạo project

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





### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

- **Cập nhật trang Details:**

- Code scaffolded cho trang Students không hiển thị dữ liệu của Enrollment. Vì vậy cần hiển thị thêm thông tin của Enrollment trong trang Details.

- **Đọc bảng Enrollments**

- Để hiển thị dữ liệu ghi danh(**enrollments**) của sinh viên trên trang, thì dữ liệu trên bảng **Enrollments** phải được đọc. Ở trang *Details* chỉ hiển thị dữ liệu ở bảng **Student**, mà không có dữ liệu của **Enrollment**. Vì vậy cần hiển thị thêm

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

- AsNoTracking: được dùng cho trường hợp mà các thực thể trả về là không được cập nhật trong context hiện tại.

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

- **Cập nhật trang Details:**

- **Đọc bảng Enrollments**

- Ở Details.xml, thêm các thẻ để hiển thị thêm thông tin ghi danh

```
<dt class="col-sm-2">
    @Html.DisplayNameFor(model => model.Student.Enrollments)
</dt>
<dd class="col-sm-10">
    <table class="table">
        <tr>
            <th>Course Title</th>
            <th>Grade</th>
        </tr>
        @foreach (var item in Model.Student.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. Thêm, Xem, Sửa, Xóa

- Cập nhật trang Details:
  - Đọc bảng Enrollments
    - Ở Details.xml, thêm các thẻ để hiển thị thêm thông tin ghi danh

# Details

Student

LastName	Alexander	
FirstMidName	Carson	
EnrollmentDate	01/09/2019 12:00:00 SA	
Enrollments	Course Title	Grade
	Macroeconomics	B
	Microeconomics	C
	Chemistry	A

[Edit](#) | [Back to List](#)

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

## • Cập nhật trang Create:

```
public async Task<IActionResult> OnPostAsync()
{
    var emptyStudent = new Student();
    if (await TryUpdateModelAsync<Student>(
        emptyStudent,
        "student", // Prefix for form value.
        s => s.FirstMidName,
        s => s.LastName,
        s => s.EnrollmentDate))
    {
        _context.Students.Add(emptyStudent);
        await _context.SaveChangesAsync();
        return RedirectToPage("./Index");
    }
    return Page();
}
```

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

## • Cập nhật trang Edit:

```
public async Task<IActionResult> OnGetAsync(int? id)
{
    if (id == null)
    {
        return NotFound();
    }
    Student = await _context.Students.FindAsync(id);
    if (Student == null)
    {
        return NotFound();
    }
    return Page();
}

public async Task<IActionResult> OnPostAsync(int id)
{
    var studentToUpdate = await _context.Students.FindAsync(id);
    if (studentToUpdate == null)
    {
        return NotFound();
    }
    if (await TryUpdateModelAsync<Student>( studentToUpdate,
        "student", s => s.FirstMidName,
        s => s.LastName, s => s.EnrollmentDate))
    {
        await _context.SaveChangesAsync();
        return RedirectToPage("./Index");
    }
    return Page();
}
```

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

## • Cập nhật trang Delete:

```
public class DeleteModel : PageModel {
    private readonly PXU_RazorPageEFCore.Data.SchoolContext _context;
    private readonly ILogger<DeleteModel> _logger;

    public DeleteModel(PXU_RazorPageEFCore.Data.SchoolContext context,
        ILogger<DeleteModel> logger)
    {
        _context = context;
        _logger = logger;
    }
    [BindProperty]
    public Student Student { get; set; }
    public string ErrorMessage { get; set; }
    public async Task<IActionResult> OnGetAsync(int? id,
        bool? saveChangesError = false)
    {
        if (id == null)
            return NotFound();
        Student = await _context.Students
            .AsNoTracking()
            .FirstOrDefaultAsync(m => m.ID == id);
        if (Student == null)
            return NotFound();
        if (saveChangesError.GetValueOrDefault())
            ErrorMessage = String.Format("Delete {ID} failed. Try again", id);
        return Page();
    }
}
```

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

## • Cập nhật trang Delete:

```
public async Task<IActionResult> OnPostAsync(int? id)
{
    if (id == null)
        return NotFound();
    var student = await _context.Students.FindAsync(id);
    if (student == null)
        return NotFound();

    try
    {
        _context.Students.Remove(student);
        await _context.SaveChangesAsync();
        return RedirectToPage("./Index");
    }
    catch (DbUpdateException ex)
    {
        _logger.LogError(ex, ErrorMessage);
        return RedirectToAction("./Delete", new { id, saveChangesError = true });
    }
}
```

```
<h1>Delete</h1>
```

```
<p class="text-danger">@Model.ErrorMessage</p>
```

```
<h3>Are you sure you want to delete this?</h3>
```

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

- **Sắp xếp:**

- Vào trang *Pages/Students/Index.cshtml.cs*

```
public string NameSort { get; set; }
public string DateSort { get; set; }
public string CurrentFilter { get; set; }
public string CurrentSort { get; set; }
public IList<Student> Students { get; set; }

public async Task OnGetAsync(string sortOrder)
{
    // using System;
    NameSort = String.IsNullOrEmpty(sortOrder) ? "name_desc" : "";
    DateSort = sortOrder == "Date" ? "date_desc" : "Date";
    IQueryable<Student> studentsIQ = from s in _context.Students
                                     select s;

    switch (sortOrder)
    {
        case "name_desc": studentsIQ = studentsIQ.OrderByDescending(s => s.LastName);
                           break;
        case "Date": studentsIQ = studentsIQ.OrderBy(s => s.EnrollmentDate);
                           break;
        case "date_desc": studentsIQ = studentsIQ.OrderByDescending(s =>
s.EnrollmentDate);
                           break;
        default: studentsIQ = studentsIQ.OrderBy(s => s.LastName);
                           break;
    }
    Students = await studentsIQ.AsNoTracking().ToListAsync();
}
```



### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

- **Sắp xếp:**

- Vào trang *Pages/Students/Index.cshtml*

```
<table class="table">
  <thead>
    <tr>
      <th>
        <a asp-page="./Index" asp-route-sortOrder="@Model.NameSort">
          @Html.DisplayNameFor(model => model.Students[0].LastName)
        </a>
      </th>
      <th>
        @Html.DisplayNameFor(model => model.Students[0].FirstMidName)
      </th>
      <th>
        <a asp-page="./Index" asp-route-sortOrder="@Model.DateSort">
          @Html.DisplayNameFor(model => model.Students[0].EnrollmentDate)
        </a>
      </th>
    </tr>
  </thead>
  <tbody>
    @foreach (var item in Model.Students)
    {
      .....
    }
  </tbody>
</table>
```

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

## Lọc dữ liệu:

- Vào trang *Students/Index.cshtml.cs*, sửa lại hàm OnGetAysn

```
public async Task OnGetAsync(string sortOrder, string searchString)
{
    NameSort = String.IsNullOrEmpty(sortOrder) ? "name_desc" : "";
    DateSort = sortOrder == "Date" ? "date_desc" : "Date";

    CurrentFilter = searchString;
    IQueryable<Student> studentsIQ = from s in _context.Students
                                     select s;

    if (!String.IsNullOrEmpty(searchString))
    {
        studentsIQ = studentsIQ.Where(
            s => s.LastName.Contains(searchString)
            || s.FirstMidName.Contains(searchString));
    }
    .....
}
```

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

## Lọc dữ liệu:

- Vào trang *Students/Index.cshtml*

```
<p>
    <a asp-page="Create">Create New</a>
</p>

<form asp-page="./Index" method="get">
    <div class="form-actions no-color">
        <p>
            Find by name:
            <input type="text" name="SearchString" value="@Model.CurrentFilter" />
            <input type="submit" value="Search" class="btn btn-primary" /> |
            <a asp-page="./Index">Back to full List</a>
        </p>
    </div>
</form>
```

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

## Phân trang:

### • Tạo class PaginatedList

- Click phải vào project, chọn add -> class để tạo class

```
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 pageSize)
    {
        PageIndex = pageIndex;
        TotalPages = (int)Math.Ceiling(count / (double)pageSize);
        this.AddRange(items);
    }
    public bool HasPreviousPage => PageIndex > 1;
    public bool HasNextPage => 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. Thêm, Xem, Sửa, Xóa

3. Sắp xếp, lọc, phân trang, gom nhóm

### Phân trang:

- Thêm PageSize vào phần configuration
  - Thêm PageSize vào trong file *appsettings.json*

```
{
  "PageSize": 3,
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "SchoolContext":
      "Server=(localdb)\\mssqllocaldb;Database=SchoolTestDB;Trusted_Connection=True;MultipleActiveResultSets=true" } }
```

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

## Phân trang:

- Thêm phần Phân trang vào IndexModel

- Vào *Students/Index.cshtml.cs* để thêm phần phân trang

```
public class IndexModel : PageModel {
    private readonly SchoolContext _context;
    private readonly IConfiguration Configuration;
    public IndexModel(SchoolContext context, IConfiguration configuration) {
        _context = context;
        Configuration = configuration;
    }
    public string NameSort { get; set; }
    .....
    public string CurrentSort { get; set; }
    public PaginatedList<Student> Students { get; set; }
    public async Task OnGetAsync(string sortOrder, string currentFilter, string
searchString, int? pageIndex) {
        CurrentSort = sortOrder;
        NameSort = String.IsNullOrEmpty(sortOrder) ? "name_desc" : "";
        DateSort = sortOrder == "Date" ? "date_desc" : "Date";
        if (searchString != null)
            pageIndex = 1;
        else
            searchString = currentFilter;
        CurrentFilter = searchString;
        .....
        var pageSize = Configuration.GetValue("PageSize", 4);
        Students = await PaginatedList<Student>.CreateAsync(
            studentsIQ.AsNoTracking(), pageIndex ?? 1, pageSize);
    }
}
```

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

## Phân trang:

- Thay đổi code trong *Students/Index.cshtml* và tiến hành chạy thử

```
<a asp-page="./Index" asp-route-sortOrder="@Model.NameSort"
    asp-route-currentFilter="@Model.CurrentFilter">
    @Html.DisplayNameFor(model => model.Students[0].LastName)
</a>

...

<a asp-page="./Index" asp-route-sortOrder="@Model.DateSort"
    asp-route-currentFilter="@Model.CurrentFilter">
    @Html.DisplayNameFor(model => model.Students[0].EnrollmentDate)
</a>

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

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

## Grouping:

- Tạo một view model
  - Tạo thư mục *Models/SchoolViewModels*
  - Tạo class *EnrollmentDateGroup.cs* trong thư mục *SchoolViewModels*

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

- Tạo trang Razor có tên *About.cshtml* trong thư mục Pages

```
@page  
@model RazorPageEFCore.Pages.AboutModel  
@{  
    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.Students) {  
        <tr>  
            <td> @Html.DisplayFor(modelItem => item.EnrollmentDate) </td>  
            <td> @item.StudentCount </td>  
        </tr>  
    }  
</table>
```



### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

## Grouping:

- Cập nhật page model:
  - Cập nhật lại file *Pages/About.cshtml.cs* và chạy thử

```
public class AboutModel : PageModel
{
    private readonly SchoolContext _context;
    public AboutModel(SchoolContext context)
    {
        _context = context;
    }
    public IList<EnrollmentDateGroup> Students { get; set; }
    public async Task OnGetAsync()
    {
        IQueryable<EnrollmentDateGroup> data =
            from student in _context.Students
            group student by student.EnrollmentDate into dateGroup
            select new EnrollmentDateGroup()
            {
                EnrollmentDate = dateGroup.Key,
                StudentCount = dateGroup.Count()
            };
        Students = await data.AsNoTracking().ToListAsync();
    }
}
```

## Grouping:

1. Khởi tạo project
2. Thêm, Xem, Sửa, Xóa
3. Sắp xếp, lọc, phân trang, gom nhóm

### Student Body Statistics

#### Enrollment Date Students

9/1/2016	1
9/1/2017	3
9/1/2018	2
9/1/2019	2

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

- **Razor Pages với EF Core migrations trong ASP.NET Core**
  - Xóa CSDL: trong **Package Manager Console** gõ lệnh  
`Drop-Database`
  - Tạo một migration: trong **Package Manager Console** gõ lệnh  
`Add-Migration InitialCreate`  
`Update-Database`
  - Trong Program.cs, xóa dòng *context.Database.EnsureCreated()*; để có thể tạo một bảng lưu lịch sử migrations và EnsureCreate không dùng được trong migrations

## Chương 3. Razor Page với Entity Framework Core

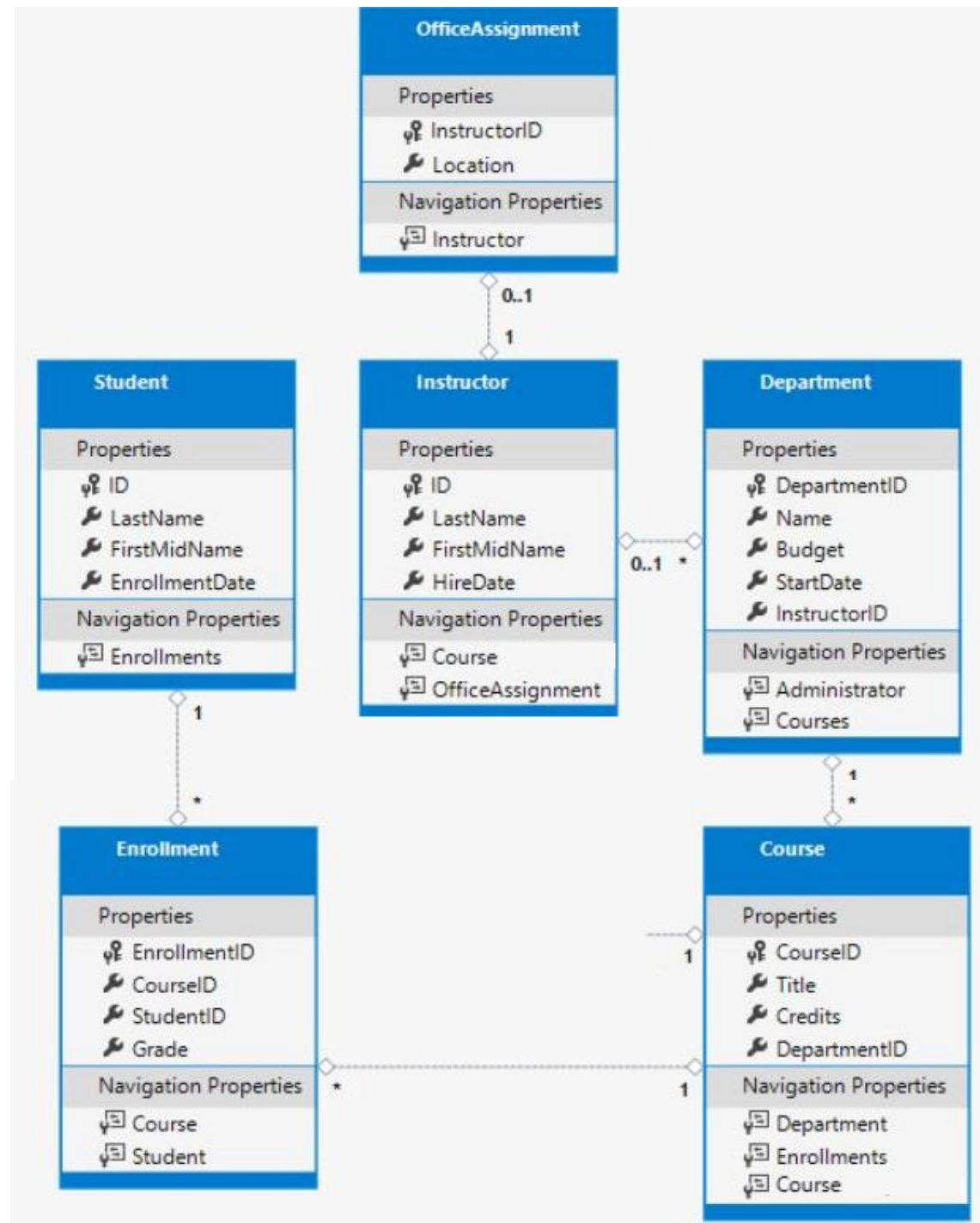
1. Khởi tạo project

2. Thêm, Xem, Sửa, Xóa

3. Sắp xếp, lọc, phân trang, gom nhóm

4. Migrations

5. Tạo một model dữ liệu phức tạp



### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

## • Thực thể Student

- Thay đổi code trong *Models/Student.cs*

```
public class Student
{
    public int ID { get; set; }
    [Required] [StringLength(50)] [Display(Name = "Last Name")]
    public string LastName { get; set; }
    [Required]
    [StringLength(50, ErrorMessage = "First name cannot be longer than 50 characters.")]
    [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. Thêm, Xem, Sửa, Xóa

3. Sắp xếp, lọc, phân trang, gom nhóm

4. Migrations

5. Tạo một model dữ liệu phức tạp

### • Tạo một migration

- Chạy thử và đến trang Student. Có một lỗi xảy ra. Do có attribute [Column] nên không tìm thấy cột FirstName, hiện tại trong database vẫn là cột FirstMidName

```
SqlException: Invalid column name 'FirstName'.  
There are pending model changes  
Pending model changes are detected in the following:  
  
SchoolContext
```

- Trong Package Manager Console (PMC), gõ

```
PowerShell  
  
Add-Migration ColumnFirstName  
Update-Database
```

- Xuất hiện cảnh báo

text

```
An operation was scaffolded that may result in the loss of data.  
Please review the migration for accuracy.
```

1. Khởi tạo project

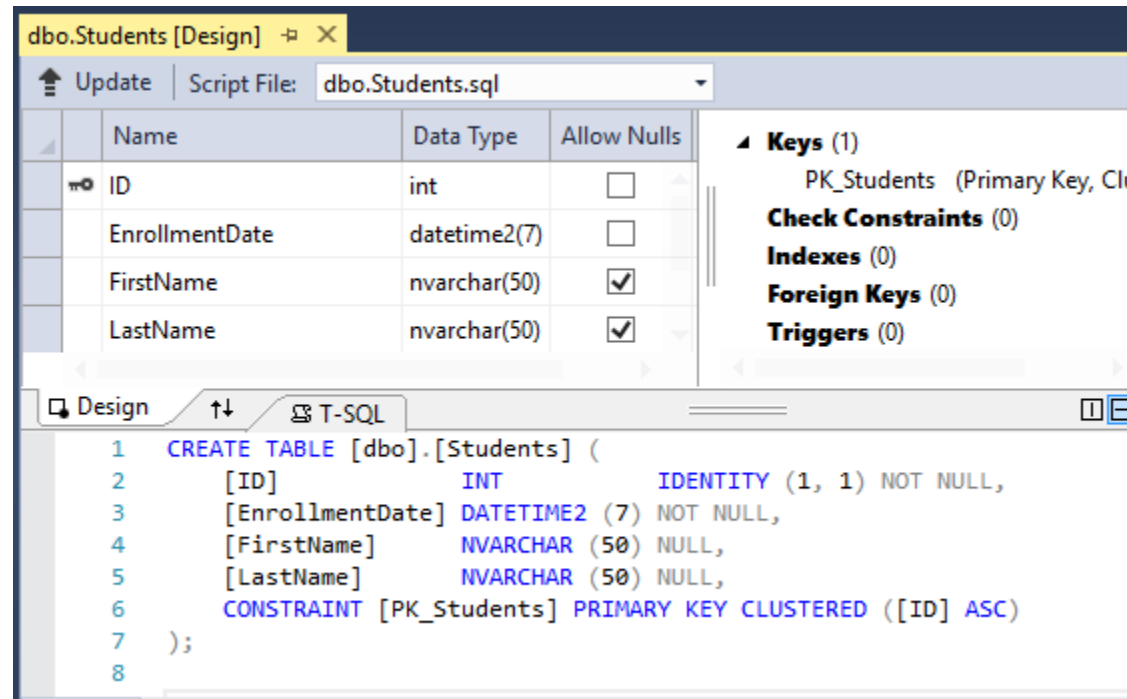
2. Thêm, Xem, Sửa, Xóa

3. Sắp xếp, lọc, phân trang, gom nhóm

4. Migrations

5. Tạo một model dữ liệu phức tạp

- Tạo một migration
  - Mở bảng Student





### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

## • Thực thể Instructor

- Tạo class *Models/Instructor.cs*

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

Instructor
Properties
ID
LastName
FirstMidName
HireDate
Navigation Properties
Course
OfficeAssignment



1. Khởi tạo project

2. Thêm, Xem, Sửa, Xóa

3. Sắp xếp, lọc, phân trang, gom nhóm

4. Migrations

5. Tạo một model dữ liệu phức tạp

## • Thực thể OfficeAssignment

- Tạo class *Models/OfficeAssignment.cs*

```
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; }
}
```

OfficeAssign...	
Properties	
ψ	InstructorID
🔑	Location
Navigation Properties	
ψ	Instructor

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

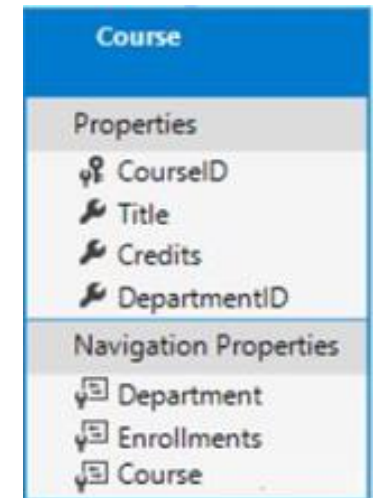
### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

## • Thực thể Course

- Cập nhật lại class *Models/Course.cs*

```
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<Instructor> Instructors { get; set; }
}
```



- Thuộc tính *[DatabaseGenerated(DatabaseGeneratedOption.None)]* chỉ định rằng PK được cung cấp bởi ứng dụng chứ không phải do cơ sở dữ liệu tạo ra.
- Mặc định, EF Core giả định rằng giá trị PK được tạo ra bởi database. Đối với thực thể Course, người dùng sẽ chỉ định PK. Ví dụ, một số khóa học, chẳng hạn như dãy số 1000 cho khoa Toán, một dãy số 2000 cho khoa tiếng Anh.

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm








### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

## • Thực thể Department

- Tạo class *Models/Department.cs*

```
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. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

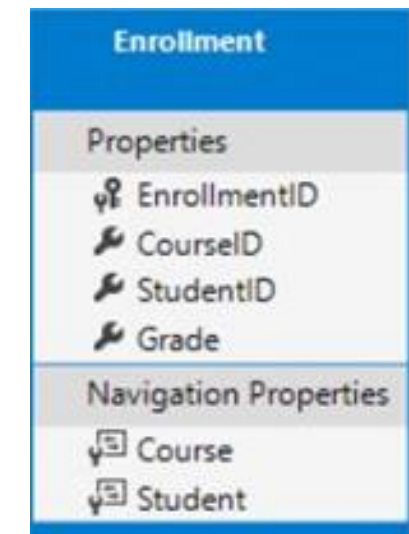
### 5. Tạo một model dữ liệu phức tạp

## • Thực thể Enrollment

- Cập nhật lại class *Models/Enrollment.cs*

```
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; }
}
```



### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

## • Cập nhật database context

### • Cập nhật lại class *Data/SchoolContext.cs*

```
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; }
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Course>().ToTable(nameof(Course))
            .HasMany(c => c.Instructors)
            .WithMany(i => i.Courses);
        modelBuilder.Entity<Student>().ToTable(nameof(Student));
        modelBuilder.Entity<Instructor>().ToTable(nameof(Instructor));
        modelBuilder.Entity<Department>()
            .HasOne(d => d.Administrator)
            .WithMany()
            .OnDelete(DeleteBehavior.Restrict);
    }
}
```

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

## • Cập nhật dữ liệu

### • Cập nhật lại class *Data/DbInitializer.cs*

```
public static class DbInitializer {
    public static void Initialize(SchoolContext context) {
        // Look for any students.
        if (context.Students.Any())
            return; // DB has been seeded

        var alexander = new Student { FirstMidName = "Carson", LastName =
"Alexander",
            EnrollmentDate = DateTime.Parse("2016-09-01") };
        var alonso = new Student { FirstMidName = "Meredith", LastName = "Alonso",
            EnrollmentDate = DateTime.Parse("2018-09-01") };
        var anand = new Student { FirstMidName = "Arturo", LastName = "Anand",
            EnrollmentDate = DateTime.Parse("2019-09-01") };
        var barzdukas = new Student { FirstMidName = "Gytis", LastName =
"Barzdukas",
            EnrollmentDate = DateTime.Parse("2018-09-01") };
        var li = new Student { FirstMidName = "Yan", LastName = "Li",
            EnrollmentDate = DateTime.Parse("2018-09-01") };
        var justice = new Student { FirstMidName = "Peggy", LastName = "Justice",
            EnrollmentDate = DateTime.Parse("2017-09-01") };
        var norman = new Student { FirstMidName = "Laura", LastName = "Norman",
            EnrollmentDate = DateTime.Parse("2019-09-01") };
        var olivetto = new Student { FirstMidName = "Nino", LastName = "Olivetto",
            EnrollmentDate = DateTime.Parse("2011-09-01") };
```



### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

## • Cập nhật dữ liệu

### • Cập nhật lại class *Data/DbInitializer.cs*

```
var abercrombie = new Instructor { FirstMidName = "Kim", LastName = "Abercrombie", HireDate = DateTime.Parse("1995-03-11") };
var fakhouri = new Instructor { FirstMidName = "Fadi", LastName = "Fakhouri", HireDate = DateTime.Parse("2002-07-06") };
var harui = new Instructor { FirstMidName = "Roger", LastName = "Harui", HireDate = DateTime.Parse("1998-07-01") };
var kapoor = new Instructor { FirstMidName = "Candace", LastName = "Kapoor", HireDate = DateTime.Parse("2001-01-15") };
var zheng = new Instructor { FirstMidName = "Roger", LastName = "Zheng", HireDate = DateTime.Parse("2004-02-12") };
var officeAssignments = new OfficeAssignment[] {
    new OfficeAssignment {
        Instructor = fakhouri, Location = "Smith 17" },
    new OfficeAssignment {
        Instructor = harui, Location = "Gowan 27" },
    new OfficeAssignment {
        Instructor = kapoor, Location = "Thompson 304" }, };
context.AddRange(officeAssignments);
var english = new Department {
    Name = "English", Budget = 350000,
    StartDate = DateTime.Parse("2007-09-01"),
    Administrator = abercrombie };
```

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

## • Cập nhật dữ liệu

### • Cập nhật lại class *Data/DbInitializer.cs*

```
var mathematics = new Department {Name = "Mathematics", Budget = 100000, StartDate = DateTime.Parse("2007-09-01"), Administrator = fakhouri };
var engineering = new Department { Name = "Engineering", Budget = 350000, StartDate = DateTime.Parse("2007-09-01"), Administrator = harui };
var economics = new Department { Name = "Economics", Budget = 100000, StartDate = DateTime.Parse("2007-09-01"), Administrator = kapoor };
var chemistry = new Course {CourseID = 1050, Title = "Chemistry", Credits = 3, Department = engineering, Instructors = new List<Instructor> { kapoor, harui } };
var microeconomics = new Course {CourseID = 4022, Title = "Microeconomics", Credits = 3, Department = economics, Instructors = new List<Instructor> { zheng } };
var macroeconomics = new Course {CourseID = 4041, Title = "Macroeconomics", Credits = 3, Department = economics, Instructors = new List<Instructor> { zheng } };
```



## Chương 3. Razor Page với Entity Framework Core

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

```
var calculus = new Course {CourseID = 1045, Title = "Calculus", Credits = 4,
    Department = mathematics,Instructors = new List<Instructor> {fakhouri} };
var trigonometry=new Course {CourseID= 3141, Title = "Trigonometry",Credits = 4,
    Department = mathematics, Instructors = new List<Instructor> { harui } };
var composition = new Course {CourseID=2021, Title = "Composition", Credits = 3,
    Department =english, Instructors = new List<Instructor> {abercrombie } };
var literature = new Course {CourseID = 2042, Title = "Literature", Credits = 4,
    Department = english, Instructors = new List<Instructor> { abercrombie } };
var enrollments = new Enrollment[] {
    new Enrollment {Student = alexander, Course = chemistry, Grade = Grade.A },
    new Enrollment {Student=alexander,Course=microeconomics, Grade = Grade.C },
    new Enrollment {Student=alexander, Course=macroeconomics, Grade = Grade.B },
    new Enrollment { Student = alonso, Course = calculus, Grade = Grade.B },
    new Enrollment {Student = alonso, Course = trigonometry, Grade = Grade.B },
    new Enrollment {Student = alonso, Course = composition, Grade = Grade.B },
    new Enrollment { Student = anand, Course = chemistry, },
    new Enrollment {Student = anand, Course =microeconomics, Grade = Grade.B },
    new Enrollment {Student = barzdukas, Course = chemistry, Grade = Grade.B },
    new Enrollment { Student = li, Course = composition, Grade = Grade.B },
    new Enrollment {Student = justice, Course =literature, Grade = Grade.B } };
context.AddRange(enrollments);
context.SaveChanges();
}
}
```

1. Khởi tạo project
2. Thêm, Xem, Sửa, Xóa
3. Sắp xếp, lọc, phân trang, gom nhóm
4. Migrations
5. Tạo một model dữ liệu phức tạp

- **Xóa và tạo lại database**

- Xóa thư mục Migrations
- Vào PMC, gõ lệnh  
[Drop-Database](#)  
[Add-Migration](#) InitialCreate  
[Update-Database](#)

# Chương 3. Razor Page với Entity Framework Core

- 1. Khởi tạo project
- 2. Thêm, Xem, Sửa, Xóa
- 3. Sắp xếp, lọc, phân trang, gom nhóm
- 4. Migrations
- 5. Tạo một model dữ liệu phức tạp
- 6. Đọc dữ liệu ở các bảng liên kết với nhau

Course
Properties
CourseID
Title
Credits
DepartmentID
Navigation Properties
Department
Enrollments
Course

Instructor
Properties
ID
LastName
FirstMidName
HireDate
Navigation Properties
Course
OfficeAssignment

Courses				
<a href="#">Create New</a>				
Number	Title	Credits	Department	
1045	Calculus	4	Mathematics	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
1050	Chemistry	3	Engineering	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
2021	Composition	3	English	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>

Courses Taught by Selected Instructor			
	Number	Title	Department
<a href="#">Select</a>	2021	Composition	English
<a href="#">Select</a>	2042	Literature	English

Students Enrolled in Selected Course	
Name	Grade
Alonso, Meredith	B
Li, Yan	B

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

### 6. Đọc dữ liệu ở các bảng liên kết với nhau

## • Truy vấn Eager, explicit, và lazy loading

- Có một vài cách load dữ liệu liên quan vào các thuộc tính điều hướng của một thực thể:

- **Eager loading**: là khi một truy vấn cho một loại thực thể cũng tải các thực thể có liên quan. Khi một thực thể được đọc, dữ liệu liên quan của nó sẽ được truy xuất. Điều này thường dẫn đến một truy vấn kết hợp duy nhất lấy tất cả dữ liệu cần thiết. EF Core sẽ đưa ra nhiều truy vấn cho một số kiểu **eager loading**. **Eager loading** được chỉ định bằng các phương pháp *Include* và *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

```
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

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

### 6. Đọc dữ liệu ở các bảng liên kết với nhau

## • Truy vấn Eager, explicit, và lazy loading

- **Explicit loading:** Khi thực thể được đọc lần đầu tiên, dữ liệu liên quan sẽ không được truy xuất. Code phải được viết để truy xuất dữ liệu liên quan khi cần. **Explicit loading** với các truy vấn riêng biệt dẫn đến nhiều truy vấn được gửi đến cơ sở dữ liệu. Với **Explicit loading**, code chỉ định các thuộc tính điều hướng sẽ được tải. Sử dụng phương thức **Load** để thực hiện **Explicit loading**. Ví dụ:

```
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

- **Lazy loading:** Khi thực thể được đọc lần đầu tiên, dữ liệu liên quan sẽ không được truy xuất. Lần đầu tiên một thuộc tính điều hướng được xử lý, dữ liệu cần thiết cho thuộc tính điều hướng đó sẽ tự động được truy xuất. Một truy vấn được gửi đến cơ sở dữ liệu mỗi khi một thuộc tính điều hướng được truy cập lần đầu tiên. **Lazy loading** có thể ảnh hưởng đến hiệu suất, chẳng hạn như khi các nhà phát triển sử dụng  $N + 1$  truy vấn.  $N + 1$  truy vấn tải một truy vấn cha và liệt kê thông qua truy vấn con.

## Chương 3. Razor Page với Entity Framework Core

1. Khởi tạo project

2. Thêm, Xem, Sửa, Xóa

3. Sắp xếp, lọc, phân trang, gom nhóm

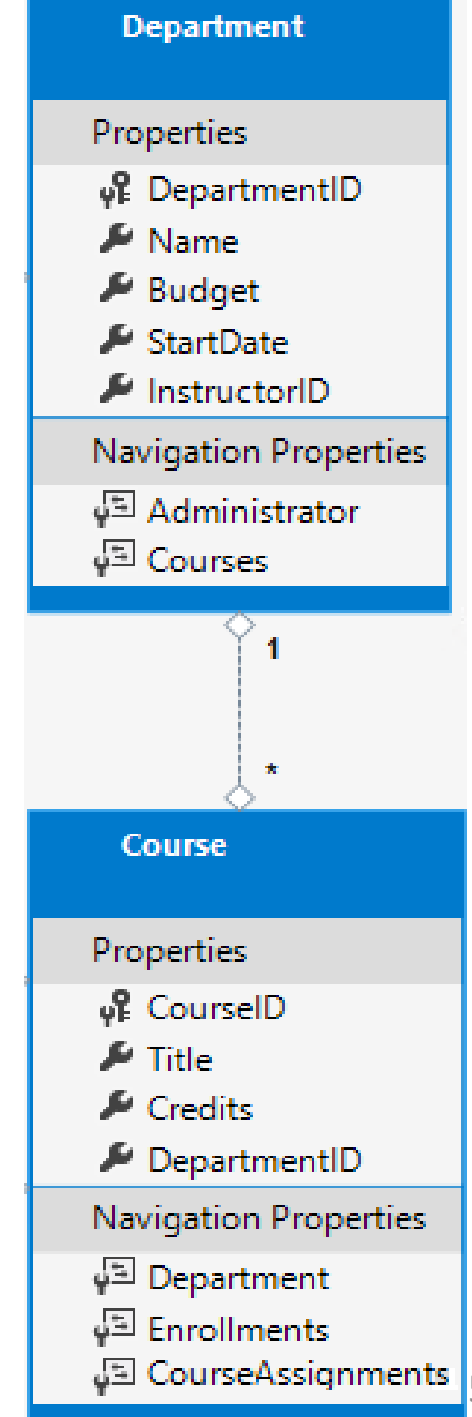
4. Migrations

5. Tạo một model dữ liệu phức tạp

6. Đọc dữ liệu ở các bảng liên kết với nhau

### • Tạo trang Course

- Thực thể **Course** gồm các thuộc tính điều hướng chứa thực thể có liên quan **Department**.
- Để hiển thị tên **department** được gán cho **course**:
  - Load thực thể có liên quan department thông qua thuộc tính điều hướng Course.Department
  - Nhận tên từ thuộc tính Name của thực thể department.





1. Khởi tạo project

2. Thêm, Xem, Sửa, Xóa

3. Sắp xếp, lọc, phân trang, gom nhóm

4. Migrations

5. Tạo một model dữ liệu phức tạp

6. Đọc dữ liệu ở các bảng liên kết với nhau

## • Các trang Scaffold Course

- Tương tự như cách tạo các trang Student bằng Scaffold, ta tiến hành:
  - Tạo thư mục *Pages/Courses*
  - Sử dụng Course để làm model class.
  - Sử dụng class context có sẵn thay vì tạo mới một context.
- Mở *Pages/Courses/Index.cshtml.cs* và kiểm tra phương thức *OnGetAsync*. Cách scaffold chỉ định *eager loading* cho thuộc tính điều hướng *Department*. Phương thức *Include* chỉ định *eager loading*.
- Chạy chương trình và chọn menu **Course**. Cột department chỉ hiển thị *DepartmentID*.

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

### 6. Đọc dữ liệu ở các bảng liên kết với nhau

## • **Hiện thị tên department**

- Cập nhật lại class *Pages/Courses/Index.cshtml.cs*

```
public class IndexModel : PageModel
{
    private readonly SchoolContext _context;
    public IndexModel(SchoolContext context)
    {
        _context = context;
    }
    public IList<Course> Courses { get; set; }

    public async Task OnGetAsync()
    {
        Courses = await _context.Courses
            .Include(c => c.Department)
            .AsNoTracking()
            .ToListAsync();
    }
}
```

- Thay đổi **Course** thành **Courses** và thêm **AsNoTracking**. **AsNoTracking** cải thiện hiệu suất vì các thực thể được trả về không được theo dõi. Các thực thể không cần được theo dõi vì chúng không được cập nhật trong context hiện tại.



### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

### 6. Đọc dữ liệu ở các bảng liên kết với nhau

## • Hiển thị tên department

- Cập nhật lại *Pages/Courses/Index.cshtml*

```
@page
@model PXU_RazorPageEFCore.Pages.Courses.IndexModel
@{
    ViewData["Title"] = "Courses";
}
<h1>Courses</h1>
<p>
    <a asp-page="Create">Create New</a>
</p>
<table class="table">
    <thead>
        <tr>
            <th>@Html.DisplayNameFor(model => model.Courses[0].CourseID) </th>
            <th> @Html.DisplayNameFor(model => model.Courses[0].Title) </th>
            <th> @Html.DisplayNameFor(model => model.Courses[0].Credits) </th>
            <th>@Html.DisplayNameFor(model=>model.Courses[0].Department) </th>
            <th></th>
        </tr>
    </thead>
    <tbody>
```

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

### 6. Đọc dữ liệu ở các bảng liên kết với nhau

## • Hiện thị tên department

- Cập nhật lại *Pages/Courses/Index.cshtml*

```
@foreach (var item in Model.Courses) {  
    <tr>  
        <td> @Html.DisplayFor(modelItem => item.CourseID) </td>  
        <td> @Html.DisplayFor(modelItem => item.Title) </td>  
        <td> @Html.DisplayFor(modelItem => item.Credits) </td>  
        <td> @Html.DisplayFor(modelItem => item.Department.Name) </td>  
        <td>  
            <a asp-page="./Edit" asp-route-id="@item.CourseID">Edit</a> |  
            <a asp-page="./Details" asp-route-id="@item.CourseID">Details</a> |  
            <a asp-page="./Delete" asp-route-id="@item.CourseID">Delete</a>  
        </td>  
    </tr>  
}  
  
</tbody>  
</table>
```

- Các thay đổi:

- Thay **Course** thành **Courses**

- Thêm 1 cột để hiển thị giá trị thuộc tính **CourseID**. Mặc định, khóa chính sẽ không được scaffold bởi vì nó ít khi dùng. Tuy nhiên, trong trường hợp này sử dụng khóa chính rất hữu ích.

- Thay cột **Department** thành hiển thị tên department thông qua thuộc tính điều hướng **department**: *@Html.DisplayFor(modelItem => item.Department.Name)*

- Chạy thử

1. Khởi tạo project
2. Thêm, Xem, Sửa, Xóa
3. Sắp xếp, lọc, phân trang, gom nhóm
4. Migrations
5. Tạo một model dữ liệu phức tạp
6. Đọc dữ liệu ở các bảng liên kết với nhau

### • Load dữ liệu liên quan với Select

- Phương thức *OnGetAsync* tải dữ liệu liên quan với phương thức *Include*. Phương pháp *Select* là một phương pháp thay thế cho việc load dữ liệu cần quan tâm. Đối với các mục đơn lẻ, chẳng hạn như *Department.Name*, chỉ cần sử dụng SQL INNER JOIN. Đối với bộ sưu tập, sử dụng một query truy cập cơ sở dữ liệu khác, nhưng cũng sử dụng toán tử Include trên bộ sưu tập.

```
public IList<CourseViewModel> CourseVM { get; set; }

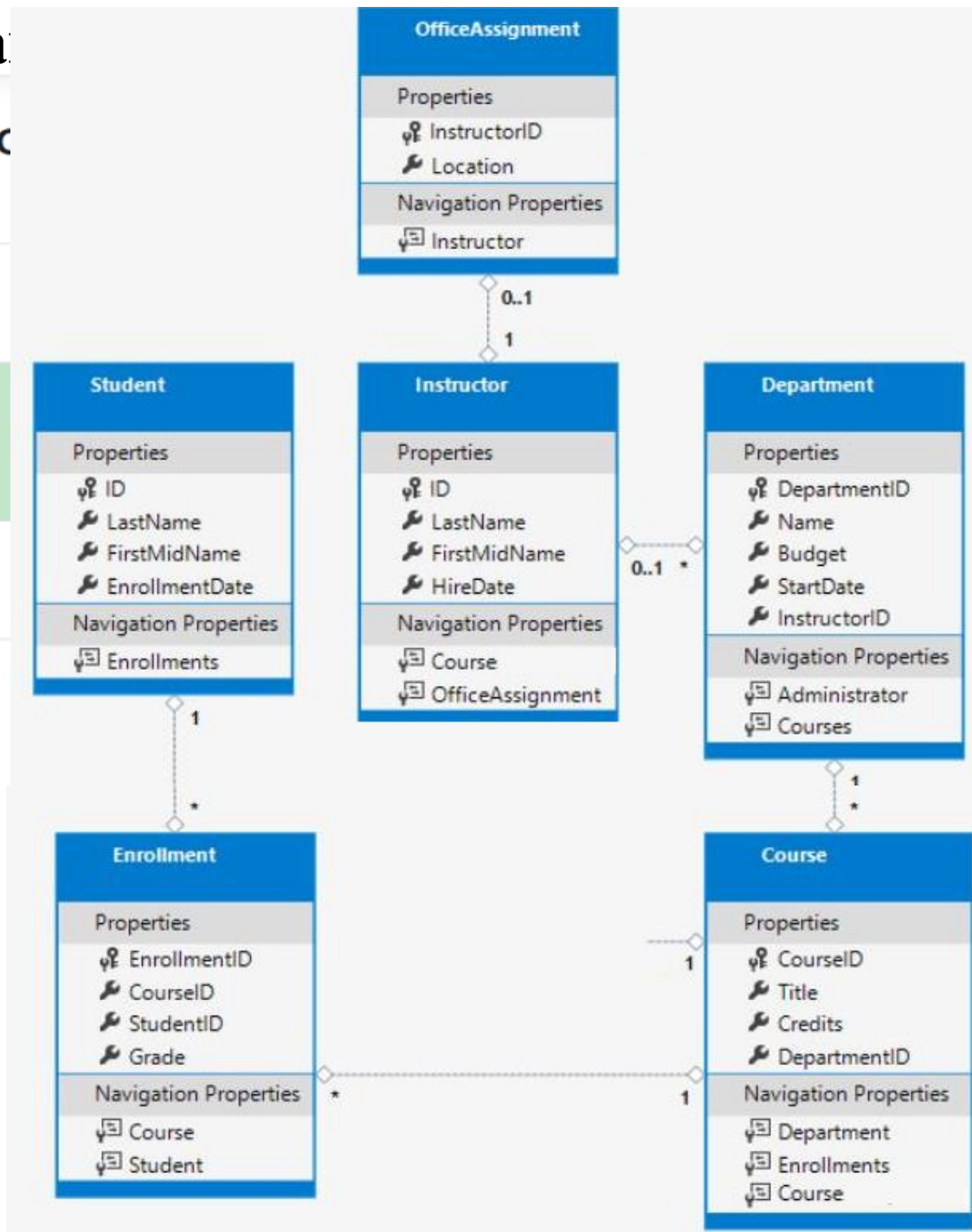
public async Task OnGetAsync()
{
    CourseVM = await _context.Courses
        .Select(p => new CourseViewModel
        {
            CourseID = p.CourseID,
            Title = p.Title,
            Credits = p.Credits,
            DepartmentName = p.Department.Name })
        .ToListAsync();
}
```

## Chương 3. Razor Page với Entity Framework Core

1. Khởi tạo project
2. Thêm, Xem, Sửa, Xóa
3. Sắp xếp, lọc, phân trang, gom nhóm
4. Migrations
5. Tạo một model dữ liệu phức tạp
6. Đọc dữ liệu ở các bảng liên kết với nhau

### • Tạo tra

Instructor
Create New
Last Name
Abercrombie
Fakhouri



Instructor
Department
English
English
Grade
B
B

1. Khởi tạo project
2. Thêm, Xem, Sửa, Xóa
3. Sắp xếp, lọc, phân trang, gom nhóm
4. Migrations
5. Tạo một model dữ liệu phức tạp
6. Đọc dữ liệu ở các bảng liên kết với nhau

### • Tạo trang Instructor

- Tạo view model: Tạo class

*Models/SchoolViewModels/InstructorIndexData.cs*

- Scaffold trang Instructor

- Tạo thư mục Instructors
- Sử dụng class Instructor làm class model
- Sử dụng class context hiện có

- Cập nhật lại class *Pages/Instructors/Index.cshtml.cs*

```
public class IndexModel : PageModel
{
    private readonly SchoolContext _context;
    public IndexModel(SchoolContext context)
    {
        _context = context;
    }
    public InstructorIndexData InstructorData { get; set; }
    public int InstructorID { get; set; }
    public int CourseID { get; set; }
```

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

### 6. Đọc dữ liệu ở các bảng liên kết với nhau

## • Tạo trang Instructor

```
public async Task OnGetAsync(int? id, int? courseID) {
    InstructorData = new InstructorIndexData();
    InstructorData.Instructors = await _context.Instructors
        .Include(i => i.OfficeAssignment)
        .Include(i => i.Courses)
        .ThenInclude(c => c.Department)
        .OrderBy(i => i.LastName)
        .ToListAsync();
    if (id != null) {
        InstructorID = id.Value;
        Instructor instructor = InstructorData.Instructors
            .Where(i => i.ID == id.Value).Single();
        InstructorData.Courses = instructor.Courses;
    }
    if (courseID != null) {
        CourseID = courseID.Value;
        var selectedCourse = InstructorData.Courses
            .Where(x => x.CourseID == courseID).Single();
        await _context.Entry(selectedCourse)
            .Collection(x => x.Enrollments).LoadAsync();
        foreach (Enrollment enrollment in selectedCourse.Enrollments) {
            await _context.Entry(enrollment)
                .Reference(x => x.Student).LoadAsync();
        }
        InstructorData.Enrollments = selectedCourse.Enrollments;
    }
}
```



### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

### 6. Đọc dữ liệu ở các bảng liên kết với nhau

## • Tạo trang Instructor

- Cập nhật trang *Pages/Instructors/Index.cshtml*

```
@page "{id:int?}"
@model PXU_RazorPageEFCore.Pages.Instructors.IndexModel
@{
    ViewData["Title"] = "Instructors";
}
<h2>Instructors</h2>
<p>
    <a asp-page="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>
```

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

### 6. Đọc dữ liệu ở các bảng liên kết với nhau

## • Tạo trang Instructor

### • Cập nhật trang *Pages/Instructors/Index.cshtml*

```
@if (Model.InstructorData.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.InstructorData.Courses) {  
            string selectedRow = "";  
            if (item.CourseID == Model.CourseID) {  
                selectedRow = "table-success"; }  
            <tr class="@selectedRow">  
                <td> <a asp-page="./Index" asp-route-courseID="@item.CourseID">Select</a>  
                </td>  
                <td> @item.CourseID </td>  
                <td> @item.Title </td>  
                <td> @item.Department.Name </td>  
            </tr>  
        }  
    </table>  
}
```



### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

### 6. Đọc dữ liệu ở các bảng liên kết với nhau

## • Tạo trang Instructor

- Cập nhật trang *Pages/Instructors/Index.cshtml*

```
@if (Model.InstructorData.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.InstructorData.Enrollments)
        {
            <tr>
                <td> @item.Student.FullName </td>
                <td> @Html.DisplayFor(modelItem => item.Grade) </td>
            </tr>
        }
    </table>
}
```

- *Chạy thử*

## Chương 3. Razor Page với Entity Framework Core

1. Khởi tạo project
2. Thêm, Xem, Sửa, Xóa
3. Sắp xếp, lọc, phân trang, gom nhóm
4. Migrations
5. Tạo một model dữ liệu phức tạp
6. Đọc dữ liệu ở các bảng liên kết với nhau
7. Cập nhật dữ liệu có liên quan

### Edit Course

---

Number  
1045

Title

Credits

Department

### Edit Instructor

---

Last Name

First Name

Hire Date

Office Location

---

<input checked="" type="checkbox"/> 1045 Calculus	<input type="checkbox"/> 1050 Chemistry	<input type="checkbox"/> 2021 Composition
<input type="checkbox"/> 2042 Literature	<input type="checkbox"/> 3141 Trigonometry	<input type="checkbox"/> 4022 Microeconomics
<input type="checkbox"/> 4041 Macroeconomics		

1. Khởi tạo project
2. Thêm, Xem, Sửa, Xóa
3. Sắp xếp, lọc, phân trang, gom nhóm
4. Migrations
5. Tạo một model dữ liệu phức tạp
6. Đọc dữ liệu ở các bảng liên kết với nhau
7. Cập nhật dữ liệu có liên quan

- **Cập nhật các trang Create và Edit của Course**

- **Tạo lớp cơ sở cho Create và Edit của Course**

- Tạo class *Pages/Courses/DepartmentNamePageModel.cs*

```
public class DepartmentNamePageModel : PageModel
{
    public SelectList DepartmentNameSL { get; set; }
    public void PopulateDepartmentsDropDownList(SchoolContext _context, object
selectedDepartment = null)
    {
        var departmentsQuery = from d in _context.Departments
                                orderby d.Name // Sort by name.
                                select d;
        DepartmentNameSL = new SelectList(departmentsQuery.AsNoTracking(),
            "DepartmentID", "Name", selectedDepartment);
    }
}
```

- Đoạn trên tạo một *SelectList* để chứa danh sách các tên department. Nếu *selectDepartment* được chỉ định, thì department được chọn trong *SelectList*.
    - Các class model của các trang Create và Edit sẽ lấy từ *DepartmentNamePageModel*.

1. Khởi tạo project
2. Thêm, Xem, Sửa, Xóa
3. Sắp xếp, lọc, phân trang, gom nhóm
4. Migrations
5. Tạo một model dữ liệu phức tạp
6. Đọc dữ liệu ở các bảng liên kết với nhau
7. Cập nhật dữ liệu có liên quan

## • Cập nhật các trang Create và Edit của Course

### • Cập nhật model trang Create của Course

- Một **Course** được chỉ định cho một **Department**. Lớp cơ sở cho các trang **Create** và **Edit** của **Course** cung cấp một **SelectList** để chọn một **Department**. Một danh sách xổ xuống (*drop-down list*) mà sử dụng **SelectList** để thiết lập thuộc tính khóa ngoại (FK) *Course.DepartmentID*. **EF Core** sử dụng *Course.DepartmentID* để load thuộc tính điều hướng khóa ngoại **Department**

### Create

#### Course

Number

Title

Credits

Department

-- Select Department --

-- Select Department --

Economics

Engineering

English

Mathematics

1. Khởi tạo project

2. Thêm, Xem, Sửa, Xóa

3. Sắp xếp, lọc, phân trang, gom nhóm

4. Migrations

5. Tạo một model dữ liệu phức tạp

6. Đọc dữ liệu ở các bảng liên kết với nhau

7. Cập nhật dữ liệu có liên quan

# • Cập nhật các trang Create và Edit của Course

## • Cập nhật model trang Create của Course

### • Cập nhật *Pages/Courses/Create.cshtml.cs*

```
public class CreateModel : DepartmentNamePageModel {
    private readonly PXU_RazorPageEFCore.Data.SchoolContext _context;
    public CreateModel(PXU_RazorPageEFCore.Data.SchoolContext context) {
        _context = context;
    }
    public IActionResult OnGet() {
        PopulateDepartmentsDropDownList(_context);
        return Page();
    }
    [BindProperty]
    public Course Course { get; set; }
    public async Task<IActionResult> OnPostAsync() {
        var emptyCourse = new Course();
        if (await TryUpdateModelAsync<Course>(
            emptyCourse,
            "course", // Prefix for form value.
            s => s.CourseID, s => s.DepartmentID, s => s.Title, s => s.Credits))
        {
            _context.Courses.Add(emptyCourse);
            await _context.SaveChangesAsync();
            return RedirectToPage("./Index");
        }
        // Select DepartmentID if TryUpdateModelAsync fails.
        PopulateDepartmentsDropDownList(_context, emptyCourse.DepartmentID);
        return Page();
    }
}
```

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

### 6. Đọc dữ liệu ở các bảng liên kết với nhau

### 7. Cập nhật dữ liệu có liên quan

## • Cập nhật các trang Create và Edit của Course

### • Cập nhật model trang Create của Course

#### • Cập nhật *Pages/Courses/Create.cshtml*

```
@page @model PXU_RazorPageEFCore.Pages.Courses.CreateModel
@{
    ViewData["Title"] = "Create Course";
}
<h2>Create</h2>
<h4>Course</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form method="post">
            .....
            <div class="form-group">
                <label asp-for="Course.Credits" class="control-label"></label>
                <input asp-for="Course.Credits" class="form-control" />
                <span asp-validation-for="Course.Credits" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Course.Department" class="control-label"></label>
                <select asp-for="Course.DepartmentID" class="form-control"
                    asp-items="@Model.DepartmentNameSL">
                    <option value="">-- Select Department --</option>
                </select>
                <span asp-validation-for="Course.DepartmentID" class="text-danger" />
            </div>
            .....
        </form>
    </div>
</div>
```

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

### 6. Đọc dữ liệu ở các bảng liên kết với nhau

### 7. Cập nhật dữ liệu có liên quan

## • Cập nhật các trang Create và Edit của Course

### • Cập nhật model trang Edit của Course

#### • Cập nhật *Pages/Courses/Edit.cshtml.cs*

```
public class EditModel : DepartmentNamePageModel {
    ....
    public async Task<IActionResult> OnGetAsync(int? id) {
        if (id == null) { return NotFound(); }
        Course = await _context.Courses
            .Include(c => c.Department).FirstOrDefaultAsync(m => m.CourseID == id);
        if (Course == null) { return NotFound(); }
        // Select current DepartmentID.
        PopulateDepartmentsDropDownList(_context, Course.DepartmentID);
        return Page();
    }
    public async Task<IActionResult> OnPostAsync(int? id) {
        if (id == null) { return NotFound(); }
        var courseToUpdate = await _context.Courses.FindAsync(id);
        if (courseToUpdate == null) { return NotFound(); }
        if (await TryUpdateModelAsync<Course>(
            courseToUpdate, "course", // Prefix for form value.
            c => c.Credits, c => c.DepartmentID, c => c.Title)) {
            await _context.SaveChangesAsync();
            return RedirectToPage("./Index");
        }
        // Select DepartmentID if TryUpdateModelAsync fails.
        PopulateDepartmentsDropDownList(_context, courseToUpdate.DepartmentID);
        return Page();
    }
}
```



### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

### 6. Đọc dữ liệu ở các bảng liên kết với nhau

### 7. Cập nhật dữ liệu có liên quan

## • Cập nhật các trang Create và Edit của Course

### • Cập nhật model trang Edit của Course

#### • Cập nhật *Pages/Courses/Edit.cshtml*

```
<form method="post">
  <div asp-validation-summary="ModelOnly" class="text-danger"></div>
  <input type="hidden" asp-for="Course.CourseID" />
  <div class="form-group">
    <label asp-for="Course.CourseID" class="control-label"></label>
    <div>@Html.DisplayFor(model => model.Course.CourseID)</div>
  </div>
  <div class="form-group">
    <label asp-for="Course.Title" class="control-label"></label>
    <input asp-for="Course.Title" class="form-control" />
    <span asp-validation-for="Course.Title" class="text-danger"></span>
  </div>
  <div class="form-group">
    <label asp-for="Course.Credits" class="control-label"></label>
    <input asp-for="Course.Credits" class="form-control" />
    <span asp-validation-for="Course.Credits" class="text-danger"></span>
  </div>
  <div class="form-group">
    <label asp-for="Course.Department" class="control-label"></label>
    <select asp-for="Course.DepartmentID" class="form-control"
      asp-items="@Model.DepartmentNameSL"></select>
    <span asp-validation-for="Course.DepartmentID" class="text-danger"></span>
  </div>
  <div class="form-group">
    <input type="submit" value="Save" class="btn btn-primary" /> </div>
</form>
```



### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

### 6. Đọc dữ liệu ở các bảng liên kết với nhau

### 7. Cập nhật dữ liệu có liên quan

## • Cập nhật các model trang Course

- Cập nhật *Pages/Courses/Delete.cshtml.cs* và *Pages/Courses/Details.cshtml.cs* bằng cách thêm ***AsNoTracking*** vào phương thức ***OnGetAsync***

```
public async Task<IActionResult> OnGetAsync(int? id)
{
    if (id == null) {
        return NotFound();
    }
    Course = await _context.Courses
        .AsNoTracking()
        .Include(c => c.Department)
        .FirstOrDefaultAsync(m => m.CourseID == id);
    if (Course == null) {
        return NotFound();
    }
    return Page();
}
```

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

### 6. Đọc dữ liệu ở các bảng liên kết với nhau

### 7. Cập nhật dữ liệu có liên quan

## • Cập nhật các trang Razor của Course

- Cập nhật *Pages/Courses/Delete.cshtml*

```
<div>
    <h4>Course</h4>
    <hr />
    <dl class="row">
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Course.CourseID)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Course.CourseID)
        </dd>
        .....title.....
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Course.Department)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Course.Department.Name)
        </dd>
    </dl>
    <form method="post">
        <input type="hidden" asp-for="Course.CourseID" />
        <input type="submit" value="Delete" class="btn btn-danger" /> |
        <a asp-page="./Index">Back to List</a>
    </form>
</div>
```

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

### 6. Đọc dữ liệu ở các bảng liên kết với nhau

### 7. Cập nhật dữ liệu có liên quan

## • Cập nhật các trang Razor của Course

- Cập nhật *Pages/Courses/Detail.cshtml*

```
<div>
    <h4>Course</h4>
    <hr />
    <dl class="row">
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Course.CourseID)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Course.CourseID)
        </dd>
        .....
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Course.Department)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Course.Department.Name)
        </dd>
    </dl>
</div>
```

- Chạy thử các trang create, edit, delete, detail của Course

1. Khởi tạo project
2. Thêm, Xem, Sửa, Xóa
3. Sắp xếp, lọc, phân trang, gom nhóm
4. Migrations
5. Tạo một model dữ liệu phức tạp
6. Đọc dữ liệu ở các bảng liên kết với nhau
7. Cập nhật dữ liệu có liên quan

### • Cập nhật các trang Create và Edit của Instructor

## Edit

### Instructor

---

Last Name

First Name

Hire Date

Office Location

---

<input checked="" type="checkbox"/> 1045 Calculus	<input type="checkbox"/> 1050 Chemistry	<input type="checkbox"/> 2021 Composition
<input type="checkbox"/> 2042 Literature	<input type="checkbox"/> 3141 Trigonometry	<input type="checkbox"/> 4022 Microeconomics
<input type="checkbox"/> 4041 Macroeconomics		

Save

1. Khởi tạo project
2. Thêm, Xem, Sửa, Xóa
3. Sắp xếp, lọc, phân trang, gom nhóm
4. Migrations
5. Tạo một model dữ liệu phức tạp
6. Đọc dữ liệu ở các bảng liên kết với nhau
7. Cập nhật dữ liệu có liên quan

## • Cập nhật các trang Create và Edit của Instructor

- Tạo một class để dữ liệu Courses được chỉ định

- Tạo class *Models/SchoolViewModels/AssignedCourseData.cs*

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

1. Khởi tạo project

2. Thêm, Xem, Sửa, Xóa

3. Sắp xếp, lọc, phân trang, gom nhóm

4. Migrations

5. Tạo một model dữ liệu phức tạp

6. Đọc dữ liệu ở các bảng liên kết với nhau

7. Cập nhật dữ liệu có liên quan

## • Cập nhật các trang Create và Edit của Instructor

### • Tạo một class cơ sở cho model trang Instructor

- Tạo class cơ sở *Pages/Instructors/InstructorCoursesPageModel.cs*

```
public class InstructorCoursesPageModel : PageModel
{
    public List<AssignedCourseData> AssignedCourseDataList;
    public void PopulateAssignedCourseData(SchoolContext
        context, Instructor instructor)
    {
        var allCourses = context.Courses;
        var instructorCourses = new HashSet<int>(
            instructor.Courses.Select(c => c.CourseID));
        AssignedCourseDataList = new List<AssignedCourseData>();
        foreach (var course in allCourses)
        {
            AssignedCourseDataList.Add(new AssignedCourseData
            {
                CourseID = course.CourseID,
                Title = course.Title,
                Assigned = instructorCourses.Contains(
                    course.CourseID)
            });
        }
    }
}
```

1. Khởi tạo project

2. Thêm, Xem, Sửa, Xóa

3. Sắp xếp, lọc, phân trang, gom nhóm

4. Migrations

5. Tạo một model dữ liệu phức tạp

6. Đọc dữ liệu ở các bảng liên kết với nhau

7. Cập nhật dữ liệu có liên quan

## • Cập nhật các trang Create và Edit của Instructor

### • Tạo một class cơ sở cho model trang Instructor

- Tạo class cơ sở *Pages/Instructors/InstructorCoursesPageModel.cs*

```
public class InstructorCoursesPageModel : PageModel
{
    public List<AssignedCourseData> AssignedCourseDataList;
    public void PopulateAssignedCourseData(SchoolContext
        context, Instructor instructor)
    {
        var allCourses = context.Courses;
        var instructorCourses = new HashSet<int>(
            instructor.Courses.Select(c => c.CourseID));
        AssignedCourseDataList = new List<AssignedCourseData>();
        foreach (var course in allCourses)
        {
            AssignedCourseDataList.Add(new AssignedCourseData
            {
                CourseID = course.CourseID,
                Title = course.Title,
                Assigned = instructorCourses.Contains(
                    course.CourseID)
            });
        }
    }
}
```



**1. Khởi tạo project**

**2. Thêm, Xem, Sửa,  
Xóa**

**3. Sắp xếp, lọc, phân  
trang, gom nhóm**

**4. Migrations**

**5. Tạo một model dữ  
liệu phức tạp**

**6. Đọc dữ liệu ở các  
bảng liên kết với nhau**

**7. Cập nhật dữ liệu có  
liên quan**

## • Cập nhật các trang Create và Edit của Instructor

### • Xử lý vị trí của office (office location)

- Một mối quan hệ khác của trang edit là mối quan hệ 1-1 giữa thực thể **Instructor** và **OfficeAssignment**. Phần code edit của **Instructor** phải được xử lý theo các kịch bản sau:
  - Nếu người dùng xóa office assignment, hãy xóa thực thể **OfficeAssignment**
  - Nếu người dùng để con trỏ vào office assignment và để trống, hãy tạo mới một thực thể **OfficeAssignment**.
  - Nếu người dùng thay đổi office assignment, hãy cập nhật thực thể **OfficeAssignment**

1. Khởi tạo project

2. Thêm, Xem, Sửa, Xóa

3. Sắp xếp, lọc, phân trang, gom nhóm

4. Migrations

5. Tạo một model dữ liệu phức tạp

6. Đọc dữ liệu ở các bảng liên kết với nhau

7. Cập nhật dữ liệu có liên quan

# • Cập nhật các trang Create và Edit của Instructor

## • Cập nhật model trang Edit của Instructor

- Cập nhật *Pages/Instructors/Edit.cshtml.cs*

```
public class EditModel : InstructorCoursesPageModel {
    private readonly PXU_RazorPageEFCore.Data.SchoolContext _context;
    public EditModel(P XU_RazorPageEFCore.Data.SchoolContext context) {
        _context = context;
    }
    [BindProperty]
    public Instructor Instructor { get; set; }
    public async Task<IActionResult> OnGetAsync(int? id) {
        if (id == null) {
            return NotFound(); }
        Instructor = await _context.Instructors
            .Include(i => i.OfficeAssignment)
            .Include(i => i.Courses)
            .AsNoTracking()
            .FirstOrDefaultAsync(m => m.ID == id);
        if (Instructor == null) {
            return NotFound(); }
        PopulateAssignedCourseData(_context, Instructor);
        return Page();
    }
}
```

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

### 6. Đọc dữ liệu ở các bảng liên kết với nhau

### 7. Cập nhật dữ liệu có liên quan

## • Cập nhật các trang Create và Edit của Instructor

### • Cập nhật model trang Edit của Instructor

#### • Cập nhật *Pages/Instructors/Edit.cshtml.cs*

```
public async Task<IActionResult> OnPostAsync(int? id, string[] selectedCourses)
{
    if (id == null) { return NotFound(); }
    var instructorToUpdate = await _context.Instructors
        .Include(i => i.OfficeAssignment)
        .Include(i => i.Courses)
        .FirstOrDefaultAsync(s => s.ID == id);
    if (instructorToUpdate == null) { return NotFound(); }
    if (await TryUpdateModelAsync<Instructor>(
        instructorToUpdate, "Instructor",
        i => i.FirstMidName, i => i.LastName,
        i => i.HireDate, i => i.OfficeAssignment))
    {
        if (String.IsNullOrEmpty(
            instructorToUpdate.OfficeAssignment?.Location)) {
            instructorToUpdate.OfficeAssignment = null;
        }
        UpdateInstructorCourses(selectedCourses, instructorToUpdate);
        await _context.SaveChangesAsync();
        return RedirectToPage("../Index");
    }
    UpdateInstructorCourses(selectedCourses, instructorToUpdate);
    PopulateAssignedCourseData(_context, instructorToUpdate);
    return Page();
}
```

1. Khởi tạo project

2. Thêm, Xem, Sửa, Xóa

3. Sắp xếp, lọc, phân trang, gom nhóm

4. Migrations

5. Tạo một model dữ liệu phức tạp

6. Đọc dữ liệu ở các bảng liên kết với nhau

7. Cập nhật dữ liệu có liên quan

# • Cập nhật các trang Create và Edit của Instructor

## • Cập nhật model trang Edit của Instructor

### • Cập nhật *Pages/Instructors/Edit.cshtml.cs*

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

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

### 6. Đọc dữ liệu ở các bảng liên kết với nhau

### 7. Cập nhật dữ liệu có liên quan

## • Cập nhật trang Razor Edit của Instructor

### • Cập nhật *Pages/Instructors/Edit.cshtml*

```
..... Instructor.HireDate
<div class="form-group">
    <label asp-for="Instructor.OfficeAssignment.Location" class="control-label"></label>
    <input asp-for="Instructor.OfficeAssignment.Location" class="form-control" />
    <span asp-validation-for="Instructor.OfficeAssignment.Location" class="text-danger" />
</div>
<div class="form-group">
    <div class="table">
        <table>
            <tr>
                @{ int cnt = 0;
                  foreach (var course in Model.AssignedCourseDataList) {
                    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>
.....btn Save
```

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

### 6. Đọc dữ liệu ở các bảng liên kết với nhau

### 7. Cập nhật dữ liệu có liên quan

## • Cập nhật trang Create của Instructor

- Cập nhật *Pages/Instructors/Create.cshtml.cs*

```
public class CreateModel : InstructorCoursesPageModel {
    private readonly PXU_RazorPageEFCore.Data.SchoolContext _context;
    private readonly ILogger<InstructorCoursesPageModel> _logger;
    public CreateModel(SchoolContext context,
        ILogger<InstructorCoursesPageModel> logger) {
        _context = context;
        _logger = logger;
    }
    public IActionResult OnGet() {
        var instructor = new Instructor();
        instructor.Courses = new List<Course>();
        // Provides an empty collection for the foreach loop
        // foreach (var course in Model.AssignedCourseDataList)
        // in the Create Razor page.
        PopulateAssignedCourseData(_context, instructor);
        return Page();
    }
    [BindProperty] public Instructor Instructor { get; set; }
```

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

### 6. Đọc dữ liệu ở các bảng liên kết với nhau

### 7. Cập nhật dữ liệu có liên quan

## • Cập nhật trang Create của Instructor

- Cập nhật *Pages/Instructors/Create.cshtml.cs*

```
public async Task<IActionResult> OnPostAsync(string[] selectedCourses)
{
    var newInstructor = new Instructor();
    if (selectedCourses.Length > 0) {
        newInstructor.Courses = new List<Course>();
        // Load collection with one DB call.
        _context.Courses.Load(); }
    // Add selected Courses courses to the new instructor.
    foreach (var course in selectedCourses) {
        var foundCourse = await
        _context.Courses.FindAsync(int.Parse(course));
        if (foundCourse != null) {
            newInstructor.Courses.Add(foundCourse); }
        else {
            _logger.LogWarning("Course {course} not
found", course);
        }
    }
}
```



### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

### 6. Đọc dữ liệu ở các bảng liên kết với nhau

### 7. Cập nhật dữ liệu có liên quan

## • Cập nhật trang Create của Instructor

- Cập nhật *Pages/Instructors/Create.cshtml.cs*

```
try {  
    if (await TryUpdateModelAsync<Instructor>(newInstructor,  
        "Instructor",  
        i => i.FirstMidName, i => i.LastName,  
        i => i.HireDate, i => i.OfficeAssignment))  
    {  
        _context.Instructors.Add(newInstructor);  
        await _context.SaveChangesAsync();  
        return RedirectToPage("./Index");  
    }  
    return RedirectToPage("./Index");  
}  
catch (Exception ex) {  
    _logger.LogError(ex.Message);  
}  
PopulateAssignedCourseData(_context, newInstructor);  
return Page();  
}  
}
```

### 1. Khởi tạo project

### 2. Thêm, Xem, Sửa, Xóa

### 3. Sắp xếp, lọc, phân trang, gom nhóm

### 4. Migrations

### 5. Tạo một model dữ liệu phức tạp

### 6. Đọc dữ liệu ở các bảng liên kết với nhau

### 7. Cập nhật dữ liệu có liên quan

## • Cập nhật trang Create của Instructor

### • Cập nhật *Pages/Instructors/Create.cshtml*

```
....HireDate....
<div class="form-group">
    <label asp-for="Instructor.OfficeAssignment.Location" class="control-label"></label>
    <input asp-for="Instructor.OfficeAssignment.Location" class="form-control" />
    <span asp-validation-for="Instructor.OfficeAssignment.Location" class="text-danger" />
</div>
<div class="form-group">
    <div class="table">
        <table>
            <tr>
                @{
                    int cnt = 0;
                    foreach (var course in Model.AssignedCourseDataList) {
                        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>
    </div>
```

**1. Khởi tạo project**

**2. Thêm, Xem, Sửa,  
Xóa**

**3. Sắp xếp, lọc, phân  
trang, gom nhóm**

**4. Migrations**

**5. Tạo một model dữ  
liệu phức tạp**

**6. Đọc dữ liệu ở các  
bảng liên kết với nhau**

**7. Cập nhật dữ liệu có  
liên quan**

## • Cập nhật trang Delete của Instructor

- Cập nhật *Pages/Instructors/Delete.cshtml.cs*

```
public async Task<IActionResult> OnPostAsync(int? id)
{
    if (id == null) {
        return NotFound();
    }
    Instructor instructor = await _context.Instructors
        .Include(i => i.Courses)
        .SingleOrDefault(i => i.ID == id);
    if (instructor == null) {
        return RedirectToPage("./Index");
    }
    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 RedirectToPage("./Index");
}
```

**Thank you!**