

CO550 – Web Applications

UNIT 9 – Complex Data Model, Data Annotations, Sending Emails, Uploading Files

Tutorial Recap

<https://docs.microsoft.com/en-us/aspnet/core/data/ef-rp/intro?view=aspnetcore-2.1&tabs=visual-studio>

✓ EF Core with Razor Pages

Overview

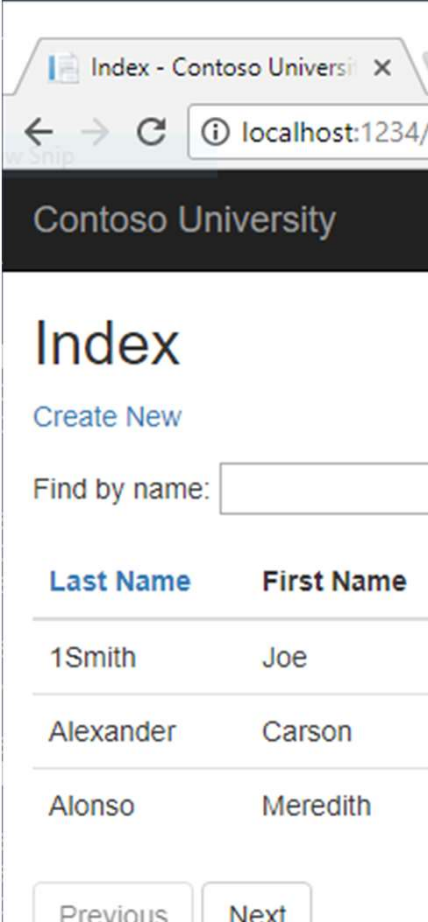
- 1 Get started
- 2 Create, Read, Update, and Delete
- 3 Sort, filter, page, and group
- 4 Migrations

Create a complex data model

Read related data

Update related data

Handle concurrency conflicts



The screenshot shows a web browser window with the title 'Index - Contoso University'. The address bar shows 'localhost:1234/'. The page content includes a dark header with 'Contoso University', a main heading 'Index', a 'Create New' link, a search input field labeled 'Find by name:', and a table of names.

Last Name	First Name
1Smith	Joe
Alexander	Carson
Alonso	Meredith

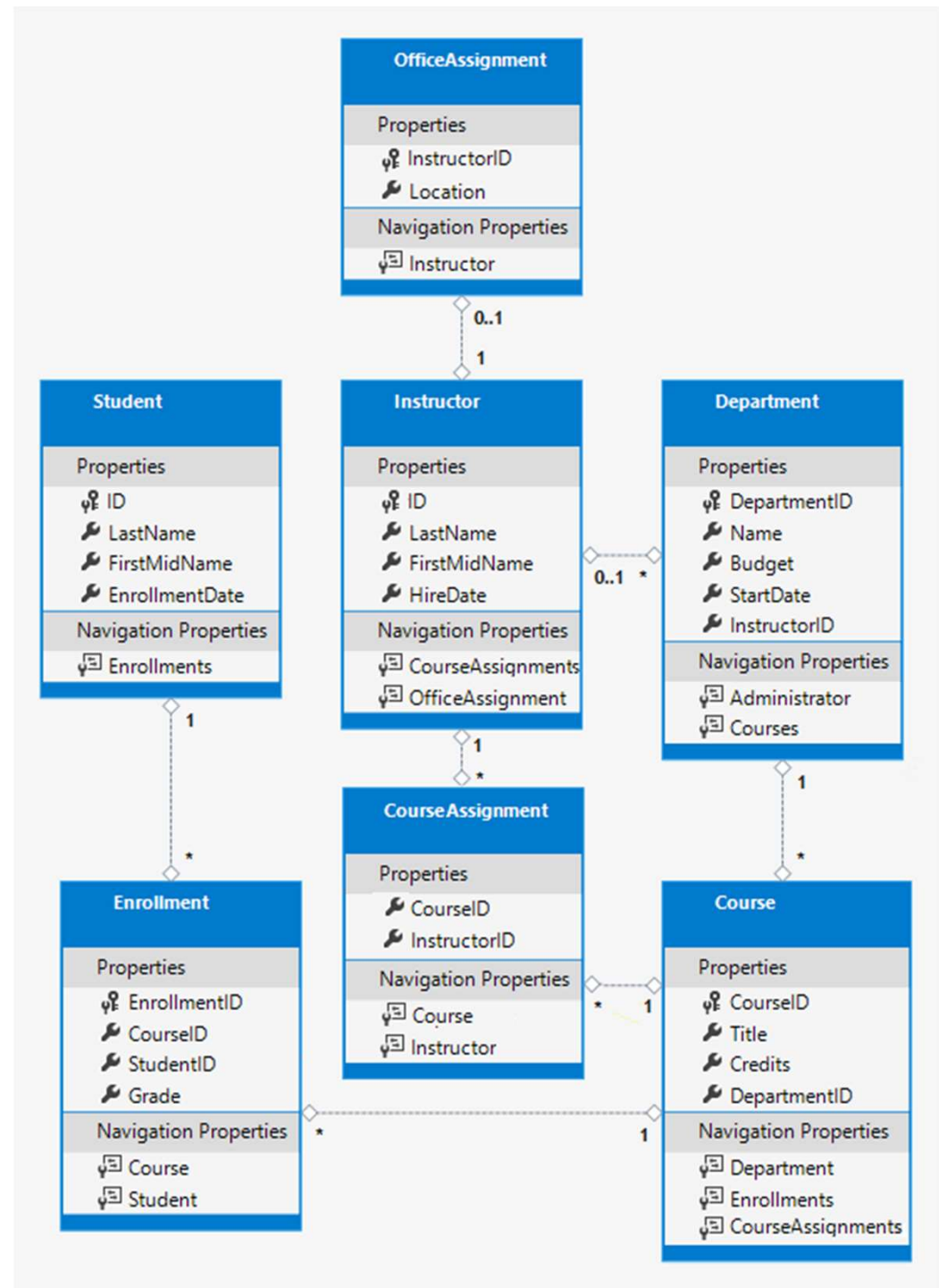
At the bottom of the page, there are two buttons: 'Previous' and 'Next'.

Tutorial: Step 5

Advanced Data Model

Advanced Data Model

- So far we've worked with a model with 3 entities (Student, Course, Enrollment)
- We'll be using data annotations to enhance our model
- We'll be adding more entities
- We'll implement validation
- We'll use migrations to update our database based on the new model
- We'll remind ourselves how many to many relationships are implemented



Data Annotations

C#

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace ContosoUniversity.Models
{
    public class Student
    {
        public int ID { get; set; }
        public string LastName { get; set; }
        public string FirstMidName { get; set; }
        [DataType(DataType.Date)]
        [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]
        public DateTime EnrollmentDate { get; set; }

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

Data Annotations

The **DataType** attribute provides the following benefits that are not available in `DisplayFormat`:

- The browser can enable HTML5 features. For example, show a calendar control, the locale-appropriate currency symbol, email links, client-side input validation, etc.
- By default, the browser renders data using the correct format based on the locale.

Data Annotations - Validation

More examples...

```
public int ID { get; set; }  
[StringLength(50)]  
public string LastName { get; set; }  
[StringLength(50, ErrorMessage = "First name cannot be longer than 50 characters.")]  
public string FirstMidName { get; set; }
```

Regex

```
[RegularExpression(@"^[A-Z]+[a-zA-Z""'\s-]*$")]
```


(Example: first character uppercase and rest alphabetical)

More Data Annotations

- [Required]
- [Display(Name = "Label Name")]

```
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; }
```


Data Annotations - Validation

Contoso University 

Create

Student

LastName

The field LastName must be a string with a maximum length of 50.

FirstMidName

First name cannot be longer than 50 characters.

EnrollmentDate

Migrating The Model Changes

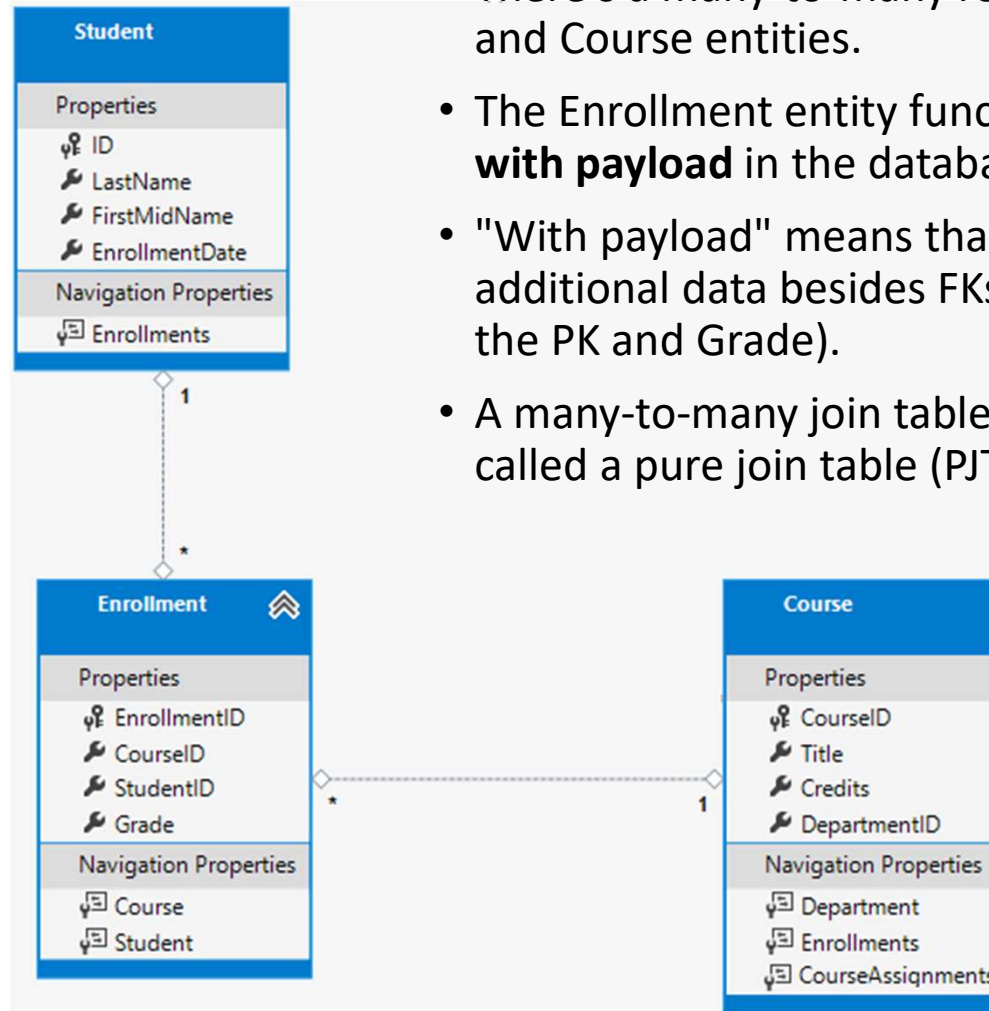
```
Add-Migration MigrationNameHere  
Update-Database
```

If the migration will potentially cause the loss of data, you will see the following warning:

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

The warning is generated because the name fields are now limited to 50 characters. If a name in the DB had more than 50 characters, the 51 to last character would be lost.

Many to many Relationships








- There's a many-to-many relationship between the Student and Course entities.
- The Enrollment entity functions as a many-to-many **join table with payload** in the database.
- "With payload" means that the Enrollment table contains additional data besides FKs for the joined tables (in this case, the PK and Grade).
- A many-to-many join table without payload is sometimes called a pure join table (PJT).

A Worked Example

Sending Email from your Razor Pages Project

Use Case: Sending Email

https://www.youtube.com/watch?v=E5SNMd8MO04&index=9&list=PLDmvsIp_VR0x2CmC6c4AZhZfYX7G2nBlo

9		Introduction to ASP.NET Core 2 Adding A Nuget Package Part 10 Eduonix Learning Solutions
10		Introduction to ASP.NET Core 2 Creating Backend Mail Send Workflow Eduonix Learning Solutions
11		Introduction to ASP.NET Core 2 Creating HTML Form With Tag Helpers Eduonix Learning Solutions
12		Introduction to ASP.NET Core 2 Contact Form Submission Part 13 Eduonix Learning Solutions
13		Introduction to ASP.NET Core 2 Tag Helpers Part 8 Eduonix Eduonix Learning Solutions

Use Case: Sending Email

Alternative email sending services

<https://sendgrid.com>

<https://www.mailgun.com>

<https://aws.amazon.com/ses/>



SendGrid

The AWS logo features the lowercase letters 'aws' in a bold, sans-serif font, with a yellow arrow pointing to the right underneath the letters.The Mailgun logo features a red '@' symbol followed by the word 'mailgun' in a lowercase, sans-serif font, with the 'i' in 'mail' and the 'u' in 'gun' being red.

A Worked Example

Uploading files

Use Case: Uploading Files

Tutorials

- <https://docs.microsoft.com/en-us/aspnet/core/razor-pages/upload-files?view=aspnetcore-2.1>
- <https://www.learnrazorpages.com/razor-pages/forms/file-upload>

Videos:

- https://www.youtube.com/watch?v=p7b_WLxoj30 (MVC approach from scratch)

Use Case: Uploading Files

The basic approach...

The front-end form:

```
@page
@model UploadFileModel
@{
}
<form method="post" enctype="multipart/form-data">
    <input type="file" asp-for="Upload" />
    <input type="submit" />
</form>
```

```

using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using System.IO;
using System.Threading.Tasks;
namespace RazorPagesForms.Pages
{
    public class UploadFileModel : PageModel
    {
        private IHostingEnvironment _environment;
        public UploadFileModel(IHostingEnvironment environment)
        {
            _environment = environment;
        }
        [BindProperty]
        public IFormFile Upload { get; set; }
        public async Task OnPostAsync()
        {
            var file = Path.Combine(_environment.ContentRootPath, "uploads", Upload.FileName);
            using (var fileStream = new FileStream(file, FileMode.Create))
            {
                await Upload.CopyToAsync(fileStream);
            }
        }
    }
}

```

IHostingEnvironment is injected into the constructor of the page model class via dependency injection, providing access to information about the current hosting environment, including the root folder

User Login Functionality

ASP.NET Core Identity

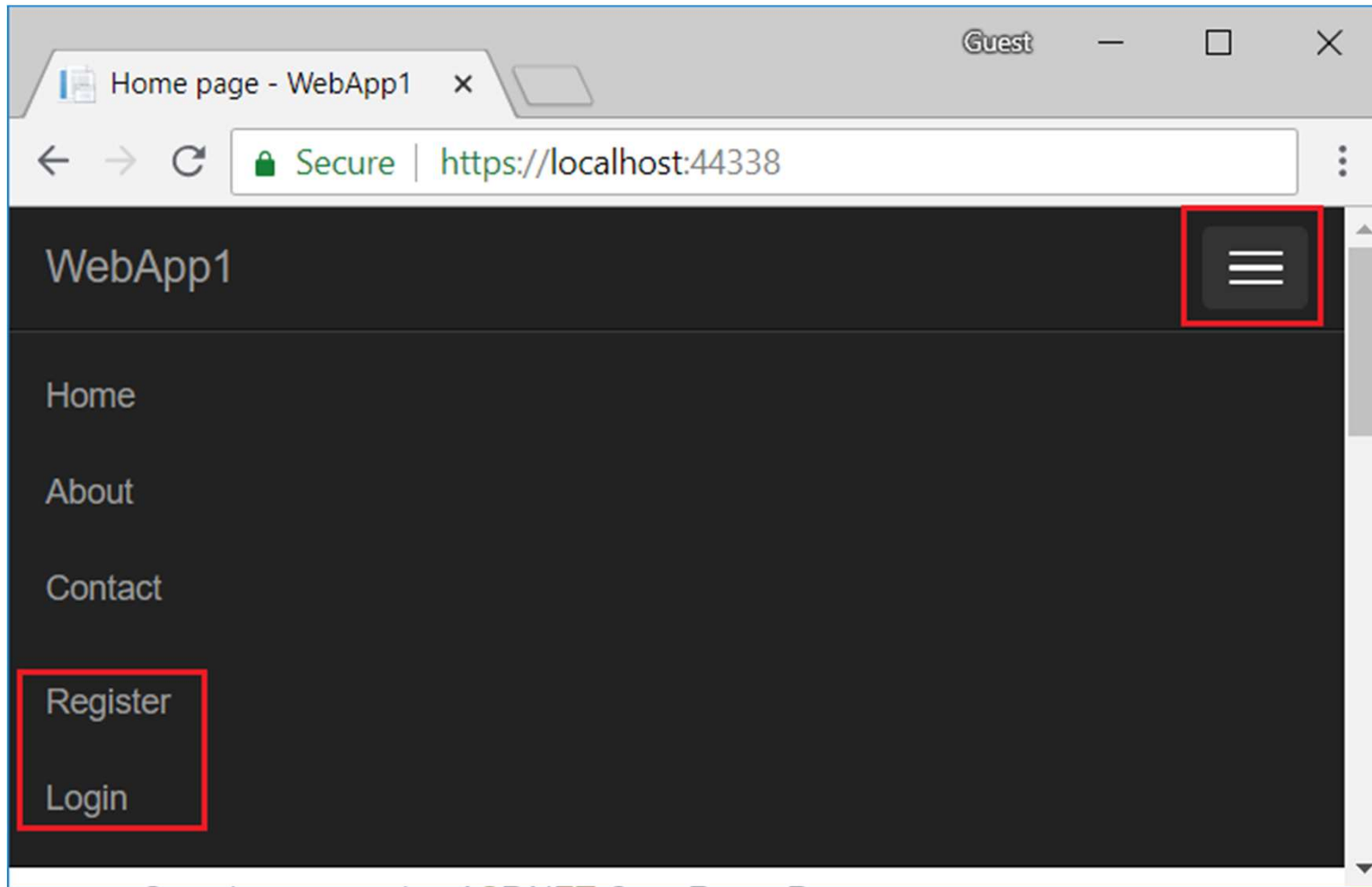
User Login (Identity)

Adding authentication to a new project...

- Select **File > New > Project**.
- Select **ASP.NET Core Web Application**. Name the project **WebApp1** to have the same namespace as the project download. Click **OK**.
- Select an ASP.NET Core **Web Application** for ASP.NET Core 2.1, then select **Change Authentication**.
- Select **Individual User Accounts** and click **OK**.

The generated project provides ASP.NET Core Identity as a Razor Class Library.

User Login (Identity)



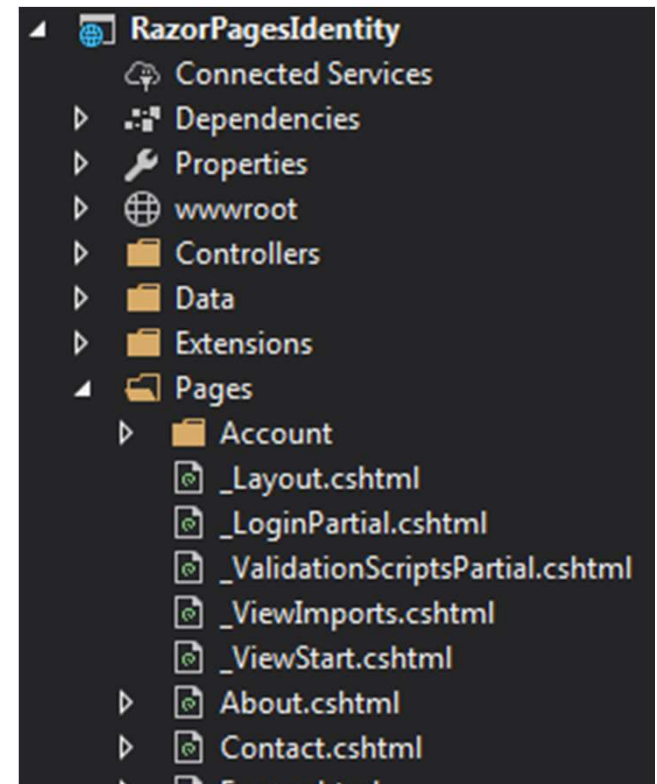
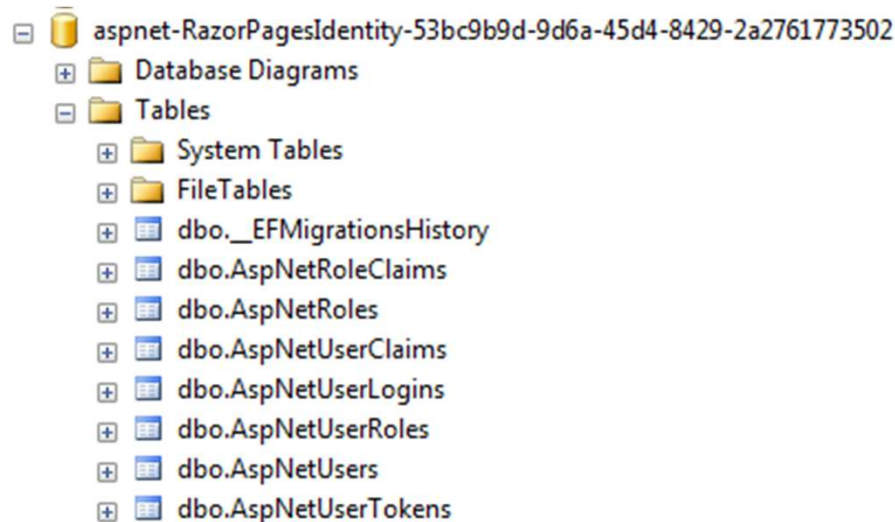
User Login (Identity)

Adding Identity to an existing Razor Pages project without existing authorization setup...

<https://docs.microsoft.com/en-us/aspnet/core/security/authentication/scaffold-identity?view=aspnetcore-2.1&tabs=visual-studio#scaffold-identity-into-a-razor-project-without-existing-authorization>

Adding Identity to a new project

<https://www.learnrazorpages.com/identity>

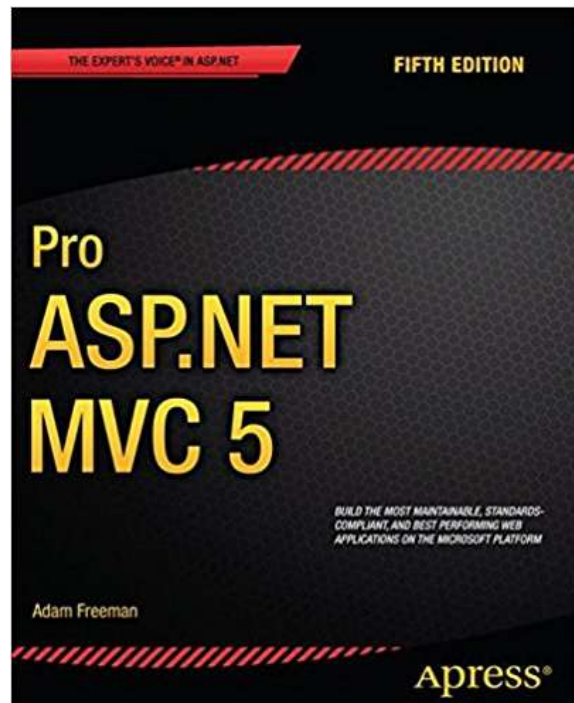


Further Reading Reminders

Further Reading

Pro ASP.NET MVC 5 (Fifth Edition)

Available online (and in print) via BNU Library



Chapter 1: Putting ASP.NET MVC in Context (p. 1-10)

Chapter 3: The MVC Pattern (p. 51-66)

- The history of MVC
- Understanding the MVC pattern
- Loose coupling
- Automated testing