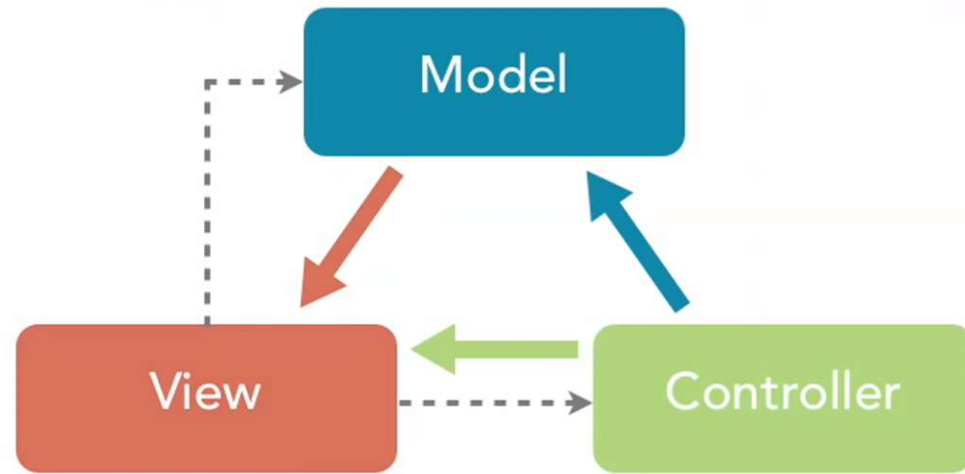


CO550 – Web Applications

UNIT 4 – Comparing MVC with Razor pages

WHAT IS MV^c

- Model View Controller



- an architectural pattern commonly used for developing user interfaces that divides an application into three interconnected parts
- Video: https://www.youtube.com/watch?v=xTPzz_2jbDc

CONTROLLERS

- A controller is responsible for controlling the way that a user interacts with an MVC application.
- A controller contains the flow control logic for an ASP.NET MVC application.
- A controller determines what response to send back to a user when a user makes a browser request.
- A controller is just a class (for example, a C# class)
- Controller actions e.g. **Index()** are exposed by a controller class e.g. **HomeController()**

Source: <https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/overview/understanding-models-views-and-controllers-cs>

Views

- Controller actions e.g. **Index()** return a view
- A view contains the HTML markup and content that is sent to the browser.
- A view is the equivalent of a page when working with an ASP.NET MVC application.

MODELS

- An MVC model contains all of your application logic that is not contained in a view or a controller.
- The model should contain all application business logic, validation logic, and database access logic.
- A **view** should contain only logic related to generating the user interface. A **controller** should only contain the bare minimum of logic required to return the right view or redirect the user to another action (flow control). Everything else should be contained in the **model**.
- In general, you should strive for *fat models* and *skinny controllers*.

Razor Pages vs MVC

- If you're using ASP.NET Core and you want to build **server-side web applications** then your two main options are MVC or Razor Pages
- Note: by talking about “**server-side web applications**” we're deliberately ignoring an alternative approach: building web APIs in order to serve front-end frameworks like Angular or React
- Which is better? Lot's of different viewpoints. Read around...
 - <https://stackoverflow.com/questions/46777404/why-is-razor-pages-the-recommended-approach-to-create-a-web-ui-in-asp-net-core-2>
 - <https://hackernoon.com/asp-net-core-razor-pages-vs-mvc-which-will-create-better-web-apps-in-2018-bd137ae0acaa>
 - https://www.reddit.com/r/dotnet/comments/7x1cjx/should_i_use_razor_pages_or_mvc_using_aspnet_core/
 - <https://stackify.com/asp-net-razor-pages-vs-mvc/>

Razor Pages vs MVC

Some key differences...

- A Razor Page is very similar to ASP.NET MVC's view component
- A key difference between Razor pages and MVC is that the model and controller code is included within the Razor Page itself.
- With Razor Pages, each page is self-contained with its view and code organized together.
- Different project folder structure

Razor Pages vs MVC – Folder Structure

The image displays two side-by-side file explorer views comparing the folder structures of an ASP.NET MVC application and a Razor Page application.

ASP.NET MVC (Left):

- RazorPageTest
 - Connected Services
 - Dependencies
 - Properties
 - wwwroot
 - Controllers
 - HomeController.cs
 - Models
 - PageClass.cs
 - Views
 - Home
 - About.cshtml
 - Contact.cshtml
 - Index.cshtml
 - ManagePage.cshtml
 - Shared
 - _ViewImports.cshtml
 - _ViewStart.cshtml
 - appsettings.json
 - bower.json
 - bundleconfig.json
 - Program.cs
 - Startup.cs

Razor Page (Right):

- RazorPageTest2
 - Connected Services
 - Dependencies
 - Properties
 - wwwroot
 - Pages
 - _Layout.cshtml
 - _ValidationScriptsPartial.cshtml
 - _ViewImports.cshtml
 - _ViewStart.cshtml
 - About.cshtml
 - Contact.cshtml
 - Error.cshtml
 - Index.cshtml
 - ManagePage.cshtml
 - ManagePage.cshtml.cs
 - appsettings.json
 - bower.json
 - bundleconfig.json
 - Program.cs
 - Startup.cs

ASP.net Core Razor Pages Tutorial

We will be working through a tutorial for this module:

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

The screenshot shows the Microsoft Docs website interface. At the top, there is a navigation bar with the Microsoft logo and links for Docs, Windows, Microsoft Azure, Visual Studio, Office, and More. Below this, the breadcrumb path is 'Docs / ASP.NET / ASP.NET Core'. On the right side of the breadcrumb path, there are links for 'Feedback' and 'Edit'. The main content area is divided into two columns. The left column is a sidebar for 'ASP.NET Core 2.1' with a search box labeled 'Filter by title'. The sidebar lists several categories, with 'Data access - with Razor Pages and EF Core' expanded to show a list of articles, including 'Get started' which is highlighted in blue. The right column displays the article title 'Razor Pages with Entity Framework Core in ASP.NET Core - Tutorial 1 of 8', the date '07/01/2017', the estimated reading time '15 minutes to read', and the authors 'Tom Dykstra' and 'Rick Anderson'. The article text begins with 'The Contoso University sample web app demonstrates how to create an ASP.NET Core Razor Pages app using Entity Framework (EF) Core.' and continues with 'The sample app is a web site for a fictional Contoso University. It includes functionality such as student admission, course creation, and instructor assignments. This page is the first in a series of tutorials that explain how to build the Contoso University sample app.' Below the text, there are links for 'Download or view the completed app' and 'Download instructions'. A 'Prerequisites' section is visible at the bottom, listing 'Visual Studio' and '.NET Core CLI' as required tools.

ASP.net Razor Tutorial

We will cover (among other things)...

1. Creating the ASP.NET core web app
2. Setting up a **data model**
3. **Scaffolding** the model (CRUD operations)
4. **Initialising a database** and seeding with test data
5. **Customising the UI** and modifying the scaffolded CRUD operations
6. How to **Sort, Filter, Page, and Group** data in the UI
7. Working with **database migrations** and why they are useful
8. Build out a more **complex data model** with data attributes and relationships


ASP.net Razor Tutorial – MAC OS

If you wish to attempt the tutorial on Mac instead of Windows:

- Download Visual Studio for Mac - <https://visualstudio.microsoft.com/vs/mac/>
- Ensure you have the .NET Core CLI installed
- Refer to the “.NET Core CLI” commands instead of the “Visual Studio” throughout the tutorial

Create the ContosoUniversity Razor Pages web app

Visual Studio **.NET Core CLI**

```
CLI  Copy
```

```
dotnet new webapp -o ContosoUniversity  
cd ContosoUniversity  
dotnet run
```

SUMMARY

- A brief overview of MVC
- Compared Razor Pages with MVC
- We will start working on a Razor Pages web application in Visual Studio

NOW: TUTORIAL Workshop

Step 1 of ASP.NET Core Razor Pages tutorial

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

1. Create the ASP.NET core web app
2. Setup a model
3. Scaffold the model (CRUD operations)
4. Initialise a database and seed with test data

Logbook 2

ASP.net Web Apps
Logbook 2 (Unit 4)

Task 1 (5 marks)

Complete all of the steps outlined on the ASP.net tutorial here:
<https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/database-first-development/setting-up-database>

Provide the following evidence in the form of screenshots:

- Provide evidence that all of the database tables have been created
- Provide evidence that you have created the post-deployment script
- Provide evidence that all of the data has been populated (into 3 tables)

[INSERT SOLUTION AND SCREENSHOTS HERE]

Question 2 (1 mark): Explain one benefit of using the Entity Framework (EF) approach instead of direct database querying?

[INSERT ANSWER HERE]