Tutorials

Follow this official Razor Pages web app tutorial, which takes you through creating an app that can display and manage a database of movies!

This tutorial here makes a slightly more fancy web site:

https://www.learnrazorpages.com/razor-pages/tutorial/bakery

And lastly, follow this fully online tutorial from Microsoft Learn. I probably recommend that you do it locally instead of in the browser, but that's up to you.

Extra Exercises:

Exercise 1

Using the dotnet CLI, create ASP.NET Core web applications of the following templates (one of each):

- Empty Web App (`dotnet new web`)
- Model-View-Controller (`dotnet new mvc`)
- Razor Pages (`dotnet new webapp`)

Compare the files and folders of the above templates.

- How are they different from one another?
- How are they similar?
- Are they different project types or simply different templates based on the same project type?
- How are they different from the Web API (`dotnet new webapi`) that we created last week?
- How are they different from a basic console application?

Consider: What is a SPA template (angular, react) and how is it different from the other options?

Going forward, we will work on a web application using the default web application template (`dotnet new webapp`). Create a new project of this type.

Exercise 2, Creating a Razor Page

In your project, create a new Razor page (.cshtml) in the *Pages* folder of your project. Use razor syntax to display the current time of the day on the page. Run your project and view the page in the browser.

Exercise 3, Using Razor Syntax

Use `ViewData` in your page file to show the same string in multiple places of the page (that is, without writing the string value multiple times).

Use a Razor code block to initialize a string with value *"Hello from code block"*. Display the value of the string on the page.

Using a Razor code block, create a list of strings. In your HTML, use embedded Razor syntax to display each element in the list.

Consider: What are the *_ViewStart.cshtml*, *_ViewImports.cshtml* and *_Layout.cshtml* files and what are they used for?

Exercise 4, Layout

Create your own HTML layout and use it in your Razor page.

Exercise 5, Creating a Model

Create a model for your newly created Razor page. Use the standard naming convention when naming the file ([pagename].cshtml.cs). Link the model to your page in your page file.

In your model file, create a public string property called `Message`. Set the value of that property in the `OnGet()` method to "Message from model". Use the `@model` directive in your Razor page to display the `Message` property to the client.

In your model file, create a method that takes a string argument and returns a string *"model says [string]"*. Call this method from the page so that the string is displayed to the client.

Change the routing of the page so that it uses a route different than the default one. Test the new route in the browser.

Exercise 6, Use your Web API

Have your web application retrieve data from the animal shelter Web API that you created last week (Remember to change the OnGet handler of your model to be asynchronous).

Hint: Use the `UseUrls` method on the `Webhost` in *Program.cs* to change the default host address and port, so that you can have both the web api and the web app running at the same time

Exercise 7, Building the Web App

Extend the application so that you can also buy cats using the animal shelter Web API. Take advantage of Tag Helpers when when creating the interface.

Exercise 8, Official Razor Pages Tutorials