Creating ASP.net Core WebApp

Oliver Badger

# Introduction to MVC

MVC or model view controller, is the architecture which the asp.net core web application runs. It utilises razor pages to dynamically assign data using C#. To make them dynamically you need to do 2 things:

* Update the view to accept the data.
* Update the controller to send the data.

Firstly change the view to accept data. In the example given is a string.



Whenever you are using C# and HTML you are using a built in engine called Razor.

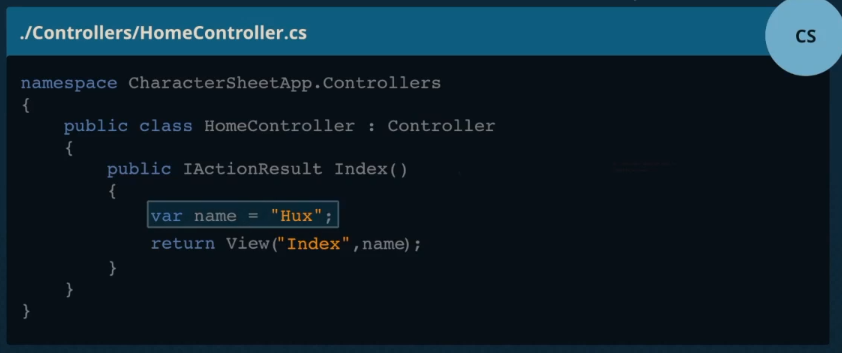
When using the data use a capital M when within the <li>@Model</li> and a non capital in the one above where is accepted:

* Lowercase “defines the type of data coming into the view”
* Uppercase “accesses the model data passed into the view”

The controller is separated into 3 separate parts.

1. Index - method
2. Class - home controller
3. Namespace - where you are

Methods that return IActionResult are called “Action Methods”, they prove responses usable by the browser.

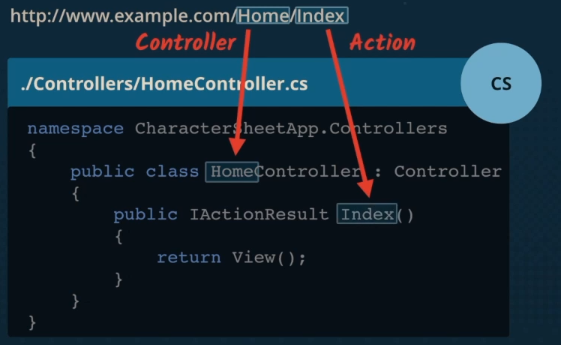


This goes into the controller to show what is being output.

Passes name in as a parameter for the view.

# Routing

Routes are how the application uses the url to determine what controller and controller action it want to access.The last action was /Home/Index



In this instance returning View will return the view in the home folder with the index action.

Default routes ensures that when the project is run, its the firs thing thats run. So if i typed in, roblox.com, it wouldnt need roblox.com/home/index to work.

# Models

We create a new class in models folder called character.cs. We make a new string called Name then initialise the class object and set that objects name to hux, that way we can just pass in the model. If we were to run the website after this change it would break because we havent told the asp.net web application to accept that, also the type or namespace name ‘Character’ could not be found. Its doing that because we dont have the namespace in the same folder. The namespace should be included in the model controller. Use model.name.