MVC – Model View Controller

Models

Akin to objects. Models the data of the website.

Game for instance holds information related to games:

- Title
- Author
- Date
- Play Count

```
namespace BucStop.Models
{
    public class Game
    {
        //The ID of the game. Starts at 1.
        public int Id { get; set; }

        public GameInfo[] Info { get; set; }

        //The title/name of the game.
        [Required]
        public string Title { get; set; }

        //The javascript file for the game
        [Required]
        [DataType(DataType.MultilineText)]
        public string Content { get; set; }

        //The author(s) of the game.
        [Required]
        public string Author { get; set; }
        //Shows the Date the game was added
        [Required]
        public string DateAdded { get; set; }
```

Structured like an object, but has additional information, such as [Required] that is caught by ASP to flag what data is needed when the model is created and used.

Controllers

Controllers manages the logic of the webapp and its views. It manages which views are used by the webpage (or layout) at a given time.

```
namespace BucStop.Controllers
{
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;
        private readonly GameService _gameService;

        public HomeController(ILogger<HomeController> logger, GameServ
        {
            _logger = logger;
            _gameService = games;
        }

        //Sends the user to the deprecated Index page.
        public IActionResult Index()
        {
            return View(_gameService.GetGames());
        }

        //Takes the user to the admin page.
        public IActionResult Admin()
        {
            return View();
        }

        //Takes the user to the about policy page.
        public IActionResult Privacy()
        {
            return View();
        }
```

Controllers hold functions that return views. These functions are then called in the webapp to dictate which view is used at a given moment. The return value of View() can take in additional parameters to specify which view will be served. If it is left blank, it will default to the method name – i.e. Admin would grab the Admin view.

```
<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
    <ul class="navbar-nav flex-grow-1">
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Games" asp-action="Index">Games</a>
        </li>
```

Controllers can be called using a links. These take in an asp-controller and asp-action. The action corresponds to the methods within the given controller.

Views

Views are html templates that are called by the controller and served to the user. They use the .cshtml file type.

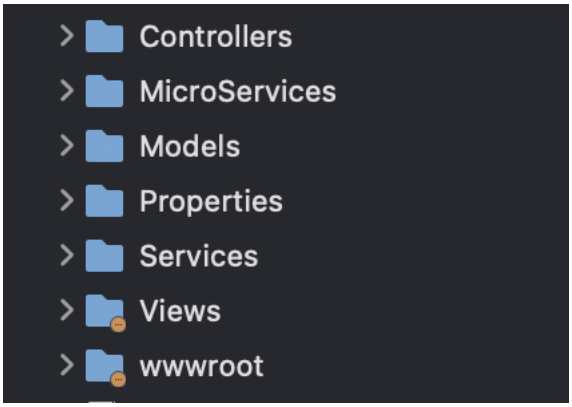
They are usually not full html pages i.e. no header and footers

```
@{
    ViewData["Title"] = "Admin Page";
}

<!-- This page will hold the page where admins can interact with the site. Not currently implemented.-->

<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    <table>
        <tr>
            <td>Admin action 1</td>
            <td>Admin action 2</td>
            <td>Admin action 3</td>
            <td>Admin action 4</td>
        </tr>

    </table>
    <p>Learn about <a href="https://docs.microsoft.com/aspnet/core">building Web apps with ASP.NET Core</a>.</p>
</div>
```

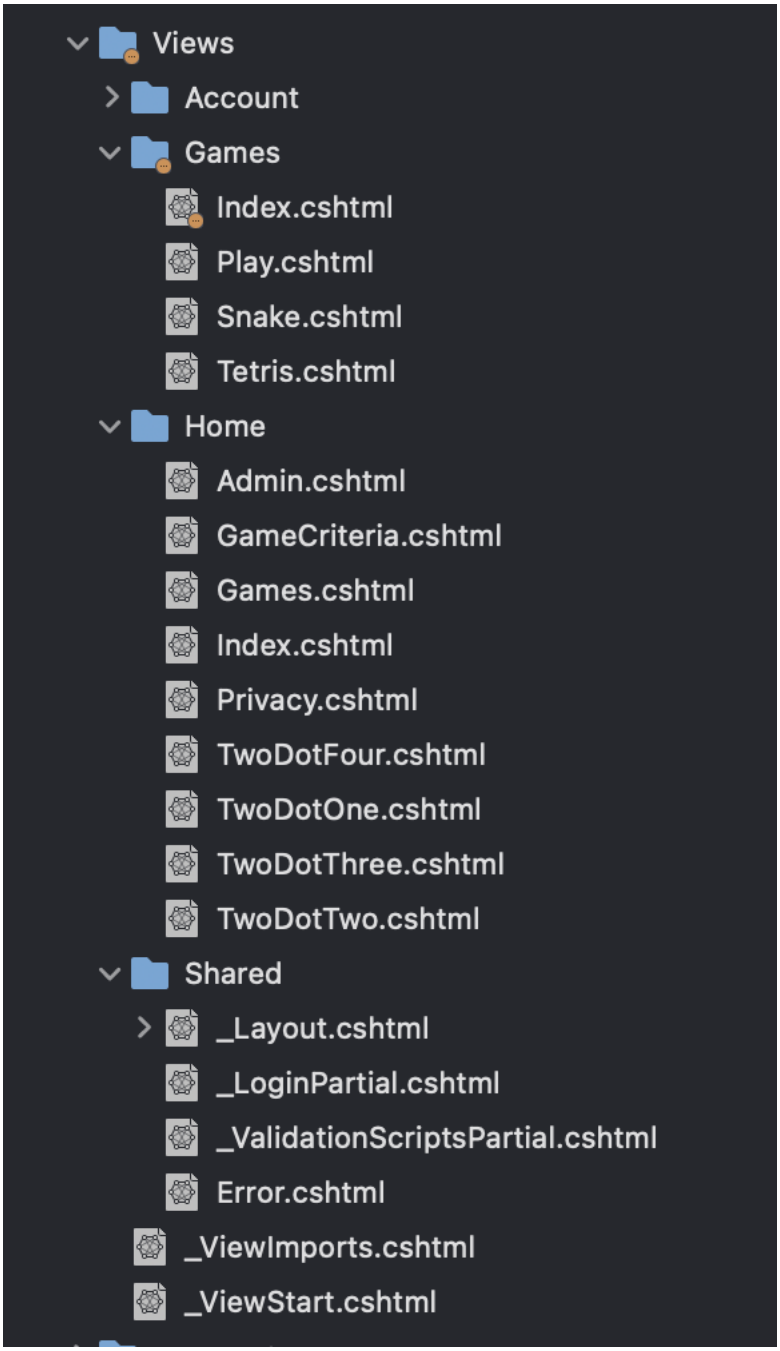
View data can be added to the top of the file

File Structure

File structure is very important. While pathnames can be used for views, Controllers will typically look in specific folders for views.



Here is a top-level view of what a ASP MVC file structure might look like. While most of the folders contain files, views require a bit more in-depth structure.



Here you can see the structure for views. We have multiple folders. Shared is a folder for views that will be used between all pages. Scroll down for more info.

The other folders (Account, Games, Home) all correlate to controllers. Underneath the folders are the specific views. ASP uses this folder structure and naming scheme to properly identify which views are meant to be rendered.

Naming Scheme

ASP MVC uses a specific naming scheme that helps it identify controllers and views.

Methods that return an empty view need to match the name of a view underneath the appropriate folder.

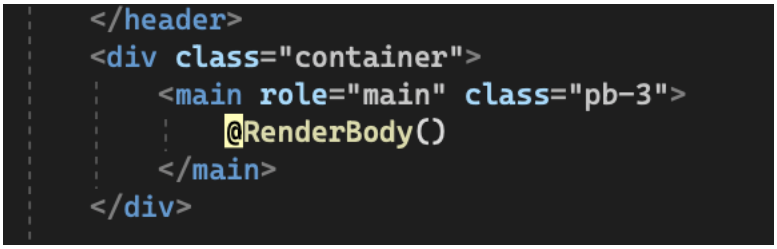
Controllers follow the convention of [Name]Controller.cs

The name is then used for ASP controller in views.

Partial Layout

Views are usually not whole webpages but rather portions of a webpage.

The rest of the webpage is usually served up via the Views/Shared/_Layout.cshtml, although other layout names can be used as well if other partial views are required.



The layout typically contains a @RenderBody call that identifies where in the html a view should be added.