ASP.NET MVC

CONVENTIONS

issntt@nus.edu.sg

Objectives

At the end of this lesson, students will be able to

- Describe the roles of conventions in web development
- Describe some key conventions implemented in ASP.NET Core and use them for development and troubleshooting

Topics

- What is convention?
- Some important conventions in ASP.NET Core
  - Naming
  - Default Route
  - Folder
  - Communication
  - Layout
- Pros (and cons) of conventions

What is convention?

The **way** in which something is done that **most people** in a society **expect** and **consider** to be polite or **the right way** to do it

The Oxford Advanced Learner's Dictionary

How is it helpful in web development?

Conventions

ASP.NET Core philosophy is **convention over configuration.** Some conventions include:

Naming

How to **name certain files**?

Default Route

For a **request URL**, which Controller/Action will be **mapped**?

Folder

Where to **place certain files**?

Communication

How to **pass data** among different components?

Layout

What **file** contains the default layout?

Naming

Controller classes should always **end** with *`Controller`* suffix

```
public class MoviesController : Controller {
    public Task<IActionResult> Details(int? id)
    {
        // Implementation is omitted for brevity
    }
}
```

```
public class HomeController : Controller {
    public IActionResult Index() {
        return View();
    }

    public IActionResult Privacy() {
        return View();
    }
}
```

Naming

An **action** is a **method with the same name** inside the respective Controller class

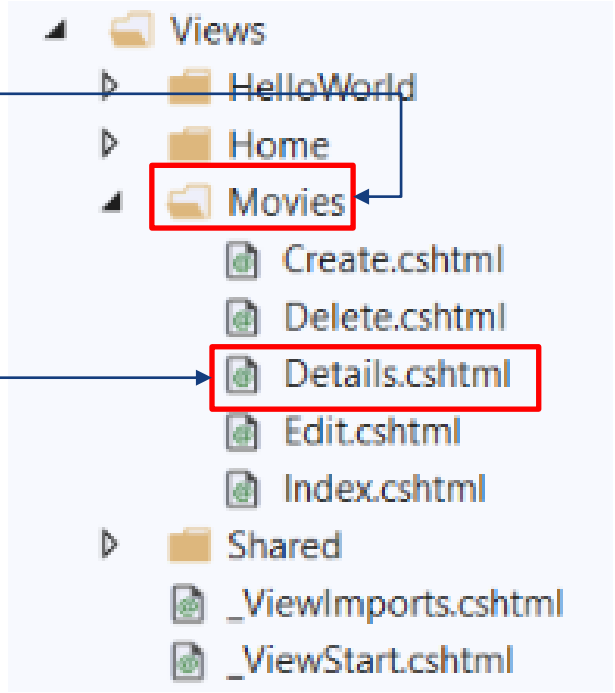
```
public class MoviesController : Controller {
    public Task<IActionResult> Details(int? id)
    {
        // Implementation is omitted for brevity
    }
}
```

```
public class HomeController : Controller {
    public IActionResult Index() {
        return View();
    }

    public IActionResult Privacy() {
        return View();
    }
}
```

Naming

Views should go into folder */Views/<Controller name>/* and carry the **same name** as the respective **Action method**

```
public class MoviesController : Controller
{
    public Task<IActionResult> Details(int? id)
    {
        // Implementation is omitted for brevity
    }

    // Other actions are omitted for brevity
}
```



What **other** actions are **likely** to be inside this controller?

Default Route

Routing is how we **turn URLs** in requests **into actions**.

Default route is:

```
app.UseEndpoints(endpoints =>
{
   endpoints.MapControllerRoute(
      name: "default",
      pattern: "{controller=Home}/{action=Index}/{id?}");
});
```

Default route pattern is

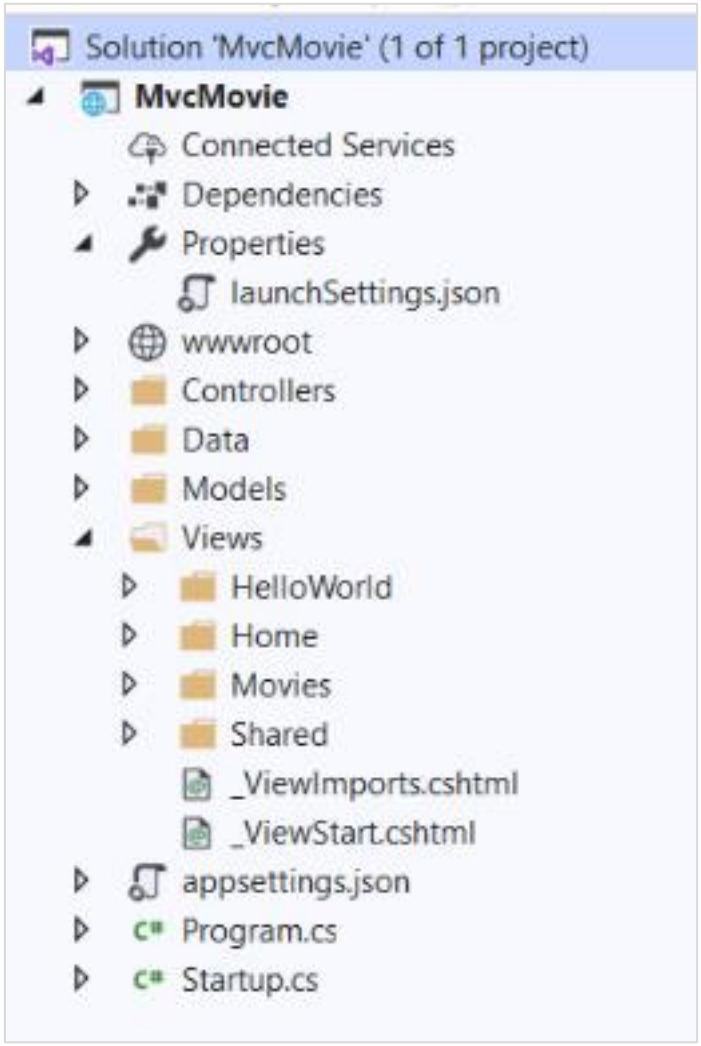
*[**Controller** Name]/[**Action** Name]/[**ID** (optional parameter)]*

What is the Controller, Action and ID for this URL:

https://localhost:8888/Movies/Details/5

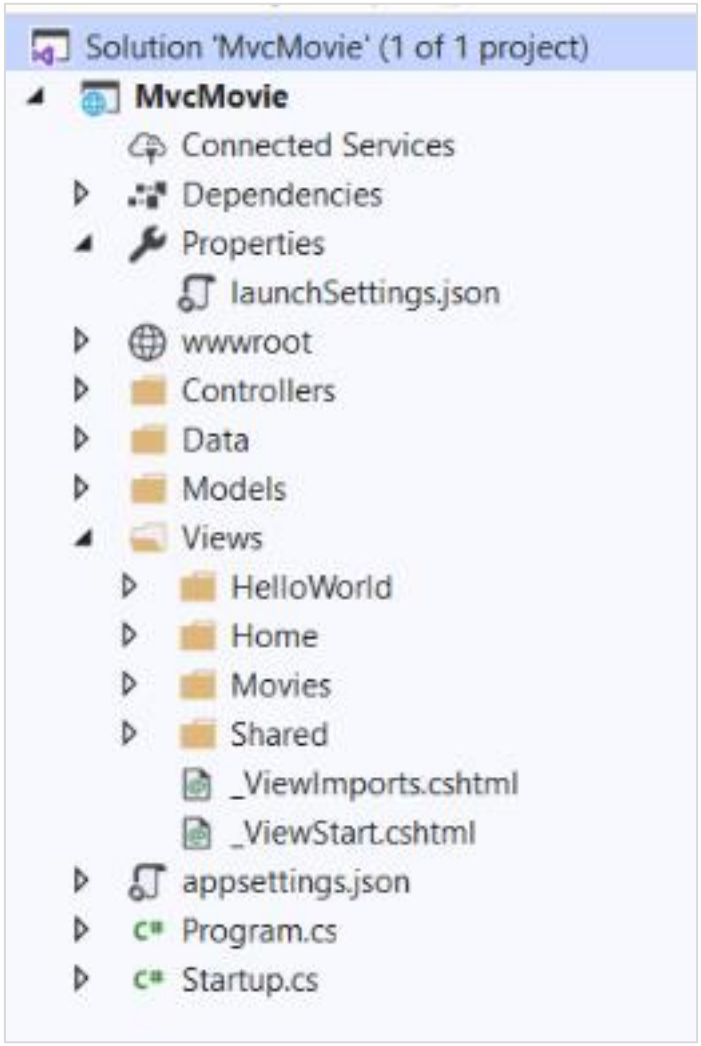
Folder

- **Controllers**
  - Contains all Controller classes
- **Views**
  - Contains all View classes
  - Grouped by the **respective Controller**, or Shared
  - ***/Views/Shared*** folder stores HTML snippets that are **shared for all views** (e.g., layouts…)
- **Models**
  - Contains all Data model classes



Folder

- **wwwroot**
  - Contains **static files** such as css and js. We can add other content such as images, video files…
- **appsettings.json**
  - Contains **application's configurations** (e.g., database connection string)
- **Program.cs**
  - Contains the `Main()` method to build and run the webhost,
- **Startup.cs**
  - Contains methods to **configure the application**, i.e., register and configure services, routes…



Communication

One way a Controller **sends data** to Views is to **write** into the **given *ViewData*** object

```
public class HelloWorldController : Controller
{
    // GET: /HelloWorld/Welcome/
    public IActionResult Welcome(string name, int numTimes = 1)
    {
        ViewData["Message"] = "Hello " + name;
        ViewData["NumTimes"] = numTimes;

        return View();
    }
}
```

ViewData is a **dictionary** type and therefore accept **key-value pair** objects

Communication

View then **read** data from the *ViewData* object by **providing keys** to **retrieve respective values**

```
<h2>Welcome</h2>
<ul>
   @for (int i = 0; i < (int) ViewData["NumTimes"]; i++) {
      <li>(string) @ViewData["Message"]</li>
   }
</ul>
```

In many cases, **type casting** is needed

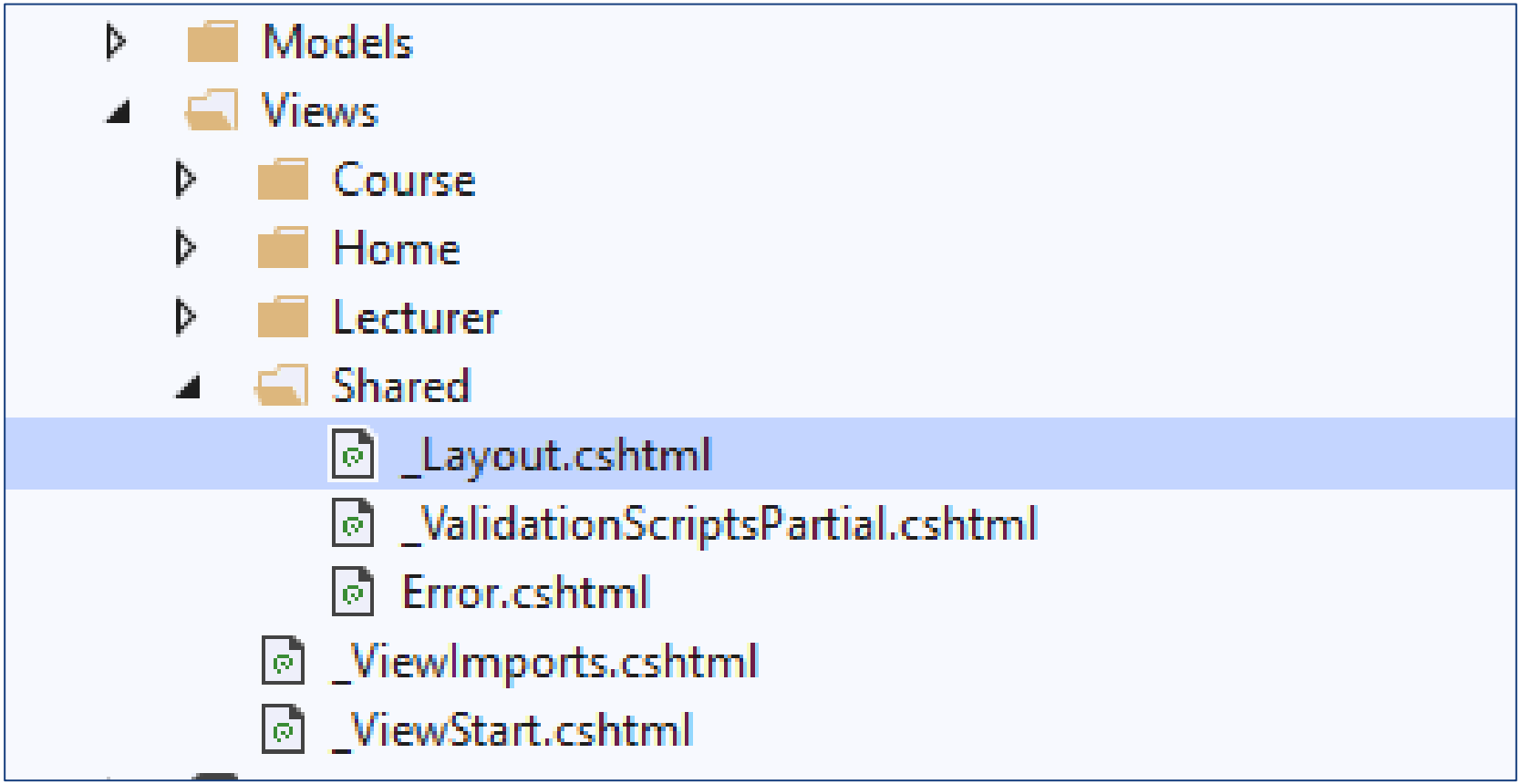
Besides writing/reading to/from *ViewData* object, another common way for Controllers to send data is using *ViewModel*

In the example, we use **string** as **keys** and **string** or **int** as **values**

What **other data types** can we use as **values**?

Layout

The **layout file** will **generate** the **final HTML**, and the **default** layout file is ***Views/Shared/_Layout.cshtml***



Layout

Any layout file must call ***@RenderBody()*** and **content generated** by a specific **view file** (e.g. ***Index.cshtml***) will **go there**

```
<!DOCTYPE html>
<html lang="en">
<head>...</head>
<body>
    <body>...</body>
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>

    <body>...</body>
    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>
    @RenderSection("Scripts", required: false)
</body>
</html>
```

Why Conventions?

- **Minimize** the amount of code
- **Reduce** of number of **decisions** we have to make
- Be much **easier** for **code maintenance**

But if we **don't know** some conventions, it will become **extremely difficult!**

Review

- Why does ASP.NET Core implement conventions?
- How does a Controller know which default View to call?
- Where are we likely to find information about Routing rules?
- How can a Controller send information to a View?
- When running our program, it complains that View cannot be found. What are likely the issues?
- After adding another JavaScript library into all pages in our app, what file are we likely to change?

Readings

- The Secret Code: ASP.NET MVC Conventions
  https://jeremybytes.blogspot.com/2020/02/the-secret-code-aspnet-mvc-conventions.html