

In ASP.NET Core, the layout view is a webpage responsible for containing the presentation logic that is common to all views.

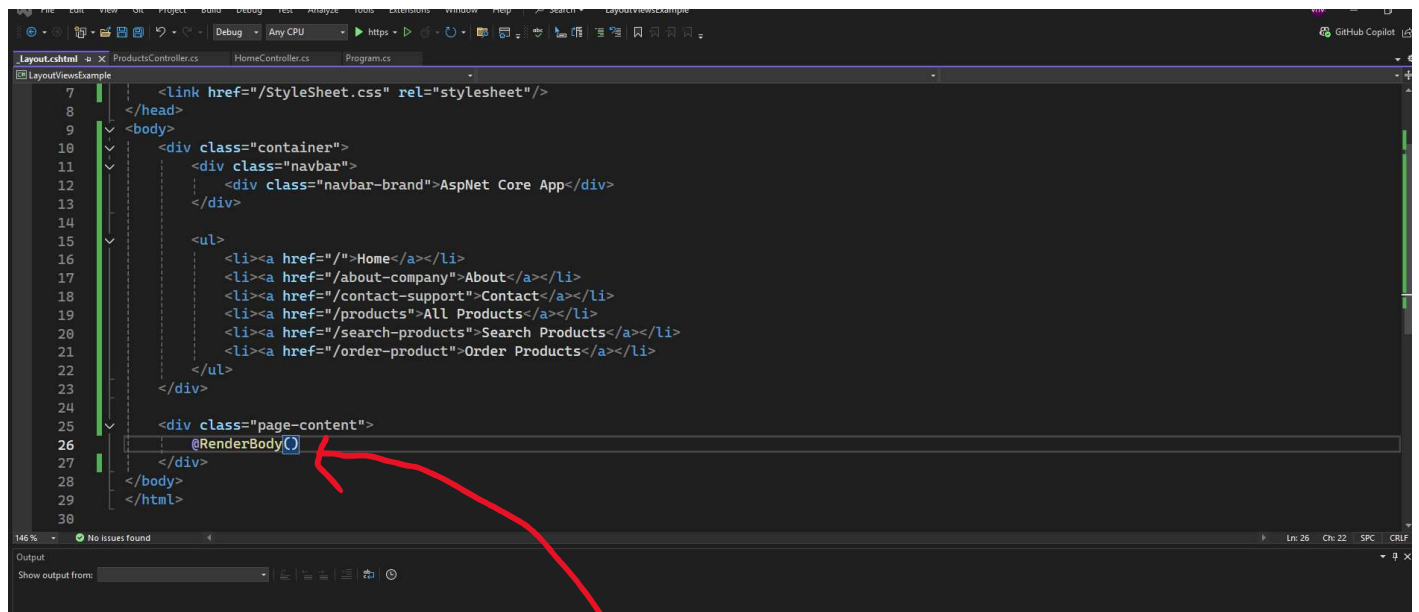
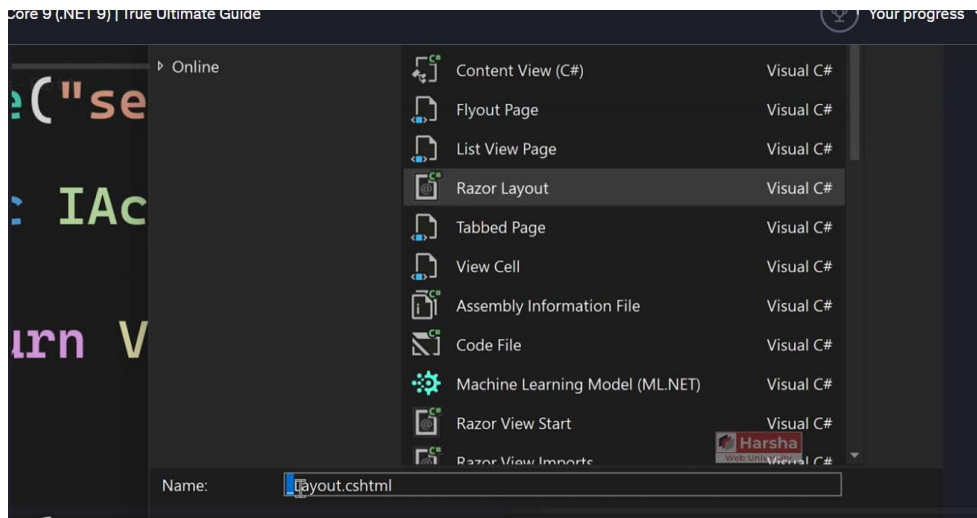
For example, if your project has 10 views and you want to maintain a consistent look and feel, along with a common header, sidebar, and footer, you would place that shared design in the layout view.

In the header, you can include the application title and navigation bar.

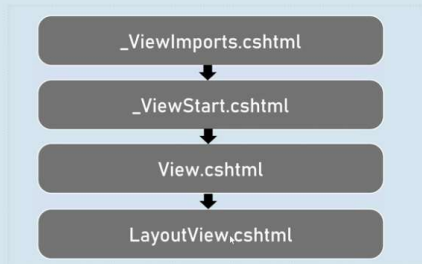
In the sidebar, you can add navigation links.

In the footer, you can display client information or copyright details to show at the bottom of the page.

The common content placed in the layout view is automatically repeated for every view in the application.



Order of Views Execution



At execution time, the **view executes first**, and the content of the view is substituted in place of the `@RenderBody` directive in the layout view.

Asp.Net Core Layout Views Part 2

```
1 <!DOCTYPE html>
2
3 <html>
4 <head>
5   <meta name="viewport" content="width=device-width" />
6   <title>Asp.Net Core Demo App</title>
7   <link href="/StyleSheet.css" rel="stylesheet" />
8 </head>
9 <body>
10  <div class="container">
11    <div class="navbar">
12      <div class="navbar-brand">AspNet Core App</div>
13      <ul>
14        <li><a href="/">Home</a></li>
15        <li><a href="/about-company">About</a></li>
16        <li><a href="/contact-support">Contact</a></li>
```

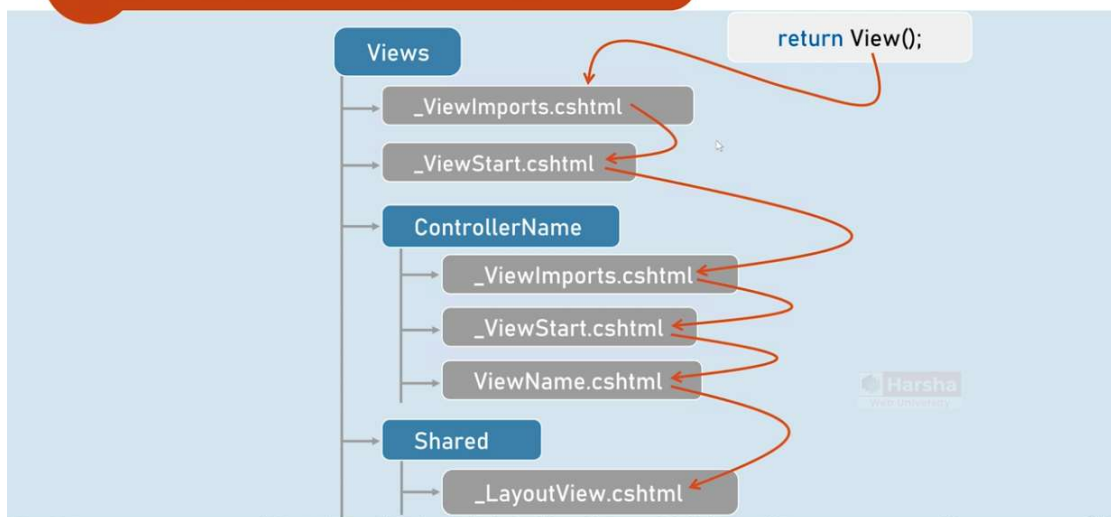
Asp.Net Core

Layout Views with Multiple Views

Asp.Net Core

| Harsha

Order of Views Execution



Asp.Net Core

| Harsha

Layout Views

The `@RenderBody()` method presents only in layout view to represent the place where exactly the content from the view has to be rendered.

Both View and Layout View shares the same ViewData object.

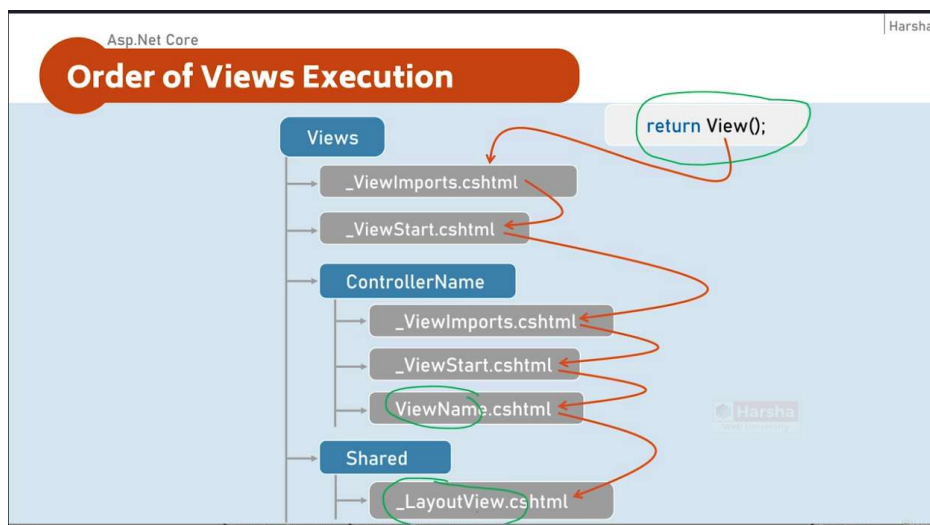
So it is possible to send data from view to layout, since the view executes first.

The "Layout" property of the view specifies path of the layout view. It can be dynamically set in the view.

The css files / js files imported in layout view will be applicable to view also, because the content of view will be merged into the layout view at run time.

Asp.Net Core

ViewData in Layout Views



This is the **order of execution** whenever you say `return View()` in the controller:

1. It first executes the `_ViewImports` and `_ViewStart` files in the **global Views folder**.
2. Then, it moves to the **controller-specific folder** within the Views folder and executes the `_ViewImports` and `_ViewStart` files there.
3. Next, the **actual view** executes.
4. Finally, the **layout view** executes.

In this sequence, whether in the **controller**, **view**, or **layout view**, you can access the same `ViewData` object. This means you can transfer information via `ViewData` in the same execution order.

For example:

- If you assign a value to `ViewData` in the **controller**, you can access that data in the subsequent files, such as the view or the layout view.
- Similarly, if you assign a value to `ViewData` in the **view**, you can access it in the **layout view**.

Use case example: Based on this, views often supply the **page title** from the view to the layout view. Let me demonstrate this.

```

1  @* Let's connect this view to Layout view *@
2
3  @{
4      Layout = "~/Views/Shared/_Layout.cshtml";
5      ViewData["Title"] = "Home";
6  }
7
8  <h1>Home</h1>
9  <p>Welcome to home page</p>

```

This ViewData can be accessed in the layout view

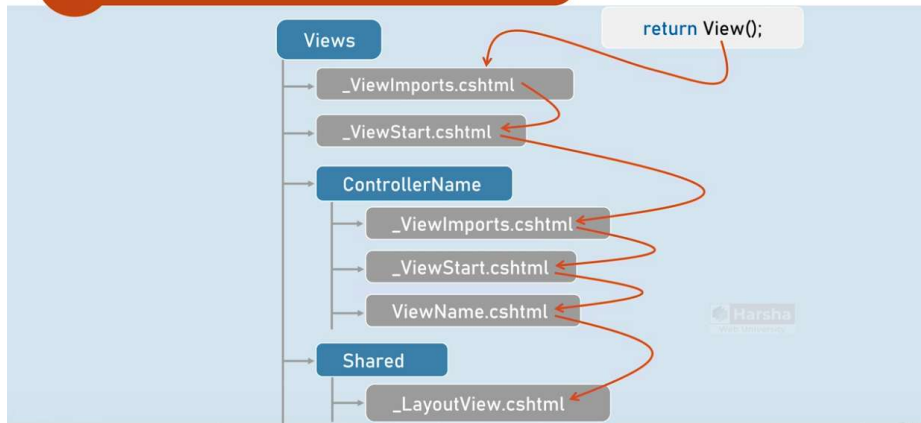
```

1  <!DOCTYPE html>
2
3  <html>
4  <head>
5      <meta name="viewport" content="width=device-width" />
6      <title>@ViewData["Title"]</title>
7      <link href="/StyleSheet.css" rel="stylesheet"/>
8  </head>
9  <body>
10     <div class="container">
11         <div class="navbar">
12             <div class="navbar-brand">AspNet Core App</div>
13
14             <ul>
15                 <li><a href="/">Home</a></li>
16                 <li><a href="/about-company">About</a></li>
17                 <li><a href="/contact-support">Contact</a></li>
18                 <li><a href="/products">All Products</a></li>
19                 <li><a href="/search-products">Search Products</a></li>
20                 <li><a href="/order-product">Order Products</a></li>
21             </ul>
22         </div>
23     </div>

```

AspNet Core
_ViewStart

Order of Views Execution



Just like the `_ViewImports.cshtml` file, the `_ViewStart.cshtml` is a special file that gets executed **before** the execution of a view. It is used to specify the **common layout view** for multiple views in the same folder.

For example:

If you have a `_ViewStart.cshtml` file and you write the following statement:

```
'Layout = "SampleLayout";'
```

Then the same layout is applied to **all the views** within the same controller's folder.

Benefits:

The developer does not need to repeat the `'Layout'` statement (e.g., `'Layout = "SomeFileName"'`) in every individual view. Instead, the `_ViewStart.cshtml` acts as a **common setting**.

Additionally, you can create a **global `_ViewStart.cshtml`** file at the **root Views folder**, which applies globally to all views in the project.

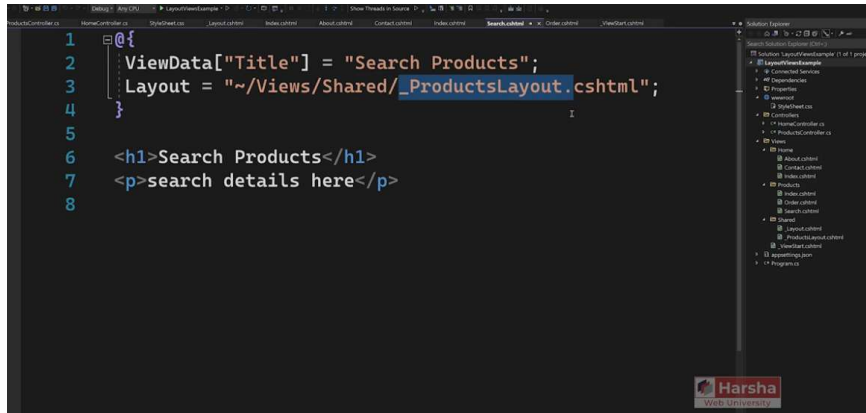
Execution Order:

1. The **global `_ViewStart.cshtml`** (in the root Views folder) executes first.
2. The **local `_ViewStart.cshtml`** (in the controller's Views folder) executes next and **overrides** the global layout setting if necessary.

This allows developers to override the layout file specified in the global `_ViewStart.cshtml` by providing a different layout in the local `_ViewStart.cshtml` file.

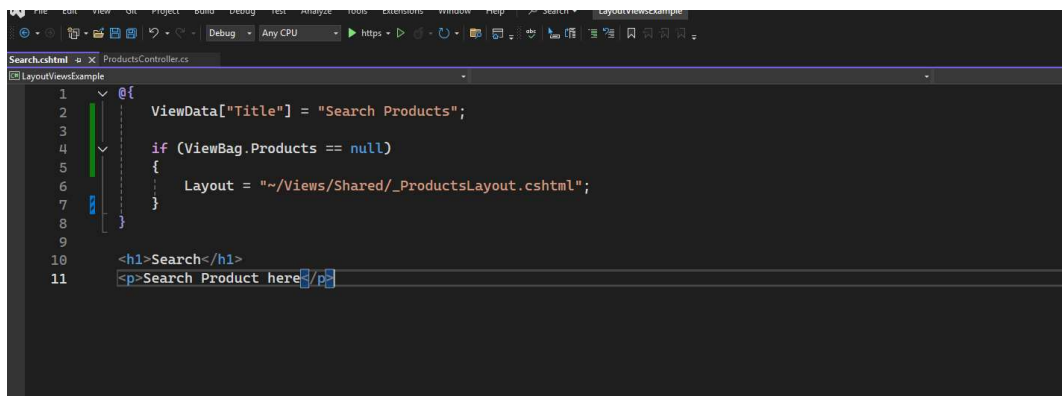
Asp.Net Core

Dynamic Layout Views



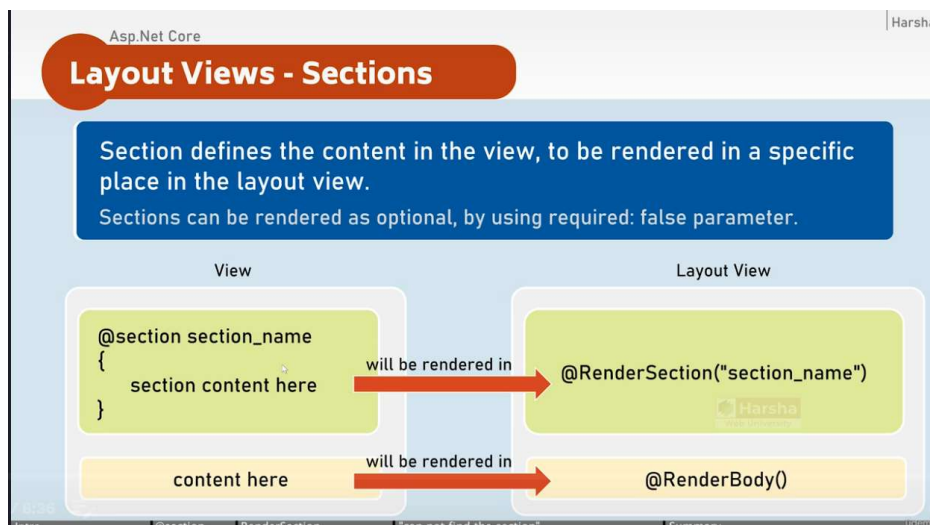
```
1 @{
2     ViewData["Title"] = "Search Products";
3     Layout = "~/Views/Shared/_ProductsLayout.cshtml";
4 }
5
6 <h1>Search Products</h1>
7 <p>search details here</p>
8
```

Dynamic layout view means you are going to set the layout property of the view dynamically based on the condition. For example, you will write a condition saying that if the user has supplied the value for this parameter, then you will set one layout view; if it is not, another layout view. So, you are going to set the layout view of a view dynamically.



```
1 @{
2     ViewData["Title"] = "Search Products";
3
4     if (ViewBag.Products == null)
5     {
6         Layout = "~/Views/Shared/_ProductsLayout.cshtml";
7     }
8 }
9
10 <h1>Search</h1>
11 <p>Search Product here</p>
```

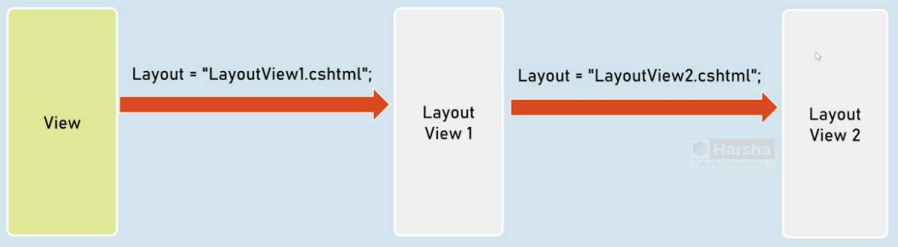

Asp.Net Core Layout Views Sections



Asp.Net Core Nested Layout Views

Nested Layout Views

A layout view that has another layout view is called as 'nested layout view'.



The meaning of a nested layout view is that you apply one layout view to another layout view. For example, there is a view that has Layout View 1, but you have applied another layout view called Layout View 2 to this Layout View 1. In other words, the master page itself has another master page. This is called a nested layout view. It is quite rare to use in real-world projects, but this lecture is meant just to show that it is possible.