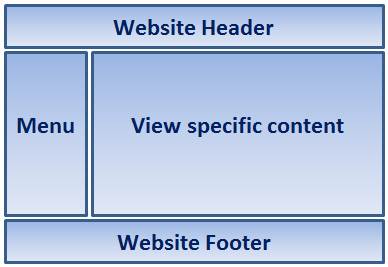
**Layout view in asp.net core mvc**

**Why a Layout View is required**  
  
Most of the web applications in general have the following sections

* Header
* Footer
* Navigation menu
* View specific content

   
  
Without a Layout View, we would repeat all the HTML for these different sections in every view in our application. With this duplicated HTML in every view, maintaining our application would be a night mare, if for example, we have to add or remove a menu item from the navigation menu or change header or footer. We would have to do this change, in every view, which is obviously tedious, monotonous, time consuming and error prone.   
  
Rather than having all of these sections, in each and every view, we can define them in a layout view and then inherit that look and feel in all the views. With layout views, maintaining the consistent look and feel across all the views becomes much easier, as we have only one layout file to modify, should there be any change. The change will then be immediately reflected across all the views in our entire application.   
  
**Layout View in ASP.NET Core MVC**

* Just like a regular view a layout view is also a file on the file system with a **.cshtml**extension
* You can think of a layout view as a master page in asp.net web forms.
* Since a layout view is not specific to a controller, we usually place the layout view in a sub folder called **"Shared"** in the **"Views"** folder
* By default, in ASP.NET Core MVC, the layout file is named **\_Layout.cshtml**.
* In ASP.NET Core MVC, the file names of other .cshtml files like **\_ViewStart.cshtml** and **\_ViewImports.cshtml**start with an underscore.
* The leading underscore in the file name indicates that these files are not intended to be served directly by the browser.
* It is also possible to have multiple layout files in a single application. Perhaps one layout file for all admin users and a different layout file for all non admin users.

**Creating a Layout View**

1. Right click on **"Views"** folder and add **"Shared"** folder.
2. Right click on **"Shared"** folder and select **"Add"** - **"New Item"**
3. In the **"Add New Item"** window search for Layout
4. Select **"Razor Layout"** and click the **"Add"** button
5. A file with name **\_Layout.cshtml**will be added to the **"Shared"** folder

**Auto-generated HTML in \_Layout.cshtml file**  
  
The following is the default generated HTML in **\_Layout.cshtml**.

<!DOCTYPE html>

<html>

<head>

    <meta name="viewport" content="width=device-width" />

    <title>@ViewBag.Title</title>

</head>

<body>

    <div>

        @RenderBody()

    </div>

</body>

</html>

Notice, the standard html, head, title and body elements are in this layout file. Since we now have them in the layout file, we do not have to repeat all this HTML in every view. At the moment, we do not have a navigation menu. We will include it in the layout view in our upcoming videos.   
  
View specific title is retrieved using @ViewBag.Title expression. For example, when "index.cshtml" view is rendered using this layout view, index.cshtml will set Title property on the ViewBag. This is then retrieved by the Layout view using the expression @ViewBag.Title and set as the value for the <title> tag.   
  
ViewBag does not provide intellisense and compile-time error checking. So using it to pass large amount of data from a regular razor view to a Layout view is not great, but for passing something very small like PageTitle, ViewBag is OK.   
  
@RenderBody() is the location where view specific content is injected. For example, if index.chtml view is rendered using this layout view, index.cshtml view content is injected at the location where we have @RenderBody() method call.   
  
**Using the Layout View**  
  
To render a view using the layout view (\_Layout.cshtml) set the Layout property. For example, to use the Layout view with details.cshtml, modify the code in details.cshtml to include the Layout property as shown below.

@model EmployeeManagement.ViewModels.HomeDetailsViewModel

@{

    Layout = "~/Views/Shared/\_Layout.cshtml";

    ViewBag.Title = "Employee Details";

}

<h3>@Model.PageTitle</h3>

<div>

    Name : @Model.Employee.Name

</div>

<div>

    Email : @Model.Employee.Email

</div>

<div>

    Department : @Model.Employee.Department

</div>

We have a better way of setting the Layout property, instead of setting it in every view.