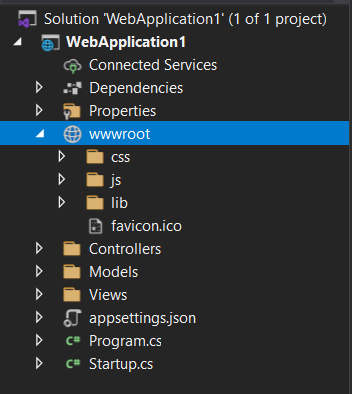
https://learn.microsoft.com/ru-ru/aspnet/core/mvc/views/layout?view=aspnetcore-6.0



**wwwroot** містить статичні файли, як-от файли CSS, файли зображень і значків

**Папка Controllers** містить файли класів для контролерів. A Controller обробляє запити користувачів і повертає відповідь. MVC вимагає, щоб ім’я всіх файлів контролера закінчувалося на «Controller».

Моделі

**Папка Models** містить файли класів моделей. Зазвичай клас моделі включає загальнодоступні властивості, які використовуватимуться програмою для зберігання та обробки даних програми.

### Представлення

**Папка Views** містить файли HTML для програми. Зазвичай файл представлення – це файл .cshtml, у якому пишеться код HTML і C#. Папка Views містить окрему папку для кожного контролера. Наприклад, усі файли .cshtml, які буде відтворено HomeController, будуть у папці «Controllers» > «Home».

Папка Shared  в папці «Views» містить усі представлення, якими користуються різні контролери, наприклад, файли макета.

## **Макет в ASP.NET Core**

## На сторінках та представленнях часто є спільні візуальні та програмні елементи. ASP.NET Core має такі можливості.

## Використання загальних макетів.

## Спільне використання директив.

## Запуск загального коду до відображення сторінок або представлень.

## Розглянемо макет підходу до ASP.NET Core MVC: контролер з представленнями

## Контролери з представленнями використовують папку Views для представлень.

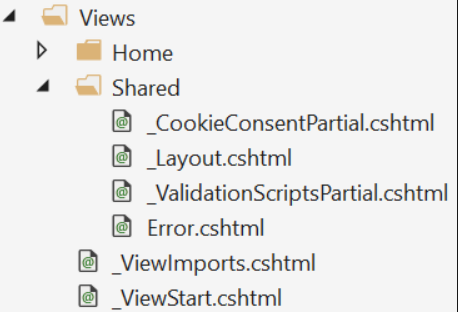
## **Що таке макет**

## Більшість веб-додатків мають спільний макет, який забезпечує узгоджений інтерфейс користувача при переході між сторінками. ***Макет, як правило, включає в себе загальні елементи інтерфейсу користувача, такі як верхній і нижній колонтитули, а також елементи навігації або меню.***

## Структури HTML, такі як скрипти та таблиці стилів, також часто використовуються кількома сторінками в додатку. Всі ці загальні елементи можуть бути визначені у файлі *макета* , на який можна посилатися будь-яким представленням, що використовується в додатку. Макети дозволяють скоротити дублювання коду у представленнях.

## Відповідно до угоди макет за замовчанням для програми ASP.NET Core має ім'я \_Layout.cshtml. У нових проектах ASP.NET Core, створених із застосуванням шаблонів, використовуються такі файли макету.

## Контролер із представленнями: Views/Shared/\_Layout.cshtml



## Макет визначає шаблон верхнього рівня для представлень у програмі. Для додатків макет не потрібний. У додатку можна визначити кілька макетів для різних уявлень.

## У наступному коді показаний файл макета для проекту, створеного за шаблоном, з контролером та уявленнями:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>@ViewData["Title"] - WebApplication1</title>

<environment include="Development">

<link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.css" />

<link rel="stylesheet" href="~/css/site.css" />

</environment>

<environment exclude="Development">

<link rel="stylesheet" href="https://ajax.aspnetcdn.com/ajax/bootstrap/3.3.7/css/bootstrap.min.css"

asp-fallback-href="~/lib/bootstrap/dist/css/bootstrap.min.css"

asp-fallback-test-class="sr-only" asp-fallback-test-property="position" asp-fallback-test-value="absolute" />

<link rel="stylesheet" href="~/css/site.min.css" asp-append-version="true" />

</environment>

</head>

<body>

<nav class="navbar navbar-inverse navbar-fixed-top">

<div class="container">

<div class="navbar-header">

<button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">

<span class="sr-only">Toggle navigation</span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<a asp-page="/Index" class="navbar-brand">WebApplication1</a>

</div>

<div class="navbar-collapse collapse">

<ul class="nav navbar-nav">

<li><a asp-page="/Index">Home</a></li>

<li><a asp-page="/About">About</a></li>

<li><a asp-page="/Contact">Contact</a></li>

</ul>

</div>

</div>

</nav>

<partial name="\_CookieConsentPartial" />

<div class="container body-content">

@RenderBody()

<hr />

<footer>

<p>&copy; 2018 - WebApplication1</p>

</footer>

</div>

<environment include="Development">

<script src="~/lib/jquery/dist/jquery.js"></script>

<script src="~/lib/bootstrap/dist/js/bootstrap.js"></script>

<script src="~/js/site.js" asp-append-version="true"></script>

</environment>

<environment exclude="Development">

<script src="https://ajax.aspnetcdn.com/ajax/jquery/jquery-3.3.1.min.js"

asp-fallback-src="~/lib/jquery/dist/jquery.min.js"

asp-fallback-test="window.jQuery"

crossorigin="anonymous"

integrity="sha384-tsQFqpEReu7ZLhBV2VZlAu7zcOV+rXbYlF2cqB8txI/8aZajjp4Bqd+V6D5IgvKT">

</script>

<script src="https://ajax.aspnetcdn.com/ajax/bootstrap/3.3.7/bootstrap.min.js"

asp-fallback-src="~/lib/bootstrap/dist/js/bootstrap.min.js"

asp-fallback-test="window.jQuery && window.jQuery.fn && window.jQuery.fn.modal"

crossorigin="anonymous"

integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa">

</script>

<script src="~/js/site.min.js" asp-append-version="true"></script>

</environment>

@RenderSection("Scripts", required: false)

</body>

</html>

**Вказівка ​​макета. Виконання коду перед кожним поданням**

Код, який повинен виконуватися перед кожним поданням або сторінкою \_ViewStart.cshtml, повинен бути поміщений у файл. За згодою \_ViewStart.cshtml файл знаходиться в папці *Pages* (або *Views* ). Оператори, перелічені у файлі \_ViewStart.cshtml, виконуються перед кожним повним уявленням (але перед макетами і перед частковими уявленнями). Як [і ViewImports.cshtml](https://learn.microsoft.com/ru-ru/aspnet/core/mvc/views/layout?view=aspnetcore-6.0#viewimports) є \_ViewStart.cshtml ієрархічним. \_ViewStart.cshtml Якщо файл визначено в папці подання або сторінки, він буде виконуватися після того, як він визначений у кореневій папці *Pages* (або *Views* ) (якщо такі є).

Приклад файлу \_ViewStart.cshtml:

Razor представлення мають Layout властивість. З його допомогою вказується макет в окремих уявленнях:

@{

Layout = "\_Layout";

}

Наведений файл наказує всім уявленням використовувати макет \_Layout.cshtml.

\_ViewStart.cshtmlі \_ViewImports.cshtml**зазвичай не** розміщуються в папці */Pages/Shared* (або */Views/Shared* ). Версії цих файлів, які повинні діяти на рівні програми, слід поміщати безпосередньо в папку */Pages* (або */Views* ).

Цей макет може використовувати повний шлях (наприклад, /Views/Shared/\_Layout.cshtml) або часткове ім'я (наприклад: \_Layout). При вказівці Razor часткового імені модуль перегляду шукає файл макета за допомогою стандартного процесу виявлення. Спочатку пошук виконується в папці, де існує метод обробника (або контролер), а потім у папці Shared. Процес виявлення аналогічний тому, що застосовується для пошуку [часткових представлень](https://learn.microsoft.com/ru-ru/aspnet/core/mvc/views/partial?view=aspnetcore-6.0#partial-view-discovery).

За замовчуванням кожен макет повинен викликати метод RenderBody. Кожний запуск RenderBody буде перетворювати вміст перегляду.

**Розділи**

Макет може за потреби посилатися на один або кілька *розділів* , викликаючи метод RenderSection. Розділи – це засіб для впорядкування розміщення певних елементів на сторінці. У кожному виклику RenderSection можна вказати, чи є розділ обов'язковим або необов'язковим:

<script type="text/javascript" src="~/scripts/global.js"></script>

@RenderSection("Scripts", required: false)

Якщо обов'язковий розділ не знайдено, створюється виняток. Окремі представлення вказують вміст для відтворення у розділі за допомогою синтаксису @sectionRazor. Якщо на сторінці або в поданні визначається розділ, він повинен бути перетворений для перегляду (інакше буде помилка).

Приклад @section визначення Razor представлення Pages:

@section Scripts {

<script type="text/javascript" src="~/scripts/main.js"></script>

}

## **Основні правила синтаксису Razor для C#**

* Блоки коду Razor вкладено в @{ ... }
* Вбудовані вирази (змінні та функції) починаються з @
* Інструкції коду закінчуються крапкою з комою
* Змінні оголошуються за допомогою ключового слова var
* Рядки взяті в лапки
* Код C# чутливий до регістру
* Файли C# мають розширення .cshtml

### Приклад C#

### <!-- Single statement block --> @{ var myMessage = "Hello World"; } <!-- Inline expression or variable --> <p>The value of myMessage is: @myMessage</p> <!-- Multi-statement block --> @{ var greeting = "Welcome to our site!"; var weekDay = DateTime.Now.DayOfWeek; var greetingMessage = greeting + " Here in Huston it is: " + weekDay; } <p>The greeting is: @greetingMessage</p>

## **Як це працює?**

Razor — це простий синтаксис програмування для вбудовування коду сервера у веб-сторінки.

Синтаксис Razor базується на структурі ASP.NET, частині Microsoft.NET Framework, яка спеціально розроблена для створення веб-додатків.

Синтаксис Razor дає вам усі можливості ASP.NET, але використовує спрощений синтаксис.

Веб-сторінки Razor можна описати як сторінки HTML із двома типами вмісту: вміст HTML і код Razor.

Коли сервер читає сторінку, він спочатку запускає код Razor, а потім надсилає HTML-сторінку в браузер. Код, який виконується на сервері, може виконувати завдання, які неможливо виконати в браузері, наприклад доступ до бази даних сервера. Серверний код може створювати динамічний вміст HTML на льоту, перш ніж він буде надісланий у браузер. З огляду на веб-переглядач, HTML-код, створений серверним кодом, нічим не відрізняється від статичного вмісту HTML.

Веб-сторінки ASP.NET із синтаксисом Razor мають спеціальне розширення файлу cshtml (Razor використовує C#) або vbhtml (Razor використовує VB).

**Створення уявлення**

Представлення, які стосуються контролера, створюються в папці Views/[ControllerName]. Представлення, які спільно використовуються контролерами, містяться в папці Views/Shared. Щоб створити представлення, додайте новий файл і надайте йому те саме ім'я, що й пов'язана з ним дія контролера з розширенням .cshtml файлу. Щоб створити уявлення, що відповідає **About** дії в контролері Home, створіть About.cshtml файл у папці Views/Home:

@{

ViewData["Title"] = "About";

}

<h2>@ViewData["Title"].</h2>

<h3>@ViewData["Message"]</h3>

<p>Use this area to provide additional information.</p>

[Razor розмітка](https://learn.microsoft.com/ru-ru/aspnet/core/mvc/views/razor?view=aspnetcore-6.0) починається з символу @. Запустіть інструкції C#, помістивши код C# у [Razor блоки коду](https://learn.microsoft.com/ru-ru/aspnet/core/mvc/views/razor?view=aspnetcore-6.0" \l "razor-code-blocks) , задані фігурними дужками ( { ... }). Прикладом може бути наведений вище оператор присвоєння значення "About" властивості ViewData["Title"]. Для відображення значень у коді HTML можна просто посилатися на них за допомогою символу @. Див. вміст <h2>і <h3>вище.

Наведений вище вміст — це лише частина веб-сторінки, що відображається для користувача. Решта макета сторінки та інші стандартні аспекти представлення визначаються інших файлах представлення.

**Вказівка представлень у контролерах**

Подання зазвичай повертаються з дій у вигляді ViewResult ActionResult типу. Метод дії може створювати та повертати об'єкт ViewResult безпосередньо, проте зазвичай так не робиться. Так як більшість контролерів успадковуються, Controller для повернення допоміжного View методу просто використовується допоміжний ViewResult метод:

HomeController.cs:

public IActionResult About()

{

ViewData["Message"] = "Your application description page.";

return View();

}