

Dr. Öğr. Üyesi Fatma AKALIN

VIEW

VIEW KULLANIMI

View, MVC'de projede kullanıcının gördüğü ve kullandığı arayüzüdür. Bu bölümde projenin kullanıcılarla sunulacak olan html dosyaları yer almaktadır. Sizler tarafından yapılan projenin kodları sunucuya aktarıldıkten sonra kullanıcının internet tarayıcısı üzerinden siteyi açtığında karşısına gelen sayfanın kodlamasıdır

Kullanıcılarından alınacak olan istekler Viewlar aracılığı ile ilgili Controller'a ilettilir. Farklı bir deyişle kullanıcılar sadece viewları gördükleri için yapacakları tüm istekler yine bu viewlar aracılığıyla yapılır. Ardından bu istekler sunucu tarafından alınıp ilgili controllera verilir

Örneğin <https://www.hepsiburada.com/ara?q=android> sitesinde anasayfada yer alan bir ürünün linkine tıkladığımız zaman isteği anasayfa View'ından yapmış oluruz. Ardından istek sunucuya gider. Sunucuda bu istekle hangi controller ilgileniyor ise ona yönlendirme sağlanır.

Dr. Öğr.

Şimdi sıfırdan yeni bir proje oluşturalım.

Yeni projenizi yapılandırin

ASP.NET Web Uygulaması (.NET Framework)

C#

Windows

Bulut

Web

Proje adı

denemeikinciders

Konum

D:\aspnet\1_ADERSLER



Çözüm adı

denemeikinciders

Çözümü ve projeyi aynı dizine yerleştirin



Altyapı

.NET Framework 4.6.2

Proje "D:\aspnet\1_ADERSLER\denemeikinciders\denemeikinciders\" içinde oluşturulacak

Yeni ASP.NET Web Uygulaması oluştur



Boş
ASP.NET uygulamaları oluşturmak için boş bir proje şablonu. Bu şablonda içerik yoktur.

Web Forms
ASP.NET Web Forms uygulamaları oluşturmak için bir proje şablonu. ASP.NET Web Forms, bilinen sürükle ve bırak yöntemiyle olay denetimli modeli kullanarak dinamik web siteleri oluşturmanızı sağlar. Tasarım yüzeyi ile yüzlerce denetim ve bileşen, veri erişimi olan gelişmiş, güçlü, kullanıcı arabirimini denetimli web sitelerini hızlı bir şekilde oluşturmanızı sağlar.

MVC
ASP.NET MVC uygulamaları oluşturmaya yarayan bir proje şablonu. ASP.NET MVC, Model-View-Controller mimarisini kullanarak uygulamalar oluşturmanıza olanak sağlar. ASP.NET MVC en son standartları kullanan uygulamalar oluşturmak için hızlı, test güdümlü geliştirmeye olanak sağlayan birçok özellik içerir.

Web API
Tarayıcılar ve mobil cihazlar dahil olmak üzere çok çeşitli istemcilere erişebilen RESTful HTTP hizmetleri oluşturmaya yarayan bir proje şablonudur.

Tek Sayfali Uygulama
ASP.NET Web API kullanarak JavaScript temelli zengin istemci tarafı HTML5 uygulamaları oluşturmak üzere kullanılan proje şablonudur. Tek Sayfalı Uygulamalar, HTML5, CSS3 ve JavaScript'in kullanıldığı istemci tarafı etkileşimler içeren zengin bir kullanıcı deneyimi sağlar.

Kimlik Doğrulama

Yok

Klasörleri ve çekirdek başvurularını ekle

Web Forms
 MVC
 Web API'si

Gelişmiş

HTTPS'yi Yapılandır
(Docker Desktop gerektiriyor)
 Docker desteği
(Docker Desktop gerektiriyor)
 Aynca birim testleri için bir proje oluştur
`denemeikinciders.Tests`

Geri Oluştur

Boş bir MVC yapısı seçmediğimiz için karşımıza çıkan ekran aşağıdaki gibidir.

The screenshot shows a Microsoft Edge browser window with the following details:

- Title Bar:** Home Page - ASP.NET Uygulama
- Address Bar:** localhost:44307
- Toolbar:** Includes standard browser icons for back, forward, search, and refresh, along with a F12 developer tools icon.
- Top Navigation:** Shows pinned sites like PAMUKKALE ÜNİVE..., Gmail, YouTube, and Tumor Detection U... alongside a search bar and a 'Tüm Yer İşaretleri' button.
- Header:** A dark header bar with links for 'Uygulama adı', 'Giriş', 'Hakkında', and 'Kişi'.
- Main Content:**
 - # ASP.NET
 - A brief introduction: "ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript."
 - [Learn more »](#)
 - ## Getting started

ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that enables a clean separation of concerns and gives you full control over markup for enjoyable, agile development.

[Learn more »](#)
 - ## Get more libraries

NuGet is a free Visual Studio extension that makes it easy to add, remove, and update libraries and tools in Visual Studio projects.

[Learn more »](#)
 - ## Web Hosting

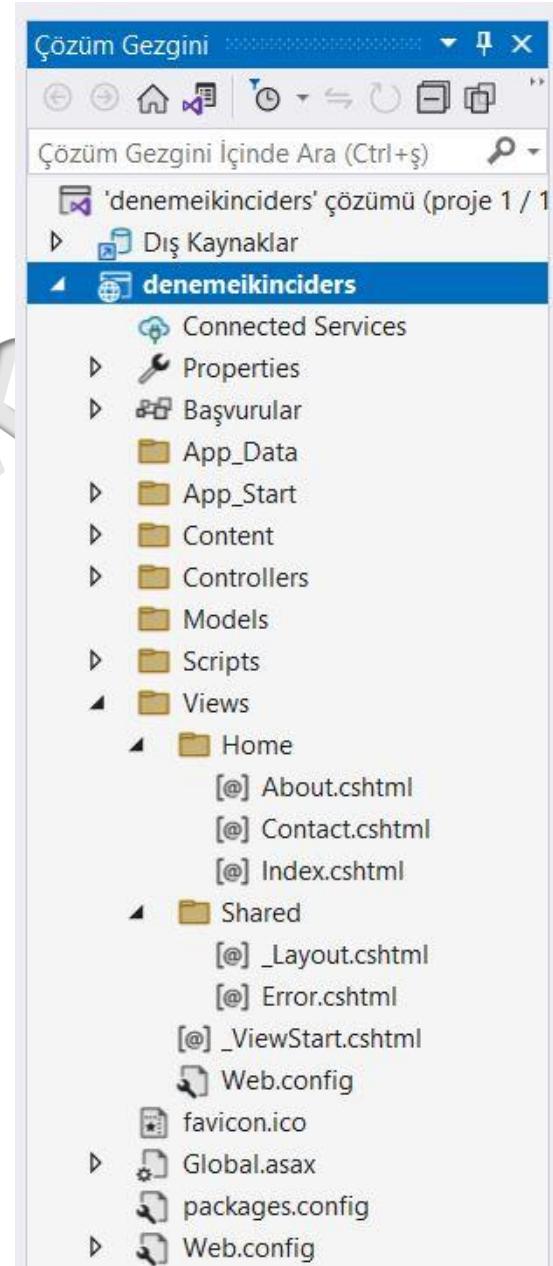
You can easily find a web hosting company that offers the right mix of features and price for your applications.

[Learn more »](#)
- Footer:** © 2023 - ASP.NET Uygulamam

DİZİN YAPISI

Dizin yapısına baktığımızda kullanıcının gördüğü ekran olarak nitelendirdiğimiz Viewlar, projede Views klasörünün altında yer almaktadır.

Oluşturduğumuz projenin dizin yapısı sağda verilmiştir.



Yeni bir proje oluşturduğunuzda View dizininin altında Home ve shared dizinleri bulunur. Bu dizinler içerisinde cshtml uzantılı viewlar bulunur. Tüm viewların uzantısı aynıdır fakat farklı yapıda olabilirler.

Bu doğrultuda View dizininin içerisinde yer alan alt dizinlerdeki viewların dizin yapısı ile kullanım yöntemleri arasındaki ilişkiyi inceleyelim

1- Home : HomeController içerisindeki Actionlara ait viewların yer aldığı bir dizindir. Her action için bir adet View oluşturulmuştur.

2- Shared : bu klasör tüm viewlar tarafından ortak kullanılan view dosyalarıdır.

2.1. _Layout.cshtml: Projenin temel şablonudur. Diğer viewlar bu şablon içerisinde görüntülenebilir.

2.2. _LoginPartial.cshtml: Kullanıcı giriş işleminin yapıldığı viewdir.

2.3. Error.cshtml: Projenin hata sayfasıdır. Bir hata ile karşılaşıldığında bu sayfaya yönlendirme sağlanır.

CONTROLLER – VIEW İLİŞKİSİ

Controllerlar, action result türünde bir veri döndürür ve geri döndürdüğü değer View metodudur. View metodu, HomeController'ın kalıtımını aldığı Controller sınıfı içerisinde yer almaktadır.

İlk derslerde Empty olarak nitelendirilen bir proje açmıştık. İçerisinde Controller ya da view yoktu. BU sayfayı tarayıcıda görüntülemek istemiştiğimizde hata almıştır. Bu nedenle ilk olarak controller ve ardından view tasarlayarak bu sorunu çözmüştük.

LAYOUT(ŞABLON SAYFA) KULLANIMI

ASP.NET MVC ile proje geliştirirken tasarımsal olarak aynı bölümlerin birçok sayfada kullanılması gerekebilir. Web sitelerimizin sabit ve değişken kısımları vardır. Sabit olan sayfalara layout ismini veriyoruz. Layout sayesinde tüm view'larda sabit olan bölümleri bir kere oluşturup değişken yerleri View bazında düzenleyebiliriz. Sabit yerleri layout üzerinden tek bir noktadan değiştirdiğinizde bu değişim çalışma anında tüm sayfalarda da otomatik olarak sağlanacaktır

Layout şablonunu anlatmak üzere yeni bir proje oluşturalım..

Yeni ASP.NET Web Uygulaması oluştur

Bos
ASP.NET uygulamaları oluşturmak için boş bir proje şablonu. Bu şablonda içerik yoktur.

Web Forms
ASP.NET Web Forms uygulamaları oluşturmak için bir proje şablonu. ASP.NET Web Forms, bilinen sürükle ve bırak yöntemiyle olay denetimli modeli kullanarak dinamik web siteleri oluşturmanızı sağlar. Tasarım yüzeyi ile yüzlerce denetim ve bileşen, veri erişimi olan gelişmiş, güçlü, kullanıcı arabirimini denetimli web sitelerini hızlı bir şekilde oluşturmanızı sağlar.

MVC
ASP.NET MVC uygulamaları oluşturmaya yarayan bir proje şablonu. ASP.NET MVC, Model-View-Controller mimarisini kullanarak uygulamalar oluşturmanıza olanak sağlar. ASP.NET MVC en son standartları kullanan uygulamalar oluşturmak için hızlı, test güdümlü geliştirmeye olanak sağlayan birçok özellik içerir.

Web API
Tarayıcılar ve mobil cihazlar dahil olmak üzere çok çeşitli istemcilere erişebilen RESTful HTTP hizmetleri oluşturmaya yarayan bir proje şablonudur.

Tek Sayfalı Uygulama
ASP.NET Web API kullanarak JavaScript temelli zengin istemci tarafı HTML5 uygulamaları oluşturmak üzere kullanılan proje şablonudur. Tek Sayfalık Uygulamalar, HTML5, CSS3 ve JavaScript'in kullanıldığı istemci tarafı etkileşimler içeren zengin bir kullanıcı deneyimi sağlar.

Kimlik Doğrulama

Yok

Klasörleri ve çekirdek başvurularını ekle

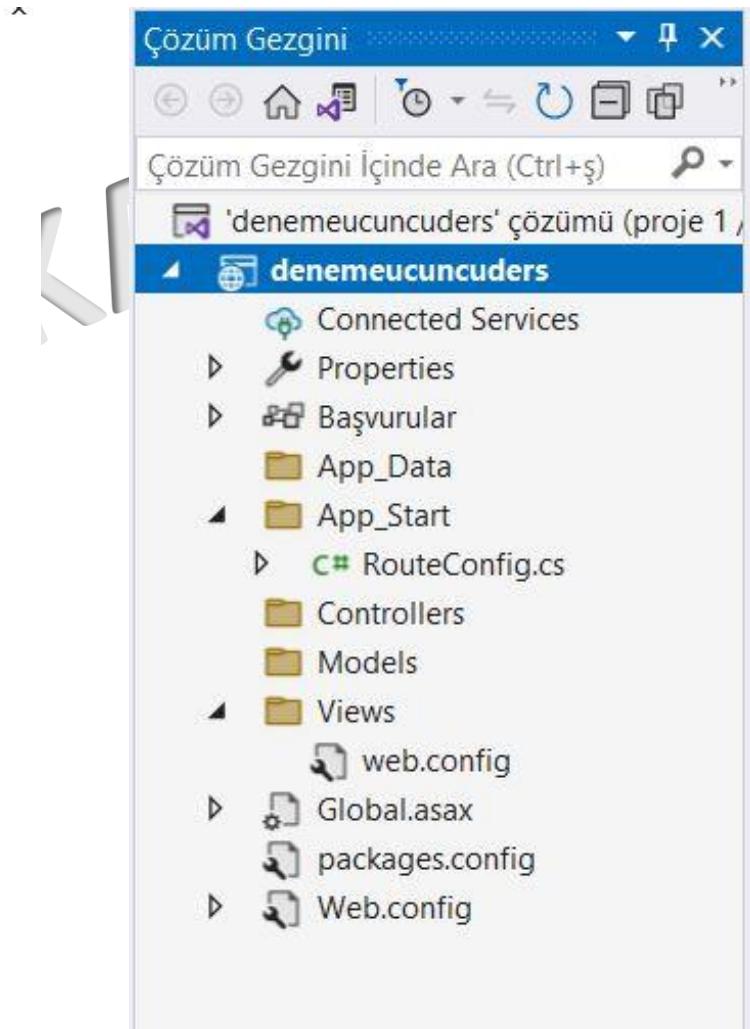
- Web Forms
- MVC
- Web API'si

Gelişmiş

- HTTPS'yi Yapılandır
- Docker desteği
(Docker Desktop gerektiriyor)
- Ayrıca birim testleri için bir proje oluştur
`denemeucuncuders.Tests`

Geri

Oluştur



HomeController isminde bir denetleyici oluşturalım.

Yeni İskeleli Öğe Ekle

The screenshot shows the 'Add New Item' dialog in Visual Studio. On the left, a navigation tree lists 'Yükü' (Installed), 'Ortak' (Common), and 'MVC'. Under 'MVC', 'Denetleyici' (Controller) is selected and highlighted with a blue border. The main area displays three template options:

- MVC 5 Denetleyici - Boş** (Empty Controller): Selected, highlighted with a blue background. Description: 'Entity Framework kullanan, görünümleri olan MVC 5 Denetleyici'. Details: 'MVC 5 Denetleyici - Boş' sunun tarafından Microsoft v5.0.0.0. Kimlik: MvcControllerEmptyScaffolder'. Below this, there is a 'Denetleyici Ekle' (Add Controller) button.
- Entity Framework kullanan, görünümleri olan MVC 5 Denetleyici**: Description: 'Okuma/yazma eylemleri olan MVC 5 Denetleyici'.
- Okuma/yazma eylemleri olan MVC 5 Denetleyici**: Description: 'Entity Framework kullanan, görünümleri olan MVC 5 Denetleyici'.

On the right side of the dialog, there is a large watermark-like text 'IKALIN' diagonally across the screen. At the bottom of the dialog, there are two buttons: 'Ekle' (Add) and 'İptal' (Cancel). The 'Denetleyici adı:' (Controller name:) input field contains the value 'HomeController'.

Controllerlarımızı oluşturalım.

Dr. Öğr. Fatih

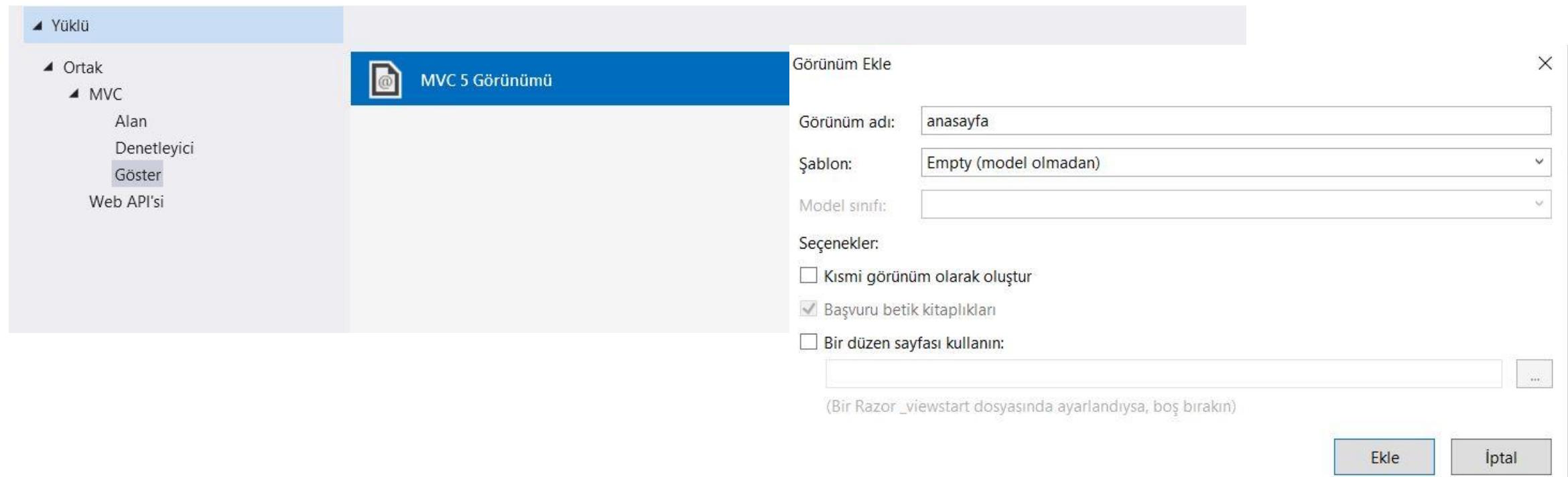
```
namespace denemeucuncuders.Controllers
{
    0 başvuru
    public class HomeController : Controller
    {
        // GET: Home
        0 başvuru
        public ActionResult anasayfa()
        {
            return View();
        }
        0 başvuru
        public ActionResult hakkimizda()
        {
            return View();
        }
        0 başvuru
        public ActionResult iletisim()
        {
            return View();
        }
        0 başvuru
        public ActionResult icicelayoutkullanimi()
        {
            return View();
        }
    }
}
```

Anasayfa ve İletişim controllerlarımız için Viewlarımıza yandaki gibi oluşturalım..

```
namespace denemeucuncuders.Controllers
{
    public class HomeController : Controller
    {
        // GET: Home
        public ActionResult Anasayfa()
        {
            return View();
        }
    }
}
```

The context menu for the 'ActionResult Anasayfa()' line shows the option 'Görünüm Ekle...' highlighted.

Yeni İskeleli Öğe Ekle



Anasayfa ve iletişim sayfalarının içerişini dolduralım aşağıdaki gibi...

```
iletisim.cshtml  anasayfa.cshtml  HomeController.cs
```

```
1
2  @{
3      Layout = null;
4  }
5  <!DOCTYPE html>
6  <html>
7      <head>
8          <meta name="viewport" content="width=device-width" />
9          <title>anasayfa</title>
10     </head>
11     <body>
12         <div>
13             <h2>anasayfa</h2>
14             <p>
15                 Fatma AKALIN Fatma AKALIN Fatma AKALIN Fatma AKALIN
16                 Fatma AKALIN Fatma AKALIN Fatma AKALIN Fatma AKALIN
17                 Fatma AKALIN Fatma AKALIN Fatma AKALIN Fatma AKALIN
18                 Fatma AKALIN Fatma AKALIN Fatma AKALIN Fatma AKALIN
19                 Fatma AKALIN Fatma AKALIN Fatma AKALIN Fatma AKALIN
20                 Fatma AKALIN Fatma AKALIN Fatma AKALIN Fatma AKALIN
21                 Fatma AKALIN Fatma AKALIN Fatma AKALIN Fatma AKALIN
22                 Fatma AKALIN Fatma AKALIN Fatma AKALIN Fatma AKALIN
23                 Fatma AKALIN Fatma AKALIN Fatma AKALIN Fatma AKALIN
24                 Fatma AKALIN Fatma AKALIN Fatma AKALIN Fatma AKALIN
25                 Fatma AKALIN Fatma AKALIN Fatma AKALIN Fatma AKALIN
26                 Fatma AKALIN Fatma AKALIN Fatma AKALIN Fatma AKALIN
27             </p>
28         </div>
29     </body>
30 </html>
```

```
iletisim.cshtml  anasayfa.cshtml  HomeController.cs
```

```
1
2  @{
3      Layout = null;
4  }
5
6  <!DOCTYPE html>
7
8      <html>
9          <head>
10             <meta name="viewport" content="width=device-width" />
11             <title>iletisim</title>
12         </head>
13         <body>
14             <div>
15             </div>
16         </body>
17     </html>
18
```

Şimdi layout aşamasına geçelim.

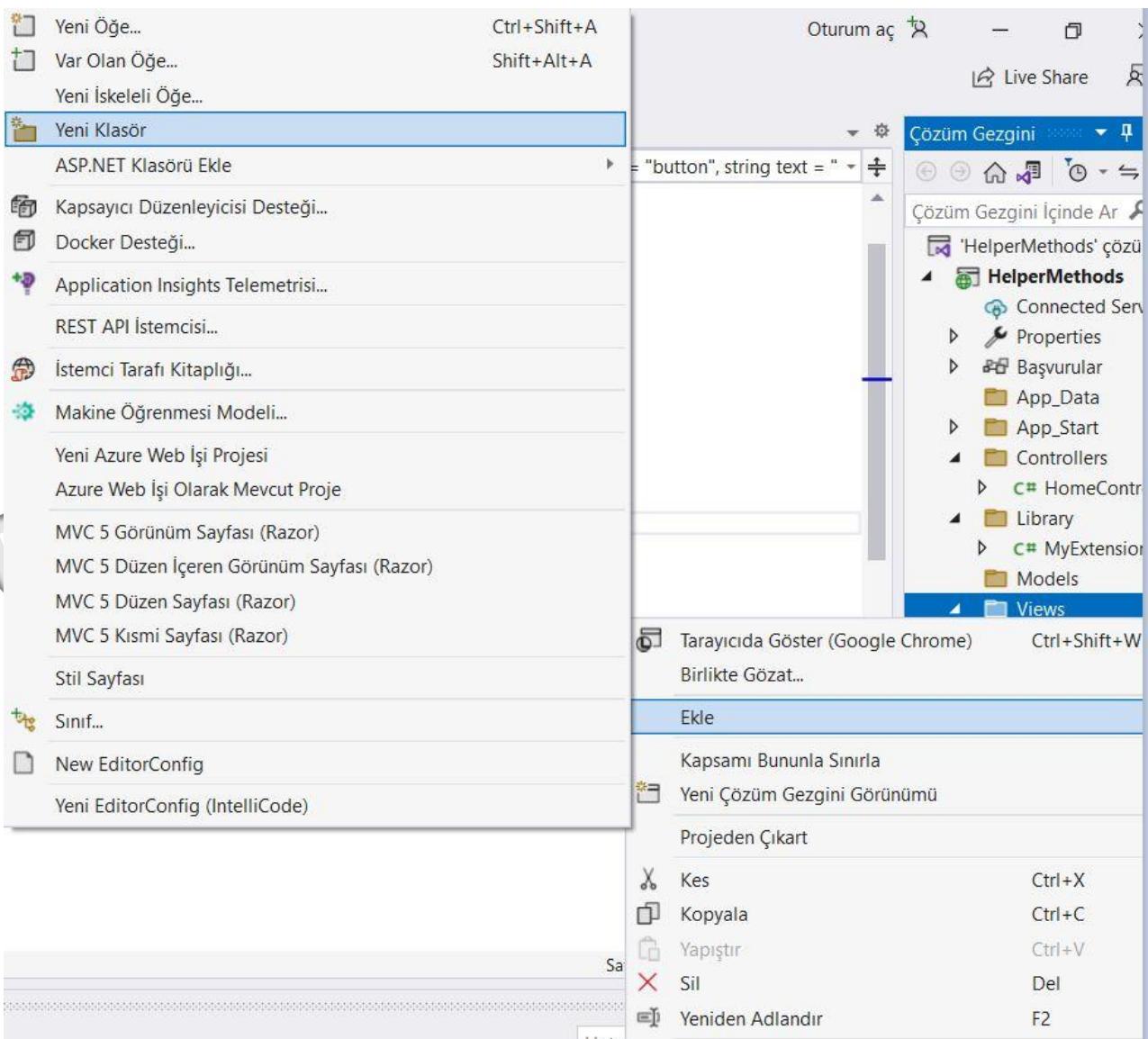
Örneğimizde, Home Controller oluşturduğum ve içerisinde 3 ActionResult tipinde metot yazdım. İletişim ve anasayfa kısımlarında herhangi bir layout kullanmaksızın bu iki sayfamı oluşturduğum. Fakat diğer iki metotta layout kullanmaya karar verdim. Layoutu şu şekilde açıklayabilirim.

Web sitelerimizin sabit ve değişken kısımları vardır. Layout kullanacaksak öncelikle sabit olan kısımlarımızı tutan bir HTML sayfa oluşturmamız gerekiyor. Yani Bu sabit olan sayfaları tutan kısımlara layout diyoruz. Layoutlar view klasörünün altında genelde shared isimli paylaşılmış anlamına gelen bir klasörde tutulur.

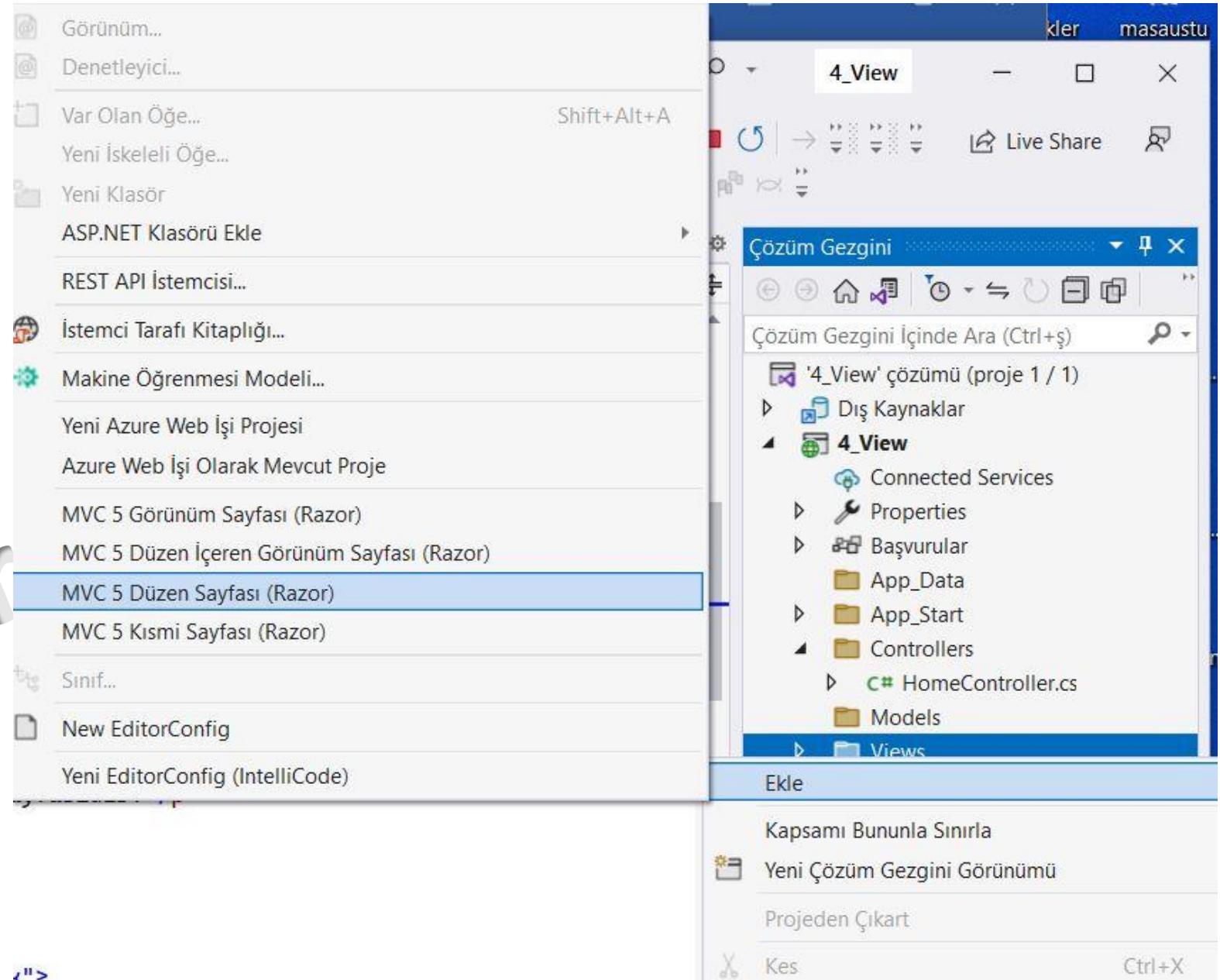
LAYOUT OLUŞTURMA

Öncelikle klasör
oluşturuyoruz

Dr. Öğr. Fatih

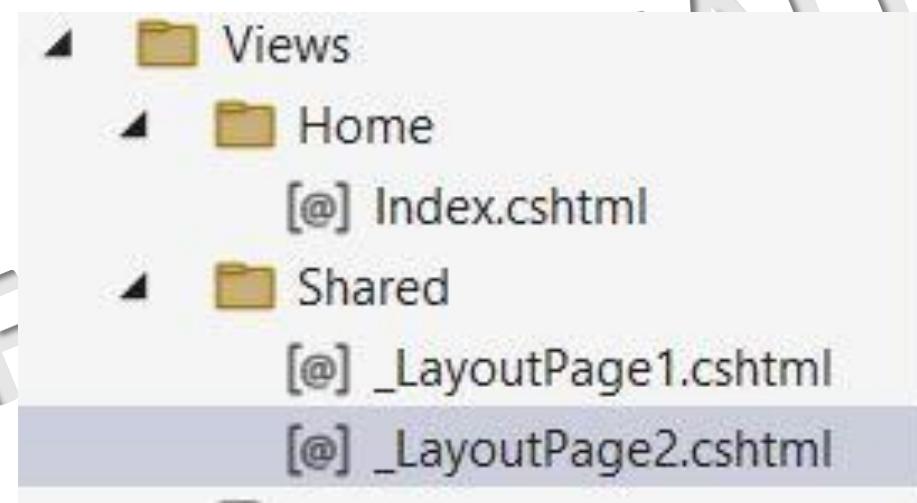


Bu klasörün
içerisine bir tane
layout page
ekleyeceğim.
Ekle ve ardından
MVC Layout
Page(Razor)
kismından bir
eklemeye
yapabilirsin.

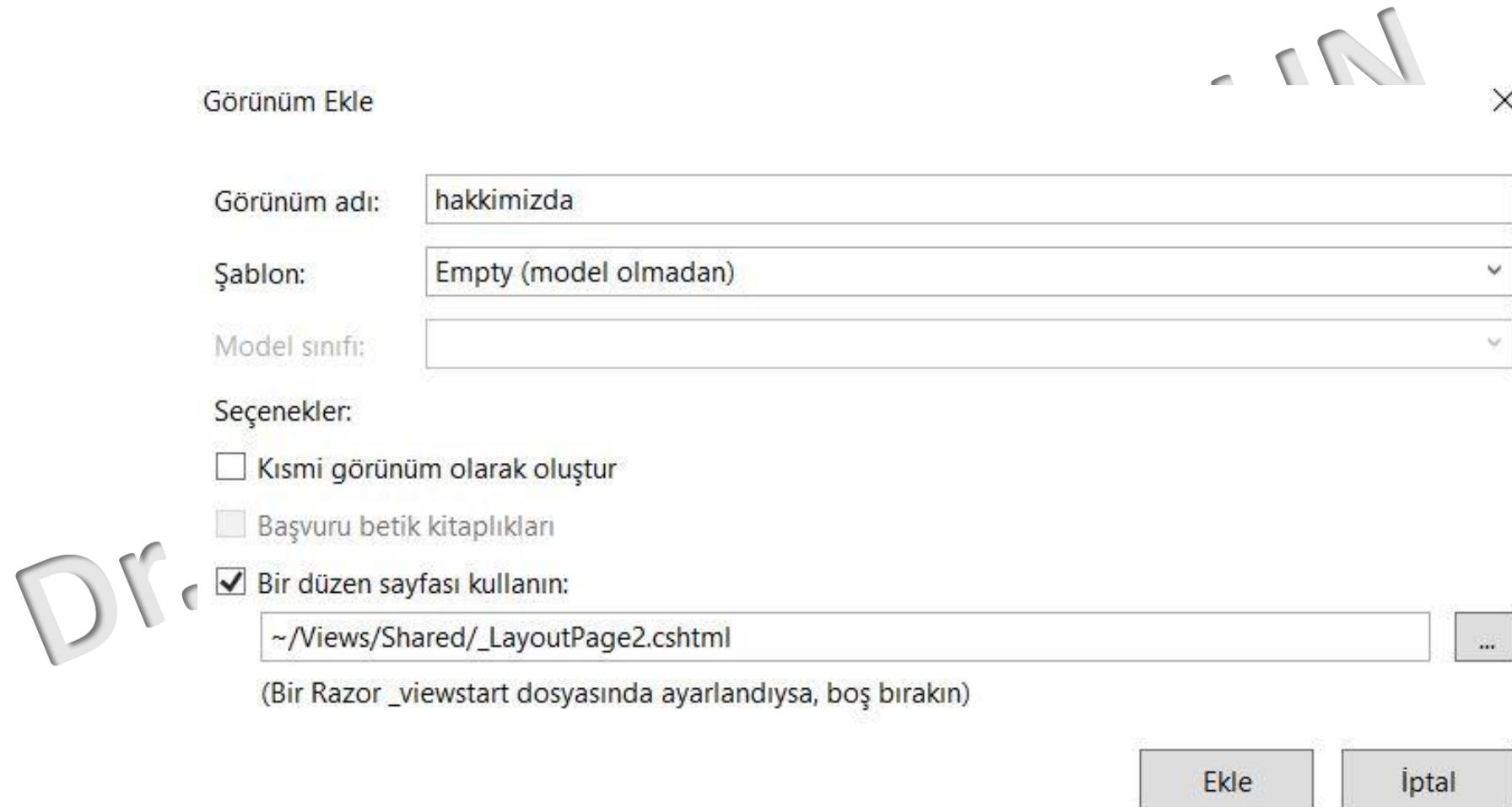


Layout 1 ve
Layout 2
sayfalarının
iceriğini web
teknolojileri
dersinde
aldığımız temel
ile inşa ediyoruz.

KLASÖR YAPISI



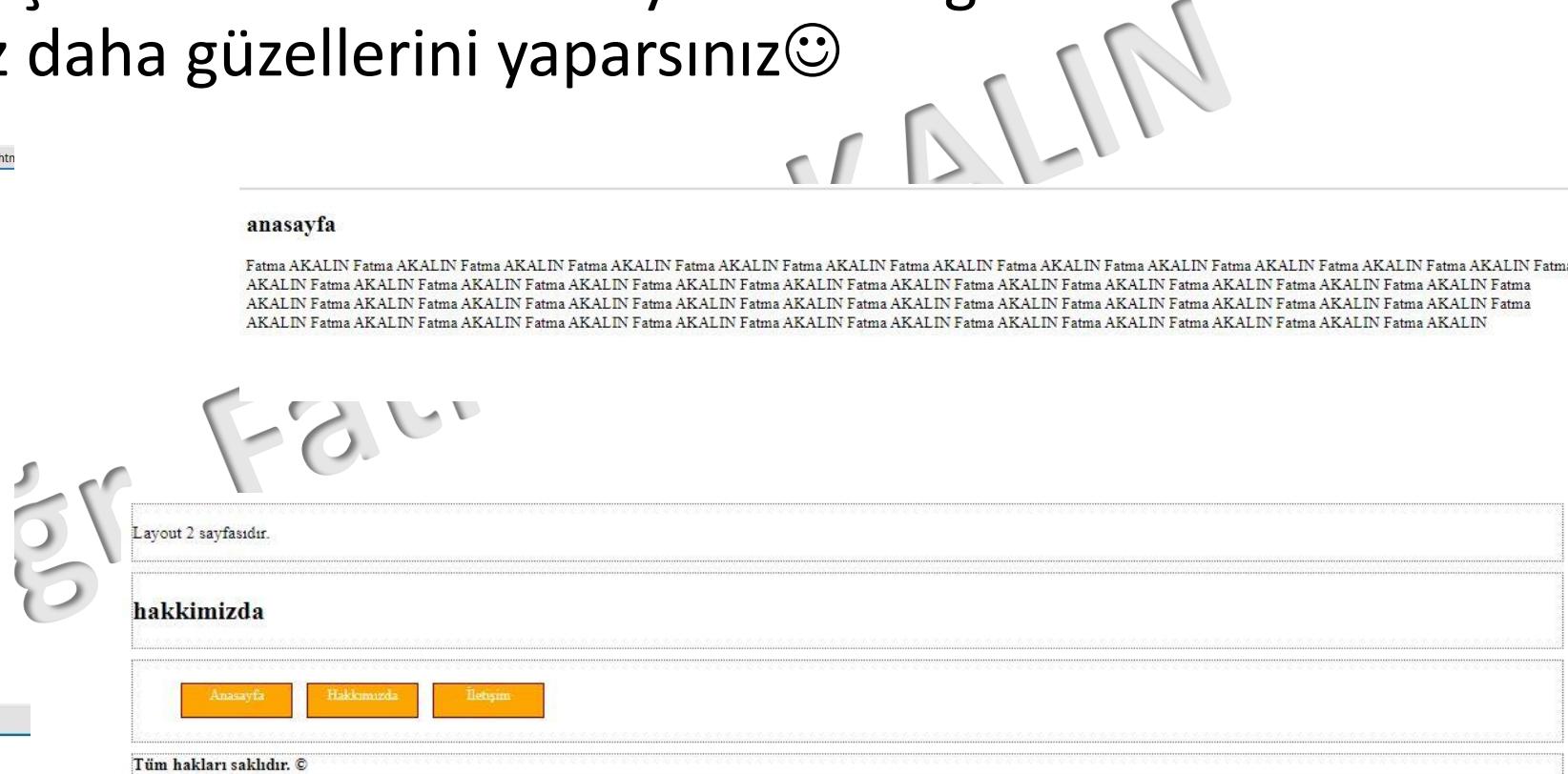
Şimdi diğer hakkımızda View'ı oluştururken Layout2 sayfasını dikkate alacağız.



Şimdi Layout kullanmadığımız anasayfa ile Layout kullandığımız Hakkımızda sayfalarının çıktılarına bakalım. Layout daha güzel olabilirdi tabiki.. Eğitim amaçlı siz daha güzellerini yaparsınız😊

```
1  @{
2      Layout = null;
3  }
4  <!DOCTYPE html>
5  <html>
6      <head>
7          <meta name="viewport" content="width=device-width" />
8          <title>anasayfa</title>
9      </head>
10     <body>
11         <div>
12             <h2>anasayfa</h2>
13             <p>
14                 Fatma AKALIN Fatma AKALIN Fatma AKALIN Fatma AKALIN
15                 Fatma AKALIN Fatma AKALIN Fatma AKALIN Fatma AKALIN
16                 Fatma AKALIN Fatma AKALIN Fatma AKALIN Fatma AKALIN
17                 Fatma AKALIN Fatma AKALIN Fatma AKALIN Fatma AKALIN
18                 Fatma AKALIN Fatma AKALIN Fatma AKALIN Fatma AKALIN
19                 Fatma AKALIN Fatma AKALIN Fatma AKALIN Fatma AKALIN
20                 Fatma AKALIN Fatma AKALIN Fatma AKALIN Fatma AKALIN
21                 Fatma AKALIN Fatma AKALIN Fatma AKALIN Fatma AKALIN
22                 Fatma AKALIN Fatma AKALIN Fatma AKALIN Fatma AKALIN
23                 Fatma AKALIN Fatma AKALIN Fatma AKALIN Fatma AKALIN
24                 Fatma AKALIN Fatma AKALIN Fatma AKALIN Fatma AKALIN
25                 Fatma AKALIN Fatma AKALIN Fatma AKALIN Fatma AKALIN
26                 Fatma AKALIN Fatma AKALIN Fatma AKALIN Fatma AKALIN
27             </p>
28         </div>
29     </body>
30 </html>
```

```
hakkimizda.cshtml + X _LayoutPage2.cshtml _LayoutPage1.cshtml
1
2     @{
3         ViewBag.Title = "hakkimizda";
4         Layout = "~/Views/Shared/_LayoutPage2.cshtml"
5     }
6
7     <h2>hakkimizda</h2>
8
9
```



Layout sayfalarını incelersek, tüm sayfaar için değişken olan bölümler için tanımlamayı C#'ın RenderBody() metodu ile yapıyoruz.

```
</head>

<body>
    <div id="menu" class="block">
        @*Header*@
        <ul>
            <li>@Html.ActionLink("Anasayfa", "anasayfa", "Home")</li>
            <li>@Html.ActionLink("Hakkımızda ", "hakkimizda", "Home")</li>
            <li>@Html.ActionLink("İletişim", "iletisim", "Home")</li>
            <li>@Html.ActionLink("Nesned Layout", "icicelayoutkullanimi", "Home")</li>
        </ul>
    </div>
    <div class="block">
        @RenderBody()
    </div>
    <div class="block">
        @*Footer*@
        <b>Tüm hakları saklıdır. ©</b>
    </div>
</body>
</html>
```

Bu şekilde aynı layoutu istediğiniz sayfalarınıza uygulayabilirsiniz.

View dosyanızın başındaki

`Layout = "~/views/Shared/_LayoutPage2.cshtml";`
ifadesi istediğiniz sayfalara uygulama işlevsellliğini sağlıyor.

Bu projede Controllerlerda oluşturduğum İçicelayoutkullanımı isimli action metodunu kullanmadım. Ek olarak _layoutPage1.cshtml 'ide kullanmadım.AKLIMIZI KARŞITIRMASIN!!!

İÇ İÇE(NESTED) LAYOUT OLUŞTURMA

ASP.NET MVC ile yazılım geliştirme sırasında iç içe layout oluşturma ihtiyacı oluşabilir. Peki iç içe layout nedir?

Layoutların birbirinden türeme durumunu ifade eden bir işlevselliktir.

Bir örnek üzerinden açıklayalım...

Yeni bir proje oluşturun

Son proje şablonları

- ASP.NET Web Uygulaması (.NET Framework) C#
- Windows Forms Uygulaması (.NET Framework) C#
- ASP.NET Core Web Uygulaması (Model-Görünüm-Denetleyici) C#
- Konsol Uygulaması C++

Yeni projenizi yapılandırın

ASP.NET Web Uygulaması (.NET Framework) C# Windows Bulut Web

Proje adı:

Konum:

Çözüm adı:

Çözümü ve projeyi aynı dizine yerleştirin

Alt yapı:

Proje "D:\aspnet\1_ADERSLER\deneme_dorduncuders\deneme_dorduncuders\" içinde oluşturulacak

Şablon ara (Alt+S)

Tüm diller

Tüm platformlar

Tüm proje türleri

C# Konsol Uygulaması
Windows, Linux ve macOS'de .NET üzerinde çalışabilen bir komut satırı uygulaması oluşturma projesi

C# ASP.NET Core Web Uygulaması
ASP.NET Core Razor Sayfalar içeriği örneğiyle ASP.NET Core uygulaması oluşturmak için proje şablonu

Blazor Server Uygulaması

Yeni ASP.NET Web Uygulaması oluştur

Boş
ASP.NET uygulamaları oluşturmak için boş bir proje şablonu. Bu şablonda içerik yoktur.

Web Forms
ASP.NET Web Forms uygulamaları oluşturmak için bir proje şablonu. ASP.NET Web Forms, bilinen sürükle ve bırak yöntemiyle denetimli modeli kullanarak dinamik web siteleri oluşturmanızı sağlar. Tasarım yüzeyi ile yüzlerce denetim ve bileşen, veri erişimi olan gelişmiş, güçlü, kullanıcı arabirimini denetimli web sitelerini hızlı bir şekilde oluşturmanızı sağlar.

MVC
ASP.NET MVC uygulamaları oluşturma yarayan bir proje şablonu. ASP.NET MVC, Model-View-Controller mimarisini kullanarak uygulamalar oluşturmanızı olanak sağlar. ASP.NET MVC en son standartları kullanan uygulamalar oluşturmak için hızlı, test güdümlü geliştirmeye olanak sağlayan bir çok özellik içerir.

Web API
Tarayıcılar ve mobil cihazlar dahil olmak üzere çok çeşitli istemcilere erişebilen RESTful HTTP hizmetleri oluşturuma yarayan bir proje şablonudur.

Tek Sayfalı Uygulama
ASP.NET Web API kullanarak JavaScript temelli zengin istemci tarafı HTML5 uygulamaları oluşturmak üzere kullanılan proje şablonudur. Tek Sayfalı Uygulamalar, HTML5, CSS3 ve JavaScript'in kullanıldığı istemci tarafı etkileşimler içeren zengin bir kullanıcı deneyimi sağlar.

Kimlik Doğrulama
Yok

Klasörleri ve çekirdek başvurularını ekle

Web Forms
 MVC
 Web API'si

Gelişmiş

HTTPS'yi Yapılandır
 Docker desteği
(Docker Desktop gereklidir)
 Aynca birim testleri için bir proje oluştur
deneme_dorduncuders.Tests

The screenshot shows the context menu of a folder named "App_Start" in the Solution Explorer. The "Ekle" (Add) option is selected, which has opened a new window titled "Yeni İşkeleli Öğe Ekle" (Add New Item).

Context Menu (Top Level):

- Yeni Öğe... (Ctrl+Shift+A)
- Var Olan Öğe... (Shift+Alt+A)
- Yeni İşkeleli Öğe...
- Yeni Klasör
- ASP.NET Klasörü Ekle
- Kapsayıcı Düzenleyicisi Desteği...
- Docker Desteği...
- Application Insights Telemetrisi...
- REST API İstemcisi...
- İstemci Tarafı Kitaplığı...
- Makine Öğrenmesi Modeli...
- Yeni Azure Web İş Projesi
- Azure Web İş Olarak Mevcut Proje
- Web API Denetleyicisi Sınıfı (v2.1)
- MVC 5 Düzen Sayfası (Razor)
- MVC 5 Kısmı Sayfası (Razor)
- Stil Sayfası
- MVC 5 Görünüm Sayfası (Razor)
- MVC 5 Düzen İceren Görünüm Sayfası (Razor)
- Sınıf...
- New EditorConfig
- Yeni EditorConfig (IntelliCode)

Context Menu (Submenu under Ekle):

- Tarayıcıda Göster (Google Chrome) (Ctrl+Shift+W)
- Birlikte Gözat...
- Ekle
- Kapsamı Bununla Sınırla
- Yeni Çözüm Gezgini Görünümü
- Projeden Çıkart
- Kes (Ctrl+X)
- Kopyala (Ctrl+C)
- Yapıştır (Ctrl+V)
- Sil (Del)
- Yeniden Adlandır (F2)
- Tam Yolu Kopyala
- Klasörü Dosya Gezgini'nde Aç
- Terminalde Aç
- Özellikler (Alt+Enter)

Yeni İşkeleli Öğe Ekle Window:

- Yüküldü:** MVC
- Ortak:** Alan, Denetleyici, Göster, Web API'si
- Entity Framework kullanan, görünümleri olan MVC 5 Denetleyici**
- MVC 5 Denetleyici - Boş** (Selected)
- Okuma/yazma eylemleri olan MVC 5 Denetleyici**

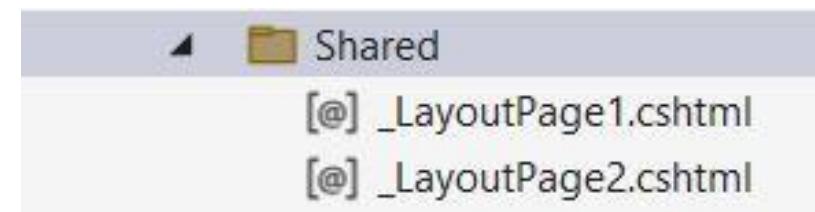
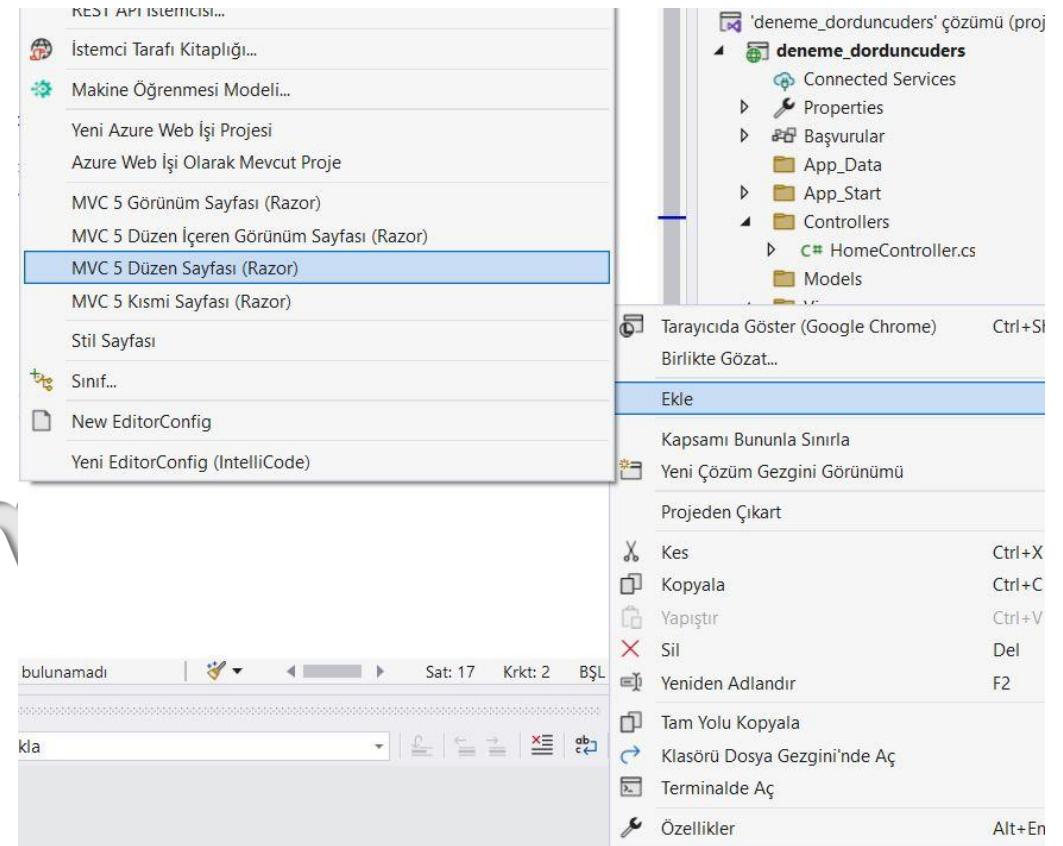
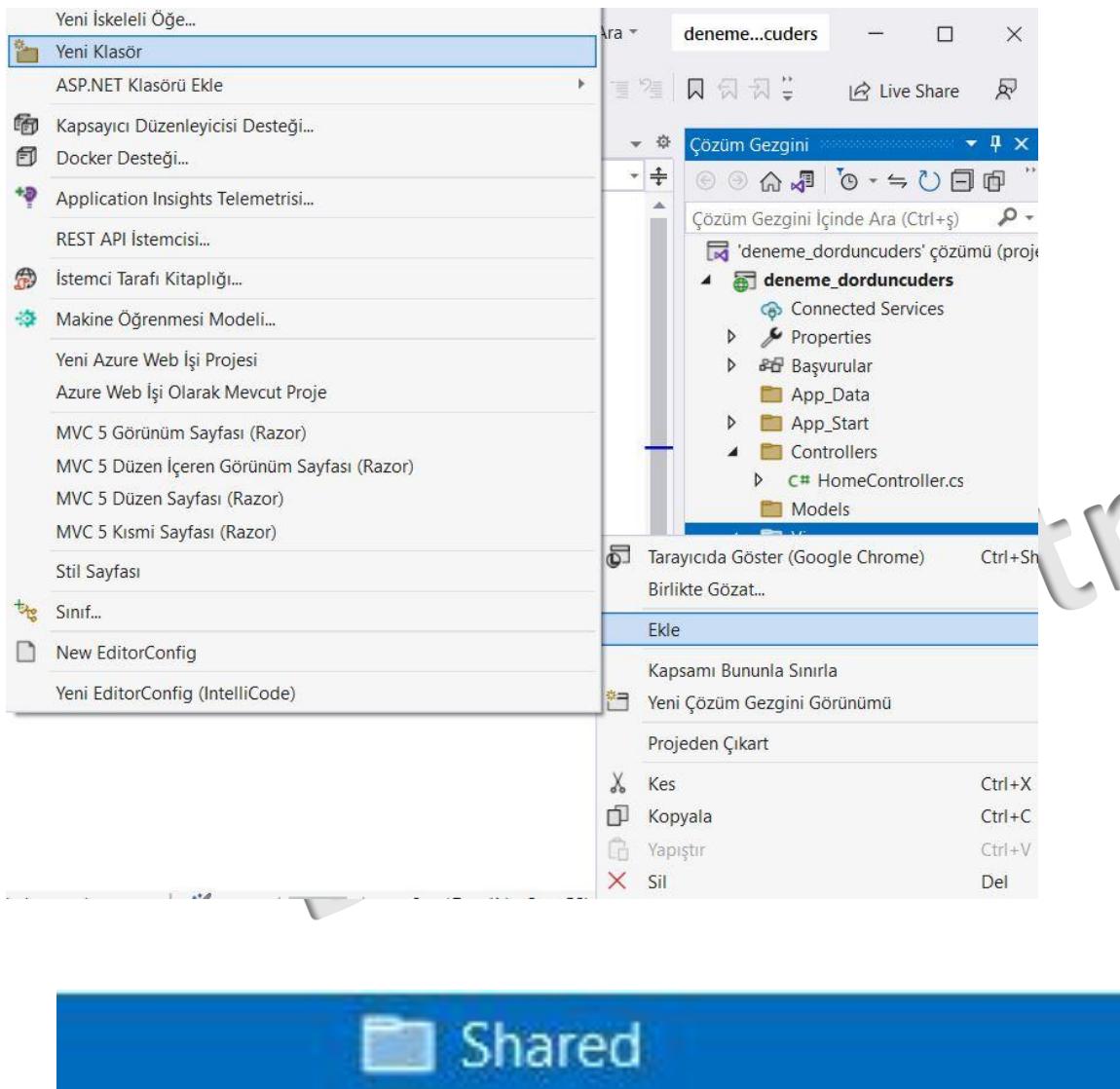
Bottom Dialog: Denetleyici Ekle

Denetleyici adı:

Buttons: Ekle (Add), İptal (Cancel)

Right Panel (Details):

MVC 5 Denetleyici - Boş
şunun tarafından Microsoft v5.0.0.
Boş bir MVC denetleyici.
Kimlik: MvcControllerEmptyScaffolder



Controller yapımızın içinde oluşturduğumuz 2 metoda view ekleyelim

```
ml      _LayoutPage1.cshtml      HomeController.cs  ✎ X
incuders
deneme_dorduncuder
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace deneme_dorduncuders.Controllers
{
    public class HomeController : Controller
    {
        // GET: Home
        public ActionResult Index1()
        {
            return View();
        }

        public ActionResult Index2()
        {
            return View();
        }
    }
}
```

Index1 View'ımız, LayoutPage1'i Index2 View'ımız, LayoutPage2'i kullanınsın istiyoruz

Görünüm Ekle

Görünüm adı: Index1

Şablon: Empty (model olmadan)

Model sınıfı:

Seçenekler:

Kısıtlı görünüm olarak oluştur

Başvuru betik kitaplıkları

Bir düzen sayfası kullanın:
~/Views/Shared/_LayoutPage1.cshtml

(Bir Razor _viewstart dosyasında ayarlandıysa, boş bırakın)

Ekle İptal

Görünüm Ekle

Görünüm adı: Index2

Şablon: Empty (model olmadan)

Model sınıfı:

Seçenekler:

Kısıtlı görünüm olarak oluştur

Başvuru betik kitaplıkları

Bir düzen sayfası kullanın:
~/Views/Shared/_LayoutPage2.cshtml

(Bir Razor _viewstart dosyasında ayarlandıysa, boş bırakın)

Ekle İptal

```
1      <!DOCTYPE html>
2
3      <html>
4          <head>
5              <meta name="viewport" content="width=device-width" />
6              <title>@ViewBag.Title</title>
7          </head>
8          <body>
9              <div>
10                 <div class="container">
11                     <div class="row">
12                         <div class="col-md12">
13                             <h2>BASLIK</h2>
14                         </div>
15                     </div>
16                 </div>
17                 @RenderBody()
18             </div>
19         </body>
20     </html>
```



← → C ⌂ localhost:44306/Home/Index1

 PAMUKKALE ÜNİVE... Gmail YouTube Tumor Detection U... ROI-Based

BASLİK

Index1

```
@{ Layout = "~/Views/Shared/_LayoutPage1.cshtml"; }
/*BU LAYOUT, LAYOUT 1'İ kullanacaktır anlamına geleceğini ifade ediyoruz*/
/*Layout 2'nin Layoutu birdir. Böylece iç içe layout oluşturduk.*@

<style>
    .row {
        justify-content: space-around;
    }

    .ac {
        display: inline-block;
        padding: 5px;
        background-color: oldlace;
        color: white;
        border: 3px solid wheat;
    }
</style>
<div class="col">
    <div class="row-md-4">
        <ul>
            <li class="ac"><a href="#">Alt Baslik1</a></li>
            <li class="ac"><a href="#">Alt Baslik2</a></li>
            <li class="ac"><a href="#">Alt Baslik3</a></li>
            <li class="ac"><a href="#">Alt Baslik4</a></li>
            <li class="ac"><a href="#">Alt Baslik5</a></li>
        </ul>
    </div>
    <div class="row-md-8">
        <p>Fatma AkalınFatma AkalınFatma AkalınFatma AkalınFatma AkalınFatma AkalınFatma A
        <p>Fatma AkalınFatma AkalınFatma AkalınFatma AkalınFatma AkalınFatma AkalınFatma AkalınFatma A
        <p>Fatma AkalınFatma AkalınFatma AkalınFatma AkalınFatma AkalınFatma AkalınFatma AkalınFatma A
    </div>
```

BASLİK

Alt Baslik1 Alt Baslik2 Alt Baslik3 Alt Baslik4 Alt Baslik5

Index2

Eğitim amaçlı olarak tasarlanan bu örnekleri temel alarak geçmiş yıl kazanmış olduğunuz frontend bilgisi ile daha güçlü layoutlar tasarlayabilirsiniz. Ek olarak zaman kalırsa bende son haftalarda sizlere verdiğim proje ödevine katkıda bulunması amacıyla frontend bilgilerinizi güçlendirmek amacıyla frontend projeleri anlatıyor olacağım

Dr.

LAYOUT İÇİNDE BÖLÜM(SECTION) TANIMLAMA (RENDERSECTION)

ASP.NET MVC'de oluşturduğunuz Layoutta sayfalara özgü değişen sadece bir bölüm vardı. Bununla birlikte projelerinizde yer alan Layoutta birden fazla bölümün sayfalarınızda değişkenlik göstermesini sağlayabilirsiniz.

Dr.

Normal şartlar altında Layout sayfalarındaki menü ya da footer bilgisi gibi ana iskelette olması gereken bilgilerin dışında Render body kısmına gelecek olan kodları view sayfalarında yazıyoruz. Yani diğer view sayfalarında yazdığımız HTML'ler gelip bu div tagları arasında yer alan renderBody kısmına yerleşiyor ve layout kullanan o sayfaların temel iskeleti ile birlikte kendi içeriklerini yazmalarına izin veriyor.

Buna ek olarak Layoutları da birbiri içerisinde kullanarak sabit olan kısımları daha da sabit hale getirmek mümkün. Değişken kısımlarda Section oluyor MVC'de. Section'dada bazı kısımlar her nekadar sabit olsa da içerisindeki içerikler değişebiliyor. Örneğin, her sayfada kategoriler vardır ama diğer sayfada ek olarak farklı etiketlerin gelmesini isteyebilirsin. Sabit kısımları böümlere ayırıp oraları değişken haline getirebiliyoruz.

Örnek
yapalım... Ö
Dr. Ö

Yeni projenizi yapılandırın

ASP.NET Web Uygulaması (.NET Framework) C# Windows Bulut Web

Proje adı: denemebesinciders

Konum: D:\aspnet\1_ADERSLER

Çözüm adı: denemebesinciders

Çözümü ve projeyi aynı dizine yerleştirmek

Altyapı: .NET Framework 4.6.2

Proje "D:\aspnet\1_ADERSLER\denem...

Yeni ASP.NET Web Uygulaması oluştur

Boş
ASP.NET uygulamaları oluşturmak için boş bir proje şablonu. Bu şablonda içerik yoktur.

Web Forms
ASP.NET Web Forms uygulamaları oluşturmak için bir proje şablonu. ASP.NET Web Forms, bilinen sürükle ve bırak yöntemiyle olay denetimli modeli kullanarak dinamik web siteleri oluşturmanızı sağlar. Tasarım yüzeyi ile yüzlerce denetim ve bileşen, veri erişimi olan gelişmiş, güçlü, kullanıcı arabirimini denetimli web sitelerini hızlı bir şekilde oluşturmanızı sağlar.

MVC
ASP.NET MVC uygulamaları oluşturmaya yarayan bir proje şablonu. ASP.NET MVC, Model-View-Controller mimarisini kullanarak uygulamalar oluşturmanıza olanak sağlar. ASP.NET MVC en son standartları kullanan uygulamalar oluşturmak için hızlı, test giderili geliştirmeye olanak sağlayan birçok özellik içerir.

Web API
Tarayıcılar ve mobil cihazlar dahil olmak üzere çok çeşitli istemcilere erişebilen RESTful HTTP hizmetleri oluşturmaya yarayan bir proje şablonudur.

Tek Sayfalı Uygulama
ASP.NET Web API kullanarak JavaScript temelli zengin istemci tarafi HTML5 uygulamaları oluşturmak üzere kullanılan proje şablonudur. Tek Sayfalık Uygulamalar, HTML5, CSS3 ve JavaScript'in kullanıldığı istemci tarafi etkileşimler içeren zengin bir kullanıcı deneyimi sağlar.

Kimlik Doğrulama
Yok

Klasörleri ve çekirdek başvurularını ekle
 Web Forms
 MVC
 Web API'si

Gelişmiş
 HTTPS'yi Yapılandır
 Docker desteği
(Docker Desktop gereklidir)
 Ayrıca birim testleri için bir proje oluştur
denemebesinciders.Tests

Geri Oluştur

Yeni İskeleli Öğe Ekle

AKALIN

▲ Yüklü

▲ Ortak

▲ MVC

- Alan
- Denetleyici
- Göster
- Web API'si

 Entity Framework kullanan, görünümleri olan MVC 5 Denetleyici

 **MVC 5 Denetleyici - Boş**

 Okuma/yazma eylemleri olan MVC 5 Denetleyici

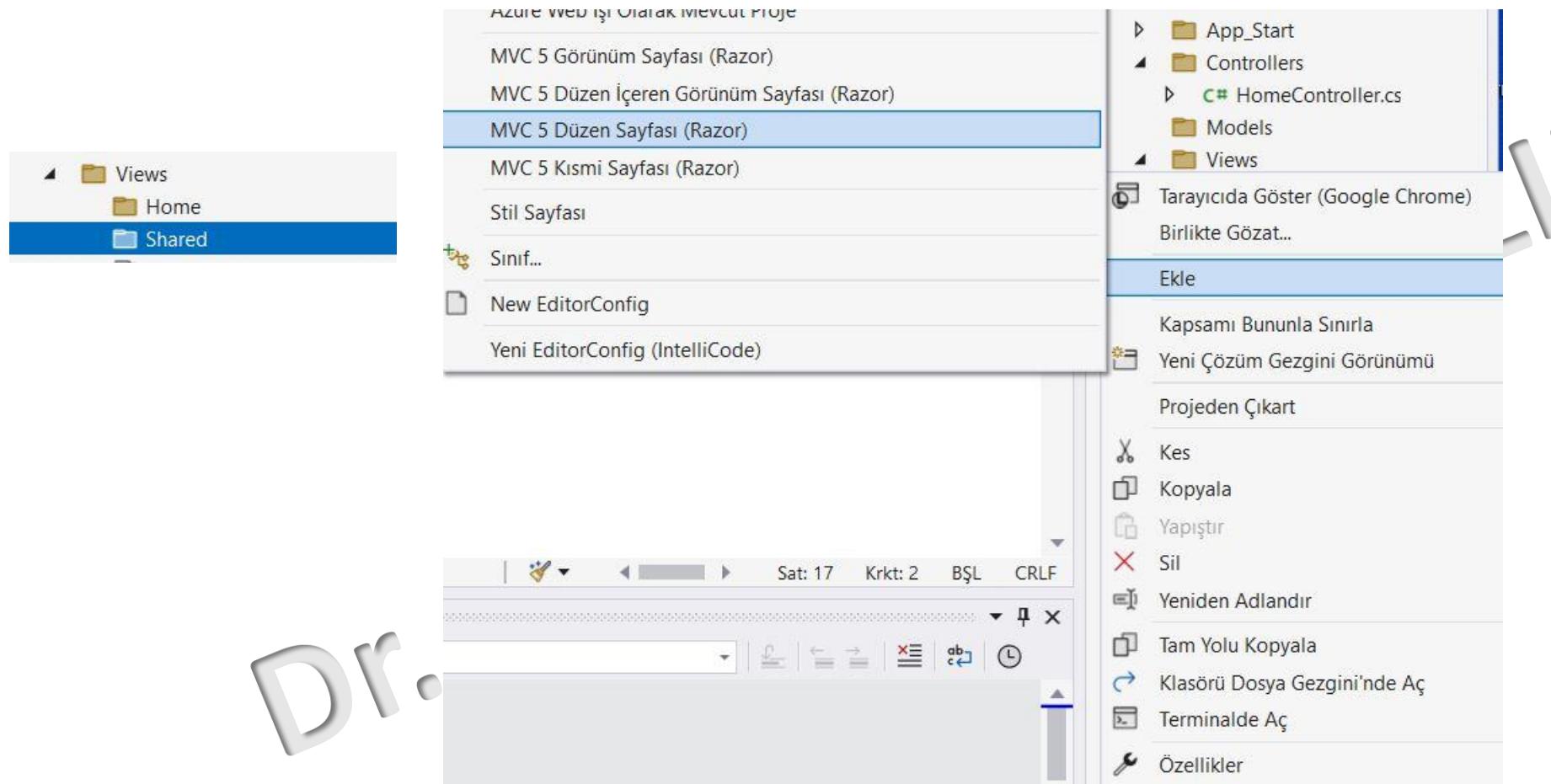
MVC 5 Denetleyici - Boş
şunun tarafından Microsoft v5.0.0.0
Boş bir MVC denetleyici.
Kimlik: MvcControllerEmptyScaffolder

Ekle İptal

Denetleyici Ekle

Denetleyici adı:

Ekle İptal



Dr.

Öğe Adı Belirt X Öğe Adı Belirt X

öge adı: öge adı:

Tamam İptal Tamam İptal

Layout sayfalarının içeriğini doldurmadan Viewları oluşturalım. Bu doğrultuda, Index1 ve Index3 viewları _LayoutPage1'i template olarak alsın.

Görünüm Ekle

X

Görünüm adı: Index1

Şablon: Empty (model olmadan)

Model sınıfı:

Seçenekler:

Kısıtlı görünüm olarak oluştur

Başvuru betik kitaplıklarını

Bir düzen sayfası kullanın:
~/Views/Shared/_LayoutPage1.cshtml

(Bir Razor _viewstart dosyasında ayarlandıysa, boş bırakın)

Ekle İptal

AKALIN

Görünüm Ekle

X

Görünüm adı: Index3

Şablon: Empty (model olmadan)

Model sınıfı:

Seçenekler:

Kısıtlı görünüm olarak oluştur

Başvuru betik kitaplıklarını

Bir düzen sayfası kullanın:
~/Views/Shared/_LayoutPage1.cshtml

(Bir Razor _viewstart dosyasında ayarlandıysa, boş bırakın)

Ekle İptal

Index2 view'ı ise _LayoutPage2'yi template olarak alınsın.

Görünüm Ekle

X N

Görünüm adı:

Şablon:

Model sınıfı:

Seçenekler:

Kısıtlı görünüm olarak oluştur

Başvuru betik kitaplıklarını

Bir düzen sayfası kullanın:

...

(Bir Razor _viewstart dosyasında ayarlandıysa, boş bırakın)

Oluşturduğumuz Index1, Index2 ve Index3 sayfalarının içeriğini dolduralım. Index 1 ve Index 3 aşağıda sunulmuştur.

```
Index3.cshtml      Index2.cshtml      Index1.cshtml      _LayoutPage2.cs
1
2  @{
3      ViewBag.Title = "Index1";
4      Layout = "~/Views/Shared/_LayoutPage1.cshtml";
5  }
6
7  <h2>Burası Index1 sayfası, Hosgeldiniz</h2>
8

Index3.cshtml      Index2.cshtml      Index1.cshtml      _LayoutPage2.cs      _LayoutPage1.cshtml      HomeController.cs
1
2  @{
3      ViewBag.Title = "Index3";
4      Layout = "~/Views/Shared/_LayoutPage1.cshtml";
5  }
6
7  <h2 style="background-color:deeppink">Direk render body kısmının içerisinde yerleşek index3 sayfaası</h2>
8
9
10
```

Index2 sayfası iki ayrı section kısmı içermektedir. BU kısımlar değişken içeriğtedir. Diğer bir ifade ile LayoutPage2 üzerine temellenen bir Index4 sayfası açsak oradaki linkler ve footer sectionlarını Index2 sectiondan ayrı bir tasarımda sunabilirim. Bacgroundu pink olur mesela oradaki sectionda vb. Tabi bu örnekleri çoğaltabileceğimiz çok fazla durum çıkacak karşımıza gerçek hayat problemleri ile karşılaşlığımızda..

```
Index3.cshtml Index2.cshtml ✎ X Index1.cshtml _LayoutPage2.cshtml _LayoutPage1.cshtml HomeController.cs
1
2  @{
3      ViewBag.Title = "Index2";
4      Layout = "~/Views/Shared/_LayoutPage2.cshtml";
5  }
6
7  <h2>Index2</h2>
8
9  <h2 style="background-color:aquamarine">Section yapısını anlatmak için bu sayfa tasarlandı.</h2>
10
11 @section linkler{
12     <p style="background-color:dodgerblue">Bu kısım sectionun kısımına tekabul edecek</p>
13 }
14
15 @section footer{
16     <p style="background-color:violet">Bu kısım footer sectionun kısımına tekabul edecek</p>
17 }
```

Layout sayfalarını
inceleyelim..

_Layout1 sayfasını her
zaman yaptığımız gibi
düzenlerken _Layout2
sayfasında 2 ayrı
section
tanımlamalıyız..

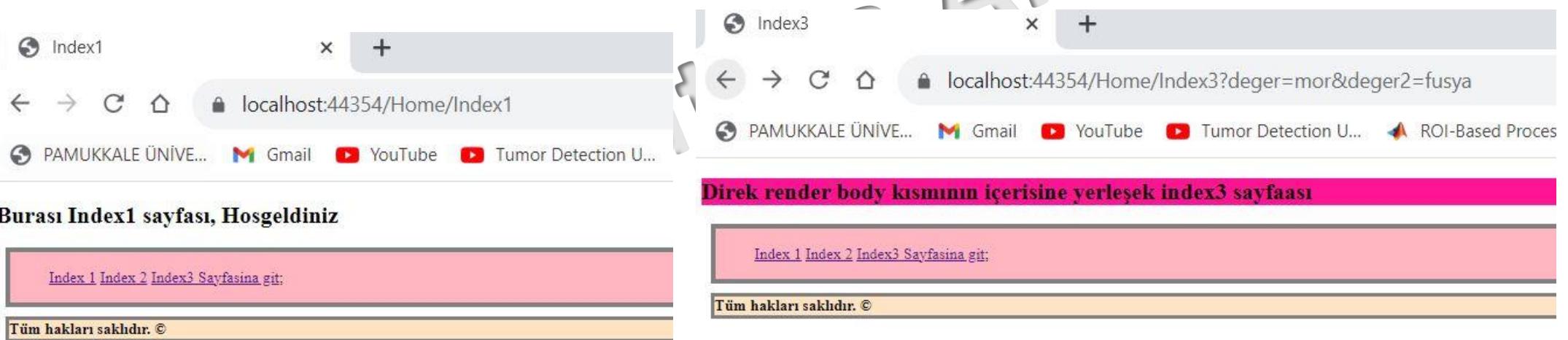
Dr. Öğ

```
<body>
    <div>
        @RenderBody()
    </div>

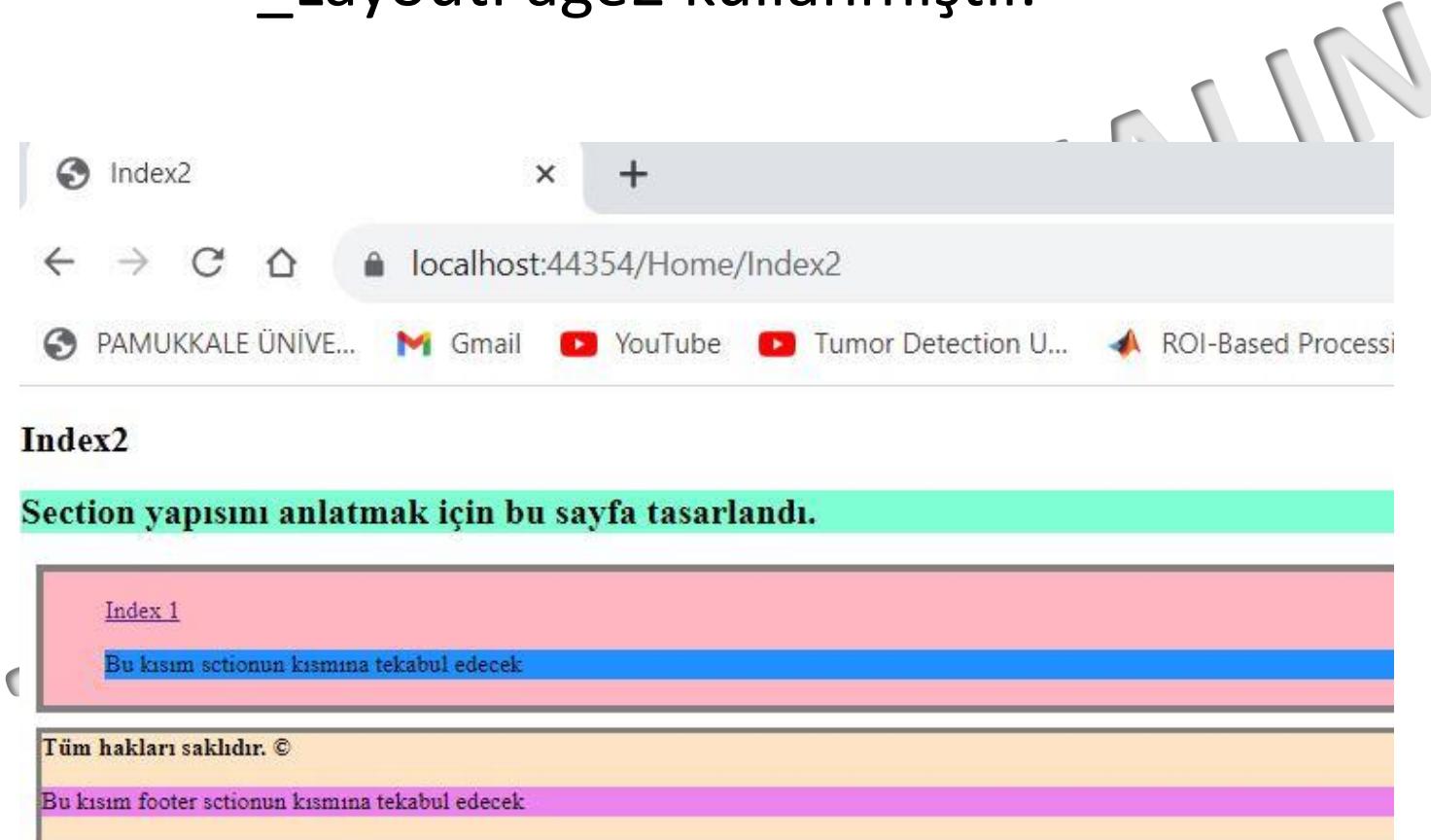
    <div id="menu" class="block">
        @*Header*@<br/>
        <ul>
            <li>@Html.ActionLink("Index 1", "Index1", "Home")</li>
            @RenderSection("linkler", false)
        </ul>
    </div>

    <div class="block2">
        @*Footer*@<br/>
        <b>Tüm hakları saklıdır. ©</b>
        @RenderSection("footer", false)
    </div>
</body>
</html>
```

Viewlar, Controllerlar ve Layout sayfalarını doldurduktan sonra ekran çıktılarına bakalım... Görüldüğü gibi Index 1 ve Index3 aynı layoutu kullanmıştır.



Index2 ise farklı section kısmının kullanılabilirliğine izin veren _LayoutPage2 kullanmıştır.



PARÇALI(PARTIAL) VIEW KAVRAMI VE KULLANIMI

ASP.net MVC'nin en kullanışlı elemanlarından biri olan partial(parçalı) viewlar yazılım geliştiricinin işini birçok noktada kolaylaştırmaktadır Partial View'lar bir kere oluşturulduktan sonra proje içinde istenilen her yerde kullanılabilir. Bu sayede parçalı olarak oluşturulan bir View'da yapılan güncellemeler tüm sayfaları etkileyecektir (Tek bir noktadan Güncelleme yapılması yeterli olacaktır)

Başka bir deyişle, parçalı görünümler anlamına gelen Partial View, bir sayfa için yaptığınız bir modülü başka bir sayfa içinde göstermek istediğimizde kullandığımız bir yapıdır. Layout pageler ilede bunu yapmak mümkündür. Fakat 2 ayrı sayfada benzer kısımlar var diye buna ayrı bir layout yapmak çok mantıklı bir eylem değildir. Bunun yerine o kısımları 1 kere kodlamalı ve kodladığımız parçaları istediğimiz yerde kullanabiliriz.

Örneğin iki farklı sayfada aynı kısımlar değil de aynı sayfanın içerisinde iki farklı kısım olduğunu varsayıalım. Bu doğrultuda sağ tarafta linkler diye bir kısım var ve aynı zamanda ben bu linkleri sayfanın en altında da göstermek istiyorum. Bu kısımda 2 layout tasarlayarak iç içe layout mantığını mı kullanmalıyım?

Hayır, bu kısmı partial view olarak hazırlayıp istediğiniz yerlerde bu partial viewı çağrılabilsiniz. Özetle ilgili kodu bir kere yazabiliriz ve istediğimiz kısımlarda çıkışını sağlayabiliriz.

Kullanım Örnekleri

Konuyu bir örnek senaryo üzerinden inceleyelim. Projemizde online satış sistemi geliştirdiğimizi varsayıalım. Bu alışveriş sistemi içerisindeki sayfaların bazlarında ürünlerin listelenmesi gerekecektir.

- 1-ana sayfada son eklenen ürünlerin listesini
- 2 ürün kategorisi sayfalarından kategoriye ait ürünlerin listesini
- 3 arama sayfasında arama yapılan kelimeye uygun ürünlerini listesini
- 4 indirimdekiler sayfasında indirime giren ürünlerin listesinin bulunduğu düşünelim

Bu 4 ürünler listesi görünümünün tasarımları aynıdır. Sadece gösterdikleri veriler yani ürünler birbirinden farklıdır.

Peki bu durumda ne kullanmalıyız?

Partial View kullanmayan bir yazılımcı ana sayfa kategori sayfası arama sayfası ve indirimdekiler sayfasında ayrı ayrı ürünler listesinin tasarımını oluşturup ilgili ürünleri Bu tasarım ile gösterecektir tasarımda bir düzenleme gereğinde tüm sayfalarda Tek Tek değişiklik yapmak zorunda kalacaktır

Partial view kullanan bir yazılımcı ise ürünler listesini bir partial view kullanarak oluşturur. Bu view'i 4 sayfaya birden yerleştirir ve ürünlerin bu kontrolde görüntülenmesini sağlar. Tasarımda bir düzenleme gereğinde ise sadece partial view içinde düzenleme yapması yeterli olacaktır. Sayfa içerisindeki değerler Partial view'dan alındığı için burası değiştiğinde tüm sayfalardaki tasarımda otomatik olarak değişecekti. Bu yöntem tasarım geliştiricinin birden fazla yerde kullandığı işlemleri veya tasarımları tek bir noktada oluşturarak yönetimi bu noktadan yapmasını sağlamaktadır.

HTML Helper metodlarını kullanarak parent view'daki partial view'leri render etmek istediğimizde sırasıyla

`@html.Partial()`

`@html.RenderPartial()`

Yaklaşımlığını kullanırız.

Derste örnekleri yapılacaktır.

<https://learn.microsoft.com/en-us/dotnet/api/system.web.mvc.html.partialextensions.partial?view=aspnet-mvc-5.2#overloads>

<https://learn.microsoft.com/en-us/dotnet/api/system.web.mvc.html.renderpartialextensions?view=aspnet-mvc-5.2#methods>

<https://learn.microsoft.com/en-us/dotnet/api/system.web.mvc.html.childactionextensions.renderaction?view=aspnet-mvc-5.2#overloads>

The screenshot shows the Visual Studio IDE interface with the following details:

- Top Bar:** Görünüm, Git, Proje, Derle, Hata Ayıklama, Test, Analiz, Araçlar, Uzantılar, Pencere.
- Toolbars:** Standard toolbar with icons for Save, Undo, Redo, Cut, Copy, Paste, Find, etc.
- Project Explorer:** Shows the project structure for 'WebApplication13' with 'HomeController.cs' selected.
- Solution Explorer:** Shows the solution structure with 'WebApplication13' selected.
- Code Editor:** The 'about.cshtml' file is open, containing the following code:

```
<body>
    <p>Burası about sayfasıdır</p>
    <div>
        1
    </div>
    <div>
        2
    </div>
    <div>
        3
        <p>
            @Html.RenderPartial("_PartialPage1");
            @Html.Partial("_PartialPage1")
        </p>
        3
    </div>
    <div>
        4
    </div>
```
- Properties Window:** Shows properties for the selected item.
- Task List:** Shows a list of tasks related to the current file.
- Output Window:** Shows build results.
- Help and Support:** Links to PAMUKKALE ÜNİVERSİTESİ, Gmail, YouTube, and other resources.
- Bottom Status Bar:** Sat: 25, Krkt: 52, BSL, CRLF.
- Right Side Panels:** 'Çözüm Gezgini' (Solution Explorer) and 'Hata Listesi' (Error List).
- Browser Preview:** Shows the rendered output of the 'about.cshtml' page at localhost:44365/Home/about.

UYGULAMA

Partial view, ortak kullanılan html parçaları oldukları için genel olarak shared klasörü oluşturup shared klasörünün içerisine yerleştirmemiz mantıklıdır. Bu süreç aşağıdaki gibi gerçekleşmektedir.

Dr. Öğr. Fatiha YILMAZ

Yeni projenizi yapılandırın

ASP.NET Web Uygulaması (.NET Framework)

Proje adı

denemealtinciders

Konum

D:\aspnet\1_ADERSLER

Çözüm adı (i)

denemealtinciders

Çözümü ve projeyi aynı dizine yerleştirin

Altyapı

.NET Framework 4.6.2

Proje "D:\aspnet\1_ADERSLER\denemealtinciders\denemealtinciders\"

Yeni ASP.NET Web Uygulaması oluştur



Boş

ASP.NET uygulamaları oluşturmak için boş bir proje şablonu. Bu şablonda içerik yoktur.



Web Forms

ASP.NET Web Forms uygulamaları oluşturmak için bir proje şablonu. ASP.NET Web Forms, bilinen sürükle ve bırak yöntemiyle olay denetimli modeli kullanarak dinamik web siteleri oluşturmanızı sağlar. Tasarım yüzeyi ile yüzlerce denetim ve bileşen, veri erişimi olan gelişmiş, güçlü, kullanıcı arabirimini denetimli web sitelerini hızlı bir şekilde oluşturmanızı sağlar.



MVC

ASP.NET MVC uygulamaları oluşturmaya yarayan bir proje şablonu. ASP.NET MVC, Model-View-Controller mimarisini kullanarak uygulamalar oluşturmanıza olanak sağlar. ASP.NET MVC en son standartları kullanan uygulamalar oluşturmak için hızlı, test güdümlü geliştirmeye olanak sağlayan birçok özellik içerir.



Web API

Tarayıcılar ve mobil cihazlar dahil olmak üzere çok çeşitli istemcilere erişebilen RESTful HTTP hizmetleri oluşturmaya yarayan bir proje şablonudur.



Tek Sayfalı Uygulama

ASP.NET Web API kullanarak JavaScript temelli zengin istemci tarafı HTML5 uygulamaları oluşturmak üzere kullanılan proje şablonudur. Tek Sayfalı Uygulamalar, HTML5, CSS3 ve JavaScript'in kullanıldığı istemci tarafı etkileşimler içeren zengin bir kullanıcı deneyimi sağlar.

Kimlik Doğrulama

Yok

Klasörleri ve çekirdek başvurularını ekle

- Web Forms
- MVC
- Web API'si

Gelişmiş

- HTTPS'yi Yapılandır
- Docker desteği
(Docker Desktop gerektiriyor)
- Ayrıca birim testleri için bir proje oluştur
denemealtinciders.Tests

Geri

Oluştur

Yeni İskeleli Öğe Ekle

▲ Yüklü

- ▲ Ortak
- ▲ MVC
 - Alan
 - Denetleyici**
- Göster
- Web API'si

 Entity Framework kullanan, görünümleri olan MVC 5 Denetleyici

 **MVC 5 Denetleyici - Boş**

 Okuma/yazma eylemleri olan MVC 5 Denetleyici

MVC 5 Denetleyici - Boş
şunun tarafından Microsoft
v5.0.0.0

Boş bir MVC denetleyici.

Kimlik: MvcControllerEmptyScaffolder

Denetleyici Ekle

Denetleyici adı:

×

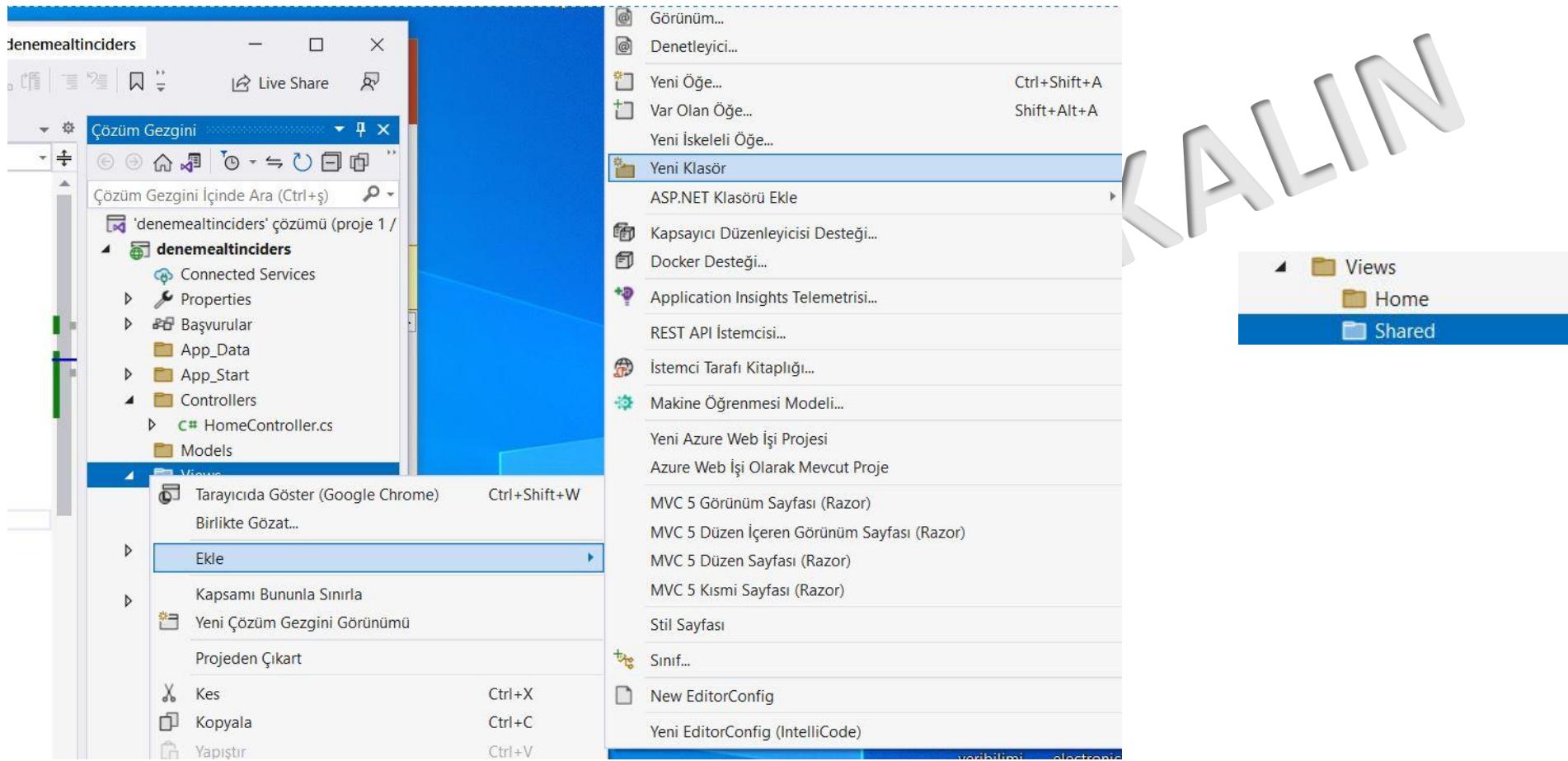
Dr. Öğr. Fauz

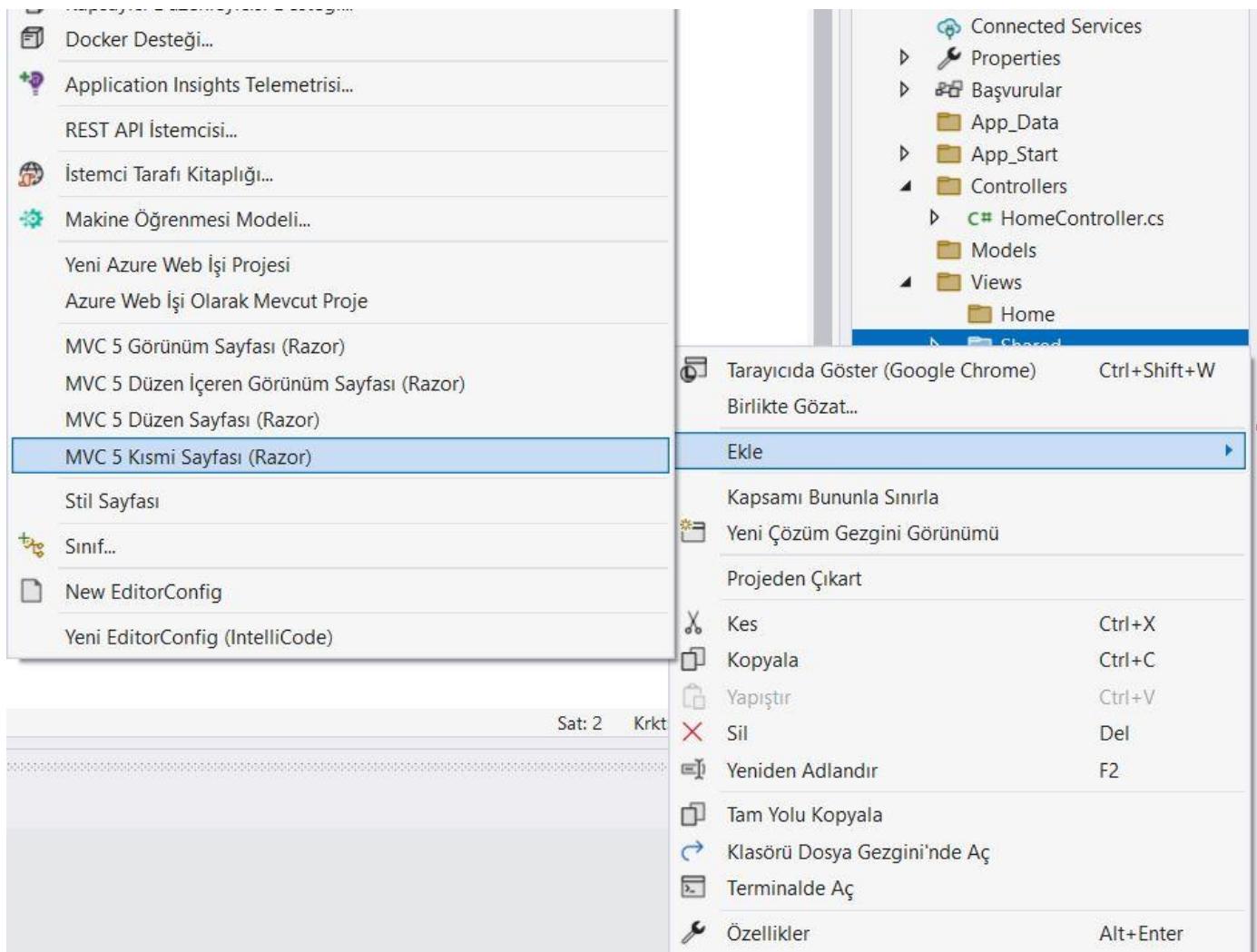
Dr. i

ALIN

```
HomeController.cs  X
denemealtinciders
denemealtinciders.Controllers

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;
5  using System.Web.Mvc;
6
7  namespace denemealtinciders.Controllers
8  {
9      public class HomeController : Controller
10     {
11         // GET: Home
12         public ActionResult homepage()
13         {
14             return View();
15         }
16         public ActionResult about()
17         {
18             return View();
19         }
20     }
21 }
```





Controller’ımızı oluşturduk
ve düzenledik. Ardından
yanda ki gibi partial
viewimizi oluşturalım ve
düzenleyelim.

Dr. Öğ

_PartialLinkler.cshtml ✎ × HomeController.cs

```
1
2  

#### 3 Partial kavramının anlatılması için oluşturulan Linkler 4


5  <hr />
6  <ul>
7      <li>
8          <a href="#">Link1</a>
9      </li>
10     <li>
11         <a href="#">Link2</a>
12     </li>
13     <li>
14         <a href="#">Link3</a>
15     </li>
16     <li>
17         <a href="#">Link4</a>
18     </li>
19     <li>
20         <a href="#">Link5</a>
21     </li>
22 </ul>
```

Şimdi, controllerimızda tanımladığımız about sayfasının viewını oluşturalım

Dr. Öğr.

Görünüm Ekle

X

Görünüm adı:

about

Şablon:

Empty (model olmadan)

Model sınıfı:

Seçenekler:

- Kısıtlı görünüm olarak oluştur
- Başvuru betik kitaplıkları
- Bir düzen sayfası kullanın:

(Bir Razor_viewstart dosyasında ayarlandıysa, boş bırakın)

Ekle

İptal

About viewimizi
düzenleyelim ve ilgili
kısma partial view
ekleyelim

Dr. Öğr.

```
about.cshtml  X _PartialLinkler.cshtml      HomeController.cs
4   <!DOCTYPE html>
5   <html>
6   <head>
7       <meta name="viewport" content="width=device-width" />
8       <title>about</title>
9   </head>
10  <body>
11      <div>
12          <p>
13              Bu derste partial view kavramı anlatılmaktadır.
14              Bu div birinci divdir
15          </p>
16      </div>
17      <br />
18      <div>
19          <p>
20              Bu div ikinci divdir
21          </p>
22      </div>
23      <br />
24      <div>
25          <p>
26              Üçüncü divin baslangıcı
27          </p>
28          <hr />
29          <p>
30              Bu div üçüncü divdir ve partial view buraya yerleşecektir.
31          </p>
32          @Html.Partial("_PartialLinkler")
33          <hr />
34          <p>
35              Üçüncü divin bitisi
36          </p>
37      </div>
38  </body>
39  </html>
```

NOT

Oluşturulan partial viewların kullanımı için razor blokların içerisine ilgili kod satırlarını yazarak süreci yöneteceğiz.

Partial view
kullanarak
gerçeklediğimiz
projede elde
ettiğimiz web
çıktısı aşağıdaki
gibi olacaktır.

about

localhost:44309/Home/about

PAMUKKALE ÜNİVE... Gmail YouTube Tumor

Bu derste partial view kavramı anlatılmaktadır. Bu div birinci divdir

Bu div ikinci divdir

Üçüncü divin başlangıcı

Bu div üçüncü divdir ve partial view buraya yerleşecektir.

Partial kavramının anlatılması için oluşturulan Linkler

- [Link1](#)
- [Link2](#)
- [Link3](#)
- [Link4](#)
- [Link5](#)

Üçüncü divin bitisi

Özetle, Partial View kullanımı sayesinde merkezi noktadan yaptığımız değişiklik, bu View'ı kullanan tüm sayfalarda uygulanacaktır. Bu işlevsellik yazılım geliştiricilerin zaman kazanmasına imkan sağlamaktadır.

Dr. Öğr. Fatma AYDIN

VIEWSTART KAVRAMI

Şimdiye kadar yaptığınız örneklerde her bir view'ın layout tanımlamasını tek tek yapıyorduk. Peki tüm viewların layoutunu bir noktada tanımlayabilir miyiz?

EVET

Geliştirdiğiniz projede tek bir Layoutunuz var ise bu Layout dosyasını her viewda tek tek tanımlamak yerine bir yerde tanımlayıp tekrar tanımlama zorunluluğunu ortadan kaldırabiliriz.

Bunun için viewstart isimli bir dosya oluşturacağız.

Yeni projenizi yapılandırin

ASP.NET Web Uygulaması (.NET Fra

Proje adı

denemeyedinciders

Konum

D:\aspnet\1_ADERSLER

Çözüm adı ①

denemeyedinciders

Çözümü ve projeyi aynı dizine yerleştirin

Altyapı

.NET Framework 4.6.2

Proje "D:\aspnet\1_ADERSLER\denemeyedinciders\den

Yeni ASP.NET Web Uygulaması oluştur



Boş

ASP.NET uygulamaları oluşturmak için boş bir proje şablonu. Bu şablonda içerik yoktur.



Web Forms

ASP.NET Web Forms uygulamaları oluşturmak için bir proje şablonu. ASP.NET Web Forms, bilinen sürükle ve bırak yöntemiyle olay denetimli modeli kullanarak dinamik web siteleri oluşturmanızı sağlar. Tasarım yüzeyi ile yüzlerce denetim ve bileşen, veri erişimi olan gelişmiş, güçlü, kullanıcı arabirimini denetimli web sitelerini hızlı bir şekilde oluşturmanızı sağlar.



MVC

ASP.NET MVC uygulamaları oluşturmaya yarayan bir proje şablonu. ASP.NET MVC, Model-View-Controller mimarisini kullanarak uygulamalar oluşturmanıza olanak sağlar. ASP.NET MVC en son standartları kullanan uygulamalar oluşturmak için hızlı, test güdümlü geliştirmeye olanak sağlayan birçok özellik içerir.



Web API

Tarayıcılar ve mobil cihazlar dahil olmak üzere çok çeşitli istemcilere erişebilen RESTful HTTP hizmetleri oluşturmaya yarayan bir proje şablonudur.



Tek Sayfalı Uygulama

ASP.NET Web API kullanarak JavaScript temelli zengin istemci tarafı HTML5 uygulamaları oluşturmak üzere kullanılan proje şablonudur. Tek Sayfalı Uygulamalar, HTML5, CSS3 ve JavaScript'in kullandığı istemci tarafı etkileşimler içeren zengin bir kullanıcı deneyimi sağlar.

Kimlik Doğrulama

Yok

Klasörleri ve çekirdek başvurularını ekle

Web Forms

MVC

Web API'si

Gelişmiş

HTTPS'yi Yapılandır

Docker desteği

(Docker Desktop gerektiriyor)

Ayrıca birim testleri için bir proje oluştur

denemeyedinciders.Tests

Geri

Oluştur

Dr.

JIN

```
HomeController.cs* + X
denemeyedinciders
denemeyedinciders.Controllers.HomeContro
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;
5  using System.Web.Mvc;
6
7  namespace denemeyedinciders.Controllers
8  {
9      public class HomeController : Controller
10     {
11         // GET: Home
12         public ActionResult homepage()
13         {
14             return View();
15         }
16
17         public ActionResult about()
18         {
19             return View();
20         }
21     }
22 }
23
```

Görünüm Ekle

Görünüm adı:

Şablon:

Model sınıfı:

Seçenekler:

 Kısıtlı görünüm olarak oluştur Başvuru betik kitaplıklarını Bir düzen sayfası kullanın:

Görünüm Ekle

Görünüm adı:

Şablon:

Model sınıfı:

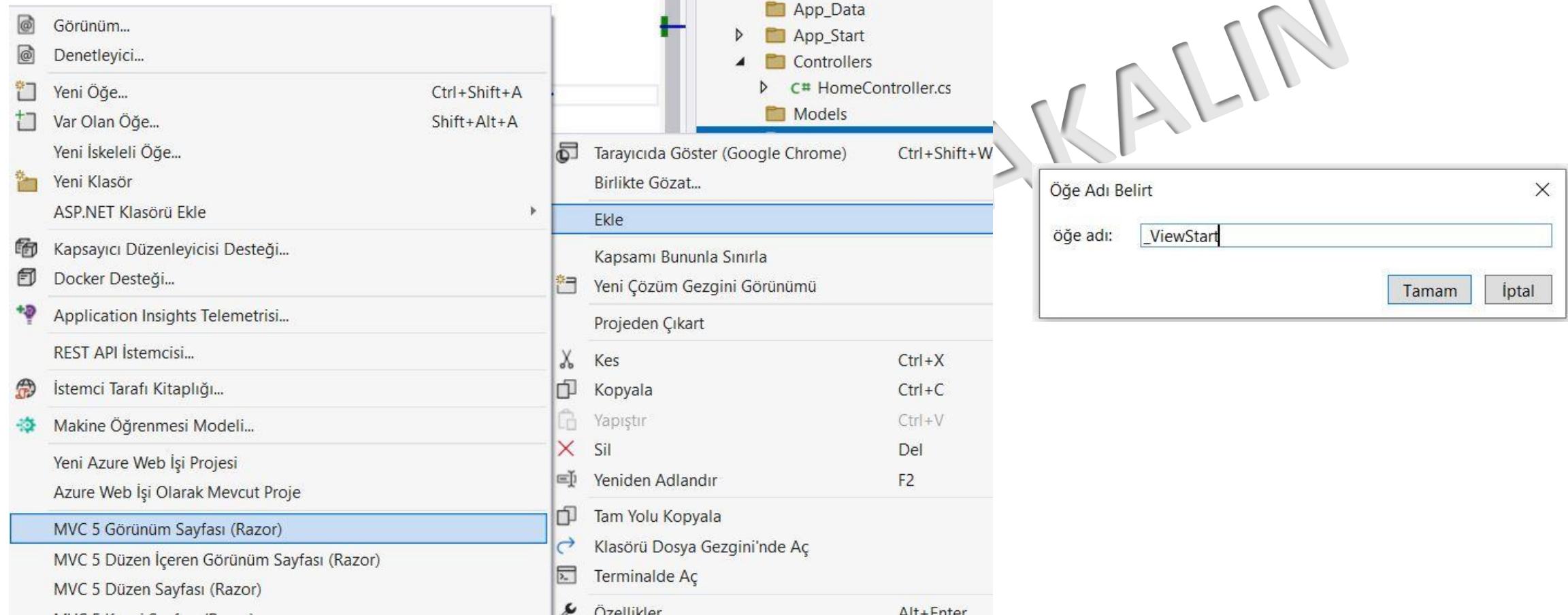
Seçenekler:

 Kısıtlı görünüm olarak oluştur Başvuru betik kitaplıklarını Bir düzen sayfası kullanın:

(Bir Razor _viewstart dosyasında ayarlandıysa, boş bırakın)

(Bir Razor _viewstart dosyasında ayarlandıysa, boş bırakın)

Controller ve View'ı düzenledikten sonra View'ın altında olacak şekilde Viewstart görünüm sayfasını ekledim.



Şimdi View klasörünün altında _ViewStart ismindeki dosyamız oluştuktan sonra ViewStart için varsayılan layout tanımını yaparız. Ardından tüm sayfalarındaki öncesinde yapılmış bir layout tanımı var ise kaldırabilirsin. Sadece ilgili view sayfalarında sayfanın HTML'i kalsa yeterlidir.

Özetle, layout tanımı yapmadığımız sayfalarda bu layout kullanılacak ve varsayılan layout sayfaları gelecek. Ama anasayfanın layoutu farklı olsun isterseniz yeni layout oluşturup anasayfaya özgü bir layout tanımlaması yapmamız gerekecek.

Viewstart dosyasında varsayılan layout olacağına ilişkin üst satırda bir tanım yapılmıştır



```
homepage.cshtml _ViewStart.cshtml ✎ X HomeController.cs

@{
    Layout = "~/Views/_ViewStart.cshtml";
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title></title>
</head>
<body>
    <div>
        <p style="border:3px solid purple; background-color:lightpink; color:deeppink" >
            Burası viewstart dosyasıdır
        </p>
    </div>
</body>
</html>
```

View sayfaları ve çıktıları :

The image shows a developer's workspace with two browser tabs and their corresponding code files.

Top Left: A code editor window titled "homepage.cshtml". It contains the following code:

```
@{ Layout = null; }

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>homepage</title>
</head>
<body>
    <div>
    </div>
</body>
</html>
```

Top Right: A browser window titled "localhost:44392/Home/homepage". The address bar shows "localhost:44392/Home/homepage". The page content is "Burası viewstart dosyasıdır".

Bottom Left: A code editor window titled "about.cshtml". It contains the following code:

```
1
2  @{
3      Layout = null;
4  }

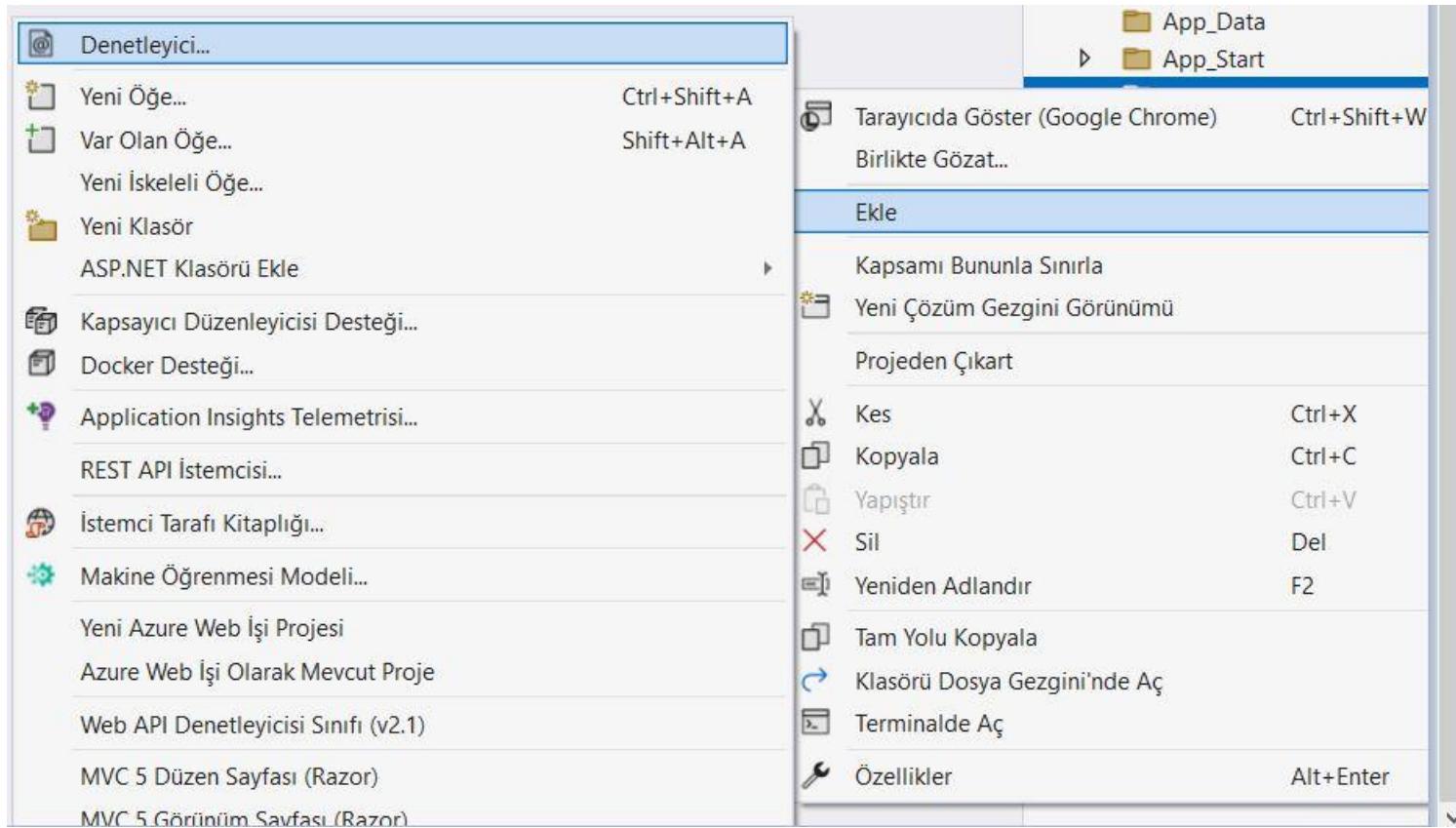
5
6  <!DOCTYPE html>

7
8  <html>
9  <head>
10     <meta name="viewport" content="width=device-width" />
11     <title>about</title>
12 </head>
13 <body>
14     <div>
15     </div>
16 </body>
17 </html>
```

Bottom Right: A browser window titled "localhost:44392/Home/about". The address bar shows "localhost:44392/Home/about". The page content is "Burası viewstart dosyasıdır".

A large watermark "KALIN" is diagonally across the top right, and a stylized "Dr." logo is on the left.

Ufak bir örnek yapalım...



Yeni İskeleli Öğe Ekle

▲ Yüklü

▲ Ortak

▲ MVC

- Alan
- Denetleyici**
- Göster
- Web API'si



Entity Framework kullanan, görüntülerini MVC 5 Denetleyici



MVC 5 Denetleyici - Boş



Okuma/yazma eylemleri olan MVC 5 Denetleyici

MVC 5 Denetleyici - Boş
şunun tarafından Microsoft
v5.0.0.0
Boş bir MVC denetleyici.
Kimlik: MvcControllerEmptyScaffold

Ekle İptal

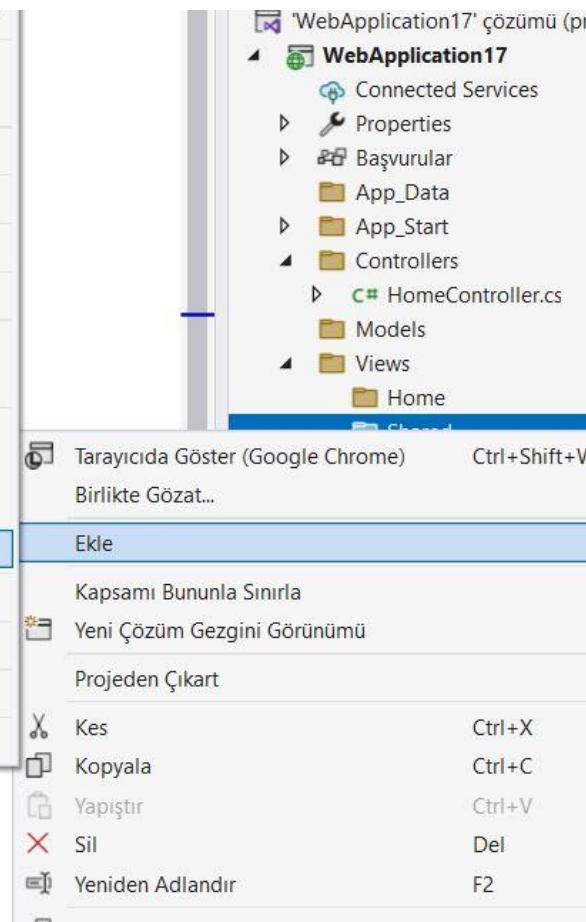
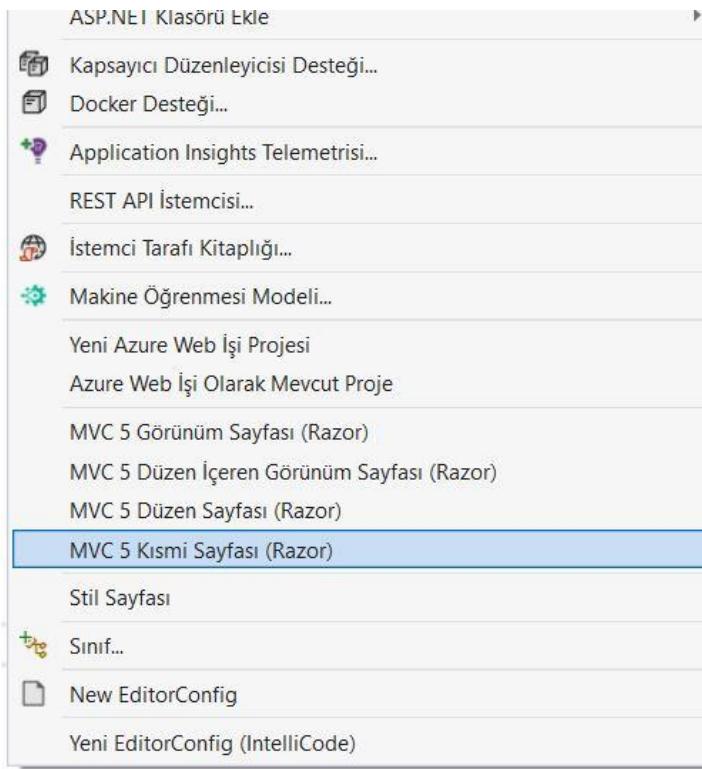


X

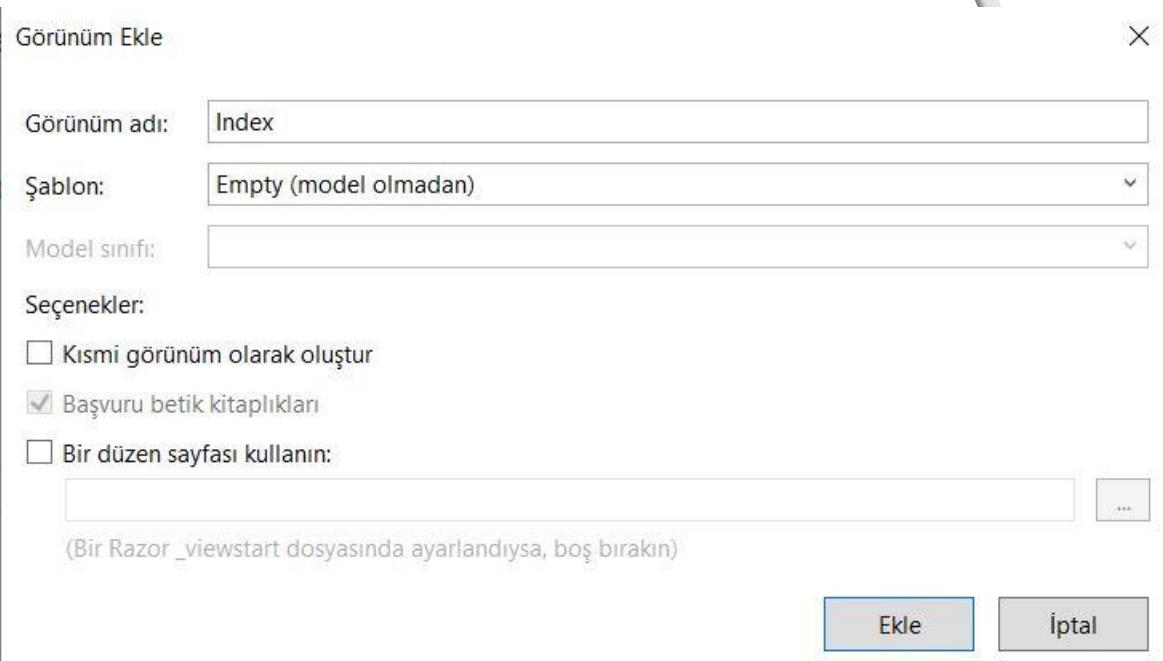
Denetleyici Ekle

Denetleyici adı:

Ekle İptal



```
namespace WebApplication17.Controllers
{
    [Route("Home")]
    public class HomeController : Controller
    {
        // GET: Home
        [HttpGet]
        public ActionResult Index()
        {
            return View();
        }
    }
}
```





```
Index.cshtml _PartialPage1.cshtml HomeController.cs
```

```
1
2     @{
3         Layout = null;
4     }
5
6     <!DOCTYPE html>
7
8     <html>
9         <head>
10            <meta name="viewport" content="width=device-width" />
11            <title>Index</title>
12        </head>
13        <body>
14            <div>
15                Burası index sayfası
16            </div>
17        </body>
18    </html>
19
```

```
Index.cshtml _PartialPage1.cshtml HomeController.cs
```

```
1     <div>Partial View Sayfası</div>
2
```

Görünüm Ekle

Görünüm adı:

Şablon:

Model sınıfı:

Seçenekler:

- Kismi görünüm olarak oluştur
- Başvuru betik kitaplıklarını
- Bir düzen sayfası kullanın:

...

(Bir Razor _viewstart dosyasında ayarlandıysa, boş bırakın)

Ekle

İptal

// GET: Home

0 başvuru

```
public ActionResult Index()
```

```
{
```

```
    return View();
```

```
}
```

0 başvuru

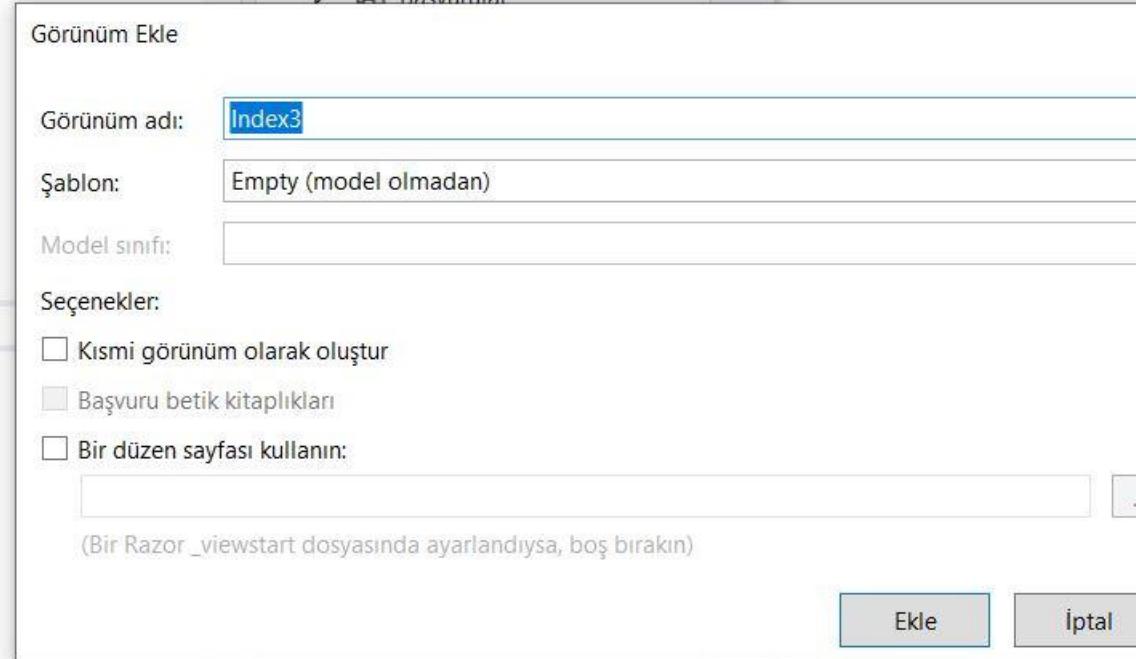
```
public ActionResult Index2()
```

```
{
```

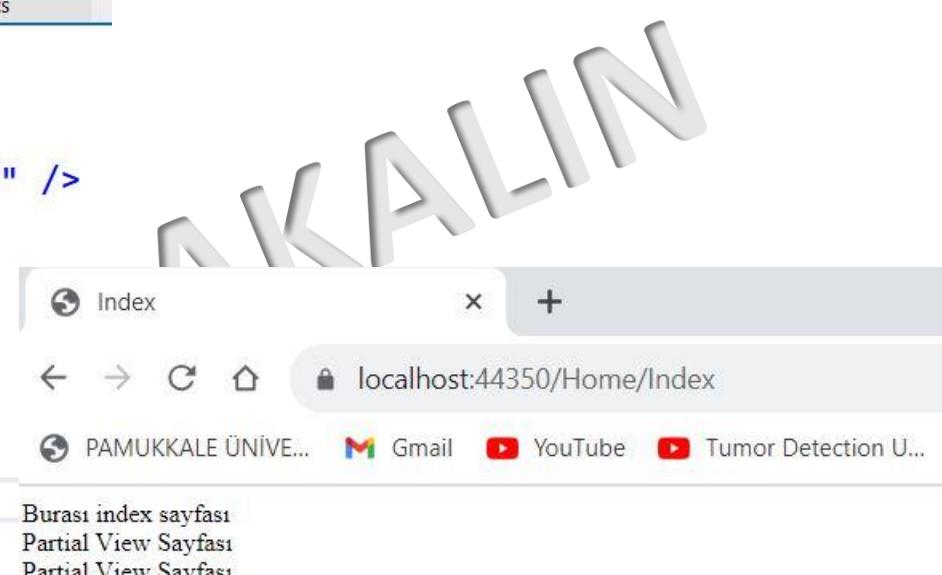
```
    return View();
```

```
}
```

```
0 başvuru  
public ActionResult Index2()  
{  
    return View();  
}  
0 başvuru  
public ActionResult Index3()  
{  
    return View();  
}
```



```
Index3.cshtml    Index2.cshtml    Index.cshtml    _PartialPage1.cshtml    HomeController.cs  
  
-<html>  
-<head>  
  <meta name="viewport" content="width=device-width" />  
  <title>Index</title>  
</head>  
-<body>  
  <div>  
    Burası index sayfası  
  
    @Html.Partial("_PartialPage1")  
    @{Html.RenderPartial("_PartialPage1");}  
    @{Html.RenderAction("Index3", "Home");}  
  </div>  
  
</body>  
</html>
```



Index4 sayfası
oluşturup içerisine
«Index4 sayfasıdır
Burası» yazalım.

Ardından bu sayfaya
ek olarak
Controller'a gelelim
ve yanda gösterilen
fonksiyonları
ekleyelim.

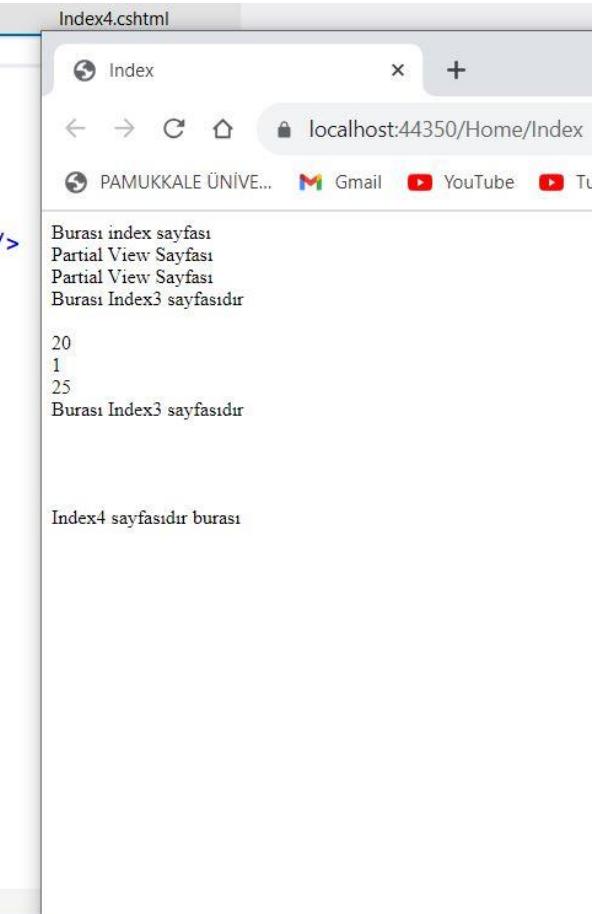
```
public ActionResult Index4()
{
    return View();
}

0 başvuru
public int Topla()
{
    return 5 + 15;
}

0 başvuru
public int Fark()
{
    return 5 - 4;
}

0 başvuru
public int carp()
{
    return 5 * 5;
}
```

Yanda
gösterilen
çağrımları
yapalım.
Dr. Öğr.



The screenshot shows a development environment with multiple tabs open. The active tab is 'Index.cshtml'. The code in this tab includes a partial view inclusion and several action renderings. To the right, a browser window displays the resulting page at 'localhost:44350/Home/Index'. The browser's status bar shows the URL. The page content includes a header with meta tags and a title, followed by a large 'div' section containing the results of the code execution: 'Burası index sayfası', a partial view inclusion, action renderings for 'Topla', 'Fark', and 'carp', and final action renderings for 'Index3' and 'Index4'. The browser's address bar also shows the URL.

```
 _PartialPage1.cshtml Index.cshtml ✘ × HomeController.cs Index3.cshtml Index4.cshtml
@{
    Layout = null;
}
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
</head>
<body>
    <div>
        Burası index sayfası
        @Html.Partial("_PartialPage1")
        @{Html.RenderPartial("_PartialPage1");}

        @{Html.RenderAction("Index3", "Home");}
        <br />
        @Html.Action("Topla")<br />
        @Html.Action("Fark")<br />
        @{Html.RenderAction("carp");}
        <br />
        @Html.Action("Index3")<br />
        @Html.Action("Index4", "Home")<br />
    </div>
</body>
</html>
```

Index

localhost:44350/Home/Index

PAMUKKALE ÜNİVE... Gmail YouTube Tu

Burası index sayfası

Partial View Sayfası

Partial View Sayfası

Burası Index3 sayfasıdır

20

1

25

Burası Index3 sayfasıdır

Index4 sayfasıdır burası

Html.Partial

Html.RenderPartial

Html.RenderAction

Html.Action

Yöntemlerine ilişkin tüm ayrıntılar dersin
akışında anlatılacaktır...

KAYNAKÇA

<https://learn.microsoft.com/tr-tr/aspnet/mvc/>

Veysel Uğur Kızmaz, ASP.NET MVC5, Kodlab Yayınları

<https://github.com/muratbaseren/udemy-yazilimcilarin-yukselisi-aspnet-my-evernote-sample>