

ASP.NET MVC İLE YAZILIM GELİŞTİRME

Dr. Öğretim Üyesi Fatma AKALIN

.NET FRAMEWORK NEDİR?

Microsoft kendi ürettiği yazılım geliştirme teknolojilerini tek çatı altında ve belirli standartlar çerçevesinde toplamak amacıyla .net platformunu (Framework)oluşturmuştur.

Yazılım geliştirme teknolojilerinin başlıcaları web uygulamaları masaüstü uygulamaları ve mobil uygulamalardır. .Net platformunun çıkış amaçlarından biri tek programlama dili ile yukarıda saydığım uygulamaların tamamını geliştirebilmektir. Farklı bir deyişle yazılımcıları her platform için farklı programlama dili öğrenmesi zorunluluğundan kurtarmaktır. Bu sayede hem dilin popülaritesi arttırılmış hem de yazılımların yazılımcıların kullanımlarını kolaylaştırmıştır.

ASP.NET MVC Nedir?

MVC yazılım geliřtirmede kullanılan mimari bir desendir. Büyük aplı projelerde kullanılan bu desen sayesinde proje farklı katmanlara ayrılarak hem geliřtirilmesi hem de yönetiminin kolaylařtırılması amaçlanır.

Özetle bir sorun vardır ve soruna karşı bir kalıp oluştururuz. Bu kalıp bir çözümü temsil eder ve bu kalıplara design pattern ismini veririz.

MVC de aslında mimarisel bir design patterndir.

Mimarisel kavramı yapıyı mantıklı kurallar dizisine dönüştürerek o kuralın dışına çıkılmamasını sağlar. Böylece sınırlamalar ile projenin daha sağlıklı büyümesini sağlar.

NOT

MVC deseni, Asp.Net ile ortaya çıkan bir desen değildir. 1979 yılında Trygve Reenskaug tarafından oluşturulmuş ve o günden itibaren PHP ve Java gibi birçok dilde kullanılmıştır. Microsoftta bunu 2009 yılında kendisine dahil etmiştir. Bizim de ASP.NET ile kullanmamızı sağlamıştır. Buna da ASP.NET MVC ismini vermiştir. Özetle MVC microsoft tarafından bulunmadı ve yeni bir teknoloji değil.

Microsoft, MVC ASP.Net'e 2009 yılında entegre etmiştir.

MVC PATTERN

Az önce bahsettiğimiz üzere MVC, büyük çaplı projelerde projenin geliştirilmesi ve yönetiminin kolaylaştırılması amacıyla ortaya çıkarılan bir desendir. Bu desen sayesinde 3 farklı yapıya sahip oluruz.

1-Model

2-View

3-Controller

Model, View ve Controller kelimelerinin baş harflerini alarak isimlendirilen bu mimari tasarım, tanımlanan klasörlerin dışına bu yapıların konulmamasını sağlayarak bize bir kalıba ve disipline sokmaya çalışır. Neyin nerde olduğu kolayca bulmamızı sağlar. Çünkü hiyerarşik düzende her şeyin yeri bellidir. Çalışma prensibi de bu hiyerarşiye göre davranır.

MODEL

Model, proje içinde kullanılacak veri türlerini(nesneleri) tanımlayacağımız bölümdür. Model içinde verilerin doğruluk, zorunluluk kontrolleri yapıp veri türleri arasında bağlantılar kurulur. Bir örnek ile somutlaştıralım,

Online satış sitemiz var ve sitede kategori ve ürün bilgileri yer alacak. Bu bilgiler bizler tarafından oluşturulacak Kategori ve Ürün ismindeki listelerde tutulsun. **Bu nesne türleri Model içerisinde tanımlanmalıdır.**

Ek olarak ürünün fiyat ve ismini girmesini(zorunluluk kontrolü) burada zorunlu tutabiliriz ya da fiyat bilgisinin bir sayı olması gerektiğini burada tanımlayabiliriz (doğruluk kontrolü). Aynı zamanda veri türleri arasında bağlantı kurarak ürünün kategori bilgisine erişim sağlamamız mümkündür (veri türleri arasında bağlantı kurma).

VIEW

Projede kullanıcının gördüğü ve kullandığı arayüzdür. Yaptığımız projenin kodlarını sunucuya aktardıktan sonra (bu kısmı detaylıca irdedeceğimiz ilerleyen aşamalarda) kullanıcının internet tarayıcısı üzerinden sayfayı açtığında karşısına gelen sayfanın kodlamasıdır.

Örneğin, <https://sakaryaadsh.saglik.gov.tr/> sitesi MVC ile yapılmış olsun. Bu siteyi internet tarayıcısı üzerinden açtığınızda karşınıza çıkan sayfa view kısmında yazdığınız kodların kullanıcı tarafından görüldüğü kısımdır.

VIEW içerisinde kodlama yapabilmek için istemci taraflı(Html, Javascript, CSS..) ve sunucu taraflı (C#...) diller kullanılmaktadır.

CONTROLLER

MVC projelerinin merkez noktası olarak düşünebiliriz. Proje içerisindeki **model ve view arasındaki iletişimin** sağlanması ve **kullanıcı talepleri** controller üzerinden yapılır.

Örneğin <https://sakaryaadsh.saglik.gov.tr/> sitesini bir internet tarayıcısı üzerinden açmak istediğimizde sunucuda yer alan projeye giden istek, projenin Controller sınıfı tarafından alınır. Ardından istek yapılan sayfanın çalışması için gerekli kısımlar Controller sınıfı tarafından çalıştırılır.

TOPLARSAK,

Controller: Olayı kontrol eden mekanizmadır. Tüm işlemler önce controller üzerinden ve controller tarafından yönetilerek sonuca yönlendirilir .

View: Görünüm olarak nitelendirebiliriz.

Biz web programlama yapıyor isek view ‘imiz web sayfaları olacaktır.

Model: Projelerdeki nesnelerimiz olarak düşünebiliriz.

ASP.NET MVC'NİN ÇALIŞMA PRENSİBİ

Asp.Net, .Net platformunun web ayağını oluşturmaktadır. Yani .net platformu ile web yazılımı(web siteleri, web projeleri..) ASP.NET ile geliştirilmektedir.

Microsoft web teknolojilerine bakıldığında ASP.NET, Asp'nin, .Net platformuna uyarlanmış hali olarak düşünülebilir. Uzun yıllar web projelerini ASP teknolojisi üzerine geliştiren Microsoft, .Net'in gücünü de arkasına alarak ASP.Net'i ortaya çıkarmıştır. Arka planda ise .Net'in temel yapıtaşlarından olan C# veya VB.Net kullanılarak yazılım geliştirme işlemleri bir bütün olarak yapılmıştır.

Asp.Net'in çalışma mantığını incelemeden önce, web projelerinin çalışma mekanizmasını incelemekte fayda var.

Web projelerinin çalışma mekanizması:

Projenin bir web sitesi olduğunu varsayar isek, bir web sitesi içerisinde birçok web sayfası yer alır. İçerisinde çeşitli yazılar, videolar, resimler bulunann bu sayfalar, HTML tabanlı kodlardan oluşmaktadır. İnternet tarayıcısı üzerinden bir web sayfası çağırdığımızda karşımıza çıkan tasarım, bu HTML kodları sayesinde oluşmaktadır. Aynı zamanda projenin dosyaları, internet üzerinde bir sunucuda yer alacaktır.

İnternet tarayıcısı üzerinde bir web sayfası çağırma işlemi yapılırken, arka planda çağrılan web sayfasının bulunduğu sunucuya hedef sayfasının görüntülenmek istendiğine dair bir istek gönderilir. BU istek, sunucu içerisinde çalıştırılır ve sonuç olarak çağrıyı yapan kullanıcının tarayıcısında görüntülemek istediği sayfa açılır. BU işlemi bir görüntü ile tasvir edelim.

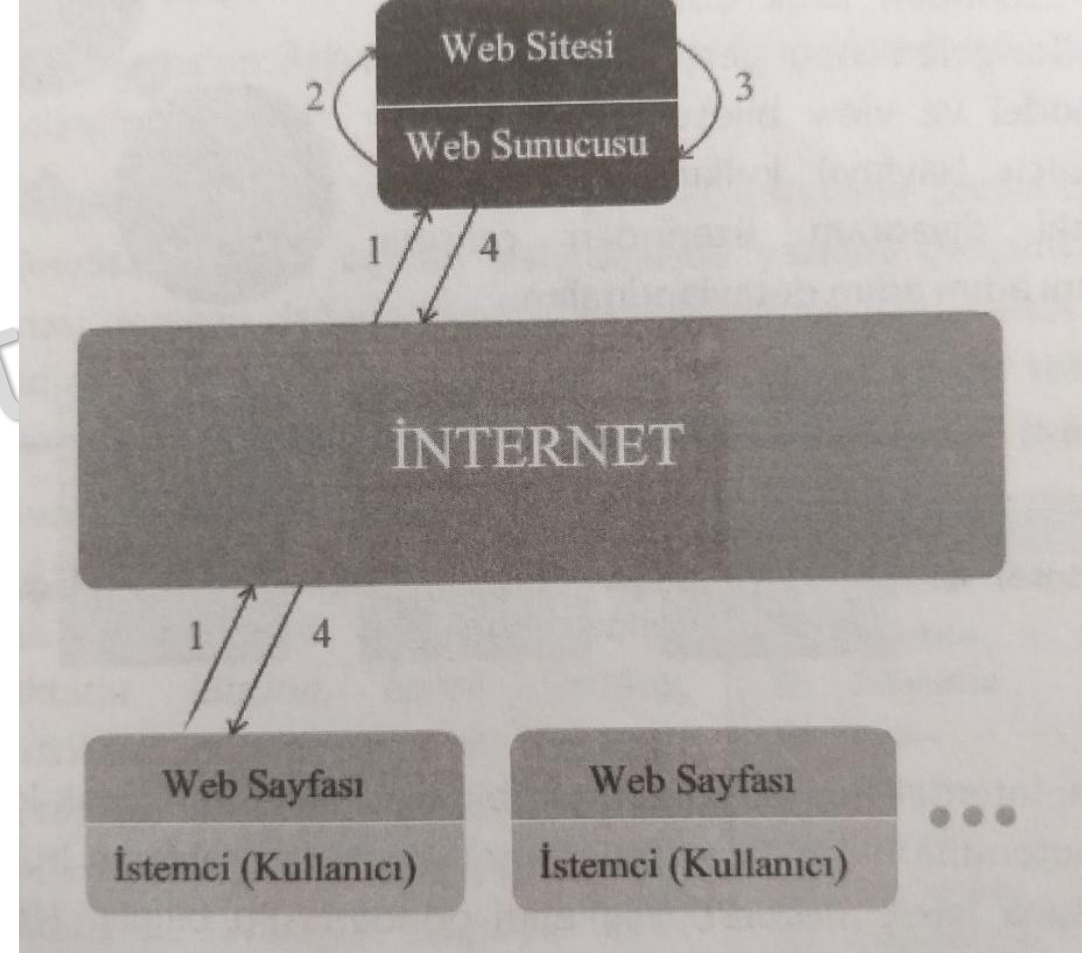
Web sunucu ya da Hosting; Web sayfalarımızın İnternet'te yayınlanması için gerekli alanın kiralanması olarak nitelendirilebilir.

1-Kullanıcı, internet tarayıcısı üzerinden bir web sayfası açmak için linki yazar ve git butonuna tıklar. Böylece Tarayıcı, internet üzerinden sayfanın bulunduğu web sunucusuna **İstek** (Request) bilgilerini gönderir.

2-Talebi alan sunucu, içerisinde bulunan web sitesine gider ve ilgili sayfayı çalıştırır, kullanıcıya gösterilecek bilgileri (web sayfası) hazırlar.

3-Sunucu, kullanıcının isteği üzerine hazırladığı web sayfasını alır.

4-Sunucu, isteği yapan kullanıcıya (istemci) web sayfasının çalıştıktan sonraki son halini **Cevap** (Response) olarak gönderir. Kullanıcı, tarayıcısında bu sayfayı görüntüler.



Anlatılan bu yapı, tüm web programlama dillerinde bezer şekilde çalışmaktadır. Aradaki farklılıklar,

1-Sunucu işletim sistemleri (Linux, Windows vb.)

2-Web sitesinin hazırlandığı programlama dilidir.

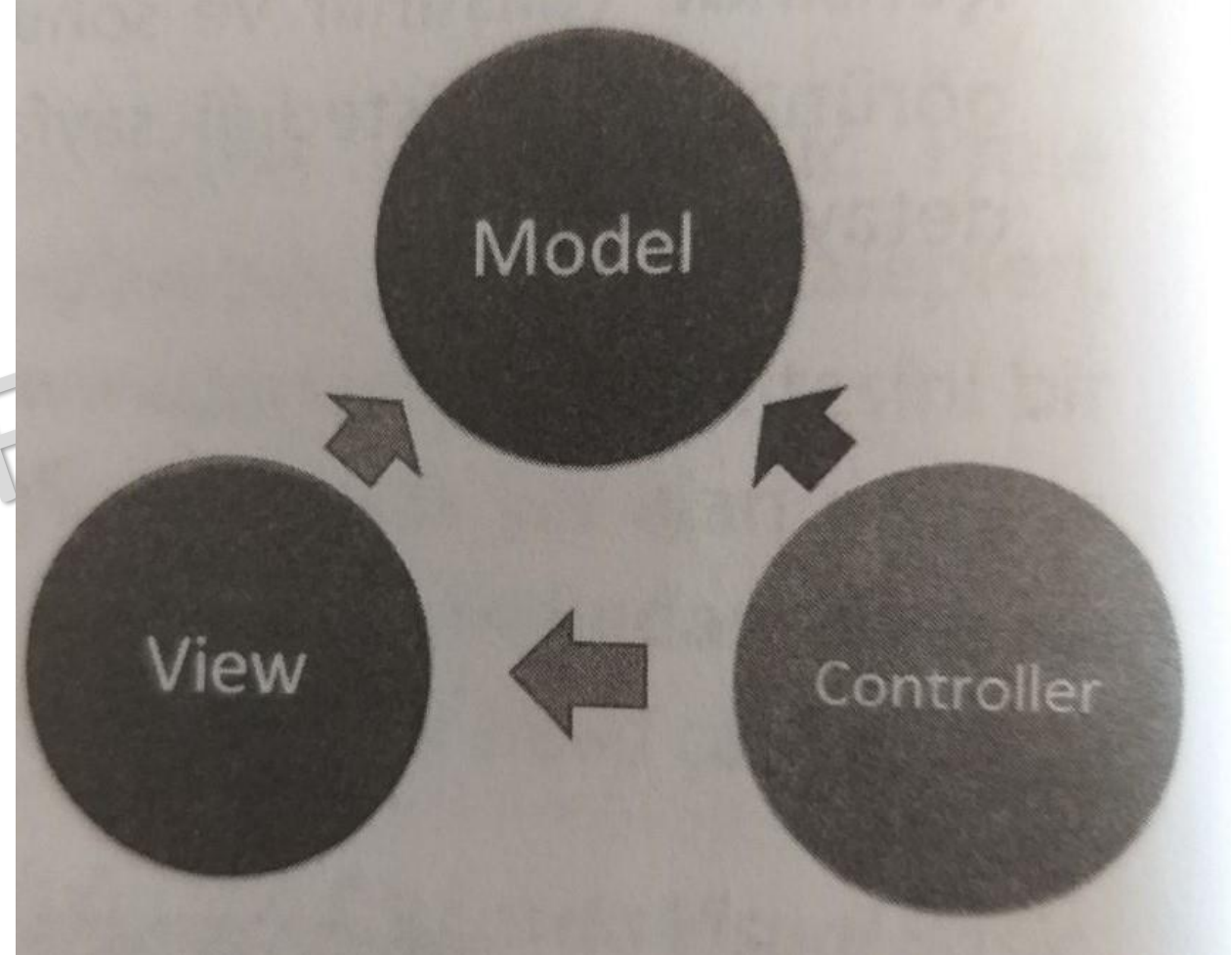
Genel hatları ile bir web uygulamasının yaşam döngüsü bu şekildedir. Peki ASP.NET'in bu döngü içerisindeki yeri ve çalışma mantığı nasıldır?

Dr. Öğr. Üyesi Fatma AKALIN

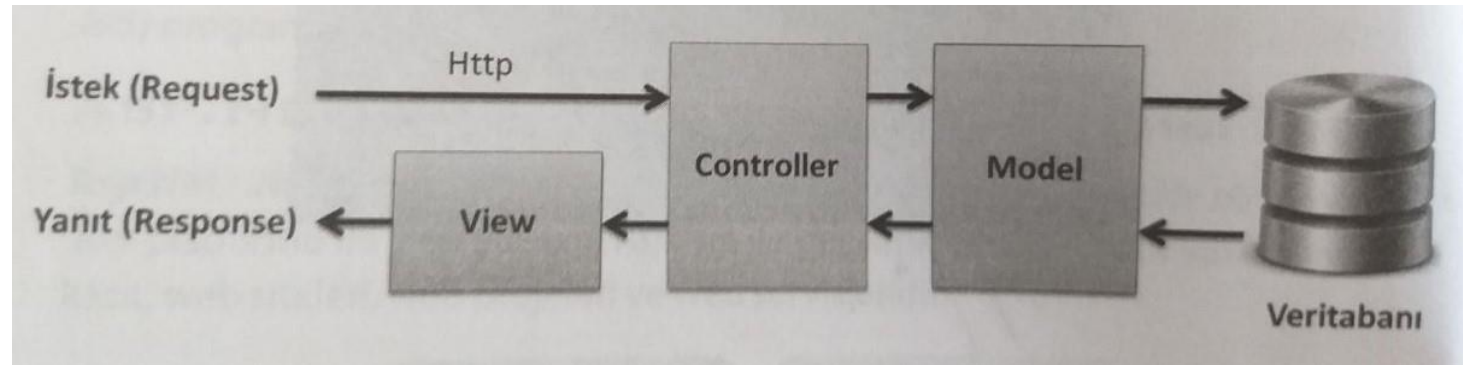
Bir MVC projesinde kullanıcı, tarayıcı üzerinden sayfayı çağırdığında Sayfa (View) üzerinden Controller'a iletilir.

Controller, gelen isteği gerçekleştirmek için ilgili model ve view bileşenlerini çalıştırır ve sonucu yani sayfayı kullanıcıya gönderir.

Bu akışın çalışma mantığını detaylandıralım...



- 1- Kullanıcı, internet tarayıcısı üzerinde bir web sayfasını açmak için linki yazar ve git butonuna tıklar. Tarayıcı, internet üzerinden sayfanın bulunduğu web sunucusuna, İstek (Request) bilgilerini gönderir. Bu bilgiler http protokolü üzerinden Controller'a gelir.
- 2-İstek Controller içerisinde incelenir ve uygun Model çağırılır.
- 3-Gerekli veritabanı işlemleri Model veya Controller tarafında gerçekleştiriliyor ve sonuç Controller'a iletiliyor.
- 4- Controller'daki veriler, View'e aktarılır. View içerisindeki kodlar gelen veriler ile çalıştırıldıktan sonra http protokolü vasıtasıyla kullanıcının http protokolü üzerinden internet tarayıcısına gönderiliyor ve sayfa tarayıcıda açılıyor.



ASP.NET MVC HAKKINDA

Temiz bir yapısı var.

Hiçbir şey kontrolümüz dışında gerçekleşmiyor.

Hazır kontrollerimiz yok.

Önümüzde bir HTML sayfa var. Ne yazarsanız o çıkıyor. İşler sayfayı çağırma ve yanıt alma mekanizmasına dönmüş durumda.

Her şey bizim kontrolümüzde.

İstedığınız kadar HTML yazdığınız için performans daha az veri ile daha hızlı projeler yazabilirsiniz.

İsteğiniz dışında olaylar fırlamıyor.

Kodlaması daha kolay.

Yeniden kullanılabilirlik var.

Fakat her şeyi siz yazmak zorundasınız ve bu süreci uzatıyor. Bazı aksiyonları almak için yazacağınız kod ve harcanan zaman daha da çoğalıyor.

ASP.NET MVC'YE BAŞLARKEN...

Microsoft.Net platformunun web ayağı olan, Asp.Net'in MVC Framework'ü ile yazılım geliştirme işlemi Visual Studio ile yapılmaktadır.

Visual Studio, Microsoft'un .Net platformunda uygulama geliştiriciler için hazırladığı bir yazılım geliştirme aracıdır. Bu araç ile .net platformunda masaüstü, mobil,web, console.. uygulama türleri geliştirilebilmektedir.

Visual Studio'nun sağladığı Kolaylıklar

Projeye hakimiyet,

Kodlama işleminde kolaylıklar

Hazır fonksiyonlar,

Veritabanına kolay erişim,

Görsel arayüz tasarımındaki kolaylıklar,

Debug modu ile yazılımın çalışma zamanında satır satır kod takibinin yapılması ,

.....

Şimdi adım adım bir proje oluşturalım...

Kullanmaya başlayın



Depoyu klonla

GitHub veya Azure DevOps gibi bir çevrimiçi depodan kod alın



Bir projeyi veya çözümü aç

Yerel bir Visual Studio projesi veya .sln dosyası açın



Yerel bir klasör aç

Herhangi bir klasör içinde koda gidip düzenleyin



Yeni bir proje oluşturun

Başlamak için kod iskelesi içeren bir proje şablonu seçin

[Kodsuz olarak devam et →](#)

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 bir proje oluřturun

Son proje řablonları

-  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++

řablon ara (Alt+S)

Tüm diller

Tüm platformlar

Tüm proje türleri



Konsol Uygulaması

Windows, Linux ve macOS'de .NET üzerinde çalışabilen bir komut satırı uygulaması oluřturma projesi

C# Linux macOS Windows Konsol



ASP.NET Core Web Uygulaması

ASP.NET Core Razor Sayfalar içerięi örneęiyle ASP.NET Core uygulaması oluřturmak için proje řablonu

C# Linux macOS Windows Bulut Hizmet Web



Blazor Server Uygulaması

Bir ASP.NET Core uygulaması içinde sunucu tarafında çalışan ve bir SignalR bağlantısı üzerinden kullanıcı etkileřimlerini işleyen bir Blazor Server uygulaması oluřturmaya yönelik proje řablonu. Bu řablon, zengin dinamik kullanıcı arabirimlerine (UI) sahip web uygulamaları için kullanılabilir.

C# Linux macOS Windows Blazor Bulut Web



ASP.NET Core Web API'si

RESTful HTTP hizmetine ait örnek bir Denetleyici içeren bir ASP.NET Core uygulaması oluřturmaya yönelik proje řablonu. Bu řablon aynı zamanda ASP.NET Core MVC Görünümleri ve Denetleyicileri için de kullanılabilir.


C# Linux macOS Windows Bulut Hizmet Web

Geri


Sonraki

Aşağıdaki fotoda MVC seçeneğini seçmeliyiz


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 arabirimi 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

FirstAspNetMvcProject.Tests

Geri

Oluştur

Yeni projenizi yapılandırın

ASP.NET Web Uygulaması (.NET Framework)

C#

Windows

Bulut

Web

Proje adı

deneme_ilkders

Konum

D:\aspnet\1_ADESLER

Çözüm adı ⓘ

deneme_ilkders

☐

Çözümü ve projeyi aynı dizine yerleştirin

Altyapı

.NET Framework 4.6.2

Proje "D:\aspnet\1_ADESLER\deneme_ilkders\deneme_ilkders\" içinde oluşturulacak

Yeni ASP.NET Web Uygulaması oluřtur



Boř

ASP.NET uygulamaları oluřturmak iin boř bir proje řablonu. Bu řablonda ierik yoktur.



Web Forms

ASP.NET Web Forms uygulamaları oluřturmak iin bir proje řablonu. ASP.NET Web Forms, bilinen srkle ve bırak yntemiyle olay denetimli modeli kullanarak dinamik web siteleri oluřturmanızı saęlar. Tasarım yzeyi ile yzlerce denetim ve bileřen, veri eriřimi olan geliřmiř, gl, kullanıcı arabirimi denetimli web sitelerini hızlı bir řekilde oluřturmanızı saęlar.



MVC

ASP.NET MVC uygulamaları oluřturmaya yarayan bir proje řablonu. ASP.NET MVC, Model-View-Controller mimarisini kullanarak uygulamalar oluřturmanıza olanak saęlar. ASP.NET MVC en son standartları kullanan uygulamalar oluřturmak iin hızlı, test gdml geliřtirmeye olanak saęlayan birok zellik ierir.



Web API

Tarayıcılar ve mobil cihazlar dahil olmak zere ok eřitli istemcilere eriřebilen RESTful HTTP hizmetleri oluřturmaya yarayan bir proje řablonudur.



Tek Sayfalı Uygulama

ASP.NET Web API kullanarak JavaScript temelli zengin istemci tarafı HTML5 uygulamaları oluřturmak zere kullanılan proje řablonudur. Tek Sayfalık Uygulamalar, HTML5, CSS3 ve JavaScript'in kullanıldıęı istemci tarafı etkileřimler ieren zengin bir kullanıcı deneyimi saęlar.

Kimlik Doęrulama

Yok

Klasrleri ve ekirdek bařvuralarını ekle

- ☐ Web Forms
- ☒ MVC
- ☐ Web API'si

Geliřmiř

- ☒ HTTPS'yi Yapılandır
- ☐ Docker desteęi
([Docker Desktop](#) gerektiriyor)

- ☐ Ayrıca birim testleri iin bir proje oluřtur

deneme_ilkders.Tests

Geri

Oluřtur

Genel Bakış

Bağlı Hizmetler

Yayımla

ASP.NET

.NET platform hakkında bilgi edinin, ilk uygulamanızı oluşturun ve buluta genişletin.



Uygulamanızı Oluşturun

ASP.NET kullanmaya başlama
.NET uygulama mimarisi



Azure'a Bağlan

Web sitenizi Azure'a
yayımlama
Azure'da ASP.NET kullanmaya
başlama



IDE'nizi öğrenin

Üretkenlik kılavuzumuza
bakın
Daha hızlı kod yazın

deneme_ilkders

Connected Services

Properties

Başvurular

App_Data

App_Start

Controllers

Models

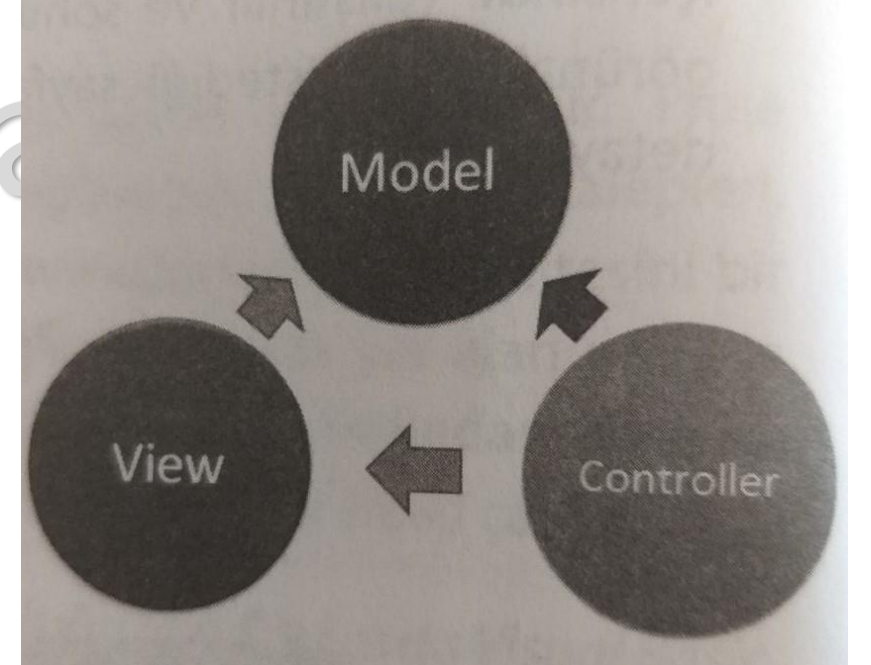
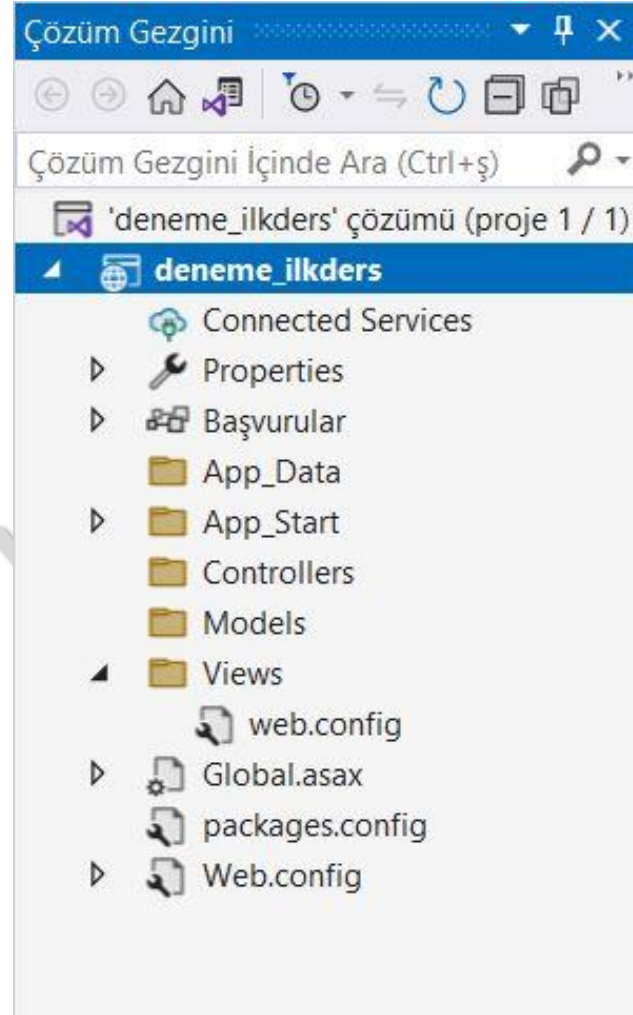
Views

Global.asax

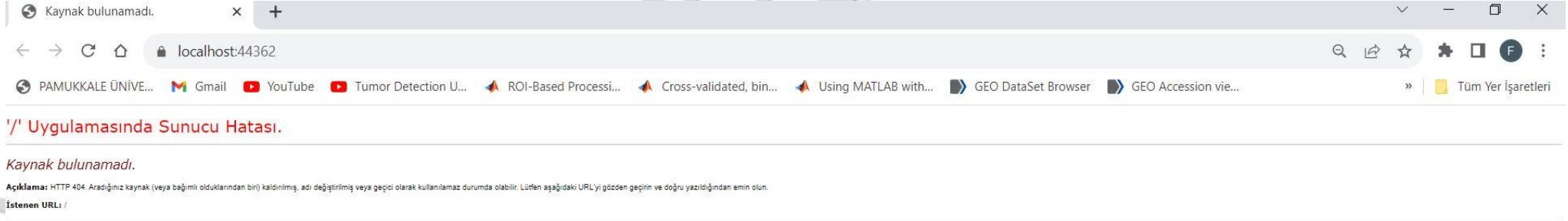
packages.config

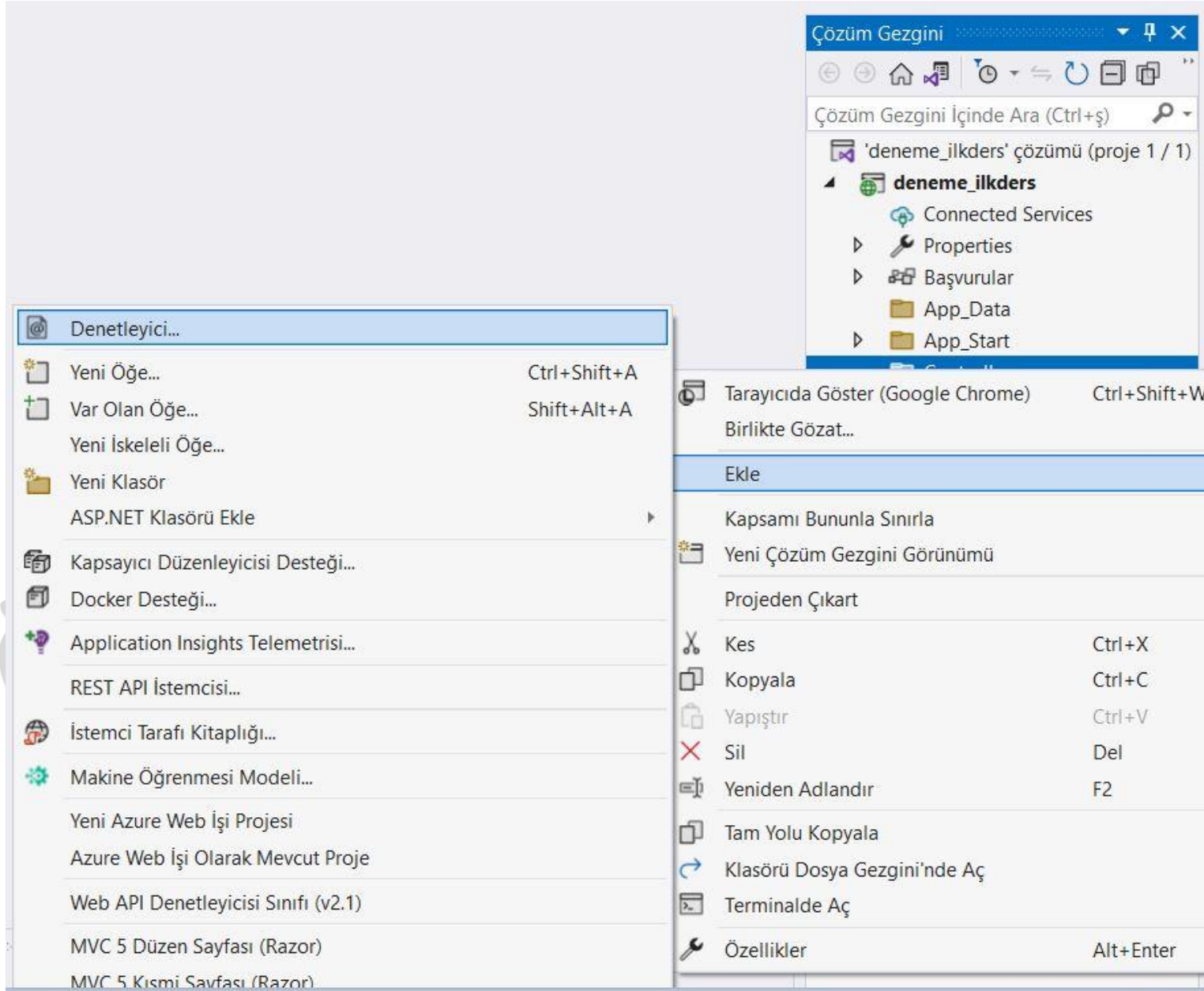
Web.config

MVC'nin bizi
kalıplara
soktuğundan
bahsetmiştim. Şimdi
o kalıpları
inceleyelim.



Bu şekilde sayfayı çalıştırdığım da aşağıdaki gibi bir hata gelecektir.
Bunun sebebi nedir?





Yeni İskeleli Öğe Ekle

Yükli

Ortak

MVC

Alan

Denetleyici

Göster

Web API'si

Entity Framework kullanan, görünümüleri 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

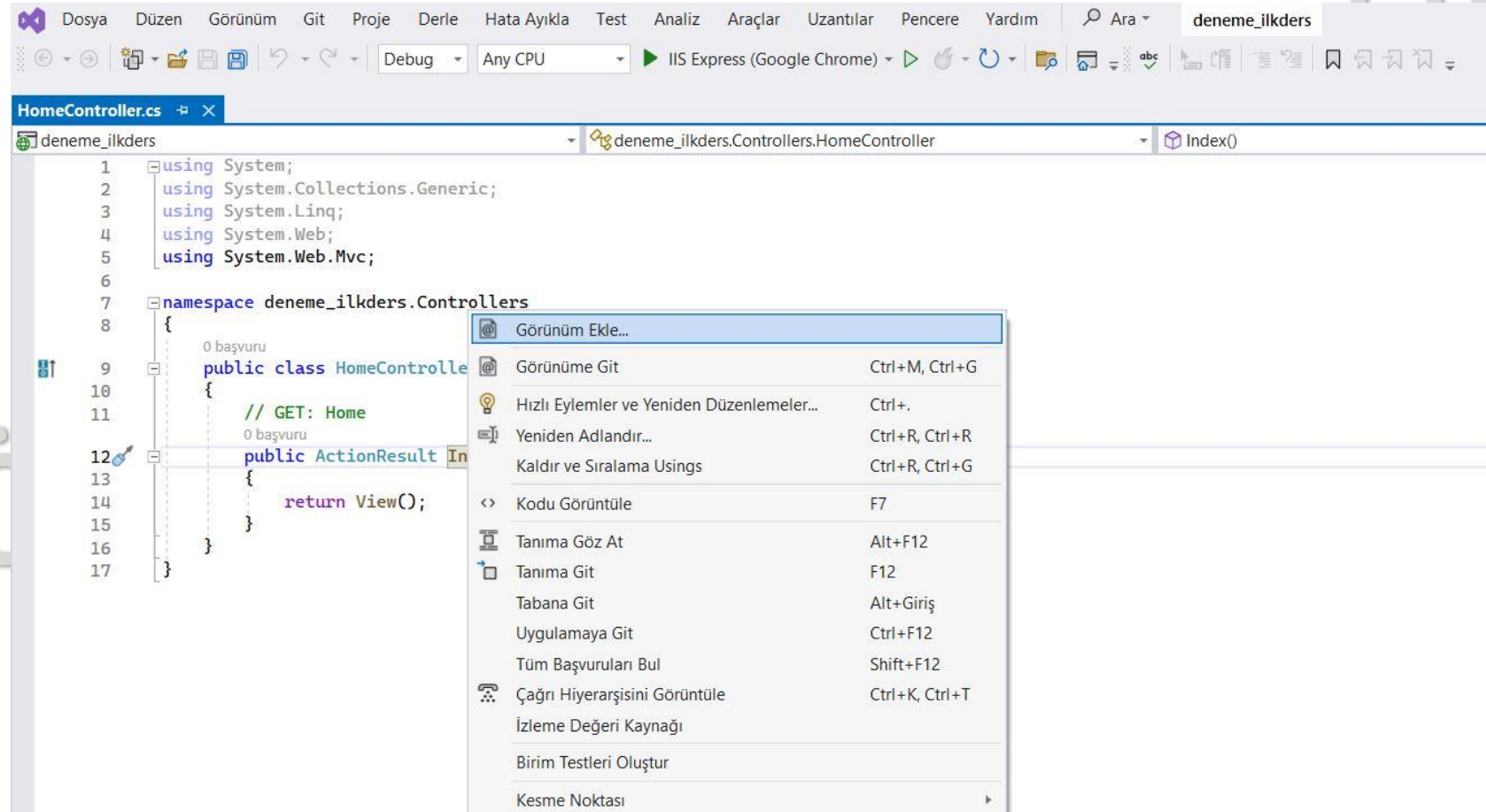
Ekleİptal

Denetleyici Ekle

Denetleyici adı: HomeController

Ekleİptal

Görünüm, bu şekilde eklenebileceği gibi sağ tarafta yer alan Çözüm Gezgini kısmından Controller ekler gibi View klasörü vasıtasıyla view'inizi ekleyebilirsiniz.



Yeni İskeleli Öğe Ekle

Yüklü

Ortak

MVC

Alan

Denetleyici

Göster

Web API'si

MVC 5 Görünümü

MVC 5 Görünümü

şunun tarafından Microsoft v5.0.0.0

Türü kesin belirtilmiş bir Modelle veya Model olmadan MVC görünümü

Kimlik: MvcViewScaffolder

Ekle

İptal

Görünüm Ekle



Görünüm adı:

Index

Şablon:

Empty (model olmadan)



Model sınıfı:



Seçenekler:

☐ Kısmi görünüm olarak oluştur

☒ Başvuru betik kitaplıkları

☐ Bir düzen sayfası kullanın:

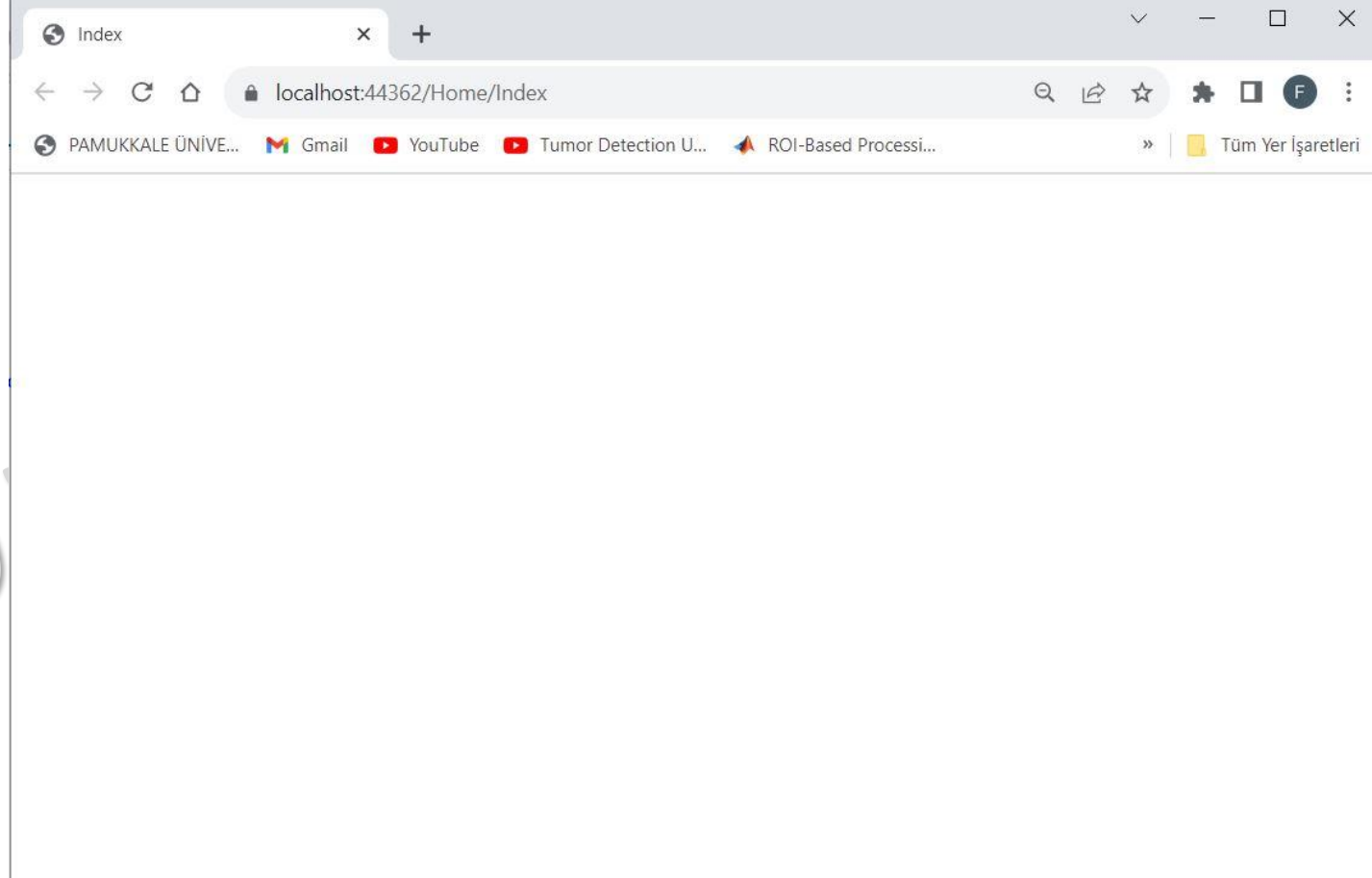


(Bir Razor _viewstart dosyasında ayarlandıysa, boş bırakın)

Ekle

İptal

Şimdi sayfamız kusursuz bir şekilde açıldı. Artık içeriğini doldurma zamanı!!!!
Fakat ilk önce kod tarafını inceleyelim..



Oluşturduğumuz henüz bir şey eklemediğimiz web sitemizde var olan Controllers, Models ve Views isminde 3 ayrı dosya yapısını inceleyelim.

Controller içerisinde yazılan metotlara actions denir. Geri dönüş tipi de action resulttir. Aslında tabir buradan gelir. ActionResult isminde bir geri dönüş tipi olduğu için bunlara metot ya da action ismi verilir.

Actionlar bizi bir sayfaya yönlendirebilir ya da yönlendirmeyebilir.

ActionResult bir sonuç türüdür.

Ozman Dersin akışında MVC'nin sonuç türlerinden irdeleyelim..

Yazdığımız her Controllerın görünüm ekranları View'ın altında ilgili controllerın ismi ile temsil edilir. Bu şekilde bir klasör oluşturulur ve bu klasörün içerisine her bir oluşturulacak ekran gelir.

Yazdığımız her metoda ait bir görünüm olabilir ya da olmayabilir. Görünüm olacaksa bunun için metoda sağ tıklayıp direk sayfadan da ADD View yapabilirsin.

View kısmında home'ın içerisine Index.cshtml sayfasını tarayıcıda açtığın zaman <https://localhost:44366/Home/Index> şeklinde bir adres karşına çıkacaktır.

MVC şöyle çalışıyor ozmn ben tarayıcıya Home/Index yazdığım zaman HomeController'a git onun Index isimli metodunu ekrana çağır. MVC'nin tam olarak anladığı budur.

Index controllera ilk önce gelir. Sonra ilgili metoda gider ve sonra da view/görünüm oluşur.

Örneğin homecontrollerın index metodunu çağırdığımızda webde GET işlemini yapıyoruz yani ilgili sayfayı getir(Indexview mesela). O sayfa form olsayda ve ben o sayfayı geri döndürüp dataları alıyor olsa idim o işlemde post işlemi diyoruz.

```
public class HomeController : Controller
{
    // GET: Home
    [HttpGet]
    public ActionResult Index()
    {
        return View();
    }

    [HttpPost]
    public ActionResult Kategoriler()
    {
        return View();
    }
}
```

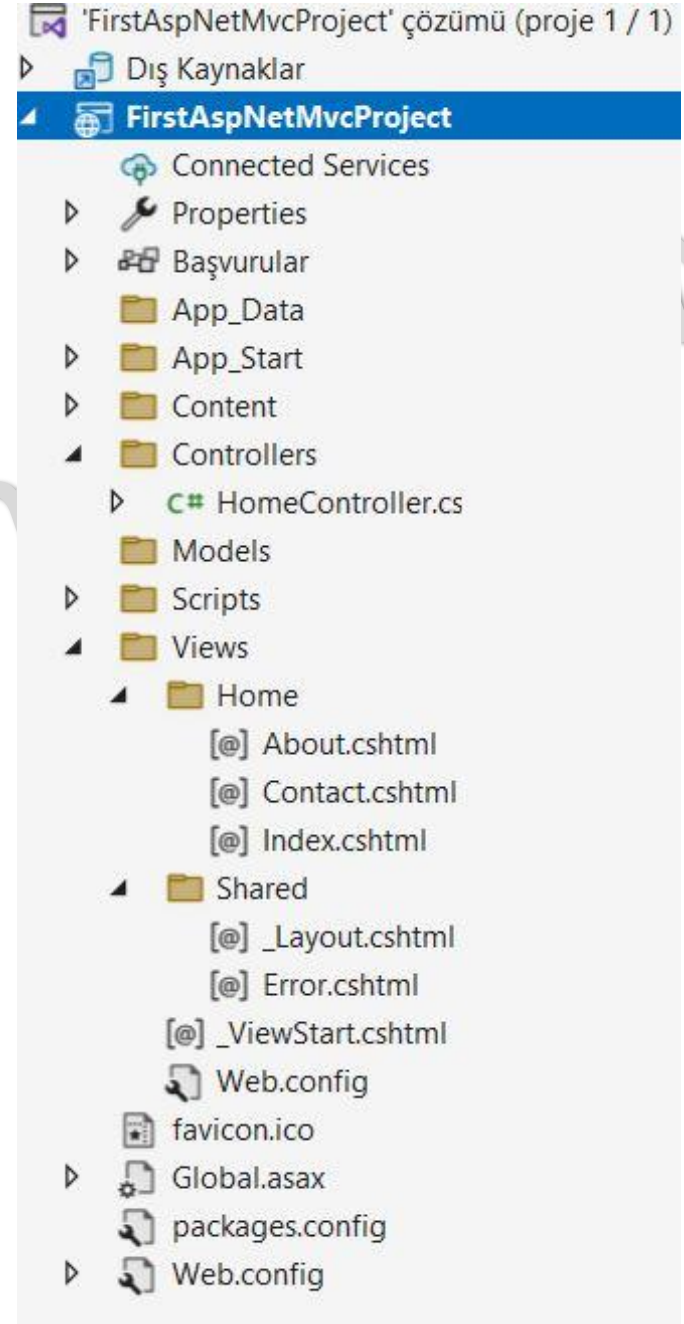
Controllerlerde get işlemi olduğunu ifade etmek için metodun üzerine bir şey yazmasanda olur ancak postta yazmalısınız.

Dersin akışında Get ve Post eylemlerini irdeleyelim

Ek olarak dersin başında söylediğim şu ifadeyi de ispatlamış oluyorum.

MVC bizi kalıplara sokuyor demiştik. Sağ görüntüyü incelediğimizde gerçekten bir kalıp olarak neyin nerede tutulduğunu ya da nereye ne yazılması gerektiğini bilerek hata yapma olasılığımızı indirgiyoruz.

Dersin akışında sağ tarafta yer alan tüm dosya yapıları irdelenecek ve detaylandırılacaktır.



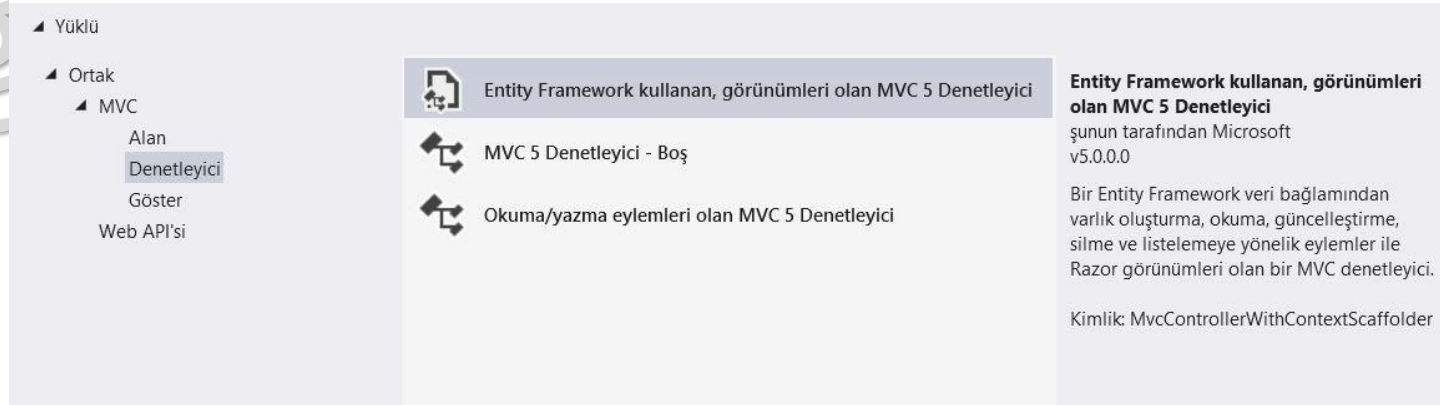
Boşluk kalmaması açısından aşağıdaki kısmı da unutmadan ifade edelim. Controller eklerken karşımıza çıkan diğer seçeneklerin anlamını öğrenelim.

Entity Framework kullanan görünümleri olan MVC5 Denetleyici: Belirtilen/Önceden tanımlanan bir modelin Entity Framework üzerinden veri tabanı işlemleri yapmasını sağlayan Controller'ı ve View'ları oluşturur.

MVC 5 Denetleyici – Boş: Boş bir MVC5 Controllerı oluşturur.

Okuma/yazma eylemleri olan MVC5 Denetleyici: Form Collection modeli üzerinden kayıt getirme(Read/Details), ekleme (Create), silme (Delete), Düzenleme (Update/Edit) CRUD işlemlerinin metodları hazır olan bir Controller sınıfı oluşturulur

Yeni İskeleli Öğe Ekle



Buraya kadar başarılı bir şekilde MVC yapımızı oluşturduk. Şimdi web sitemizin içeriğini oluşturmadan önce aşağıda sunulan başlıklar hakkında dersimizin akışında yeni bilgiler öğreneceğiz ve mevcut bilgilerimizi tazeleyeceğiz.

1-MVC Mimarisini anlayalım

2-MVC ve ASP.NET MVC arasındaki farkları açıklayalım.

3-MVC’de Model, Controller ve View bileşenlerini açıklayalım.

4-MVC’nin faydalarını konuşalım

5-MVC’nin avantajlarına değinelim.

6-MVC’nin hayat döngüsü hakkında bilgi edinelim

7-ASP.NET’in özelliklerini açıklayalım

8-MVC bileşenlerinin rollerini anlatalım.

ASP.NET MVC ile C# KULLANIMI

C# Nedir?

C#, microsoftun oluşturduğu ve .Net platformu ile yazılım hayatına başlayan bir programlama dilidir. Zamanlama olarak C ve C++'tan sonra olduğu için C ve C++'ın güçlü yapısı üzerine kurulmuştur.

Dr. Öğr. Üyesi Fatma AKALIN

ASP.NET MVC İLE PROJE GELİŞTİRİLİRKEN KULLANILAN DİLLER

Bir ASP.NET MVC projesinde sunucu taraflı kodlama (Controller ve Model) C# ile yapılabilir. Bu derste tüm ASP.NET MVC projeleri C# ile kodlanacaktır.

Şimdi C#'ın Asp.Net MVC ile kullanımını inceleyelim. Çünkü bir ASP.NET MVC projesinde birden fazla dil ile kodlama yapılabilir.

ASP.NET MVC ile HTML Kullanımı

ASP.NET MVC, bir web teknolojisi olduğu için ASP.NET MVC ile oluşturulacak viewlarda web arayüzü olan kontrollerin tasarımı HTML ile yapılmaktadır.

Bu sebeple tasarımın, formların oluşturulması ve yönetimi noktalarında HTML kullanılmaktadır.

Dr. Öğr. Üyesi Fatma AKALIN

ASP.NET MVC ile JAVASCRIPT Kullanımı

ASP.NET MVC View'ı üzerinde istemci tarafında çalışması istenen işlemler (kullanıcının internet tarayıcısında sayfayı görüntülerken tarayıcı tarafında çalıştırılabileceği işlemler /kayan yazı,resimli haber geçişi vb.) Javascript programlama dili ile kodlanmaktadır.

Dr. Öğr. Üyesi Fatma AKALIN

ASP.NET MVC İLE C# Kullanımı

Asp.Net MVC üzerinde tasarımların HTML ile kodlandığını, istemci tarafında çalıştırılacak işlemlerin ise Javascript programlama dili ile kodlandığını ifade ettik. Bununla birlikte ASP.NET MVC projesi üzerinde kullanılan üçüncü ve en önemli programlama dili ise C#'tır.

C# ile sunucu tarafında yapılabilecek işlemler kodlanır. Bu doğrultuda birkaç örnek verelim.

- Veritabanından istenen verilerin sorgulanması
- Yapacağınız web projesinde üye kaydının alınması
- Üye kaydı sırasında şifrenin MD5 ile şifrelenerek kaydedilmesi
- Üye kaydı sırasında kullanıcı adı başka bir üye tarafından kullanılıyorsa uyarı verilmesi
- Kategorilerin listelenmesi
- Site üzerinde yeni bir kategori oluşturma
- Hedef kategoriye yeni bir içerik ekleme
- İçeriğe resim ekleme vb....

Bu işlemlerin bir kısmı ver tabanı üzerinden bilgi kaydetme, düzenleme ve sorgulama işlemleri iken bir kısmı .Net framework kütüphaneleri ile yapılan işlemlerdir.

ASP.NET üzerinde C# kodlaması Controller ve Model'in yanı sıra View üzerinde Inline kodlama ile de yapılabilmektedir.

Inline Kodlama: HTML kodları içerisinde C# kodlarının yazılarak kodlanmasıdır. ASP.NET MVC projelerinde View'ın sunucu taraflı kodlamasında kullanılmaktadır.

Dr. Öğr.

ASP.NET MVC KODLAMA YÖNTEMLERİ

Şimdiye kadar Asp.Net MVC'de HTML, C# ve Javascript kullanarak kodlama yapacağımızı öğrendik. Şimdi bu kodlamayı nasıl ve hangi standartlara göre yapacağımızı öğreneceğiz. İlk olarak View Engine kavramını ve ASP.NET MVC'de kullanılan View Engine'lerin neler olduğunu öğrenelim.

Dr. Öğr. Üyesi Fatma AKALIN

VIEW ENGINE NEDİR?

View Engine, ASP.NET MVC'de View'da yazılan kodlar sonucunda oluşan çıktıyı HTML olarak sunmak (render etmek) için kullanılan teknolojidir.

Önceki sayfalarda View içerisinde C# ve HTML kodlarının bir arada yazılabildiğini ifade etmiştik. C# sunucu tarafında ve HTML ise istemci tarafında çalışmaktadır. Sunucu tarafında çalışan kodlar, kullanıcının (istemci-sayfayı çağıran kişi) göremeyeceği, sayfaya yapılan istek doğrultusunda çalışan ve sonuç üreten HTML kodlarıdır. BU iki kodun çalışması için belirlenen standartlar vardır. ASP.NET MVC'de kullanılan View Engine'ler:

-Razor, -Web Form, -Spark, ve Nhaml'dır.

Bu derste Razor View Engine ile çalışacağız. Bu nedenle Razor View Engine'nin nasıl kullanıldığını detaylı öğreneceğiz.

RAZOR VIEW ENGINE

Razor View Engine, ASP.NET sayfalarının kodlanmasında kullanılan bir sözdizimidir.

Razor kodlaması, View'lar içerisine yazılmaktadır. Client tarafında çalışan HTML, CSS ve Javascript kodları ile iç içe yazılırlar ve sunucu tarafında da çalışıp sonucu istemciye gönderirler. Örneğin bir alışveriş sitesinde gezerken herhangi bir ürünün detay bilgilerine bakmak için resmine tıklayıp ürünün detay sayfasını açığımızda karşınıza ürünün tüm bilgileri gelmektedir. Farklı bir ürüne baktığımızda ise aynı sayfanın baktığımız ürüne ilişkin yazılarının ve resimlerinin geldiğini görürüz. Aşağıdaki link vasıtasıyla süreci somutlaştırabilirsiniz.

<https://www.hepsiburada.com/ara?q=laptop>

Bu durum řu anlama gelir. Sayfanın HTML(View) tasarımı sabit(HTML,JavaScript ve CSS ile hazırlanan kısım),resim ve yazılarının olduđu kısım(deđiřken) ise sunucu taraflı (C#) kodlanmıřtır. Ürün detayı için proje içerisinde 1 tane view oluşturulmuř, üründen ürüne deđiřen bölümleri belirlenmiř belirlenen yerler sunucu taraflı kodlanmıř(C# & Razor View Engine ile) ve hangi ürünün bilgileri isteniyorsa veritabanından o ürünün bilgileri alınıp View içerisinde(sunucu tarafında) yerlerine yerleřtirilip kullanıcıya sunulmaktadır.

Dersin akıřında Razor View Engine'nin nasıl kullanıldıđı ve püf noktalarını örneklerle detaylıca inceleyeceđiz.

@ Operatörü

Razor View engine'nin Yazım kurallarının temelinde @ sembolü bulunmaktadır. HTML kodlarının arasında yazılan sunucu taraflı kodlamada kullanılan @sembolü ile okunabilirlik iyileştirilir.

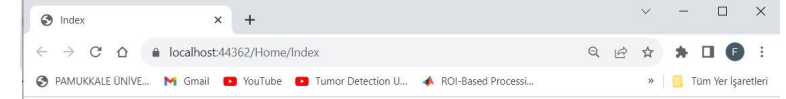
Dr. Öğr. Üyesi Fatma AKALIN

Dersin başında oluşturduğumuz projemizden devam edelim..

En son controller ve view oluşturmuştuk ve bu dosyaların içerisi aşağıda sunulduğu gibi idi. Ek olarak çalıştırıldığında herhangi bir ekran çıktısı gelmediğini deneyimlemiştik. Şimdi Bu ekranı öğrendiğimiz Razor View Engine yapısı ile dolduracağız.

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

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



Örneğin boş bir MVC projesi oluşturalım. Ardından Controller ve View ekleyelim. View altında yer alan Index.cshtml sayfasında aşağıda verilen kod bloğu yer almaktadır.

```
@{  
    ViewBag.Title = "Home Page";  
}
```

RazorSyntaxı HTML içerisinde C# kodlarını kullanmak istediğimizde kullanabileceğimiz bir syntax. Hızlı kod yazmamızı ve C#'ın gücünü arkamıza almamızı sağlayacaktır. @ işaretinden sonra süslü parantez ile bir razor bloğu açılabilir. BU blok içerisinde artık özgürsünüz.

```
@{  
    string ad = "Fatma";  
    string soyad = "Akalin";  
    string tamisim = ad + " " + soyad;  
}
```

```
@ad  
<br/>  
@soyad  
<br />  
@tamisim
```

Akabinde yandaki gibi bazı değişkenler tanımlayabilirsiniz. Bu şekilde C# kodları yazıp bunları sayfada kullanmak isteyebilirsiniz. **Bunlar HTML'e yansımaz tamamen server tarafında çalışır.** Ardından HTML kısmında @ işareti koyarak bu değişkenlere de ulaşabilirsiniz. Çünkü razor sayfanızı oluştururken bu sayfadan bir HTML çıkaracaktır.

Bize dönen her zaman HTML'dir. C# kodları kullanıcıya gitmez. Buradaki kodların hangisinin HTML hangisinin C# olduğunu anlamamız için @ işaretini kullanıyor olacağız.

Örnek 2

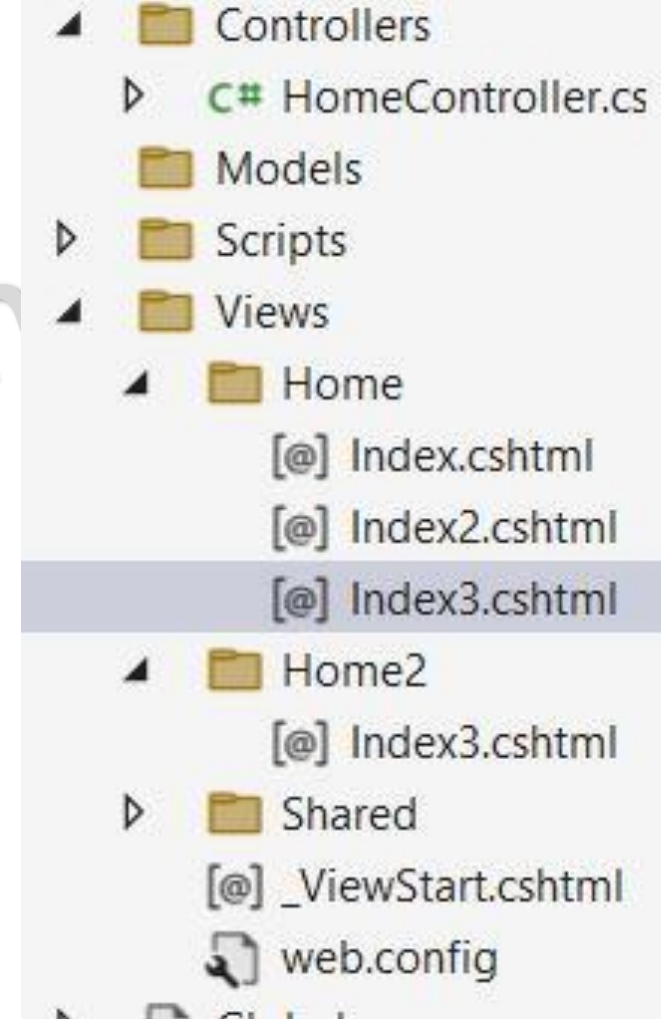
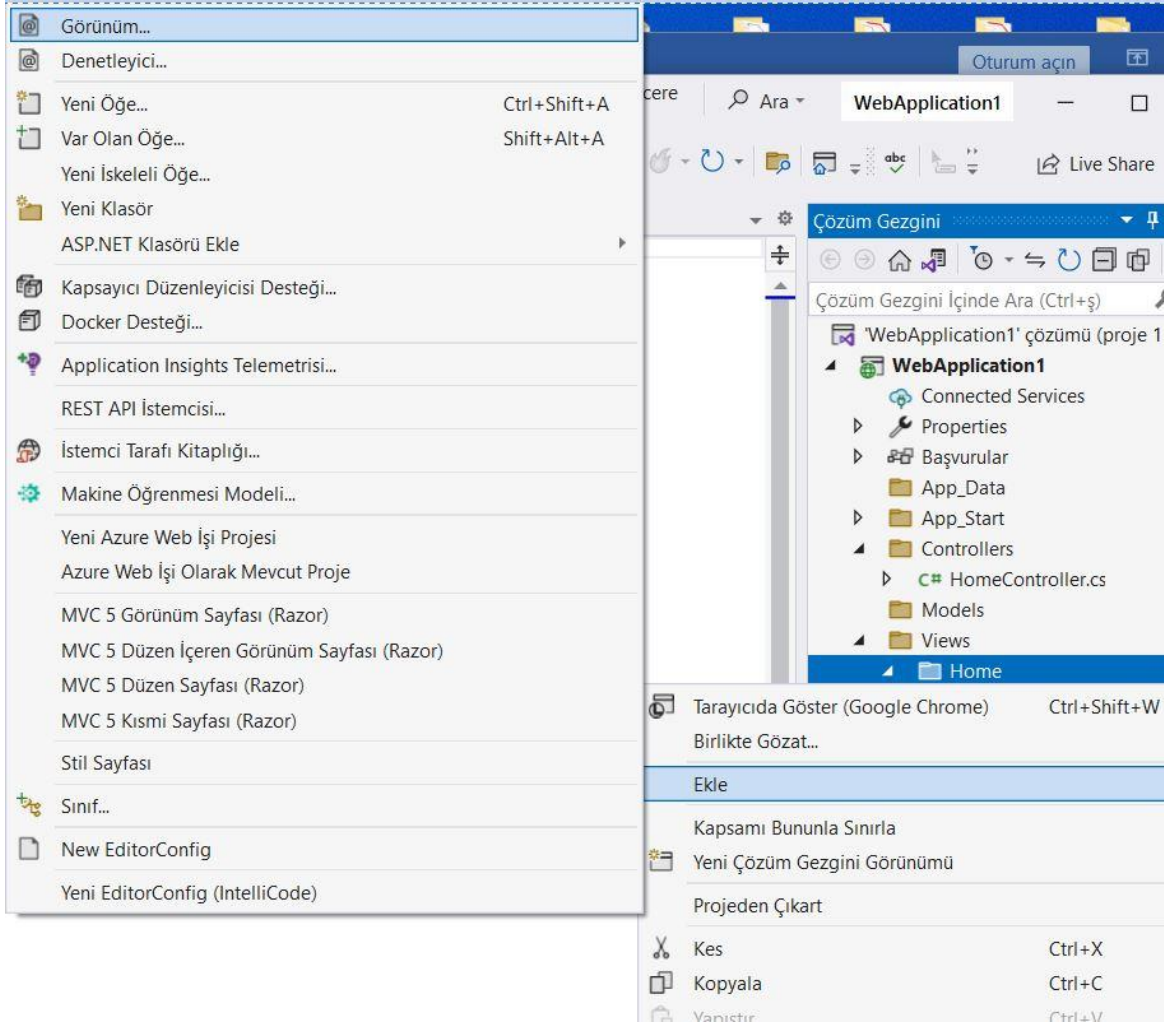
```
@{  
    string ad = "Fatma";  
    string soyad = "Akalin";  
    string tamisim = ad + " " + soyad;  
}
```

```
@{  
    string ad1 = "x";  
    string soyad1 = "y";  
    string tamisim1 = ad1 + " " + soyad1;  
    <ul>  
        <li>@ad1</li><li>@soyad1</li><li>@tamisim1</li>  
    </ul>  
}
```

```
@ad  
<br />  
@soyad  
<br />  
<b>@tamisim</b>
```

Dr. Öğr. Üye

Bununla birlikte aynı controller içerisinde birden fazla view oluşturabilirim.



0 başvuru

```
public class HomeController : Controller
{
    // GET: Home
    0 başvuru
    public ActionResult Index()
    {
        return View();
    }
    0 başvuru
    public ActionResult Index2()
    {
        return View();
    }
    0 başvuru
    public ActionResult Index3()
    {
        return View();
    }
}
```

AKALIN

Dr. Öğr

Index2 ve Index3 sayfalarını da oluşturduktan sonra artık bu sayfalarda da Razor Syntax kullanabilirim.



The image shows a code editor window with two tabs: 'Index2.cshtml' and 'Index.cshtml'. The 'Index.cshtml' tab is active. The code is as follows:

```
1
2   @{
3       ViewBag.Title = "Index";
4   }
5
6   <h2>Index</h2>
7
8
9
```

The code is written in a light-themed editor with syntax highlighting. The '@{' and '}' characters are highlighted in yellow. The 'ViewBag.Title = "Index";' line is highlighted in light gray. The '<h2>Index</h2>' line is highlighted in light blue. The line numbers 1 through 9 are visible on the left side of the editor.

@{

ViewBag.Title = "Index2";

}

<h2>Index2</h2>

@{

string[] adlar = new string[3] { "pembe", "mor ", "turuncu" };

@foreach(string ad in adlar)

{

@ad

}

}

@for(int i=0; i<10; i++)

{

@("Sayi : "+i.ToString())

}

LIN

Dr. Öğr.

ASP.NET MVC'DE EN SIK KULLANILAN C# KODLARI

LİNK OLUŞTURMA

Bu bölümde ASP.NET MVC'de bir sayfadan başka bir sayfaya link verme işleminin nasıl yapıldığını öğreneceğiz.

Index1, Index2 ve Index3 şeklinde 3 ayrı view vardı. Şimdi Index1'e bir link vererek Index3'e gitmeyi öğreneceğiz.

Link verme işlemi için Html.ActionLink metodunu kullanacağız HTML
Html.ActionLink metodunun alacağı parametreler şunlardır.

1-linkText:Linkte yazacak yazıyı belirtir. Yapacağımız örnekte Index3 sayfasına gideceğimiz için Index3 yazabiliriz.

2-actionName: Linke tıklanıldığında hangi action metodunun çalıştırılacağını tanımlar. HomeController içerisinde Index3 Actionunu çalıştıracığımız için bu parametreyi Index3 olarak tanımlayabiliriz.

3-controllerName:Linke tıklandığında hangi controllerdaki action'ın çalıştırılacağını tanımlar. HomeController üzerindeki bir actionun çalışmasını istediğimiz için bu parametreyi Home olarak tanımlayalım.

```
@Html.ActionLink("Index3", "Index3", "Home")
```


Link oluşturma işlemi **HTML.ActionLink** metodunun yanı sıra **Url.Action** metodu ile de yapılabilir. Url. Action metodunda sadece linke gidilecek linkin yolu verilmektedir. Url. Action metodunun alacağı parametreler şunlardır:

1. **actionName**: Linke tıklandığında hangi action metodunun çalıştırılacağını tanımlar. HomeController içerisindeki Index Actionu çalıştıracığımız için bu parametreyi Index olarak tanımlayalım..
2. **controllerName**: Linke tıklandığında hangi controllerdaki action'un çalıştırılacağını tanımlar. HomeController üzerindeki bir actionun çalıştırılmasını istediğimiz için bu parametre Home olacaktır.

```
@Html.ActionLink("Index3", "Index3", "Home")  
<a href="@Url.Action("Index2", "Home")">Anasayfa2</a>
```

Dersin akışında örnekler yapılacaktır.

SAYFAYA HTML KODU YAZDIRMA(HTML.RAW)

Sunucu tarafında tanımladığınız String değişkeni ekrana yazdırdığınızda HTML kodlarını normal metin gibi ekrana yazdırmaktadır Aslında «ilk örnek» yazısını html'de kalın yazdırmak isterseniz metnin başına sonuna etiketlerini koymanız yeterli olacaktır. Şöyle ifade edersek, bu hedef yazıyı htmlornek1 isminde bir String değişkende tuttuğumuzu varsayalım. Bu değişkenin değerini sayfaya yazdırdığımızda ekranda ifade olduğu gibi kalın bir şekilde «**ilk örnek**» olarak yazılacaktır.

Ek olarak eşdeğer kodları sayfa içerisinde çalışacak halde yazdırabilmek için HTML.RAW metodunu kullanacağız.

Şimdi Index2 sayfasında, HTML.Raw metoduna ilişkin bir örnek yapalım.

```
<h1>Index 2</h1>
<div>
    @{
        string htmlornek1 = "<b>İlk örnek</b>";
        string htmlornek2 = "<i>İkinci örnek</i>";
        string htmlornek3 = "<ins>Ucuncu örnek</ins>";
        @Html.Raw(htmlornek1)
        @Html.Raw(htmlornek2)
        @Html.Raw(htmlornek3)
    }
</div>
```

Proje çalıştırıldığında string değişkenlerinin ekrana bold olarak yazıldığını görürüz.

UYARI MESAJI PENCERESİ OLUŞTURMA

postback asp.net mvs sayfalarında bir butona tıklandığında aynı sayfanın sunucu tarafında işlem yapıp yeniden açılması işlemidir. Örneğin sayfanızda sadece 1 adet buton olsun bu butona tıkladığınızda butonun altına Merhaba dünya yazsın.

Bu süreçte butona tıkladığınız anda sunucuya bir istek gönderilir ve sunucu tarafından bu istek yorumlanıp aynı sayfanın çalışma sonucu oluşan hali geri gönderilir ve bu işlemi postback denilir. Asp.net MVC'de bu işlem IsPost özelliğiyle kontrol edilir.

Bu doğrultuda sayfamıza 1 adet uyarı penceresi ekleyelim.

Javascript kodlarını kullanarak Index2 sayfasında oluşturduğumuz uyarı penceresi aşağıda örneklenmiştir.

```
<div>
  @{string mesaj = "";}

  if (IsPost)
  {
    mesaj = "<script lang='Javascript'>alert('Butona tıkladınız');</script>";
  }

  <form method="post">
    <input type="submit" value="Tıkla" />
    @Html.Raw(mesaj);
  </form>
}
<br />
</div>
```

Bu kod satırları ile Index2 sayfasında butona tıkladığımız vakit, Butona tıkladınız şeklinde aynı sayfada uyarı gelecektir.

Uyarı Mesajı Penceresi Sonrası Otomatik Sayfa Yönlendirme

Bir önceki örnekte butonuna tıklanıldığında ekrana butona tıklanıldı şeklinde bir uyarı geliyor idi. Şimdiki örneğimizde ise tıklanıldığı andan itibaren farklı bir sayfaya yönlendirilme işlevini gerçekleştiriyor olacağız.

```
@{string mesaj2 = "";  
  
    if (IsPost)  
    {  
        mesaj2 = "<script lang='Javascript'>alert('Sayfaya yönlendiriliyorsunuz');window.location='/Home/Index';</script>";  
    }  
    <form method="post">  
        <input type="submit" value="Tıkla2" />  
        @Html.Raw(mesaj2);  
    </form>  
}
```

Index3 sayfasında yazılan bu kod, tıklanıldığı andan itibaren Index sayfasına gitme eylemi gerçekleştirecektir.

KAYNAKÇA

<https://learn.microsoft.com/tr-tr/aspnet/mvc/overview/getting-started/introduction/getting-started>

<https://learn.microsoft.com/tr-tr/aspnet/mvc/overview/getting-started/introduction/adding-a-controller>

<https://learn.microsoft.com/tr-tr/aspnet/mvc/overview/getting-started/introduction/adding-a-view>

<https://learn.microsoft.com/tr-tr/aspnet/mvc/overview/getting-started/introduction/adding-a-model>

Veysel Uğur Kızmaz, ASP.NET MVC5, Kodlab Yayınları

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