

MODEL

Dr. Öğr. Üyesi Fatma AKALIN

Models ve ModelBinding

Su ana kadar controller, view kavramları ile bu kavamlar arasındaki etkileşimi gördük. Viewdan controllera ve controllerden viewe veri aktardık. Şimdi controllerden viewe ve viewdan controllera veri transferinde kullanabileceğimiz model kavramını inceleyeceğiz.

Dr. Öğr.

Normal şartlar altında Controllerdan view'e farklı yöntemler ile data göndermiştık. Şimdi ise model klasörünü kullanacağız. Model klasörü verilerimizi taşıyan nesneleri içeren bir klasördür. Genelde veri tabanındaki nesneleri de bunun içerisinde koyarız. Hem veritabanından çektiğimiz nesneler için hem de view'a giden nesneler için model klasörünün içerisinde tutma eylemi gösterirsek yeri belirli olacaktır.

Dr. Öğr. Üyesi

MODEL KULLANIMI

Model, MVC'de projede kullanılan veri türlerine(nesnelere) verilen isimdir. Örneğin bir online satış sitesinde birden fazla model (veri türü) bulunmaktadır. Ürün, kategori, üye, sipariş gibi veriler birer model üzerinde tutulmaktadır. Ürün modelinde (veri türünde) ürünün adı, ürünün fiyatı, özellikleri, stok durumu gibi bilgiler ürün modelinin özelliklerinde tutulmaktadır. Benzer şekilde üye modelinin (siteye üye olan kişilerin) adı, soyadı, e-posta adresi, şifresi, kimlik numarası, yaşadığı şehir bilgileri üye modelinin özelliklerinde tutulmaktadır

Dr.

Proje içinde yer alan tüm veri türleri için bir model oluşturulması gerekmektedir. Yukarıdaki örnek üzerinden devam edersek online satış sitesinde ürünlerin siteye kaydedildiği ve yönetildiği bir yönetim paneli olacaktır. Yönetim paneli üzerinden yapılan ürün ekleme, silme, ürünün bilgilerini değiştirme işlemlerinde üzerinde, çalışılan ürün bilgileri bir model üzerinden yapılmaktadır. Ürün bilgilerinin modelde tanımlanan özellikleri veritabanına kaydedilebilir (ürün ekleme). Ürün bilgilerinin modelde tanımlanan özellikleri veri tabanında kayıtlı özellikleri değiştirilebilir (ürün bilgilerini değiştirme) veya ürün yine model üzerinden silinebilir.

Dr. Öğr. Üyesi
Mehmet Yıldız

Model içinde, verilerin doğruluk zorunluluğu vb. kontrolleri de yapılır Örneğin online satış sitesinde bir ürün kaydı yapılırken ürünün adı, fiyatı, özellikleri ve stok durumu bilgileri bulunsun.

Bu bilgilerden adı fiyatı ve stok durumu bilgisinin kullanıcı tarafından bilinmesi gereğinden bu bilgilerin girişini zorunlu yapabilirsiniz. Ürün çok bilinen bir ürün ise özelliklerinin girişini isteğe bağlı yapabilirsiniz. Benzer şekilde fiyatı alanına rakamsal bir değer girilmesini engelleyebilir, sadece sayısal değer girilmesini sağlayabilirsiniz. Model içerisinde yapılacak tanımlamalar sayesinde bu kontroller kolaylıkla gerçekleştirilebilmektedir

Dr. Oğuzhan

**MODEL VE MODELBINDING DERSİN
AKIŞINDA AYRINTILI OLARAK
ANLATILACAKTIR**

Dr. Öğr. Üyesi FETİH ALIN

NELER ÖĞRENECEĞİZ?

Bu bölümde ASP.NET MVC'de model tanımlama işleminin nasıl yapıldığını, ardından modelin kayıt ekleme, silme, düzenleme, listeleme, ve kayıt görüntüleme işlemlerinde nasıl kullanılacağını öğreneceğiz.

ASP.net mvc projelerinde model dosyaları (sınıfları), Models klasörünün içerisinde yer alırlar. Model aslında bir classtır (sınıftır). Projeye bir model ekleneceği zaman Models klasörüne bir sınıf oluşturulur. Oluşturulan sınıfa istenen özellikler (properties) eklenir ve model kurulur.

Model klasörü verilerimizi taşıyan nesneleri içeren bir klasördür. Genelde veri tabanındaki nesneleri de bunun içerisine koyarız. Hem veritabanından çektiğimiz nesneler için hem de viewa giden nesneler için model klasörünün içerisinde tutma eylemi gösterirsek yeri belirli olacaktır.

İlk olarak projemizi oluşturalım. Ardından Models klasöründen class oluşturarak süreci inşa edelim.

Dr. Özgür

Yeni projenizi yapılandırın

ASP.NET Web Uygulaması (.NET Framework)

C#

Proje adı

denemeonuncuders

Konum

D:\aspnet\1_ADERSLER

Çözüm adı 

denemeonuncuders

Çözümü ve projeyi aynı dizine yerleştirin

Altyapı

.NET Framework 4.6.2

Proje "D:\aspnet\1_ADERSLER\denemeonuncuders\denemeonuncuders\" içinde

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

denemeonuncuders.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.
Boş bir MVC denetleyici.
Kimlik: MvcControllerEmptyScaffolder

Çözüm Gezgini İçinde Ara (Ctrl+Ş)

'denemeonuncuders' çözümü (proje 1)

denemeonuncuders

- Connected Services
- Properties
- Başvurular
- App_Data
- App_Start
- Controllers**
- Models
- Views
- Global.asax
- packages.config
- Web.config

Denetleyici Ekle

Denetleyici adı:

Ekle **İptal**

Dr. Öğr. Üyesi

Kisi.cs

Adres.cs

HomeController.cs



denemeonuncuders

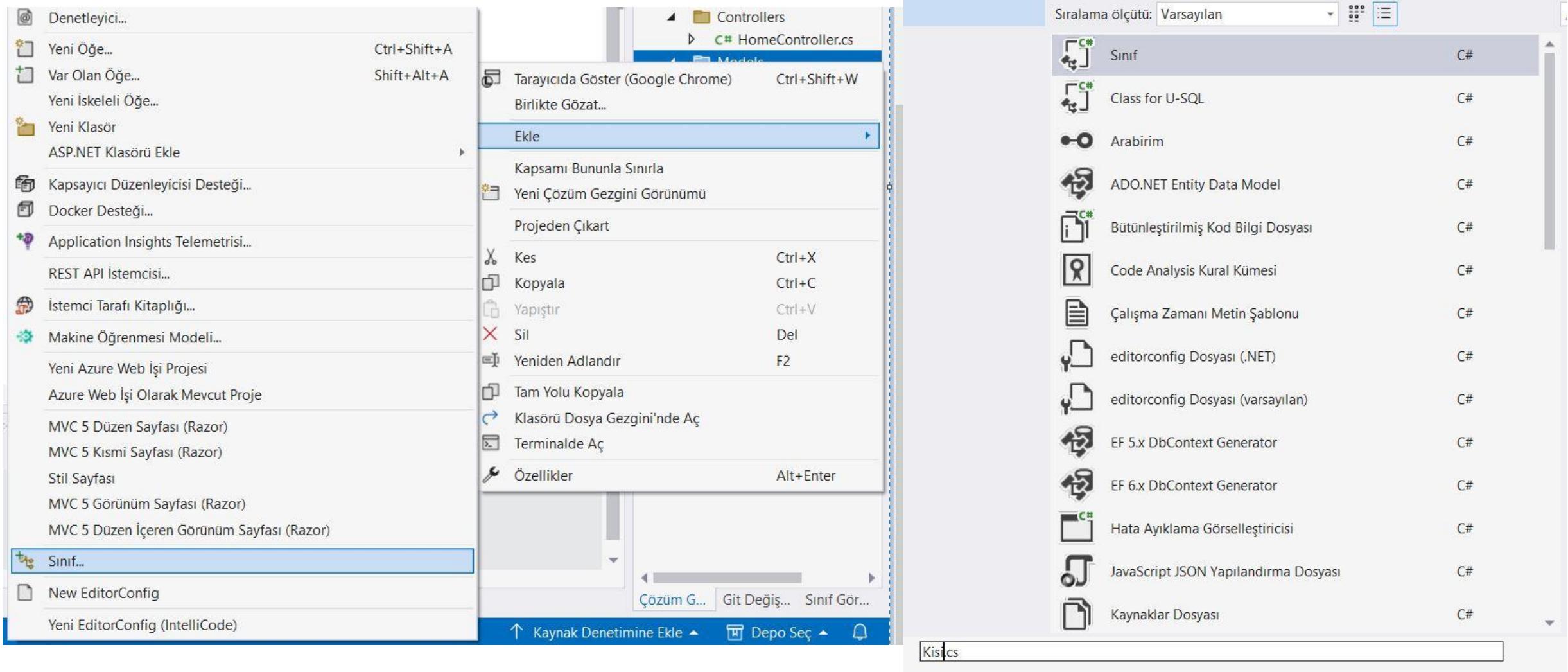
denemeonuncuders.Controllers

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

ALIN

D

Proje oluşturulduktan sonra models klasörüne bir sınıf ekleyelim..



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

- Left Sidebar:** Shows project navigation options like Denetleyici..., Yeni Öğe..., Var Olan Öğe..., Yeni İскеlevi..., Yeni Klasör, Kapsayıcı Düzenleyicisi Desteği..., Docker Desteği..., Application Insights Telemetrisi..., REST API İstemcisi..., İstemci Tarafı Kitaplığı..., Makine Öğrenmesi Modeli..., Yeni Azure Web İşi Projesi, Azure Web İşi Olarak Mevcut Proje, MVC 5 Düzen Sayfası (Razor), MVC 5 Kısmi Sayfası (Razor), Stil Sayfası, MVC 5 Görünüm Sayfası (Razor), and MVC 5 Düzen İçeren Görünüm Sayfası (Razor). The "Sınıf..." option is highlighted.
- Center Area:** A context menu is open over the "Models" folder in the Solution Explorer. The "Ekle" (Add) option is selected, and its submenu is visible, showing options like "Kapsamı Bununla Sınırla", "Yeni Çözüm Gezgini Görünümü", "Projeden Çıkart", "Kes", "Kopyala", "Yapıştır", "Sil", "Yeniden Adlandır", "Tam Yolu Kopyala", "Klasörü Dosya Gezgini'nde Aç", "Terminalde Aç", and "Özellikler".
- Right Area:** The "Add New Item" dialog is displayed. It lists various item types under the "C#" category, with "Sınıf" (Class) being the selected item. Other items listed include Class for U-SQL, Arabirim, ADO.NET Entity Data Model, Bütünleştirilmiş Kod Bilgi Dosyası, Code Analysis Kural Kümesi, Çalışma Zamanı Metin Şablonu, editorconfig Dosyası (.NET), editorconfig Dosyası (varsayılan), EF 5.x DbContext Generator, EF 6.x DbContext Generator, Hata Ayıklama Görselleştiricisi, JavaScript JSON Yapılandırma Dosyası, and Kaynaklar Dosyası.

Dr. Ö

Kisi.cs + X Adres.cs homepageController.cs

denemeonuncuders

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;
5
6  namespace denemeonuncuders.Models
7  {
8      public class Kisi
9      {
10         public string Ad { get; set; }
11         public string Soyad { get; set; }
12         public int Yas { get; set; }
13     }
14 }
15 }
```

ALIN

Model klasörümüzün altında, nesnelerimizi taşıyacak olan Kisi.cs sınıfı oluşturduktan sonra Controllerden view'e veri aktarım sürecinde bu sınıfları kullanacağımızı controllera söylememiz gerek. Bu nedenle aşağıdaki kod satırını Controller'a eklemeyi unutmayalım...

```
using denemeonuncuders.Models;
```

şeklinde modeli ekledikten sonra artık Kişi nesnesi oluşturup set edeceğiz.

Ardından set ettiğim bu nesneleri sayfaya göndereceğiz..

Dr. Öğ

Kisi.cs Adres.cs **HomeController.cs** X

denemeonuncuders denemeonuncuders.Controllers

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;
5  using System.Web.Mvc;
6  using denemeonuncuders.Models;
7
8
9  namespace denemeonuncuders.Controllers
10 {
11
12     public class HomeController : Controller
13     {
14
15         // GET: homepage
16
17         public ActionResult homepage()
18         {
19             Kisi kisi = new Kisi();
20             kisi.Ad = "Fatma";
21             kisi.Soyad = "Akalın";
22             kisi.Yas = 28;
23             return View(kisi);
24         }
25     }
26 }
```

AKALIN

Ardından view sayfasına gelen model nesnesini tanımını yapalım.

Fakat controller yardımımı ile ilk olarak view'ımızı oluşturalım.

The screenshot shows the Microsoft Visual Studio IDE interface. On the left, the code editor displays the `HomeController.cs` file:Adres.cs HomeController.cs X
denemeonuncuders
denemeonuncuders.Controllers.HomeController homepage()
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5 using System.Web.Mvc;
6 using denemeonuncuders.Models;
7
8
9 namespace denemeonuncuders.Controllers
10 {
11 public class HomeController : Controller
12 {
13 // GET: homepage
14 public ActionResult homepage()
15 {
16 Kisi kisi = new Kisi();
17 kisi.Ad = "Fatma";
18 kisi.Soyad = "Akalın";
19 kisi.Yas = 28;
20 return View(kisi);
21 }
22 }
23 }

In the center, a modal dialog titled "Yeni İskeleli Öğe Ekle" (Add New Scaffolded Item) is open. The "Ortak" (Common) category is selected under "MVC". The "Alan" (Content) section has "Denetleyici" (Controller) selected. The "Göster" (Show) button is highlighted. The "Web API'si" (Web API) section is also visible.

The "MVC 5 Görünümü" (MVC 5 View) tab is active in the dialog. The "Görünüm Ekle" (Add View) section contains the following fields:

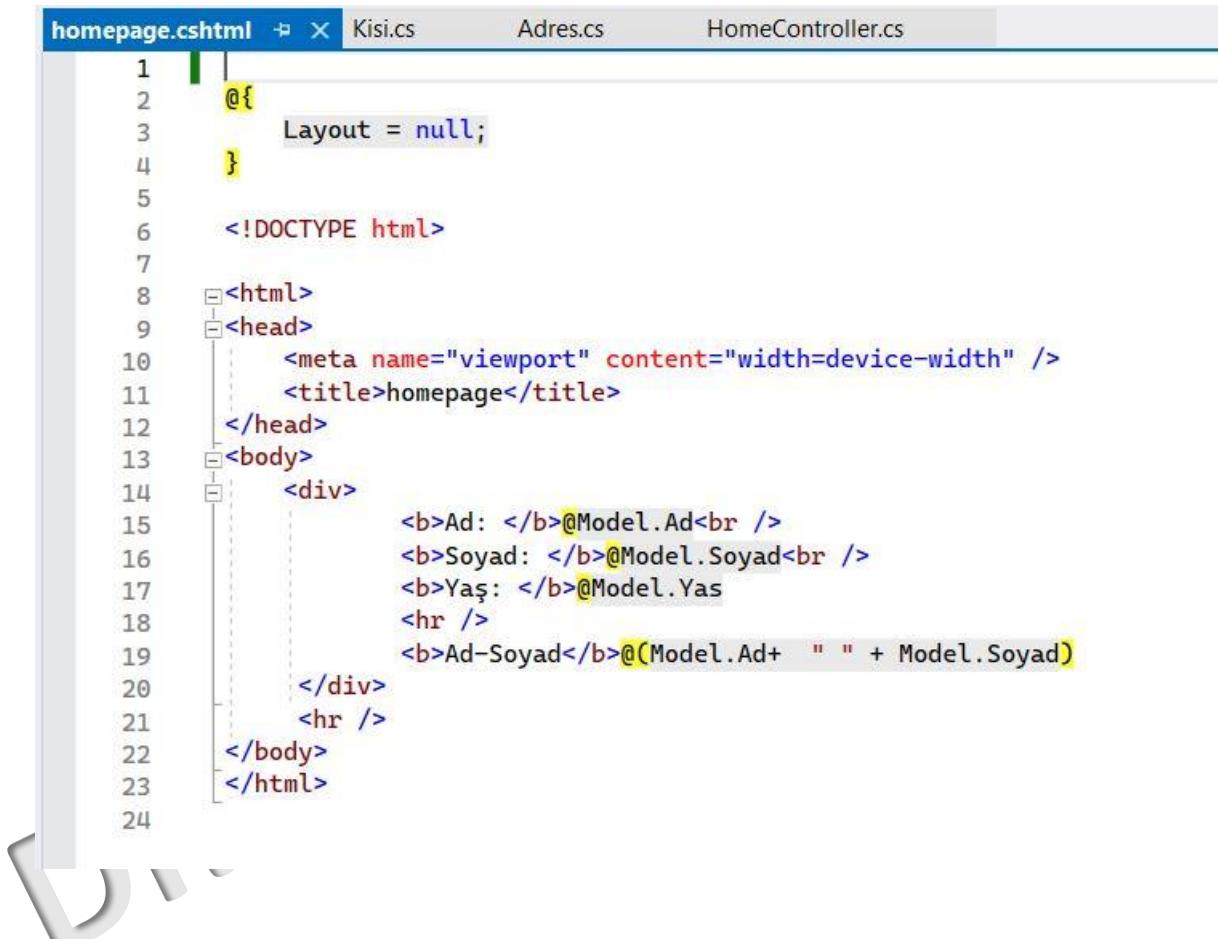
- Görünüm adı: homepage
- Şablon: Empty (model olmadan)
- Model sınıfı: (empty dropdown)

The "Seçenekler" (Options) section contains three checkboxes:

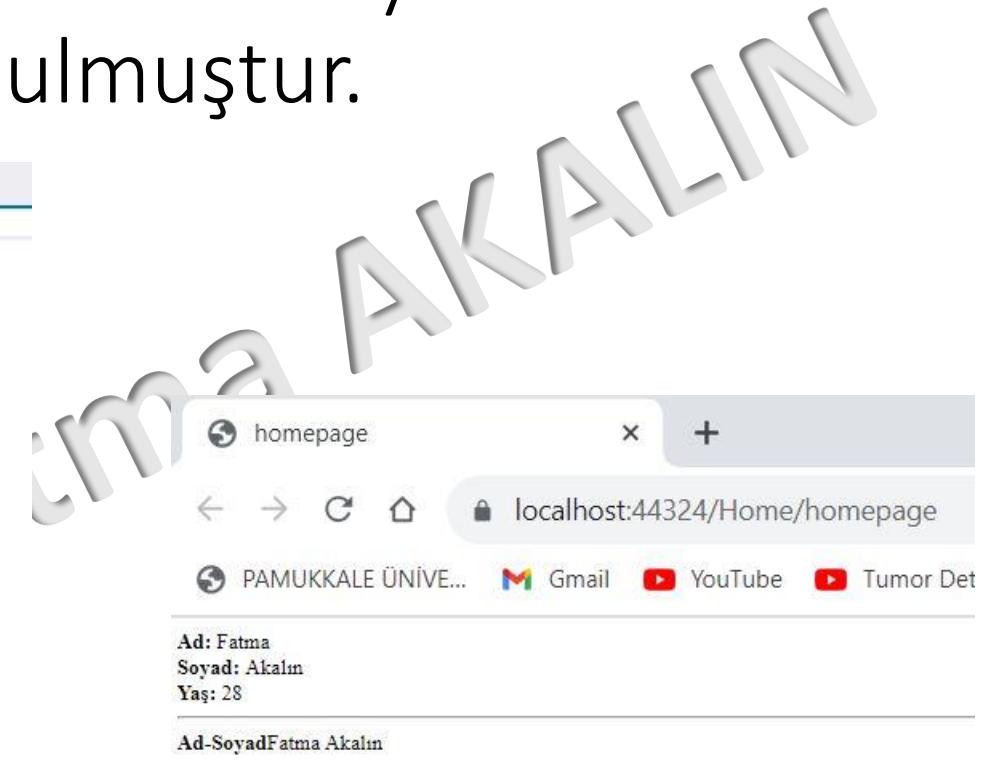
- Kısımlı görünüm olarak oluştur
- Başvuru betik kitaplıklarını
- Bir düzen sayfası kullanın:

At the bottom of the dialog, the text "(Bir Razor _viewstart dosyasında ayarlandıysa, boş bırakın)" is displayed. At the very bottom are the "Ekle" (Add) and "İptal" (Cancel) buttons.

Homepage.cshtml viewını yazdık. View sayfası ve ekran çıktısı aşağıda sunulmuştur.



```
homepage.cshtml  X Kisi.cs      Adres.cs      HomeController.cs
1  @{
2      Layout = null;
3
4
5
6  <!DOCTYPE html>
7
8  <html>
9    <head>
10       <meta name="viewport" content="width=device-width" />
11       <title>homepage</title>
12    </head>
13   <body>
14     <div>
15       <b>Ad: </b>@Model.Ad<br />
16       <b>Soyad: </b>@Model.Soyad<br />
17       <b>Yaş: </b>@Model.Yas
18       <hr />
19       <b>Ad-Soyad</b>@(Model.Ad + " " + Model.Soyad)
20
21     </div>
22     <hr />
23   </body>
24   </html>
```



Buraya kadar Model sınıfımız vasıtayıla Controller nesnemizi oluşturup view'a gönderdik. Ardından Viewda nesnemizin tanımı

```
@using denemeonuncuders.Models
```

```
@model Kisi
```

şeklinde yaptık. Şimdi Model classını kullanarak verilerini istediğiniz gibi taşıyabilirsin.

Pekiii, Model tipinde büradan veriyi nasıl geri alırsın?

Normalde Homepagemiz get işlemi yapıyor ve ekrana açlıyor idi. Peki bu homepageden veri alırken ne yapacağız?

Dr. Öğr.

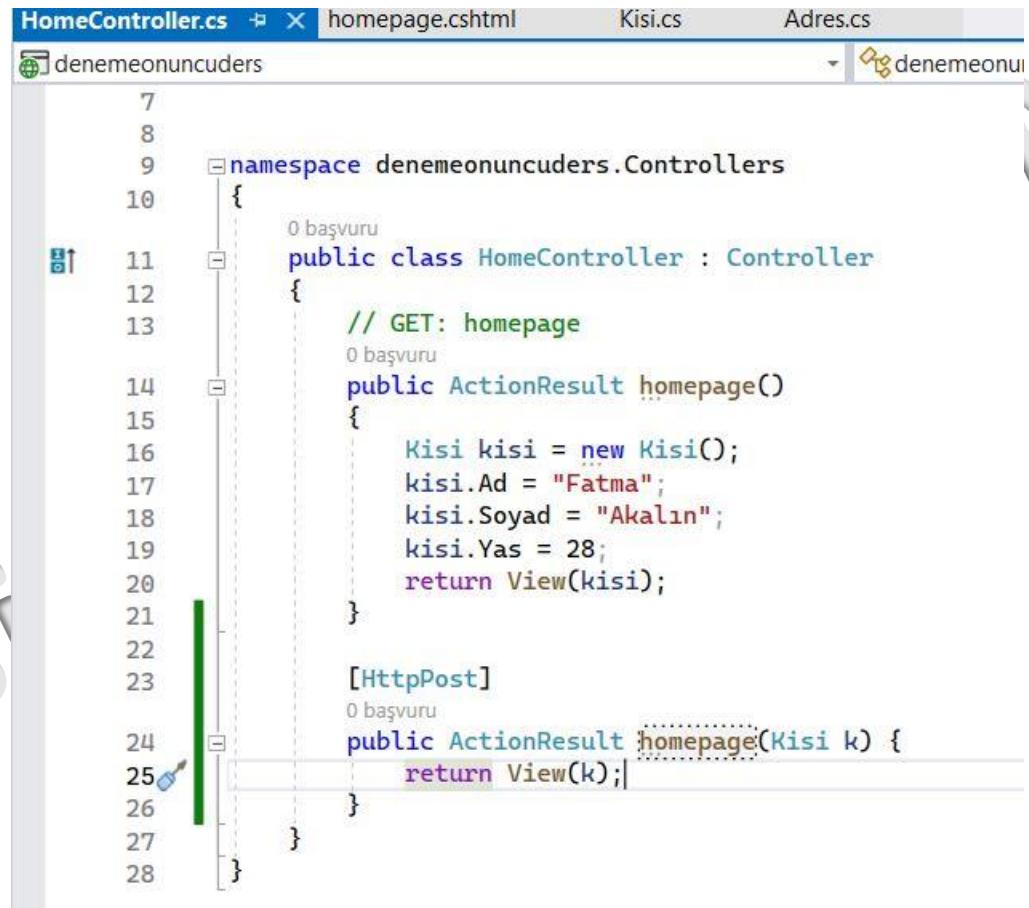
Üyesi

Fatma AKALIN

Dr. Öğr. Üyesi Fatma AKALIN

HttpPost

Homepageden veri almak için post metodunu yazmamız gerek. Post metoduma geri dönüş parametresi olarak string, string ve yaşı verebilirim ama burada alacağım değeri Kisi tipinde tanımlayacağım. Kullandığım o sayfanın modelinin tipi ne ise buraya o isimli bir parametre ekliyorum.



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

- Project Name:** denemeonuncuders
- File:** HomeController.cs
- Code Content:**

```
7
8
9  namespace denemeonuncuders.Controllers
10 {
11     public class HomeController : Controller
12     {
13         // GET: homepage
14         public ActionResult homepage()
15         {
16             Kisi kisi = new Kisi();
17             kisi.Ad = "Fatma";
18             kisi.Soyad = "Akalin";
19             kisi.Yas = 28;
20             return View(kisi);
21         }
22
23         [HttpPost]
24         public ActionResult homepage(Kisi k)
25         {
26             return View(k);
27         }
28     }
```

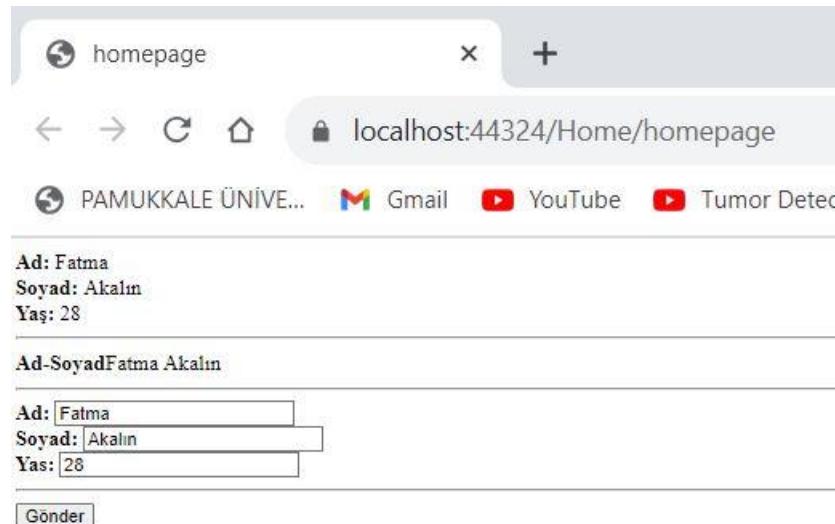
View klasöründeki homepage.cshtml 'ye gelelim.

Veriyi gönderirken `Html.BeginForm()` kullanacağım ve buton ile formu göndermesini isteyeceğim.

```
@using (Html.BeginForm())
{
    <b>Ad: </b>@Html.TextBoxFor(x => x.Ad)<br />
    <b>Soyad: </b>@Html.TextBoxFor(x => x.Soyad)<br />
    <b>Yas: </b>@Html.TextBoxFor(x => x.Yas)
    <hr />
    <input type="submit" value="Gönder" />
}
```

homepage.cshtml'nin son hali yanda verilmiştir.

Ekran çıktısı aşağıda sunulmuştur.



```
controller.cs      homepage.cshtml ✎ X Kisi.cs      Adres.cs
@using denemeonuncuders.Models
@model Kisi
{
    Layout = null;
}
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>homepage</title>
</head>
<body>
    <div>
        <b>Ad: </b>@Model.Ad<br />
        <b>Soyad: </b>@Model.Soyad<br />
        <b>Yaş: </b>@Model.Yas
        <hr />
        <b>Ad-Soyad</b>@(Model.Ad + " " + Model.Soyad)
    </div>
    <hr />
    <div>
        @using (Html.BeginForm())
        {
            <b>Ad: </b>@Html.TextBoxFor(x => x.Ad)<br />
            <b>Soyad: </b>@Html.TextBoxFor(x => x.Soyad)<br />
            <b>Yaş: </b>@Html.TextBoxFor(x => x.Yas)
            <hr />
            <input type="submit" value="Gönder" />
        }
    </div>
</body>
</html>
```

Model kullanımında bir sayfaya sadece bir model tanımlanabiliyor tip olarak. Benim iki nesne döndürmem gerek derseniz yeni bir model daha yapılması gerekir o sayfa için. Bu doğrultuda **Kisi classımız birde adres classımız olsun.**(Bunların hepsi databasede bir tabloya karşılık gelir.)

Örneğin adres ve kişi classımızdaki datalar birbirini tamamlayıcı özellikte olabilir ve ikisinin datasını birden sayfaya göndermek isteyebilirsin.

Return View (kisi) ile sadece bir tane alabidiği için **sayfaya özel class yapmalıyız.** Bunun için projenin içerisinde bir tane daha klasör ekleyip ViewModel deyip o sayfaya özgü bir class oluşturmak istiyorum. Çünkü o sayfada birden fazla class ihtiyacım olabilir. **Bu çalışmamız ile birleştirme işlemini yapacağız.** Ek olarak bir önceki örneği yeniden tekrar etmek ve net olmayan kısımları netleştirmek için sıfırdan başlamak suretiyle bir proje yapalım..

SIFIRDAN TEKRAR EDEREK PROJEMİZE BAŞLAYALIM

Yeni projenizi yapılandırın

ASP.NET Web Uygulaması (.NET Framework)

Proje adı

denemeonbirinciders

Konum

D:\aspnet\1_ADESLER

Çözüm adı (i)

denemeonbirinciders

Çözümü ve projeyi aynı dizine yerleştir

Altıyağı

.NET Framework 4.6.2

Proje "D:\aspnet\1_ADESLER\denemeonb

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 denetimi modeli kullanarak dinamik web siteleri oluşturmanızı sağlar. Tasarım yüzeyi ile yüzlerce denetim ve bileşen, veri erişimi olan gelişmiş, güçlü, kullanıcı arabirimi denetimi 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 gereklidir)
- Ayrıca birim testleri için bir proje oluştur
denemeonbirinciders.Tests

Geri

Oluştur

Yeni İскеlevi Öğe Ekle

Yükü
Ortak
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: MvcControllerEmpty

Kisi.cs X Adres.cs X HomeController.cs X

denemeonbirinciders

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

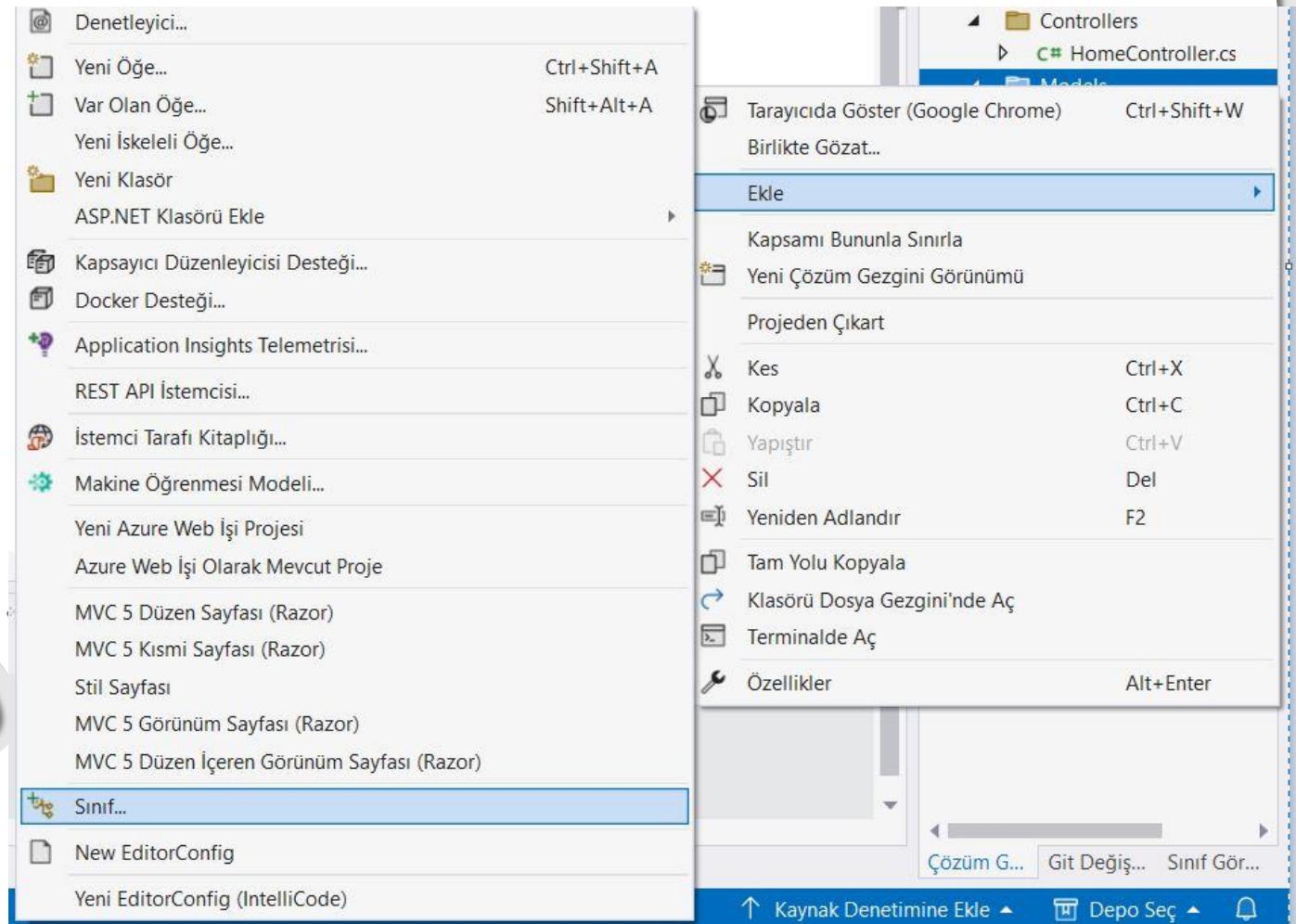
Ekle

Denetleyici Ekle

Denetleyici adı: HomeController

Ekle İptal

Proje oluşturulduktan sonra models klasörüne iki sınıf ekleyelim..



Sıralama ölçütü: Varsayılan

C#

- Sınıf C#
- Class for U-SQL C#
- Arabirim C#
- ADO.NET Entity Data Model C#
- Bütünleştirilmiş Kod Bilgi Dosyası C#
- Code Analysis Kural Kümesi C#
- Çalışma Zamanı Metin Şablonu C#
- editorconfig Dosyası (.NET) C#
- editorconfig Dosyası (varsayılan) C#
- EF 5.x DbContext Generator C#
- EF 6.x DbContext Generator C#
- Hata Ayıklama Görelleştiricisi C#
- JavaScript JSON Yapılandırma Dosyası C#
- Kaynaklar Dosyası C#

Kısıcs

Yeni Öğe Ekle - denemeonbirinciders

Yüklenen

C#

Genel

Kod

Veri

Web

SQL Server

Storm Items

Çevirmiçi

Sıralama ölçütü: Varsayılan

C#

Sınıf

C#

Class for U-SQL

C#

Arabirim

C#

ADO.NET Entity Data Model

C#

Bütünleştirilmiş Kod Bilgi Dosyası

C#

Code Analysis Kural Kümesi

C#

Çalışma Zamanı Metin Şablonu

C#

editorconfig Dosyası (.NET)

C#

editorconfig Dosyası (varsayılan)

C#

EF 5.x DbContext Generator

C#

EF 6.x DbContext Generator

C#

Hata Ayıklama Görelleştiricisi

C#

JavaScript JSON Yapılandırma Dosyası

C#

Kaynaklar Dosyası

C#

Ad:

Adres.cs

Ekle

İptal

Adres.cs X Kisi.cs HomeController.cs

denemeonbirinciders

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;

5
6  namespace denemeonbirinciders.Models
7  {
8      public class Adres
9      {
10         public string AdresTanim { get; set; }
11         public string Sehir { get; set; }
12     }
13 }
```

Kisi.cs X Adres.cs HomeController.cs

denemeonbirinciders

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;

5
6  namespace denemeonbirinciders.Models
7  {
8      public class Kisi
9      {
10         public string Ad { get; set; }
11         public string Soyad { get; set; }
12         public int Yas { get; set; }
13     }
14 }
```

Dr. Öğr. Üy.

Adres ve Kişi classımızdaki
datalar birbirini tamamlayıcı
özellikte olduğu ve ikisinin
datasını birden sayfaya
göndermek istediğim için
Models klasörünün altında
ViewModels isimli klasörü
oluşturup ViewModels
klasörünün altına
homepageViewModels isimli
yeni bir sınıf daha eklemeliyim.

The screenshot shows the Microsoft Visual Studio IDE interface. The code editor at the top has tabs for Adres.cs, homepageViewModel.cs (which is currently selected), and HomeController.cs. The Solution Explorer on the right shows a project named 'denemeonbirinciders' with its files and folders. A context menu is open over the 'ViewModels' folder in the 'Models' directory, listing various template options like 'Sınıf' (Class), 'Class for U-SQL', 'Arabirim' (Interface), etc. The 'Arabirim' (Interface) option is highlighted. The status bar at the bottom indicates the file is 'homenameViewModel.cs'.

```
cs Adres.cs x homepageViewModel.cs* HomeController.cs*
lenemeonbirinciders denemeonbirinciders.Models.\

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;

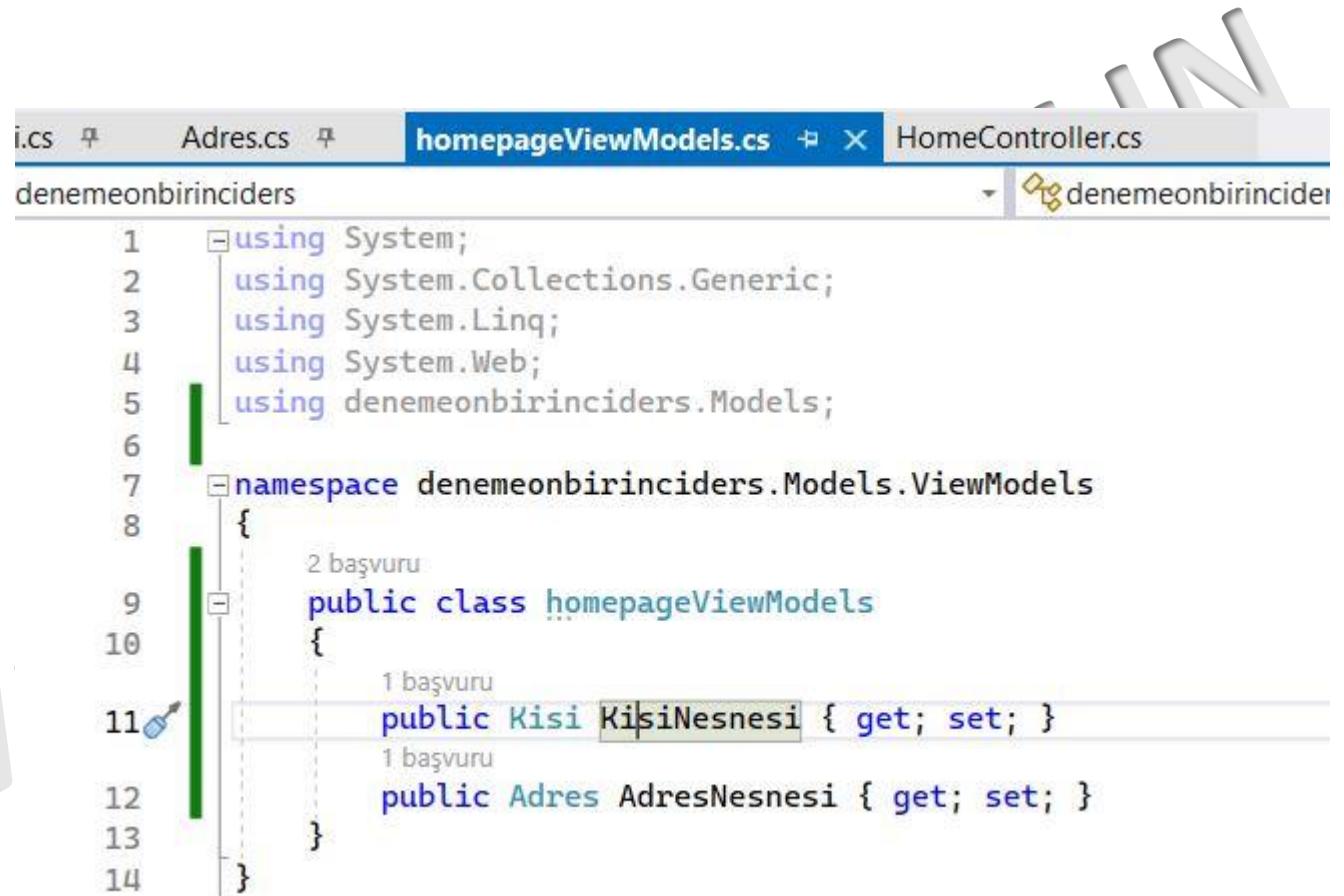
5
6  namespace denemeonbirinciders.Models.ViewModels
7  {
8      public class homepageViewModel
9      {
10  }
11 }
```

Sıralama ölçütü: Varsayılan

- Sınıf C#
- Class for U-SQL C#
- Arabirim C# (selected)
- ADO.NET Entity Data Model C#
- Bütünleştirilmiş Kod Bilgi Dosyası C#
- Code Analysis Kural Kümesi C#
- Çalışma Zamanı Metin Şablonu C#
- editorconfig Dosyası (.NET) C#
- editorconfig Dosyası (varsayılan) C#
- EF 5.x DbContext Generator C#
- EF 6.x DbContext Generator C#
- Hata Ayıklama Görüntüleme Dosyası C#
- JavaScript JSON Yapılandırma Dosyası C#
- Kaynaklar Dosyası C#

homepageViewModels
sayfasında biri kişi tipinde
kişi nesnesi ve diğeride
adres tipinde adres
nesnesi olarak iki farklı
nesne oluşturacağım.

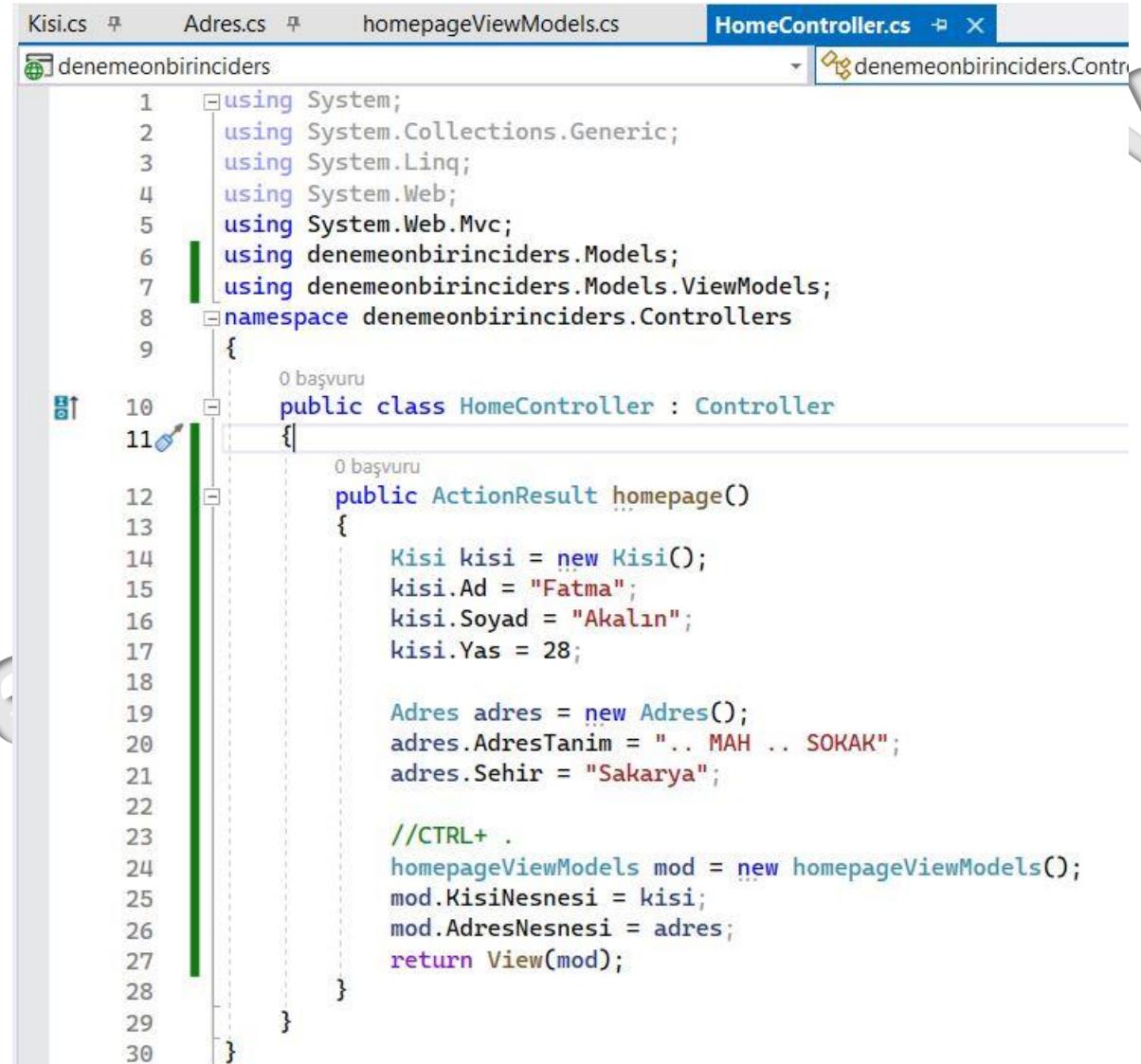
Dr. Öğr. ÜY



```
i.cs Adres.cs homepageViewModels.cs HomeController.cs
denemeonbirinciders denemeonbirinciders
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;
5  using denemeonbirinciders.Models;
6
7  namespace denemeonbirinciders.Models.ViewModels
8  {
9      public class homepageViewModels
10     {
11         public Kisi KisiNesnesi { get; set; }
12         public Adres AdresNesnesi { get; set; }
13     }
14 }
```

Ardından biri kişi tipinde kişi nesnesi ve diğerinin adres tipinde adres nesnesi olarak iki tane özellik ekleyeceğim.

Controllerda yapacağım bu işlemde verilerimi girerken Modellerimi tanıtmayı unutmamalıyım.



```
Kisi.cs  Adres.cs  homepageViewModels.cs  HomeController.cs  denemeonbirinciders.Controllers HomeController.cs
denemeonbirinciders
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;
5  using System.Web.Mvc;
6  using denemeonbirinciders.Models;
7  using denemeonbirinciders.Models.ViewModels;
8  namespace denemeonbirinciders.Controllers
9  {
10     public class HomeController : Controller
11     {
12         public ActionResult homepage()
13         {
14             Kisi kisi = new Kisi();
15             kisi.Ad = "Fatma";
16             kisi.Soyad = "Akalın";
17             kisi.Yas = 28;
18
19             Adres adres = new Adres();
20             adres.AdresTanim = "... MAH .. SOKAK";
21             adres.Sehir = "Sakarya";
22
23             //CTRL+ .
24             homepageViewModels mod = new homepageViewModels();
25             mod.KisiNesnesi = kisi;
26             mod.AdresNesnesi = adres;
27             return View(mod);
28         }
29     }
30 }
```

Ardından view'ı oluşturalım



```
using system;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using denemeonbirinciders.Models;
using denemeonbirinciders.Models.ViewModels;
namespace denemeonbirinciders.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult homepage()
        {
            Kisi kisi = new Kisi();
            kisi.Ad = "Fatma";
            kisi.Soyad = "Akalin";
            kisi.Yas = 28;

            Adres adres = new Adres();
            adres.AdresTanim = "... MAH .. SOKA";
            adres.Sehir = "Sakarya";

            //CTRL+ .
            homepageViewModels mod = new homepageViewModels();
            mod.KisiNesnesi = kisi;
            mod.AdresNesnesi = adres;
            return View(mod);
        }
    }
}
```

Sorun bulunamadi

Data Ayıklama

Yeni İskeleli Öğe Ekle

Yükü
Ortak
MVC
Alan
Denetleyici
Göster
Web API'si

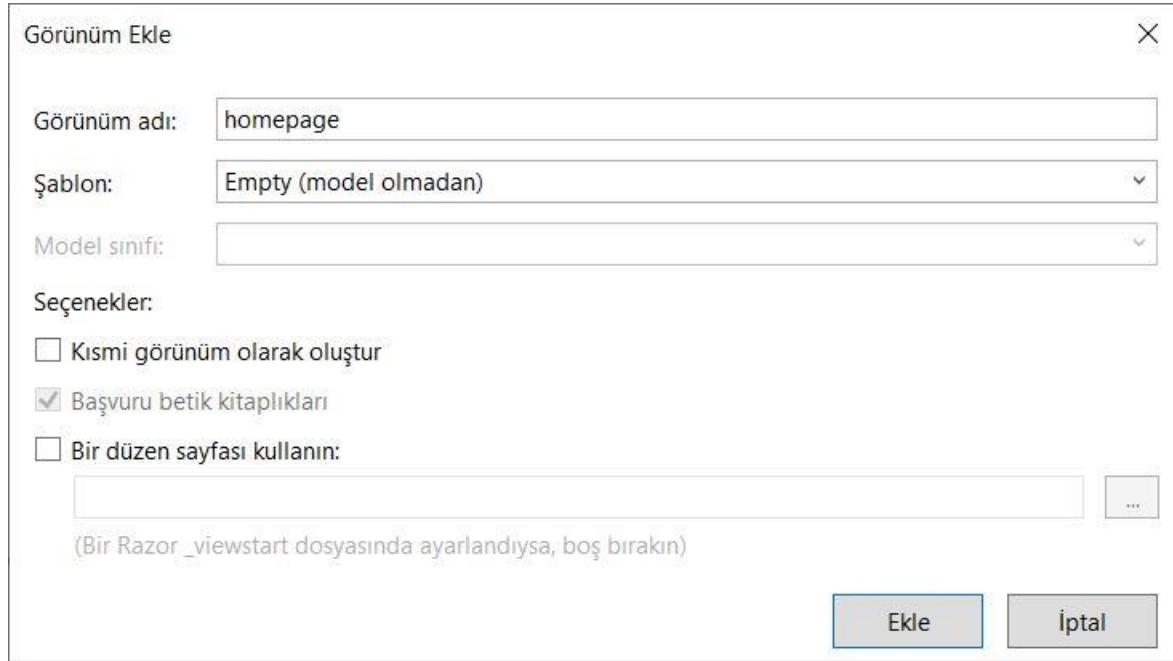
MVC 5 Görünümü

MVC 5 Görünümü
şunun tarafından Microsoft
v5.0.0.0

Türü kesin belirtilmiş bir Modelle veya
Model olmadan MVC görünümü

Kimlik: MvcViewScaffolder

Ekle İptal

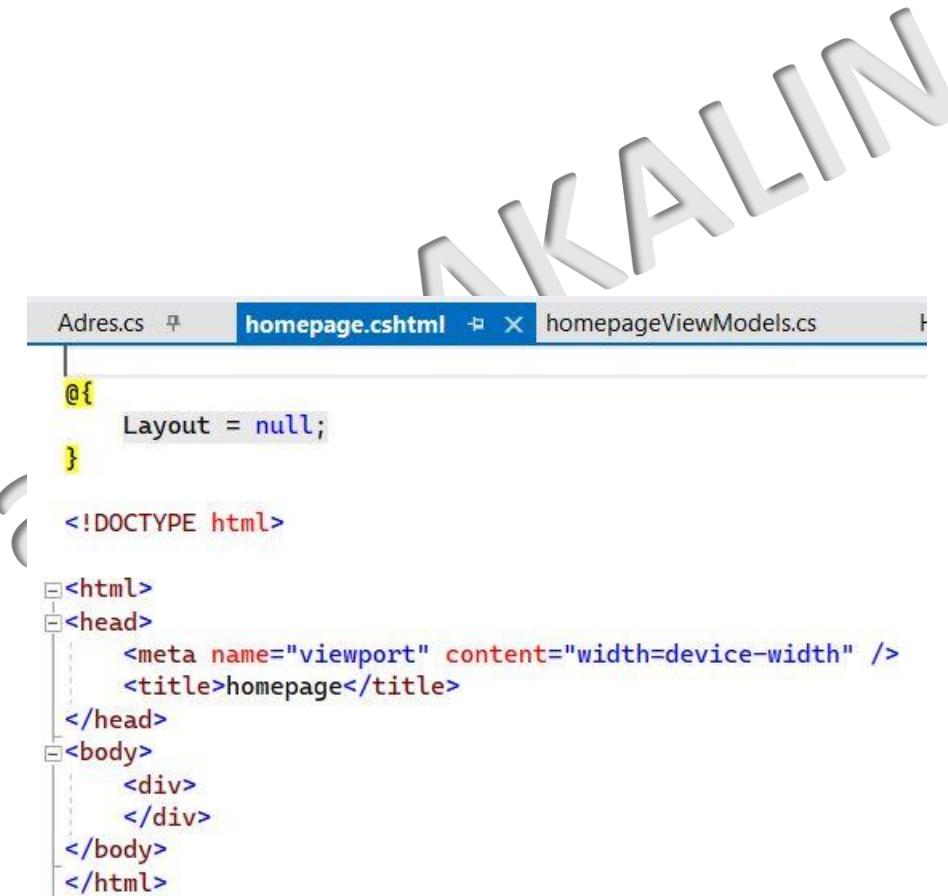


Adres.cs **homepage.cshtml** X homepageViewModels.cs

```
@{ Layout = null; }

<!DOCTYPE html>

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



Fakat oluşturduğum view klasörünün içindeki cshtml sayfam, bu modelin geleceğinden habersiz. Bu nedenle,

```
@using denemeonbirinciders.Models.ViewModels  
@model homepageViewModels
```

Kodları ile homepage.cshtml sayfama haber veriyorum.

denemeonbirinciders.Models.ViewModels->ilgili isme sahip projemin
içindeki Models klasörümün altında ViewModels isimli klasörümü eriş
anlamına gelir

Şimdi homepage.cshtml sayfamı düzenleyeceğim.

Önceki örnekte bu kısımları tanımlamamıştık. Bir önceki örnektenden farkı homepageViewModels klasörünü kullanmamızdır iki ayrı sınıf yapısını kullanmak istediğimiz için.

```
homepageViewModels.cs Adres.cs Kisi.cs Index.cshtml HomeController.cs
1 @using WebApplication22.Models
2 @using WebApplication22.Models.ViewModels;
3 @model homepageViewModels
4 @{
5     Layout = null;
6 }
```

```
Adres.cs homepage.cshtml homepageViewModels.cs HomeController.cs
@using denemeonbirinciders.Models.ViewModels
@model homepageViewModels
{@
    Layout = null;
}
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>homepage</title>
</head>
<body>
    <div>
        <b>Ad: </b>@Model.KisiNesnesi.Ad<br />
        <b>Soyad: </b>@Model.KisiNesnesi.Soyad<br />
        <b>Yaş: </b>@Model.KisiNesnesi.Yas
        <hr />
        <b>Ad-Soyad</b>@(Model.KisiNesnesi.Ad + " " + Model.KisiNesnesi.Soyad)
    </div>
    <hr />
    <div>
        @using (Html.BeginForm())
{
    <b>Ad: </b>@Html.TextBoxFor(x => x.KisiNesnesi.Ad)<br />
    <b>Soyad: </b>@Html.TextBoxFor(x => x.KisiNesnesi.Soyad)<br />
    <b>Yas: </b>@Html.TextBoxFor(x => x.KisiNesnesi.Yas)
    <hr />
    <b>Adres: </b>@Html.TextBoxFor(x => x.AdresNesnesi.AdresTanim)<br />
    <b>Sehir: </b>@Html.TextBoxFor(x => x.AdresNesnesi.Sehir)<br />
    <input type="submit" value="Gönder" />
}
    </div>
</body>
</html>
```

Bu işlemlerin ardından controller vasıtasıyla get ve post metotları tanımlayarak kullanıcidan gelen cevabın işlenmesini sağlayabilirsın.

Dr. Öğr. Üyesi

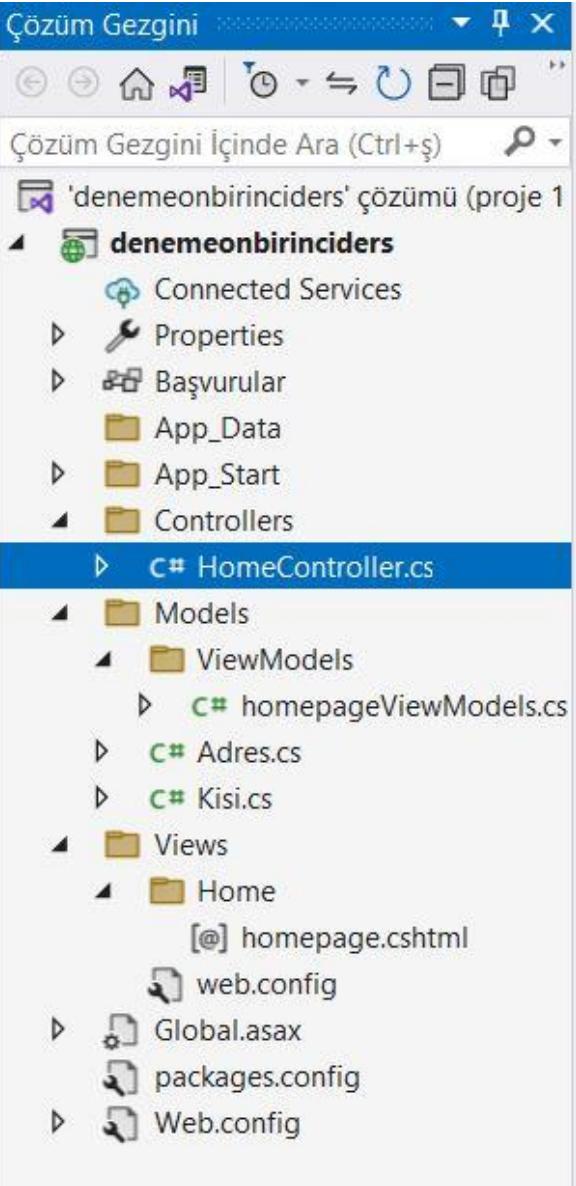
```
Adres.cs  homepage.cshtml HomeController.cs  deneme
irinciders
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using denemeonbirinciders.Models;
using denemeonbirinciders.Models.ViewModels;
namespace denemeonbirinciders.Controllers
{
    0 başvuru
    public class HomeController : Controller
    {
        0 başvuru
        public ActionResult homepage()
        {
            Kisi kisi = new Kisi();
            kisi.Ad = "Fatma";
            kisi.Soyad = "Akalın";
            kisi.Yas = 28;

            Adres adres = new Adres();
            adres.AdresTanim = "... MAH ... SOKAK";
            adres.Sehir = "Sakarya";

            //CTRL+ .
            homepageViewModels mod = new homepageViewModels();
            mod.KisiNesnesi = kisi;
            mod.AdresNesnesi = adres;
            return View(mod);
        }
        [HttpPost]
        0 başvuru
        public ActionResult homepage(homepageViewModels k)
        {
            return View(k);
        }
    }
}
```

Dr. Öğr. Üyesi

Klasör
hiyerarşisi
su şekilde
olacaktır.



İKALIN

View dosyamızın
ekran çıktısı yanda
verilmiştir.

Dr. Öğr.

The screenshot shows a Windows desktop environment. On the left, the Visual Studio IDE is open, displaying the 'homepage.cshtml' file. The code uses Razor syntax to render a person's information and a form for editing it. On the right, a web browser window is open at 'localhost:44336/Homepage'. It shows the rendered HTML from the view, which includes bolded labels ('Ad:', 'Soyad:', 'Yaş:'), a separator line ('

'), and a bolded section ('Ad-Soyad'). Below this, there is a text input field for 'Ad:' containing 'Fatma', a text input field for 'Soyad:' containing 'Akalin', and a text input field for 'Yaş:' containing '28'. Further down, there is a text input field for 'Adres:' containing '... MAH ... SOKAK', a text input field for 'Sehir:' containing 'Sakarya', and a red 'Gönder' button. The browser also displays the URL 'localhost:44336/Homepage' in the address bar and some other links at the top right.

```
Adres.cs  homepage.cshtml  HomeController.cs
@using denemeonbirinciders.Models.ViewModels
@model homepageViewModels
 @{
     Layout = null;
 }
 <!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>homepage</title>
</head>
<body>
    <div>
        <b>Ad: </b>@Model.KisiNesnesi.Ad<br />
        <b>Soyad: </b>@Model.KisiNesnesi.Soyad<br />
        <b>Yaş: </b>@Model.KisiNesnesi.Yas
        <hr />
        <b>Ad-Soyad</b>@(Model.KisiNesnesi.Ad + " " + Model.KisiNesnesi.Soyad)
    </div>
    <hr />
    <div>
        @using (Html.BeginForm())
        {
            <b>Ad: </b>@Html.TextBoxFor(x => x.KisiNesnesi.Ad)<br />
            <b>Soyad: </b>@Html.TextBoxFor(x => x.KisiNesnesi.Soyad)
            <b>Yaş: </b>@Html.TextBoxFor(x => x.KisiNesnesi.Yas)
            <hr />
            <b>Adres: </b>@Html.TextBoxFor(x => x.AdresNesnesi.Adres)
            <b>Sehir: </b>@Html.TextBoxFor(x => x.AdresNesnesi.Sehir)
            <input type="submit" value="Gönder" />
        }
    </div>
</body>
</html>
```

Kodları yazım aşamasında da fark edeceğimiz gibi model kullanmak çok rahat bir süreç sunuyor. Sizlere anlatılan işlemler ile sayfalara özel yazdığınız birden fazla model classlarını viewModel vasıtasıyla birleştirebilirsiniz.

Özetle aynı sayfada birden fazla model tanımlamak istiyor isen bu sefer model klasörünün içerisinde ekstra bir dosya açıp her iki sınıfın atama yapacaksın ve controllera gelip bu sınıflar için get ve post metotları oluşturacaksın. Sonra da view sayfası ile kullanıcıda gözükecek sayfanın düzenlemesini, atamasını ve tanımlamalarını yapacaksın.

Dr. Öğr.

KAYNAKÇA

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

Veysel Uğur Kızmaz, ASP.NET MVC5, Kodlab Yayımları

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