

ASP. NET CORE 5.0 KONU BAŞLIKLARI

◦ Asp.NET - Asp.NET Core – Asp.NET Core 5.0

◦ Web Mantığı

◦ HTTP Protokolü

◦ Sunucu Çeşitleri

◦ Backend – Frontend Kavramları

◦ Web Uygulaması Geliştirme Modelleri

◦ MVC(Model – View – Controller)

◦ Model

◦ View

◦ Controller

◦ Razor

◦ HtmlHelper & TagHelper & UrlHelper

◦ Model Binding

◦ Kullanıcıdan Veri Alma Yöntemleri

◦ Validation

◦ Layout

◦ Modüler Tasarım Yapılanması

◦ Route Yapılanması

◦ Asp.NET Core Özellikleri

◦ Dependency Injection

◦ Areas

Dipnot:

Bu İçeriği yani Asp.Net Core Mvc özet dökümanını hazırlarken Gençay Yıldız Hocanın kendi web sitesinde <https://ngakademi.com/courses/ozel-ders-formatinda-adan-zye-asp-net-core-5-0-web-programlama-egitimi/> ve youtube kanalında olan

<https://www.youtube.com/watch?v=RM EhZjnoTrY&list=PLQVXoXFVVtp33KHoTkWkIAo72l5bcjPVL> ve [Genel bakış ASP.NET Core | Microsoft Docs](#) kaynak olarak edindim burdan yardım aldım. Bu doküman bir özet niteliğindedir ve degenilmeyen ve ileride degenilecek konuların olduğu bir dökümandır.

Asp.NET Nedir ?

Bir programlama dili değildir . Bir programlama dili olan C# 'la kodlanabilen web geliştirme, web uygulaması geliştirme mimarisidir

- Microsoft tarafından geliştirilmiş bir Web Uygulama Geliştirme mimarisidir.

Mimari ne demek? Belirli bir yaklaşımı kabul etmiş , benimsemiş bir programlama diliyle desteklenen ve belirli bir amaca hizmet eden bir yapılan yaklaşımıdır.

Asp.NET Core Nedir?

- Mikrosoft tarafından geliştirilen ücretsiz ve açık kaynak(open source) Web Geliştirme Mimarisiidir.
- Asp.NET'in halefidir.
- Windows, Linux vs. Çalışır.
- Tüm Asp.NET altyapısı yeniden tasarlanmıştır.

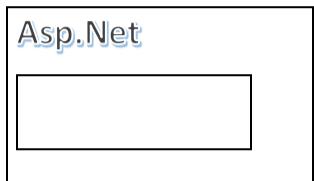
Asp.NET Core'un Asp.NET'ten Farkları

Daha Performans	Cross Platform Her yerde çalışabilmektedir	Modüler Yapı Bütünü Parçala-yönet	Bağımlılık Enjeksiyon (Dependency Injection)
Asenkron İşlemler	Kolay Bakım	Razor Pages	

Asp.NET Core 5.0

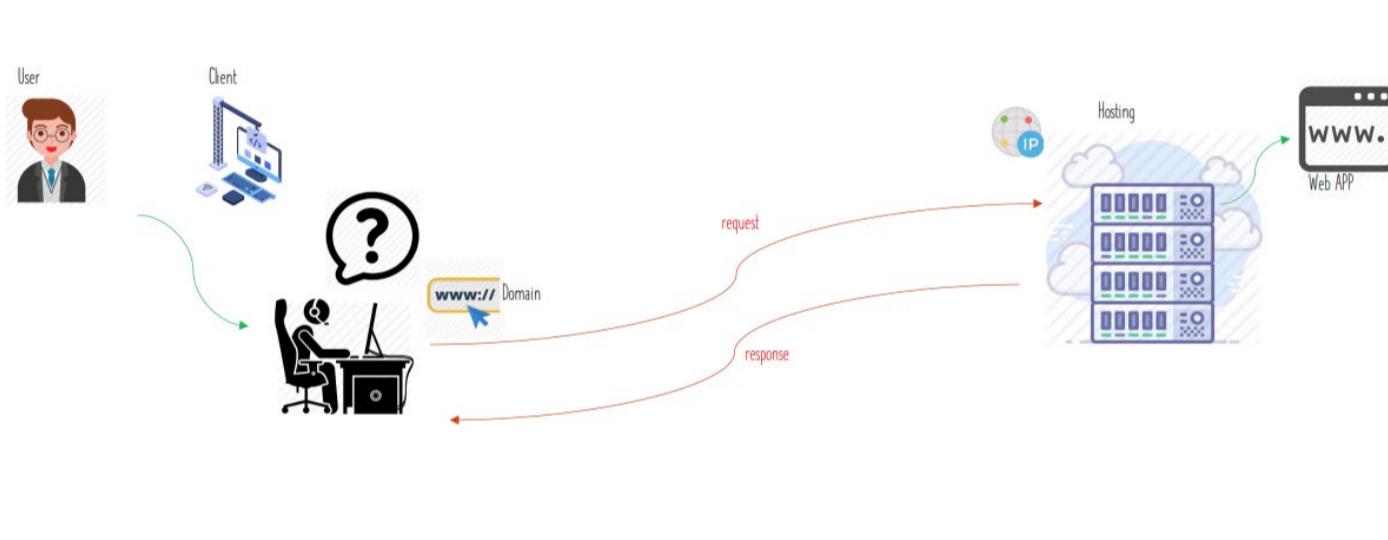
- Asp.NET Core'un günümüzdeki versiyonudur.
- Bazı geliştirmelere sahiptir.
- Open API - SwaggerDoc
- Performans vs.

Asp.Net Core 5.0



Web Mantığı

İnternette Gezerken Perde Arkası Nasıldır?



İnternetteki mekanizma request-response mantığına dayanır

Kullanıcı olarak ilgili websitesine bir request atılır ve bu request server denen noktadan karşılanır ilgili istek için sonuç üretir ve bu sonucu döndürür ve kullanıcının ekranında gözükmür (response).

Request(İstek) : Bir web sitenin açılması için yaptığımız eyleme girdik/tıkladık vs değil, istek gönderdik diyeceğiz!

Temel Kavramlar

User	Client	Hosting	Ip
Domain	Request	Responce	

User: İlgili işlemi Kullanacak birey.

Client: User'ın kullandığı teknolojidir. Günlük hayatı interneti kullanan birey tablet , pc ,telefon kullanılır, Cihazdaki tarayıcı arayıcılığıyla internete client'i(istemci) kullanarak internete giriyor.Tarayıcı da çalışan istemci client'dır.

Hosting: Server,sunucu... Normalde bir web sitesinin dış dünyaya açıldığıda (public) 7/24 yayını sunan , bakım yapılan bilgisayar sunucu kiralamasıdır.

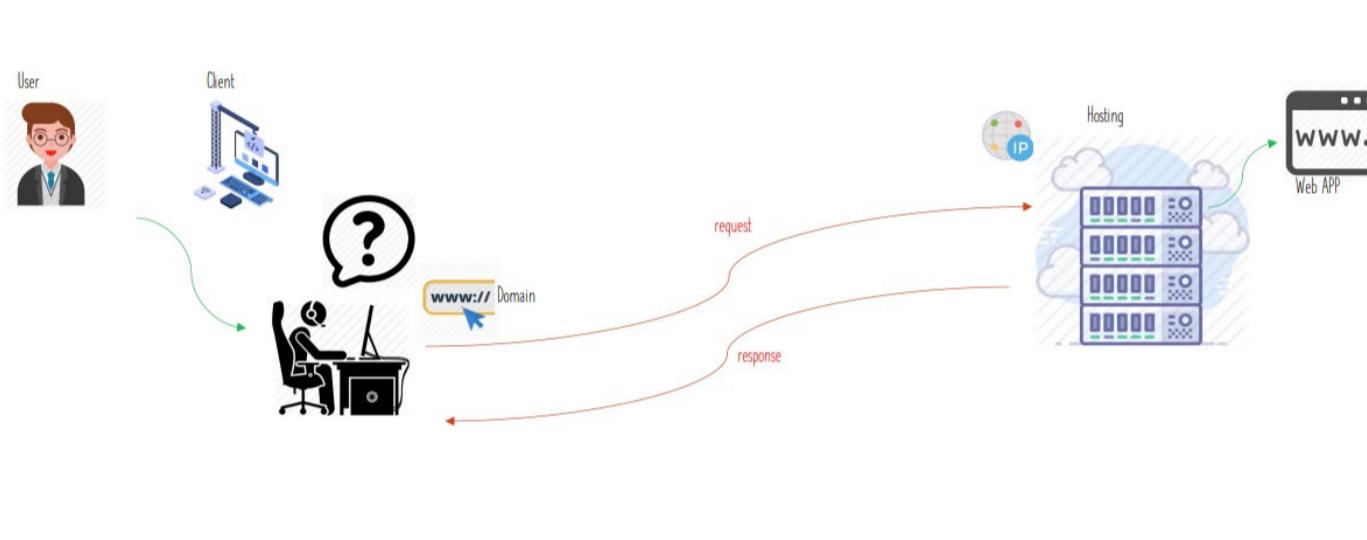
Hosting firmaları web uygulamalarını 7/24 yanında tutabilmemizi sağlayan bilgisiyarları barındıran firmadır. Kendi Bilgisiyarımızı da hosting server yapabiliriz ama bu sorunlu bir yöntem biraz .

Request: . Request içerisinde hangi adrese/ip/domaine istek gönderdiği bilgi tutulur.

Ip: Client'ın yaptığı istek sonucu hosting web uygulamasını çalıştırılır. , Hosting web uygulamasına bir tane ip adresi veriyor bu ip'ye istek gönderiliyor .Anlamsız sayısal değerlerdir

Domain: web uygulamasına verilen ip'yi kullanıcı daha kolay anlamdırabilisin diye verilen ip'ye yönlendirilmiş adresdir.

Response: İstek sonucunda sunucudan gönderilen sonuçtır. Terimsel bir ifadeyle ; Sunucu tarafından client'e dönülen cevaptır.Bu cevap sunucu tarafında üretilen result(sonuç) barındırabilir.



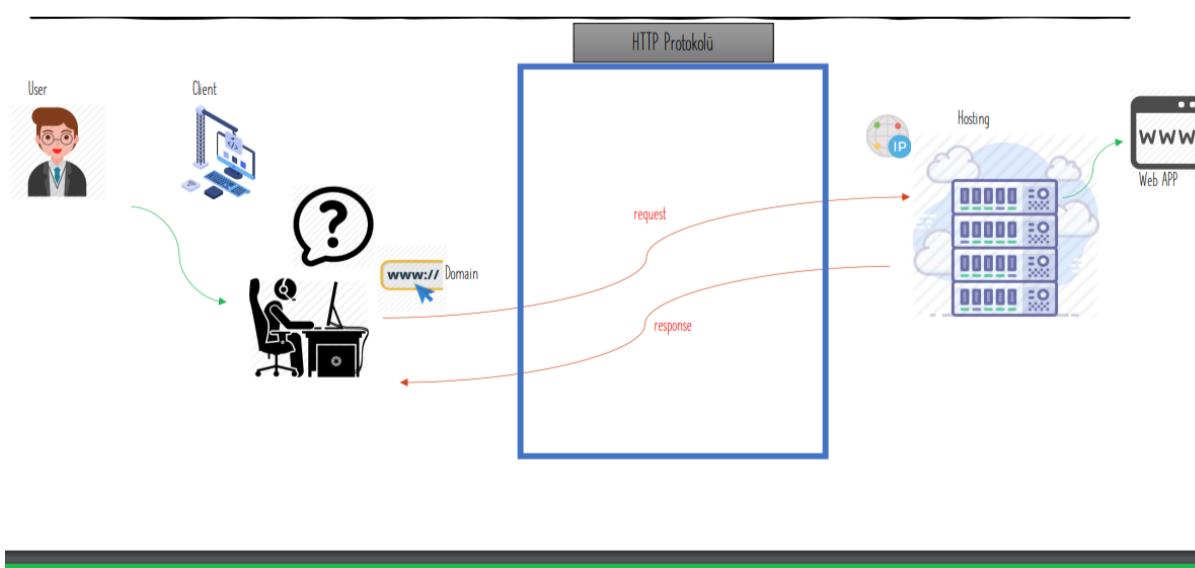
HTTP Protokolü

- Client ile server arasındaki haberleşmeyi sağlayan ilişki protokolüdür.

İki farklı yapılanmanın buluşmasıdır.

- HTTP, client ile server arasındaki ilişkiyi 9 farklı fonksiyonla gerçekleştirir.

- Get • Head • Post • Put • Delete • Trace • Options
- Connect • Patch



HTTP Fonksiyonları

GET: Sunucudan veri(leri) listelemek, elde etmek için kullanılır. Verilere başka etkisi yoktur. (Select) ➔• Tüm haberleri görme • Tüm kullanıcıları listeleme • 5 id'sine sahip kullanıcı bilgilerini edinme • En çok satan ürünleri listeleme • En az satış yapan personelleri listeleme

POST: Sunucuya veri göndermek, eklemek için kullanılır. (Insert) ➔• Haber ekleme • Yorum yapma • Not girme • Profil fotoğrafı ekleme • Sipariş verme • Mesaj gönderme

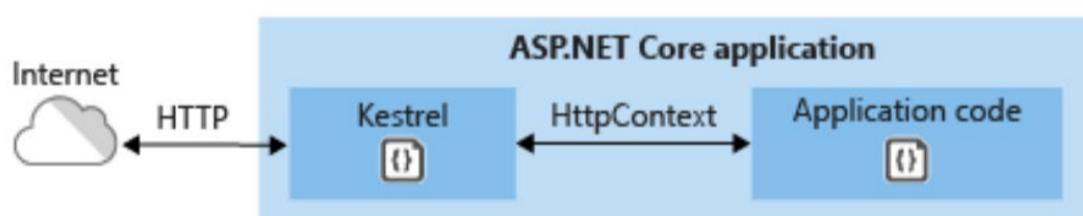
PUT: Var olan bir veriyi değiştirmek, güncellemek için kullanılır. Bir verinin bütününe değil kısmi bir alanını güncelliyorsak eğer bu **PATCH** isteğiidir. (Update) ➔• Haber güncelleme • Yorum güncelleme • Not düzeltme • Profil fotoğrafı değiştirme • Profil bilgileri değiştirme • Sipariş durumunu güncelleme

DELETE: Var olan veriyi silmek için kullanılır. (Delete) ➔• Haber silme • Yorum silme • Not silme • Profil fotoğrafını silme • 5 id'sine sahip kullanıcıyı silme • Sipariş silme

Sunucu Çeşitleri

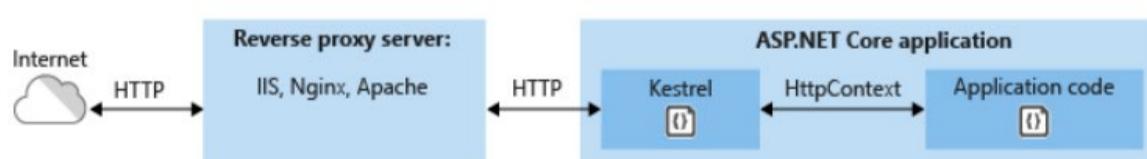
IIS: Internet Information Services(IIS) Asp.NET Core dahil olmak üzere web uygulamalarını barındırmak için esnek, güvenli ve yönetilebilir bir Web Sunucusudur.

Kestrel : Asp.NET Core uygulamalarında dahili olarak gelen bir web sunucusudur. Asp.Net Core'un kendi kendini ayağa kaldırmasına imkan veriyor ve Linux, ubuntu gibi işletim sisteminde kullanılabilir bu sayede.

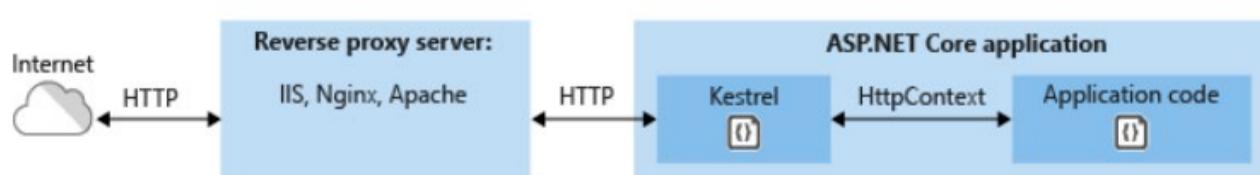


Nginx : • Ubuntu/Linux sistemlerde Asp.NET Core uygulamalarını çalıştırabilmemizi sağlayan bir sunucudur.

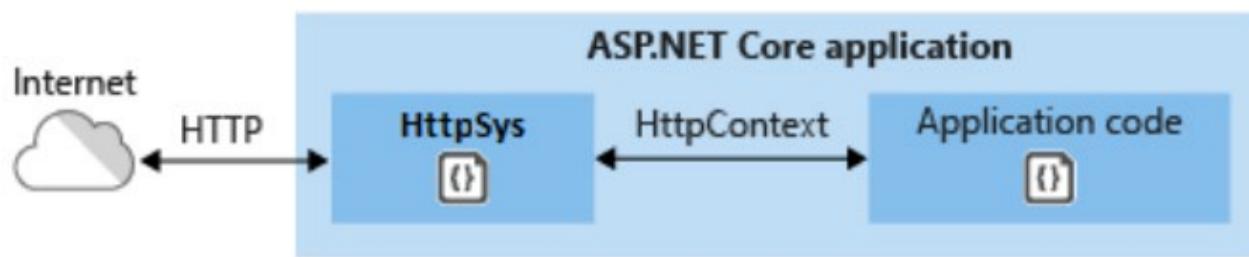
- Reverse Proxy olarak Asp.NET Core uygulamalarındaki dahili sunucusu(Kestrel) işlevsellik gösterir.



Apache : Linux vs. gibi ortamlarda Apache ile Asp.NET Core uygulamalarını ayağa kaldırabilirsiniz.



HTTP.sys : Yanlızca Windows üzerinde çalışan Asp.NET Core için bir web sunucusudur. • Kestrel'in bir alternatifidir.



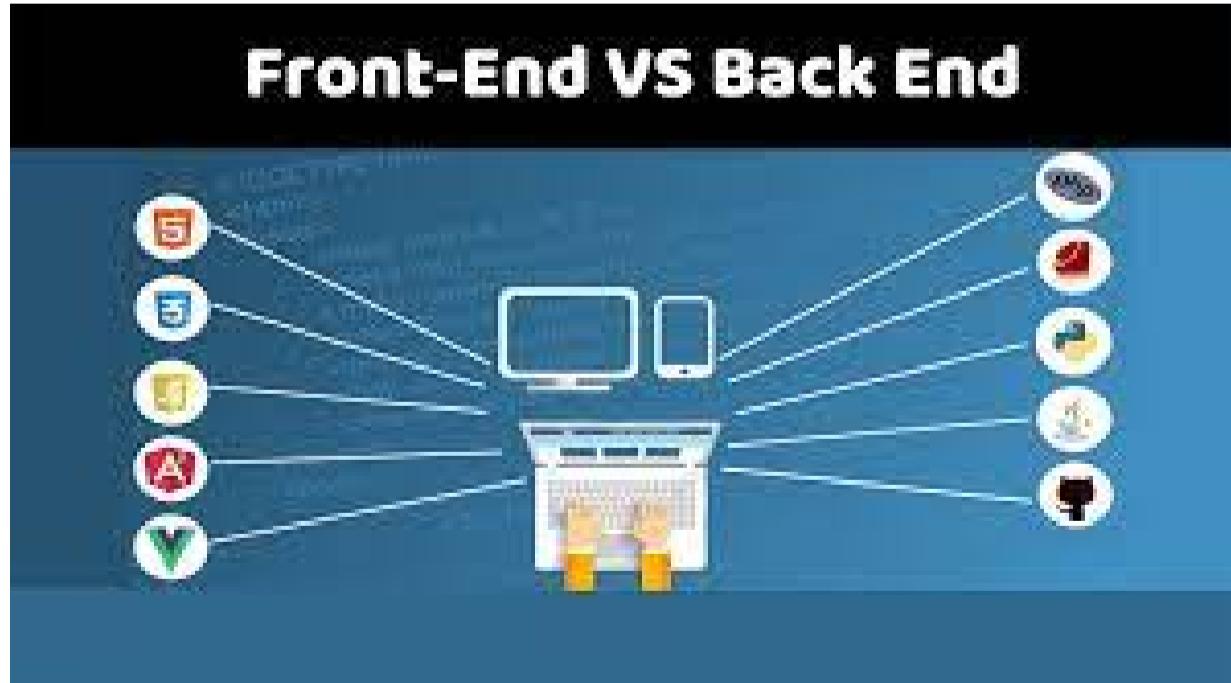
Docker : Yazılım geliştiriciler ve sistem yöneticileri için geliştirilmiş açık kaynak olan bir yeni nesil sanallaştırma platformudur. Bunu genellikle microservices yapılanmasında çalışırken elimizdeki birden fazla servisi ayağa kaldırırmak için kullanılan sanallaştırma platformudur. Tek başına bir sunucu değildir. Çalışacağımız zaman içerisinde biz devlopeerler sunucu kurarak çalıştırırız.

Backend – Frontend

Kavramları



Front-End VS Back End





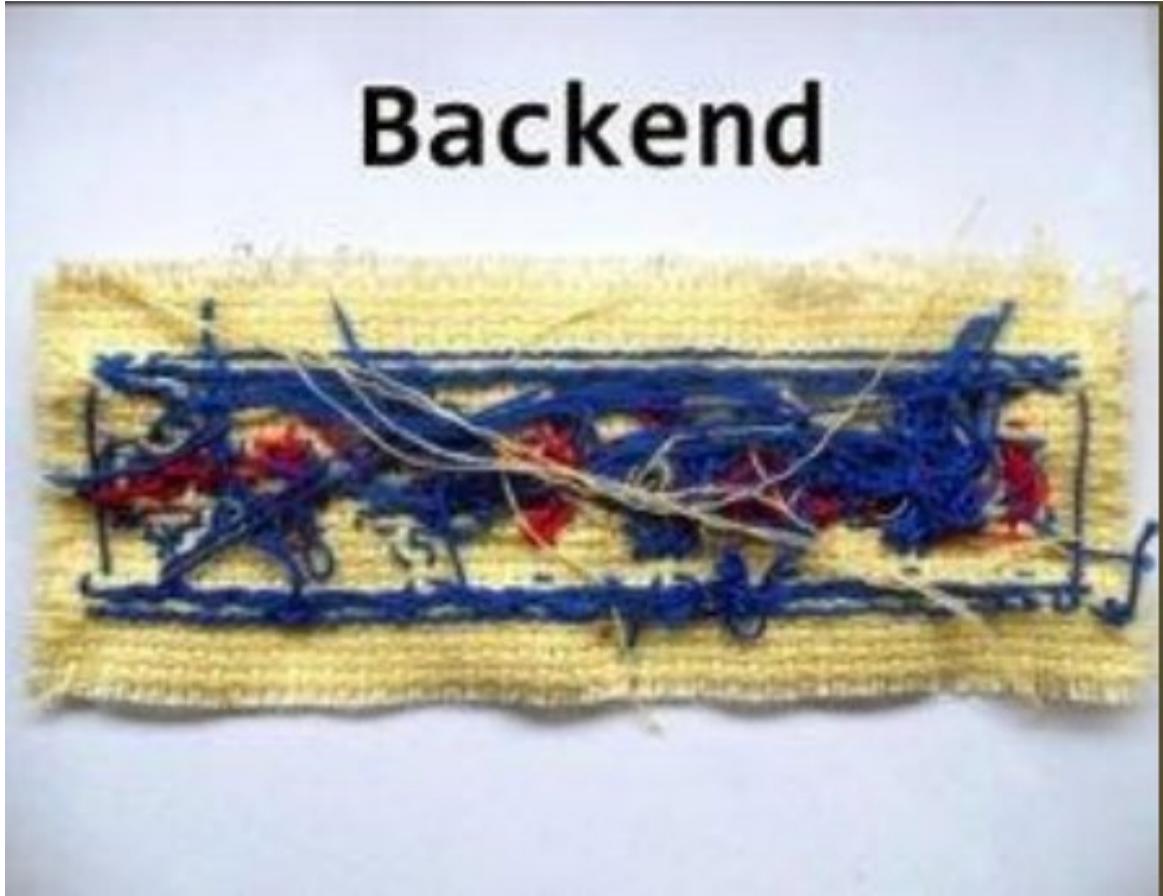
Backend Back-End adı üstünde kodlamanın arka yüz kodlaması için kullanılan bir yazılım tabidir. Bu alanda nihai kullanıcı için görsel anlamda bir farklılık sezilmese de sitenin doğru çalışması anlamında Back-End Developer'a çok iş düşer. Tíkþı Front-End'de olduğu gibi; Back-End kısmında da elbette projeler tek bir yazılım geliştirici üzerinden ilerlemiyor. Projenin oluşumunda bir analiz uzmanının data mining yapması, proje road map'ini oluþturması, proje büyüklüğüne göre farklı title ve sayıda geliştirici ile oluşturulan ekibe görevlerini deklare etmesi gereklidir. Burada proje yöneticisinin de farklı alanlarda minimum 7-8 proje bitirme tecrübesinin olması gereklidir. Yeri gelmişken bazı yanlış algılara da dejinmeye yarar var. Bir yazılım mühendisinin yazılım alanında her bir yazılım dilini veya framework'lerin tamamını sular seller gibi bilmesi diye bir şey söz konusu olamaz.

Back-end nedir konusunda Steve Jobs'un kısa ve basit tanımına katılmamak elde değil:

“Tasarım bir şeyin yalnızca nasıl göründüğü ve nasıl hissettiði ile ilgili değildir. Tasarım bir şeyin nasıl çalışlığıyla da alakalıdır..”

Özetleyeceð olursak Back-End; projenin mutfağı olan server side ile web sitesinin düzgün çalışmasını sağlayan arka plan kısmını kapsıyor ve ön yüz ile veri alışverişine cevap verip verileri saklıyor. Front-End ise; web sitesinde kullanıcıların gördüğü ve direkt etkileşimde bulundukları ön yüzü kapsayıp kullanıcının back-end katmanı ile veri alışverişi yapabilmesini sağlıyor.

Buradan özetle şöyle diyebiliriz; **Backend'in Türkçe karşılığı "Arkayüz"dür. Kullanıcıların görmediği arka yüz (Server-side) geliştiren kişidir.** Yani sistemin mimarisini oluþtururan, veri tabanı yönetimini planlayan, sunucu ayarlamalarından sorumlu, sistemin maksimum seviyede verimli ve hızlı çalışmasını sağlayan kişidir. Back-end teknolojilerinde Python, PHP, Ruby, Java, C#, ASP.NET, MySQL, MS Sql, MongoDB diye özetleyebiliriz. Teknolojiler gelişikçe bu iki kavram ortaya çıkmış ve zaman içerisinde kendi aralarında dallanmaya başladılar. Örneğin front-end kısmında CSS Developer, (yada UI/UX Developer) ve JavaScript Developer gibi alt dallar çıkmaya başladı. Back-end kısmında ise Database Developer, Java Developer yada Software/Application Developer gibi alt kırılımlar oluþmaya başladı. Her iki alanda bulunan alt kırılımlar aslında uzmanlaşmayı beraberinde getirmektedir. Veriyi üreten kısımdır.



Backend Server Side : Uygulamanın arkaplanı, mutfağıdır. Algoritmik ve mimarisel kodların yazıldığı alandır. Veritabanı işlemleri vs. backend'de gerçekleştirilir. Verinin/bilginin üretildiği yerdir.

Backend'de Kullanılan Teknolojiler : Node.JS , PHP, Ruby, Java, C#(.Net- Asp.Net Core) , Phyton , C++ , C , Scala , Golang

Bir backend bazen başka bir backend'in frontend'i olabilir

Front-End : Frontend'in Türkçe karşılığı "Önyüz"dür. Yapılma aşamasındaki bir web sitesinin ön yüzünü (client-side) HTML, CSS ve JavaScript gibi teknolojileri kullanarak web sitesinin görsel tarafını oluşturan kişilere ise front-end developer (Ön yüz geliştirici) denir. Web sitesinde kullanılacak renkler, içeriklerin yerleşimi, yazı tipinin seçilmesi ve uygulanması gibi birçok görevi bünyesinde barındırır. Yani Front-end sadece görsellendirme demek değildir. Elimizdeki verileri anlamlı hale getirdiğimiz kısımdır. Veri kullanılıyor ve anlamlı hale getiriliyorsa burası front-end'dir.



Frontend

Frontend Client Side : Uygulamanın önyüzü, grafiksel arayüzüdür. Görsel kısmıdır. Makyajıdır. Backend'de üretilen veri Frontend'e gönderilir ve uygun şekilde kullanıcıya sunulur.

Frontend'de Kullanılan Teknolojiler : • Angular • React

- Vue.js
- Bootstrap
- HTML
- CSS
- JavaScript

Web Uygulaması Geliştirme Modelleri

Kabul edilen yaklaşımı(Mimarisel) göre geliştirme yapılır.

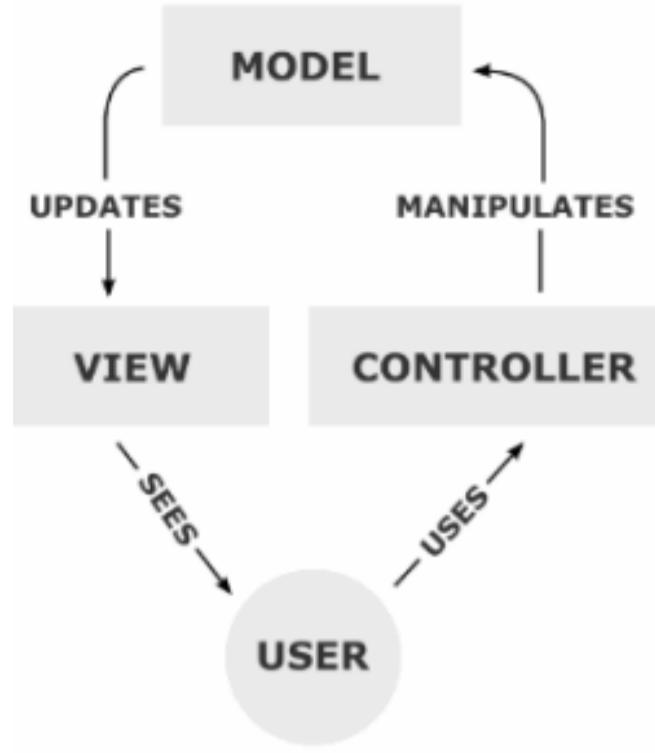
Olay Tabanlı Programlama YAKLASIMI :

- Olay güdümlü programlama veya Olay yönlendirmeli programlama da denmektedir.
- Programın akışını kullanıcı eylemlerine göre yönlendiren programlama yaklaşımıdır.
- Kullanıcı eylemlerine uygun bir şekilde olaylar tanımlanmıştır. Bu olaylar çalıştırılacak kodu barındırmaktadır.
- Uygulama hızlı bir şekilde inşa edilebilir. Lakin bakım ve sonraki gelişim süreci oldukça maliyetli olan bir yaklaşımındır. Maliyeti sebebiyle pek kullanılmaz.

Aslında bu bir web yaklaşımı değil programlama yaklaşımıdır. Her şey bir olayın etrafında gerçekleşir. Misal, Karanlık bir odada düğmeye basarak ışığın açılması buna örnektir.

Teknik olarak ele alındığında ; bu yaklaşımın iki tane aktör vardır. 1. Olay, kendisi... 2. İş,eylem... Misal web'de bir linke tıklandığında o linke gidilmesi olay, tıklanması ise istir.

Model View Controller(MVC) Yaklaşımı :



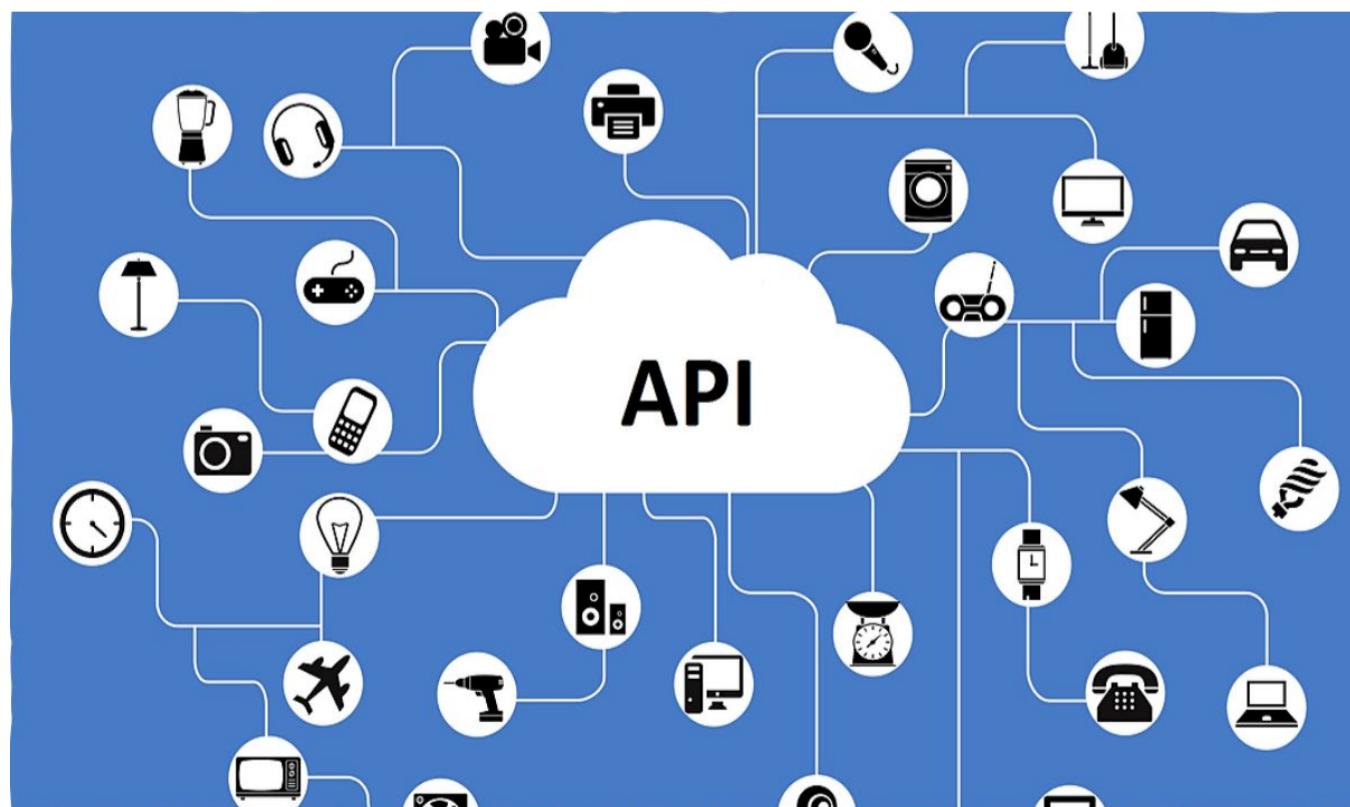
- MVC pattern'ını kullanan bir yaklaşımındır.
- Üretilen veri ile gösterim arasında bir soyutlama esas alınmıştır.
- MVC pattern'ı Microsoft tarafından üretilmemiştir. Böyle bir yanlış algı vardır. Bir tasarım desenidir.

3 tane katmanı bulunur ; Model-View-Controller

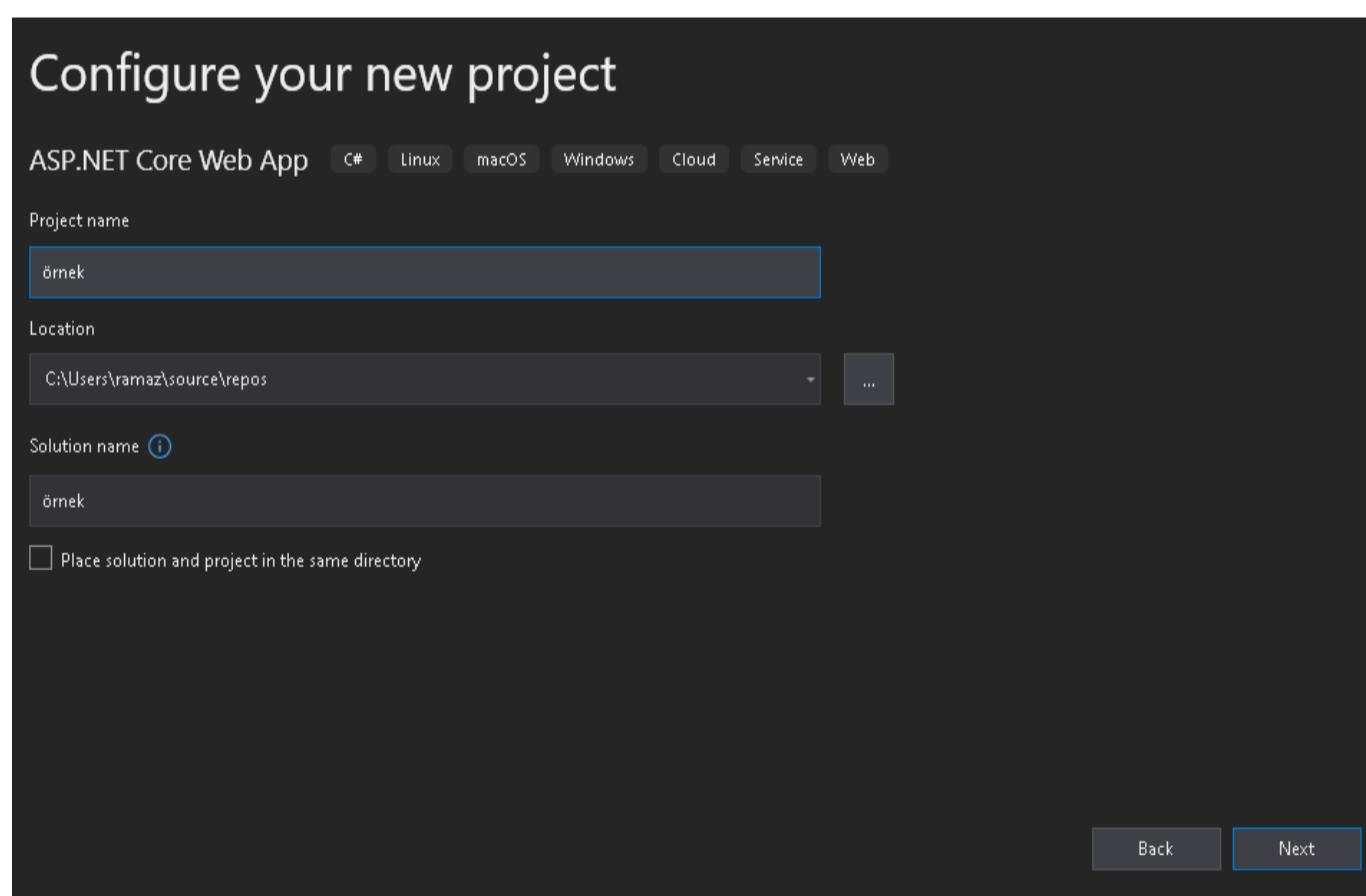
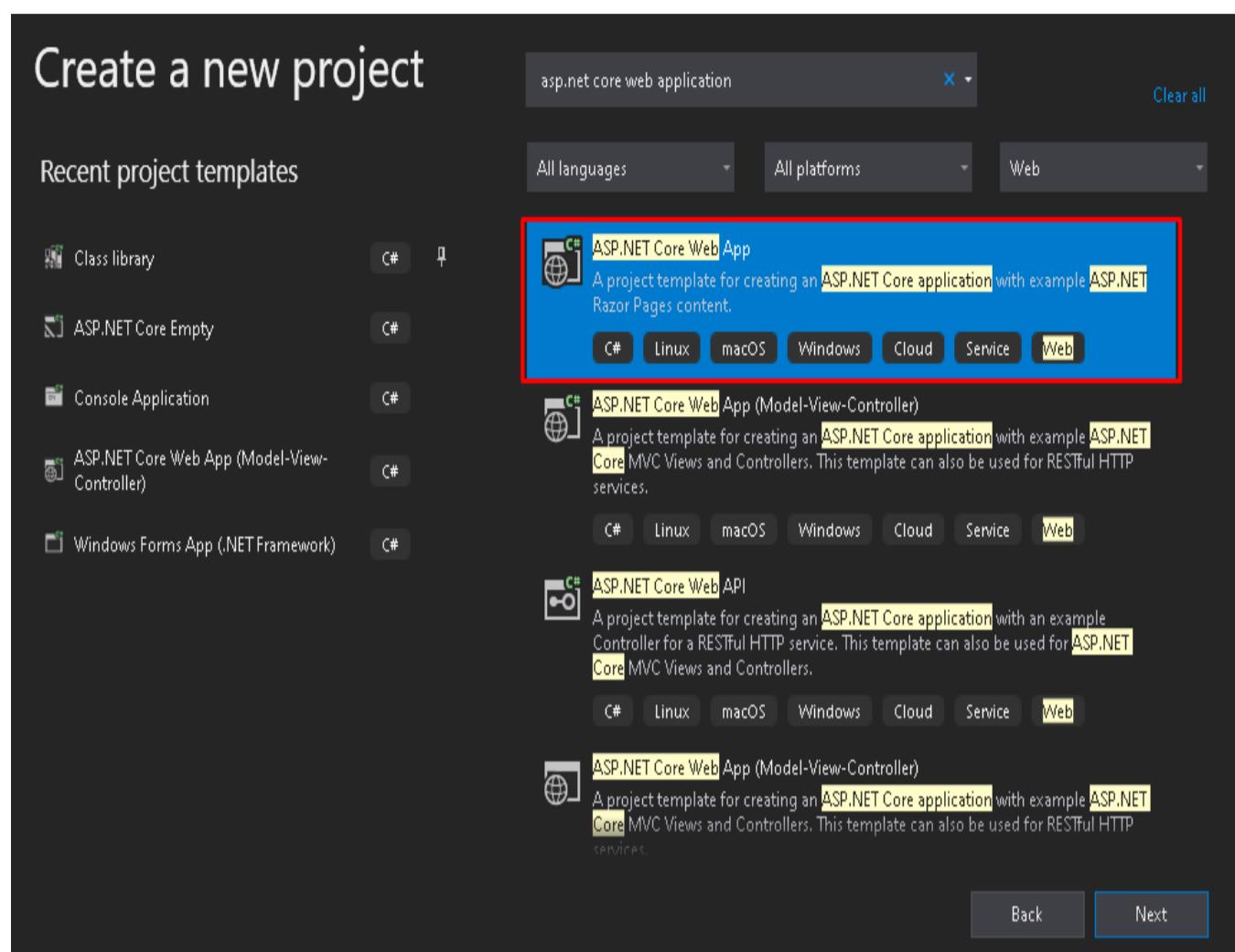
- MVC, Microsoft'un kurulduğu yillardan (1979 yılından) beri tasarlanmış bir kalıptır.

API : Application Programming Interface

- Web'de çalışabilen ve web uygulamaları, işletim sistemleri, veritabanları, donanımlar yahut yazılım kütüphaneleri ile iletişim kurabilen bir arayüzdür.
- Direkt olarak Web Uygulaması yaklaşımıdır diyemeyiz. Lakin genellikle web tabanlı uygulamalarda client ve server arasındaki iletişimini sağlayan bir sözleşme olarak kullanılmaktadır. Bu forma Web API ismi verilmektedir.



Asp.NET Core Proje Oluşturma ve Dosya Yapısı



Additional information

ASP.NET Core Web App C# Linux macOS Windows Cloud Service Web

Target Framework ⓘ

.NET 5.0 (Current)

Authentication Type ⓘ

None

Configure for HTTPS ⓘ

Enable Docker ⓘ

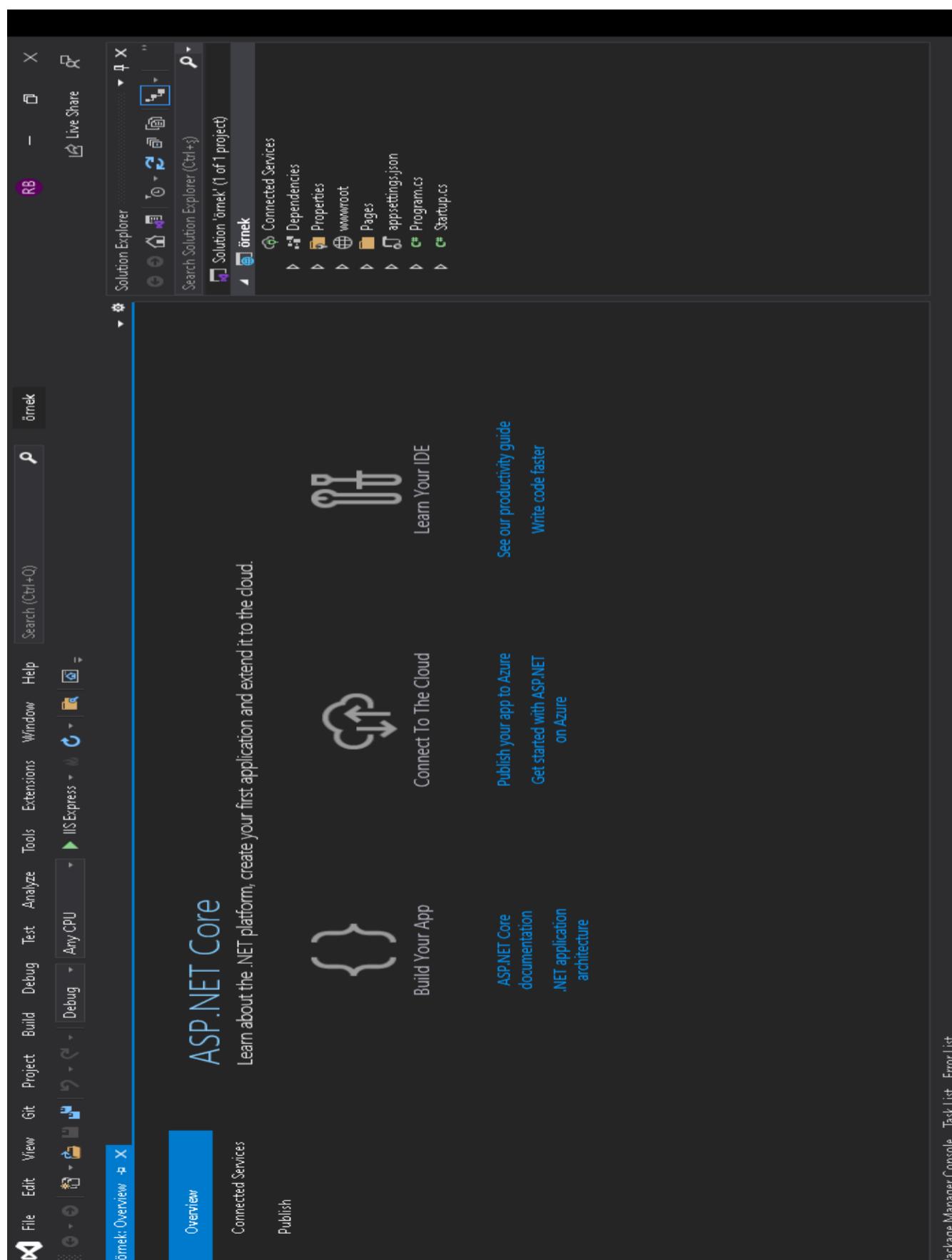
Docker OS ⓘ

Linux

Enable Razor runtime compilation ⓘ

Back

Create



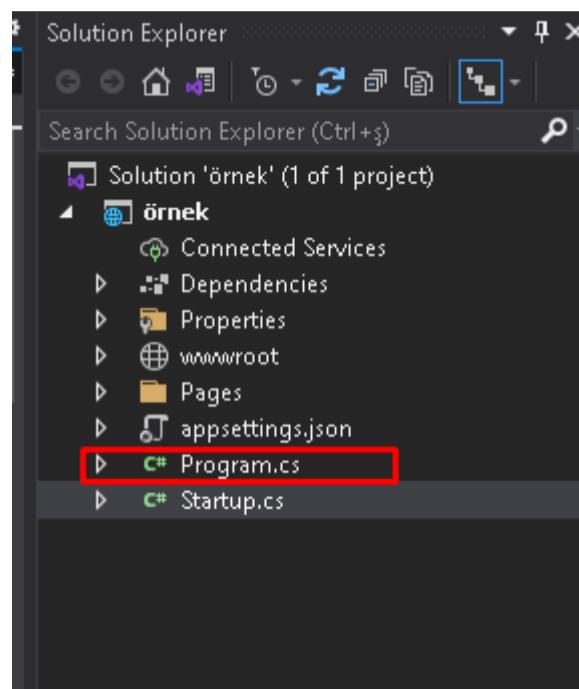
Program.cs Nedir :

Asp.Net Core'un fitratı konsol uygulamasıdır.

Dolayısıyla Program.cs dosyası var .

Peki bu Program.cs ne işe yarıyor? Esasında yapılandırılmış varsayılanlarla burada web barındırıcıyı oluşturabilmek için bir yöntem ,bir metot çağrıiyor ve bu metot → **CreateHostBuilder(args).Build().Run();**. Kendi dahilinde uygulama sunucusunu kaldırma işlemlerlerini burda yapıyoruz.

Asp.Net Core kendi dahilinde sunucu barındırır. İşte o sunucuyu ayağa kaldırdığı nokta Program.cs dosyasıdır.



The screenshot shows the Visual Studio IDE with the 'Startup.cs' file open in the code editor. The code is as follows:

```
1  using Microsoft.AspNetCore.Hosting;
2  using Microsoft.Extensions.Configuration;
3  using Microsoft.Extensions.Hosting;
4  using Microsoft.Extensions.Logging;
5  using System;
6  using System.Collections.Generic;
7  using System.Linq;
8  using System.Threading.Tasks;
9
10 namespace OrnekProje
11 {
12     public class Program
13     {
14         public static void Main(string[] args)
15         {
16             CreateHostBuilder(args).Build().Run();
17         }
18
19         public static IHostBuilder CreateHostBuilder(string[] args) =>
20             Host.CreateDefaultBuilder(args)
21                 .ConfigureWebHostDefaults(webBuilder =>
22                     {
23                         webBuilder.UseStartup<Startup>();
24                     });
25     }
26 }
27
```

Annotations in the code editor highlight specific parts of the code with red boxes and arrows:

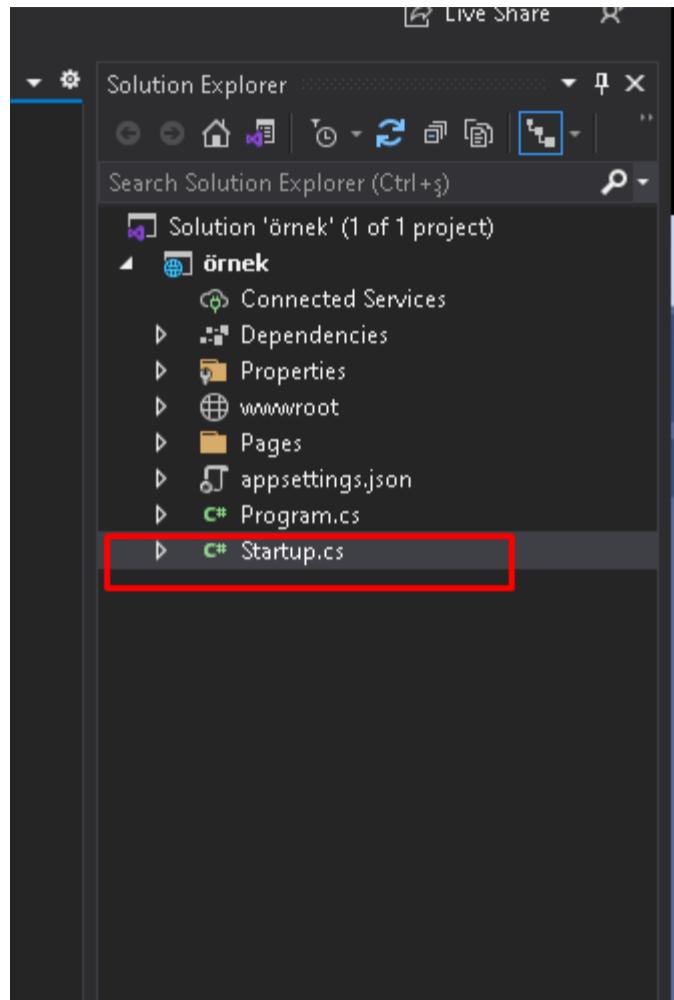
- A red arrow points from the text "Program.cs içerisinde ayaga kaldırılmış host'un kullanacağı konfigürasyonlar neden alacağını bildirmektedir." to the line `public static void Main(string[] args)`.
- A red arrow points from the text "Asp.NET Core kendi dahili filtrelerinde sunucu barındırır demektir. İste o sunucuya ayaga kaldırıldığı nokta bu Program.cs dosyasıdır." to the line `CreateHostBuilder(args).Build().Run();`.
- A red box surrounds the entire configuration logic starting from line 19, labeled "Temel konfigürasyon sınıfımızdır.".

Startup.cs Nedir?

Web Uygulamasında belirli konfigürasyonları yaptığımız dosyadır. Buradaa temelde iki tane fonksiyon vardır. 1.

ConfigureServices(); → Servis Konfigürasyonu

2. **Configure();** → Konfigürasyon



```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace OrnekProje
{
    public class Startup
    {
        public void ConfigureServices(IServiceCollection services)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }

            app.UseRouting();

            app.UseEndpoints(endpoints =>
            {
                endpoints.MapGet("/", async context =>
                {
                    await context.Response.WriteAsync("Hello World!");
                });
            });
        }
    }
}
```

Bu uygulamada kullanılcak servislerin��ndiği konfigüre edildiği metottur.

Belirli işlere odaklanmış ve o işin sorumluluğum üstlenmiş kütüphaneler/sınıflar vs. servis = modul = kütüphane

appsettings.json Nedir?

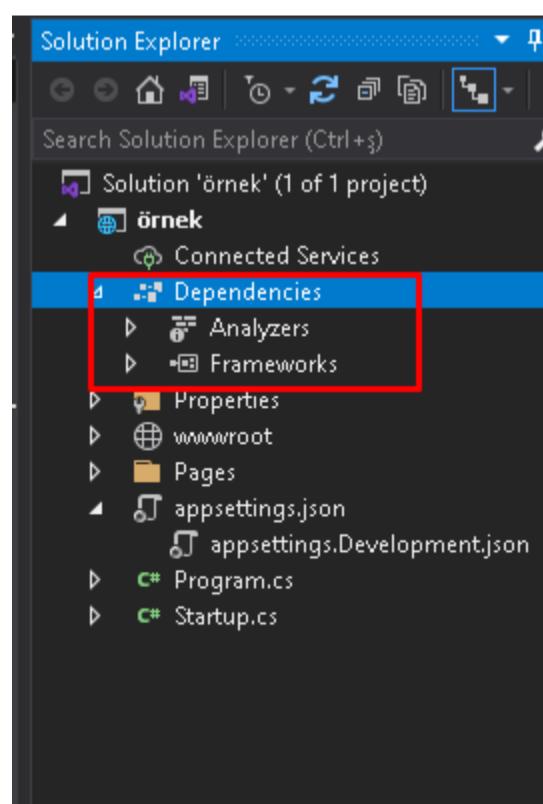
Uygulamada belirli değerleri tuttuğumuz bir configurasyon dosyasıdır. Misal ; Veri tabanıyla ilgili bir connectionstringi burda tutarız uygulamanın heryerinden ulaşmak için. Yönetimi tek ele almak için yazılımda kullanılması gereken metinsel static değerleri burda tutarız. Yani bu değer değiştirilmesi gerektiğinde kodun heryerinde bunuuygulamak yerine tek bir yerde değiştirerek her yere bu değişikliği sunmak için .

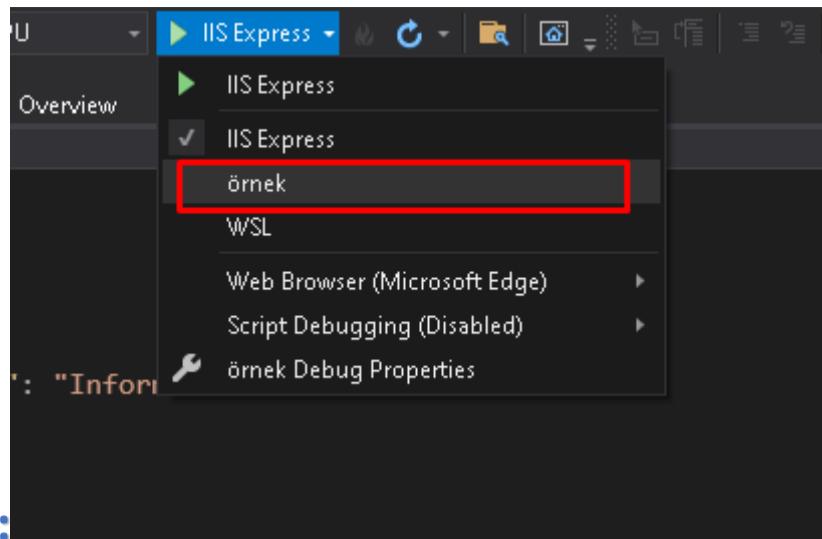
The screenshot shows the Visual Studio interface. The top menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a search bar. The status bar at the bottom indicates '115%', 'No issues found', 'Ln: 11 Ch: 1 SPC CRLF'. The main area consists of two panes: the left pane shows the code editor with 'appsettings.json' containing configuration settings, and the right pane shows the Solution Explorer with project files like 'Connected Services', 'Dependencies', 'Properties', 'wwwroot', 'Pages', 'Program.cs', and 'Startup.cs'. A specific file, 'appsettings.json', is highlighted in the Solution Explorer.

```
1  {
2    "Logging": {
3      "LogLevel": {
4        "Default": "Information",
5        "Microsoft": "Warning",
6        "Microsoft.Hosting.Lifetime": "Information"
7      }
8    },
9    "AllowedHosts": "*"
10 }
```

Dependencies Nedir?

Referansların, Bağımlılıkların, uygulamada kullanılan harici kütüphanelerin bulunduğu/yönetildiği yerdir.

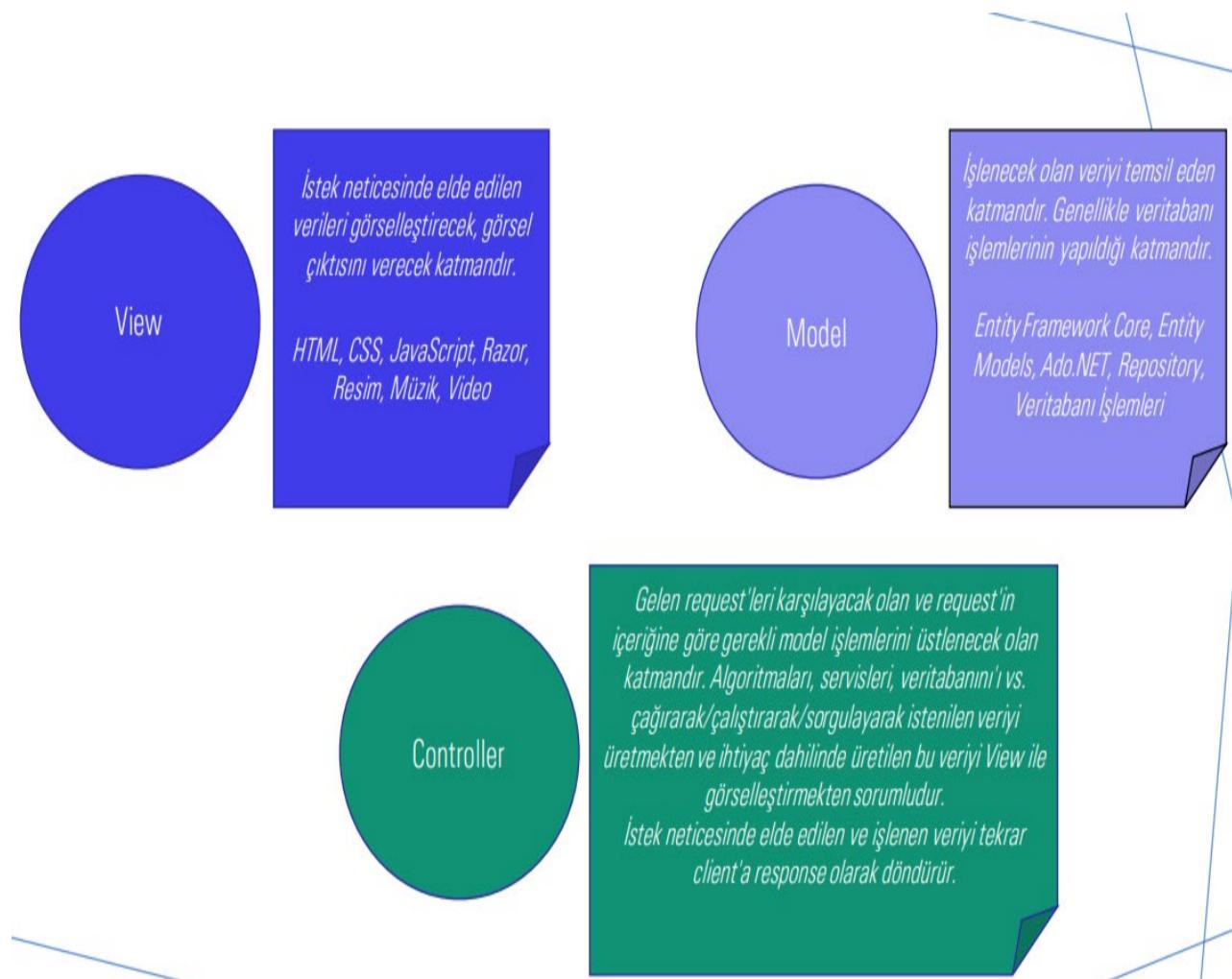


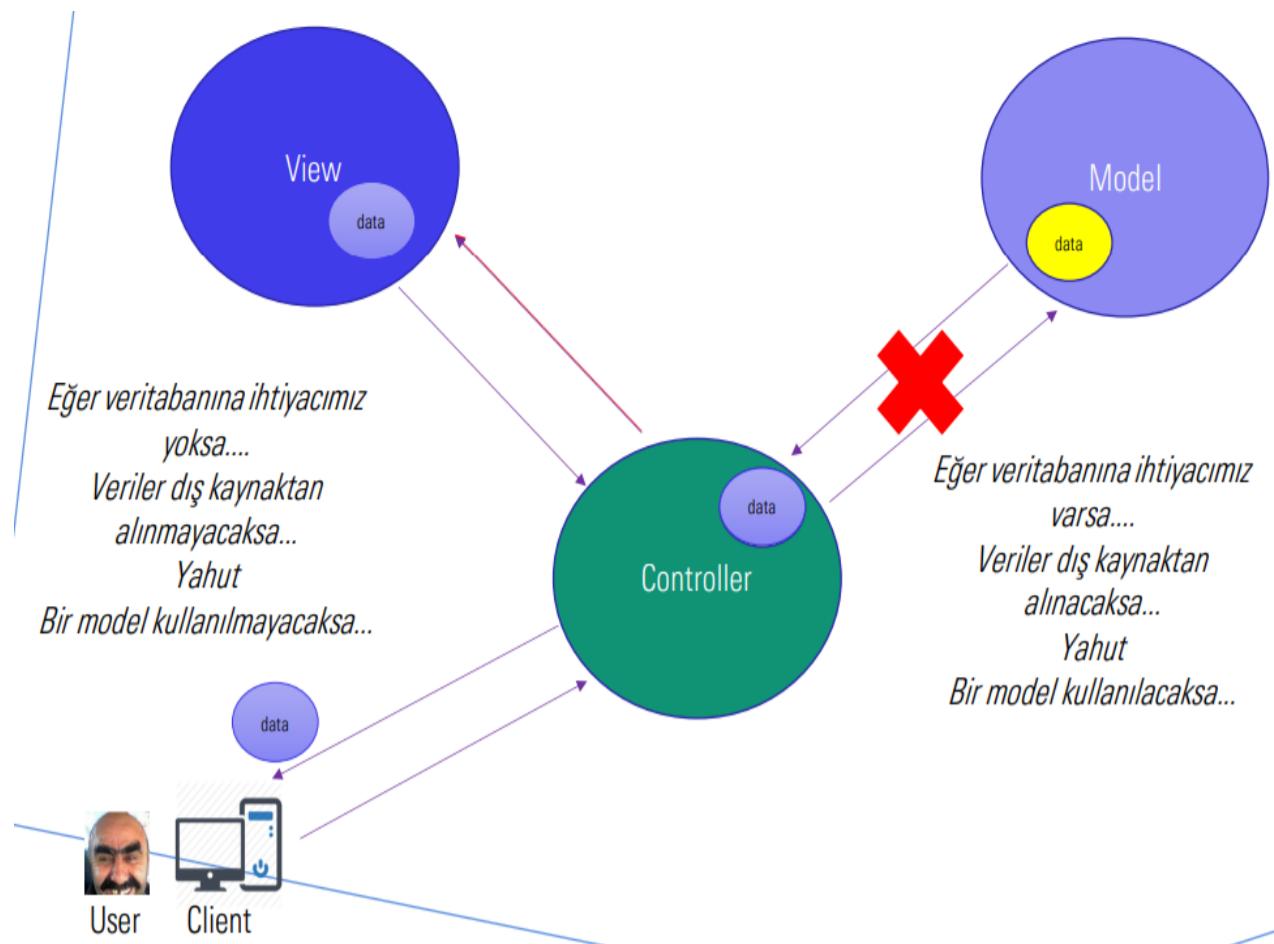


Kestrel Sunucu Seçimi:

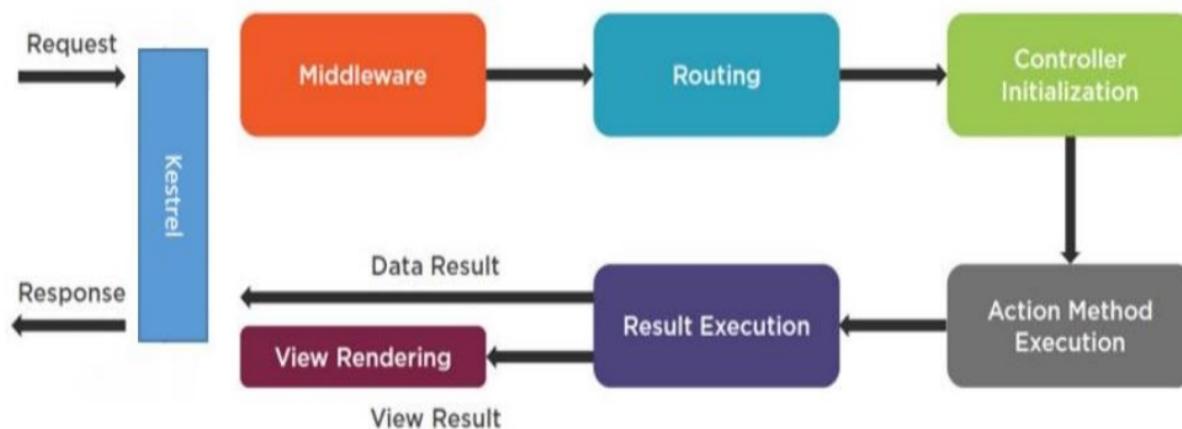
MVC(MODEL – VIEW – CONTROLLER)

- MVC, birbirinden bağımsız üç katmanı esas alan bir Mimarisel Desen(Architectural Pattern)'dir.
- Özünde Observer, Decorator gibi design pattern'ları kullanan bir mimarisel desendir.
- Microsoft bu desen üzerine oturtulmuş Asp.NET Core MVC mimarisini geliştirmiştir.





ASP.NET CORE MVC PIPELINE



CONTROLLER

Action Types (Türleri):

ViewResult : Response olarak bir View dosyasını(.cshtml) render etmemizi sağlayan action türüdür.

PartialViewResult :

Yine bir View dosyasını(.cshtml) render etmemizi sağlayan bir action türüdür.

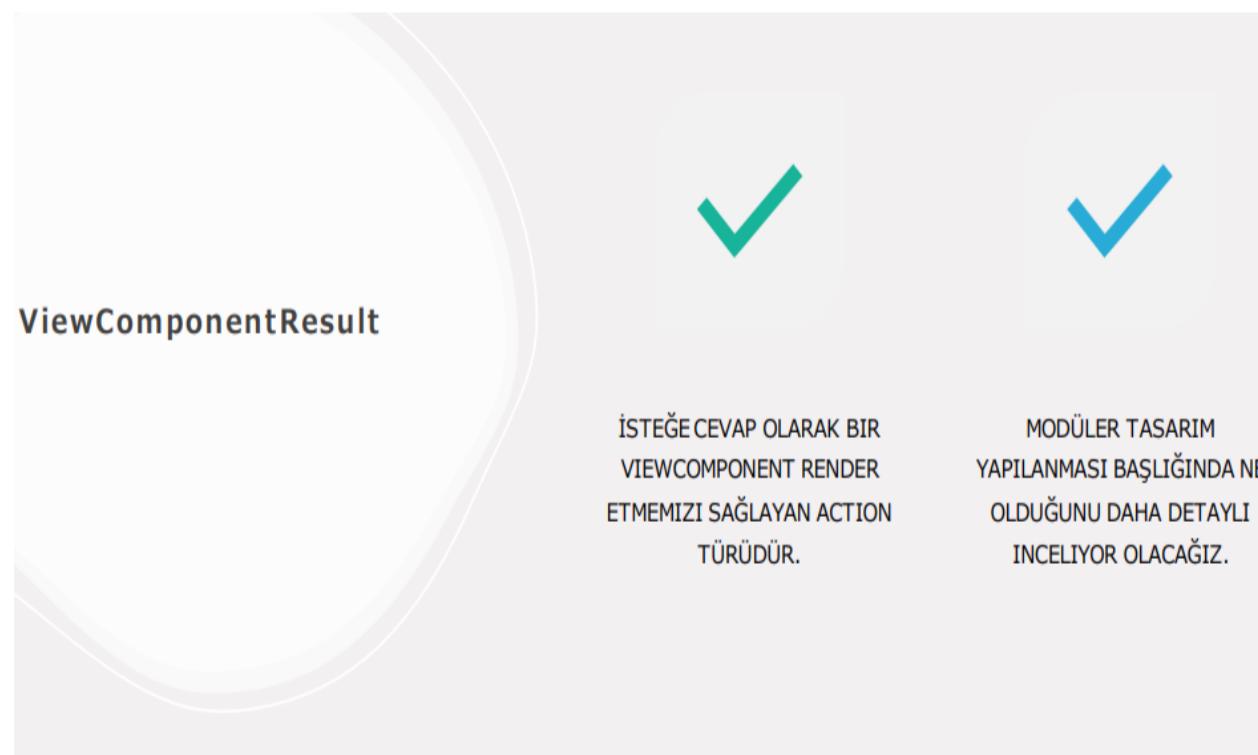
ViewResult'tan temel farkı, client tabanlı yapılan Ajax isteklerinde kullanıma yatkın olmasıdır.

Teknik fark ise ViewResult _ViewStart.cshtml dosyasını baz alır. Lakin PartialViewResult ise ilgili dosyayı baz almadan render edilir.

JsonResult : Üretilen datayı JSON türüne dönüştürüp döndüren bir action türüdür.

EmptyResult : Bazen gelen istekler neticesinde herhangi bir şey döndürmek istemeyebiliriz. Böyle bir durumda EmptyResult action türü kullanılabilir .

ContentResult : İstek neticesinde cevap olarak metinsel bir değer döndürmemizi sağlayan action türüdür.

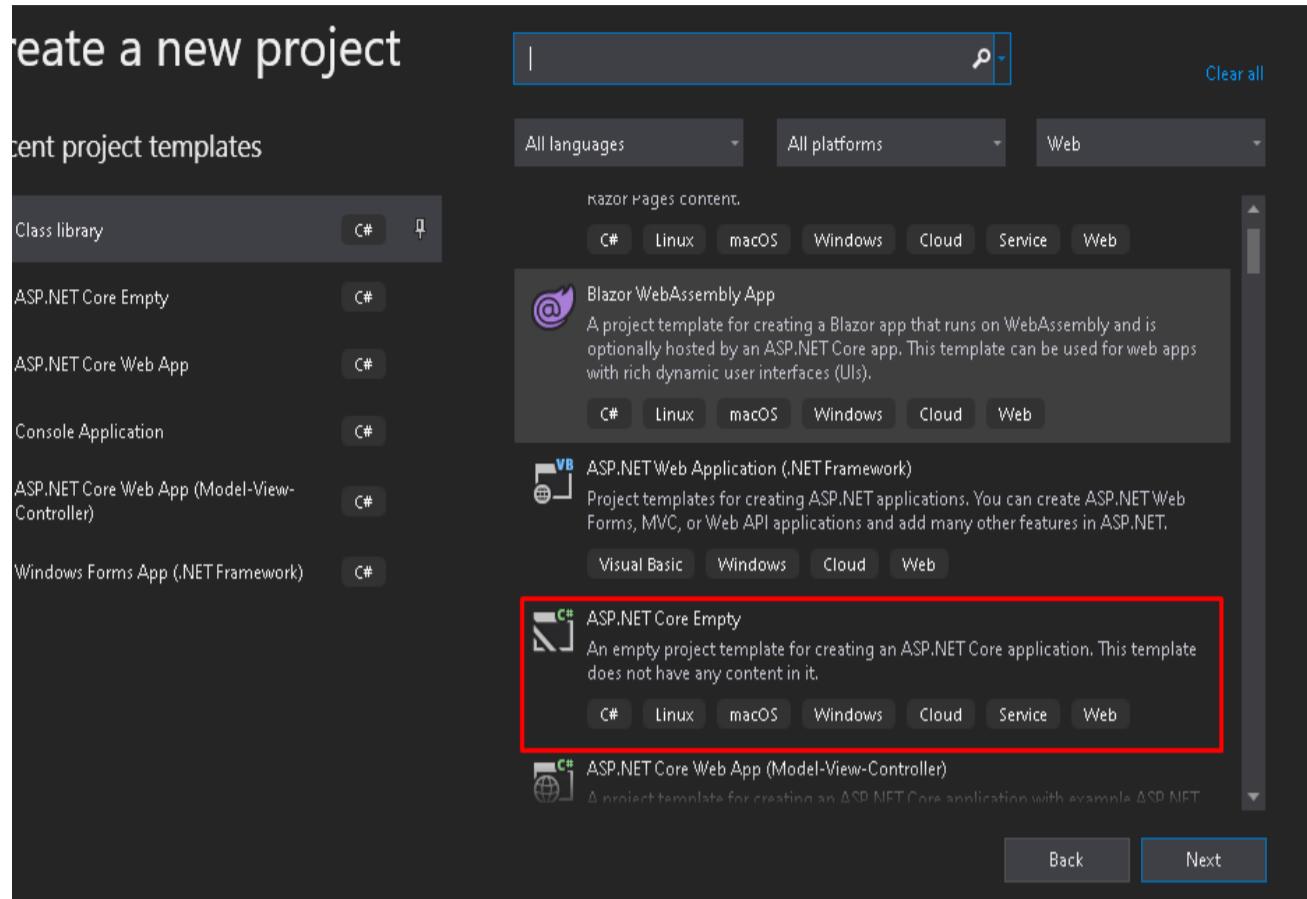


ActionResult : Gelen bir istek neticesinde geriye döndürülecek action türleri değişkenlik gösterebildiği durumlarda kullanıldığı bir action türüdür.

ActionResult, tüm action türlerini karşılayan ana türdür.

Mvc Proje Altyapısı oluşturma & Temel

Konfigürasyonlar



1) Startup.cs 'de Mvc davranışını sergilemesini tanımlama (ConfigureServices())

```
13 public class Startup
14 {
15     // This method gets called by the runtime. Use this method to add services to the container.
16     // For more information on how to configure your application, visit https://go.microsoft.com/fwlink/?LinkID=398940
17     public void ConfigureServices(IServiceCollection services)
18     {
19         // services.AddControllers(); // Sadece controller yapılanması ile oluşturulması isteniyorsa
20         services.AddControllersWithViews(); // view yapılanması ile oluşturulması isteniyorsa. Artık uygulma
21         // mvc mimarisini ile kalkındırılacağını öğrendi. Artık uygulama Mvc davranışını sergileyebilcektir
22     }
}
```

2) Startup.cs' da Configure()

The screenshot shows the Visual Studio IDE with the Startup.cs file open in the editor. The code is written in C# and defines the configuration logic for an ASP.NET Core application.

```
Startup.cs*  önek uygulama: Overview
önek uygulama
19     // services.AddControllers(); // Bu controller yapılmaması ile oluşturulması isteniyorsa
20     services.AddControllersWithViews(); // view yapılmaması ile oluşturulması isteniyorsa. Artık uygulama
21     // mvc mimarisini kalkındırılacağını öğrendi. Artık uygulama Mvc davranışını sergileyebilcektir
22 }
23
24 // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
25 public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
26 {
27     if (env.IsDevelopment())
28     {
29         app.UseDeveloperExceptionPage(); // Gelen isteğin rotası bu middleware sayesinde belirlenir.
30     }
31     app.UseRouting();
32
33     app.UseEndpoints(endpoints =>
34     {
35         endpoints.MapGet("/", async context =>
36         {
37             await context.Response.WriteAsync("Hello World!");
38         });
39     });
40 }
41 }
42 }
43 }
```

A red box highlights the `app.UseRouting();` line, and a red arrow points from this box to a callout bubble containing the following text:

Endpoint : Yapılan isteğin varış noktası, Url
İstek adresi... Bu uygulamaya gelen isteklerin
hangi rotalar/şablonlar eşliğinde gelebilceğini
buradan bildireceğiz

Önemek

Default olan endpoint şeması

Bu uygulamaya yapılabilecek olan istekler bu şemaya uygun bir şekilde yapılır.

```

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    app.UseRouting();

    app.UseEndpoints(endpoints =>
    {
        //endpoints.MapGet("/", async context =>
        //{
        //    await context.Response.WriteAsync("Hello World!");
        //});

        endpoints.MapDefaultControllerRoute(); // istek yapacağımız varsayılan rotayı belirliyor
    });
}

```

Request

<https://www....com/ununter/getir/{id}>

<https://www....new/x/y>

<https://....com/>

Endpoint tanımlamasında süslü parantez içerisinde tanımlanabilmektedir. Bu parametreler herhangi bir isimde olabilir.

{controller}{action}{ramazan}

controller ve action => Ön tanımlı olan parametrelerdir.

Q: (extension) ControllerActionEndpointConventionBuilder

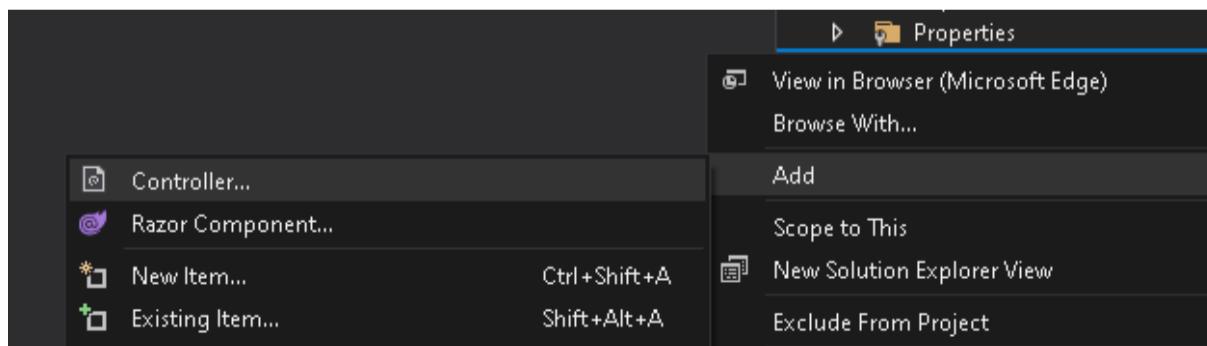
Microsoft.AspNetCore.Routing.EndpointRouteBuilder.MapDefaultControllerRoute()

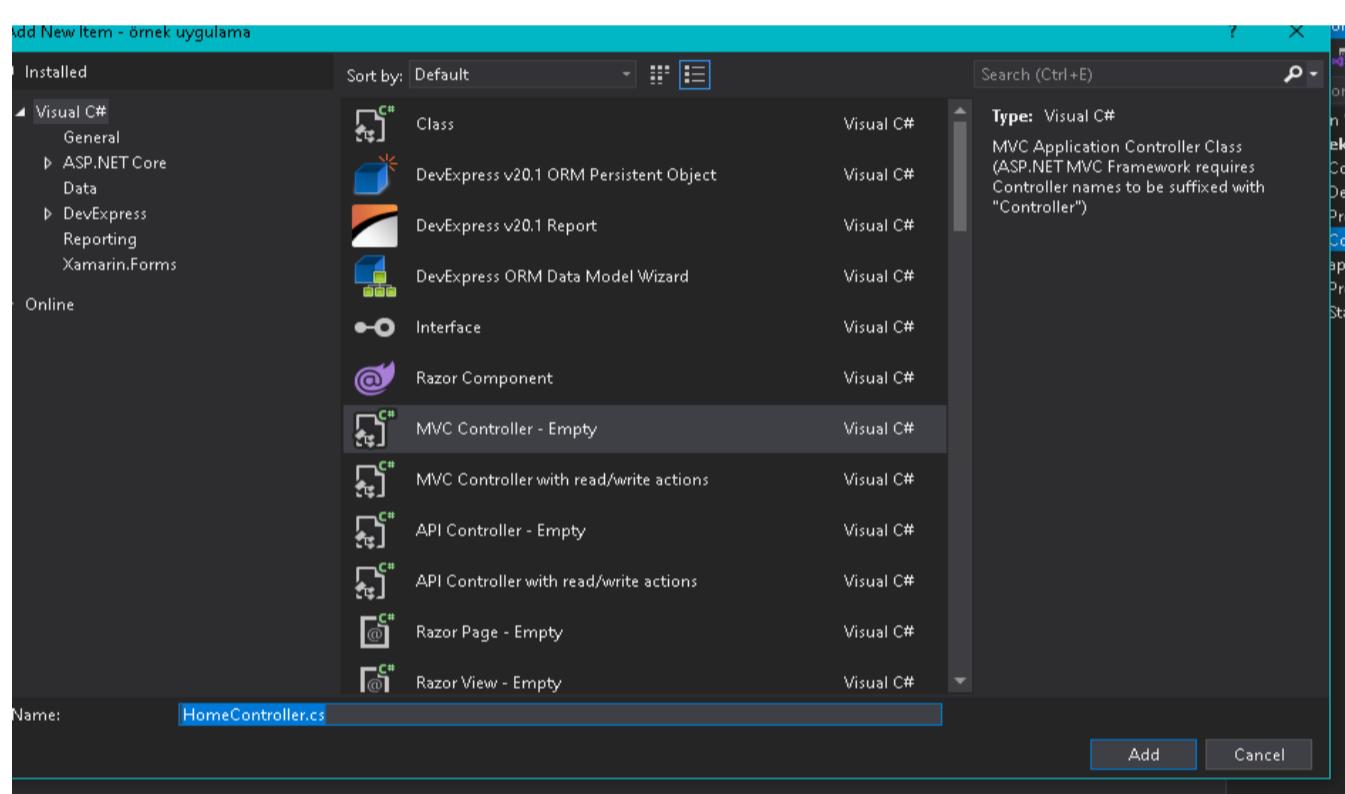
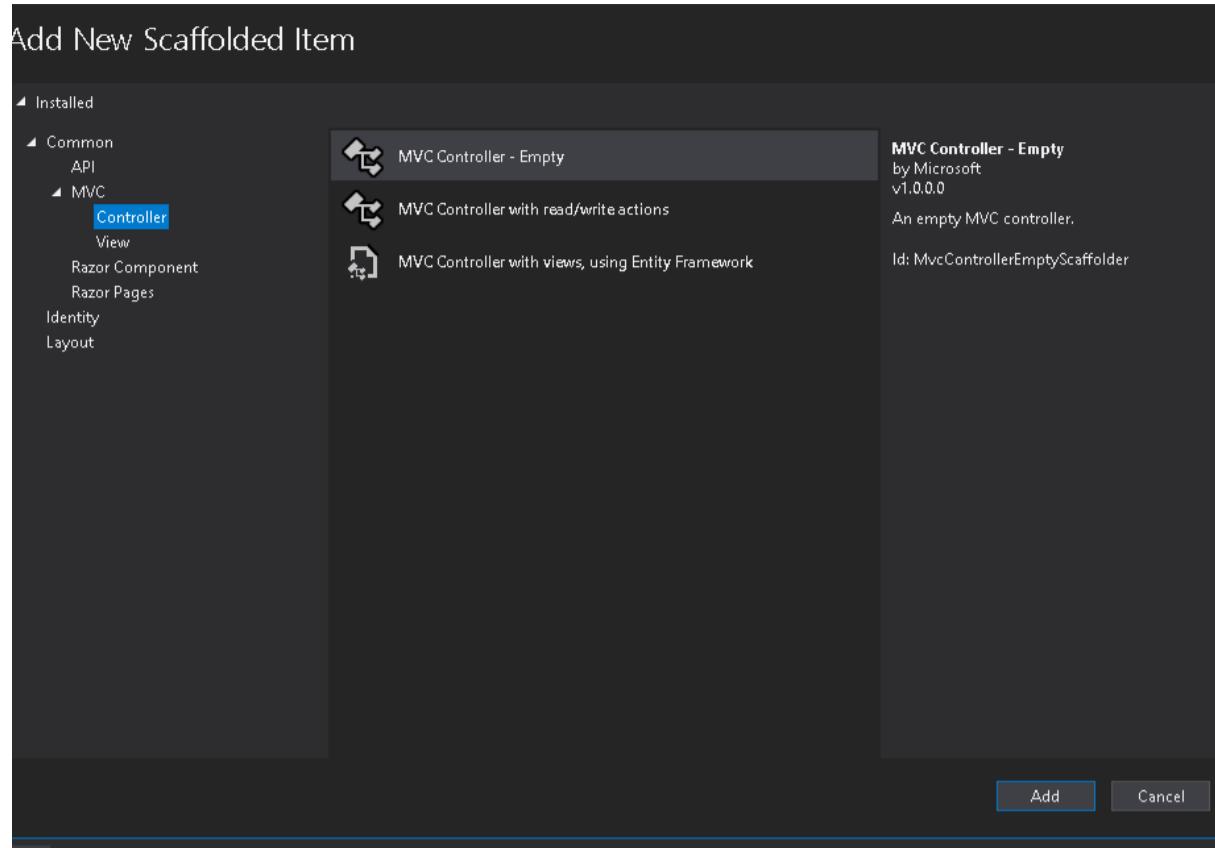
Adds endpoints for controller actions to the Microsoft.AspNetCore.Routing.EndpointRouteBuilder and adds the default route {controller=Home}/{action=Index}/{id?}.

Returns:

An ControllerActionEndpointConventionBuilder for endpoints associated with controller actions for this route.

Controller: Uygulamaya gelen istekleri karşılayabilmek için kullandığımız sınıflardır. Bu sınıflar Controllers Klasörü altında tutulurlar





HomeController.cs

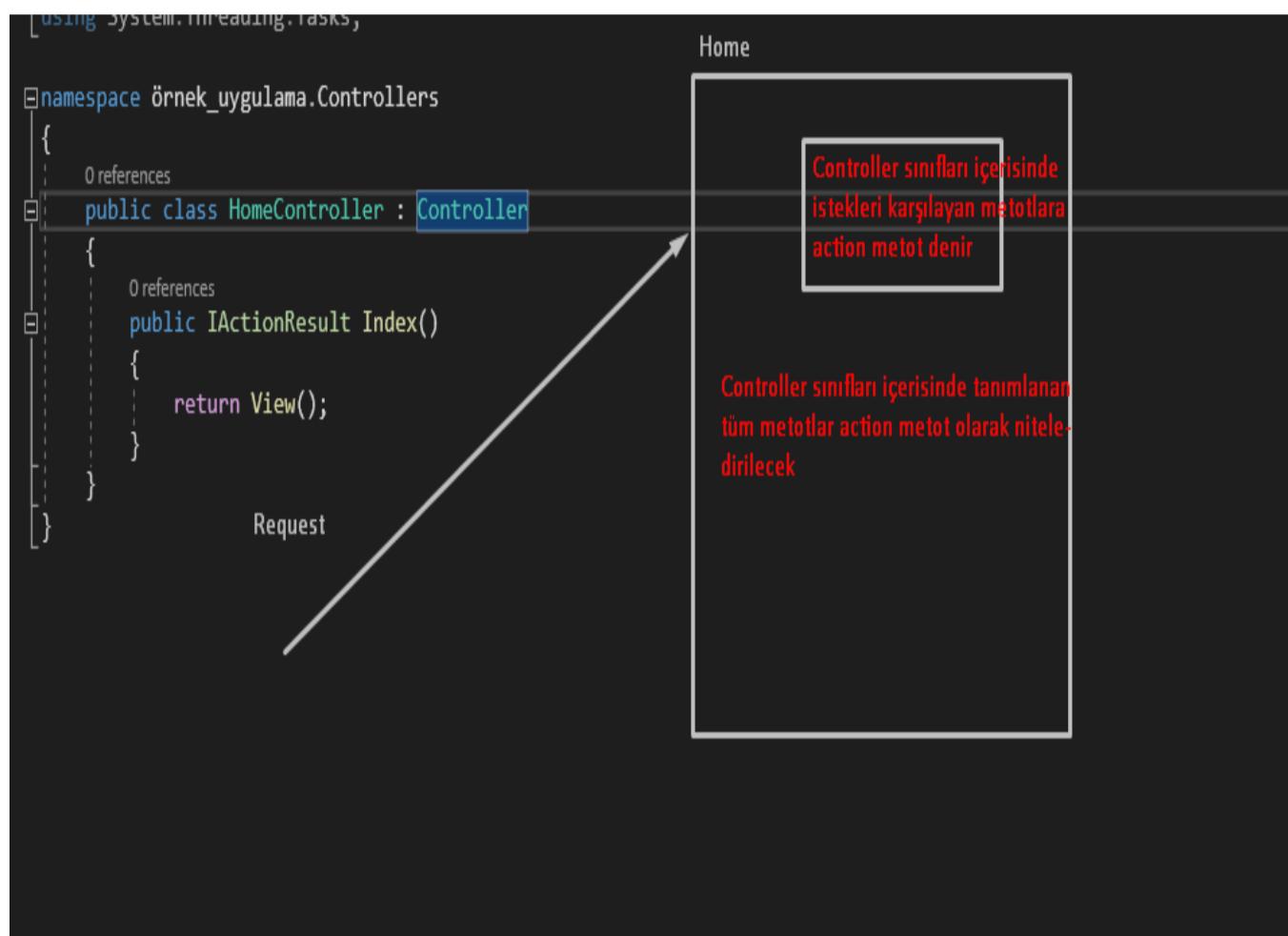
```
1 using Microsoft.AspNetCore.Mvc;
2 using System;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Threading.Tasks;
6
7 namespace örnek_uygulama.Controllers
8 {
9     public class HomeController : Controller
10    {
11        public IActionResult Index()
12        {
13            return View();
14        }
15    }
16}
17
```

```

using System.Threading.Tasks;
namespace örnek_uygulama.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

Bir sınıf request alabilir ve response döndürebilir yani Controller yapabilmek için o sınıfı controller class'ından türetmemiz gerekmektedir.



Asp.Net Core Mvc Çalışma :

```

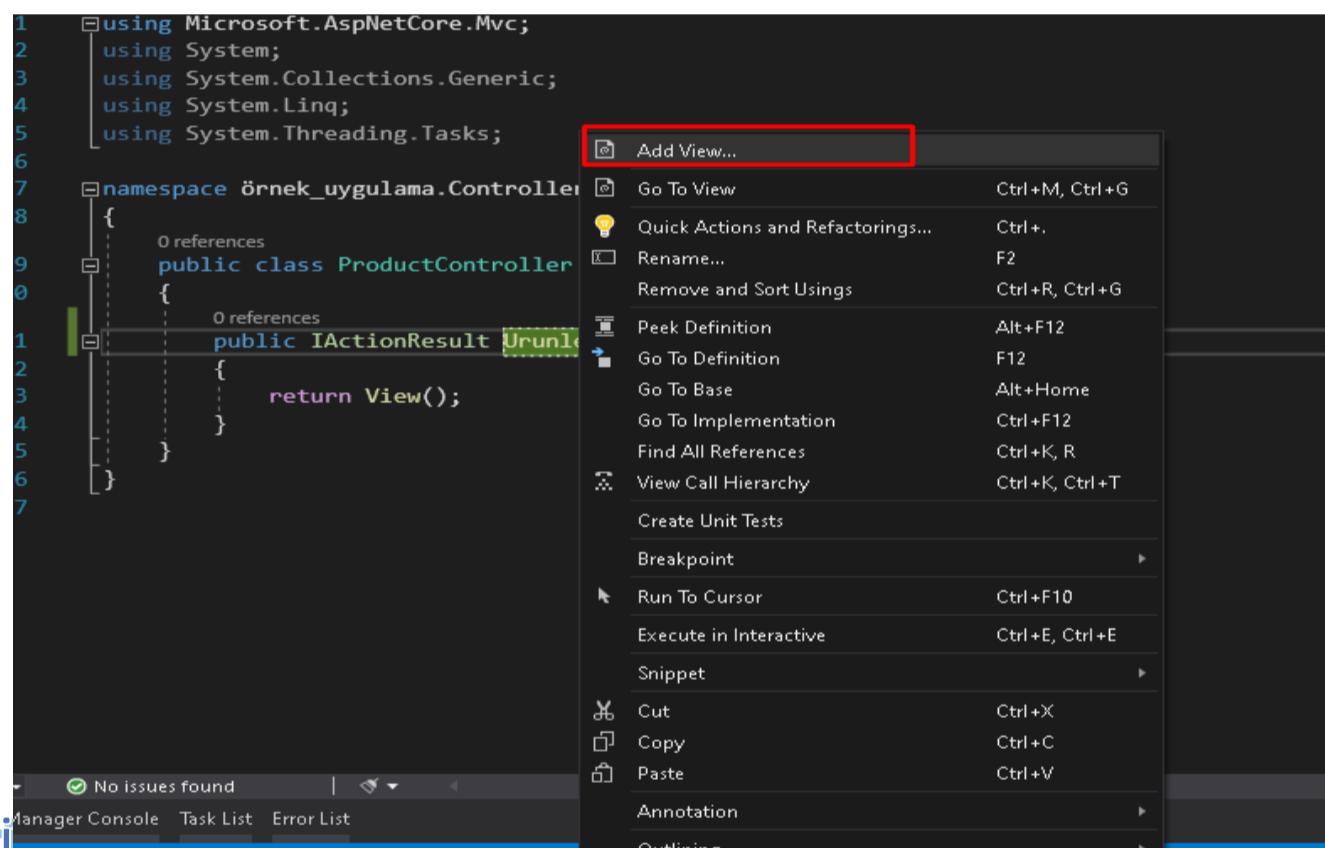
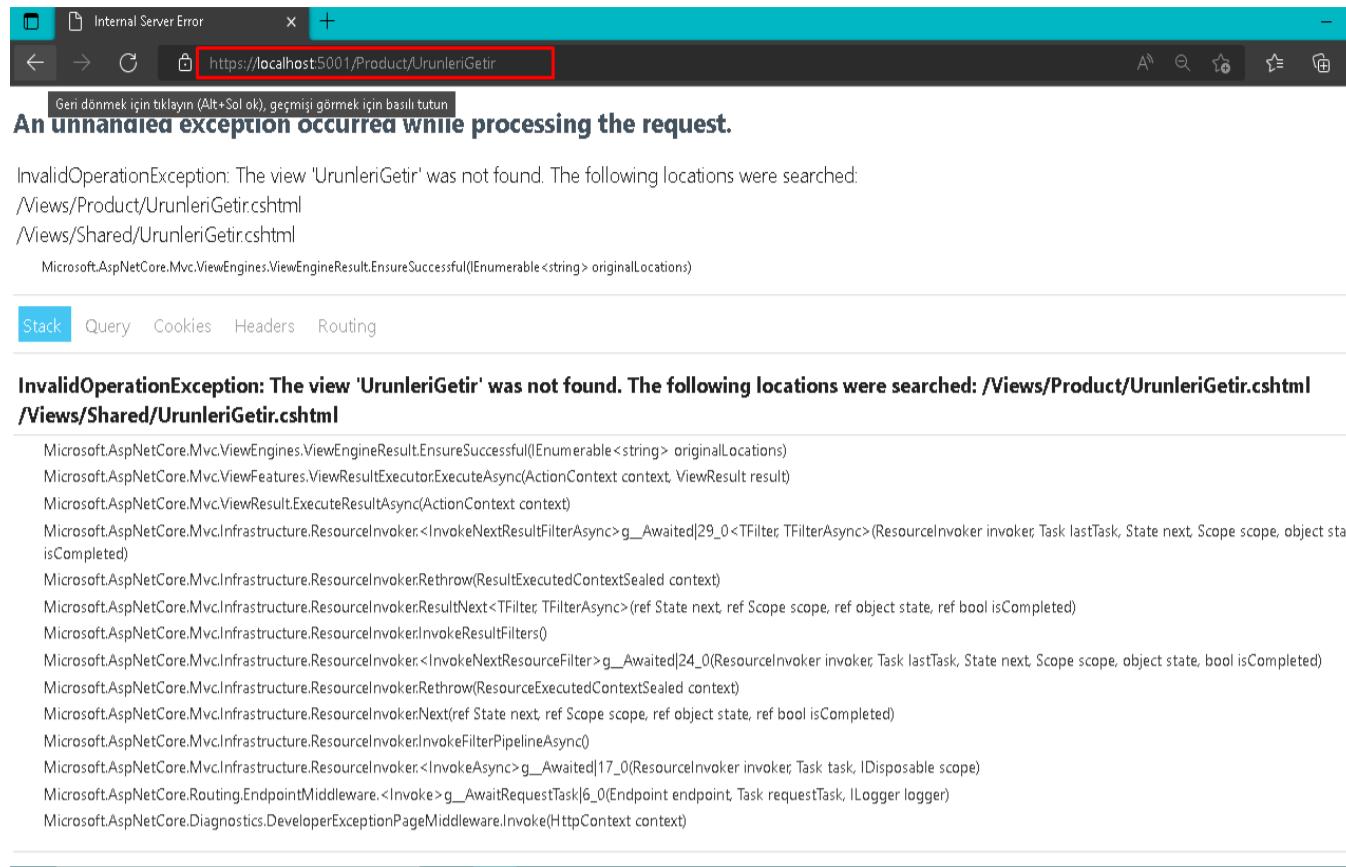
ProductController.cs  X  HomeController.cs
örnek uygulama  örnekte_uygulama.Controllers.ProductController  Startup
1  using Microsoft.AspNetCore.Mvc;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Threading.Tasks;
6
7  namespace örnek_uygulama.Controllers
8  {
9      public class ProductController : Controller
10     {
11         public IActionResult ÜrünleriGetir()
12         {
13             return View();
14         }
15     }
16 }
17

```

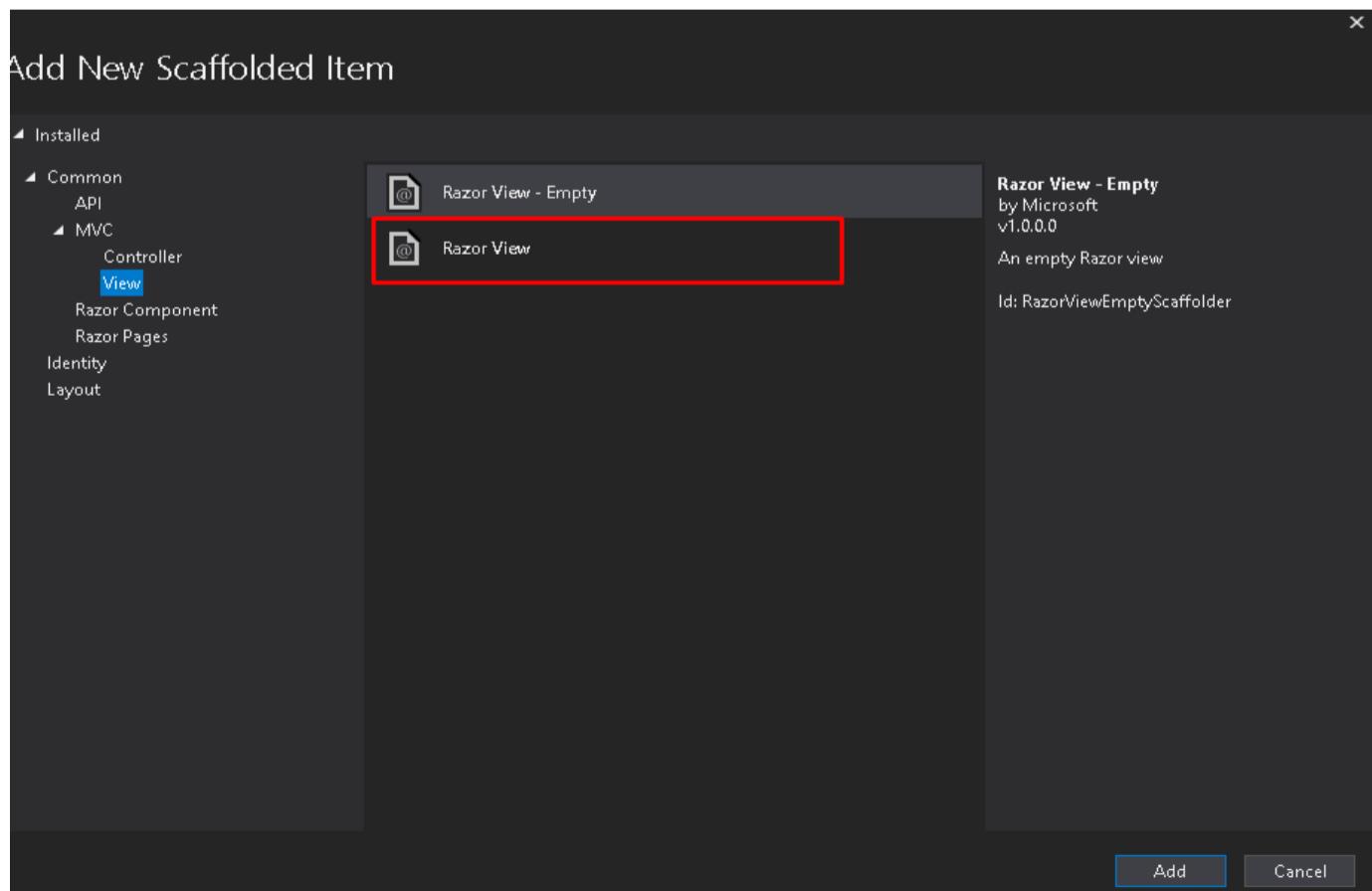
No issues found | Call Stack

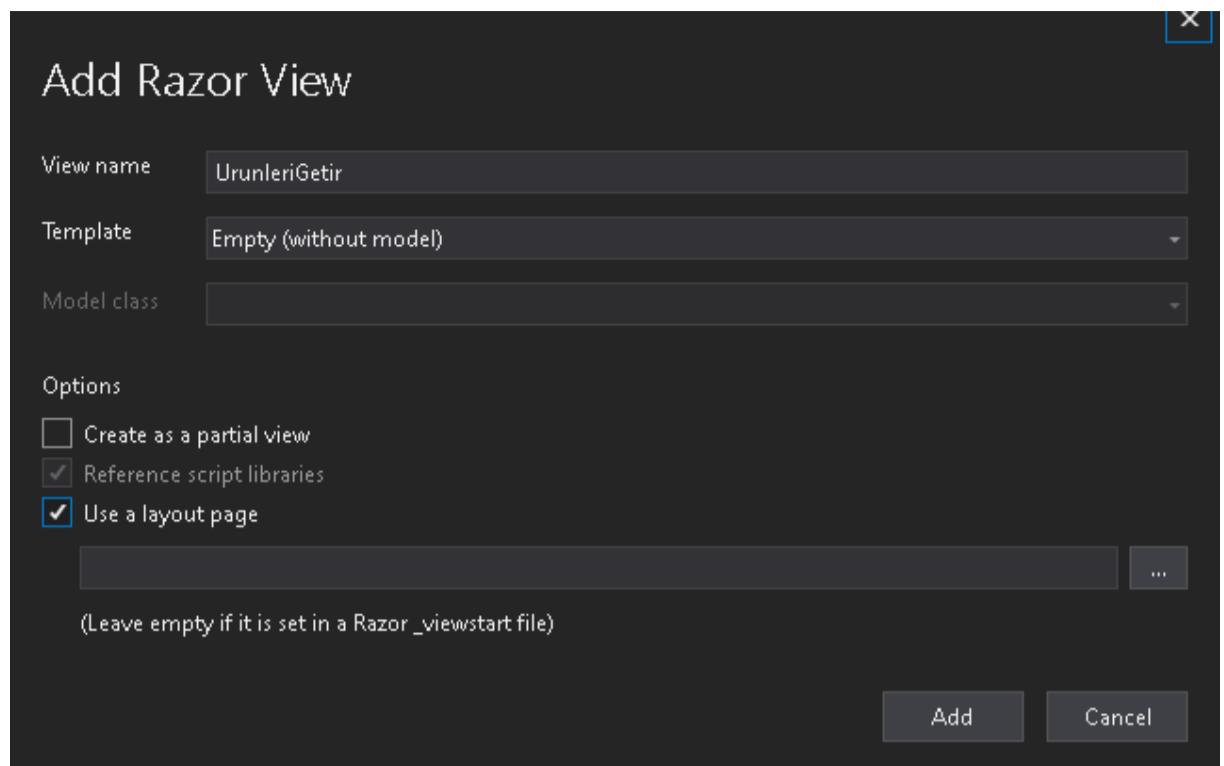
Show output from: Debug

--- End of stack trace from previous location ---
at Microsoft.AspNetCore.Mvc.Infrastructure.ResourceInvoker.<InvokeAsync>g__Awaited|17_1
at Microsoft.AspNetCore.Routing.EndpointMiddleware.<Invoke>g__AwaitRequestTask|6_0(Endi
at Microsoft.AspNetCore.Diagnostics.DeveloperExceptionPageMiddleware.Invoke(HttpContext)



View İşlemleri





Solution Explorer

Search Solution Explorer (Ctrl+F)

Örnek uygulama (1 of 1 project)

- Connected Services
- Dependencies
- Properties
- Controllers
- Views
- Home
- Product
- UrunleriGetir.cshtml
- Shared
- appsettings.json
- Program.cs
- Startup.cs

UrunleriGetir.cshtml

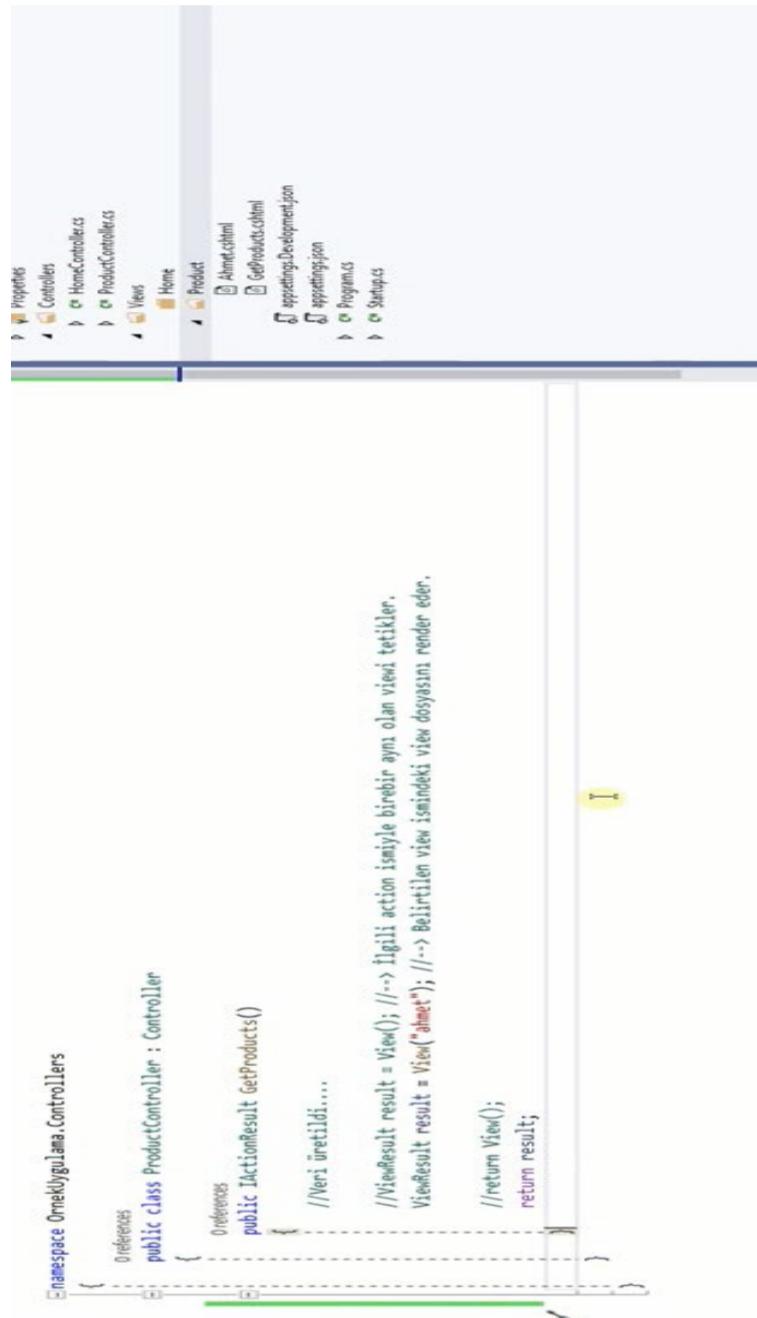
```
1 C:\Users\ranaz\source\repos\örnek uygulama\örnek uygulama\Views\Product\UrunleriGetir.cshtml
2
3 ViewData["Title"] = "ÜrünleriGetir";
4
5 <h1>ÜrünleriGetir</h1>
6
7
8
```

View dosyaları cshtml uzantılıdır.

Cs+html → **Razor**

Razor: MVC teknolojisine özel Html içerisinde C# kodlarını yazmamızı sağlayan bir teknolojidir.

View Yönetdirmesi:



Models

Solution Explorer

Search Solution Explorer (Ctrl + S)

Solution 'örnek uygulama' (1 of 1 project)

örnek uygulama

- Connected Services
- Dependencies
- Properties
- Controllers
- Models
- C# Product.cs
- Views
 - Home
 - Product
 - UrunleriGetir.cshtml
 - Shared
- appsettings.json
- C# Program.cs
- C# Startup.cs

Product.cs

örnek uygulama

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Threading.Tasks;
5
6 namespace örnek_uygulama.Models
7 {
8     public class Product
9     {
10         public int Id { get; set; }
11         public string productName { get; set; }
12         public int Quanty { get; set; }
13     }
14 }
```

Product.cs

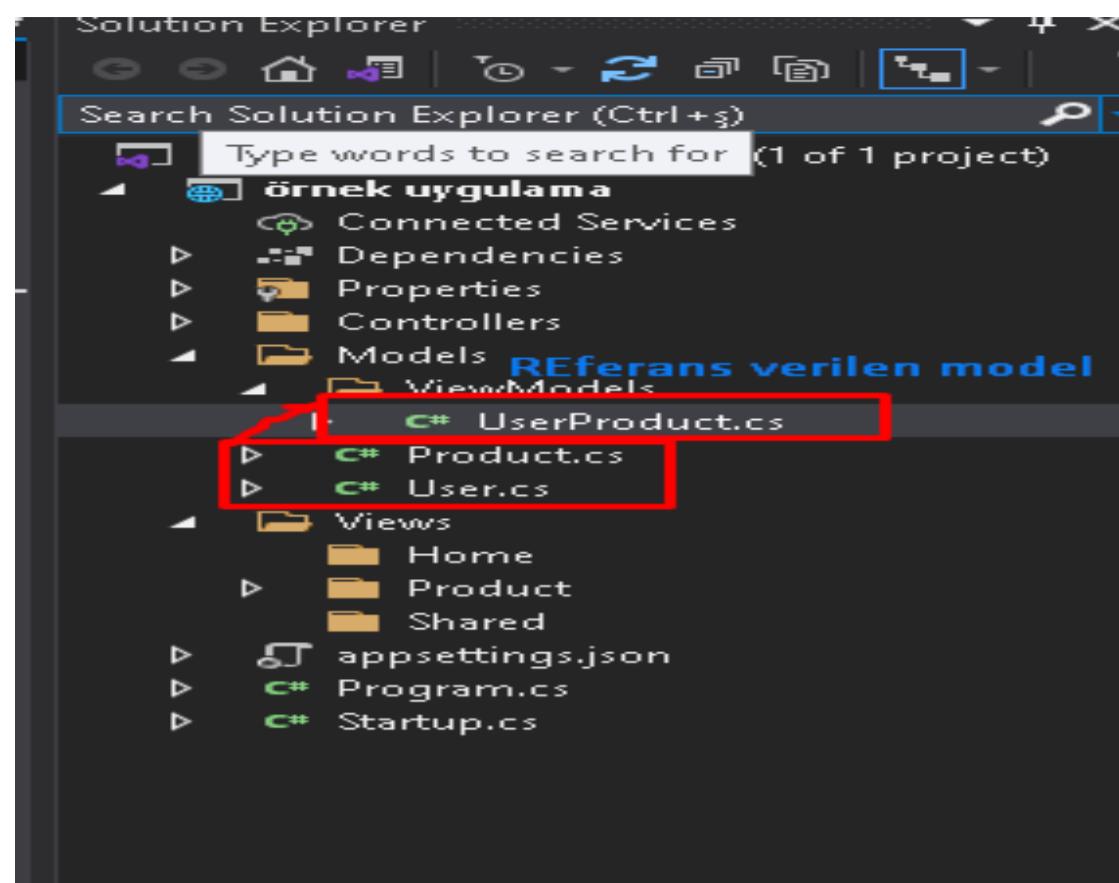
UrunleriGetir.cshtml

ProductController.cs*

HomeController.cs

örnek uygulama

```
1 using Microsoft.AspNetCore.Mvc;
2 using örnek_uygulama.Models;
3 using System;
4 using System.Collections.Generic;
5 using System.Linq;
6 using System.Threading.Tasks;
7
8 namespace örnek_uygulama.Controllers
9 {
10     public class ProductController : Controller
11     {
12         Product p = new Product();
13
14         public IActionResult UrunleriGetir()
15         {
16             //Veri üretilir
17
18             return View();
19         }
20     }
21 }
```



View model yapımı

```
UserProduct.cs  X  UrunleriGetir.cshtml      User.cs      Product.cs      ProductController.cs
önek uygulama
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5
6  namespace örnek_uygulama.Models.ViewModels
7  {
8      public class UserProduct
9      {
10         public Product Product { get; set; }
11         public User User { get; set; }
12     }
13 }
14
```

```

0 references
public IActionResult UrunleriGetir()
{
    Product product = new Product
    {
        Id = 1,
        productName = "Bilgisiyar",
        Quanty = 2500
    };

    User user = new User
    {
        Id = 1,
        nameSurname = "Ramazan Bilgiç",
        age = 24
    };

    UserProduct userProduct = new UserProduct
    {
        User = user,
        Product = product
    };
    return View(userProduct);
}

```

uct.cs UrunleriGetir.cshtml ✘ X User.cs Product.cs ProductController.cs HomeController.cs

```

1 @model örnek_uygulama.Models.ViewModels.UserProduct
2 @{
3     ViewData["Title"] = "UrunleriGetir";
4 }
5 ViewDataDictionary<örnek_uygulama.Models.ViewModels.UserProduct>
6 Microsoft.AspNetCore.Mvc.Razor.RazorPage<örnek_uygulama.Models.ViewModels.UserProduct>.ViewData { get; set; }
7 <h1>U Gets or sets the dictionary for view data.
8 <h3>@Model.Product.productName</h3>
9 <h3>@Model.User.nameSurname</h3>
0

```

← → ⌂ https://localhost:5001/Product/UrunleriGetir

Urunleri Listele

Bilgisiyar

Ramazan Bilgiç

Asp.NET Core 5.0 – View'e Tuple Nesne ve Kullanımı

Gönderimi

Tuple Nesne: İçerisinde birden fazla değeri, veriyi, nesneyi Referans edebilen, semantik açıda bize kazandırmış olduğu syntax/sözdizimine sahip olan bir nesnedir. Birden fazla nesnenin bir bütün olarak tasarılanması ve bu nesne üzerinden referans edilmesini sağlıyor

```
UrunleriGetir.cshtml      User.cs      Product.cs      ProductController.cs*  HomeC
ProductController.cs
Product p = new Product();

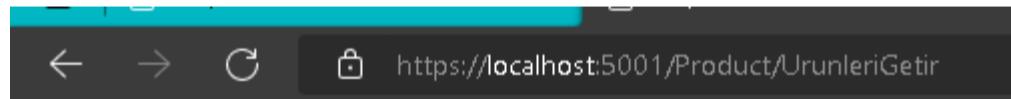
0 references
public IActionResult UrunleriGetir()
{
    Product product = new Product
    {
        Id = 1,
        productName = "Bilgisiyar",
        Quanty = 2500

    };
    User user = new User
    {
        Id = 1,
        nameSurname = "Ramazan Bilgiç",
        age = 24
    };

    var userProduct = (product, user);
    return View(userProduct);
}
```

Örnek:

```
UrunleriGetir.cshtml
1 @model örnekte_uygulama.Models.Product p, örnekte_uygulama.Models.User u
2 @{
3     ViewData["Title"] = "Urunleri Listele";
4 }
5
6 <h1>Urunleri Listele</h1>
7
8 <h3>@Model.u.nameSurname</h3>
9 <h3>@Model.p.productName</h3>
10
```



Urunleri Listele

Ramazan Bilgiç

Bilgisiyar

Razor Nedir?

Asp. Net Mvc ,Asp.Net Core Mvc mimarisinde cshtml dosyasında C#

Kodları yazmamızı sağlayan ve bu kodlarla serverde çalışmamızı sağlayan teknolojidir.

Microsoft tarafından geliştirilmiştir. Html ile beraber kod yazmamızı sağlayan teknolojidir.

@Operatörü: Razor teknolojisini kullanmamızı sağlayan operatördür.

```
etir.cshtml ➔ X UrunleriGetir.cshtml.g.cs ProductController.cs UserProduct.cs
1  @model örnek_uygulama.Models.Product p, örnek_uygulama.M
2  @{
3      ViewData["Title"] = "UrunleriGetir";
4  }
5
6  <h1>Urunleri Listele</h1>
7
8  <h3>@Model.u.nameSurname</h3>
9  <h3>@Model.p.productName</h3>
10 @if (true)
11 {
12     <h1>Burda Razor teknolojisini kullandık</h1>
13 }
14
```

← → ⌂ https://localhost:5001/Product/UrunleriGetir

Urunleri Listele

Ramazan Bilgiç

Bilgisiyar

Burda Razor teknolojisini kullandık

```
<h1>Urunleri Listele</h1>

<h3>@Model.u.nameSurname</h3>
<h3>@Model.p.productName</h3>
@if (true)
{
    <h1>Burda Razor teknolojisini kullandık</h1>
}

@*"Razor Teknolojisi ile Değişken Tanımlama*@
@{
    string metin = "Razor Teknolojisi ile Değişken Tanımlama ve okuma";
    string metin2 = "Razor Teknolojisi ile Parçalı scop yapısını kullanarak başka scope lardan member'a ulaşmak";
}

@*"Razor Teknolojisi ile Değişken Tanımlama ve okuma*@
@{
    int a = 24;
    Console.WriteLine(a + metin);
}

@*"Razor Teknolojisi ile Parçalı scop yapısını kullanarak başka scope larda member'a ulaşmak*@
@{
    Console.WriteLine(metin2);
}

@*"Razor İçerisinde Html Kullanımı*@
@{
    <div><h2>Razor İçerisinde Html Kullanımı</h2></div>
}
```

```
/*Razor İçerisinde <text> etiketi Kullanımı*/
@if (a == 24)
{
    <text> Razor İçerisinde text etiketi Kullanımı</text>
}
else
{
    <text>Hayır</text>
}

/*Razor ile Tek satırlık işlemler*/
<h3>@(35%5==0? "Razor ile Tek satırlık işlemler" : "Razor ile Tek satırlık işlemler")</h3>
<h3>@(5+4)</h3>

/*Razor ile Ternary Operatörü */
<h3>@(35%5==0? "Razor ile Ternary Operatörü" : "Razor ile Ternary Operatörü")</h3>

/*Razor ile Koşul Yapıları*/
@if (sayi == 5)
{
    <h3>Razor ile Koşul Yapıları</h3>
}

/*Razor ile Döngüler*/
@for (int i = 1; i <= 5; i++)
{
    <h1> Razor ile Döngüler @i</h1>
}
```

Urunleri Listele

Ramazan Bilgiç

Bilgisiyar

Burda Razor teknolojisini kullandık

Razor İçerisinde Html Kullanımı

Razor İçerisinde text etiketi Kullanımı

Razor ile Tek satırlık işlemler

9

Razor ile Ternary Operatörü

Razor ile Koşul Yapıları

Razor ile Döngüler 1

Razor ile Döngüler 2

Razor ile Döngüler 3

Razor ile Döngüler 4

Razor ile Döngüler 5

Helpers

UrlHelper-HtmlHelper-TagHelper

Yardımcı metotlardır

UrlHelpers : Asp.Net Core Mvc uygulamalarında url oluşturmak için yardımcı metotlar içeren ve o anki url'ye dair bizlere bilgi veren bir sınıfır

METOTLAR

Action
ActionLink
Content
RouteUrl

Propertyler

ActionContext

Action Metodu

- Verilen Controller ve Action'a ait url oluşturmayı sağlayan metottur.

```
Url.Action("index", "product", new { id = 5 })
```

```
/product/index/5
```

ActionLink Metodu

- Verilen Controller ve Action'a ait url oluşturmayı sağlayan metottur.

```
Url.ActionLink("index", "product", new { id = 5 })
```

```
https://localhost:5001/product/index/5
```

Content Metodu

- Genellikle CSS ve Script gibi dosya dizinlerini programatik olarak tarif etmek için kullanmaktadır.

```
Url.Content("~/site.css")
```

```
/site.css
```

- UseStaticFiles middleware'ı ile gelen static dosya yapılanması bu fonksiyonun işlevsellliğini daha efektif üstlenmektedir

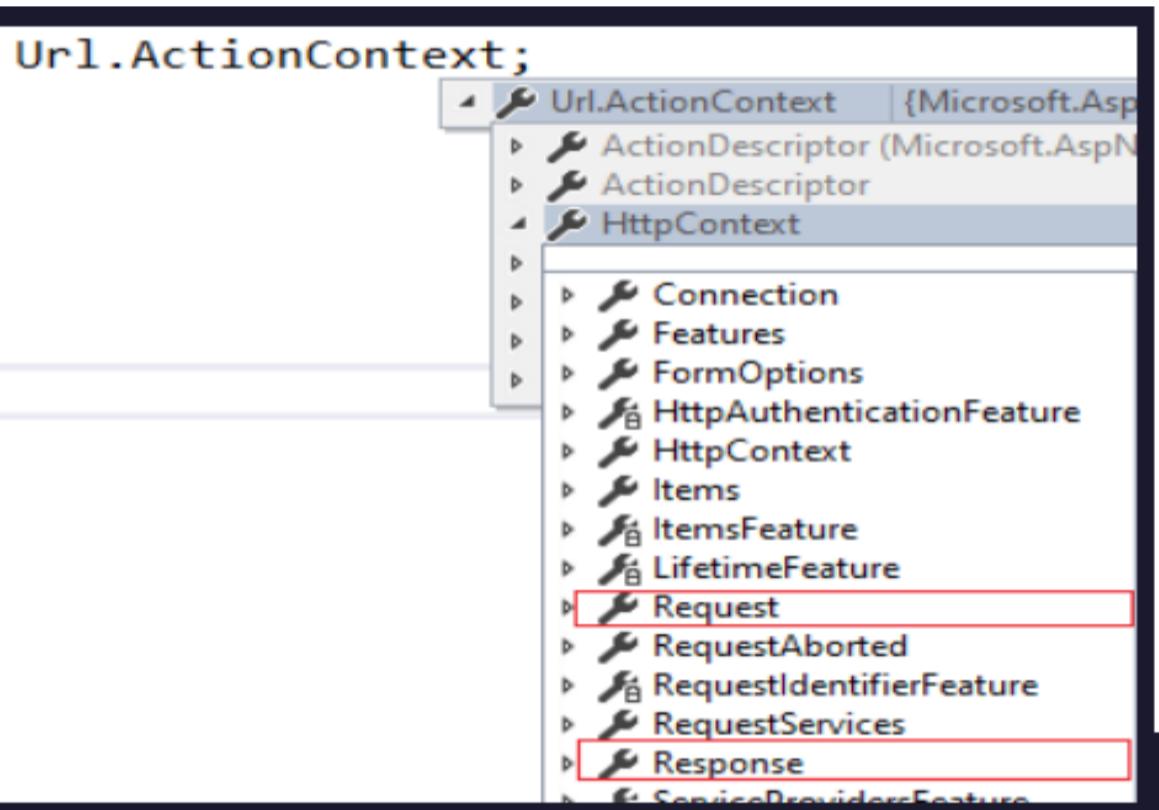
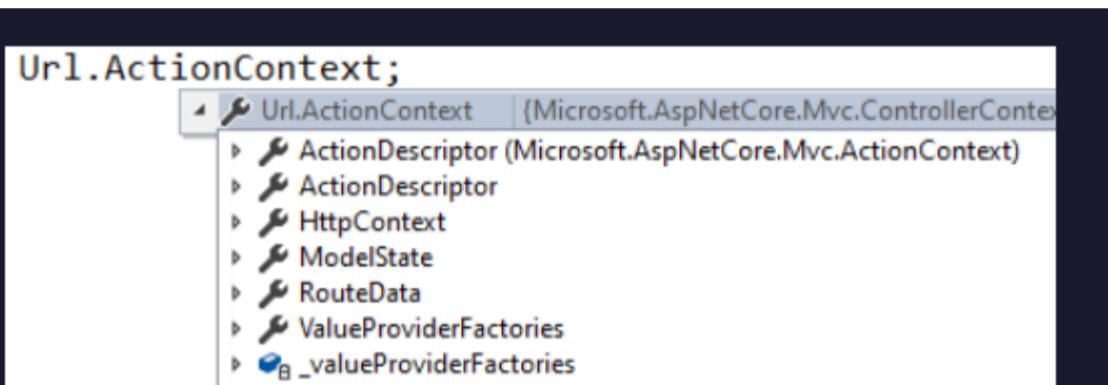
RouteUrl Metodu

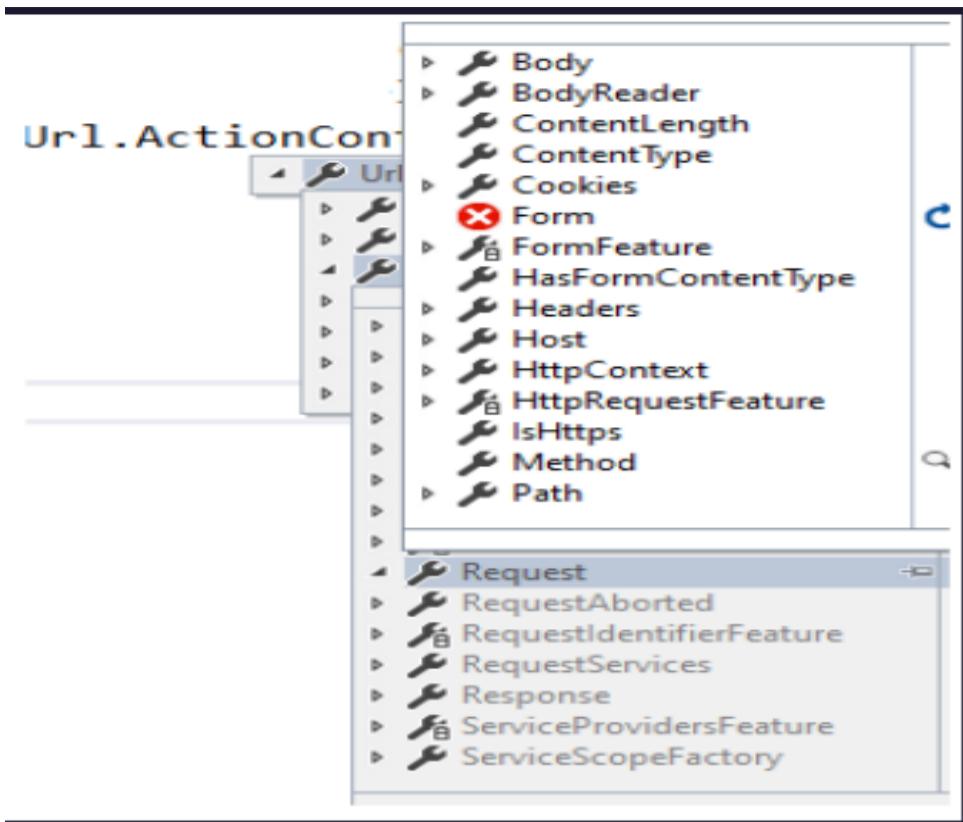
- Mimaride tanımlı olan Route isimlerine uygun bir şekilde url oluşturan bir metottur.

```
Url.RouteUrl("Default")
```

```
/Product/GetProducts
```

ActionContext Property'si O anki url'de tüm bilgilere erişebilmemizi sağlayan property'dir.





HtmlHelpers:

- Html etiketlerini server tabanlı oluşturmamızı sağlayan sözde :) yardımcı metotları barındırmakta,
- Hedeflenen .cshtml dosyalarını render etmemizi sağlamakta,
- O anki context'e dair bilgiler edinmemizi sağlamakta,
- Veri taşıma kontrollerine erişmemizi sağlamaktadır.

Metotlar

Propertyler

Html.Partial	ViewContext
Html.RenderPartial	TempData
Html.ActionLink	ViewData
Html Form Metotları	ViewBag

Html.Partial Metodu

- Hedef View'i render etmemizi sağlayan bir fonksiyondur

Render edilen view'e ilgili action'dan model/data gönderilebilmektedir

```
<div style="border-top-color:ActiveBorder">
    @Html.Partial("~/Views/Product/Index.cshtml")
</div>
```

```
1 @model (örnek_uygulama.Models.Product p, örnek_uygulama.Models.User u)
2 @{
3     ViewData["Title"] = "UrunleriGetir";
4 }
5 ViewDataDictionary<(örnek_uygulama.Models.Product p, örnek_uygulama.Models.User u)> Microsoft.AspNetCore.Mvc.Razor.RazorPage<
6     (örnek_uygulama.Models.Product p, örnek_uygulama.Models.User u)>.ViewData { get; set; }
7 Gets or sets the dictionary for view data.
8 <h1>Urunleri Listele</h1>
9
10 <h3>@Model.u.nameSurname</h3>
11 <h3>@Model.p.productName</h3>
12 @if (true)
13 {
14     <h1>Burda Razor teknolojisini kullandık</h1>
15 }
16 @*Razor Teknolojisi ile Değişken Tanımlama*@
17 @{
18     string metin = "Razor Teknolojisi ile Değişken Tanımlama ve okuma";
19     string metin2 = "Razor Teknolojisi ile Parçalı scop yapısını kullanarak başka scope lardan member'a ulaşmak";
20 }
21 @*Razor Teknolojisi ile Değişken Tanımlama ve okuma*@
22 @{
23     int a = 24;
24     Console.WriteLine(a + metin);
25 }
26 @*Razor Teknolojisi ile Parçalı scop yapısını kullanarak başka scope larda member'a ulaşmak*@
27 @{
28     Console.WriteLine(metin2);
29 }
30
31 @*Razor İçerisinde Html Kullanımı*@
32 @{

<-- → ⌂ https://localhost:5001
```

PARTİAL VIEW RENDER ETTİK

Urunleri Listele

Ramazan Bilgiç

Bilgisiyar

Burda Razor teknolojisini kullandık

Razor İçerisinde Html Kullanımı

Razor İçerisinde text etiketi Kullanımı

Razor ile Tek satırlık işlemler

9

Razor ile Ternary Operatörü

Razor ile Koşul Yapıları

Razor ile Döngüler 1

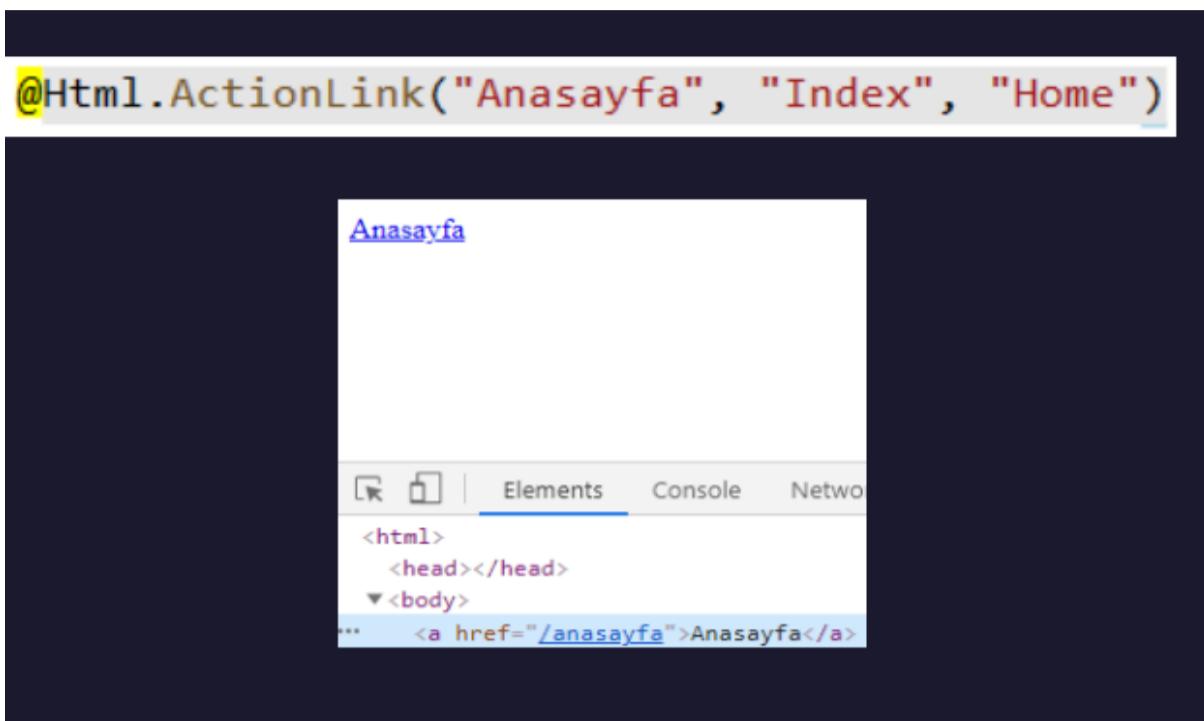
Html.RenderPartial Metodu

- Hedef View'i render etmemizi sağlayan bir fonksiyondur.

```
<div style="border-top-color:ActiveBorder">
    @Html.RenderPartial("~/Views/Product/Index.cshtml");
</div>
```

`Html.RenderPartial` sayfanın `TextWriter`'ını kullandığı için(yani `Http response stream`'e yazıldığı için) `Html.Partial`'a nazaran daha hızlı render işlemini yürütür. Dolayısıyla daha performanslıdır.

Html.ActionLink Metodu • Url oluşturur.



Html Form Metotları

- Kullanıcıyla etkileşime girmemizi sağlayan form ve input nesneleri oluşturmamızı sağlayan metotlardır.
- `Html.BeginForm`
- `Html.CheckBox`
- `Html.TextBox`
- `Html.Display`
- `Html.Password`
- `Html.TextArea`
- `Html.ValidationMessage`

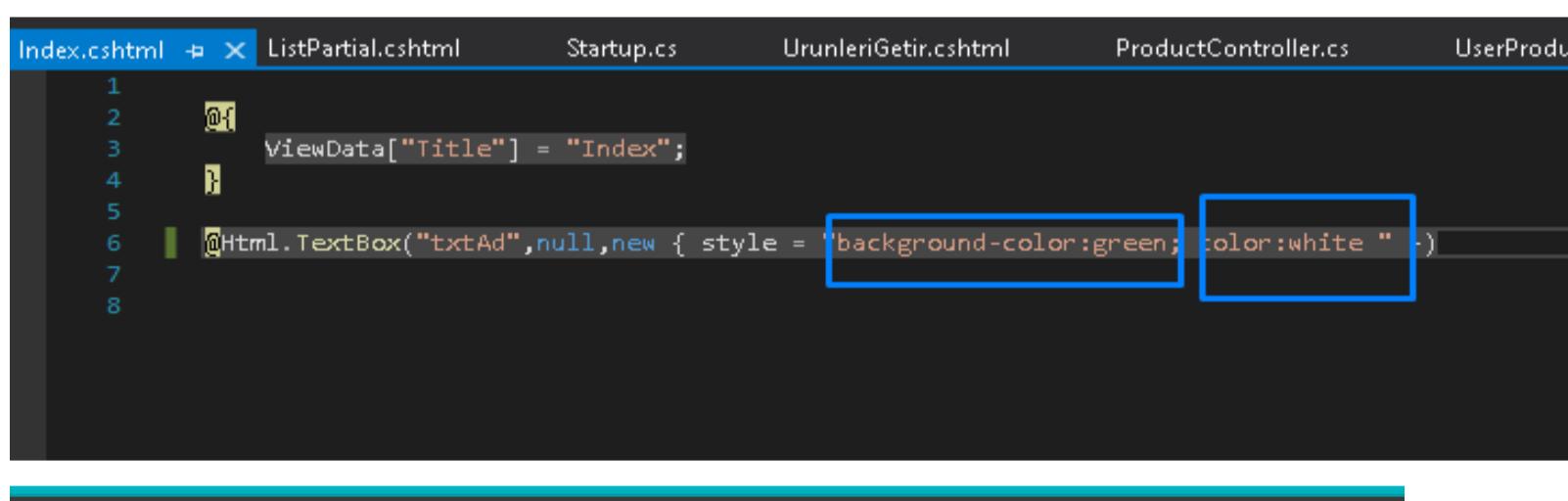
```
@Html.BeginForm()
@Html.CheckBox("cb")
@Html.TextBox("txt")
@Html.Display("display")
@Html.Password("pwd")
@Html.TextArea("area")
@Html.ValidationMessage("vldt")
```

```
<form action="/product/getproducts" method="post">
    <input id="cb" name="cb" type="checkbox" value="true">
    <input id="txt" name="txt" type="text" value>
    <input id="pwd" name="pwd" type="password">
    <textarea id="area" name="area"></textarea>
    <span class="field-validation-valid" data-valmsg-for="vldt" data-valmsg-replace="true"></span>
</form>
```

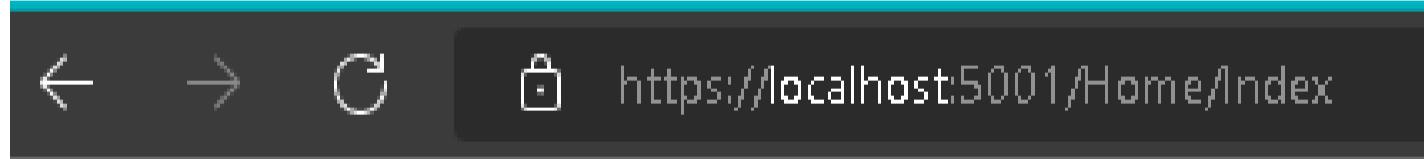
HtmlHelper ile form ve input nesnesinin oluşturulması server tarafından üstlenildiği için ekstradan maliyetli yapılmalıdır. Bu maliyeti ortadan kaldırmak için Asp.NET Core MVC'de TagHelpers yapıları gelmiştir.

Custom HtmlHelper Fonksiyonu Oluşturmak

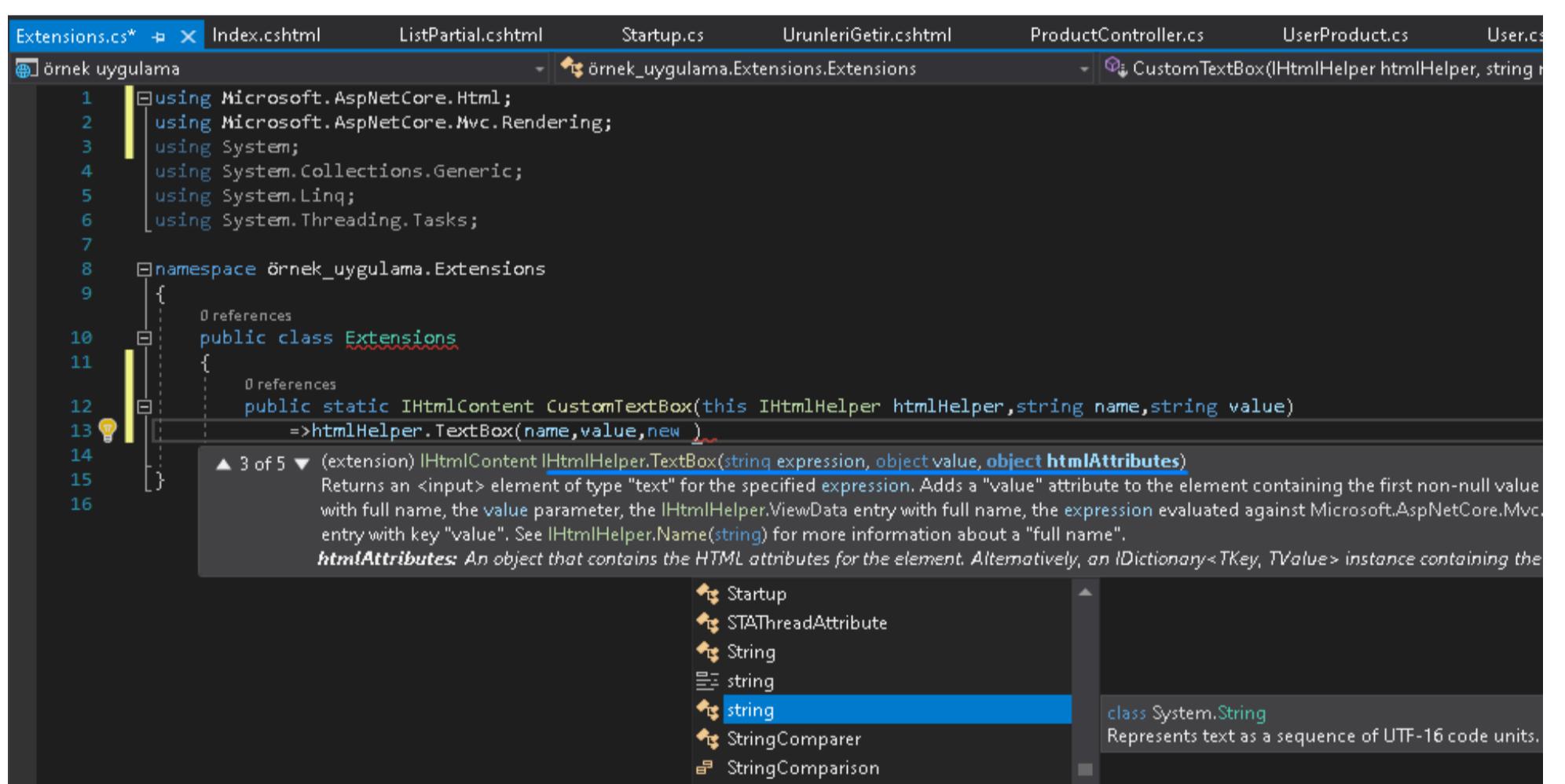
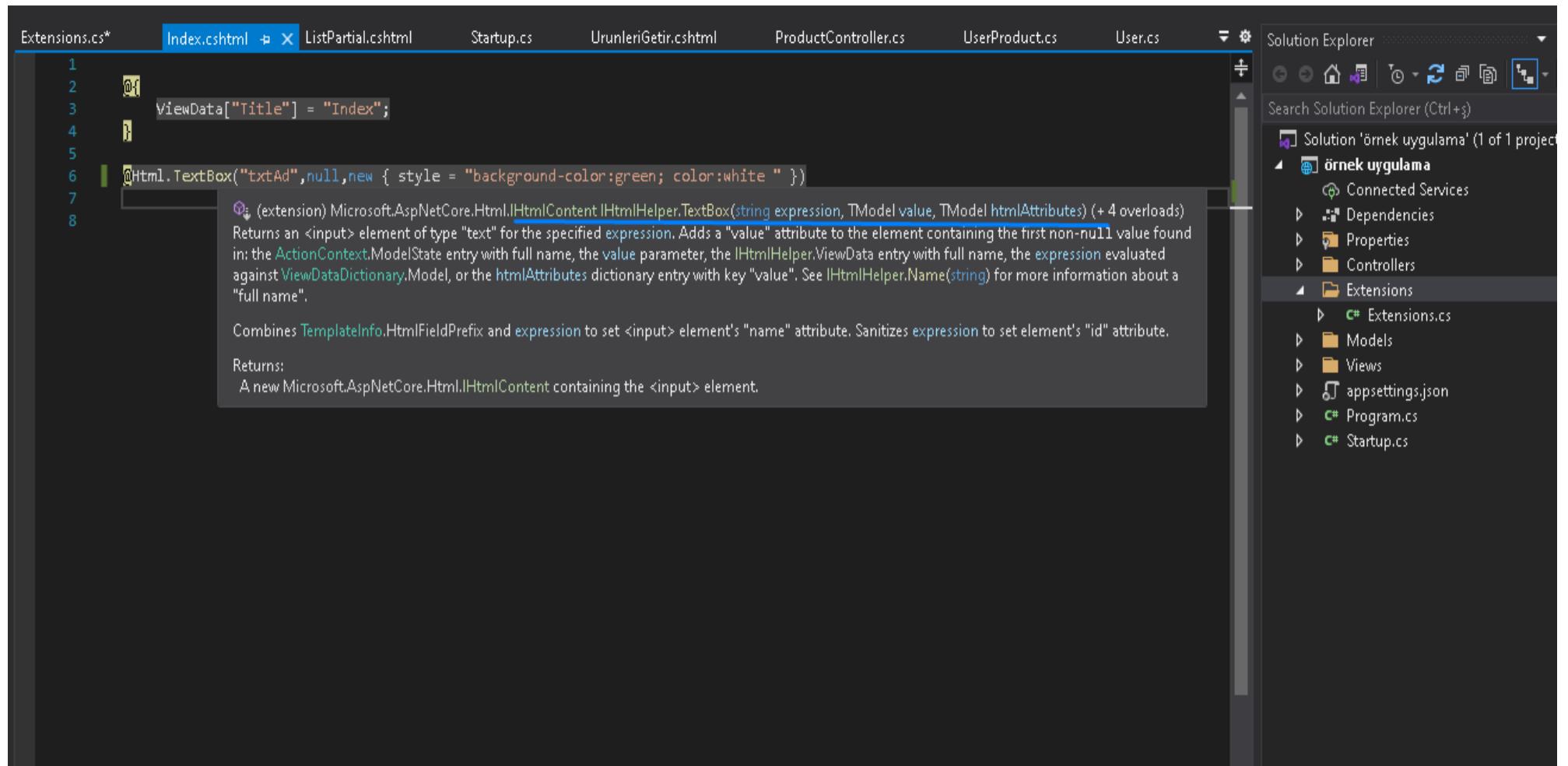
Extensions fonksiyonları ile bu yapı oluşturulur.

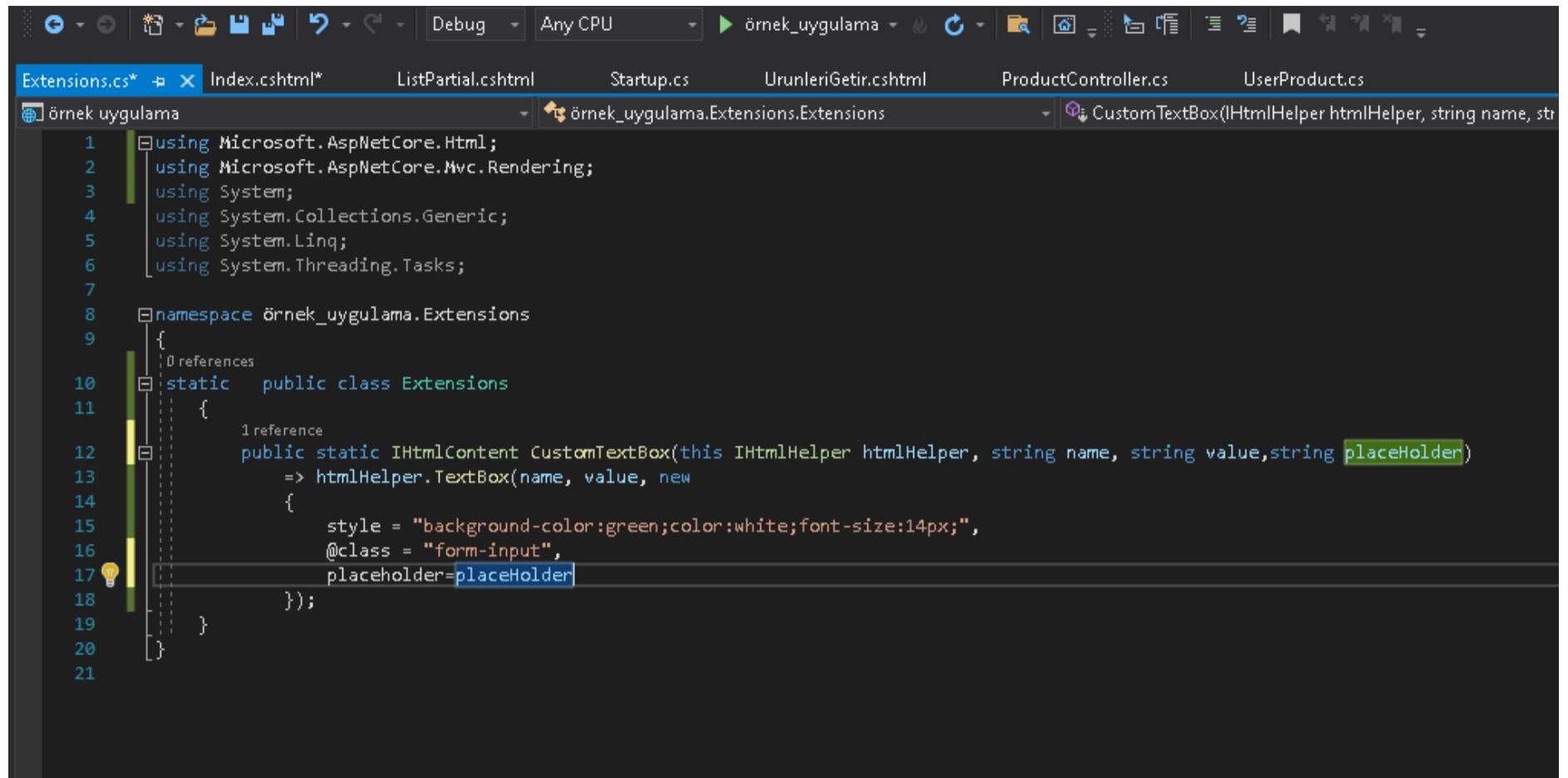


```
Index.cshtml  X ListPartial.cshtml      Startup.cs      UrunleriGetir.cshtml      ProductController.cs      UserProdu
1
2     @{
3         ViewData["Title"] = "Index";
4     }
5
6     @Html.TextBox("txtAd", null, new { style = "background-color:green; color:white" })
7
8
```

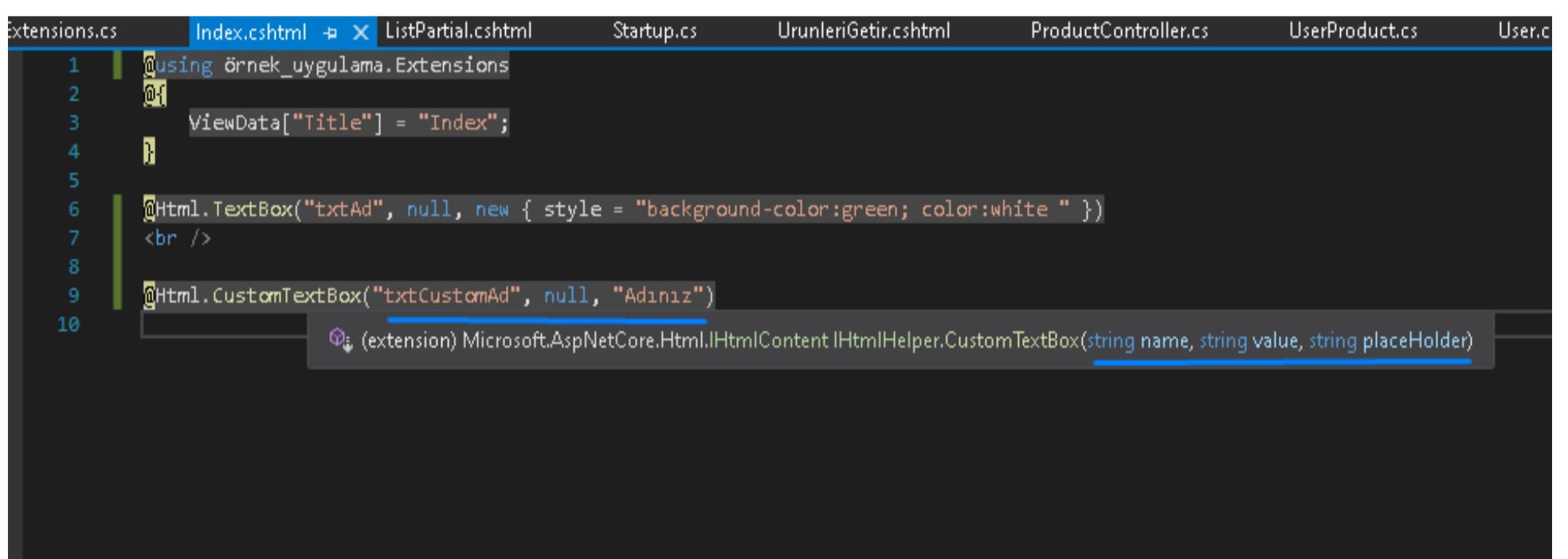


sdasdsad



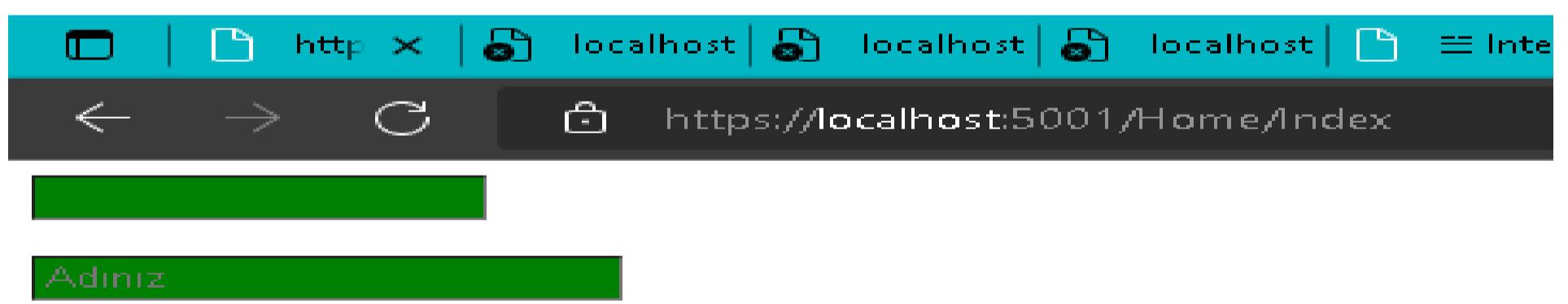


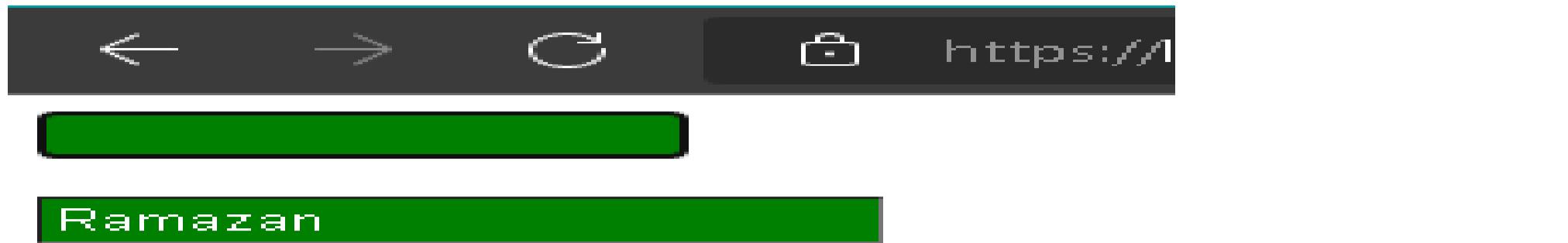
```
Extensions.cs*  Index.cshtml*  ListPartial.cshtml  Startup.cs  UrunleriGetir.cshtml  ProductController.cs  UserProduct.cs
@örnek uygulama
@örnek_uygulama.Extensions.Extensions
1  using Microsoft.AspNetCore.Html;
2  using Microsoft.AspNetCore.Mvc.Rendering;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Threading.Tasks;
7
8  namespace örnek_uygulama.Extensions
9  {
10     static public class Extensions
11     {
12         1 reference
13         public static IHtmlContent CustomTextBox(this IHtmlHelper htmlHelper, string name, string value, string placeHolder)
14             => htmlHelper.TextBox(name, value, new
15             {
16                 style = "background-color:green;color:white;font-size:14px;",
17                 @class = "form-input",
18                 placeholder=placeHolder
19             });
20     }
21 }
```



```
Extensions.cs  Index.cshtml*  ListPartial.cshtml  Startup.cs  UrunleriGetir.cshtml  ProductController.cs  UserProduct.cs  User.c
1  @using örnek_uygulama.Extensions
2  @{
3      ViewData["Title"] = "Index";
4  }
5
6  @Html.TextBox("txtAd", null, new { style = "background-color:green; color:white" })
7  <br />
8
9  @Html.CustomTextBox("txtCustomAd", null, "Adınız")
10
```

(extension) Microsoft.AspNetCore.Html.IHtmlContent IHtmlHelper.CustomTextBox(string name, string value, string placeHolder)





TAGHELPERS

TagHelper Nedir? Tag Helpers, daha okunabilir, anlaşılabilir ve kolay geliştirilebilir bir view inşa etmemizi sağlayan, Asp.NET Core ile birlikte HtmlHelpers'ların yerine gelen yapılardır.

TagHelpers vs HtmlHelpers



TagHelper'lar view'lerde ki kod maliyetini oldukça düşürmektedirler.



HtmlHelpers'ların html nesnelerinin generate edilmesini server'a yüklemesinin getirdiği maliyetide ortadan kaldırılmaktadır.



HtmlHelpers'lar da ki programatik yapılanma, programlama bilmeyen tasarımcıların çalışmasını imkansız hale getirmektedir. TagHelpers'lar ile buradaki kusur giderildi ve tasarımcılar açısından programlama bilgisine ihtiyaç duyulmaksızın çalışma yapılabılır nitelik kazandırdı.



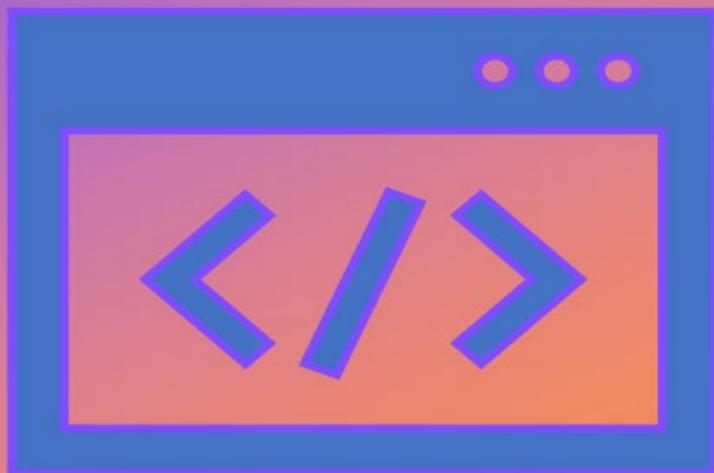
HtmlHelpers ile oluşturulan html nesnesinin attribute'ları 'htmlAttribute' parametresi üzerinden anonim nesne ile verilmektedir. Bu durum hem bellek optimizasyonu açısından hemde kod maliyeti açısından oldukça zararlıdır. TagHelpers'lar bu maliyeti ortadan kaldırırmakta ve html nesnelerine sadece ilgili attribute'ları normal sözdizimiyle vermekle ilgilenmektedirler.

TAGHELPERS ENTEGRASYONU : @addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers

Viewlere bu kütüphaneyi entegre etmemiz gerekmektedir .

```
cs Index.cshtml* ListPartial.cshtml Startup.cs UrunleriGetir.cshtml
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

HTML NESNELERİNDE TAGHELPERS

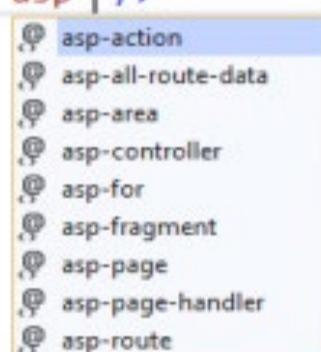


FORM TAGHELPER :Form oluşturmak için kullandığımız format.

```
<form action="/" asp-action="Index" asp-controller="Home" method="post"></form>
```

INPUT TAGHELPER: Input nesnelerinde kullanabildiğimiz belirli model binding işlemlerinde kullanırız

```
<input type="text" style="color:green; background-color:black;" asp- />
```



CACHE TAGHELPER: o anın tarihini cache'liyor. Üstteki uygulamayı ayağa kaldırıldığımızda zaman hafızada kalıyor ve sayfa yenilendiğinde değişmiyor aşağıdakinde ise her yenilendiğinde zamanı yeniliyor.

```
Cache:2.06.2022 15:24:34  
2.06.2022 15:25:00
```

o

```
<cache>  
| Cache : @DateTime.Now  
</cache>  
<br />  
@DateTime.Now
```

ENVIRONMENT TAGHELPER

```
<environment names="Development">
|   <p>Development ortamı</p>
</environment>
<environment names="Production, Staging">
|   <p>Production veya Staging ortamı</p>
</environment>
```

IMAGE TAGHELPER

- Tarayıcılar static dosyaları local cache üzerinde saklamaktadır.
- Cachelenmiş bir dosya tekrar istenildiği taktirde bunun için server'a istek gönderilmez ve local cache üzerinden ilgili dosyanın cache'İ gönderilir. Böylece sayfalar ilk açılışlarından sonraki taleplerde daha hızlı yüklenebilmektedirler.
- Lakin bazen dosya adı değişmeden içeriği değişimleştirmektedir. Böyle bir durumda ilgili dosyanın cache'den değil, server'dan yüklenmesi gerekmektedir. Bu duruma biz ETag yöntemiyle müdahale edebilmekteyiz.
- Asp.NET Core MVC mimarisinde TagHelper'lar içerisinde static dosyalara etag yöntemini uygulayabilir ve dosyanın adı değişmesede içeriği değiştiği taktirde etag üzerinden bu değişikliği fark ederek ilgili dosyanın server'dan talep edileceği bilinebilmektedir.

```

```

PARTIAL TAGHELPER

```
<partial name="~/Views/Product/Partials>ListPartial.cshtml" />
```

REMOVE TAGHELPER SAYFA SEVIYESİNDE

```
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

```
@removeTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

```

```

REMOVE ! TAGHELPER TAG SEVIYESİNDE

```
<form asp-action="Index" asp-controller="Home"></form>
<!form asp-action="Index" asp-controller="Home"><!form>
```

Custom TagHelpers Oluşturma

Tekrar tekrar bazı yapıları oluşturmaktansa onu bir kere tanımlayıp lazımlı olduğunda çağrılması durumunda lazımdır.

EmailTagHelper.cs

```

using Microsoft.AspNetCore.Razor.TagHelpers;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace önek_ugulama.TagHelpers
{
    public class EmailTagHelper : TagHelper //Bütün isimleri viewde küçük harfle alır
    {
        public string Mail { get; set; }
        public string Display { get; set; }

        public override void Process(TagHelperContext context, TagHelperOutput output) //Process : ilgili taghelperin işleminin yapıldığı
        {
            output.TagName = "a";
            output.Attributes.Add("href", $"mailto:{Mail}");
            output.Content.Append(Display);
        }
    }
}

```

Deneme.cshtml

```

@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
@addTagHelper *, önek_ugulama.TagHelpers

@if (true)
{
    ViewData["Title"] = "Deneme";
}

<a href="mailto:ramazanbilgic0947@gmail.com">E-mail</a>@*Mail linki*@  

<br />  

<br />

<email a="abc" mail="ramazanbilgic0947@gmail.com" display="Benim Mailim"></email>
<br />
<mail-gonder-form></mail-gonder-form>

```

Solution Explorer

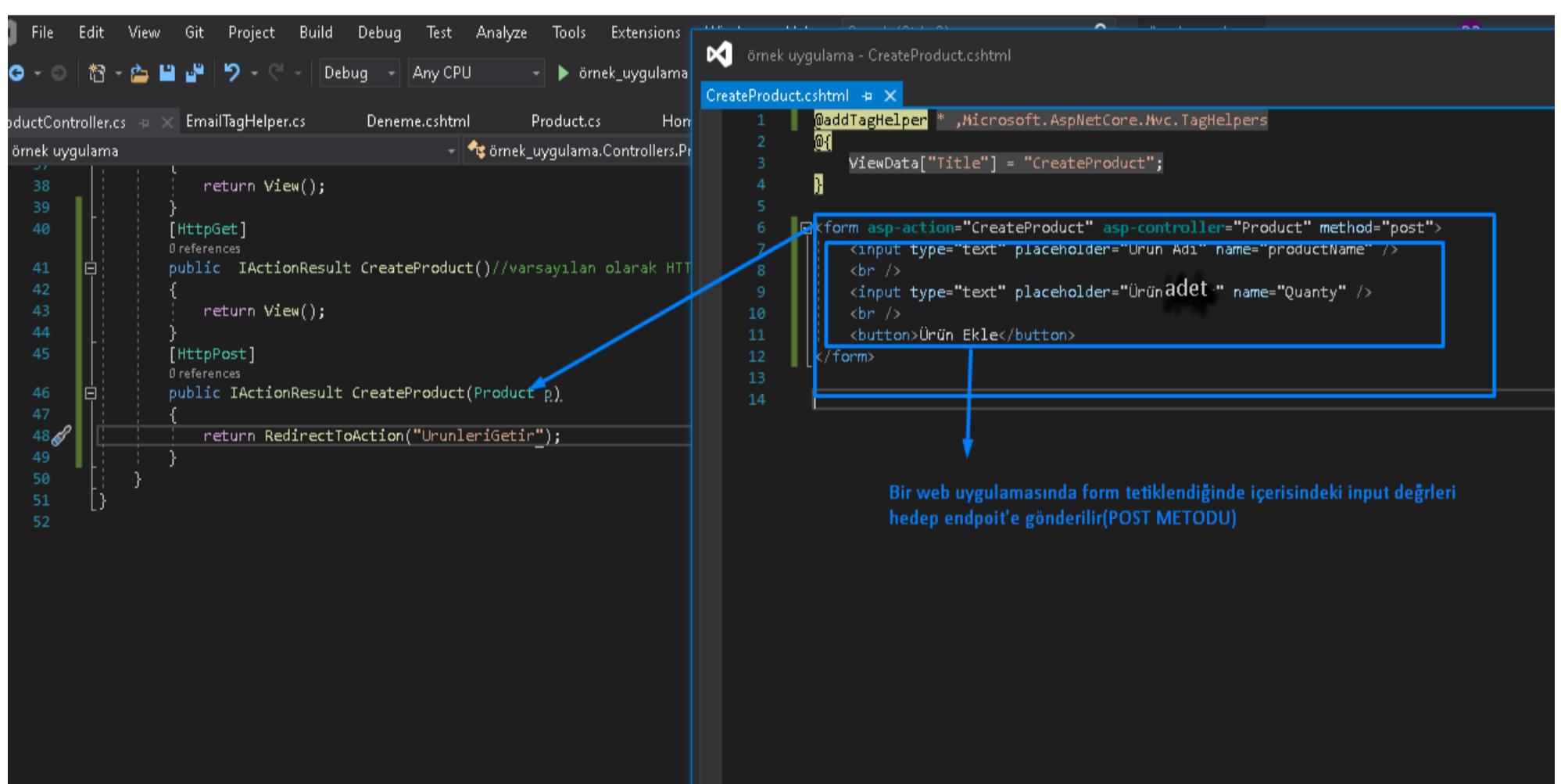
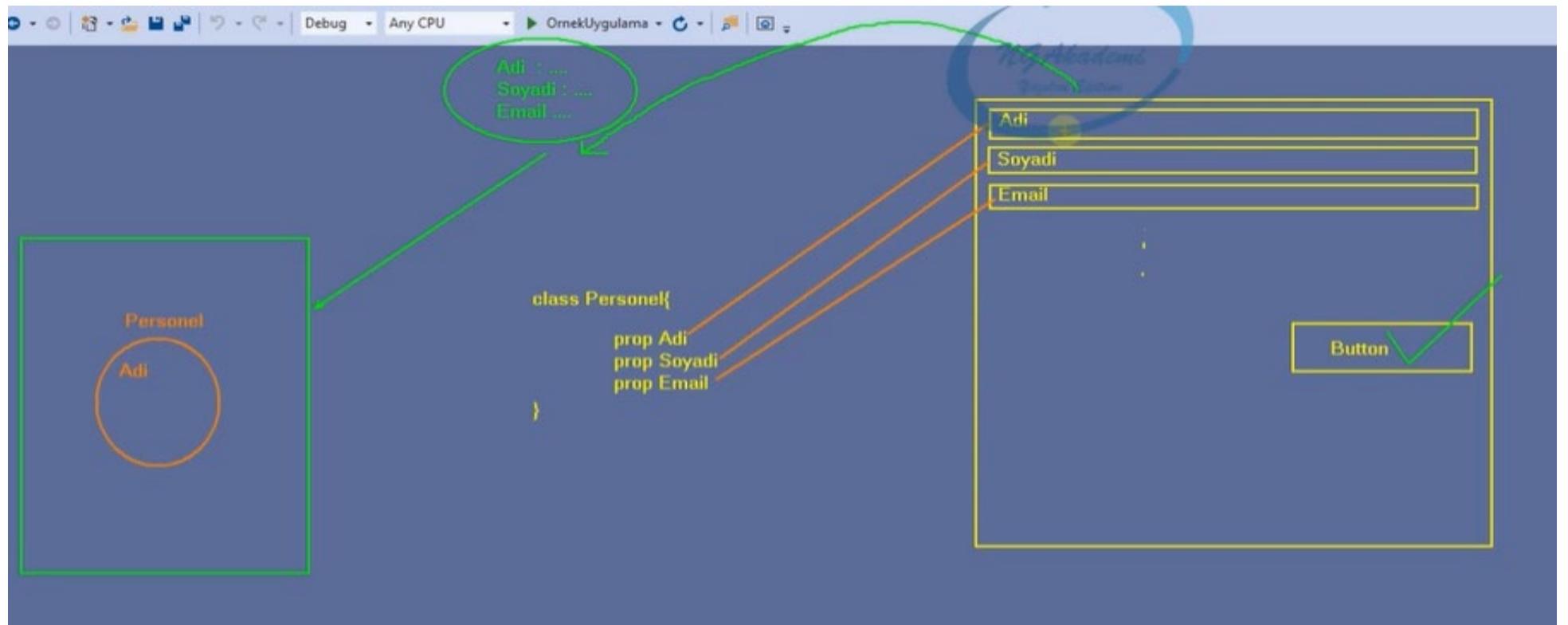
- önek_ugulama
 - Connected Services
 - Dependencies
 - Properties
 - Controllers
 - Extensions
 - C# Extensions.cs
 - Models
 - TagHelpers
 - C# EmailTagHelper.cs
 - Views
 - appsettings.json
 - Program.cs
 - Startup.cs

Annotations:

- Bir sınıfın TagHelper olabilmesi için TagHelper sınıfından türemesi gerekmektedir.
- TagHelper'ın işlevsellik gösterebilmesi için Process metodunun override edilmesi gerekmektedir.
- İlgili Attribute'in bizlere gizlsini rmektedir.
- İlgili TagHelper'ı getirmektedir.
- Attributes, uniqueid

Model Binding Mekanizması

HttpRequest ile gelen verileri ayırtılarak ilgili Controller'da bulunan action metodlarında uygun herhangi türde dönüştürülmüşdür. Yani Kullanıcının form üzerinden girmiş olduğu dataları/verileri biz controllerda kendimize ait türlerde yakalamak/yönetmek istediğimizde kullandığımız mekanizmadır.



```
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
@model örnек_uygulama.Models.Product
ViewData["Title"] = "CreateProduct";

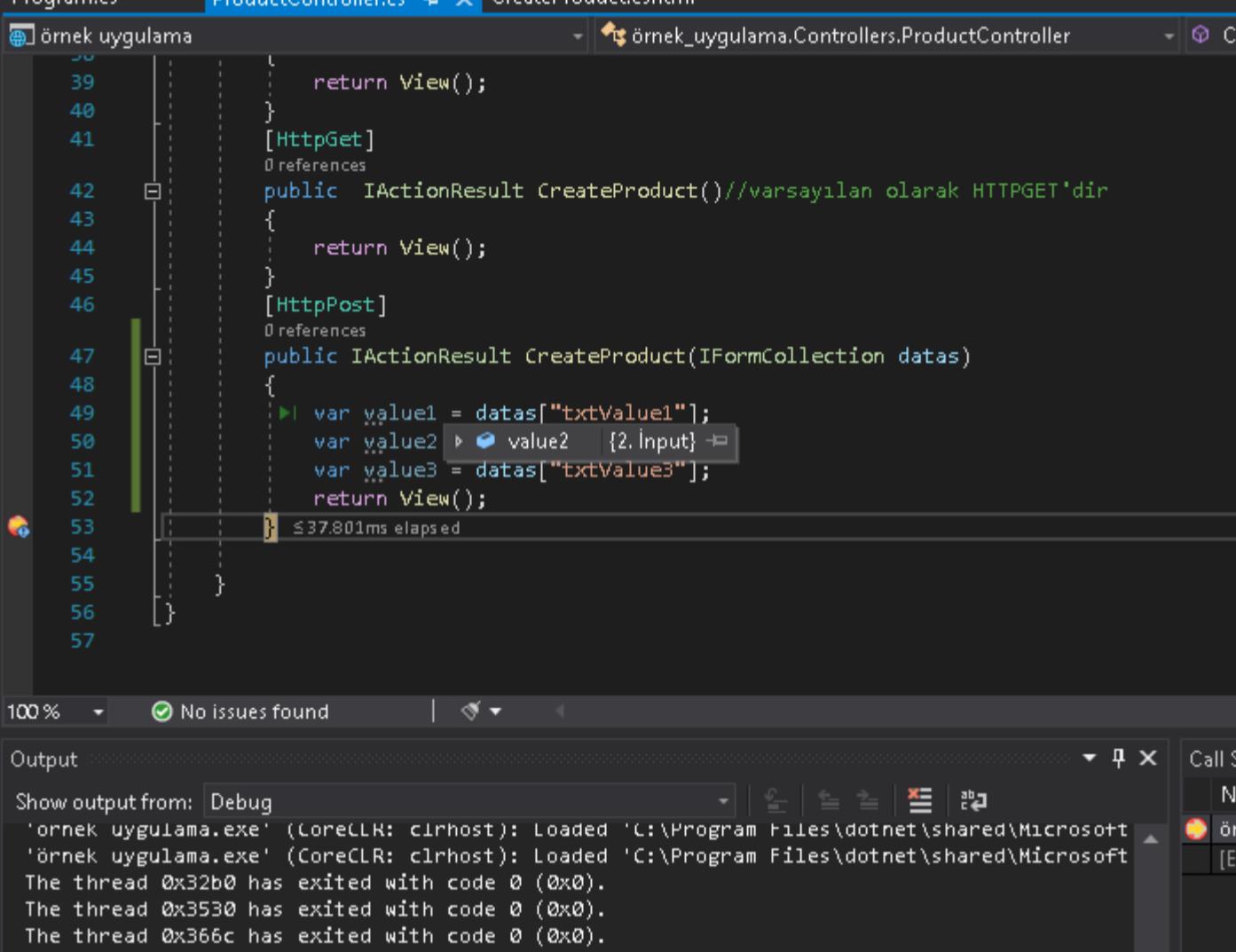
<form asp-action="CreateProduct" asp-controller="Product" method="post">
    <input type="text" placeholder="Ürün Adı" asp-for="productName" />
    <br />
    <input type="text" placeholder="Ürün Adet" asp-for="Quanty" />
    <br />
    <button>Ürün Ekle</button>
</form>
```

KULLANICIDAN VERİ ALMA YÖNTEMLERİ

Form Üzerinden Veri Alma

Kullanıcının kendi isteği ile form üzerinde verdiği bilgilerdir. İlerde başka yöntemlerin olduğunu göreceğiz.

IFormCollection İle Veri Alma

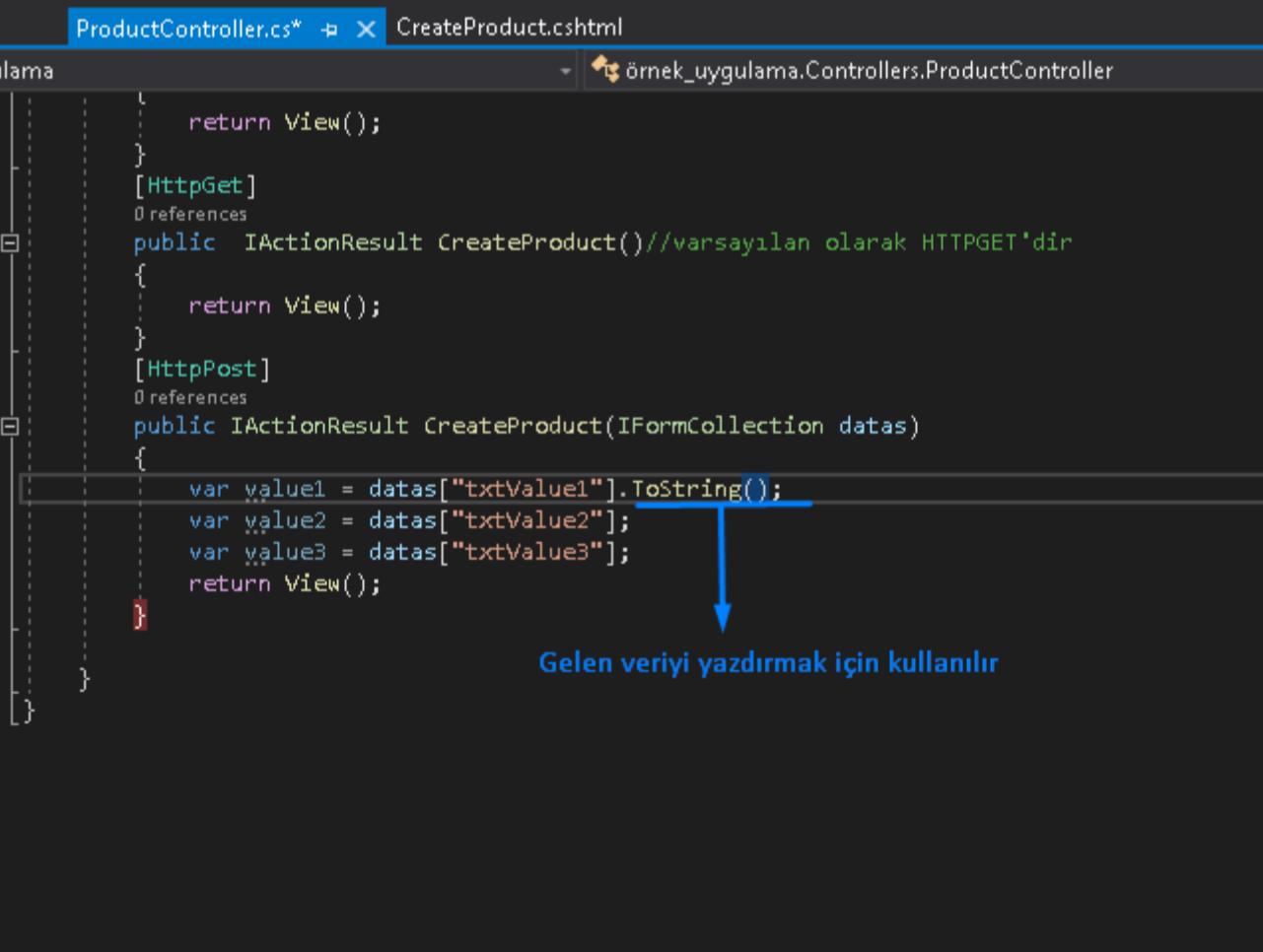


The screenshot shows the ProductController.cs file in Visual Studio. The code defines two actions: CreateProduct (HTTP GET) and CreateProduct (HTTP POST). The HTTP POST action takes an IFormCollection parameter named 'datas'. Inside the method, three variables are assigned values from the collection: value1, value2, and value3. A tooltip indicates that value2 is a 2. input type. The output window shows the application has exited successfully.

```
Program.cs*  ProductController.cs*  CreateProduct.cshtml
Örnek uygulama
38     return View();
39 }
40 }
41 [HttpGet]
42 public IActionResult CreateProduct() //varsayılan olarak HTTPGET'dir
43 {
44     return View();
45 }
46 [HttpPost]
47 public IActionResult CreateProduct(IFormCollection datas)
48 {
49     var value1 = datas["txtValue1"];
50     var value2 = datas["txtValue2"]; // 2. input
51     var value3 = datas["txtValue3"];
52     return View();
53 }
54
55 }
56
57 }

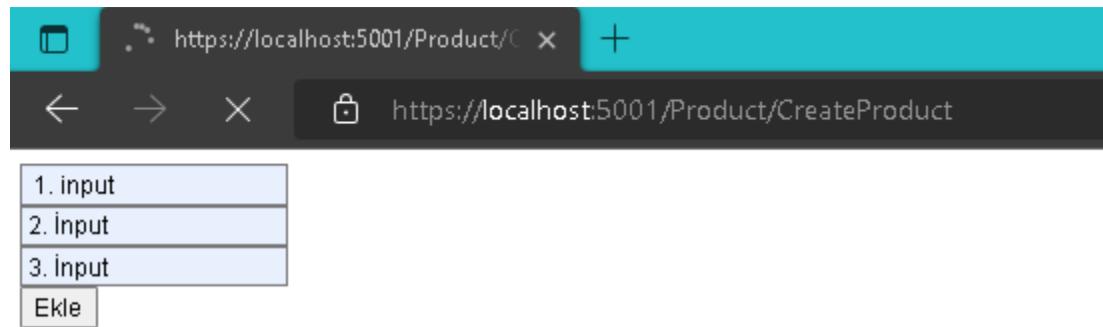
≤37.801ms elapsed

100%  No issues found  Output  Show output from: Debug
'ornek_ugulama.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft
'ornek_ugulama.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft
The thread 0x32b0 has exited with code 0 (0x0).
The thread 0x3530 has exited with code 0 (0x0).
The thread 0x366c has exited with code 0 (0x0).
The thread 0x366c has exited with code 0 (0x0).
```



The screenshot shows the same ProductController.cs file. The code is identical to the previous one, but it includes a callout arrow pointing to the line where 'datas["txtValue1"].ToString()' is called. The callout text reads 'Gelen veriyi yazdırmak için kullanılır' (Used to print the received data).

```
ProductController.cs*  CreateProduct.cshtml
Örnek uygulama
38     return View();
39 }
40 }
41 [HttpGet]
42 public IActionResult CreateProduct() //varsayılan olarak HTTPGET'dir
43 {
44     return View();
45 }
46 [HttpPost]
47 public IActionResult CreateProduct(IFormCollection datas)
48 {
49     var value1 = datas["txtValue1"].ToString(); // Callout starts here
50     var value2 = datas["txtValue2"];
51     var value3 = datas["txtValue3"];
52     return View();
53 }
54
55 }
```



Paremetre ile Denk Gelecek Şekilde Veri Alma

```
        return View();
    }
    [HttpGet]
    public IActionResult CreateProduct() // varsayılan olarak HTTPGET'dir
    {
        return View();
    }
    [HttpPost]
    public IActionResult CreateProduct(string txtvalue1, string txtvalue2, string txtvalue3)
    {
        return View();
    }

```

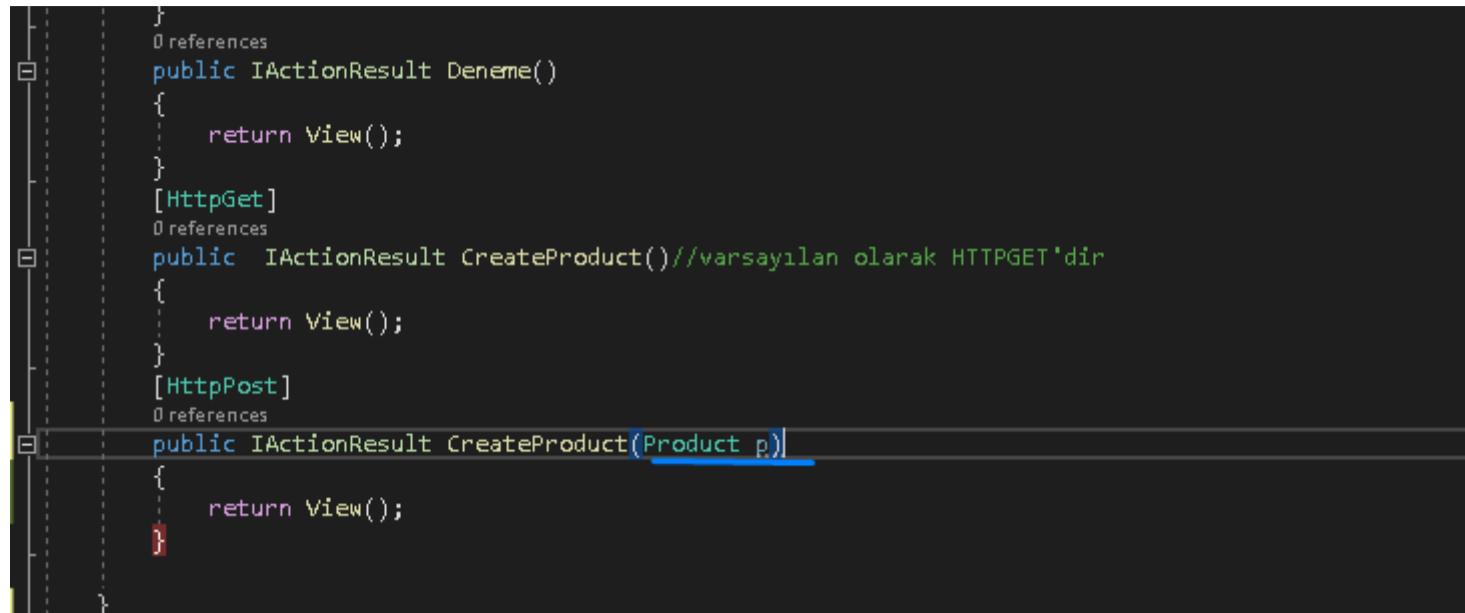
view'de denk gelecek değerlerle paralel bir şekilde
parametre olarak tanıma

Class üzerinden Veri Alma

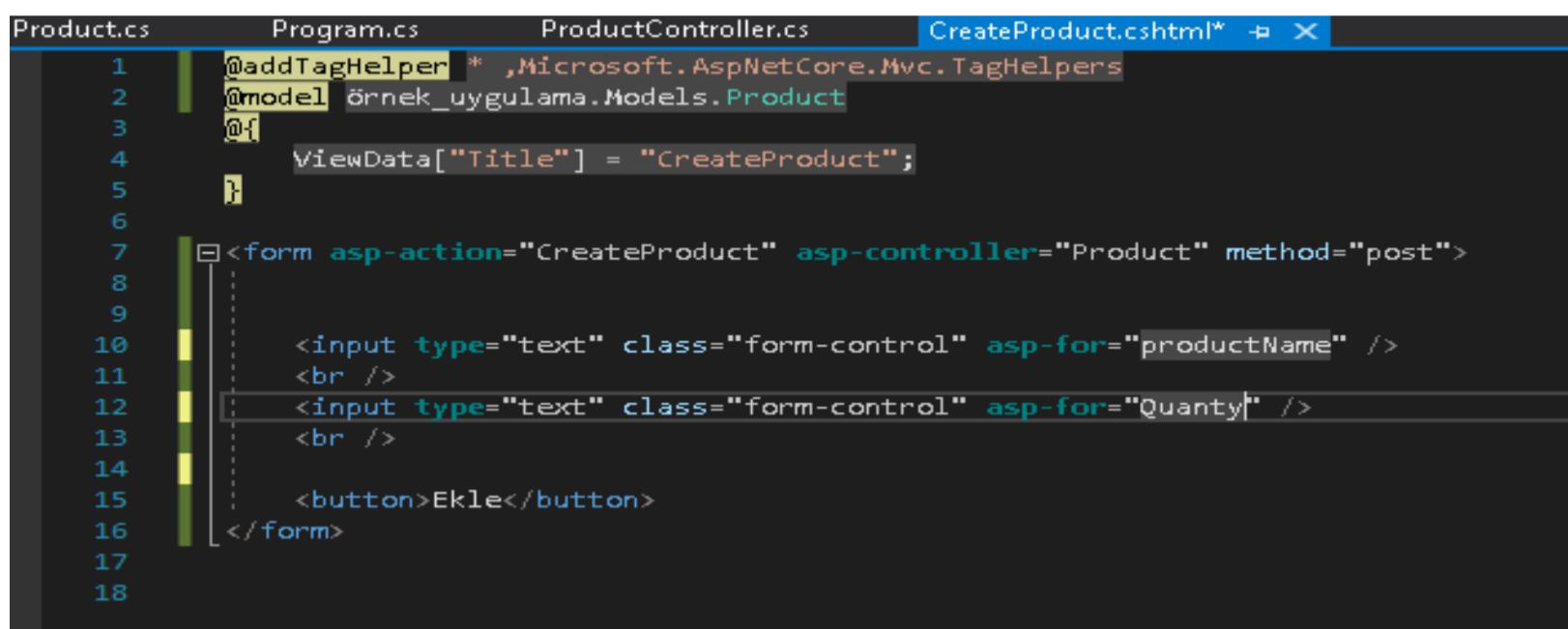
```
        var userProduct = (product, user);
        return View(userProduct);
    }
    public IActionResult Deneme()
    {
        return View();
    }
    [HttpGet]
    public IActionResult CreateProduct() // varsayılan olarak HTTPGET'dir
    {
        return View();
    }
    [HttpPost]
    public IActionResult CreateProduct(Model model)
    {
        return View();
    }
}

public class Model
{
    public string txtvalue1 { get; set; }
    public string txtvalue2 { get; set; }
    public string txtvalue3 { get; set; }
}
```

Models Üzerinden Veri Alma



```
        }
        0 references
    public IActionResult Deneme()
    {
        return View();
    }
    [HttpGet]
    0 references
    public IActionResult CreateProduct() //varsayılan olarak HTTPGET'dir
    {
        return View();
    }
    [HttpPost]
    0 references
    public IActionResult CreateProduct(Product p)
    {
        return View();
    }
}
```



```
Product.cs          Program.cs          ProductController.cs      CreateProduct.cshtml*  ✎ ✗
1  @addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
2  @model örnek_uygulama.Models.Product
3  @{
4      ViewData["Title"] = "CreateProduct";
5  }
6
7  <form asp-action="CreateProduct" asp-controller="Product" method="post">
8
9      <input type="text" class="form-control" asp-for="productName" />
10     <br />
11     <input type="text" class="form-control" asp-for="Quanty" />
12     <br />
13     <button>Ekle</button>
14
15 </form>
16
17
18
```

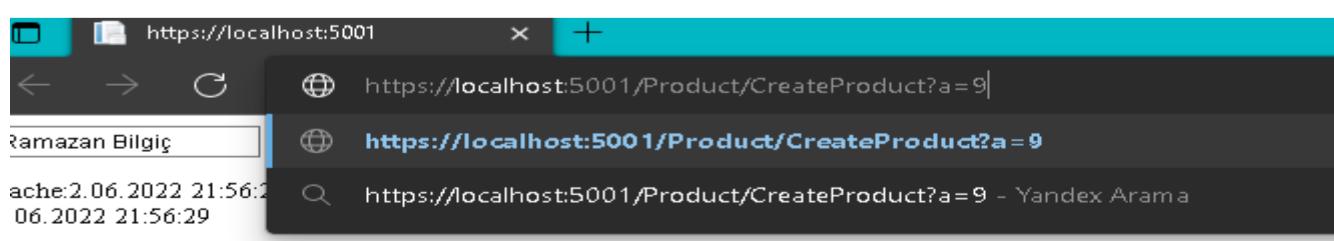
QueryString Üzerinden Veri Alma

Güvenlik gerektirmeyen url üzerinde taşınması için kullanılan yapılmamıştır.İstek yapılan serverlara/ servislere hızlı bir şekilde veri taşımıası için kullanılır.Yapılan Request'in türü ne olursa olsun query string değerleri taşınabilir.

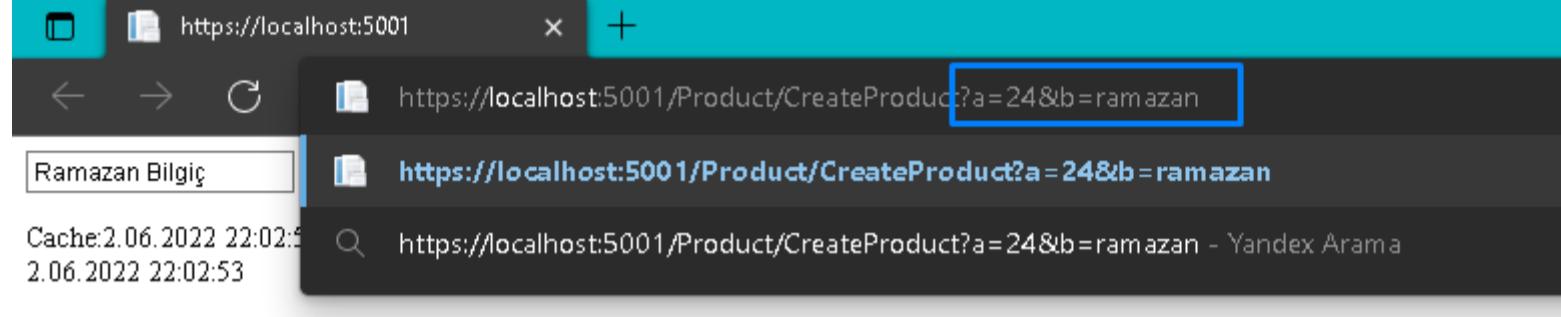
<https://.....com/sehir/ankara?ilce=4>



```
[HttpGet]
0 references
public IActionResult CreateProduct(string a) //varsayılan olarak HTTPGET'dir
{
    ≤ 34.471ms elapsed
    return View();
}
```



& Operatörü ile



Code snippets illustrating the use of the & operator in query strings:

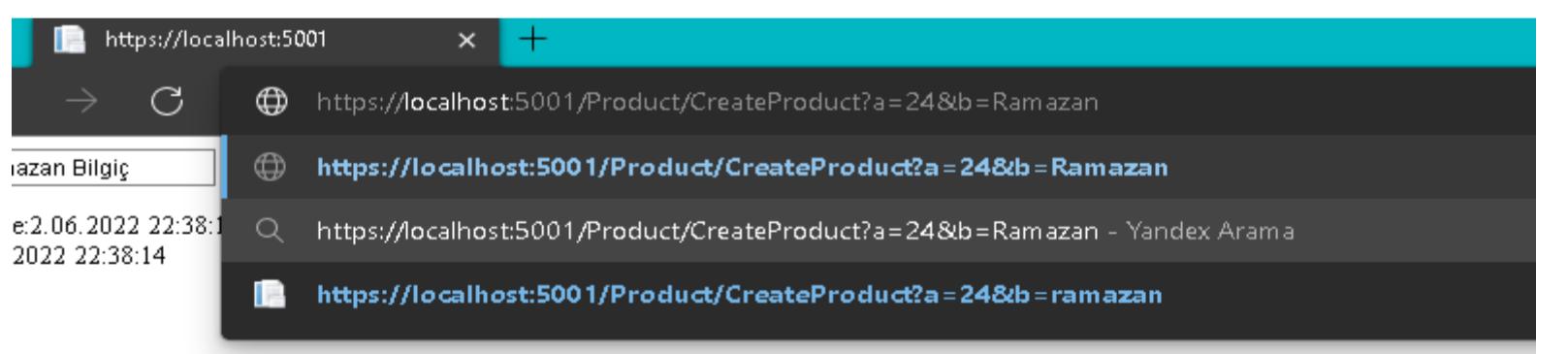
```
[HttpGet]
public IActionResult CreateProduct(string a,string b)//varsayılan olarak HTTPGET'de
{
    return View();
}
```

```
[HttpGet]
public IActionResult CreateProduct(string a,string b)//varsayılan olarak
{
    return View();
}
```

```
var queryString = Request.QueryString; //Request.QueryString
var a = Request.Query["a"].ToString();
var b = Request.Query["b"].ToString();
return View(); ≤ 3ms elapsed
```

```
0 references
public IActionResult VeriAl()
{
    var queryString = Request.QueryString; //Request yapılan endpoint'e query string parametresi eklenmiş mi eklenmemiş mi bununla ilgili bilgi verir.
    var a = Request.Query["a"];
    var b = Request.Query["b"];
    return View();
}
```

Query Data ile

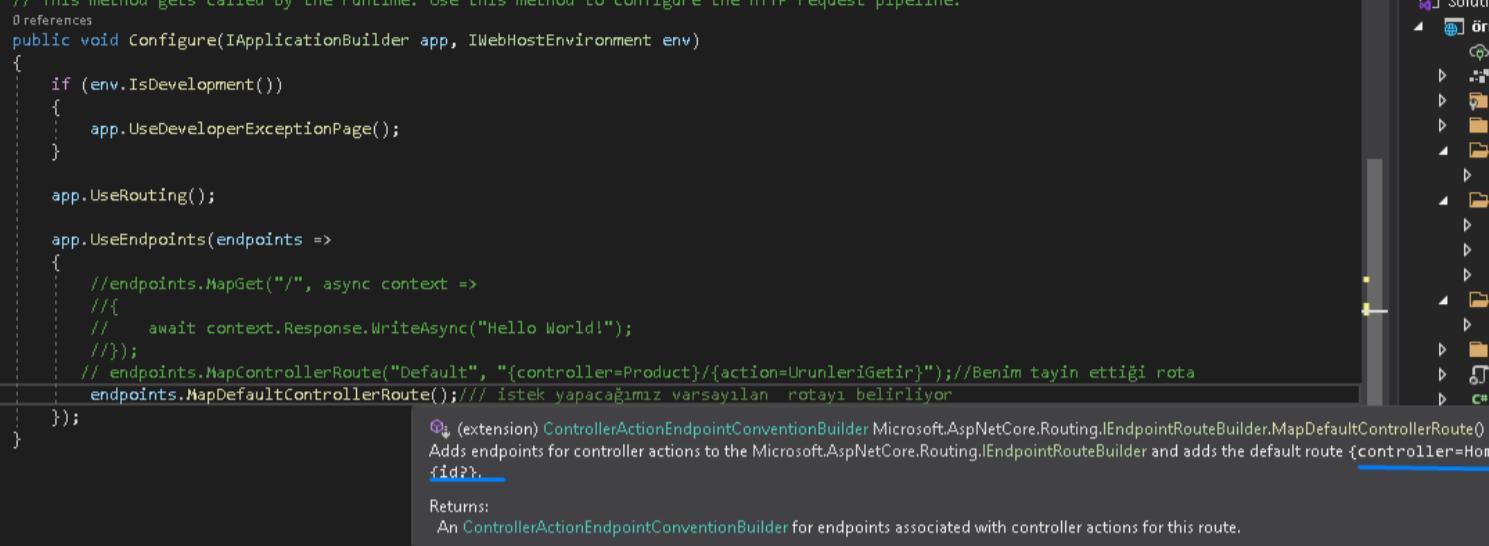


Code snippet illustrating the use of QueryData class to handle query parameters:

```
0 references
public IActionResult CreateProduct(QueryData data)
{
    return View();
}
```

```
1 reference
public class QueryData
{
    public string A { get; set; }
    public string B { get; set; }
}
```

Route Parameter Üzerinden Veri Alma



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

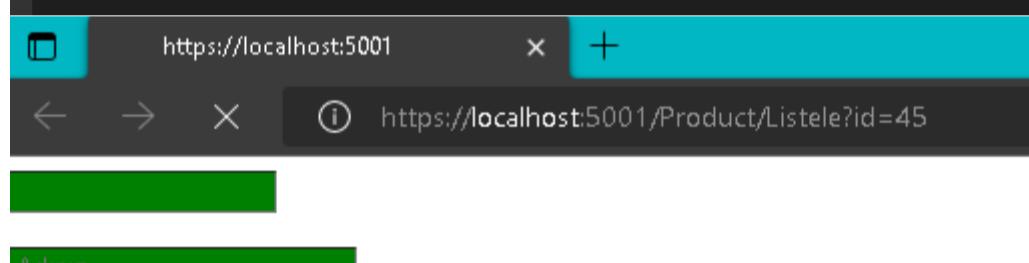
- Solution Explorer:** Shows the project "örnek uygulama" with files like Program.cs, Product.cs, ProductController.cs, CreateProduct.cshtml, and Startup.cs.
- Startup.cs:** The current file being edited, containing the following code:

```
21 }
22 }
23 // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
24 public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
25 {
26     if (env.IsDevelopment())
27     {
28         app.UseDeveloperExceptionPage();
29     }
30
31     app.UseRouting();
32
33     app.UseEndpoints(endpoints =>
34     {
35         //endpoints.MapGet("/", async context =>
36         //{
37         //    await context.Response.WriteAsync("Hello World!");
38         //});
39         // endpoints.MapControllerRoute("Default", "{controller=Product}/{action=UrunleriGetir}");//Benim tayin ettiğim rota
40         endpoints.MapDefaultControllerRoute();/// istek yapacağımız varsayılan rotayı belirliyor
41     });
42 }
43 }
44 }
```

A tooltip is displayed over the line `endpoints.MapDefaultControllerRoute();` with the following content:

ControllerActionEndpointConventionBuilder Microsoft.AspNetCore.Routing.IEndpointRouteBuilder.MapDefaultControllerRoute()
Adds endpoints for controller actions to the Microsoft.AspNetCore.Routing.IEndpointRouteBuilder and adds the default route {controller=Home}/{actions=Index}/{id?}.

Returns:
An ControllerActionEndpointConventionBuilder for endpoints associated with controller actions for this route.

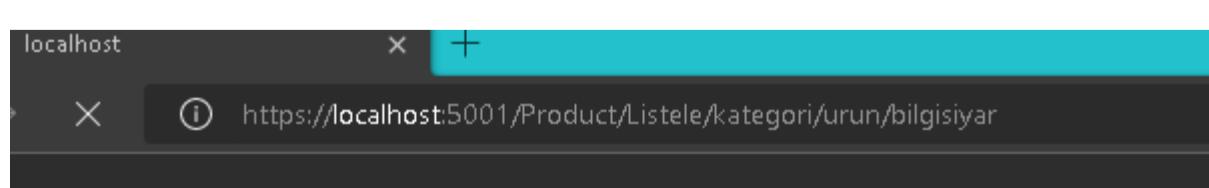


```
    }  
  
    0 references  
    public IActionResult Listele(string id)  
    {  
        ≤34.298ms elapsed  
        id  
        return View();  
    }  
  
    id  
    "45"  
}
```

```
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }

            app.UseRouting();

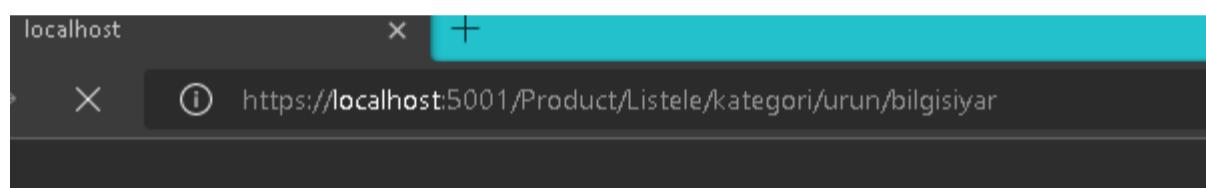
            app.UseEndpoints(endpoints =>
            {
                //endpoints.MapGet("/", async context =>
                //{
                //    await context.Response.WriteAsync("Hello World!");
                //});
                // endpoints.MapControllerRoute("Default", "{controller=Product}/{action=UrunleriGetir}");//Benim tayin ettiğim rota
                // endpoints.MapDefaultControllerRoute();/// istek yapacağımız varsayılan rotayı belirliyor
                endpoints.MapControllerRoute("CustomRoute", "{controller=Home}/{action=Index}/{a}/{b}/{id}");
            });
        }
    }
```



```
0 references
▶ public IActionResult Listele(string a,string b,string id)// parametre sıralaması önemli değil.
{ ≤49.946ms elapsed
    return View();
}
```

```
0 references
public IActionResult Listele(string a,string b,string id)// parametr
{ ≤49.946ms elapsed
    return View();
}
```

```
0 references
▶ public IActionResult Listele(string a,string b,string id)// parametre sıralaması önemli değil.
{ ≤49.946ms elapsed
    return View();
}
```



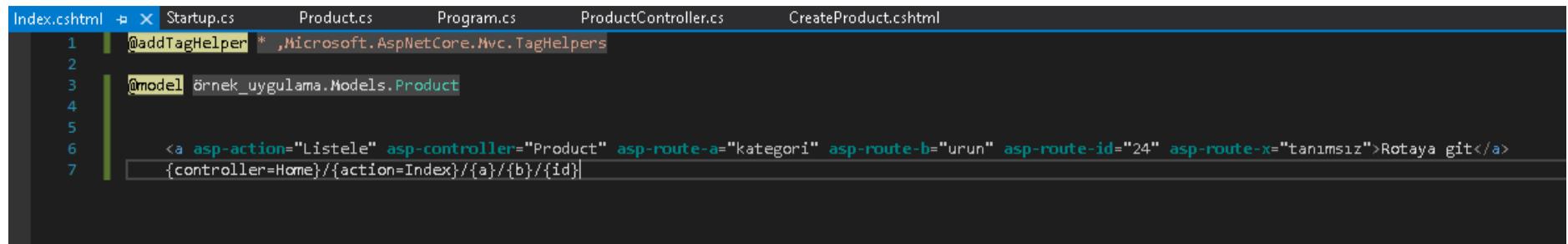
```
0 references
public IActionResult Listele(RouteData datas)
{ ≤64.509ms elapsed
    var values = Request.RouteValues;
    return View();
}
1 reference
public class RouteData
{
    0 references
    public string A { get; set; }
    0 references
    public string B { get; set; }
    0 references
    public string id { get; set; }
}
```

TagHelper ile İlgili Parametrelere Değer Atama

```
// This method gets called by the container and this method to configure the main request pipeline.
0 references
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    app.UseRouting();

    app.UseEndpoints(endpoints =>
    {
        //endpoints.MapGet("/", async context =>
        //{
        //    await context.Response.WriteAsync("Hello World!");
        //});
        // endpoints.MapControllerRoute("Default", "{controller=Product}/{action=UrunleriGetir}");//Benim t
        endpoints.MapDefaultControllerRoute();/// istek yapacağımız varsayılan rotayı belirliyor
        endpoints.MapControllerRoute("CustomRoute", "{controller=Home}/{action=Index}/{a}/{b}/{id}");
    });
}
```



```
Index.cshtml  X Startup.cs      Products.cs     Program.cs     ProductController.cs     CreateProduct.cshtml
1  @addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
2
3  @model örnek_ugulama.Models.Product
4
5
6  <a asp-action="Listele" asp-controller="Product" asp-route-a="kategori" asp-route-b="urun" asp-route-id="24" asp-route-x="tanimsiz">Rotaya git</a>
7  {controller=Home}/{action=Index}/{a}/{b}/{id}
```

[Rotaya git](https://localhost:5001/Product/Listele/24?a=kategori&b=urun&x=tanimsiz) { controller=Home } / { action=Index } / { a } / { b } / { id }

<https://localhost:5001/Product/Listele/24?a=kategori&b=urun&x=tanimsiz>

Header Üzerinden Veri Alma

```

Controller.cs
public string Id { get; set; }

public class ProductController : Controller
{
    public IActionResult GetProducts()
    {
        return View();
    }

    public IActionResult CreateProduct()
    {
        return View();
    }

    public IActionResult VeriAl()
    {
        var headers = Request.Headers.ToList();
        return headers;
    }
}

at Microsoft.AspNetCore.Mvc.ControllerBase.OnActionExecuted
Boolean& isCompleted)

```

Postman

File Edit View Help

+ New Import Runner My Workspace & Invite

No Environment

Untitled Request

GET https://localhost:5001/product/verial

Params Authorization Headers Body Pre-request Script Tests Settings

Headers 7 hidden

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> Adi	Gencay	
Key	Value	Description

Body Cookies (1) Headers (4) Test Results

Status: 500 Internal Server Error Time: 1901 ms Size:

Bootcamp

Ajax Tabanlı Veri Alma

Client tabanlı istek yapmamızı sağlayan ve bu isteklerin sonucu almamızı sağlayan javascript temelli mimaridir.

<https://jquery.com/> → ilgili linkden dökümanlara ulaşabilir.

Tuple Nesne Post Etme ve Yakalama

Örnek Uygulama

```

.cs Tuple.cshtml ProductController.cs
tuple.cshtml
önek_uygulama.Controllers.ProductController
55
56
57
58     0 references
59     public class QueryData
60     {
61         0 references
62             public string A { get; set; }
63             public string B { get; set; }
64
65     0 references
66     public IActionResult Tuple()
67     {
68         var tuple = (new Product(), new User());
69         return View(tuple);
70     }
71
72     [HttpPost]
73     0 references
74     public IActionResult CreateProduct([Bind(Prefix = "Item1")]Product p, [Bind(Prefix = "Item2")] User u)
75     {
76         return View();
77     }
78
79     1 reference
80     public class AjaxData
81     {
82         0 references

```

https://localhost:5001/Product/Tuple

Bilgisayar
Ramazan Bilgiç
Gönder

Startup.cs

```

Startup.cs Tuple.cshtml ProductController.cs
1 @addTagHelper * ,Microsoft.AspNetCore.Mvc.TagHelpers
2 @model önek_uygulama.Models.Product p, önek_uygulama.Models.User u
3 @{
4     ViewData["Title"] = "Tuple";
5 }
6
7 <form asp-action="CreateProduct" asp-controller="Product" method="post">
8     <input type="text" asp-for="p.productName" placeholder="Ürün adı" /><br /> <br />
9     <input type="text" asp-for="u.nameSurname" placeholder="Ad Soyad" /><br /> <br />
10    <button class="btn btn-info">Gönder</button>
11
12 </form>
13
14

```

Örnek Uygulama

```

Örnek Uygulama
0 references
public IActionResult Tuple()
{
    var tuple = (new Product(), new User());
    return View(tuple);
}

[HttpPost]
0 references
public IActionResult CreateProduct([Bind(Prefix = "Item1")]Product p, [Bind(Prefix = "Item2")] User u)
{
    return View();
}
1 reference
public class AjaxData
{
    0 references
}

```

p {örnek_uygulama.Models.Product} ↗

Id	0
Quanity	0
productName	"Bilgisayar"

Örnek Uygulama

```

Örnek Uygulama
0 references
public IActionResult Tuple()
{
    var tuple = (new Product(), new User());
    return View(tuple);
}

[HttpPost]
0 references
public IActionResult CreateProduct([Bind(Prefix = "Item1")]Product p, [Bind(Prefix = "Item2")] User u)
{
    return View();
}
1 reference

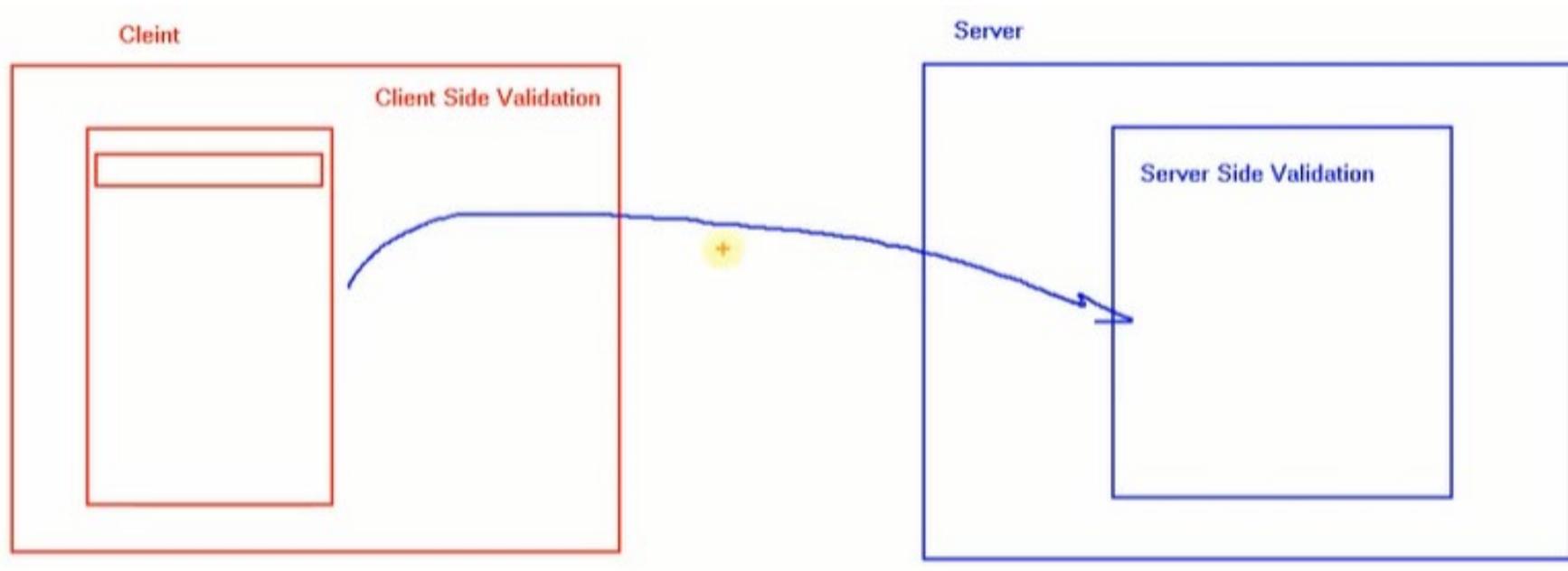
```

u {örnek_uygulama.Models.User} ↗

Id	0
age	0
nameSurname	"Ramazan Bilgiç"

VALIDATION(VERİ DOĞRULAMA)

KULLANICIDAN GELEN VERİNİN/DATANIN SERVER TARAFINDA KONTROL EDİLİR DOĞRULANMASIDIR.BİR DEĞERİN İÇERİNDE BULUNDUĞU ŞARTLARA UYGUN OLUP OLMADIĞININ KONTROL EDİLMESİDİR.DOĞRU VERİ ALINIR VERİ TABANINA KAYDEDİLİR. BU KONTROLE GÖRE İŞLEM TABİ TUTULUP TUTULMAMASINI KONTROL EDER. MİSAL; KULLANICIDAN MAILGİRMESİ İSTENİLDİĞİ BİR DURUMDA ARKA PLANDA MAIL FORMATINA GÖRE GİRİLİR GİRİLMEDİĞİNİ DENETLEYEN BİR KOMUT SATIRI/ SÜZGEÇ OLUŞTURULUP ŞARTA GİRİLEN MAIL FORMATI DOĞRU İSE BUNU İŞLEM TABİ TUTULMASI UYUYORSA EĞER BİR UYARI İLE ÇIKTI VERİLMESİ VE İŞLEM YAPILMAMASI DURUMUDUR. WEB YAZILIMINDA İKİ AŞAMALIDIR. 1. CLIENT (CLIENT SIDE VALIDATION) TARAFINDA BUNUN KONTROL EDİLMESİ VE DOĞRULANMASI DURUMUNDA, 2. AŞAMASI SERVER'DA(SERVER SIDE VALIDATION) KONTROL EDİLMESİDİR.



Validation : Paralel bir şekilde client ve server taraflarında uygulanmalıdır...

```
20 references
▶ public class Product
{
    [Required(ErrorMessage = "Lütfen ürün adını giriniz.")]
    [StringLength(35,ErrorMessage = "Lütfen ürün adını maximum 35 karakter giriniz")]
    4 references
    public string productName { get; set; }
    2 references
    public int Quanty { get; set; }
    [EmailAddress(ErrorMessage = "Lütfen doğru bir email adresi giriniz.")]
    1 reference
    public string Email { get; set; }
}
```

```

}
0 references
public IActionResult Validationsss()
{
    return View();
}

[HttpPost]
0 references
public IActionResult Validationsss(Product p)
{// ModelState: Mvc'de bir type'ın data annotations durumunu kontrol eden ve geriye
    property'dir
    if (!ModelState.IsValid)// sonuç false ise
    {
        ViewBag.hataMesaji = ModelState.Values.FirstOrDefault
            (x=>x.ValidationState == ValidationState.Invalid).Errors[0].ErrorMessage;
        //Loglama
        //kullanıcı bilgilendirme
        return View(p); ≤6ms elapsed
    }
    //Gelen datalar işleme/kontrole /operasyona /algoritmaya tabi tutulur
    return View();
}

```

Ürün
25
ramazanbilgic0947@gmail.com
Gönder

FORMA YAZDIRMA

```

.cs Validationsss.cshtml* ✘ X ProductController.cs
1 @addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
2 @model örnek_uygulama.Models.Product
3 @{
4     ViewData["Title"] = "Validationsss";
5 }
6
7 <form asp-action="Validationsss" asp-controller="Product" method="post">
8
9     <input type="text" asp-for="productName" placeholder="Ürün" /><span asp-validation-for="productName"></span><br /><br />
10    <input type="text" asp-for="Quanty" placeholder="Adet" /><br /><br /><span asp-validation-for="Quanty"></span>
11    <input type="email" asp-for="Email" placeholder="Email" /> <span asp-validation-for="Email"></span><br /><br />
12
13    <button>Gönder</button>
14
15
16
17 </form>
18
19

```

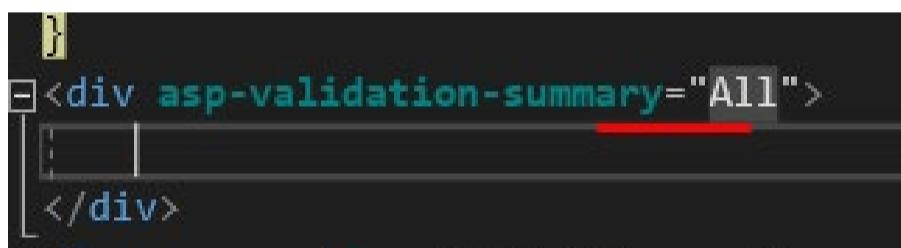
```

public IActionResult Validationsss()
{
    return View();
}

[HttpPost]
public IActionResult Validationsss(Product p)
{// ModelState: Mvc'de bir type'in data annotations durumunu kontrol etmek için bir property'dir
    if (!ModelState.IsValid)// sonuç false ise
    {
        //ViewBag.hataMesaji = ModelState.Values.FirstOrDefault(x=>x.ValidationState==Microsoft.AspNetCore.Mvc.ModelStateValidationStatus.Invalid).Errors[0].ErrorMessage;
        //Loglama
        //kullanıcı bilgilendirme
        var messges = ModelState.ToList();
        return View(p);
    }
    //Gelen datalar işleme/kontrole /operasyona /algoritma uygula
    return View();
}

```

HATALARI TEK BİR DIV İÇİNDE GÖSTERME



ModelMetadataType İle Validation Sorumluluğunu Başka Bir Sınıfa Yükleme

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Threading.Tasks;

namespace örnek_uygulama.Models.ModelMetadataTypes
{
    public class ProductMetaData
    {
        [Required(ErrorMessage = "Lütfen ürün adını giriniz.")]
        [StringLength(35, ErrorMessage = "Lütfen ürün adını maximum 35 karakter giriniz")]
        public string productName { get; set; }

        public int Quanty { get; set; }

        [EmailAddress(ErrorMessage = "Lütfen doğru bir email adresi giriniz.")]
        public string Email { get; set; }
    }
}

```

```
cshtml      ProductMetaData.cs      Product.cs  ✘  ProductController.cs
ulama      -  örnekleme.Models.Product      -  productName
using Microsoft.AspNetCore.Mvc;
using örnekleme.Models.ModelMetadataTypes;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Threading.Tasks;

namespace örnekleme.Models
{[ModelMetadataType(typeof(ProductMetaData))]/// ilgili validation kütüphanesi ve sınıfını tanımladık
    20 references
    public class Product
    {
        5 references
        public string productName { get; set; }
        3 references
        public int Quanty { get; set; }

        2 references
        public string Email { get; set; }
    }
}
```

FluentValidation Kütüphanesi İle Validation İşlemleri

The screenshot shows the NuGet Package Manager interface within a Visual Studio environment. The search bar at the top contains the text "fluent". The results list includes:

- FluentAssertions** by Dennis Doomen, Jonas Nyrup, 154M downloads, Version 6.7.0. Description: A very extensive set of extension methods that allow you to more naturally specify the expected outcome of a TDD or
- FluentValidation** by Jeremy Skinner, 150M downloads, Version 11.0.2. Description: A validation library for .NET that uses a fluent interface to construct strongly-typed validation rules.
- FluentValidation.AspNetCore** by Jeremy Skinner, 55.2M downloads, Version 11.0.2. Description: AspNetCore integration for FluentValidation. This item is highlighted with a red box.
- FluentValidation.DependencyInjectionExtension** by Jeremy Skinner, 2M downloads, Version 11.0.2. Description: Dependency injection extensions for FluentValidation.
- FluentMigrator** by Sean Chambers, Josh Coffman, Tom Marien, Mark Junker, 11.4M downloads, Version 3.3.2. Description: FluentMigrator is a database migration framework for .NET written in C#. The basic idea is that you can create migrations which are simply classes that derive from the Migration base class and have a Migrati...

On the right, the detailed view for **FluentValidation.AspNetCore** is shown, including its version (11.0.2), authors (Jeremy Skinner), license (Apache-2.0), and project URL (<https://fluentvalidation.net/>). The "Dependencies" section lists ".NETCoreApp,Version=v3.1" and "FluentValidation.DependencyInjectionExtensions (>= 11.0.2)".

```

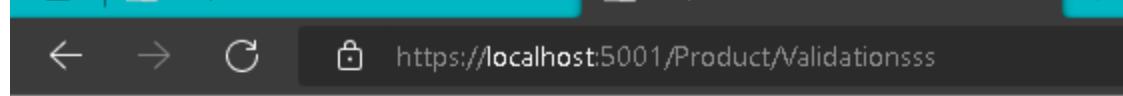
11
12     namespace örnek_uygulama
13     {
14         public class Startup
15         {
16             // This method gets called by the runtime. Use this method to add services to the container.
17             // For more information on how to configure your application, visit https://go.microsoft.com/fwlink/?LinkID=398940
18             public void ConfigureServices(IServiceCollection services)
19             {
20                 // services...
21                 // services.AddControllersWithViews(); // view yapılanması ile oluşturulması isteniyorsa
22                 // mvc mimarisi ile kalkındırılacağıını öğrendi. Artık uygulama Mvc davranışını sergileyebilcektir
23
24                 services.AddControllersWithViews().AddFluentValidation
25                     (x=>x.RegisterValidatorsFromAssemblyContaining<Startup>()); // Fluent kütüphanesinin entegre ediyoruz ve
26                     // tanımladığımız validationları sistemin kendisinin bulup uygulamasını söylüyoruz
27             }
28         }
29     }

```

```

1  using FluentValidation;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Threading.Tasks;
6
7  namespace örnek_uygulama.Models.Validators
8  {
9      public class ProductValidator : AbstractValidator<Product>
10     {
11         public ProductValidator()
12         {
13             RuleFor(x => x.Email).NotNull().WithMessage("Email Boş olamaz!");
14             RuleFor(x => x.Email).EmailAddress().WithMessage("Lütfen Geçerli bir Email Giriniz.");
15             RuleFor(x => x.productName).NotNull().NotEmpty().WithMessage("Lütfen Ürün İsmini Boş geçmeyin.");
16             RuleFor(x => x.productName).MaximumLength(35).WithMessage("Ürün Adını Max. 35 Karakter Giriniz.");
17             RuleFor(x => x.productName).MinimumLength(2).WithMessage("Ürün Adını Min. 2 Karakter Giriniz.");
18         }
19     }
20 }

```



- Lütfen ürün adını giriniz.
- 'product Name' boş olamaz.
- Lütfen Ürün İsmini Boş geçmeyin.
- The value " is invalid.
- Email Boş olamaz!

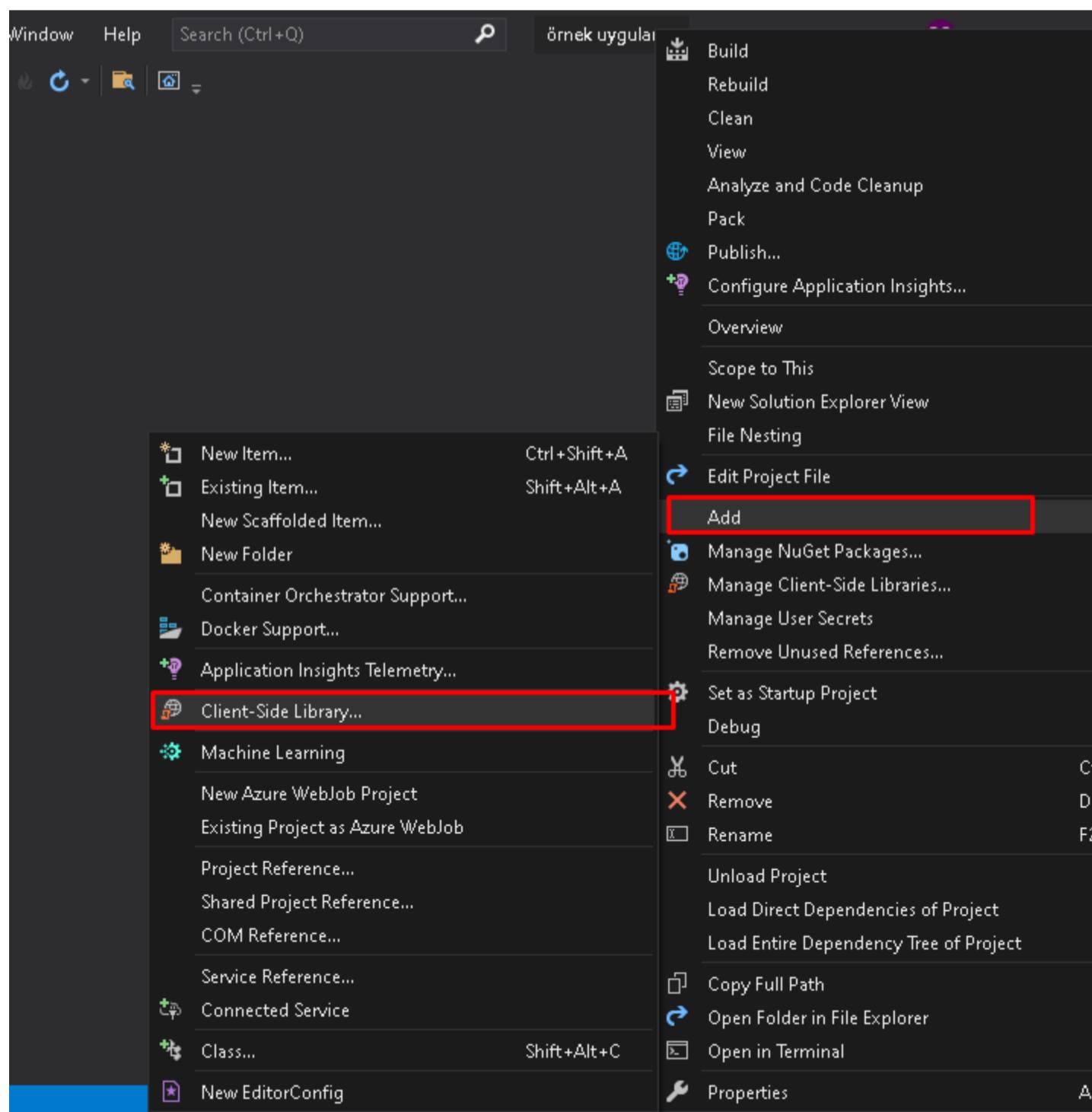
Lütfen ürün adını giriniz.

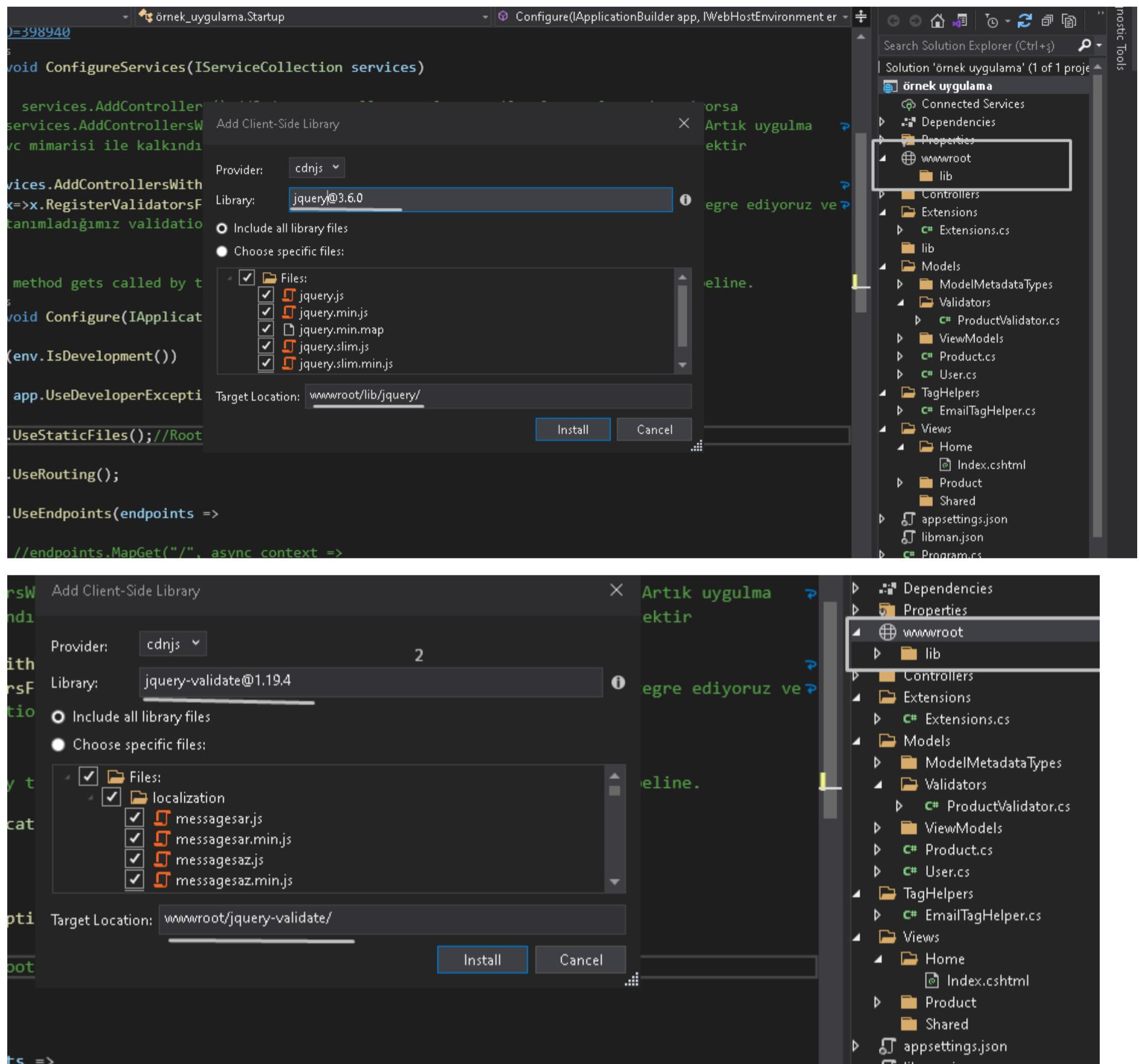
The value " is invalid.

Email Boş olamaz!

Server'da ki Validation'ları Dinamik Olarak Client Tabanlı Uygulamak

```
Startup.cs*  X  önek uygulama  Configure(IApplicationBuilder app, IWebHostEnvironment env)  Solution Explorer
18     public void ConfigureServices(IServiceCollection services)
19     {
20         // services.AddControllers(); // Sadece controller yapılanması ile oluşturulması isteniyorsa
21         // services.AddControllersWithViews(); // view yapılanması ile oluşturulması isteniyorsa. Artık uygulma
22             // mvc mimarisi ile kalkındırılacağını öğrendi. Artık uygulama Mvc davranışını sergileyebilcektir
23
24         services.AddControllersWithViews().AddFluentValidation
25             (<x=>x.RegisterValidatorsFromAssemblyContaining<Startup>()); // Fluent kütüphanesinin entegre ediyoruz ve
26             // tanımladığımız validationları sistemin kendisinin bulup uygulamasını söylüyoruz
27
28     }
29
30     // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
31     public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
32     {
33         if (env.IsDevelopment())
34         {
35             app.UseDeveloperExceptionPage();
36
37             app.UseStaticFiles(); // Root dizinini tanımamı için belirtiyorum
38
39             app.UseRouting(); // Enables static file serving for the current request path
40
41             app.UseEndpoints(endpoints =>
42                 {
43                     //endpoints.MapGet("/", async context =>
44                     //{
45                         //await context.Response.WriteAsync("Hello World!");
46                     //});
47                 });
48         }
49     }
50 }
```





Örnek uygulama

Artık uygulama ektir

egre ediyoruz ve

orsa

Add Client-Side Library

Provider: cdnjs

Library: jquery-validation-unobtrusive@3.2.12

Include all library files

Choose specific files:

Files:
 jquery.validate.unobtrusive.js
 jquery.validate.unobtrusive.min.js

Target Location: wwwroot/jquery-validation-unobtrusive/

Install Cancel

Validationss.cshtml

```

1 @addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
2 @model örnek uygulama.Models.Product
3
4 <script src="~/jquery/jquery.min.js"></script>
5
6 <script src="~/jquery-validate/jquery.validate.min.js"></script>
7
8 <script src="~/jquery-validation-unobtrusive/jquery.validate.unobtrusive.min.js"></script>
9
10 <form asp-action="Validationss" asp-controller="Product" method="post">
11
12     <input type="text" asp-for="productName" placeholder=" Ürün" />
13     <span asp-validation-for="productName"></span>
14     <br /><br />
15     <input type="text" asp-for="Quany" placeholder=" Adet" />
16     <span asp-validation-for="Quany"></span><br /><br />
17     <input type="email" asp-for="Email" placeholder=" Email" />
18     <span asp-validation-for="Email"></span>
19     <br /><br />
20
21     <button>Gönder</button>
22
23
24
25
26
27

```

115 % No issues found

Ln: 8 Ch: 1 SPC CRLF

Solution Explorer

Search Solution Explorer (Ctrl + S)

Örnek uygulama

Connected Services

Dependencies

Properties

wwwroot

jquery

jquery-validate

lib

Controllers

Extensions

C# Extensions.cs

Models

ModelMetadataTypes

Validators

C# ProductValidator.cs

ViewModels

C# Products.cs

C# User.cs

TagHelpers

C# EmailTagHelper.cs

Views

Home

Index.cshtml

Product

Shared

anycaching.json

jquery

jquery.js

jquery.min.js

jquery.slim.js

jquery-validate

localization

additional-methods.js

jquery.validate.js

jquery.validate.min.js

jquery-validation-srijson

jquery-validation-unobtrusive

jquery.validate.unobtrusive.js

jquery.validate.unobtrusive.min.js

lib

Controllers

C# HomeController.cs

C# ProductController.cs

Extensions

Models

TagHelpers

Views

appsettings.json

libman.json

https://localhost:5001/Product/Validationss

← → ⌂ https://localhost:5001/Product/Validationss

Ürün Lütfen ürün adını giriniz.

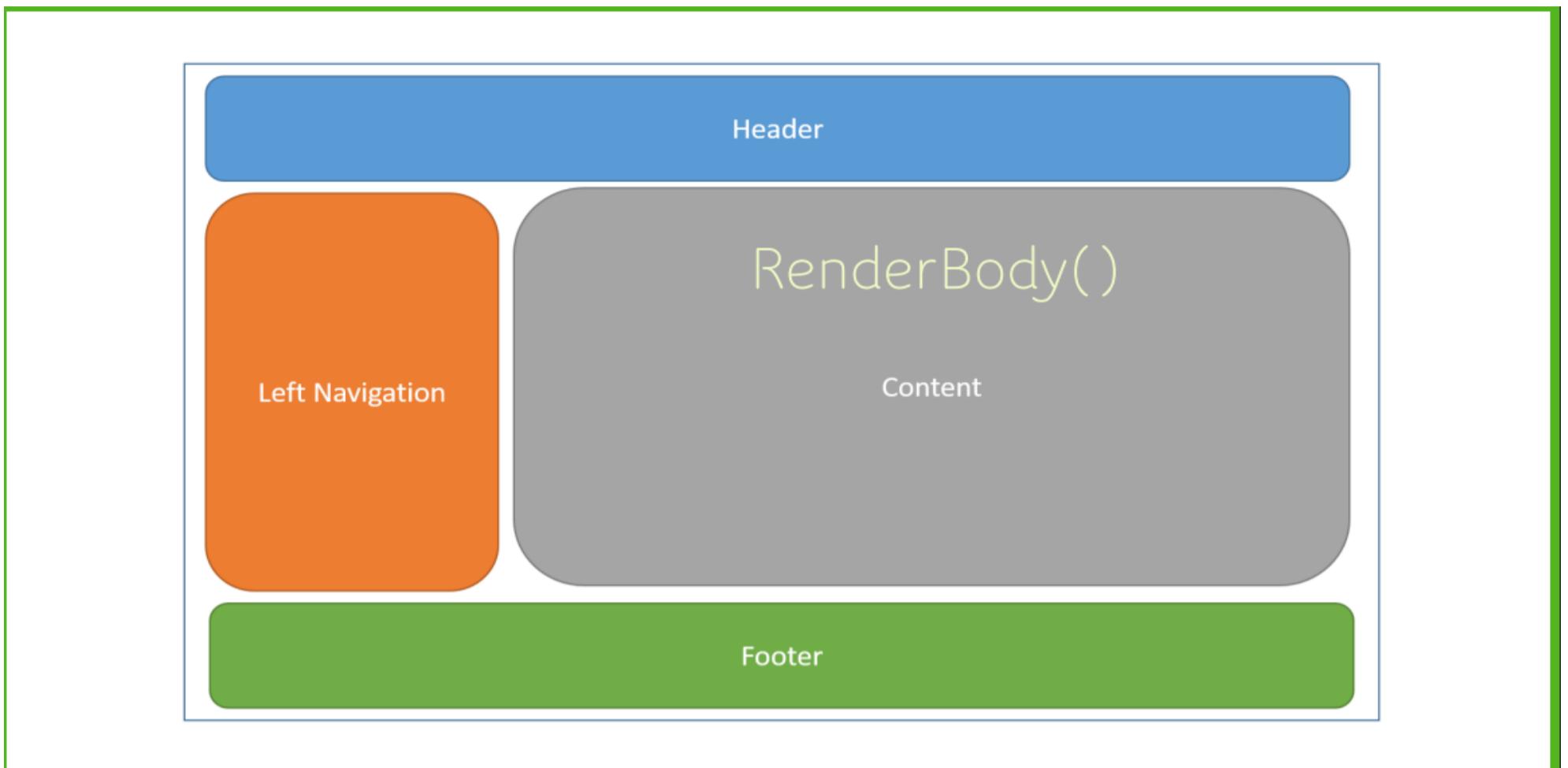
Adet The Quany field is required.

Email Email Boş olamaz!

Gönder

Layout Yapılanması Nedir? RenderBody ve RenderSection Fonksiyonları Nelerdir?

Layout Nedir? Web sayfalarının olmazsa olmazıdır. Sayfadan sayfaya gezinirken, kullanıcıya tutarlı bir deneyim sağlayan ortak sayfa taslağıdır. Tutarlı bir düzene sahip bir web sitesi oluşturmak için kullanılır.



The screenshot shows the Visual Studio IDE interface. On the left, the code editor displays the `_Layout.cshtml` file with the following code:

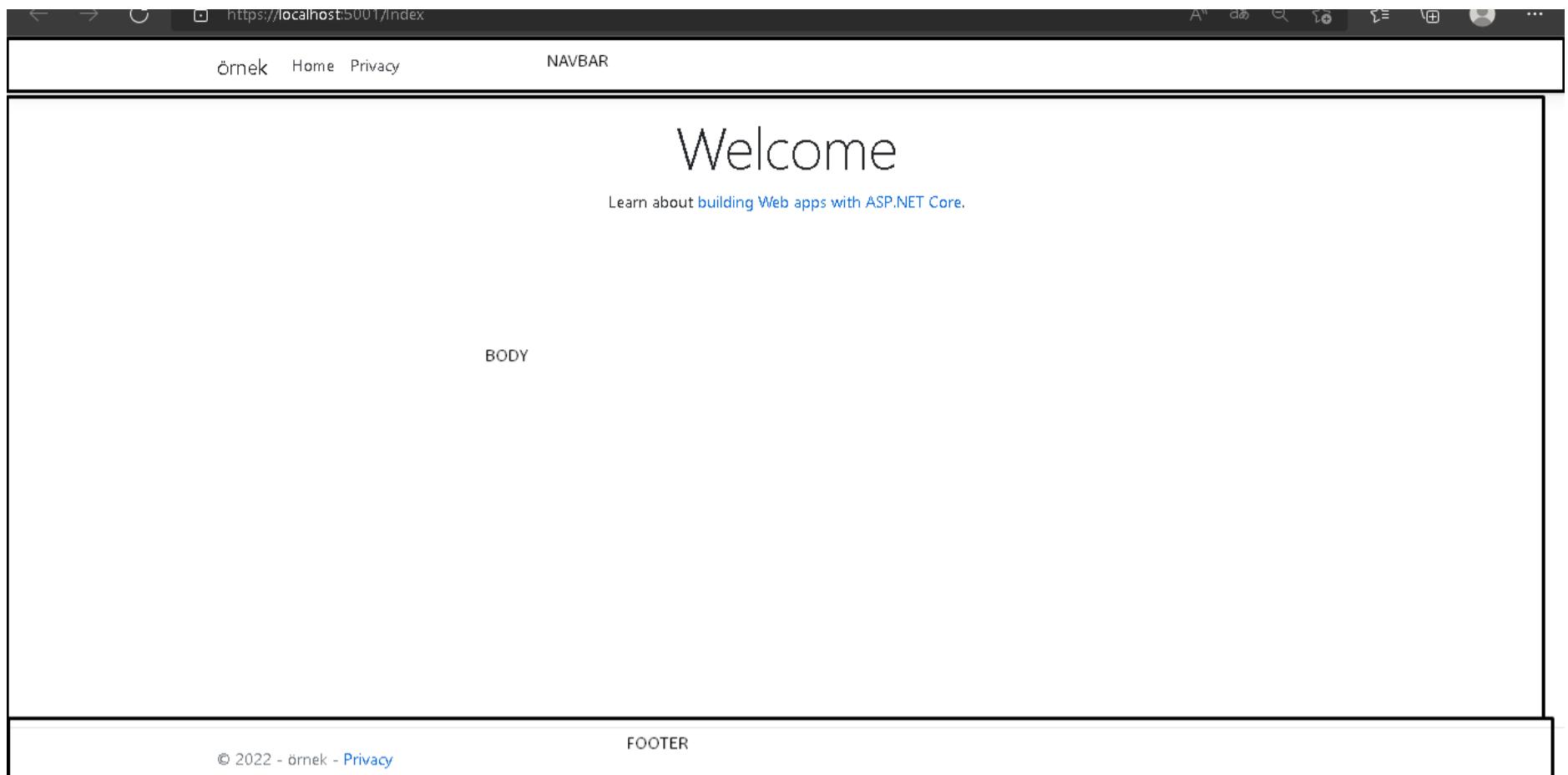
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - örnek</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" />
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">
            <div class="container">
                <a class="navbar-brand" asp-area="" asp-page="/Index">örnek</a>
                <button class="navbar-toggler" type="button" data-toggle="collapse" data-target=".navbar-collapse" aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
                    <ul class="navbar-nav flex-grow-1">
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-page="/Index">Home</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-page="/Privacy">Privacy</a>
                        </li>
                    </ul>
                </div>
            </div>
        </nav>
    </header>
    <div class="container">
        <div>
        </div>
    </div>
</body>
```

On the right, the Solution Explorer window shows the project structure:

- Solution 'örnek' (1 of 1 project)
 - örnek
 - Connected Services
 - Dependencies
 - Properties
 - wwwroot
 - Pages
 - Shared
 - _Layout.cshtml
 - _ValidationScriptsPartial.cshtml
 - ViewImports.cshtml
 - ViewStart.cshtml
 - Error.cshtml
 - Index.cshtml
 - Privacy.cshtml
 - appsettings.json
 - appsettings.Development.json
 - Program.cs
 - Startup.cs

The screenshot shows the code editor with the `Details.cshtml` file open. The code includes directives to set the model and layout:

```
@model satis.Models.OrderDetailsVM
 @{
    ViewData["Title"] = "Details";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```



RenderBody: Render edilen view'lerin layout'ta nereye basılacağını temsil eder.
Layout'ta tanımlanması zorunludur.

```
13 <div class="container">
14     <a class="navbar-brand" asp-area="" asp-page="/Index">örnek</a>
15     <button class="navbar-toggler" type="button" data-toggle="collapse" data-target=".navbar-collapse"
16         aria-expanded="false" aria-label="Toggle navigation">
17         <span class="navbar-toggler-icon"></span>
18     </button>
19     <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
20         <ul class="navbar-nav flex-grow-1">
21             <li class="nav-item">
22                 <a class="nav-link text-dark" asp-area="" asp-page="/Index">Home</a>
23             </li>
24             <li class="nav-item">
25                 <a class="nav-link text-dark" asp-area="" asp-page="/Privacy">Privacy</a>
26             </li>
27         </ul>
28     </div>
29 </div>
30 </nav>
31 </header>
32 <div class="container">
33     <main role="main" class="pb-3">
34         @RenderBody()
35     </main>
36 </div>
37
38 <footer class="border-top footer text-muted">
39     <div class="container">
40         &copy; 2022 - örnek - <a asp-area="" asp-page="/Privacy">Privacy</a>
41     </div>
```

The code editor shows the 'Layout.cshtml' file. Line 34 contains the '@RenderBody()' directive, which is highlighted with a red box and an arrow pointing to the text 'Body,content'in sergilendiği kism' (The part where Body,content is displayed).

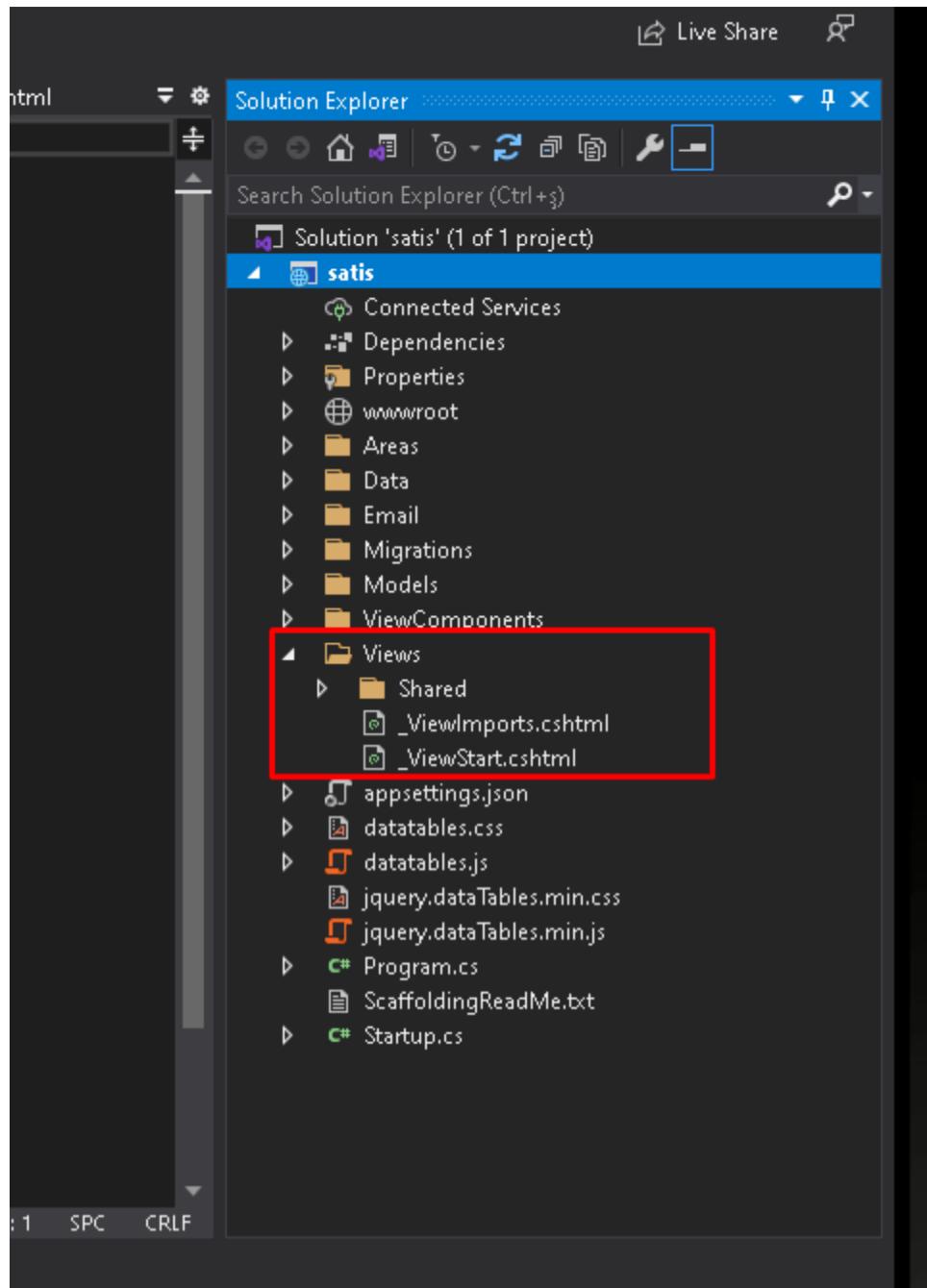
RenderSection: Layout içerisinde isimsel bölümleri oluşturmak için kullanılır. İhtiyaç doğrultusunda bu bölümlere render edilen view'lerden de içerikler atanabilir. Genellikle; JS referansları,sayfadan sayfaya fark eden alanlarda kullanılır.

A screenshot of the Visual Studio code editor showing the `_Layout.cshtml` file. The cursor is over the line `@await RenderSectionAsync("Scripts", required: false)`. A tooltip box appears with the text:

İlgili Section kullanılmiyorsa eğer hata vermicek bu durumda
Ha! kullaniliyorsa zaten sorun yok

A screenshot of the Visual Studio code editor showing the `Index.cshtml` file. The cursor is over the line `@section Scripts{`. A tooltip box appears with the text:

İlgili Section ve ismi



_ViewStart Dosyası Nedir?

Asıl amacı tüm view'lerde
kullanılması/yapılması gereken
ortak çalışmaların yapıldığı view'dır.

Bir nevi tüm view'lerin atasıdır diyebiliriz.

Views klasörü altında _ViewStart.cshtml
olarak oluşturulması gereklidir.

Genellikle tüm view'lerin ortak kullanacağı
Layout tanımlaması bu dosya içerisinde
gerçekleştirilir.

```
_ViewImports.cshtml _ViewStart.cshtml _ViewStart.cshtml ProductDetails.cshtml CategoryDetails.cshtml
1  @{
2      Layout = "_Layout";
3  }
4
```



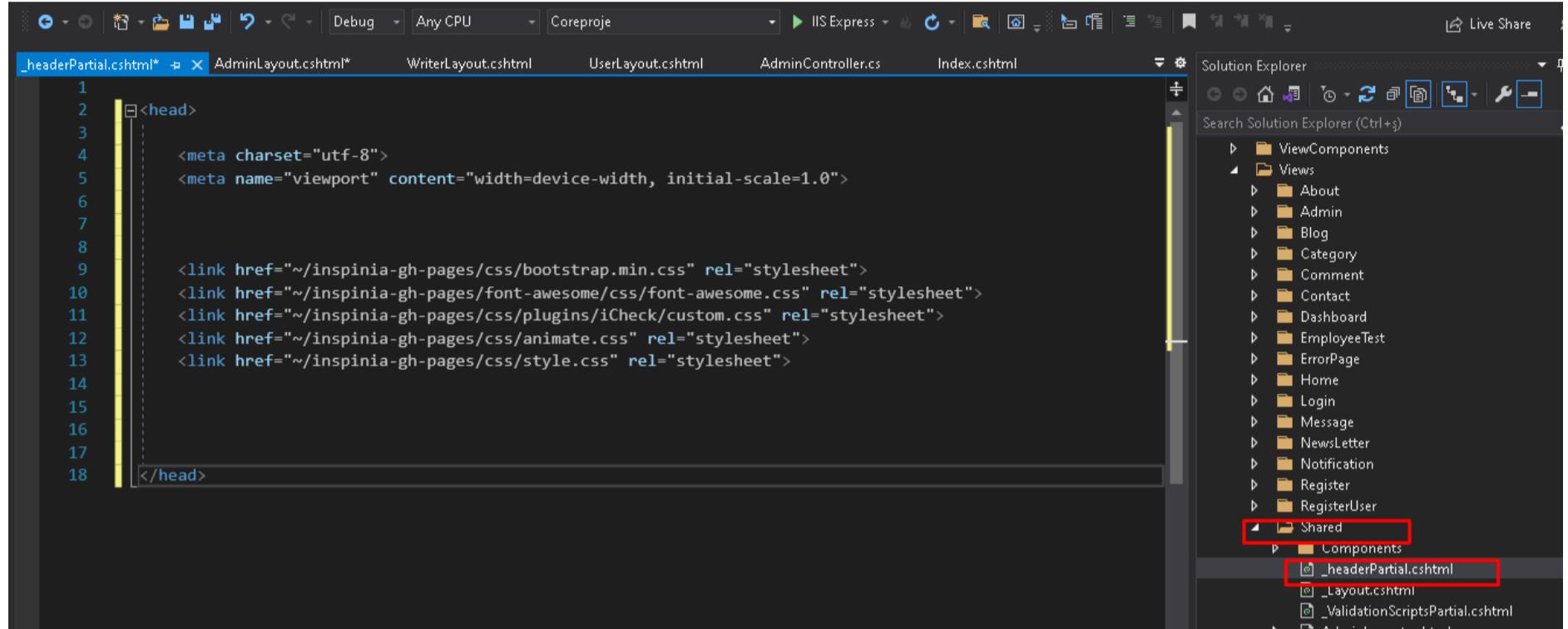
```
_ViewImports.cshtml _ViewStart.cshtml _ViewStart.cshtml ProductDetails.cshtml
1  @using satis
2  @using satis.Models
3  @addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
4
```

Modüler Tasarım Yapılanması Nedir? Nasıl Uygulanır?

Örneğin bir araba bir bütün olarak tek olsaydı bir hata olduğunda bir bütününe değişmesi lazımdı ve bu maliyet ve zaman kaybıdır, ama bu araba modüler yapıda olduğu için bir

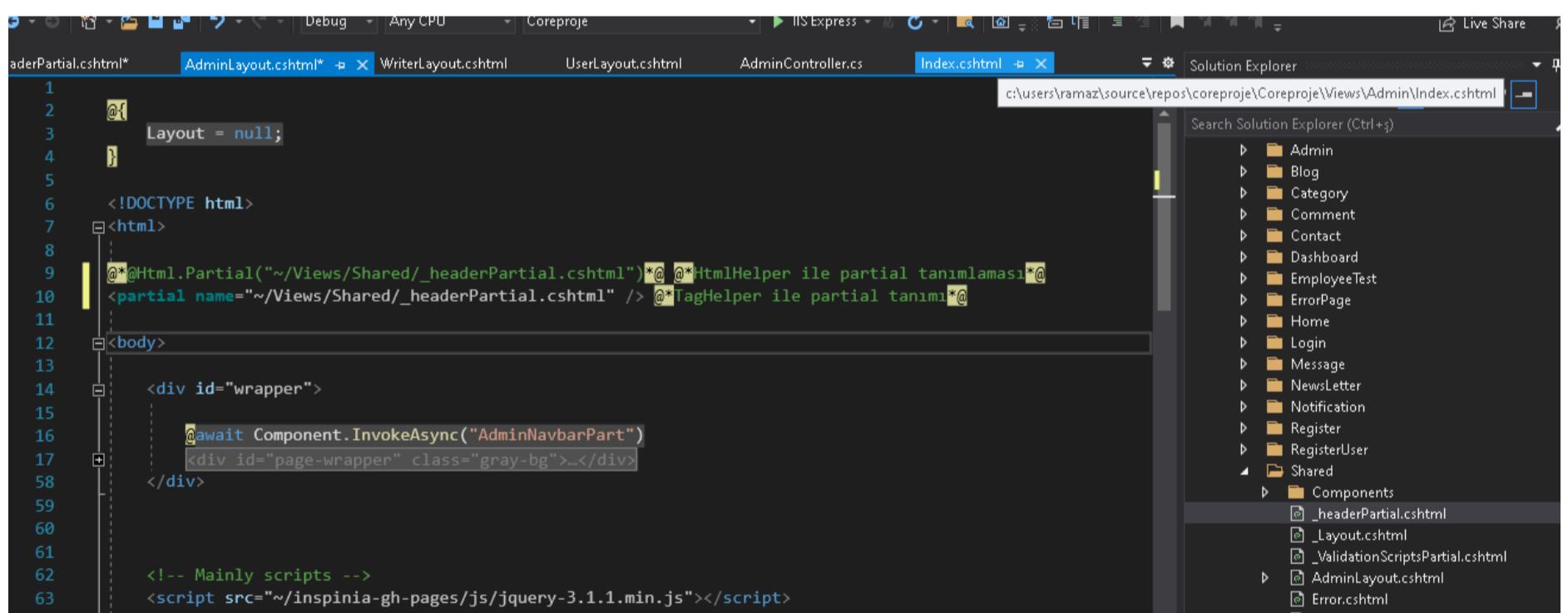
modüldeki hata veya değişim diğer modüllere ellemeden zarar vermeden daha az maliyetle süreç halolur.

PartialView: Modüler tasarımda her bir modülün ayrı bir .cshtml/parça olarak tasarlanması ve ihtiyaç doğrultusunda ilgili parçanın çağrılmamasını sağlayan bir yöntemdir.



The screenshot shows the Visual Studio IDE interface. In the top navigation bar, the tabs include 'Coreprojek' (highlighted), 'Debug', 'Any CPU', 'Coreprojek', 'IIS Express', and several other tabs like 'AdminController.cs' and 'Index.cshtml'. On the left, the 'Solution Explorer' pane is open, showing a tree structure of files and folders. A red box highlights the 'Shared' folder under 'Views', which contains '_headerPartial.cshtml'. Other files in the Shared folder include '_Layout.cshtml' and '_ValidationScriptsPartial.cshtml'. The main code editor window displays the content of '_headerPartial.cshtml', which includes the head section of an HTML document with meta tags and CSS links.

```
1<head>
2    <meta charset="utf-8">
3    <meta name="viewport" content="width=device-width, initial-scale=1.0">
4
5    <link href="~/inspinia-gh-pages/css/bootstrap.min.css" rel="stylesheet">
6    <link href="~/inspinia-gh-pages/font-awesome/css/font-awesome.css" rel="stylesheet">
7    <link href="~/inspinia-gh-pages/css/plugins/iCheck/custom.css" rel="stylesheet">
8    <link href="~/inspinia-gh-pages/css/animate.css" rel="stylesheet">
9    <link href="~/inspinia-gh-pages/css/style.css" rel="stylesheet">
10
11
12
13
14
15
16
17
18</head>
```



This screenshot shows the Visual Studio IDE with the 'Index.cshtml' file selected in the top navigation bar. The code editor contains the following C# Razor syntax for rendering a partial view:

```
1@{
2    Layout = null;
3}
4
5<!DOCTYPE html>
6<html>
7    @*Html.Partial("~/Views/Shared/_headerPartial.cshtml")*@
8    @*HtmlHelper ile partial tanımlaması*@
9    <partial name="~/Views/Shared/_headerPartial.cshtml" /> @*TagHelper ile partial tanımı*@
10
11<body>
12
13    <div id="wrapper">
14        @await Component.InvokeAsync("AdminNavbarPart")
15        <div id="page-wrapper" class="gray-bg">...</div>
16    </div>
17
18    <!-- Mainly scripts -->
19    <script src="~/inspinia-gh-pages/js/jquery-3.1.1.min.js"></script>
20
```

The 'Solution Explorer' pane on the right shows the project structure, with a red box highlighting the 'Shared' folder under 'Views', which contains '_headerPartial.cshtml'. Other files in the Shared folder include '_Layout.cshtml' and '_ValidationScriptsPartial.cshtml'.

PartialView Veri Gönderme Kritiği

Solution Explorer

```

10
11     namespace Coreproje.Controllers
12     {
13
14         public class AdminController : Controller
15         {
16             UserManager um = new UserManager(new EfUserRepository());
17             Context c = new Context();
18             public IActionResult Index()
19             {
20                 return View();
21             }
22             public PartialViewResult AdminNavbarPartial()
23             {
24                 var username = User.Identity.Name;
25                 ViewBag.name = username;
26
27                 var usermail = c.Users.Where(x => x.UserName == username).Select(y => y.Email).FirstOrDefault();
28                 var userID = c.Users.Where(x => x.Email == usermail).Select(y => y.Id).FirstOrDefault();
29
30                 var values = um.TGetByID(userID);
31
32                 return PartialView(values);
33             }
34         }
35     }

```

Solution Explorer

```

1 @model EntityLayer.Concrete.AppUser
2
3
4
5 <nav class="navbar-default navbar-static-side" role="navigation">
6
7     <div class="sidebar-collapse">
8         <ul class="nav metismenu" id="side-menu">
9             <li class="nav-header">
10                 <div class="dropdown profile-element">
11                     
12                     <span> An img element represents an image.
13                     <span> Learn more (F1)
14                     <a data-toggle="dropdown" class="dropdown-toggle" href="/Admin/AdminProfile/Profile/">
15                         <span class="block m-t-xs font-bold">@Model.NameSurname </span>
16                         <span class="text-muted text-xs block">@Model.UserTitle <b><span> caret</span></b></span>
17                     </a>
18                     <ul class="dropdown-menu animated fadeInRight m-t-xs">
19                         <li><a class="dropdown-item" href="/Admin/AdminProfile/Profile/">Profile</a></li>
20                         <li><a class="dropdown-item" href="/Admin/AdminContact/ContactList/">Contacts</a></li>
21                         <li><a class="dropdown-item" href="/Admin/AdminMessage/InBox/">Mailbox</a></li>
22                         <li class="dropdown-divider"></li>
23                         <li><a class="dropdown-item" href="/Login/LogOut/">Logout</a></li>
24                     </ul>
25                 </div>
26                 <div class="logo-element">
27                     IN+
28                 </div>
29             </li>
30         </ul>
31     </div>
32 
```

Solution Explorer

```

4
5
6     <!DOCTYPE html>
7     <html>
8
9         @*Html.Partial("~/Views/Shared/_headerPartial.cshtml")*@
10        @*HtmlHelper ile partial tanımlaması*@
11        @*TagHelper ile partial tanımı*@
12
13        <body>
14            <div id="wrapper">
15                <partial name="~/Views/Admin/AdminNavbarPartial.cshtml" />
16
17
18
19
20
21            <div id="page-wrapper" class="gray-bg">
22                <div class="row border-bottom">
23                    <nav class="navbar navbar-static-top" role="navigation" style="margin-bottom: 0">
24                        <div class="navbar-header">
25                            <a class="navbar-minimize minimize-styl-2 btn btn-primary" href="#"><i class="fa fa-bars"></i></a>
26                            <form role="search" class="navbar-form-custom" action="search_results.html">
27                                <div class="form-group">
28                                    <input type="text" placeholder="Search for something..." class="form-control" name="q">
29                                </div>
30                            </form>
31                        </div>
32                    </nav>
33
34                    <div id="page-inner">
35                        <div class="row" style="padding-top: 10px;">
36                            <div class="col-md-12" style="text-align: center; margin-bottom: 20px;">
37                                <h1>Welcome to Admin Panel</h1>
38                                <img alt="Admin Panel Logo" style="width: 150px; height: auto; margin: 10px 0;"/>
39                                <p>This is the Admin Panel for your application. You can manage users, posts, comments, and more here. Stay safe and happy coding!</p>
40                            </div>
41                        </div>
42
43                    </div>
44
45                </div>
46            </div>
47
48        </body>
49    </html>

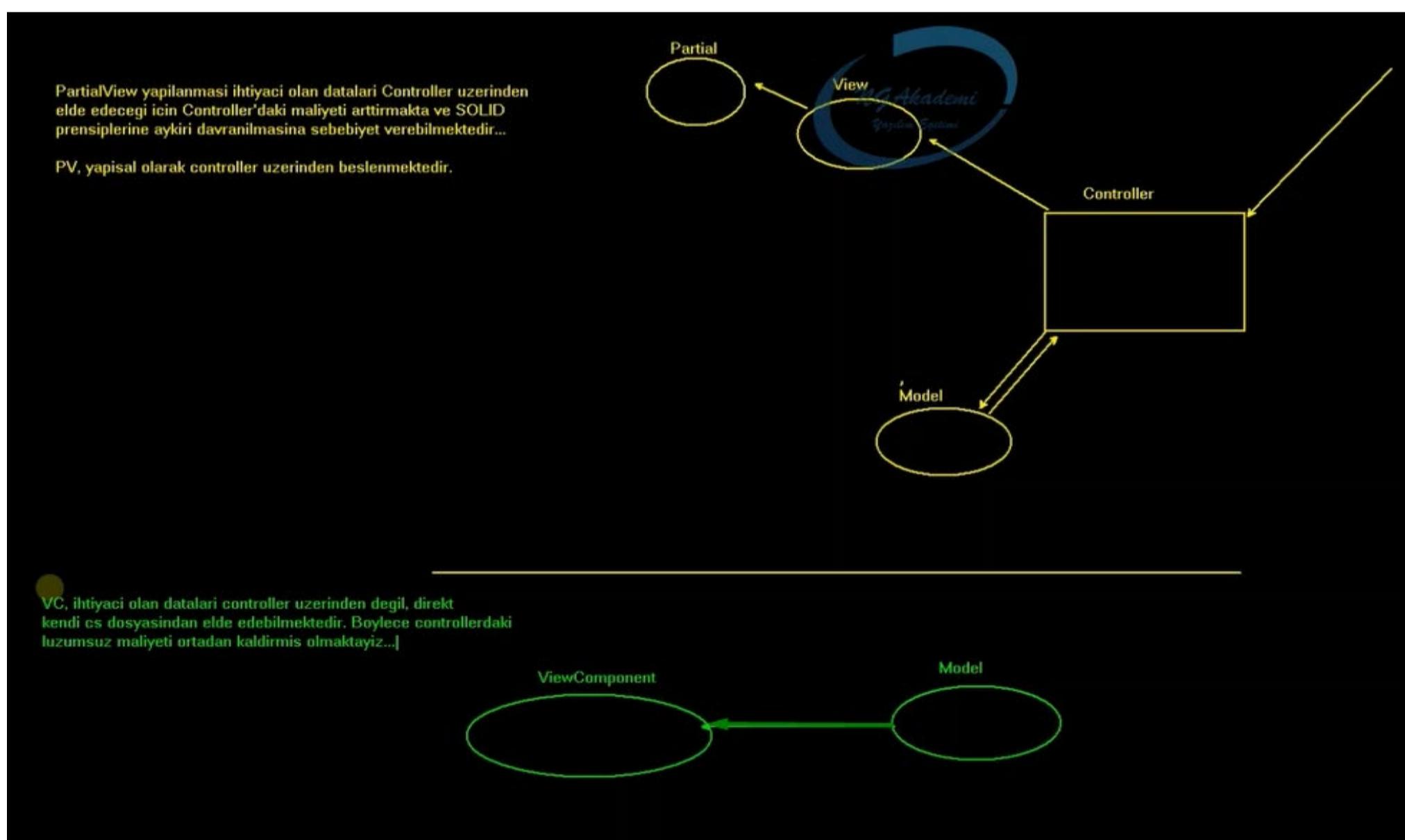
```

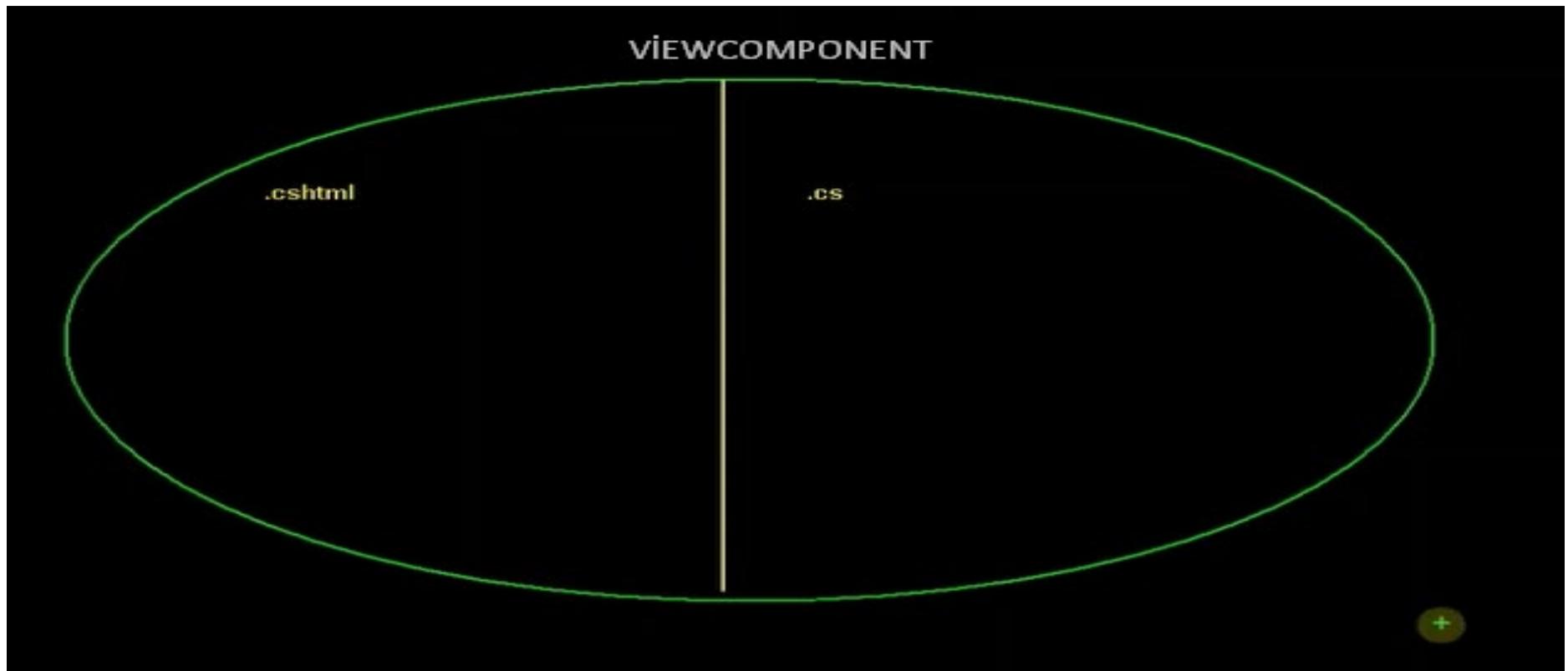
The screenshot shows a web application interface. At the top, there are browser navigation buttons (back, forward, refresh) and a search bar with placeholder text "Search for something...". Below the header, a sidebar on the left contains a user profile picture, the name "Turgay Baldan", the title "Yazar", and two menu items: "Dashboards" and "Grafikler". The main content area on the right displays a larger profile picture, the name "Turgay Baldan", the title "Yazar", and a short biography: "17/04/1999 Kocaeli Kandıra doğumluyum. 4 çocuklu bir evde büyümü.".

PartialView içinde RenderSection işlemi varsa bu Section işlemi çalışmaz.

ViewComponent Nedir? Nasıl Oluşturulur? Nasıl Kullanılır?

Get işlemleri yapılabilmekde ama post işlemleri yapılamamaktadır eğer bir post işlemi yapılacaksa ilgili get request'nin bulunduğu Componenet'i ilgili controller'a yönlendirilmesi gereklidir.





Solution Explorer

Search Solution Explorer (Ctrl+G)

Coreproje

```

1 using System.Linq;
2 using System.Threading.Tasks;
3
4 namespace Coreproje.ViewComponents.AdminNavbarPartial
5 {
6     public class AdminNavbarPart : ViewComponent
7     {
8         UserManager um = new UserManager(new EfUserRepository());
9         Context c = new Context();
10
11         public IViewComponentResult Invoke()
12         {
13             var username = User.Identity.Name;
14             ViewBag.name = username;
15
16             var usermail = c.Users.Where(x => x.UserName == username).Select(y => y.Email).FirstOrDefault();
17             var userID = c.Users.Where(x => x.Email == usermail).Select(y => y.Id).FirstOrDefault();
18
19             var values = um.TGetByID(userID);
20             return View(values);
21         }
22     }
23 }
```

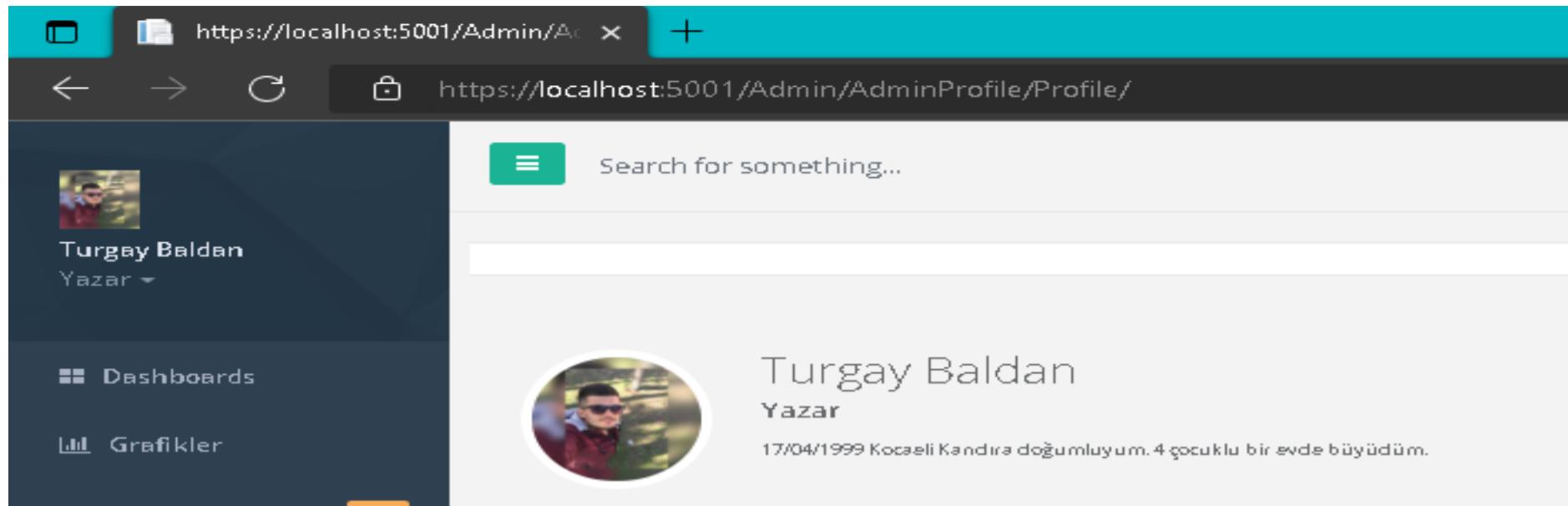
Tasarlanan viewComponent çağrılmış render edildiğinde içerisinde çalışmasını istediğimiz kodları bu imzada bir metodun içerisinde yerleştirmeliyiz.

Connected Services

- Dependencies
- Properties
- wwwroot
- Areas
- Controllers
- Models
- ViewComponents
 - AdminNavbarPartial
 - AdminNavbarPart.cs
 - Blog
 - Category
 - UserNavbar
 - Writer
 - WriterNavbarPart
 - WriterUpNavbar
 - CategoryList.cs
 - CommentListByBlog.cs
 - CommentsList.cs
- Views
- About
- Admin

```
1 @model EntityLayer.Concrete.AppUser
2
3
4 <nav class="navbar-default navbar-static-side" role="navigation" role="navigation">
5     <div class="sidebar-collapse">
6         <ul class="nav metismenu" id="side-menu">
7             <li class="nav-header">
8                 <div class="dropdown profile-element">
9                     
10                    <span class="login-status online"></span>
11
12                     <a data-toggle="dropdown" class="dropdown-toggle" href="/Admin/AdminProfile/Profile/">
13                         <span class="block m-t-xs font-bold">@Model.NameSurname </span>
14                         <span class="text-muted text-xs block">@Model.UserTitle <b>@Model.Caret</b></span>
15                     </a>
16                     <ul class="dropdown-menu animated fadeInRight m-t-xs">
17                         <li><a class="dropdown-item" href="/Admin/AdminProfile/Profile/">Profile</a></li>
18                         <li><a class="dropdown-item" href="/Admin/AdminContact/ContactList/">Contacts</a></li>
19                         <li><a class="dropdown-item" href="/Admin/AdminMessage/InBox/">Mailbox</a></li>
20                         <li class="dropdown-divider"></li>
21                         <li><a class="dropdown-item" href="/Login/LogOut/">Logout</a></li>
22                     </ul>
23                 </div>
24             <div class="logo-element">
25                 IN+
26             </div>
27         </li>
28     </ul>
29 </div>
```

```
4 }
5
6 <!DOCTYPE html>
7 <html>
8     @*Html.Partial("~/Views/Shared/_headerPartial.cshtml")*@
9     @*HtmlHelper ile partial tanımlaması*@
10    <partial name="~/Views/Shared/_headerPartial.cshtml" /> @*TagHelper ile partial tanımlama*@
11
12 <body>
13
14     <div id="wrapper">
15         @await Component.InvokeAsync("AdminNavbarPart")
16
17
18     <div id="page-wrapper" class="gray-bg">
19         <div class="row border-bottom">
20             <nav class="navbar navbar-static-top" role="navigation" style="margin-bottom: 0">
21                 <div class="navbar-header">
22                     <a class="navbar-minimize minimize-styl-2 btn btn-primary" href="#"><i class="fa fa-bars"></i></a>
23                     <form role="search" class="navbar-form-custom" action="search_results.html">
24                         <div class="form-group">
25                             <input type="text" placeholder="Search for something..." class="form-control">
26                         </div>
27                     </form>
28                 </div>
29                 <ul class="nav navbar-top-links navbar-right">
30                     <li class="dropdown">
31                         <a href="#" class="dropdown-toggle" data-toggle="dropdown"><i class="fa fa-envelope-o"></i> Notifications <span class="label label-warning">28</span></a>
32                         <ul class="dropdown-menu dropdown-menu-right dropdown-menu-inverse">
33                             <li><a href="#">View All</a></li>
34                             <li><a href="#">Mark as Read</a></li>
35                             <li><a href="#">Delete</a></li>
36                         </ul>
37                     </li>
38                 </ul>
39             </nav>
40         </div>
41     </div>
42
43     <div id="page-content-wrapper">
44         <div class="row">
45             <div class="col-sm-12">
46                 <div class="card card-block">
47                     <h3>Dashboard</h3>
48                     <p>Welcome to the Admin Dashboard!</p>
49                     <ul class="list-group">
50                         <li>Total Posts: 123</li>
51                         <li>Total Comments: 456</li>
52                         <li>Total Readers: 789</li>
53                     </ul>
54                 </div>
55             </div>
56         </div>
57     </div>
58
59     <div id="page-content">
60         <div class="row">
61             <div class="col-sm-12">
62                 <div class="card card-block">
63                     <h3>Recent Posts</h3>
64                     <ul class="list-group">
65                         <li>Post 1: Title 1</li>
66                         <li>Post 2: Title 2</li>
67                         <li>Post 3: Title 3</li>
68                     </ul>
69                 </div>
70             </div>
71         </div>
72     </div>
73
74     <div id="page-footer">
75         <div class="row">
76             <div class="col-sm-12">
77                 <div class="card card-block">
78                     <h3>Footer Content</h3>
79                     <p>Copyright © 2023 Your Company. All Rights Reserved.</p>
80                 </div>
81             </div>
82         </div>
83     </div>
84
85     <div id="page-copyright">
86         <div class="row">
87             <div class="col-sm-12">
88                 <div class="card card-block">
89                     <h3>Page Copyright</h3>
90                     <p>Page copyright content here.</p>
91                 </div>
92             </div>
93         </div>
94     </div>
95
96     <div id="page-back-to-top">
97         <div class="row">
98             <div class="col-sm-12">
99                 <div class="card card-block">
100                    <h3>Back to Top</h3>
101                    <p>Back to top button content here.</p>
102                </div>
103            </div>
104        </div>
105    </div>
106
107    <div id="page-modal">
108        <div class="row">
109            <div class="col-sm-12">
110                <div class="card card-block">
111                    <h3>Modal Content</h3>
112                    <p>Modal content here.</p>
113                </div>
114            </div>
115        </div>
116    </div>
117
118    <div id="page-modal2">
119        <div class="row">
120            <div class="col-sm-12">
121                <div class="card card-block">
122                    <h3>Modal Content 2</h3>
123                    <p>Modal content 2 here.</p>
124                </div>
125            </div>
126        </div>
127    </div>
128
129    <div id="page-modal3">
130        <div class="row">
131            <div class="col-sm-12">
132                <div class="card card-block">
133                    <h3>Modal Content 3</h3>
134                    <p>Modal content 3 here.</p>
135                </div>
136            </div>
137        </div>
138    </div>
139
140    <div id="page-modal4">
141        <div class="row">
142            <div class="col-sm-12">
143                <div class="card card-block">
144                    <h3>Modal Content 4</h3>
145                    <p>Modal content 4 here.</p>
146                </div>
147            </div>
148        </div>
149    </div>
150
151    <div id="page-modal5">
152        <div class="row">
153            <div class="col-sm-12">
154                <div class="card card-block">
155                    <h3>Modal Content 5</h3>
156                    <p>Modal content 5 here.</p>
157                </div>
158            </div>
159        </div>
160    </div>
161
162    <div id="page-modal6">
163        <div class="row">
164            <div class="col-sm-12">
165                <div class="card card-block">
166                    <h3>Modal Content 6</h3>
167                    <p>Modal content 6 here.</p>
168                </div>
169            </div>
170        </div>
171    </div>
172
173    <div id="page-modal7">
174        <div class="row">
175            <div class="col-sm-12">
176                <div class="card card-block">
177                    <h3>Modal Content 7</h3>
178                    <p>Modal content 7 here.</p>
179                </div>
180            </div>
181        </div>
182    </div>
183
184    <div id="page-modal8">
185        <div class="row">
186            <div class="col-sm-12">
187                <div class="card card-block">
188                    <h3>Modal Content 8</h3>
189                    <p>Modal content 8 here.</p>
190                </div>
191            </div>
192        </div>
193    </div>
194
195    <div id="page-modal9">
196        <div class="row">
197            <div class="col-sm-12">
198                <div class="card card-block">
199                    <h3>Modal Content 9</h3>
200                    <p>Modal content 9 here.</p>
201                </div>
202            </div>
203        </div>
204    </div>
205
206    <div id="page-modal10">
207        <div class="row">
208            <div class="col-sm-12">
209                <div class="card card-block">
210                    <h3>Modal Content 10</h3>
211                    <p>Modal content 10 here.</p>
212                </div>
213            </div>
214        </div>
215    </div>
216
217    <div id="page-modal11">
218        <div class="row">
219            <div class="col-sm-12">
220                <div class="card card-block">
221                    <h3>Modal Content 11</h3>
222                    <p>Modal content 11 here.</p>
223                </div>
224            </div>
225        </div>
226    </div>
227
228    <div id="page-modal12">
229        <div class="row">
230            <div class="col-sm-12">
231                <div class="card card-block">
232                    <h3>Modal Content 12</h3>
233                    <p>Modal content 12 here.</p>
234                </div>
235            </div>
236        </div>
237    </div>
238
239    <div id="page-modal13">
240        <div class="row">
241            <div class="col-sm-12">
242                <div class="card card-block">
243                    <h3>Modal Content 13</h3>
244                    <p>Modal content 13 here.</p>
245                </div>
246            </div>
247        </div>
248    </div>
249
250    <div id="page-modal14">
251        <div class="row">
252            <div class="col-sm-12">
253                <div class="card card-block">
254                    <h3>Modal Content 14</h3>
255                    <p>Modal content 14 here.</p>
256                </div>
257            </div>
258        </div>
259    </div>
260
261    <div id="page-modal15">
262        <div class="row">
263            <div class="col-sm-12">
264                <div class="card card-block">
265                    <h3>Modal Content 15</h3>
266                    <p>Modal content 15 here.</p>
267                </div>
268            </div>
269        </div>
270    </div>
271
272    <div id="page-modal16">
273        <div class="row">
274            <div class="col-sm-12">
275                <div class="card card-block">
276                    <h3>Modal Content 16</h3>
277                    <p>Modal content 16 here.</p>
278                </div>
279            </div>
280        </div>
281    </div>
282
283    <div id="page-modal17">
284        <div class="row">
285            <div class="col-sm-12">
286                <div class="card card-block">
287                    <h3>Modal Content 17</h3>
288                    <p>Modal content 17 here.</p>
289                </div>
290            </div>
291        </div>
292    </div>
293
294    <div id="page-modal18">
295        <div class="row">
296            <div class="col-sm-12">
297                <div class="card card-block">
298                    <h3>Modal Content 18</h3>
299                    <p>Modal content 18 here.</p>
300                </div>
301            </div>
302        </div>
303    </div>
304
305    <div id="page-modal19">
306        <div class="row">
307            <div class="col-sm-12">
308                <div class="card card-block">
309                    <h3>Modal Content 19</h3>
310                    <p>Modal content 19 here.</p>
311                </div>
312            </div>
313        </div>
314    </div>
315
316    <div id="page-modal20">
317        <div class="row">
318            <div class="col-sm-12">
319                <div class="card card-block">
320                    <h3>Modal Content 20</h3>
321                    <p>Modal content 20 here.</p>
322                </div>
323            </div>
324        </div>
325    </div>
326
327    <div id="page-modal21">
328        <div class="row">
329            <div class="col-sm-12">
330                <div class="card card-block">
331                    <h3>Modal Content 21</h3>
332                    <p>Modal content 21 here.</p>
333                </div>
334            </div>
335        </div>
336    </div>
337
338    <div id="page-modal22">
339        <div class="row">
340            <div class="col-sm-12">
341                <div class="card card-block">
342                    <h3>Modal Content 22</h3>
343                    <p>Modal content 22 here.</p>
344                </div>
345            </div>
346        </div>
347    </div>
348
349    <div id="page-modal23">
350        <div class="row">
351            <div class="col-sm-12">
352                <div class="card card-block">
353                    <h3>Modal Content 23</h3>
354                    <p>Modal content 23 here.</p>
355                </div>
356            </div>
357        </div>
358    </div>
359
360    <div id="page-modal24">
361        <div class="row">
362            <div class="col-sm-12">
363                <div class="card card-block">
364                    <h3>Modal Content 24</h3>
365                    <p>Modal content 24 here.</p>
366                </div>
367            </div>
368        </div>
369    </div>
370
371    <div id="page-modal25">
372        <div class="row">
373            <div class="col-sm-12">
374                <div class="card card-block">
375                    <h3>Modal Content 25</h3>
376                    <p>Modal content 25 here.</p>
377                </div>
378            </div>
379        </div>
380    </div>
381
382    <div id="page-modal26">
383        <div class="row">
384            <div class="col-sm-12">
385                <div class="card card-block">
386                    <h3>Modal Content 26</h3>
387                    <p>Modal content 26 here.</p>
388                </div>
389            </div>
390        </div>
391    </div>
392
393    <div id="page-modal27">
394        <div class="row">
395            <div class="col-sm-12">
396                <div class="card card-block">
397                    <h3>Modal Content 27</h3>
398                    <p>Modal content 27 here.</p>
399                </div>
400            </div>
401        </div>
402    </div>
403
404    <div id="page-modal28">
405        <div class="row">
406            <div class="col-sm-12">
407                <div class="card card-block">
408                    <h3>Modal Content 28</h3>
409                    <p>Modal content 28 here.</p>
410                </div>
411            </div>
412        </div>
413    </div>
414
415    <div id="page-modal29">
416        <div class="row">
417            <div class="col-sm-12">
418                <div class="card card-block">
419                    <h3>Modal Content 29</h3>
420                    <p>Modal content 29 here.</p>
421                </div>
422            </div>
423        </div>
424    </div>
425
426    <div id="page-modal30">
427        <div class="row">
428            <div class="col-sm-12">
429                <div class="card card-block">
430                    <h3>Modal Content 30</h3>
431                    <p>Modal content 30 here.</p>
432                </div>
433            </div>
434        </div>
435    </div>
436
437    <div id="page-modal31">
438        <div class="row">
439            <div class="col-sm-12">
440                <div class="card card-block">
441                    <h3>Modal Content 31</h3>
442                    <p>Modal content 31 here.</p>
443                </div>
444            </div>
445        </div>
446    </div>
447
448    <div id="page-modal32">
449        <div class="row">
450            <div class="col-sm-12">
451                <div class="card card-block">
452                    <h3>Modal Content 32</h3>
453                    <p>Modal content 32 here.</p>
454                </div>
455            </div>
456        </div>
457    </div>
458
459    <div id="page-modal33">
460        <div class="row">
461            <div class="col-sm-12">
462                <div class="card card-block">
463                    <h3>Modal Content 33</h3>
464                    <p>Modal content 33 here.</p>
465                </div>
466            </div>
467        </div>
468    </div>
469
470    <div id="page-modal34">
471        <div class="row">
472            <div class="col-sm-12">
473                <div class="card card-block">
474                    <h3>Modal Content 34</h3>
475                    <p>Modal content 34 here.</p>
476                </div>
477            </div>
478        </div>
479    </div>
480
481    <div id="page-modal35">
482        <div class="row">
483            <div class="col-sm-12">
484                <div class="card card-block">
485                    <h3>Modal Content 35</h3>
486                    <p>Modal content 35 here.</p>
487                </div>
488            </div>
489        </div>
490    </div>
491
492    <div id="page-modal36">
493        <div class="row">
494            <div class="col-sm-12">
495                <div class="card card-block">
496                    <h3>Modal Content 36</h3>
497                    <p>Modal content 36 here.</p>
498                </div>
499            </div>
500        </div>
501    </div>
502
503    <div id="page-modal37">
504        <div class="row">
505            <div class="col-sm-12">
506                <div class="card card-block">
507                    <h3>Modal Content 37</h3>
508                    <p>Modal content 37 here.</p>
509                </div>
510            </div>
511        </div>
512    </div>
513
514    <div id="page-modal38">
515        <div class="row">
516            <div class="col-sm-12">
517                <div class="card card-block"&gt
```



ROUTE YAPILANMASI

Route: Gelecek olan isteğin hangi rotaya gideceğini belirleyen şablonlardır.

Derinlemesine Route Yapılanması :

MapDefaultControllerRoute();

```
40     app.UseHsts();
41
42     app.UseHttpsRedirection();
43     app.UseStaticFiles();
44
45     app.UseRouting();
46
47     app.UseAuthorization();
48
49     app.UseEndpoints(endpoints => //Gelen rotaların tanımlandığı yer
50     {
51         //Otomatik ön tanımlı rota getiriyor. Controller veya action boş gelirse kendisinin
52         endpoints.MapDefaultControllerRoute(); // tayin ettiği rota izlenilcek
53     });
54
55 }
56
57 }
58 }
```

Tooltip for `endpoints.MapDefaultControllerRoute()`:

(extension) `Microsoft.AspNetCore.Routing.IEndpointRouteBuilder.MapDefaultControllerRoute()`
Adds endpoints for controller actions to the `Microsoft.AspNetCore.Routing.IEndpointRouteBuilder` and adds the default route
`{controller=Home}/{action=Index}/{id?}`
Returns:
An `ControllerActionEndpointConventionBuilder` for endpoints associated with controller actions for this route.

MapControllerRoute(); Rotayı özelleştirdiğimiz fonksiyon yani CustomRoute

The screenshot shows the Visual Studio IDE with the 'Startup.cs' file open. The code is as follows:

```
40     app.UseHsts();
41
42     app.UseHttpsRedirection();
43     app.UseStaticFiles();
44
45     app.UseRouting();
46
47     app.UseAuthorization();
48
49     app.UseEndpoints(endpoints => //Gelen rotaların tanımlandığı yer
50     {
51         endpoints.MapControllerRoute();
52     });
53 }
54 }
55 }
56 }
57 }
58 }
```

A tooltip is displayed over the line 'endpoints.MapControllerRoute();'. The tooltip contains the following information:

- Method: Microsoft.AspNetCore.Routing.IEndpointRouteBuilder.MapControllerRoute(string name, string pattern, [object defaults = null], [object constraints = null], [object dataTokens = null])
- Description: Adds endpoints for controller actions to the Microsoft.AspNetCore.Routing.IEndpointRouteBuilder and specifies a route with the given name, pattern, defaults, constraints, and dataTokens.
- Returns: An ControllerActionEndpointConventionBuilder for endpoints associated with controller actions for this route.
- Error message: CS7036: There is no argument given that corresponds to the required formal parameter 'name' of 'ControllerEndpointRouteBuilderExtensions.MapControllerRoute(IEndpointRouteBuilder, string, string, object, object, object)'.
- Action: Show potential fixes (Alt+Enter or Ctrl+.)

The screenshot shows the Visual Studio IDE with the 'Startup.cs' file open. The code is as follows:

```
40     app.UseHsts();
41
42     app.UseHttpsRedirection();
43     app.UseStaticFiles();
44
45     app.UseRouting();
46
47     app.UseAuthorization();
48
49     app.UseEndpoints(endpoints => //Gelen rotaların tanımlandığı yer
50     {
51         //birden fazla rota tanımlanacaksa özelden genele doğru sıralamalıyız.
52
53         endpoints.MapControllerRoute("Custom", "{controller=Home}/{action=Index}");
54         endpoints.MapControllerRoute("Custom2", "anasayfa", new { controller = "Home", action =
55             "Index" });
56     });
57 }
58 }
59 }
```

Annotations are present in the code:

- A yellow lightbulb icon is shown at the start of the block '//birden fazla rota tanımlanacaksa özelden genele doğru sıralamalıyız.'
- Red arrows point from the annotations below to specific parts of the code:
 - An arrow points from the text 'url \'çubüğuna anasayfa diye yazdığımızd bu rotaya yönlendirilsin' to the 'Custom2' route entry.
 - Another arrow points from the same text to the 'Index' action parameter in the 'Custom2' route entry.
 - A third arrow points from the text to the '{controller=Home}' placeholder in the 'Custom' route entry.
- The text 'url \'çubüğuna anasayfa diye yazdığımızd bu rotaya yönlendirilsin' is centered at the bottom of the annotations.

Startup.cs

```
40     app.UseHsts();
41 }
42 app.UseHttpsRedirection();
43 app.UseStaticFiles();
44
45 app.UseRouting();
46
47 app.UseAuthorization();
48
49 app.UseEndpoints(endpoints => //Gelen rotaların tanımlandığı yer
50 //birden fazla rota tanımlanacaksa özelden genele doğru sıralamalıyız.
51
52     endpoints.MapControllerRoute("Custom", "{controller=Home}/{action=Index}");
53     endpoints.MapControllerRoute("Custom2", "anasayfa", new { controller = "Home", action =
54         "Index" });
55     endpoints.MapControllerRoute("Custom", "{controller=Home}/{action=Index}/{id:int?}/
56     {x:length(11)?}/{y:int?}");
57 }
58 }
59 }
60 }
```

Solution Explorer

- RouteYapilanmasi
- Connected Services
- Dependencies
- Properties
- wwwroot
- Controllers
- Models
- Views
- Home
- Index.cshtml
- Privacy.cshtml
- Shared
- _ViewImports.cshtml
- _ViewStart.cshtml

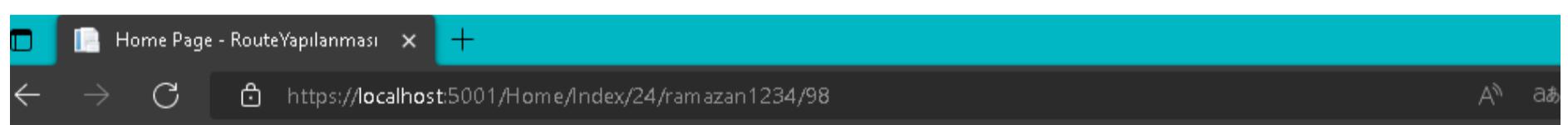
appsettings.json

Program.cs

class System.String
Represents text as a sequence of UTF-16 code units.

HomeController.cs

```
12     public class HomeController : Controller
13     {
14         private readonly ILogger<HomeController> _logger;
15
16         public HomeController(ILogger<HomeController> logger)
17         {
18             _logger = logger;
19         }
20
21         public IActionResult Index(int id, string x, int y)
22         {
23             return View();
24         }
25     }
```



RouteYapilanmasi Home Privacy

Welcome

Learn about [building Web apps with ASP.NET Core](#).

Custom Constraint Oluşturma

The screenshot shows the Visual Studio IDE. On the left is the code editor with the file `CustomConstraint.cs` open. The code defines a class `CustomConstraint` that implements the `IRouteConstraint` interface. The `Match` method is implemented to always return `true`. The `CustomConstraint.cs` file is highlighted in the Solution Explorer on the right.

```
Startup.cs    CustomConstraints*  X
RouteYapilanmasi  RouteYapilanmasi.Constraints.CustomConstraint  Match(HttpContext httpContext, IRouter route, string routeKey, RouteValueDictionary values, RouteDirection routeDirection)
1  using Microsoft.AspNetCore.Http;
2  using Microsoft.AspNetCore.Routing;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Threading.Tasks;
7
8  namespace RouteYapilanmasi.Constraints
9  {
10     public class CustomConstraint : IRouteConstraint
11     {
12         public bool Match(HttpContext httpContext, IRouter route, string routeKey, RouteValueDictionary values, RouteDirection routeDirection)
13         {
14             return true;
15         }
16     }
17 }
18
```

The screenshot shows the `Startup.cs` file in the code editor. It contains the `ConfigureServices` and `Configure` methods. In the `ConfigureServices` method, a custom constraint is registered with the name "custom". In the `Configure` method, a controller route is mapped with the name "Custom" and the pattern "{controller=Home}/{action=Index}/{id:custom}/{x:length(11)?}/{y:int?}".

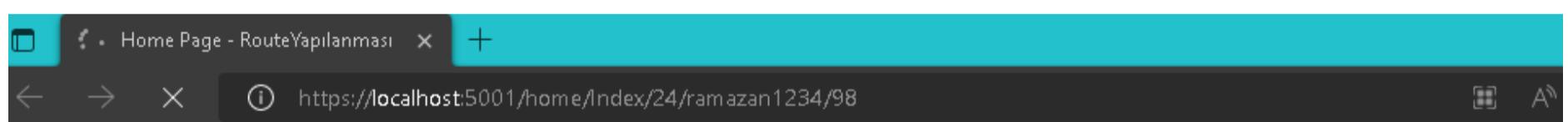
```
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllersWithViews();
    services.Configure<RouteOptions>(options=> options.ConstraintMap.Add("custom",typeof(CustomConstraint)));
}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Home/Error");
        // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/
        //aspnetcore-hsts.
        app.UseHsts();
    }
    app.UseHttpsRedirection();
    app.UseStaticFiles();

    app.UseRouting();

    app.UseAuthorization();

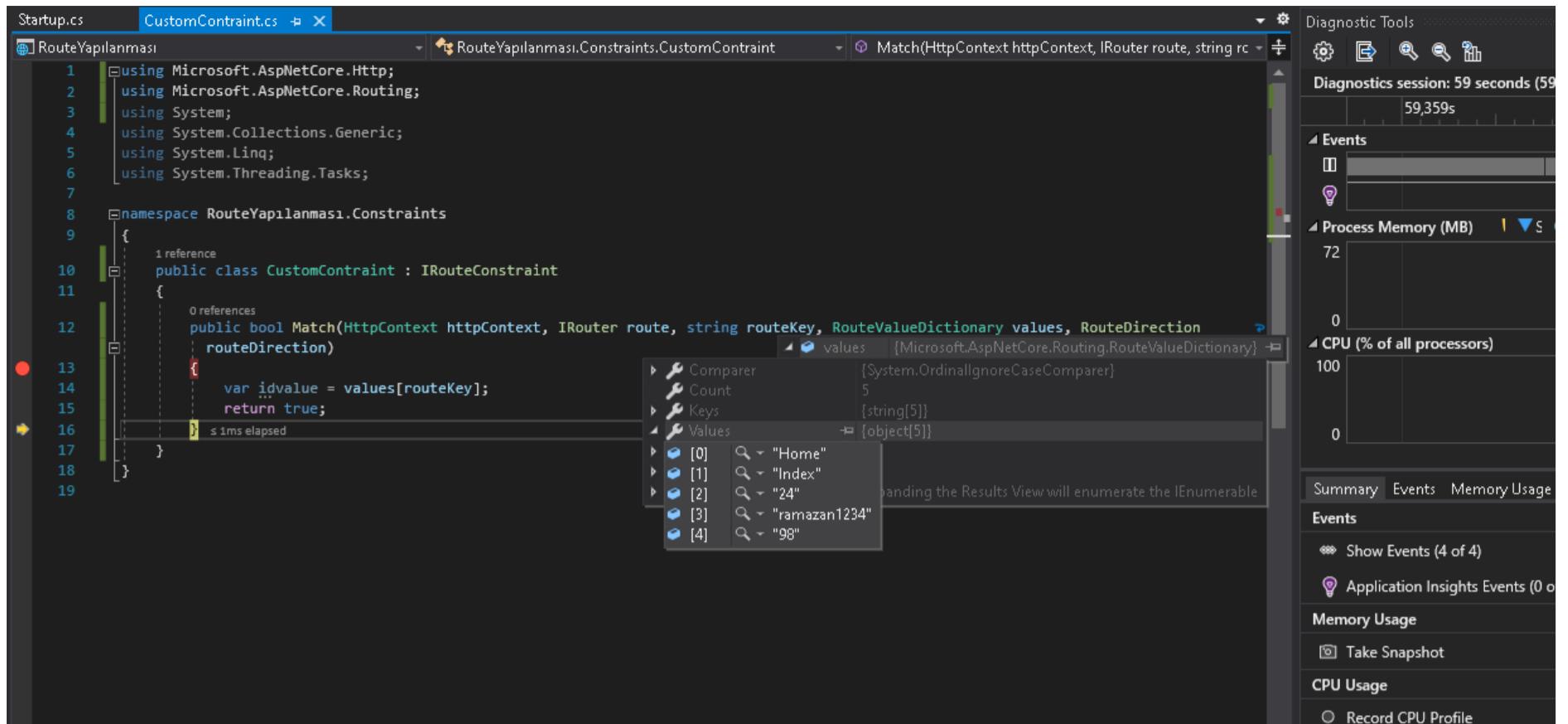
    app.UseEndpoints(endpoints => //Gelen rotaların tanımlandığı yer
    {//birden fazla rota tanımlanacaksa özelden genele doğru sıralamalıyız.
        endpoints.MapControllerRoute("Custom", "{controller=Home}/{action=Index}/{id:custom}/{x:length(11)?}/{y:int?}");
    });
}
```



RouteYapilanmasi Home Privacy

Welcome

Learn about [building Web apps with ASP.NET Core](#).



Attribute Routing

```

}
app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints => //Gelen rotaların tanımlandığı yer
{ //birden fazla rota tanımlanacaksa özelden genelle doğru sıralamalıyız.
    //endpoints.MapControllerRoute("Custom", "{controller=Home}/{action=Index}/{id:custom}/{x:length(11)?}/{y:int?}");

    //endpoints.MapControllerRoute("Custom", "{controller=Home}/{action=Index}");
    //endpoints.MapControllerRoute("Custom2","anasayfa", new { controller = "Home", action = "Index" });

    endpoints.MapControllers(); //Controllerlardan gelen isteği controllerlarla eşleştir.
});

```

The screenshot shows the Visual Studio code editor with the `HomeController.cs` file open. A tooltip is displayed over the `[Route("[controller]/[action]")]` attribute, providing information about the `Index()` method. The tooltip content includes:

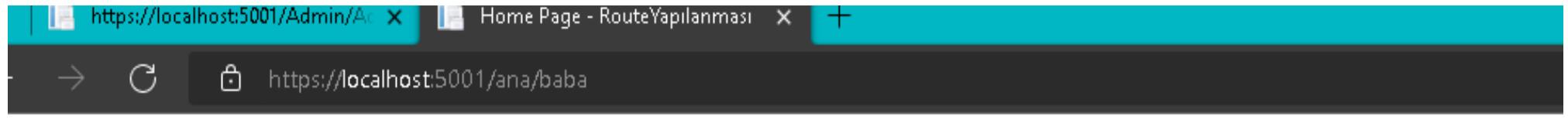
- Index()
- Use the dropdown to...

```
4  using System;
5  using System.Collections.Generic;
6  using System.Diagnostics;
7  using System.Linq;
8  using System.Threading.Tasks;
9
10 namespace RouteYapilanmasi.Controllers
11 {
12     [Route("[controller]/[action]")]
13     public class HomeController : Controller
14     {
15         private readonly ILogger<HomeController> _logger;
16
17         public HomeController(ILogger<HomeController> logger)
18         {
19             _logger = logger;
20         }
21
22         public IActionResult Index()
23         {
24             return View();
25         }
26
27         public IActionResult Privacy()
28         {
29             return View();
30         }
31 }
```

Özelleştirilmiş Attribute Routing Yapısı

The screenshot shows the Visual Studio code editor with the `HomeController.cs` file open. The `Index()` method now has two `[Route]` attributes: one for "ana" and one for "baba". The tooltip for the first `[Route("ana")]` attribute is visible.

```
4  using System;
5  using System.Collections.Generic;
6  using System.Diagnostics;
7  using System.Linq;
8  using System.Threading.Tasks;
9
10 namespace RouteYapilanmasi.Controllers
11 {
12     [Route("ana")]
13     public class HomeController : Controller
14     {
15         private readonly ILogger<HomeController> _logger;
16
17         public HomeController(ILogger<HomeController> logger)
18         {
19             _logger = logger;
20         }
21
22         [Route("baba")]
23         public IActionResult Index()
24         {
25             return View();
26         }
27 }
```



RouteYapılanması Home Privacy

Welcome

Learn about [building Web apps with ASP.NET Core.](#)

Custom Route Handler Nedir? Nasıl İnşa Edilir?

Controller'dan bağımsız Business mantığında direkt isteği karşılayıp direkt operasyon gerçekleştirmek istiyorsak bu operasyonu bu yapı ile yapabiliriz. Yani herhangi bir belirlenmiş route şemasının controller sınıflarından ziyade business mantığında karşılaşması ve orada iş görüp response dönülmesi operasyonudur.

Genel geçer operasyonlarda klasik Controller mekanizmasıyla gelen request'i karşılayıp gerekli operasonu gerçekleştiriceksiniz.

Örnek; bir Resim uygulaması...

Nasıl İnşa Edilir?

```
ExampleHandler.cs*    Startup.cs*  HomeController.cs
CustomRouteHandler
else
{
    app.UseExceptionHandler("/Home/Error");
    // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}
app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>
{
    /* endpoints.Map("example-route", async c =>*/ //ilgili fonksiyonun asenkron olabilceğini ifade ediyor (async)
    //{
        // https://localhost:5001/example-route endpoint'e gelen herhangi bir istek Controller'dan ziyade direkt olarak buradaki fonksiyon tarafından karşılaşacaktır.
    //}
    //);
    endpoints.Map("example-route", new ExampleHandler().Handler());
}

endpoints.Map("example-route", new ExampleHandler().Handler());
}
}
});
```

(extension) IEndpointConventionBuilder Microsoft.AspNetCore.Routing.IEndpointRouteBuilder.Map(string pattern, Microsoft.AspNetCore.Http.RequestDelegate<T> requestDelegate) (+ 1 overload)
Adds a Microsoft.AspNetCore.Routing.RouteEndpoint to the Microsoft.AspNetCore.Routing.IEndpointRouteBuilder that matches HTTP requests for the specified pattern.
Returns:
A IEndpointConventionBuilder that can be used to further customize the endpoint.

Solution Explorer

- Solution 'CustomRouteHandler' (1 of 1 project)
 - CustomRouteHandler
 - Connected Services
 - Dependencies
 - Properties
 - wwwroot
 - Controllers
 - HomeController.cs
 - Handlers
 - ExampleHandler.cs
 - ExampleHandler
 - Models
 - Views
 - appsettings.json
 - Program.cs
 - Startup.cs

The screenshot shows a Visual Studio IDE environment. In the top-left, there are three tabs: 'ExampleHandler.cs', 'Startup.cs', and 'HomeController.cs'. The 'ExampleHandler.cs' tab is active, displaying the following C# code:

```
1  using Microsoft.AspNetCore.Http;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Threading.Tasks;
6
7  namespace CustomRouteHandler.Handlers
8  {
9      public class ExampleHandler
10     : IRouteHandler // This interface is highlighted with a red box
11     {
12         public RequestDelegate Handler() // This method is highlighted with a red box
13         {
14             return async c =>
15             {
16                 await c.Response.WriteAsync("Merhaba Burada Custom Route Handler oluşturduk");
17             };
18         }
19     }
20 }
```

The 'Solution Explorer' window on the right shows a project named 'CustomRouteHandler' with files like 'Startup.cs', 'HomeController.cs', and 'ExampleHandler.cs' under the 'Handlers' folder.

In the bottom half of the image, a browser window is open at <https://localhost:5001/example-route>. The page content is "Merhaba Burada Custom Route Handler oluşturduk".

Custom Route ile Resim Boyutlandırma

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

- File Menu:** File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help.
- Search Bar:** Search (Ctrl+Q).
- Solution Explorer:** Shows the project structure for 'CustomRouteHandler' with files like 'Connected Services', 'Dependencies', 'Properties', 'wwwroot' (containing 'css', 'js', 'lib' with 'favicon.ico', 'mailarkaplan.jpg', 'mail-background-12578668.jpg', 'mailll.png', 'mvc sertifikajpg', 'otomasyon.jpg', 'salesforce-e-commerce.jpg', 'satis.jpg'), 'Controllers' (containing 'HomeController.cs'), 'Handlers' (containing 'ExampleHandler.cs', 'ImageHandler.cs'), 'Models', 'Views', 'appsettings.json', 'Program.cs', and 'Startup.cs'.
- Code Editor:** The 'ImageHandler.cs' file is open, showing the following code:

```
9  namespace CustomRouteHandler.Handlers
10 {
11     public class ImageHandler
12     {
13         public RequestDelegate Handler(string filePath)
14         {
15             return async c =>
16             {
17                 FileInfo fileInfo = new FileInfo($"{filePath}\\{c.Request.RouteValues["imageName"].ToString()}");
18                 using MagickImage magick = new MagickImage(fileInfo);
19                 int width = magick.Width, height = magick.Height;
20                 if (!string.IsNullOrEmpty(c.Request.Query["w"].ToString()))
21                     width = int.Parse(c.Request.Query["w"].ToString());
22                 if (!string.IsNullOrEmpty(c.Request.Query["h"].ToString()))
23                     height = int.Parse(c.Request.Query["h"].ToString());
24                 magick.Resize(width, height);
25                 var buffer = magick.ToByteArray();
26                 c.Response.Clear();
27                 c.Response.ContentType = string.Concat("image/", fileInfo.Extension.Replace(".", ""));
28                 await c.Response.Body.WriteAsync(buffer, 0, buffer.Length);
29                 await c.Response.WriteAsync(filePath);
30             };
31         }
32     }
33 }
34 }
35 }
36 }
37 }
```

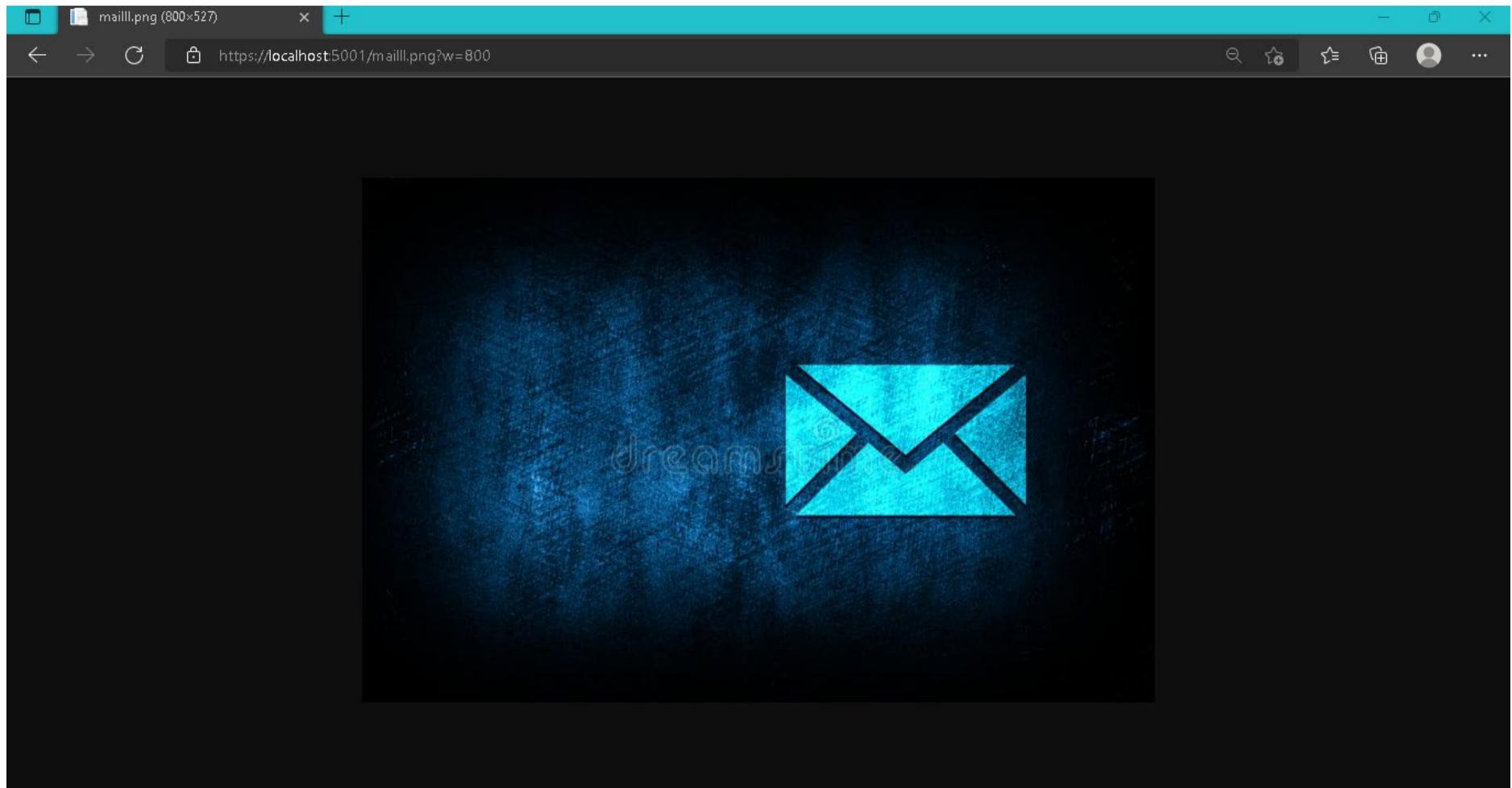
Status Bar: 95%, No issues found, In: 28 Ch: 92 SPC CRLF.

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

- File Menu:** File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help.
- Search Bar:** Search (Ctrl+Q).
- Solution Explorer:** Shows the project structure for 'CustomRouteHandler' with files like 'Connected Services', 'Dependencies', 'Properties', 'wwwroot' (containing 'css', 'js', 'lib' with 'favicon.ico', 'mailarkaplan.jpg', 'mail-background-12578668.jpg', 'mailll.png', 'mvc sertifikajpg', 'otomasyon.jpg', 'salesforce-e-commerce.jpg', 'satis.jpg'), 'Controllers' (containing 'HomeController.cs'), 'Handlers' (containing 'ExampleHandler.cs', 'ImageHandler.cs'), 'Models', 'Views', 'appsettings.json', 'Program.cs', and 'Startup.cs'.
- Code Editor:** The 'Startup.cs' file is open, showing the following code:

```
40     // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/
41     //aspnetcore-hsts.
42     app.UseHsts();
43 }
44 app.UseEndpoint // (extension) IApplicationBuilder IApplicationBuilder.UseHsts()
45     app.UseEndpoint Adds middleware for using HSTS, which adds the Strict-Transport-Security header.
46
47     app.UseRouting();
48
49     app.UseAuthorization();
50
51     app.UseEndpoints(endpoints =>
52     {
53         endpoints.Map("image/{imageName}", new ImageHandler().Handler(env.WebRootPath));
54         /* endpoints.Map("example-route", async c =>*/ //ilgili fonksiyonun asenkron olabileceğini ifade ediyor (async)
55         //{
56         //    // https://localhost:5001/example-route endpoint'e gelen herhangi
57         // bir istek Controller'dan ziyade direkt olarak buradaki fonksiyon tarafından karşılanacaktır.
58     });
59 }
```

A tooltip is displayed over the line `app.UseEndpoint` with the following text:
Adds middleware for using HSTS, which adds the Strict-Transport-Security header.



Asp.NET Core Middleware

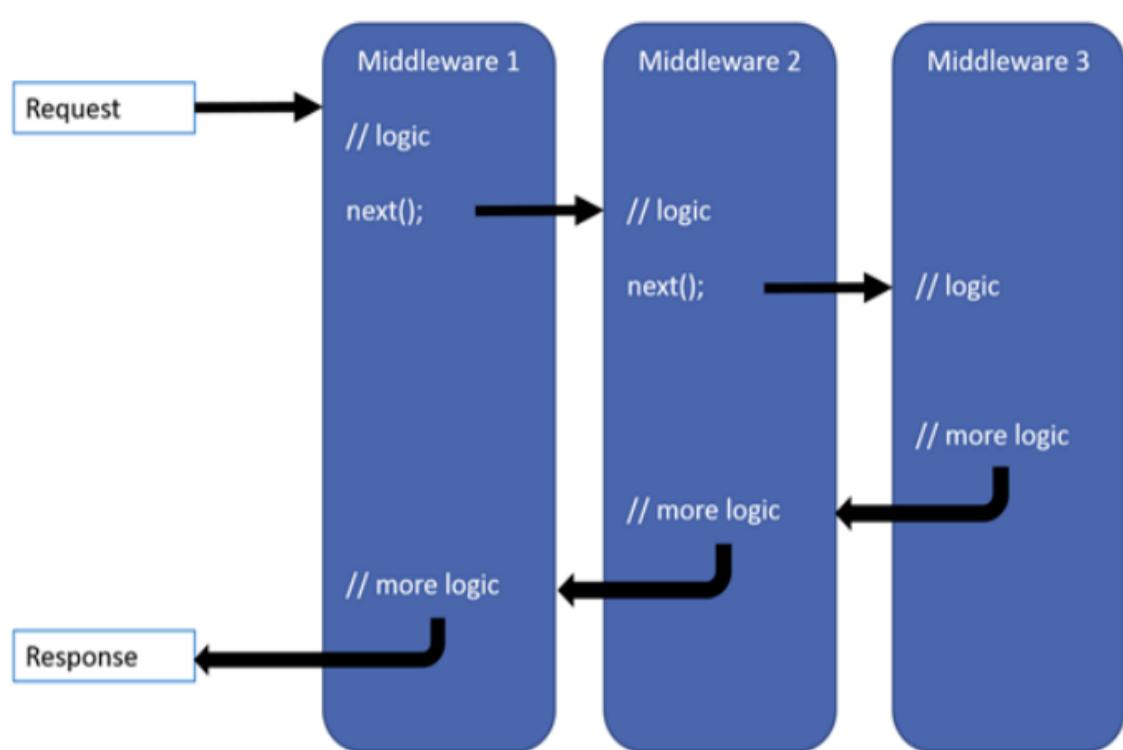
Asp.NET Core Middleware Nedir?

- Web uygulamasına client'tan gelen request'e karşılık verilecek response'a kadar arada farklı işlemler gerçekleştirmek ve sürecin gidişatına farklı yön vermek isteyebiliriz.



Middleware'ler Nasıl Çalışır?

- Middleware'ler sarmal bir şekilde tetiklenirler.



Asp.NET Core Uygulamasında Middleware

Asp.NET Core, yapısal olarak middleware yapılanmasını destekleyen bir çekirdeğe sahiptir.

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
        app.UseSwagger();
        app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "MiddlewareExample v1"));
    }

    app.UseHttpsRedirection();

    app.UseRouting();

    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllers();
    });
}
```

Hazır Middleware'ler:

- Run
- Use
- Map
- MapWhen

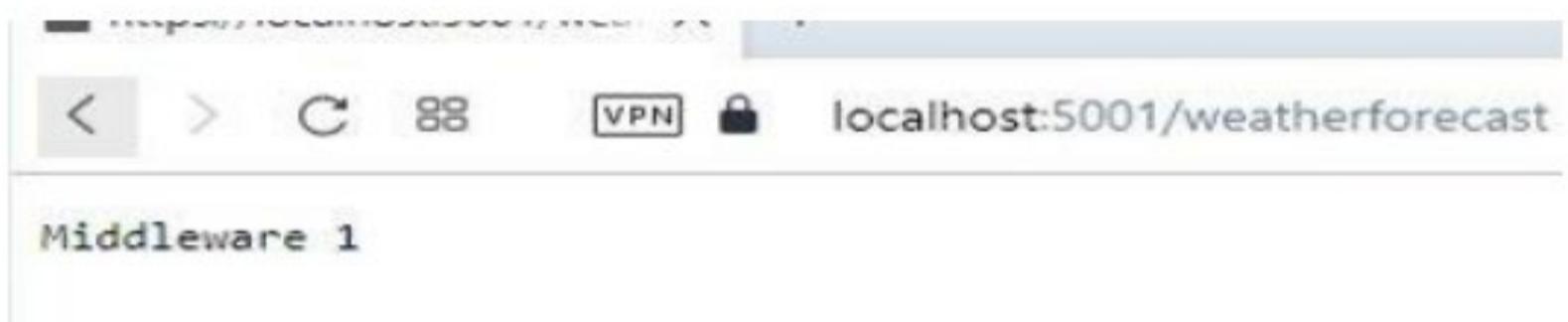
Run Metodu: Run fonksiyonu kendisinden sonra gelen middleware'i tetiklemez!

- Dolayısıyla kullanıldığı yerden sonraki middleware'ler tetiklenmeyeceğinden dolayı akış kesilecektir.
- Bu etkiye Short Circuit(Kısa Devre) denir.

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
        app.UseSwagger();
        app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "Middleware"));
    }

    app.Run(async c => { await c.Response.WriteAsync("Middleware 1"); });
    app.Run(async c => { await c.Response.WriteAsync("Middleware 2"); });

    app.UseHttpsRedirection();
    app.UseRouting();
}
```



Use Metodu

- Run metoduna nazaran, devreye girdikten sonra süreçte sıradaki middleware'i çağrılmakta ve normal middleware işlevi bittikten sonra geriye dönüp devam edebilen bir yapıya sahiptir.

```
app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.j
}

app.Use(async (context, task) =>
{
    Console.WriteLine("Start Use Middleware");
    await task.Invoke();
    Console.WriteLine("Stop Use Middleware");
});
app.Run(async c =>
{
    Console.WriteLine("Start Run Middleware");
});

app.UseHttpsRedirection();
```

```
Start Use Middleware
Start Run Middleware
Stop Use Middleware
```

```
app.Use(async (context, task) =>
{
    Console.WriteLine("Start Use Middleware 1");
    await task.Invoke();
    Console.WriteLine("Stop Use Middleware 1");
});
app.Use(async (context, task) =>
{
    Console.WriteLine("Start Use Middleware 2");
    await task.Invoke();
    Console.WriteLine("Stop Use Middleware 2");
});
app.Use(async (context, task) =>
{
    Console.WriteLine("Start Use Middleware 3");
    await task.Invoke();
    Console.WriteLine("Stop Use Middleware 3");
});
```

```
Start Use Middleware 1
Start Use Middleware 2
Start Use Middleware 3
Stop Use Middleware 3
Stop Use Middleware 2
Stop Use Middleware 1
```

Map Metodu

- Bazen middleware'i talep gönderen path'e göre filtrelemek isteyebiliriz. Bunun için Use ya da Run fonksiyonlarında if kontrolü sağlayabilir yahut Map metodu ile daha profesyonel işlem yapabiliriz.

```
app.Use(async (context, task) =>
{
    Console.WriteLine("Start Use Middleware 1");
    await task.Invoke();
    Console.WriteLine("Stop Use Middleware 1");
});

app.Map("/weatherforecast", builder =>
{
    builder.Run(async c => await c.Response.WriteAsync("Run middleware'i tetiklendi"));
});

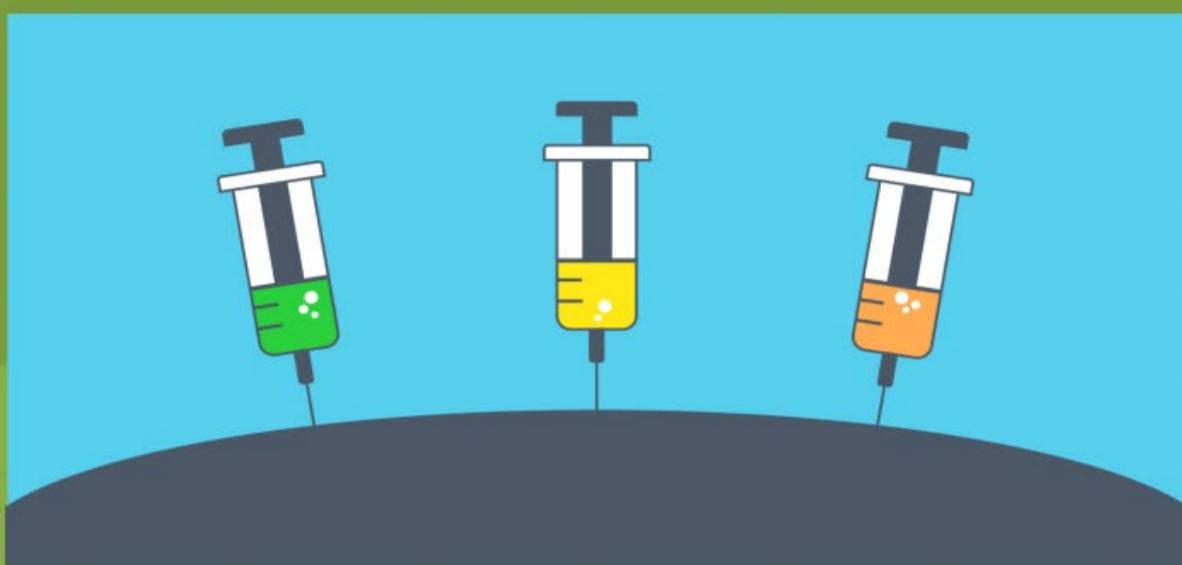
app.Map("/home", builder =>
{
    builder.Use(async (context, task) =>
    {
        Console.WriteLine("Start Use Middleware2 ");
        await task.Invoke();
        Console.WriteLine("Stop Use Middleware2 ");
    });
});
```

MapWhen Metodu

- Map metodu ile sadece request'in yapıldığı path'e göre filtreleme yapılırken, MapWhen metodu ile gelen request'in herhangi bir özelliğine göre bir filtreleme işlemi gerçekleştirilebilir.

```
app.MapWhen(c => c.Request.Method == "GET", builder =>
{
    builder.Use(async (context, task) =>
    {
        Console.WriteLine("Start Use Middleware2 ");
        await task.Invoke();
        Console.WriteLine("Stop Use Middleware2 ");
    });
});
```

Derinlemesine Dependency Injection – IoC Yapılanması

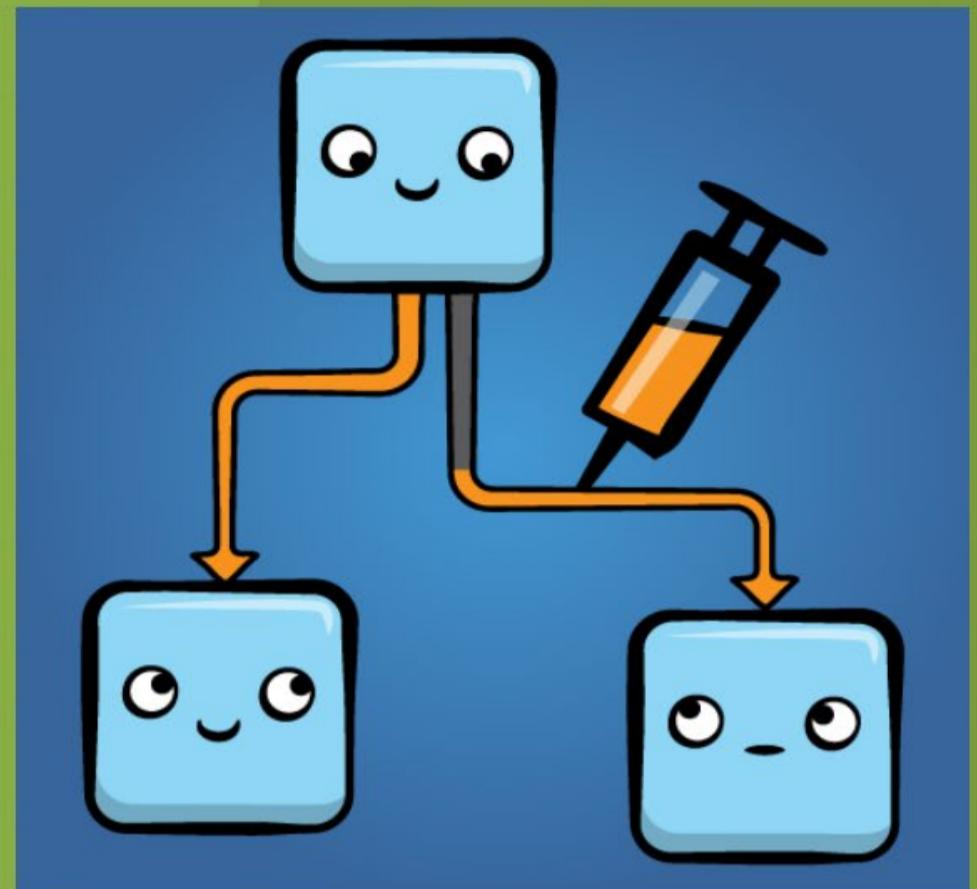


Dependency Injection Nedir?

Bağımlılık oluşturacak parçaların ayrılp, bunların dışarıdan verilmesiyle sistem içerisindeki bağımlılığı minimize etme işlemidir.

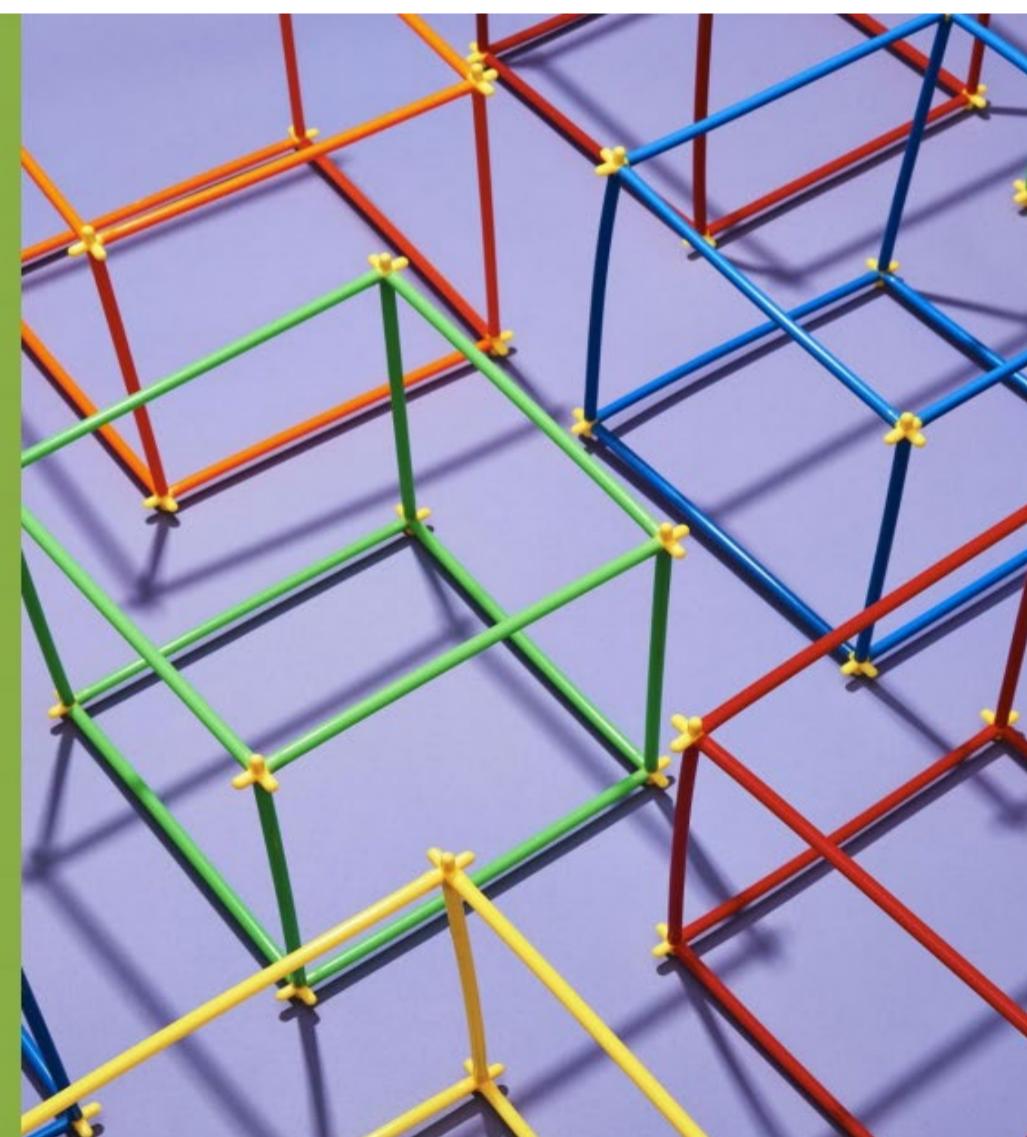
Bir prensibdir yaparsan doğru yapmazsan yanlış değildir, daha uygun bir yapıdır.Herhangi bir programlamada kodlarımızı sınıflarda yazıyoruz.Herhangi bir sınıfın herhangi bir noktasında başka bir sınıfın elemanına ihtiyacın varsa bunu new ile oluşturmamaya çalış eğer new ile oluşturulusan bağımlılık oluşur kaidesi vardır. Bağımlılığı soyutlaştırma üzerine oluşmuştur.Bağımlılığı ortadan kaldırmaya çalışmak gereklidir. Ortak bir atadan türetmek buna çözümüdür yani ata bir sınıfta gereklili olan elemana sahip olunur.

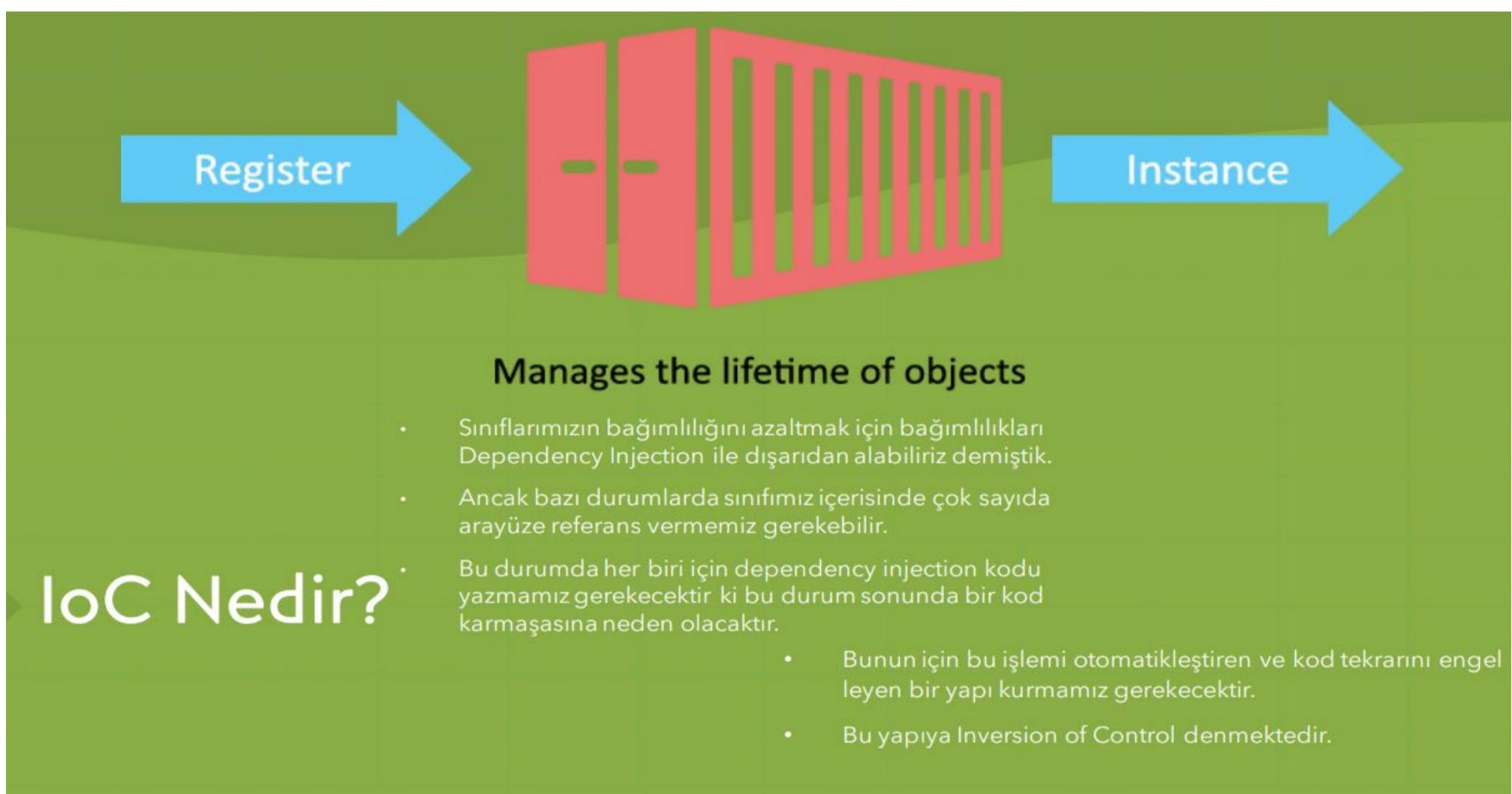
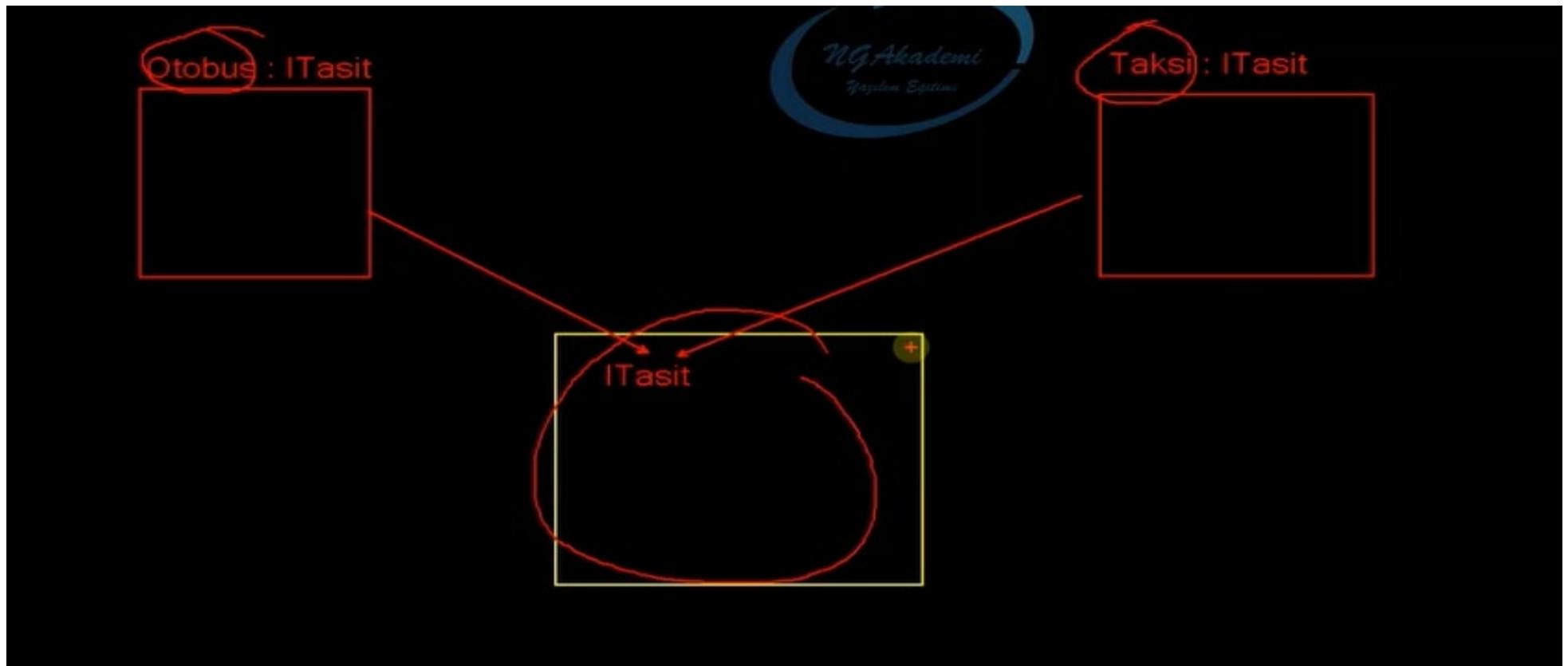
Yani, temel olarak oluşturulan bir sınıf(class) içerisinde başka bir sınıfın nesnesini new anahtar sözcüğüyle oluşturulmamasını söyleyen bir yaklaşımDependency Injection.



Sınıf içerisinde ihtiyaç olan nesnenin ya constructor'dan ya da setter metoduyla parametre olarak alınması gerektiğini savunur. Böylece her iki sınıfı birbirinden izole etmiş olduğumuzu savunmaktadır.

Dependency Injection, bağımlılıkları soyutlamak demektir.





IoC Çalışma Mantığı : Dependency Injection kullanılarak enjekte edilecek olan tüm değerler/nesneler IoC Container dediğimiz bir sınıfı tutulurlar. Ve ihtiyaç doğrultusunda bu değerler/nesneler çağrılarak elde edilirler.

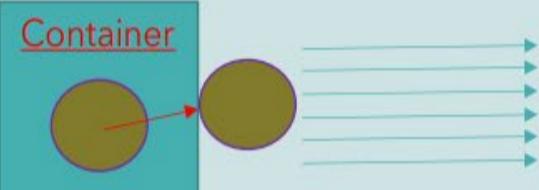
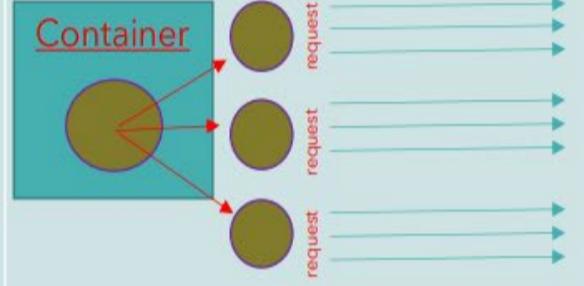
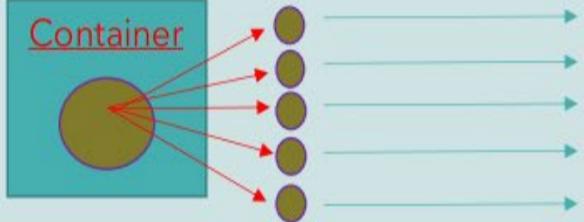
Asp.NET Core'da IoC Yapılanması : .NET uygulamalarında IoC yapılmasını sağlayan third party frameworkler mevcuttur.

- Structuremap
- AutoFac
- Ninject

- Vs...
- Asp.NET Core mimarisi, bu third party kütüphaneler kadar yetenekli olmasada içerisinde built-in(dahili) olarak IoC Container modülü barındırmaktadır.

Built-in IoC Container

Built-in IoC Container, içerisinde koyulacak değerleri/nesneleri üç farklı davranışla alabilmektedir.

Singleton	Scoped	Transient
<p>Uygulama bazlı tekil nesne oluşturur. Tüm taleplere o nesneyi gönderir.</p> 	<p>Her request başına bir nesne üretir ve o request pipeline'nında olan tüm isteklere o nesneyi gönderir.</p> 	<p>Her request'in her talebine karşılık bir nesne üretir ve gönderir.</p> 

Areas Yapılanması

Area Nedir? Bir Web Uygulamasında Ne Amaçla Kullanılır? Web uygulamasında, farklı işlevsellikleri ayırmak için kullanılan özelliklektir.

- Bu farklı işlevsellikler için farklı katmanda, bir route ayarlamamızı sağlayan ve bu katmanda o işleve özel yönetim sergileyen bir yapılmadır.
- Her bir area, içerisinde View, Controller ve Model katmanı barındırabilir.
- Böylece Asp.NET Core uygulamalarında yönetilebilir küçük ve işlevsel gruplar oluşturulabilir.

Area Nerelerde Kullanılabilir? Yönetim panelleri, Faturalandırma sayfaları, İstatiksel paneller, İşlevsel modüller, Uygulamada mantıksal olarak ayrılabilen üst düzey işlevsel bileşenler , Vs...

Area Çok Katmanlı Mimari Değildir!

Area Oluşturma

Area Attribute'u , Area içerisindeki controller Area Attribute'u ile işaretlenmelidir.

- Böylece ilgili alanın uygulamadaki adı resmiyette belirtilmiş olacaktır.
- Aksi takdirde farklı area'larda ki controller'lardan aynı isimde olanların çakışma ihtimalleri vardır.

```
[Area("yonetim")]
0 references
public class HomeController : Controller
{
```

exists : Route'un bir Area ile eşleşmesi için kısıtlama uygular.

Area'ya Route Belirleme

- Her bir area, içerisindeki controller'lara erişim için farklı bir route sağlamaktadır.
- Dolayısıyla bu route'ların tarafımızca tasarlanması gerekmektedir.

```
app.UseEndpoints(endpoints =>
{
    endpoints.MapControllerRoute
        name: "MyArea",
        pattern : "[area:exists}/{controller=Home}/{action=Index}/{id?}"
});
```

MapAreaControllerRoute Fonksiyonu : MapControllerRoute, Default area rotası belirlememizi sağlar.

- MapAreaControllerRoute isle bir area'ya ait/özel rota belirlememizi sağlar.

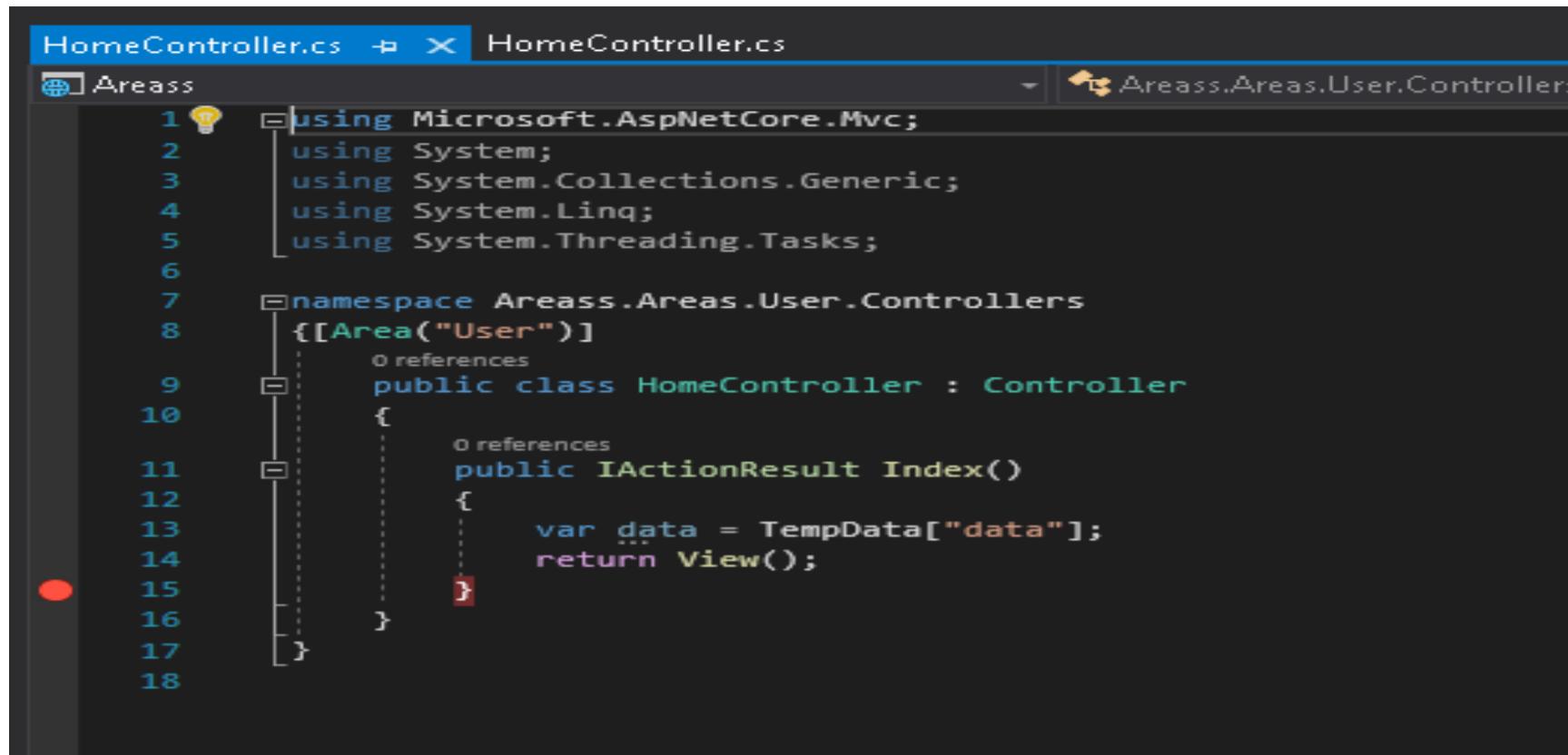
```
app.UseEndpoints(endpoints =>
{
    endpoints.MapAreaControllerRoute(
        name: "MyArea1",
        areaName: "yonetim",
        pattern: "yonetim_sayfasi/{controller=Home}/{action=Index}/{id?}"
    );
});
```

Area'lar Arası Bağlantı Oluşturma : İhtiyaç doğrultusunda area'lar arası bağlantı verilebilmektedir.

TagHelpers	HtmlHelpers
<pre><a asp-action="Index" asp-controller="Home" asp-area="yonetim">Yonetim</pre>	<pre>@Html.ActionLink("Yonetim", "Index", "Home", new { area = "yonetim" }) Yonetim</pre>

Area'lar Arası Veri Taşıma : TempData keyword'ünü kullanarak Area'lar arası veri taşıması yapabiliriz.

```
namespace Areas.Areas.Admin.Controllers
{[Area("Admin")]
public class HomeController : Controller
{
    public IActionResult Index()
    {
        TempData["data"] = "Arealar arası veri taşıma işlemi başarılı";
        return RedirectToAction("Index", "Home", new { area = "User" });
    }
}}
```



```
HomeController.cs  X  HomeController.cs
Areas
1 using Microsoft.AspNetCore.Mvc;
2 using System;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Threading.Tasks;
6
7 namespace Areas.Areas.User.Controllers
8 {
9     [Area("User")]
10    public class HomeController : Controller
11    {
12        public IActionResult Index()
13        {
14            var data = TempData["data"];
15            return View();
16        }
17    }
18}
```

VİEWMODEL & DTO

ViewModel Nedir? ViewModel, temelde iki farklı senaryoya karşılık sorumluluk üstlenen ve biz yazılım geliştiricilerinişini kolaylaştıran operasyonel nesnelerdir.

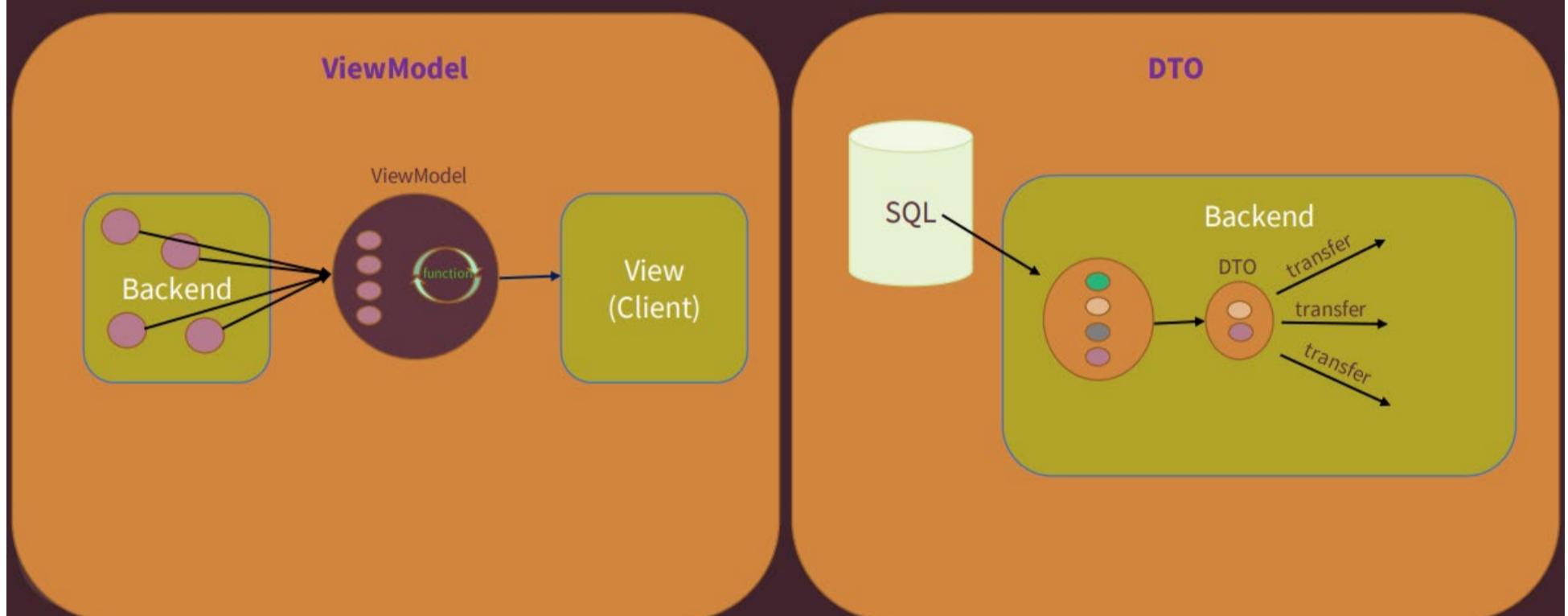
- **1. Senaryo** OOP yapılanmasında bir modelin kullanıcıyla etkileşimi neticesinde kullanılan ve esas datanın memberlarını temsil eden ve süreçte ilgili model yerine veri taşıma/transfer operasyonunu üstlenen bir nesnedir.
- **2. Senaryo** Birden fazla modeli/değeri/veriyi tek bir nesne üzerinde birleştirme görevi gören nesnedir.

DTO Nedir? DTO(Data Transfer Object)

- Herhangi bir davranışlı olmayan ve uygulamanın çeşitli yerlerinde yalnızca bir veri tüketimi ve iletimi için kullanılan, veritabanındaki herhangi bir verinin transfer nesnesidir/karşılığıdır/görünümüdür.

ViewModel	DTO
<ul style="list-style-type: none"> Kullanıcıya sunulacak verinin view'e uygun/view'in beklediği şekilde tasarlanmış modelidir. Veriyi görünüm/sunum/presentation için anlamlı hale getirir. İşlevsel fonksiyonlar(metot) barındırabilir. İçerisinde bir veya birden fazla DTO temsil edebilir. DTO'ya nazaran daha karmaşıktır. 	<ul style="list-style-type: none"> Bir verinin(genellikle veritabanından gelen verinin) transfer modellemesidir. Transfer edilecek olan ilgili verideki sadece ihtiyaç olunandataları temsil eder. Görünüm/sunum/presentation için kullanılabilir lakin bunun dışında uygulamanın herhangi bir katmanında çeşitli veri tüketimi ve transferi içinde kullanılmaktadır. Herhangi bir fonksiyonellik barındırmaz. Salt veriyi temsil eder.

Temel Amaç



Senaryo 1 Bir Model'in View'de ki Etkileşimine Uygun Parçasını Temsil Etme.

Kullanıcıya sunulan hiçbir veri direkt olarak veritabanındaki entity türünden olmamalıdır. Bu tarz durumlarda ViewModel kullanılmalıdır.

Senaryo 2 Birden fazla nesneyi tek bir nesneye bağlama.

Sözleşme/Kontrat Mantığı Nedir? Backend'de üretilen bir verinin client'a gönderilmesi için tasarlanan ViewModel o işlemin sözleşmesi/kontratı olmaktadır.

- Haliyle Backend'den gelecek datayı client'ın uygun formatta karşılayabilmesi için kesinlikle o türden bir nesne oluşturulması gerekecektir.

ViewModel'lar da Validation Durumları : Kullanıcıdan alınan veriler iş kuralı gereği kontrol edilirler. Bizler bu kontrollere validation diyoruz.

Kullanıcılarından gelen veriler kesinlikle veritabanı tablolarının karşılığı olan entity modelleri olmamalıdır! ViewModel olarak alınmalıdır! Ve tüm validation'lar bu ViewModel nesneleri üzerinde gerçekleştirilmelidir.

ViewModel'ı Entity Model'e Dönüştürme : Kullanıcıdan gelendataları ViewModel ile karşıladıktan sonra bu ViewModel'da ki verileri veritabanına kaydetmek isteyebiliriz. Bu durumda, bu verileri Entity Model'a dönüştürmemiz gerekecektir. Bunun için aşağıdaki yöntemlerden herhangibiri kullanılabilir:

- Manuel Dönüştürme

```
BlogValidator.cs*          BlogController.cs  X  Blog.cs           UserComment.cs      BlogModel2.cs       BlogImageadd.cs    BlogAdd.cshtml
Coreproje
89
90
91     [HttpPost]
92     public IActionResult BlogAdd(BlogImageadd b)
93     {
94         var username = User.Identity.Name;
95
96         ViewBag.name = username;
97
98         var usermail = c.Users.Where(x => x.UserName == username).Select(y => y.Email).FirstOrDefault();
99         Blog p = new Blog();
100        var UserID = c.Users.Where(x => x.Email == usermail).Select(y => y.Id).FirstOrDefault();
101        if (b.BlogImage != null)
102        {
103            var extension = Path.GetExtension(b.BlogImage.FileName);
104            var newimagename = Guid.NewGuid() + extension;
105            var location = Path.Combine(Directory.GetCurrentDirectory(),
106                                         "wwwroot/BlogImages/", newimagename);
107            var stream = new FileStream(location, FileMode.Create);
108            b.BlogImage.CopyTo(stream);
109            p.BlogImage = newimagename;
110
111            p.BlogStatus = true;
112            p.UserId = UserID;
113            p.BlogCreateDate = DateTime.Now;
114            p.BlogThumbnailImage = p.BlogImage;
115            p.CategoryID = b.CategoryID;
116            p.BlogContent = b.BlogContent;
117            p.BlogTitle = b.BlogTitle;
118            bm.Tadd(p);
119            return RedirectToAction("BlogListByWriter");
120        }
121    }
```

The Solution Explorer on the right shows the project structure with various controllers and validation files like AbstractValidator.cs and BlogValidator.cs.

• Implicit Operator Overload İle Dönüştürme

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

- Solution Explorer:** Shows the solution 'Coreproje' containing seven projects: BlogApiDemo, BusinessLayer, CoreLayer, Coreproje, Coreproje, Models, and ViewComponents.
- Code Editor:** The file `BlogImageadd.cs` is open, showing the implementation of an implicit operator. A red box highlights the code block:

```
#region implicit/Bilinçsiz Tür Dönüşü
public static implicit operator Blog(BlogImageadd model)//View modeli deki veriler entity'ye dönüştürme
{
    return new Blog
    {
        BlogTitle = model.BlogTitle,
        BlogContent = model.BlogContent,
        BlogCreateDate = model.BlogCreateDate,
        BlogStatus = model.BlogStatus,
        CategoryID = model.CategoryID,
        UserId = model.UserId
    };
}
```
- Code Editor:** The file `BlogController.cs` is open, showing the `BlogAdd` action method. A red box highlights the `#region implicit` block:

```
public IActionResult BlogAdd(BlogImageadd b)
{
    var username = User.Identity.Name;
    ViewBag.name = username;

    var usermail = c.Users.Where(x => x.UserName == username).Select(y => y.Email).FirstOrDefault();
    Blog p = new Blog();
    var UserID = c.Users.Where(x => x.Email == usermail).Select(y => y.Id).FirstOrDefault();
    if (b.BlogImage != null)
    {
        var extension = Path.GetExtension(b.BlogImage.FileName);
        var newimagename = Guid.NewGuid() + extension;
        var location = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot/BlogImages/", newimagename);
        var stream = new FileStream(location, FileMode.Create);
        b.BlogImage.CopyTo(stream);
        p.BlogImage = newimagename;
    }
    [Amaçla yöntemi]
    #region implicit
    Blog blog = b;
    #endregion
    blm.Add(p);
    return RedirectToAction("BlogListByWriter");
}
```

• Explicit Operator Overload İle Dönüştürme

The screenshot shows a Visual Studio code editor with several tabs open. The active tab is `BlogImageadd.cs`, which contains C# code. A red box highlights a specific block of code within a region named `#region Explicit/Açık/Bilincli`. This block converts a `BlogImageadd` model object into a `Blog` entity object by setting properties like `BlogTitle`, `BlogContent`, `BlogCreateDate`, `BlogStatus`, `CategoryID`, and `UserId`.

```

17     public bool BlogStatus { get; set; }
18     public int CategoryID { get; set; }
19     public Category Category { get; set; }
20     public int UserId { get; set; }
21
22     [implicit/Bilincsiz Tür Dönüşü]
23     #region Explicit/Açık/Bilincli
24     public static explicit operator Blog(BlogImageadd model)//View modeli deki veriler entity'ye dönüştürme
25     {
26         return new Blog
27         {
28             BlogTitle = model.BlogTitle,
29             BlogContent = model.BlogContent,
30             BlogCreateDate = model.BlogCreateDate,
31             BlogStatus = model.BlogStatus,
32             CategoryID = model.CategoryID,
33             UserId = model.UserId
34         };
35     }
36     #endregion
37 }

```

The screenshot shows another Visual Studio code editor with the `BlogController.cs` tab active. A red box highlights a block of code within a region named `#region explicit`. This block performs a file operation to copy a blog image from the temporary location to a permanent `wwwroot/BlogImages/` directory, renaming it to a unique GUID-based name.

```

91     public IActionResult BlogAdd(BlogImageadd b)
92     {
93         var username = User.Identity.Name;
94
95         ViewBag.name = username;
96
97         var usermail = c.Users.Where(x => x.UserName == username).Select(y => y.Email).FirstOrDefault();
98         Blog p = new Blog();
99         var UserID = c.Users.Where(x => x.Email == usermail).Select(y => y.Id).FirstOrDefault();
100        if (b.BlogImage != null)
101        {
102            var extension = Path.GetExtension(b.BlogImage.FileName);
103            var newimagename = Guid.NewGuid() + extension;
104            var location = Path.Combine(Directory.GetCurrentDirectory(),
105                "wwwroot/BlogImages/", newimagename);
106            var stream = new FileStream(location, FileMode.Create);
107            b.BlogImage.CopyTo(stream);
108            p.BlogImage = newimagename;
109        }
110        [Amele yöntemi]
111        [implicit]
112        #region explicit
113        Blog blog = (Blog)b;
114        #endregion
115        bm.Add(p);
116        return RedirectToAction("BlogListByWriter");
117    }
118
119    public IActionResult DeleteBlog(int id)
120    {
121        var username = User.Identity.Name;
122        ViewBag.name = username;
123    }

```

● Reflection İle Dönüştürme

The screenshot shows a Visual Studio code editor with the `TypeConversion.cs` tab active. A red box highlights a method named `Conversion<T, TResult>(T model)` within the `TypeConversion` class. This method uses reflection to iterate through the properties of the input `T` model and create a corresponding `TResult` object by setting each property's value.

```

1     using System;
2     using System.Collections.Generic;
3     using System.Linq;
4     using System.Reflection;
5     using System.Threading.Tasks;
6
7     namespace Coreproje.reflectionn
8     {
8
9         public static class TypeConversion
10     {
11         public static TResult Conversion <T, TResult>(T model) where TResult: class,new ()
12         {
13             TResult result = new TResult();
14             typeof(T).GetProperties().ToList().ForEach(p=>
15             {
16                 PropertyInfo property = typeof(TResult).GetProperty(p.Name);
17                 property.SetValue(result,p.GetValue(model));
18             });
19         }
20         return result;
21     }
22 }

```

```
90 }  
91 [HttpPost]  
92 public IActionResult BlogAdd(BlogImageadd b)  
93 {  
94     var username = User.Identity.Name;  
95  
96     ViewBag.name = username;  
97  
98     var usermail = c.Users.Where(x => x.UserName == username).Select(y => y.Email).FirstOrDefault();  
99     Blog p = new Blog();  
100    var class EntityLayer.Concrete.Blog  
101    if  
102    {  
103        var extension = Path.GetExtension(b.BlogImage.FileName);  
104        var newimagename = Guid.NewGuid() + extension;  
105        var location = Path.Combine(Directory.GetCurrentDirectory(),  
106            "wwwroot/BlogImages/", newimagename);  
107        var stream = new FileStream(location, FileMode.Create);  
108        b.BlogImage.CopyTo(stream);  
109        p.BlogImage = newimagename;  
110    }  
111    [Amele yöntemi]  
112    implicit  
113  
114    [Explicit]  
115    #region Reflection  
116    Blog blog = TypeConversion.Conversion<BlogImageadd, Blog>(b);  
117    #endregion  
118    bm.Tadd(p);  
119    return RedirectToAction("BlogListByWriter");  
120 }  
121  
122 }  
123  
124 }  
125  
126 }  
127  
128 }  
129  
130 }  
131  
132 }  
133 }  
0 references
```

• AutoMapper Kütüphanesi İle Dönüştürme

Örnek ViewModel:

```
10 }  
11 public class Blog  
12 { [Key]  
13     35 references  
14     public int BlogID { get; set; }  
15     23 references  
16     public string BlogTitle { get; set; }  
17     22 references  
18     public string BlogContent { get; set; }  
19     4 references  
20     public string BlogThumbnailImage { get; set; }  
21     18 references  
22     public string BlogImage { get; set; }  
23     24 references  
24     public DateTime? BlogCreateDate { get; set; }  
25     14 references  
26     public bool? BlogStatus { get; set; }  
27     1 reference  
28     public int? BlogDisLike { get; set; }  
29     0 references  
30     public int? BlogTotalScore { get; set; }  
31     5 references  
32     public int? BlogLike { get; set; }  
33     6 references  
34     public int? BlogRatingCount { get; set; }  
35     2 references  
36     public int? BlogRatingAvg { get; set; }  
37     4 references  
38     public int CategoryID { get; set; }  
39     10 references  
40     public Category Category { get; set; }  
41     0 references  
42     public List<Comment> Comments { get; set; }  
43 }
```

The screenshot shows the Visual Studio IDE interface. The code editor displays the `BlogImageadd.cs` file, which defines a class `BlogImageadd` with properties like `BlogTitle`, `BlogContent`, `BlogThumbnailImage`, `BlogImage`, `BlogCreateDate`, `BlogStatus`, `CategoryID`, `Category`, and `UserId`. The Solution Explorer on the right lists various projects and files, including `BlogImageadd.cs` under the `Models` folder of the `Coreproj` project.

```
public class BlogImageadd
{
    public string BlogTitle { get; set; }
    public string BlogContent { get; set; }
    public string BlogThumbnailImage { get; set; }
    public IFormFile BlogImage { get; set; }
    public DateTime BlogCreateDate { get; set; }
    public bool BlogStatus { get; set; }

    public int CategoryID { get; set; }
    public Category Category { get; set; }
    public int UserId { get; set; }
}
```

The screenshot shows the Visual Studio IDE interface. The code editor displays the `BlogController.cs` file, which contains the `BlogAdd` action method. This method handles both GET and POST requests. For a GET request, it retrieves the writer's information and returns a view. For a POST request, it adds a new blog entry to the database, handling file uploads for the blog image. The Solution Explorer on the right shows the `BlogController.cs` file under the `Controllers` folder of the `Coreproj` project.

```
[HttpGet]
public IActionResult BlogAdd()
{
    ViewBag.id = writer.id;
    var values = bm.GetBlog();
    return View(values);
}

[HttpPost]
public IActionResult BlogAdd(BlogImageadd b)
{
    var username = User.Identity.Name;

    ViewBag.name = username;

    var usermail = c.Users.Where(x => x.UserName == username).Select(y => y.Email).FirstOrDefault();
    Blog p = new Blog();
    var UserID = c.Users.Where(x => x.Email == usermail).Select(y => y.Id).FirstOrDefault();
    if (b.BlogImage != null)
    {
        var extension = Path.GetExtension(b.BlogImage.FileName);
        var newimagename = Guid.NewGuid() + extension;
        var location = Path.Combine(Directory.GetCurrentDirectory(),
            "wwwroot/BlogImages/", newimagename);
        var stream = new FileStream(location, FileMode.Create);
        b.BlogImage.CopyTo(stream);
        p.BlogImage = newimagename;
    }
    p.BlogStatus = true;
    p.UserId = UserID;
    p.BlogCreateDate = DateTime.Now;
    p.BlogThumbnailImage = p.BlogImage;
    p.CategoryID = b.CategoryID;
    p.BlogContent = b.BlogContent;
    p.BlogTitle = b.BlogTitle;
    bm.Tadd(p);
    return RedirectToAction("BlogListByWriter");
}
```

```
1 @model Coreproje.Models.BlogImageadd
2
3     ViewData["Title"] = "BlogAdd";
4     Layout = "~/Views/Shared/WriterLayout.cshtml";
5
6     <form class="form-group" method="post" asp-action="BlogAdd" asp-controller="Blog" enctype="multipart/form-data">
7
8         <h1>Blog Ekleme Sayfası</h1>
9         <script src="https://unpkg.com/sweetalert/dist/sweetalert.min.js"></script>
10        <script src="~/lib/jquery/dist/jquery.js"></script>
11        <br />
12        <div class="form-group">
13            @Html.Label("Resim")
14            <input asp-for="BlogImage" class="form-control" type="file" />
15        </div>
16        <br />
17        @Html.Label("Blog")
18        @Html.TextBoxFor(x => x.BlogTitle, new { @class = "form-control" })
19        @Html.ValidationMessageFor(x => x.BlogTitle, "", new { @class = "text-danger" })
20        <br />
21        @Html.Label("Kategori")
22        <br />
23        <select asp-for="CategoryID" class="form-control" asp-items="ViewBag.CategoryID"></select>
24        <br />
25        @Html.Label("İçerik")
26        @Html.TextAreaFor(x => x.BlogContent, 10, 8, new { @class = "form-control" })
27        @Html.ValidationMessageFor(x => x.BlogContent, "", new { @class = "text-danger" })
28        <br />
29        <button type="submit" class="btn btn-primary submit mb-4">EKLE</button>
30
31    </form>
```

https://localhost:5001/Blog/BlogAdd/

Resim

Dosya Seç

Blog

ASP.NET Core 5.0 yenilikleri -Swagger

Kategori

Yazılım

İçerik

MVC ve Razor geliştirmeleri ASP.NET Core
Model bağlama DateTime değerini UTC olarak ayarlama
Model bağlama artık UTC saat dizelerini ye bağlamayı DateTimedestekliyor. İstek bir UTC saat dizesi içeriyorsa, model bağlaması bunu UTC'ye DateTime'e bağlar. Örneğin, aşağıdaki saat dizesi UTC DateTime ile ilişkilidir: <https://example.com/mycontroller/myaction?time=2019-06-14T02%3A30%3A04.0576719Z>

C# 9 kayıt türleriyle model bağlama ve doğrulama
C# 9 kayıt türleri, MVC denetleyicisinde veya Razor Sayfada model bağlama ile kullanılabilir. Kayıt türleri, ağ üzerinden iletilen verileri modellemek için iyi bir yoludur.
Örneğin, aşağıdakiler PersonController model bağlama ve form doğrulama ile kayıt türünü kullanır Person :

C#

EKLE

appsettings.json

appsettings.json Dosyası Nedir? Ne İse Yarar?
APPSETTINGS.JSON DOSYASI, ASP.NET CORE UYGULAMALARINI YAPILANDIRMA ARACLARINDAN BİRİSİDİR.

Peki Yapılandırma Ne Demektir? Yapılandırma, bir uygulamanın herhangi bir ortamda gerçekleştireceği davranışlarını belirlememizi sağlayan statik değerin tanımlanmasıdır.

Eski Asp.NET uygulamalarında kullanılan web.config yahut Global.asax gibi dosyalar standart framework'ün yapılandırmasında kullanılan ortamlardır.

Best practices açısından kodun içerisinde username, password, connection string vs. gibi statik tanımlamalar yapılmamalıdır!

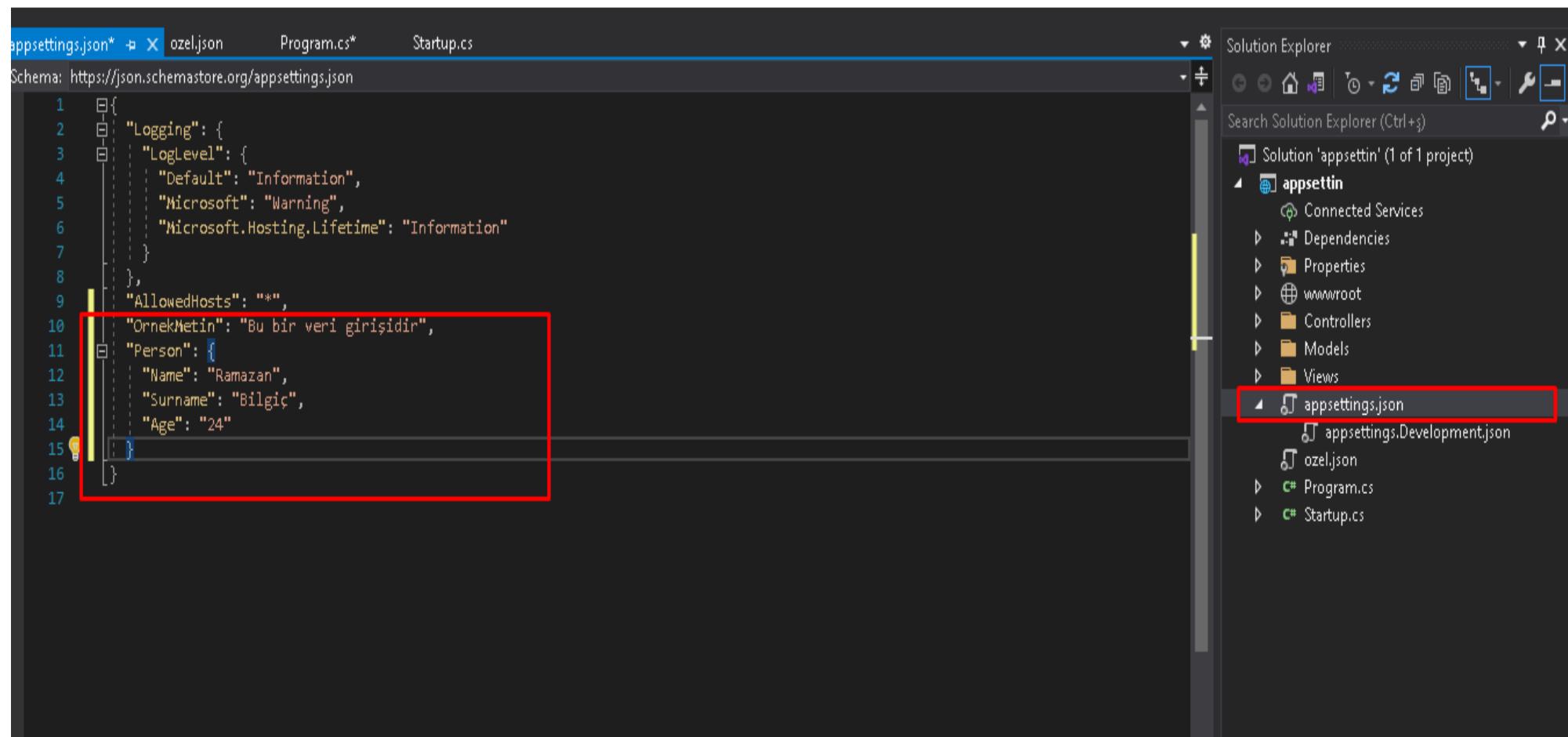
Asp.NET Core'da ki appsettings.json Dışındaki Yapılandırma Araçları Nelerdir?

Appsettings.json | appsettings.{Environment}.json

- **Secrets.json(Secret Manager Tools)**
- **Environment Variables**

appsettings.json Konfigürasyonu :

• Nasıl Veri Eklenir?



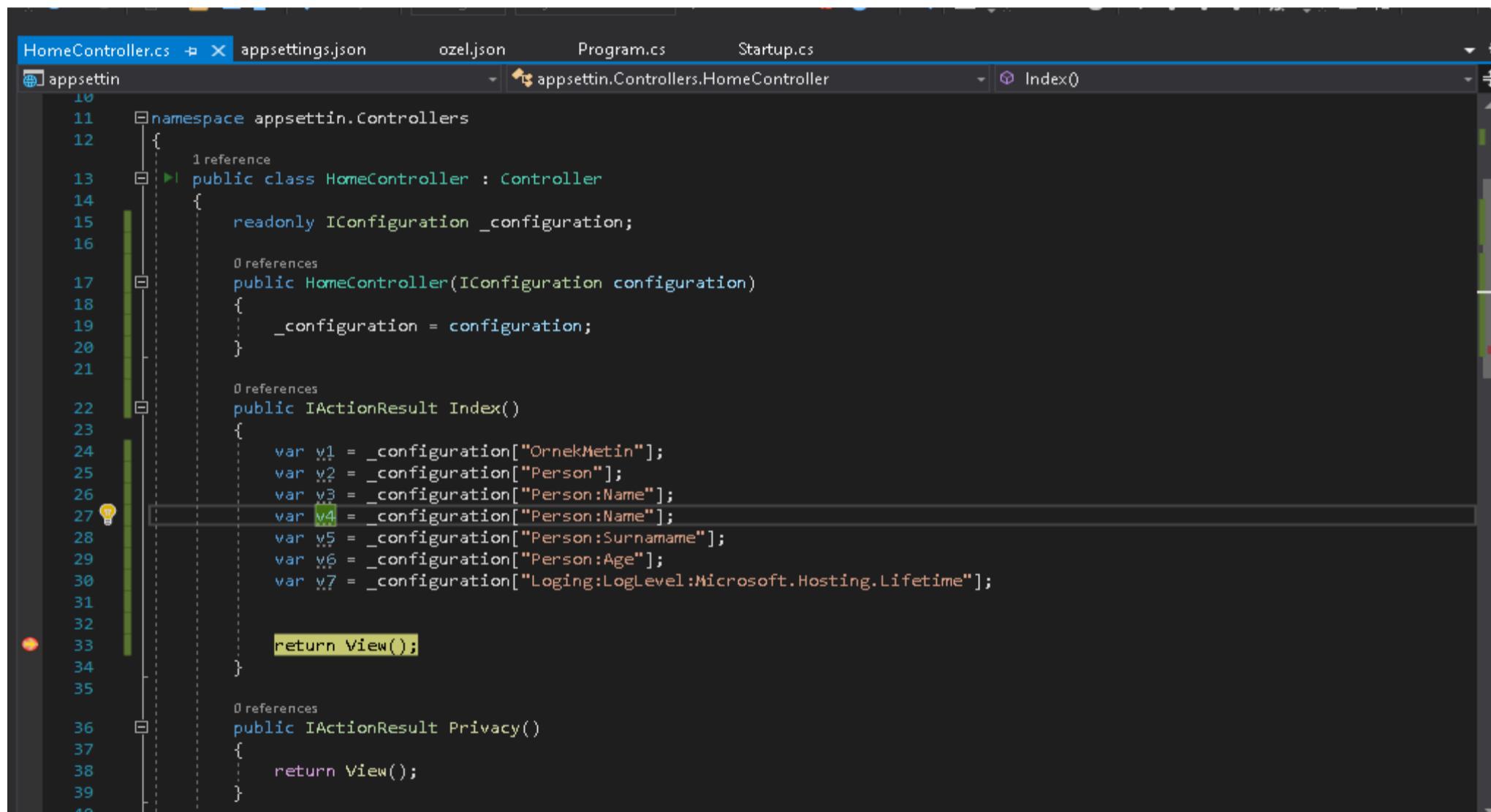
```
appsettings.json* ozel.json Program.cs* Startup.cs
Schema: https://json.schemastore.org/appsettings.json
1  {
2    "Logging": {
3      "LogLevel": {
4        "Default": "Information",
5        "Microsoft": "Warning",
6        "Microsoft.Hosting.Lifetime": "Information"
7      }
8    },
9    "AllowedHosts": "*",
10   "OrnekMetin": "Bu bir veri girişidir",
11   "Person": {
12     "Name": "Ramazan",
13     "Surname": "Bilgiç",
14     "Age": "24"
15   }
16 }
17 ]
```

The screenshot shows the Visual Studio IDE interface. On the left, the code editor displays the contents of the appsettings.json file. A red rectangular box highlights the JSON object starting at line 11, which contains the "Person" configuration. On the right, the Solution Explorer pane shows the project structure, including files like ozel.json, Program.cs, and Startup.cs, along with the appsettings.json file, which is also highlighted with a red box.

- **Eklenen Veri Nasıl Okunur?**

- **IConfiguration Arayüzü Nedir?** Asp.NET Core IOC provider'ında bulunan bir servistir. Bu servis uygulamalarındaki appsetting.json dosyasını okumakla ve içerisindeki valueleri bizlere getirmektedir. Dolayısıyla IOC'dan bu servisi herhangi bir controller'a dependency Injection ile talep edebilir ve appsetting.json dosyasındaki konfigrasyonları kullanabiliriz.

- **Indexer İle Veri Okuma Nasıl Yapılır?**

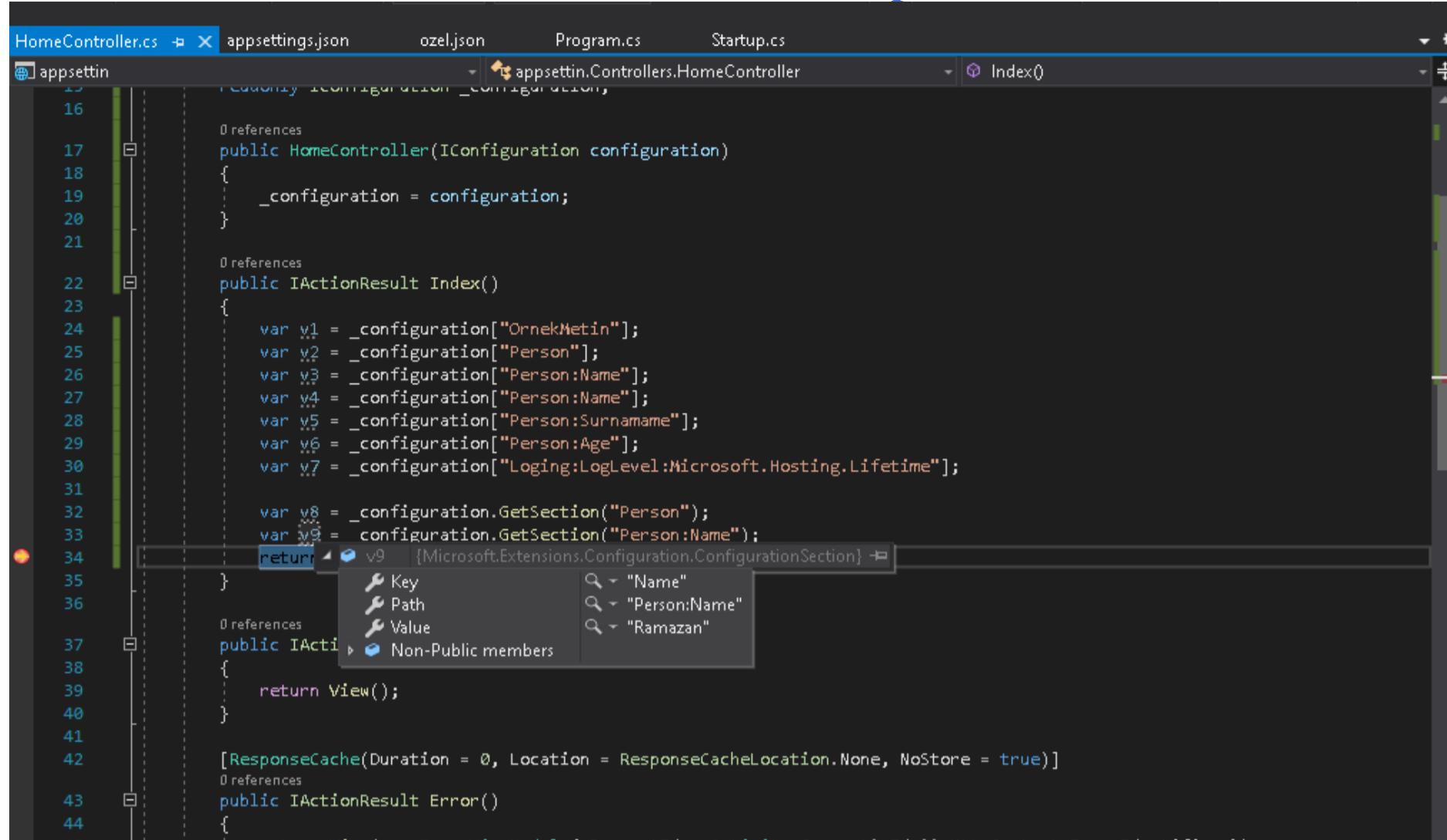


The screenshot shows the Visual Studio IDE interface with the following details:

- Project Explorer:** Shows files like HomeController.cs, appsettings.json, ozel.json, Program.cs, and Startup.cs.
- Code Editor:** Displays the HomeController.cs code. The code uses the IConfiguration interface to read configuration values from the appsettings.json file.
- Toolbars and Menus:** Standard Visual Studio toolbars and menus are visible at the top.
- Status Bar:** Shows the current file as HomeController.cs and the line number as Index() at line 33.

```
10
11     namespace appsettin.Controllers
12     {
13         public class HomeController : Controller
14         {
15             readonly IConfiguration _configuration;
16
17             public HomeController(IConfiguration configuration)
18             {
19                 _configuration = configuration;
20             }
21
22             public IActionResult Index()
23             {
24                 var v1 = _configuration["OrnekMetin"];
25                 var v2 = _configuration["Person"];
26                 var v3 = _configuration["Person:Name"];
27                 var v4 = _configuration["Person:Name"];
28                 var v5 = _configuration["Person:Surname"];
29                 var v6 = _configuration["Person:Age"];
30                 var v7 = _configuration["Logging:LogLevel:Microsoft.Hosting.Lifetime"];
31
32
33                 return View();
34             }
35
36             public IActionResult Privacy()
37             {
38                 return View();
39             }
40         }
41     }
```

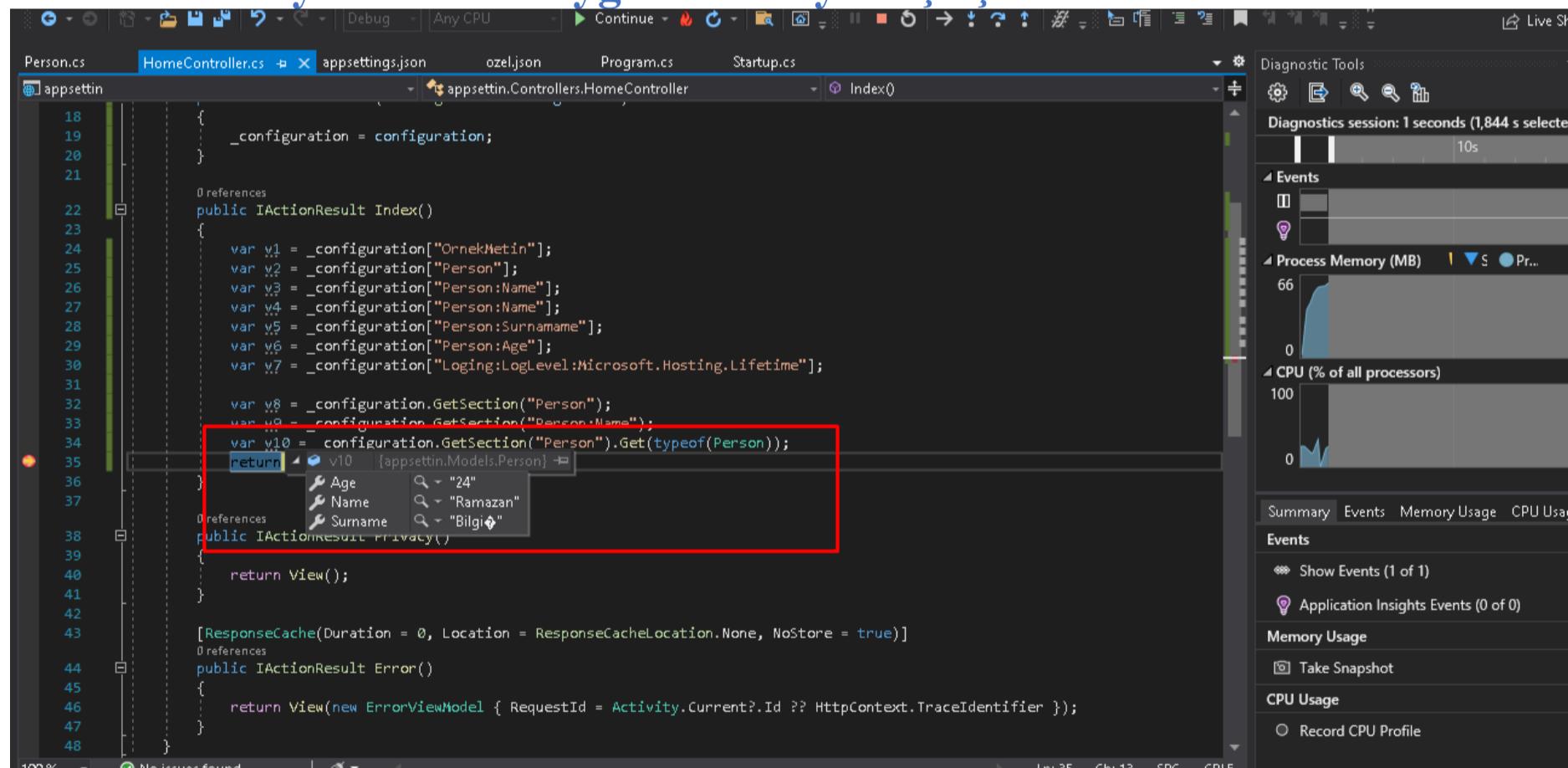
• GetSection Metodu İle Veri Okuma Nasıl Yapılır?



```
HomeController.cs appsettings.json ozel.json Program.cs Startup.cs
appsettin
15     public HomeController(IConfiguration configuration)
16     {
17         _configuration = configuration;
18     }
19
20     public IActionResult Index()
21     {
22         var v1 = _configuration["OrnekMetin"];
23         var v2 = _configuration["Person"];
24         var v3 = _configuration["Person:Name"];
25         var v4 = _configuration["Person:Name"];
26         var v5 = _configuration["Person:Surname"];
27         var v6 = _configuration["Person:Age"];
28         var v7 = _configuration["Logging:LogLevel:Microsoft.Hosting.Lifetime"];
29
30         var v8 = _configuration.GetSection("Person");
31         var v9 = configuration.GetSection("Person:Name");
32         return v9; // Microsoft.Extensions.Configuration.ConfigurationSection
33     }
34
35     public IActionResult Privacy()
36     {
37         return View();
38     }
39
40     [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
41     public IActionResult Error()
42     {
43         return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });
44     }

```

• Get Metoduyla Verileri Uygun Nesneye Eşleştirme

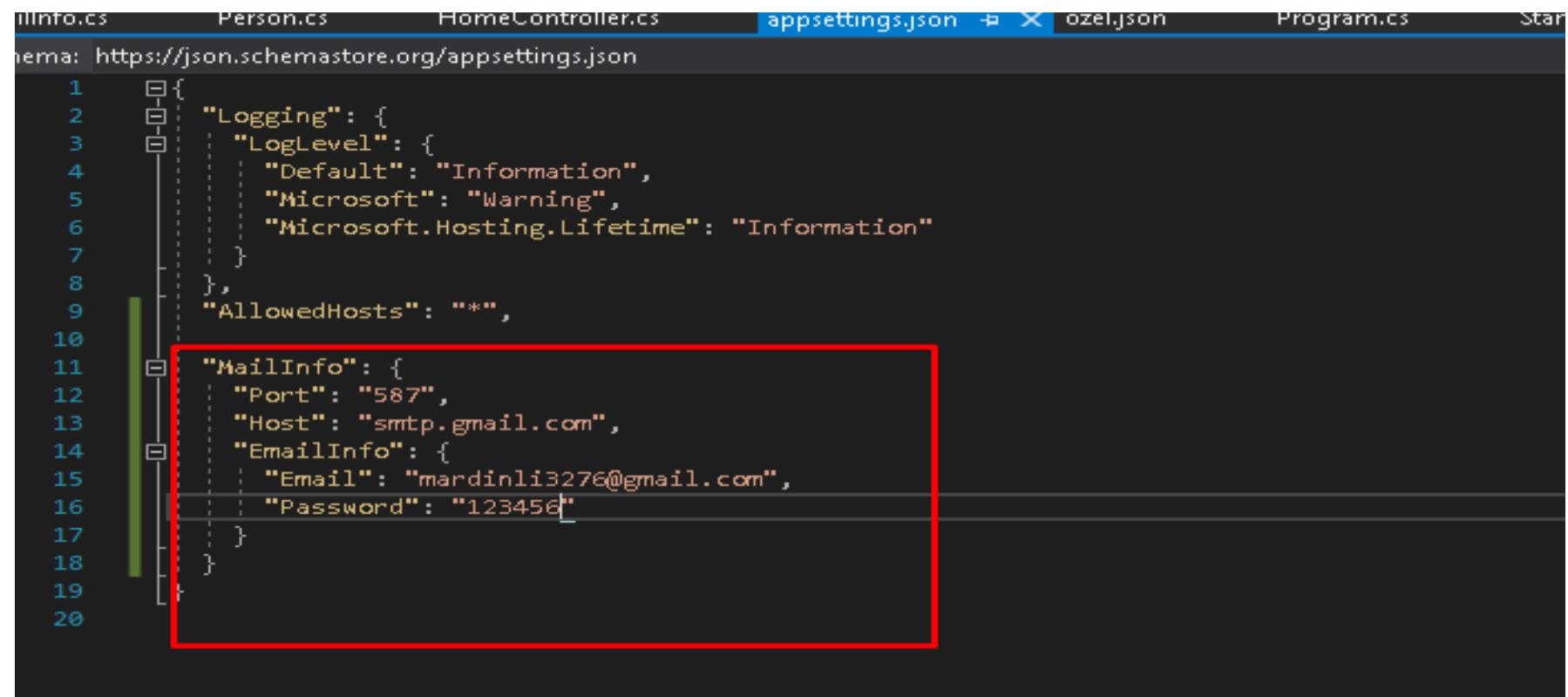


```
Person.cs HomeController.cs appsettings.json ozel.json Program.cs Startup.cs
appsettin
18     _configuration = configuration;
19 }
20
21     public IActionResult Index()
22     {
23         var v1 = _configuration["OrnekMetin"];
24         var v2 = _configuration["Person"];
25         var v3 = _configuration["Person:Name"];
26         var v4 = _configuration["Person:Name"];
27         var v5 = _configuration["Person:Surname"];
28         var v6 = _configuration["Person:Age"];
29         var v7 = _configuration["Logging:LogLevel:Microsoft.Hosting.Lifetime"];
30
31         var v8 = _configuration.GetSection("Person");
32         var v9 = _configuration.GetSection("Person:Name");
33         var v10 = configuration.GetSection("Person").Get(typeof(Person));
34         return v10; // appsettin.Models.Person
35     }
36
37     public IActionResult Privacy()
38     {
39         return View();
40     }
41
42     [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
43     public IActionResult Error()
44     {
45         return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });
46     }
47
48 }
```

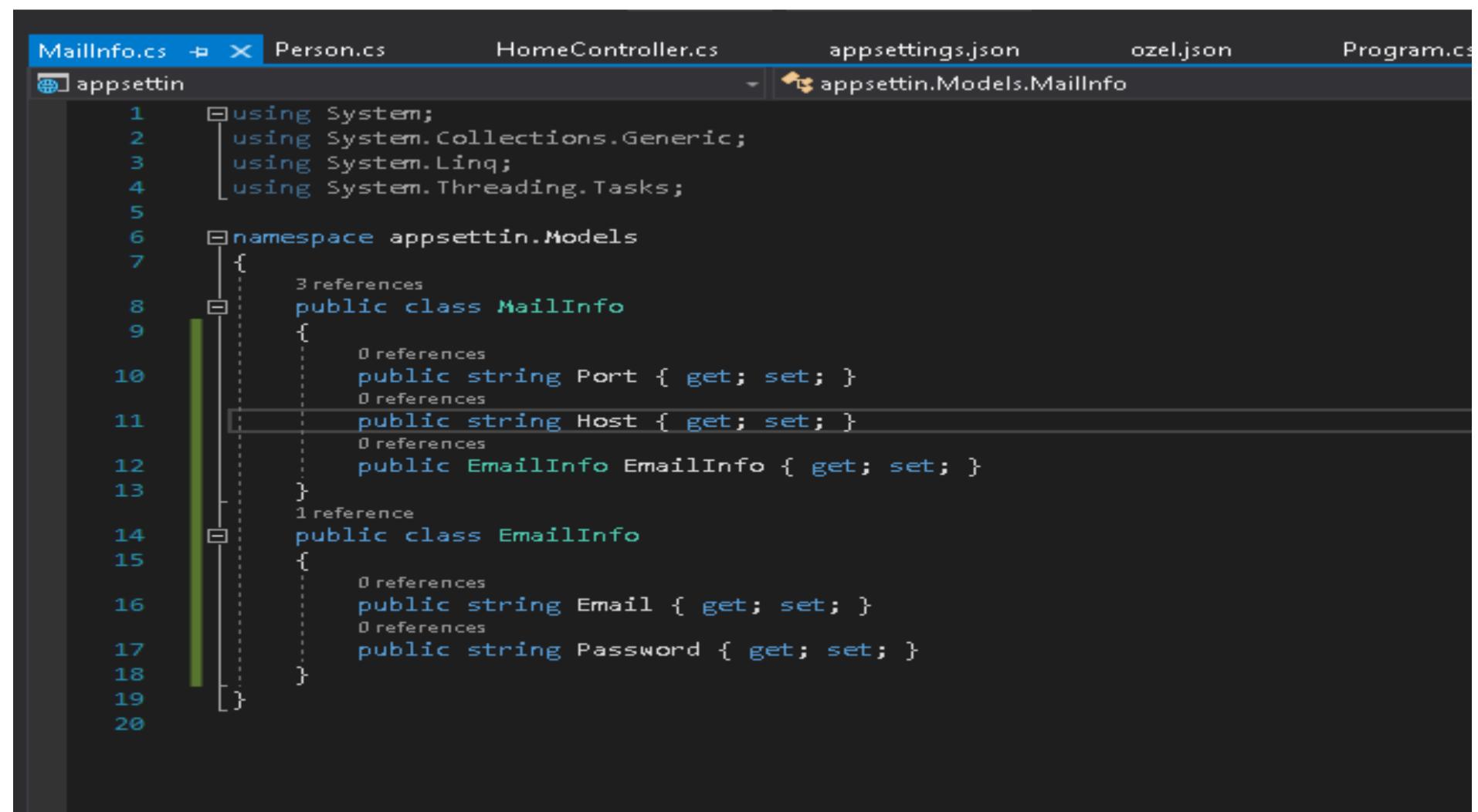
• : Operatörü Nedir?

Options Pattern İle Konfigürasyonları Dependency Injection İle Yapılandırma

Options Pattern appsetting.json dosyasındaki konfigürasyonları
Dependency Injection ile sağlayan nesne modellerle ilgili
konfigürasyonları temsil etmemizi sağlayan bir tasarım desenidir.



```
1  {
2    "Logging": {
3      "LogLevel": {
4        "Default": "Information",
5        "Microsoft": "Warning",
6        "Microsoft.Hosting.Lifetime": "Information"
7      }
8    },
9    "AllowedHosts": "*",
10   "MailInfo": {
11     "Port": "587",
12     "Host": "smtp.gmail.com",
13     "EmailInfo": {
14       "Email": "mardinli3276@gmail.com",
15       "Password": "123456"
16     }
17   }
18 }
19 }
20 }
```



```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5
6  namespace appsetting.Models
7  {
8    public class MailInfo
9    {
10      public string Port { get; set; }
11      public string Host { get; set; }
12      public EmailInfo EmailInfo { get; set; }
13    }
14    public class EmailInfo
15    {
16      public string Email { get; set; }
17      public string Password { get; set; }
18    }
19  }
```

```

14     2 references
15     public class Startup
16     {
17         0 references
18         public Startup(IConfiguration configuration)
19         {
20             Configuration = configuration;
21         }
22
23         2 references
24         public IConfiguration Configuration { get; }
25
26         // This method gets called by the runtime. Use this method to add services to the container.
27         0 references
28         public void ConfigureServices(IServiceCollection services)
29         {
30             services.Configure<MailInfo>(Configuration.GetSection("MailInfo"));
31             services.Add
32             // This method
33             Returns:
34             The IServiceCollection so that additional calls can be chained.
35
36             public void Con
37             {
38                 if (env.IsDevelopment())
39                 {
40                     app.UseDeveloperExceptionPage();
41                 }
42                 else
43                 {
44                     app.UseExceptionHandler("/Home/Error");
45                     // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/
46                     aspnetcore-hsts.
47                     app.UseHsts();
48             }
49         }
50
51         0 references
52     }
53
54     0 references
55 
```

```

1    0 references
2    using System.Collections.Generic;
3    using System.Diagnostics;
4    using System.Linq;
5    using System.Threading.Tasks;
6
7    1 reference
8    namespace appsettin.Controllers
9    {
10        public class HomeController : Controller
11        {
12            0 references
13            public HomeController(IOptions<MailInfo> mailInfo)
14            {
15                _mailInfo = mailInfo.Value ;
16            }
17
18            0 references
19        }
20    }
21
22    0 references
23 
```

```

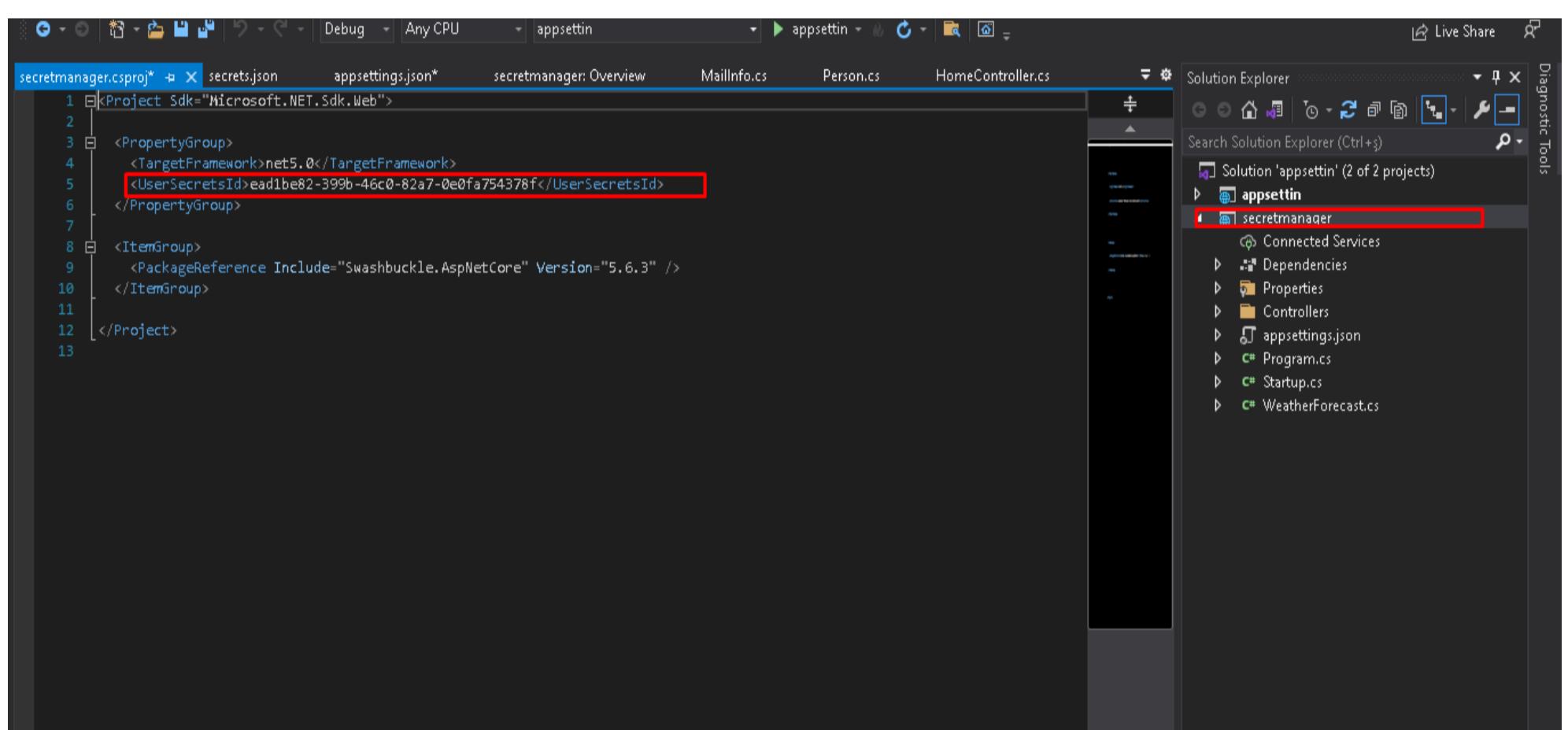
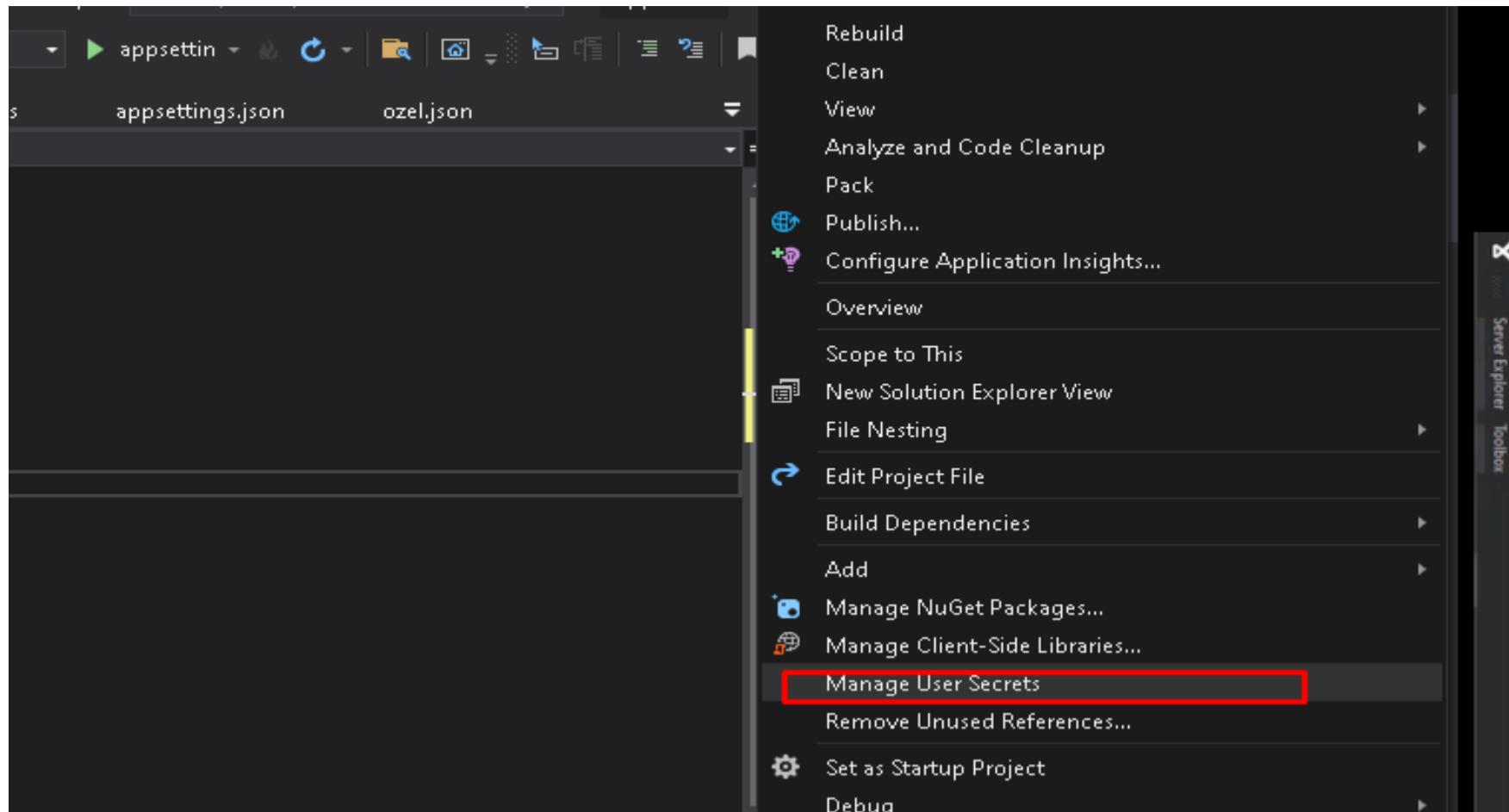
1    0 references
2    using System.Collections.Generic;
3    using System.Diagnostics;
4    using System.Linq;
5    using System.Threading.Tasks;
6
7    1 reference
8    namespace appsettin.Controllers
9    {
10        public class HomeController : Controller
11        {
12            0 references
13            public HomeController(IOptions<MailInfo> mailInfo)
14            {
15                _mailInfo = mailInfo.Value ;
16                _mailInfo {appsettin.Models.MailInfo} →
17                    EmailInfo → {appsettin.Models.EmailInfo}
18                    Email "mardinli3276@gmail.com"
19                    Password "123456"
20            }
21
22            0 references
23            public IActionResult Index()
24            {
25                //var v1 = _configuration["OrnekMetin"];
26                //var v2 = _configuration["Person"];
27                //var v3 = _configuration["Person:Name"];
28                //var v4 = _configuration["Person:Name"];
29                //var v5 = _configuration["Person:Name"];
30            }
31
32            0 references
33        }
34    }
35
36    0 references
37 
```

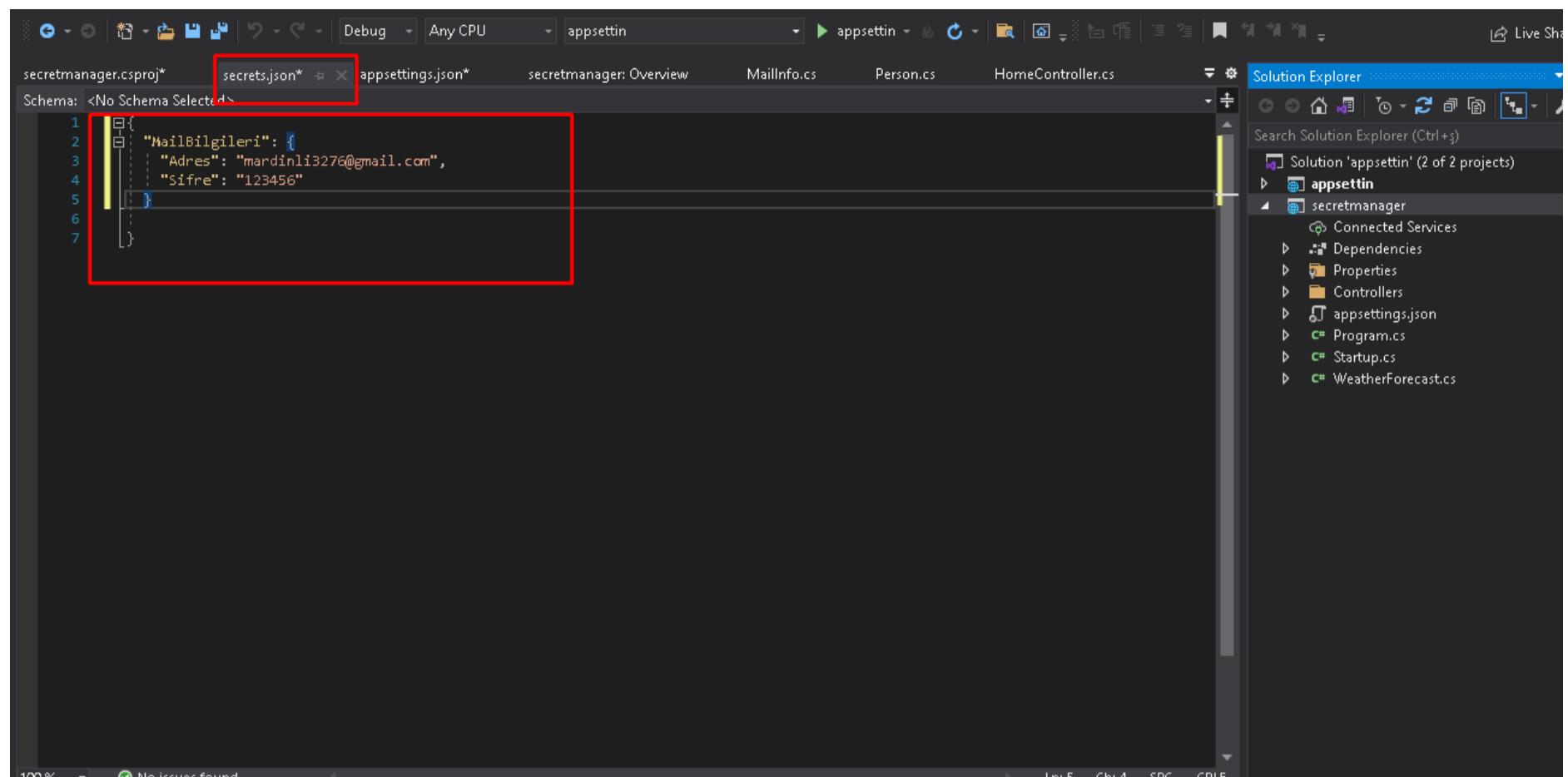
Asp.NET Core Secret Manager Tools secret.json ile Hassas Verilerin Korunması

[Secret manager Tools Nedir?](#) Web uygulamalarında development ortamında kullandığımız bazı verilerimizin canlıya deploy edilmesini istemeyebiliriz.

- Bu verilerimiz;
- Veritabanı bilgilerini barındıran connection string bilgisi,
- Herhangi bir kritik arz eden token değeri,
- Facebook veya Google gibi third party authentication işlemleri yapmamızı sağlayan client secret id değerler vs. olabilir.
- Bu veriler developer ortamında kullanılırken, production ortamında kötü niyetli kişilerin uygulama dosyalarına erişim sağladıkları durumlarda elde edemeyecekleri vaziyette bir şekilde ezilmeleri gerekmektedir. İşte bunun için Secret Manager Tool geliştirilmiştir.
- Web uygulamalarında static olan verileri tekrar tekrar yazmak yerine bir merkezde depolayarak kullanmayı tercih ediyoruz.
- Asp.NET Core uygulamalarında bu merkez genellikle 'appsettings.json' dosyası olmaktadır.
- Bu dosya içerisinde yazılan değerler her ne olursa olsun uygulama publish edildiği taktirde çıktıdan erişilebilir vaziyette olacaktır.
- Dolayısıyla bizler static verilerimizi 'appsettings.json' içerisinde tutabiliyoruz lakin kritik arz eden veriler için burasının pekte ehemmiyetli bir yer olmadığı aşikardır diyebiliriz.







```
secretmanager
1  using Microsoft.AspNetCore.Mvc;
2  using Microsoft.Extensions.Configuration;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Threading.Tasks;
7
8  namespace secretmanager.Controllers
9  {
10     [Route("api/[controller]")]
11     [ApiController]
12
13     public class HomeController : Controller
14     {
15         readonly IConfiguration configuration;
16
17         public HomeController(IConfiguration configuration)
18         {
19             this.configuration = configuration;
20         }
21
22         public IActionResult Index()
23         {
24             var kullanici = configuration["MailBilgileri:Adres"];
25             var sifre = kullanici ?? "mardinli3276@gmail.com";
26             return View();
27         }
28     }
29 }
```

The screenshot shows the Visual Studio IDE with the 'secretmanager' project open. The 'HomeController.cs' file is the active tab, displaying the following C# code:

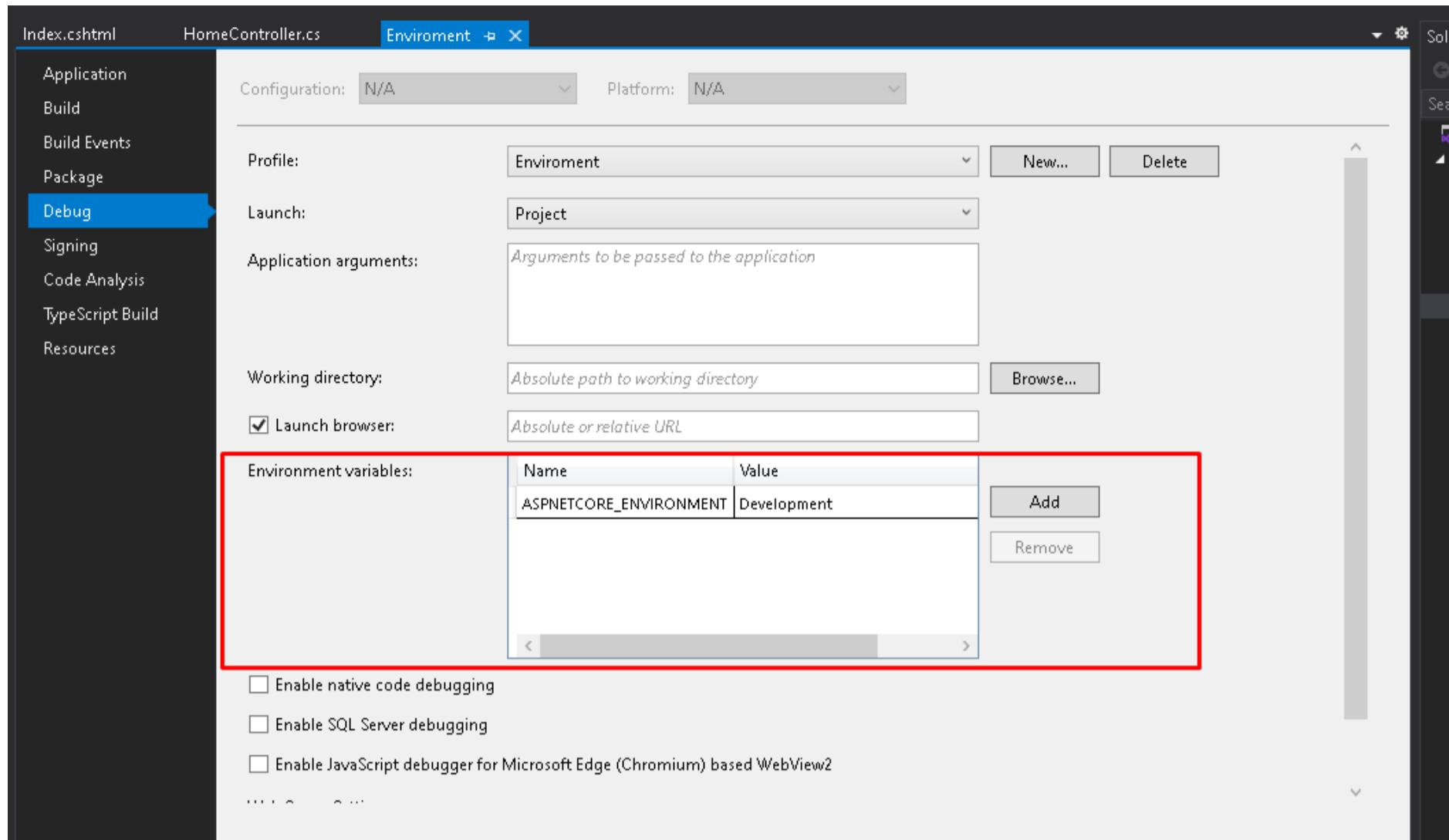
```
1  using Microsoft.AspNetCore.Mvc;
2  using Microsoft.Extensions.Configuration;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Threading.Tasks;
7
8  namespace secretmanager.Controllers
9  {
10    [Route("api/[controller]")]
11    [ApiController]
12
13    public class HomeController : Controller
14    {
15      readonly IConfiguration configuration;
16
17      public HomeController(IConfiguration configuration)
18      {
19        this.configuration = configuration;
20      }
21
22      public IActionResult Index()
23      {
24        var kullanici = configuration["MailBilgileri:Adres"];
25        var sifre = configuration["MailBilgileri:Sifre"];
26        return sifre "123456";
27      }
28    }
29 }
```

Environment

Environemnt Nedir? Bir web uygulamasında, uygulamanın bulunduğu aşamalara dayalı, davranışını kontrol etmek ve yönlendirmek isteyebiliriz. İşte bunun Environment dediğimiz değişkenler mevcuttur.

[Environment Variables](#) Asp.NET Core uygulamalarının runtime'da ki davranışını belirlememizi sağlayan değişkenlerdir.

[ASPNETCORE_ENVIRONMENT](#) Nedir?



IWebHostEnvironment Arayüzü ile Runtime Environment Ortamına Erişim :

The screenshot shows the 'HomeController.cs' file in the Visual Studio code editor. The code defines a HomeController class with an injected IWebHostEnvironment dependency. It includes a constructor that initializes _webHostEnvironment and an Index action method. The action method checks the environment using IsDevelopment(), IsProduction(), and IsStaging() methods. A conditional comment at the end of the method body is intended to check for a custom environment named 'Deneme'. A red box highlights the injection of IWebHostEnvironment and its usage in the Index action method.

```
Index.cshtml      HomeController.cs  Enviroment
Index.cshtml      HomeController.cs  Enviroment
13
14  public class HomeController : Controller
15  {
16      IWebHostEnvironment _webHostEnvironment;
17
18      public HomeController(IWebHostEnvironment webHostEnvironment)
19      {
20          _webHostEnvironment = webHostEnvironment;
21      }
22
23
24      public IActionResult Index()
25      {
26          if (_webHostEnvironment.IsDevelopment())
27          {
28              ViewBag.env = "Development";
29          }
29          else if (_webHostEnvironment.IsProduction())
30          {
31              ViewBag.env = "Production";
32          }
32          else if (_webHostEnvironment.IsStaging())
33          {
34              ViewBag.env = "Staging";
35          }
35
36          _webHostEnvironment.IsDevelopment();
37          _webHostEnvironment.IsProduction();
38          _webHostEnvironment.IsStaging(); //test aşaması ise
39          // _webHostEnvironment.IsEnvironment("Deneme"); //Kendi özel environmentimiz varsa onu yazıyoruz
40
41
42      }
43
44      return View();
45  }
```

Home Page - Enviroment

Enviroment Home Privacy

Welcome Development

Learn about [building Web apps with ASP.NET Core.](#)

Environment Değişklerin secrets.json ve appsettings.json Dosyalarını Ezmesi :
Aynı isimde değerler olduğunda öncelik Enviroment'de olduğu için Enviroment'deki değer gelir. Mesela Enviroment'de (Location) diye bir ad olsun secrets.json veya appsetting.json 'da da (Location) diye bir ad olsun , Direk Enviroment'deki (Location)' a karşılık gelen değeri alır.

Name	Value
ASPNETCORE_ENVIRONMENT	Development
a	A Enviroment

```
1  {
2    "a": "A secret"
3  }
```

The screenshot shows the Visual Studio code editor with the file 'appsettings.json' open. The file contains JSON configuration for logging and environment settings. A specific entry, 'a': "A appsetting", is highlighted with a red box.

```
1  {
2    "Logging": {
3      "LogLevel": {
4        "Default": "Information",
5        "Microsoft": "Warning",
6        "Microsoft.Hosting.Lifetime": "Information"
7      }
8    },
9    "AllowedHosts": "*",
10   "a": "A appsetting"
11 }
12
```

The screenshot shows the Visual Studio code editor with the file 'HomeController.cs' open. The code defines a controller with an Index action. In the action, it retrieves a configuration setting 'a' and checks its value against 'A Enviroment'. The code block is highlighted with a red box.

```
14  {
15    public class HomeController : Controller
16    {
17      IWebHostEnvironment _webHostEnvironment;
18      IConfiguration _configuration;
19
20      public HomeController(IWebHostEnvironment webHostEnvironment, IConfiguration configuration )
21      {
22        _webHostEnvironment = webHostEnvironment;
23        _configuration = configuration;
24      }
25
26      public IActionResult Index()
27      {
28        var a = _configuration["a"];
29        if (a == "A Enviroment")
30        {
31          if (_webHostEnvironment.IsDevelopment())
32          {
33            ViewBag.env = "Development";
34          }
35          else if (_webHostEnvironment.IsProduction())
36          {
37            ViewBag.env = "Production";
38          }
39          else if (_webHostEnvironment.IsStaging())
40          {
41            ViewBag.env = "Staging";
42          }
43        }
44      }
45    }
46 }
```

.cshtml'de Environment Kontrolü :

The screenshot shows the Visual Studio code editor with the file 'Index.cshtml' open. The page contains standard HTML and Razor syntax. It includes environment control logic using the '@environment' directive. A specific section of the code, which uses '@environment' tags to conditionally render content based on the environment, is highlighted with a red box.

```
1  @{
2    ViewData["Title"] = "Home Page";
3  }
4
5  <div class="text-center">
6    <h1 class="display-4">Welcome</h1>
7    <p>Learn about <a href="https://docs.microsoft.com/aspnet/core">building Web apps with</a></p>
8  </div>
9  <h1>Ortam: @ViewBag.env</h1>
10 <environment names="Development">
11   Development Ortamındayız.
12 </environment>
13 <environment names="Production">
14   Production Ortamındayız.
15 </environment>
16 <environment names="@ViewBag.env">
17   @ViewBag.env Ortamındayız.
18 </environment>
19
```



Welcome

Learn about [building Web apps with ASP.NET Core.](#)

Ortam: Development

Development Ortamındayız. Development Ortamındayız.