# 1.) MVC :

* Bu dersimizde MVC Mimarisini anlayalım.
* MVC bilinenin aksine ASP.NET MVC ile ortaya çıkmış bir kavram değildir.
* MVC ASP.NET MVC den önce Java ve PHP gibi dillerde kullanılmaktaydı.
* MVC 3 Kavramdan oluşur

1. **Model :** ilgili modellerimizi temsil eder.
2. **View :** ilgili kullanıcının gördüğü kısmı temsil eder. (web sayfalarını)
3. **Controller :** Viewleri kontrol eden bir yapıdır.

* Bir senaryo üzerinde konuşalım
* bir veri kaynağından bir veri alındıktan sonra model e bind edilir. Controller aracılı ile ilgili view e veri aktarılır. (controller içerisindeki action methodları aracılığı ile gerçekleşir. )

# 2.) Projeye MVC’nin Eklenmesi

* Bu dersimizde uygulamamıza MVC’yi ekleyelim.
* Uygulamamıza MVC’yi ekleyebilmemiz için yani biz birşeyleri ekleyebilmemiz için ne yapıyorduk. ConfigureServices içerisine ekliyorduk.

1. **Services.AddControllerWithViews();**

* İlgili middlewarelerimizi ekliyorduk

1. **App.UseRouting();**
2. **App.UseEndpoints();**

* Burada endpoint konfigürasyonumuzu gerçekleştiriyoruz. **IEndpointRouteBuilder** interface i alana bir **action** delegate istemektedir.

1. **App.UseEndpoint(e => e.mapcontrollerroute());** eklenecektir.

* Dipnot olarak kısa bir hatırlatma yapacağız name i herhangi bir name olabilir, patten belirlenirken şu esas alınacaktır.
* MVC mimarisinde bir controller bir de view olduğunu biliyoruz. Controller ne yapıyordu ilgili controller klasöründeki viewlerimizi yönetiyordu. Home controller varsa views içerisindeki home klasöründeki viewleri kontrol ediyor. Controller içerisine action methodları var (index) ilgili index ne yapıyordu index ismindeki view i getiriyordu. Actionlar viewleri kontrol etmektedir.
* O zaman ben öyle bir endpoint yapmalıyım ki / Home dediğimde buradaki Home controller / index dediğimde ise burası index action ina karşılık gelmelidir.
* Sen böyle bir url geldiğinde ilk /(slash) ı Controller olarak algıla 2. /(slash) i action olarak algıla. (action methodlarının adı ile viewleri arayacak ve yönetecektir ayriyetten isimlendirme yapmazsan index methodu index viewini arar yavuz action u yavuz cshtml i aramaktadır. )

1. Bu saatten sonra MVC klasörlerinizi ekleyebilirsiniz.

* **Bu işlem sonrasında bizler controller ve views oluşturduktan sonra bizler ilgili controller içerisindeki aciton a istek atarsak viewlerimiz iki path üzerinde aranacaktır (Home / index url i için)**

1. **Views/Home/Index.cshtml**
2. **Views/Shared/Index.cshtml (ya controller ya shared de ararız. Tüm controller ve componentler için shared ortaktır. )**

* Bu aşamadan sonra dikkat edilecek ince detay şu diyelimki bizim home alıtnda index isminde bir view imiz var ve index view i aynı anda shared içeriisnde de var **(home / index için hem home klasöründe hem Shared klasöründe viewlerimiz var her zaman ilk home çalışır Shared a bakmaz. Eger home altında bulamazsa o zaman shared a bakar.**)

# 3.) Default Routing :

* Mapcontroller route içerisinde bir parametredir. Defaults isimsiz bir obje notasyonu alabilmektedir. Gelen endpoint isteğinde Controller boş ise sen bunu controllerlar içerisinden home controller olarak algıla, action boş ise gidip index i çalıştır. / şöyle bir url gelirse git sen şunları parametre olarak al manasına gelir.

# 4.) Razor Syntax :

* Bu dersimizde Razor operatörünü göreceğiz. Bizler bir view de bir .cshtml dosyası içerisinde bir C# kodu yazmak istersek o zaman kullandığımız bir operatördür.
* Razor operatörü için @ işareti kullanmak yeterlidir.
* **Tek satırlık** bir c# kodu yazacaksak **@** , **çok satırlık** bir kod yazacak isek **@{}** şeklinde devam edebiliriz.
* Tek satırlık için şu işlemler genellikle kullanılır :

1. **Ternary operatörü (ternary esasında çıktı veren bir yapıdır. ),**
2. **geri dönüş tipi olan fonkisyonlar (VOİD olmayan string, int vs olabilen),**
3. **sayfadaki model veya razor içerisindeki tanımlanan değişkenleri ekrana basabilmek için değişken çağırma,**
4. **aritmetik işlemler yada bool gibi sonuç çeviren parantezli işlemler**

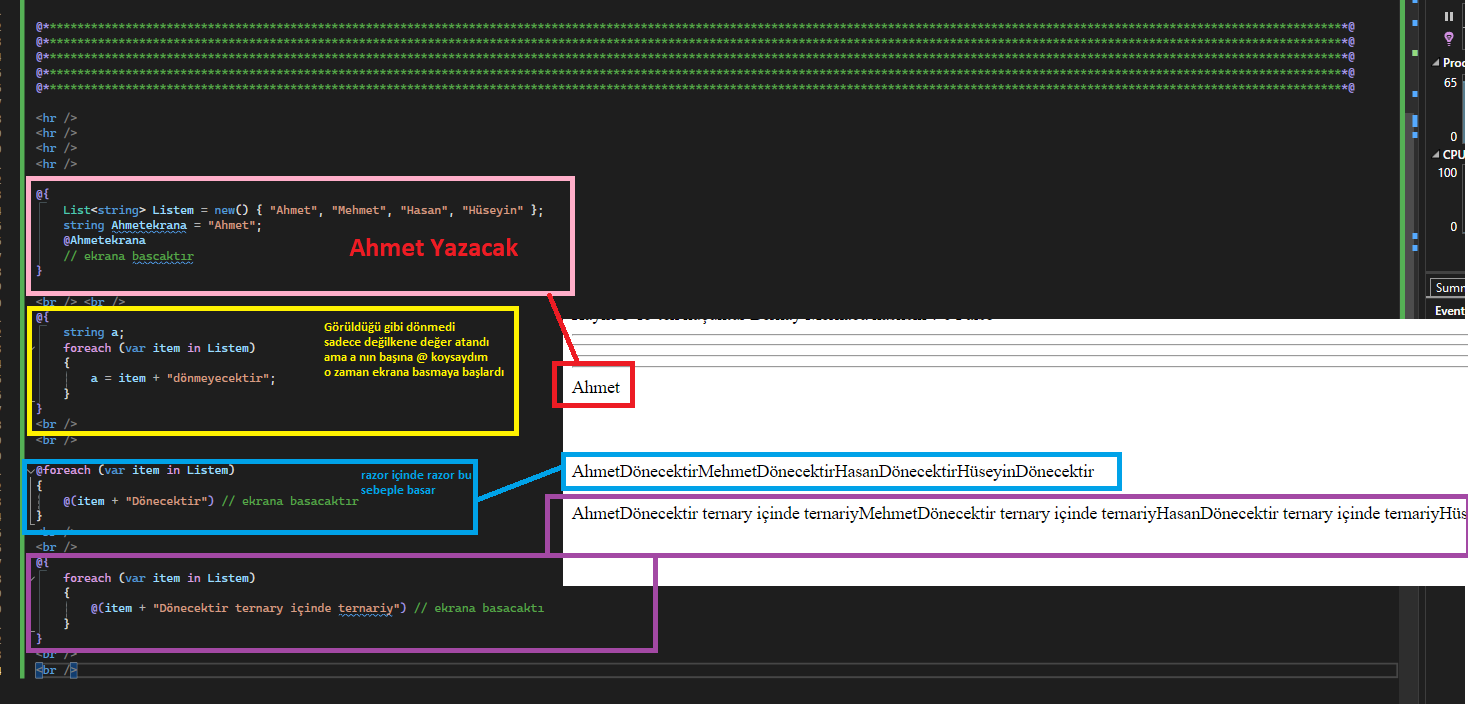
* Tek satırlık işlemler yukarıdaki sıralamadan da anlaşılacağı üzere ekrana bir çıktı vermesi gereken kodlardır. (çıktı zorunlu değer dönmek zorunlu ).
* Çoklu satırda ise istediğin şeyleri kullanabilirsin.

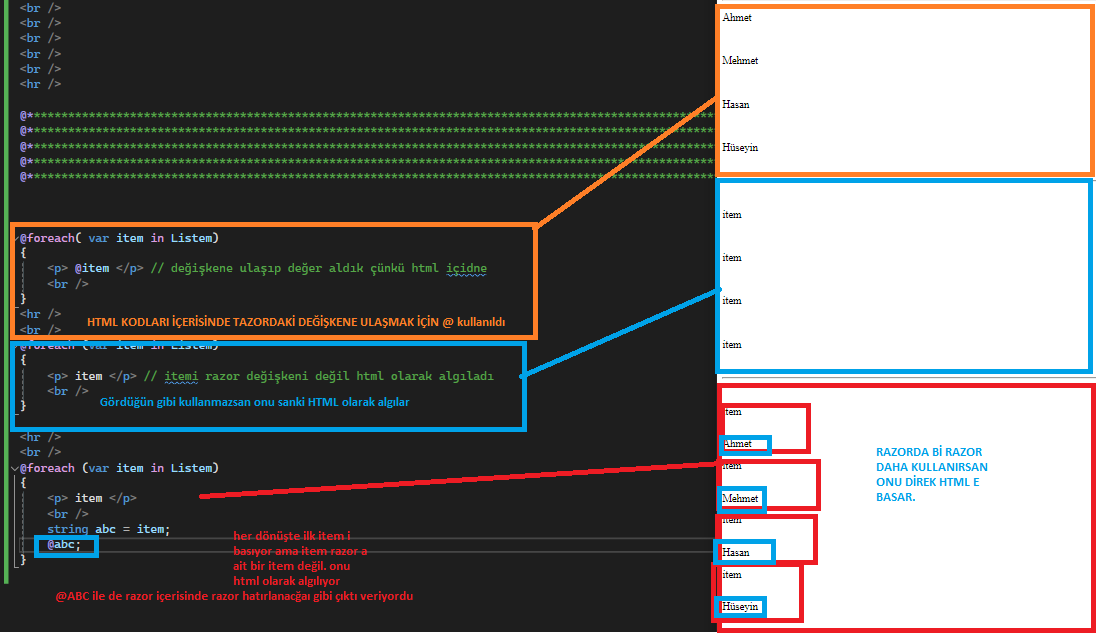
ekran görüntüsü, metin, yazılım, ekran, görüntüleme içeren bir resim

Açıklama otomatik olarak oluşturuldu

* Neticesinde eklemek istediğim bir şey var misal ternary içerisinde ternay kullanırsan 2.ternary i direk ekrana basacaktır.

* Razor içerisinde HTML Yazarsan tekrardan değişkene ulaşmak istersen razor kullanabilirsin o zaman HTML kodu olarak algılamaz razor kullanmaszan tüm değişkenleri sanki HTML kodu gibi olur
* HTML kodunun üstünde razor aktif iken değişkenlere ulaşmak için Razor kullanmana ihtiyaç yok o zaman aktif.

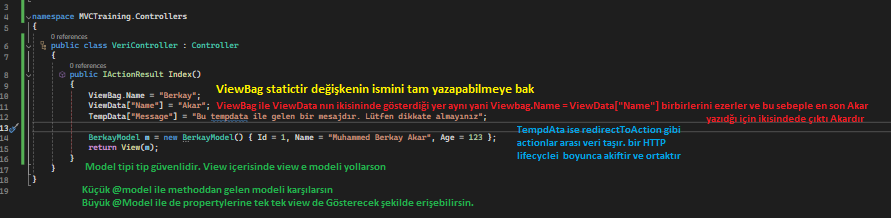


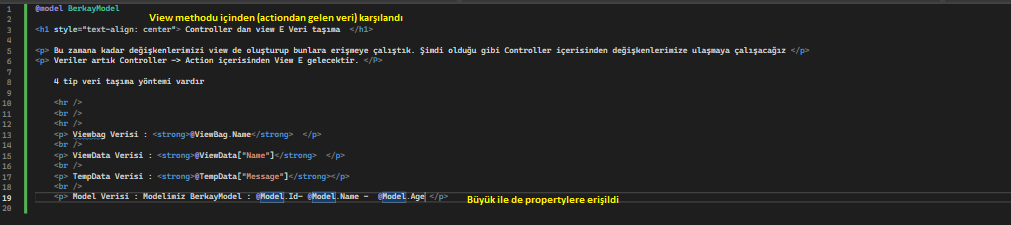


# 5.) Controller Üzerinden View e Veri Taşımak :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Taşıma Yöntemleri | ViewBag | ViewData | TempData | Model |
| Controller Tarafı | ViewBag.Name = “Berkay AKAR”; | ViewData[“Name”] = “Berkay AKAR”; | TempData[“Name”] = “Berkay AKAR”; | Önce bir Class ile model oluştur  Örnek al  View in içerisine modelini ver  Costumer c = new Costumer {id = 1 , name = “Berkay”};  Return View(c); |
| View tarafı | @ViewBag.Name | @ViewData[“Name”] | @TempData[“Name”] | @model Costumer  @Model.Id  @Model.Name |

* ViewBag : Dinamik bir Veri tipidir. İsmini doğru yazmanız gerekecektir.
* ViewData[“”] : ViewBag ile ViewData aynı değişkendir. ViewBag.Name de bir değişken tanımlasan ve ViewData[“Name”] ile tekrar bir değişken tanımlasan ikiside aynı olmaktadır. Sözlük gibi erişim sağlayabilirsin
* TempData[“”] : Actionlar arası veri taşımanı sağlayacaktır. Aralarındaki en büyük fark bu. Yani redirecttoaction ile bir aktarım yapsan yine değişken son etapta saklanacaktır. Bir HTTPRequest lifecyclie i boyunca ayakta kalacaktır.
* Model : Tip güvenlidir. Models klasörü içerisinde modellerinizi tanımlayabilirsiniz. View Methodu içerisine örnek modelinizi koyun View sayfasına gelin ve küçük model ile karşılayın @model Costumer gibi. Küçük model (@model) ile actiondan gelen veri karşılanır büyük Model (@Model.Id) ile propertylere erişim sağlanabilmektedir.





metin, yazılım, web sayfası, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

* Değişkenleri Action dan View e taşıyacaz. Controller içerisindeki Action dan View e veri taşınabilir.

TempData (bir makale okundu )

TempData’nın çok kısa süren bir instance’ı olması amaçlanmıştır ve bunu yalnızca mevcut ve sonraki isteklerde kullanabilirsiniz! TempData bu şekilde çalıştığından dolayı bir sonraki isteğin ne olacağını kesin olarak bilmeniz gerekir ve bunu garanti edebilmenin tek zamanı başka bir view’a yönlendirmenizdir. Bu nedenle, TempData’nın güvenilir şekilde çalışacağı tek senaryo yeniden yönlendirme yaptığınız zamandır. Çünkü bir yönlendirme(redirect) mevcut isteği(request) öldürür (ve 302 HTTP durum kodunu gönderir yani nesnenin istemciye(client) taşınması), sonra yeniden yönlendirilen view’a hizmet etmek için sunucuda yeni bir istek oluşturur. Yukarıdaki HomeController örneğine baktığımızda TempData nesnesi beklenenden farklı sonuçlar verebilir çünkü bir sonraki isteğin kaynağı garanti edilememiştir. Örneğin, bir sonraki istek tamamen farklı bir makineden ya da tarayıcıdan kaynaklanabilir. Aşağıda açıklandığı gibi, TempData’yı kullanmak için kullanılan söz dizimi(syntax) ViewData ile aynıdır.

CHAT GPT:

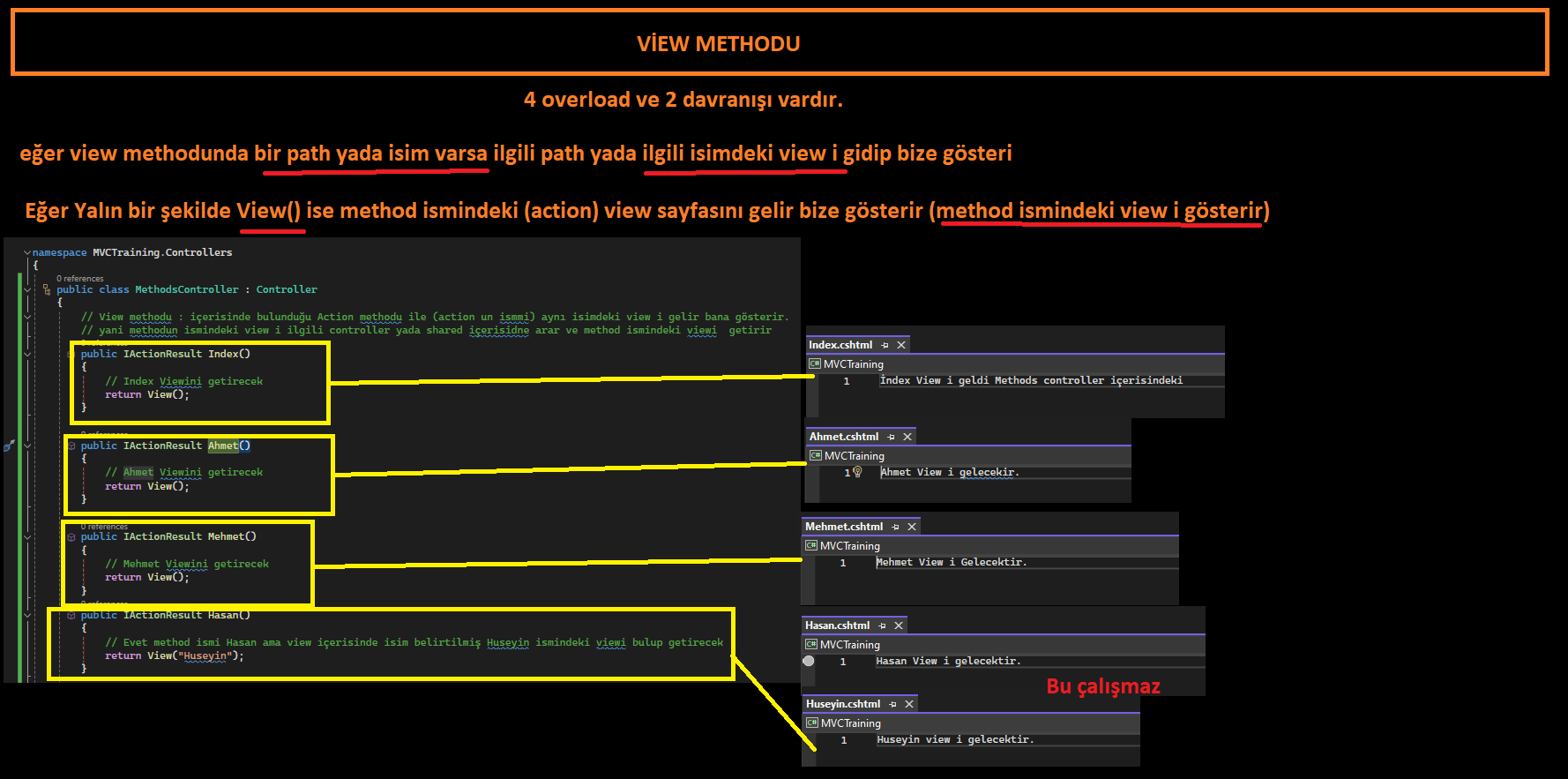
Evet, doğru bir gözlem. RedirectToAction kullanıldığında, bir Controller metodu içindeki bir HTTP isteği tamamlanmış ve tarayıcıya yönlendirme başlatılmış olur. Ancak, tarayıcı yeni bir HTTP isteği başlatır ve hedef Controller metodu çalıştırılır. İşte bu noktada TempData devreye girer. TempData, bir HTTP isteği ile bir sonraki HTTP isteğine veri taşımak için kullanılır. Yani, bir Controller metodu içinde TempData'ya bir değer atandığında, bu değer tarayıcıya yönlendirme yapıldığında diğer Controller metodu tarafından okunabilir.

# 6.) View Methodu :

View Methodu : İçerisinde bulunduğu Action Methodu ile Aynı isimdeki View i gelip bize gösterir.

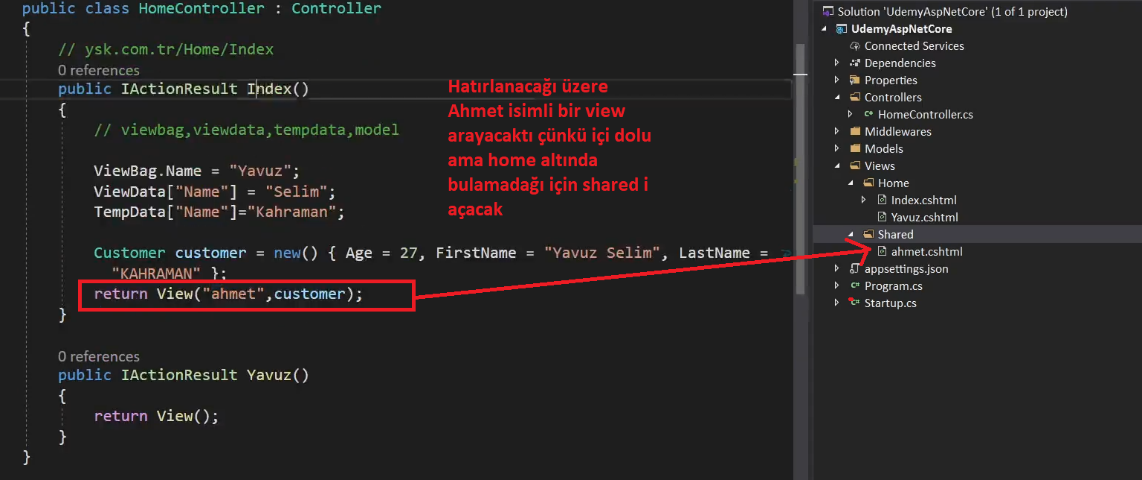
4 adet Overload ı mevcuttur.

1. Sade hali View () : içerisinde bulunduğu – çağrıldığı action methodu ismi ile aynı isimdeki viewi gelir gösterir.
2. View(string name ) : içerisine verdiğin isimdeki viewi gelir gösterir. İçerisine ya path yazarsın yada view ismi verirsin o view ismini baz alarak ilgili view i gösterir. Action ismini baz almaz.
3. View (obj model) : action ismindeki view e gider model yollar
4. View (objmodel, viewname) : belirtmiş olduğun isimdeki view e ilgili modeli yollar



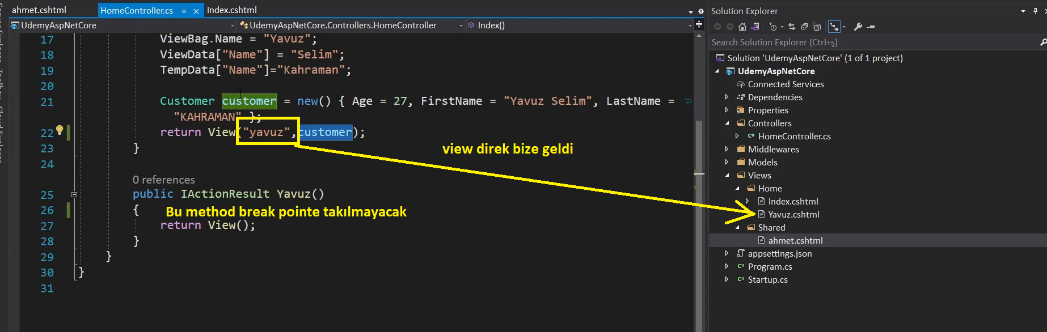
# 7.) RedirectToAction Methodu :

View methodu çalıştırdığımız zaman. View gidip ilgili controller klasörünün içerisinde ilgili isimle aramaa yapıyordu. Bulamazsa shared a bakıp ilgili cshtml dosyası geliyordu.

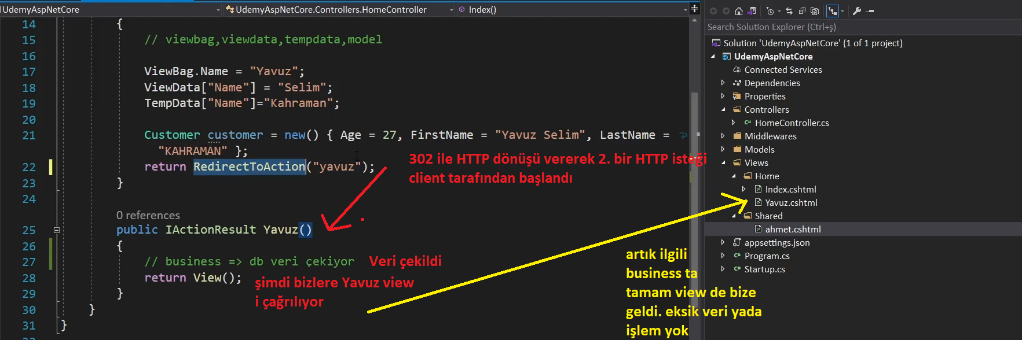


Burda senaryomuz çalışırken şöyle bir sıkıntımız var. View methodu view i ilgili action ı çalıştırmadan doğrudan çalıştırıyor.

**Yani ahmet isimli viewimizin bir action methodu olsaydı o action methodu çalışmadan doğrudan bu view sayfası bize gözükecekti – açılacaktı deneme amaçlı olarak aşağıdaki örneği yapalım.**



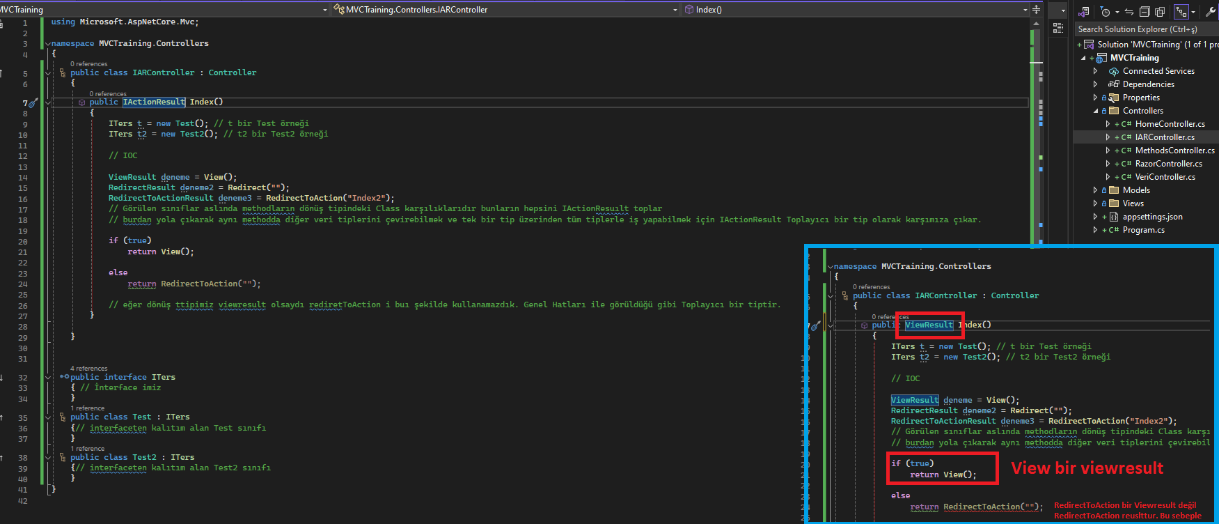
* İlgili debuga düşmöeden yavuz methodu çalışmadan view açıldı. Yavuz içerisinde bir business işlemi yapıyor olabilirm. Veri çekiyor yada business işlemi yapıyor olabilirim. Bu sebeple direk view i çalıştırdığımızda bizler esasında bu aşamaları atlamış oluyoruz .bu sebeple bizler ilgili methodu çalıtşırabilmemiz için gidip bu işlemi yapıyoruz. RedirectToAction ile ilgili action methodunu çalıştırıp doğrudan devam edebilirsininz.



* overloadlar ile ilgili areas, controller, action yönlendirmeleri ve yönlendirme ile birlikte modelde yollayabilirsiniz. 8 overload i mevcuttur.

# 8.) IActionResult İnterface i :

Bu dersimizde **IActionResult** konusunu inceleyelim. Öncelikle bilmemiz gereken şey bir **interface** **örneklenmez** ama bir interface **kendisinden kalıtılan nesne (class) ın örneğini taşıyabilir**. Burdan yola çıkarak şunu söyleyebiliriz.



Dönen methodlardaki değerlerin hepsi IActionResult ile karşılanıyor görülüdğü gibi ilgili nesnelerin hepsi esasında Itest gibi bir IActionResult olarak adlandırılır.

# 9.) Routing Konusu:

Bu dersimizde Routing Kavramını inceleyelim. Startup a bir dönelim

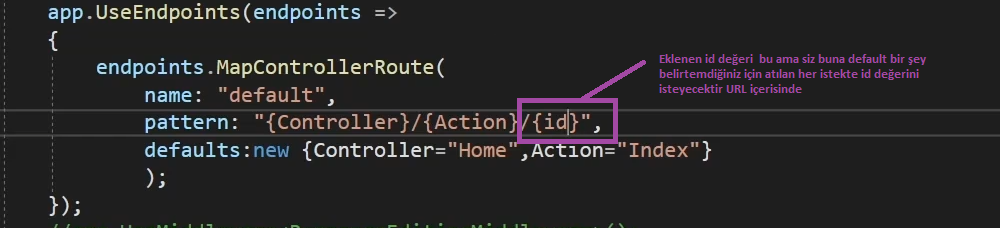
metin, ekran görüntüsü, yazılım, ekran, görüntüleme içeren bir resim

Açıklama otomatik olarak oluşturuldu

Endpoints.MapControllerRoute methodu aracılığı ile belirtmiş olduğumuz pattern a bakarak hareket ediyorduk. Eğer herhangi bir şey gelmezse **(/ diye sadece böyle bir URL gelirse)** sen default-direk Controller lardan Home olarak algıla, Actionlardan ise Index olarak algıla.

**Peki ya şöyle senaryolarımız olsaydı ?** Home / İndex / 1 şeklinde bir senaryom olsaydı? **şöyle düşünün Product / Detail / 1** ürünlerimden **1 numaralı ürün için detay sayfasını aç gibi.** 1 numaralı id ye sahip olan product i görmek istiyorum gibi. Bu yapıyı karşılayacak herhangi bir route yazmadık. O yüzden 3 / lı bir url i karşılamayacaktır şuanki yazılı olan Route (pattern sebebi ile). Home / Index / 1 şeklinde bir istek atarsak **404** şeklinde bir hata atacaktır. **Buna uygun bir Route yazmamız gerekecektir.**

Peki ben bu id değerini nasıl taşırım ? Pattern e bir adet id eklerim (Query ile de yollanabilir ama parametre olarak aktarmak istiyoruz şuan öyle düşün Query için böyle bir şeye ihtiyaç yok.)



(paramtere içierisinde istediğin değerleri / lar ile verebilirsin spesifik bir isim olmadığı için etkileymeyecektir MVC yapısını / Akar / Ahmet vs diye de gidebilirsin.)

Buradaki id değerini opsiyonel yapabilmek için bir soru işareti koyabiliriz. Artık id değeri vermeden bizlere bir hata mesajı dönmeyecektir (default bir id değeri belirtmedik çünkü).

İd: int dersen id değeri sadece integer değer alabilir manasına gelmektedir. string yazarsan hata atar. Pattern eşleşmez. İd:alpha yazarsanız id değerini sadece harfler ile alabileceğiniz manasına gelmektedir. Türkçe harflere izin verilmez.

Aynı şekilde burada birden fazla pattern belirtebililirsiniz. (Özel Route lar da belirleyebilirsiniz.)

* url product ile başlarsa ve bu product in pattern i gelen url in patternine uyarsa artık buradaki url yapılanması çalışacaktır. Misal burada product dedik ama controller içerisine home dediğimiz için aslında Product/index dediğimizde Home/index e gidecek. Bu sayede product isimli bir controller yazmadan direk bu şekilde çalıştırabiliyoruz.
* Dikkat edilmesi gereken şey ise bizlerin bir http isteği geldiğinde URL i yukarıdan aşağı okuyarak ilk eşleşen veriye göre çalıştırmasıdır. Bu yüzden sen eğer alttaki Controller / Action / id li route u üste koyarsan sistemde product isimli controller arayacaktır. Ama sen product li url i üste koyarsan direk product ile eşleşeceği için bu şekilde bir işlem gerçekleşecektir.
* Sen burada product ile başlayan bir şey gördüğünde be tek slash ile iki bölümlüyse buradaki endpoint patternini baz al. Controller olarak ise Home çalışsın diyebiliriz. (product yavuza döndü) sen Product/ındex dersen artık Home altındaki İndex action u çalışacaktır.

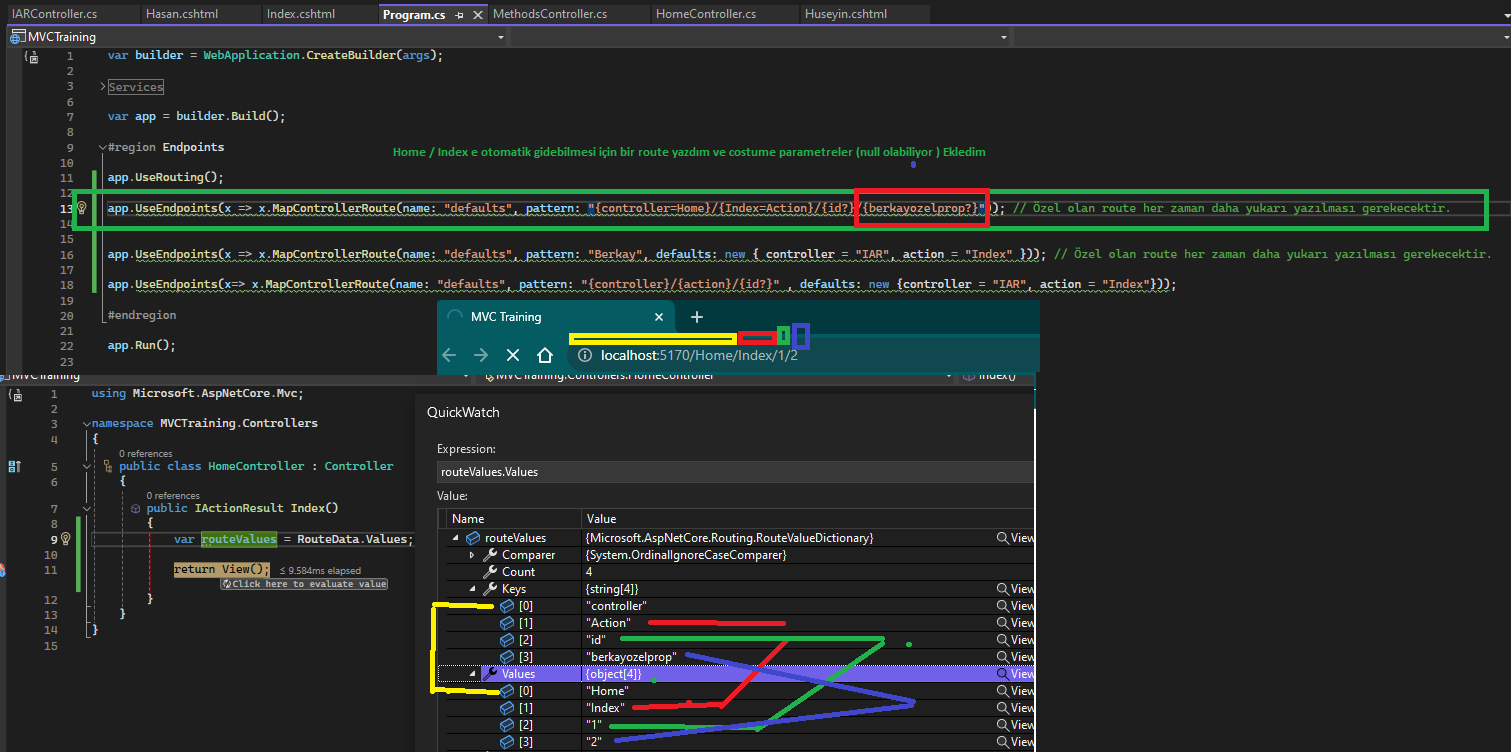
metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

* Daha özel olan route her zaman daha yukarı yazılır. Gelen route yukarıdan aşağı bakarak ilk eşleşen ile çalışacaktır.

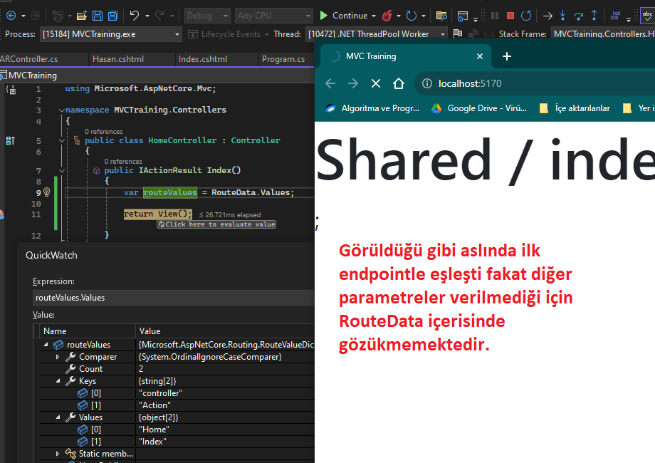
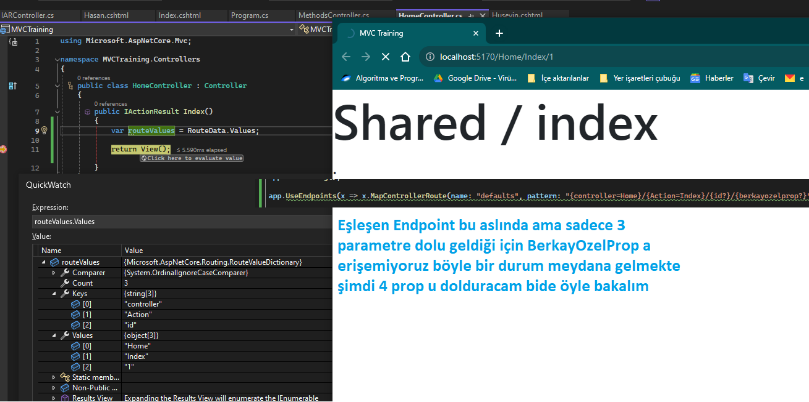
# 10.) RouteData Property:

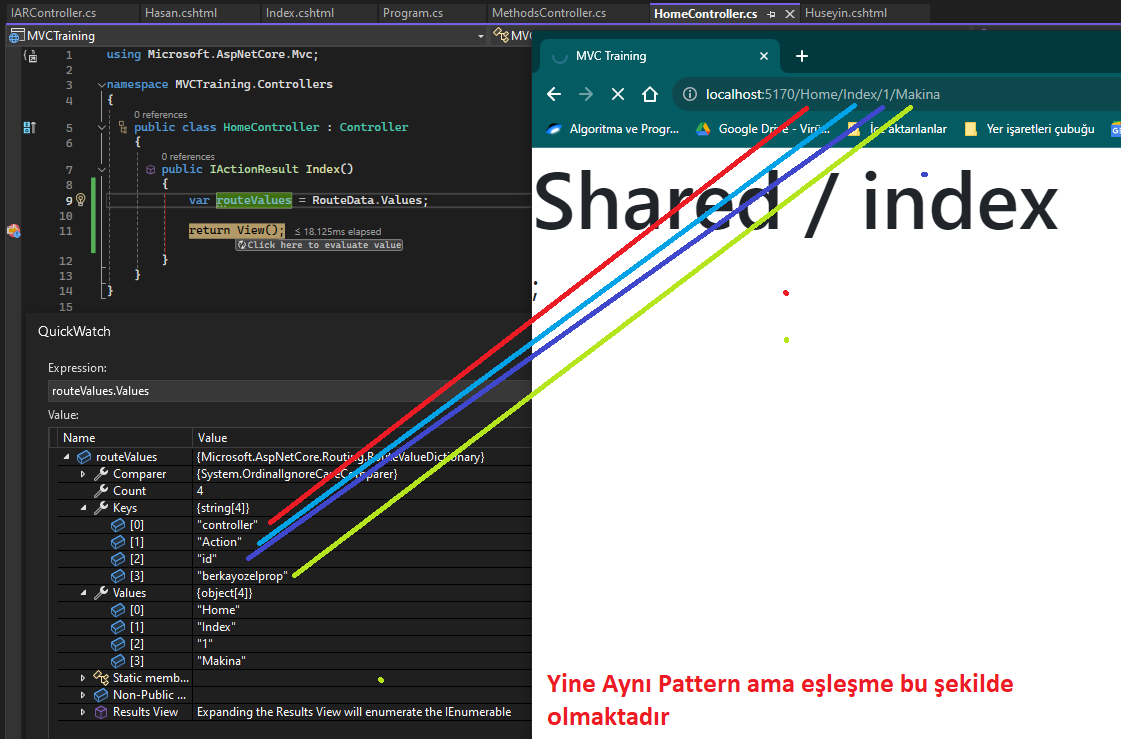
Bir önceki derste id değeri nin route üzerinden nasıl alınabileceğini konuştuk. Bu başlık altında route içerisindeki id değerinin nasıl yakalanabileceği hakkında konuşalım.



Bu dersimizde URL deki İd değerinin nasıl yakalanabileceği hakkında konuşalım. **Asp.Net Core** da özel bir property vardır. Bu property **RouteData** propertysidir. **Bu property ile sizler bir Route u olduğu gibi ele alabilmektesiniz. Routedata ile bütün route u olduğu gibi ele alabiliyoruz.**

Sözlük mantığında çalışır Key - value değerleri vardır. Eşleşen Endpointteli patterni tek tek Key ve Value bazında alabilmektesiniz. İlgili endpointte görülüdüğ gibi opisyonel parametreler var id ve berkayozelprop gibi. Eğer siz onları vermezseniz Routedata içerisindeki propertylerde bu durum çıkmamaktadır.





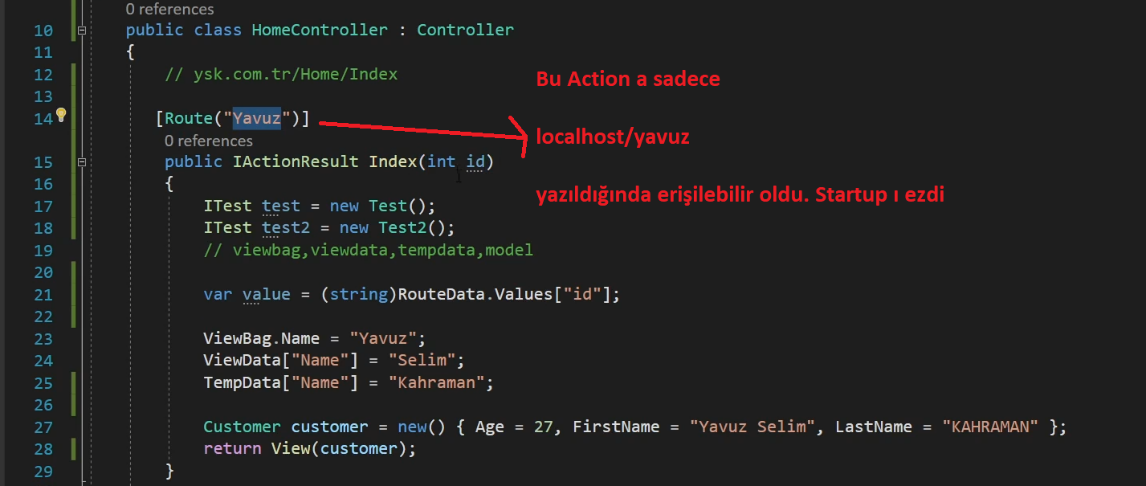
metin, ekran görüntüsü, multimedya yazılımı, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

# 11.) Attribute Routing :

Attribute bazında route kurmamızı sağlayacak olan yapıdır. Apilerde çok kullanıyoruz Asp.Net Core MVC tarafı içinde geçerlidir. Action bazında veya controller bazında yazılan Route her zaman startuptakini ezer yani Home / action için ben startup ta global bir route yazmışken Action methodu üzerine [**Route(“Ahmet”)]** yazarsam /ahmet dediğimde Home/index çalışacaktır. Home/Index yazarsam çalışmayacaktır.

Starupta değişkenler yani sabit olmayan ifadeler {} ile gösterilirken Route Attribute ü içerisinde ise [] şeklinde gösterilebilir ve ilgili [ahmet] ahmet değişkeni methodda model binding ile alınabilmektedir. Bu şekilde de ek bir bilgilendirme geçmek istedim yani **Route attribute ü startup i her zaman ezer.**



Artık direk / yazınca çalışmaz anca /yavuz ile çalışır.

ekran görüntüsü, metin, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

# 12.) wwwroot :

Sizler static dosyalarınızı asp.net mvc frameworkteki gibi bir content klasörü açıp doğrudan indirme işlemi yapamazsınız. Bunun sebebi **Asp Net core da varsayılanda Dosyalar (klasörler) Erişime (Dışarıya) KAPALIDIR.**

Bu sebeple sizler static dosyalarınız (resim, video, css, js) direk bir klasör üzerinden tanımlayıp viewde çağırsanız bile gözükmeyecektir. Asp.Net Core da bu işi **App.UsestaticFiles()** => wwwroot adında bir klasör erişimi dışarı açılır. isminde bir middleware yapabilmektedir. Bu middeware i middleware ler içerisine eklerseniz artık sizler **wwwroot ismindeki özel olan klasörü erişime (dışarıya) açabilir ve içerisine koyduğunuz statik öğeleri view içerisinden çağırdığınızda görebilirsiniz**

**Wwwroot isminde bir klasör açıldı içine resim kondu ve viewden çağrılııyor (middleware eklenmedi)**

metin, yazılım, multimedya yazılımı, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, multimedya yazılımı, yazılım, yazı tipi içeren bir resim

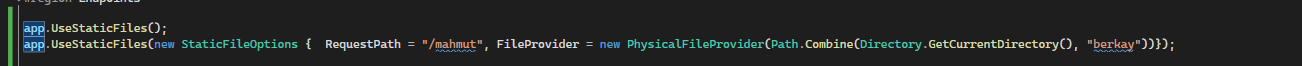
Açıklama otomatik olarak oluşturuldu

* Varsayılanda dosyalar (klasörler) erişime (dışarıya) kapalıydı şimdi app.UseStaticFiles ile birlikte biz bu klasörü komple dışarı açacağız. Middleware i ekleyince otomatik olarak geldi resim.

taşıt, araç, kara taşıtı, tekerlek, klasik araba içeren bir resim

Açıklama otomatik olarak oluşturuldu

* Bunun overlodlari var NewStaticFileOptions ile birlikte Request path ve file provider vererek costume yolunuzda ilgili aramaları gerçekleştirebilirsiniz.



metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturulduArtık view de ben /mahmut dediğimde aslında ilgili dosya Berkay isimli klasörde aranacaktır. **(fileprovider, request path iki özel propertydir. Request path istendiğinde /mahmut path i istendiğinde fileprovider ise hangi dosyayı açayım demektir. )**

metin, multimedya yazılımı, yazılım, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Aşağıda görüldüğü gibi ilgili /mahmut diyince fiziksel olarak Berkay klasörü içerisinde ilgili dosyayı aradı

metin, ekran görüntüsü, multimedya yazılımı, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

# 12.) NPM – Libman Konfigürasyonları :

* Npm ile bir dosya indirmek istiyorsanız npm configuration file isminde bir dosyayı projeye ekleyin. Sonrasında dev dependencies kısmına eklemek istediğiniz kütüphaneleri yazın ve kaydedin. Sonrasında ilgili konumda NPM i diyerek yüklemeyi gerçekleştirin. Aynı mantıkla da app.usestaticFiles ile new staticFilesoptions belirterek sürece devam edebilirsiniz.
* Libman da buna benzer. Yapmanız gereken şey sağ tık ile client libary ye gidip indirilecek konuma ilgili paketleri ve lokasyonu belirlemeniz olacaktır. Buda size gün sonunda bir json verir içerisinde paketleriniz vardır paketler bazında dosyalar vardır fiziksel yollarının olduğu bir konfigürasyon vardır.