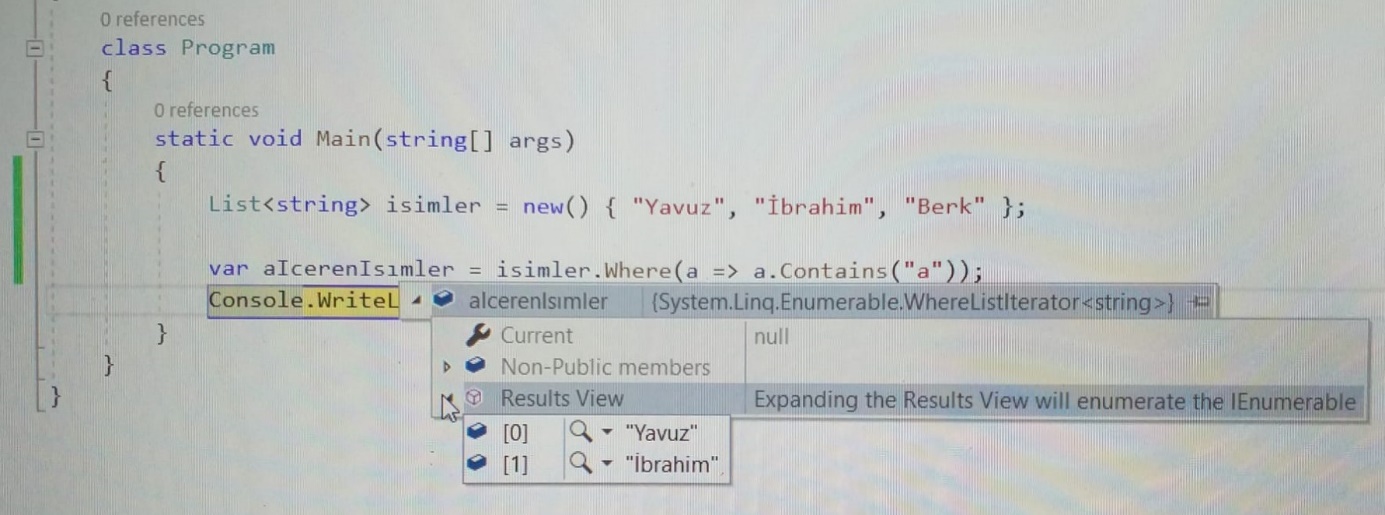
# **NOTLAR**

* Action Delege birçok değer alabiliyor, geriye bir değer döndürmüyor.
* Predicate Delege bir değer alıp sadece bool değer dönüyor.
* Func Delege ise birden fazla parametre alıp geriye herhangi bir tipten bir şey dönüyor. Mesela bir string bir integer değer alıp integer değer dönebilirsiniz. Veya bir müşteri nesnesi, ürün nesnesi alıp tekrar bir müşteri nesnesi dönebilirsiniz.

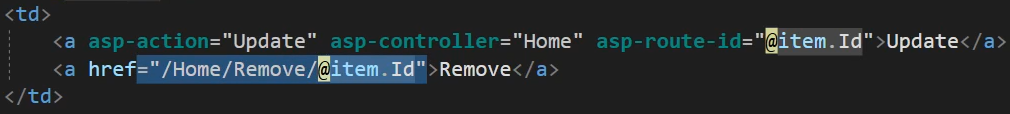


Burada görüldüğü üzere isimlerin olduğu string bir listeden içerisinde “a” harfi olan sonuçları önümüze getiriyor.

* Tag Helper’ ların özelliği bizi uzun uzun href vermekten kurtarıyor. Hreflerimizin olduğu sayfalarda sayfanın ilk satırına

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

kodunu yazdığımızda tüm TagHelperlar ekleniyor.



Görüldüğü üzere buttona verdiğimiz href linki yerine action adını, controllerını ve taşıyacağınız data değerini girince çok daha kolay olmaktadır. Biz id datası taşıdığımız için id dedik.

* TagHelperlar .NetCore’ a özeldir.
* Model Binding View’dan post ile gönderilen verilerin controller’ımızda bulunan action metot parametrelerine dönüştürülme işlemidir. Eğer bir action metotumuz parametre almış ise ve bir post işlemi kendisine geliyorsa post işlemi ilk olarak Model Bindera düşer. Model Binder gelen bu posttaki verileri bizim uygun olan modelimiz ile eşleştirme yapar. Böylece controller’da view’dan gönderdiğimiz verilere erişim sağlayabiliriz. Eğer istersek default model binding mekanizması yerine kendi model binding mekanizmamızı yazabiliriz.
* Views’ e sağ tıklayıp Add, New Item dedikten sonra Razor View Imports sayfası oluşturabiliriz. Buraya Tag Helpers’ ımızı ve usingimizi eklersek diğer sayfalarda her sayfanın başına using ve tag helpers kodlarını yazma zorunluluğumuz kalkar.

Örnek;

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

* DirectoryInfo classıyla bir pathteki klasörleri listeleyebilirsiniz.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

* FileInfo classından faydalanarak txt dosyasını oluşturarak içine yazı yazdık.
* Asp.Net Core dünyasında özel bir interface vardır ve siz bu interface aracılığıyla Asp.Net Core’ da ilgili dosyanızı upload edebilirsiniz. Bu interface’ in adı IFormFile’ dır.

Örnek;

public IActionResult Upload(IFormFile formFile)

{

return RedirectToAction(“Upload”);

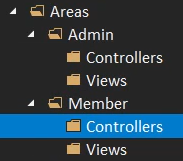
}

* FormFile’ daki CopyTo() metotuyla dosya upload işlemini gerçekleştirebiliriz.
* Abstract bir class örneklenemez.
* FileStream, Stream nesnesini kalıtsal yollarla alır.
* Guid.NewGuid() denerek benzersiz işlemler oluşturulabilir.
* View Componentler Asp.Net Core ile gelmiş yeni özellik.
* View Component’ in Partial View’ den farkı sanki bir controller gibi kendi kendine çalışmasıdır.
* View Componentler asenkronik bir şekilde çalışırlar. Yani çağırırken asenkronik olarak çağırılır.
* Bir View Component’ i kullanmak için await keywordünü kullanmamız gerekir. Çünkü asenkronik olarak çalışıyor.
* View Component’ i parametreli olarak ele alabilmek için Invoke() metotu içerisinde bir parametre yazmamız lazım.

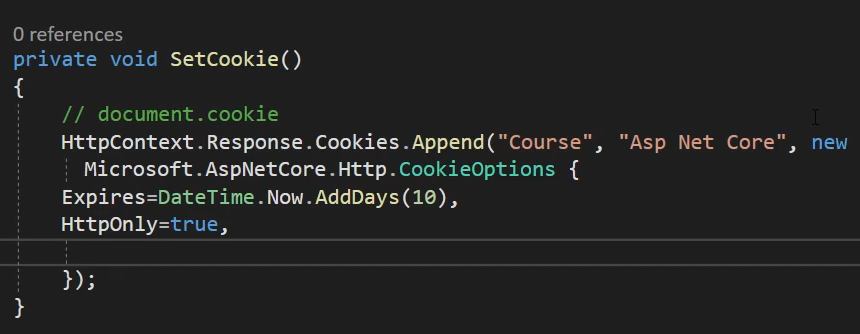
Örnek;



* Microsoft’ un bize sunduğu Tag Helperları kullanabileceğimiz gibi kendimiz de bir Tag Helper oluşturarak kullanabiliriz.
* Area alan anlamına gelir. Bir areada controllerlarımızı, modellerimizi ve viewlerimizi tutabiliyoruz. Veya parçalayabiliyoruz.



* Basit bir web sitesi yaparken bile eğer area kullanılmazsa, controller kısmı şişer, birçok controller olur. Aynı şekilde views kısmı da şişer ve uygulama yönetilmesi zor bir hale dönüşür.
* Bir validationun gerçekleşip gerçekleşmediğini İsValid ile kontrol ediyoruz.
* İlgili modele eklenecek özel bir validation yazabiliriz.
* Cookieler kullanıcıların tarayıcılarında veri tutmamızı sağlayan yapılardır.
* Bir cookieyi setleyebilmek için HttpContext’ in üstünde Response’ ta hareket edilir.



* Expires propertysi cookienin ne kadar süre ilgili kullanıcının (ilgili clientın) tarayıcısında ne kadar süre tutulacağını belirtir.
* HttpOnly ilgili document.cookie yazdığı zaman, yani javascript ile bu cookieyi çekmeye çalıştığı zaman siz HttpOnly’ i true olarak setlerseniz javascriptte ilgili cookienizi kapatırsınız. Yani document.cookie ile bu arkadaş sizin cookienize ulaşamaz.
* Bir cookieyi getirebilmek için de Request kullanılır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

* Session sunucu tarafında verilerimizi taşıyabildiğimiz yapılardır.
* Eğer uygulamada session kullanmak istiyorsanız Startup tarafına gidip app.UseRouting’ in altına app.UseSession, ConfigureServices kısmına da services.AddSession demeniz gerekir.

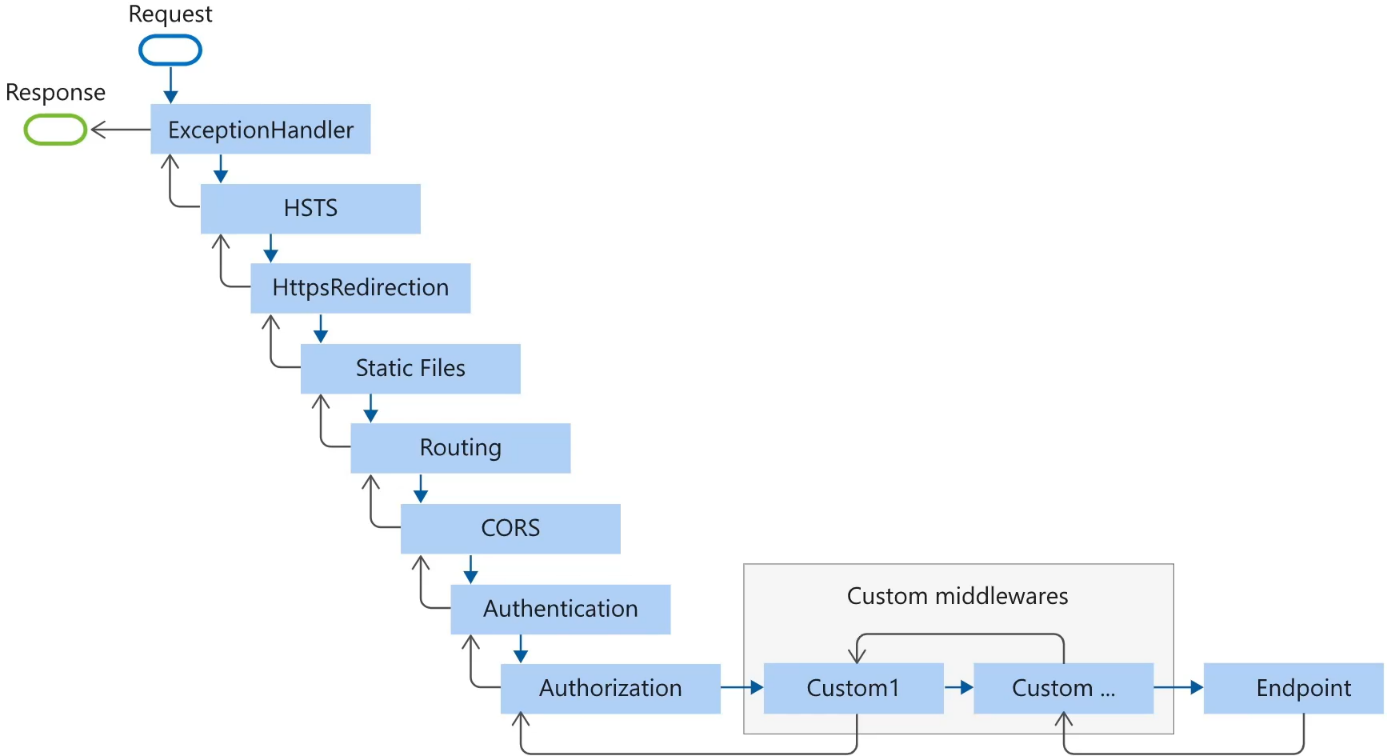
metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

* Bir filter ile parametrelere ulaşabilirim.
* Bir filterın filter olabilmesi için ActionFilterAttribute classından kalıtsal yollarla geçmesi lazım. (Bu abstract bir class)



* Use.Routing gibi kodlarımız da birer middleware’dir ve bu şekilde bir sıralaması vardır.
* ExceptionHandler: Hata yakalama.
* Uygulamada oluşan hataları takip edebilmek için loglama yapmak şarttır.
* appsetting.json dosyasının içerisinde bir veri tutup bu verileri çekebiliyoruz.