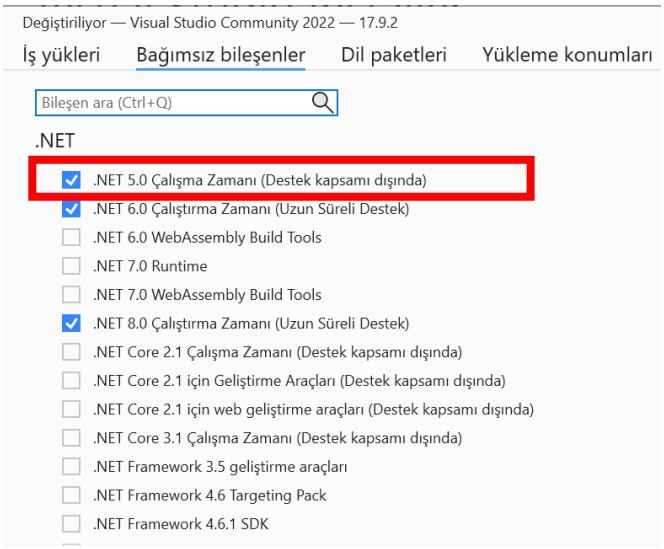
ANKARA
HACI BAYRAM VELİ ÜNİVERSİTESİ
UZAKTAN EĞİTİM UYGULAMA VE ARAŞTIRMA MERKEZİ

İNTERNET PROGRAMCILIĞI

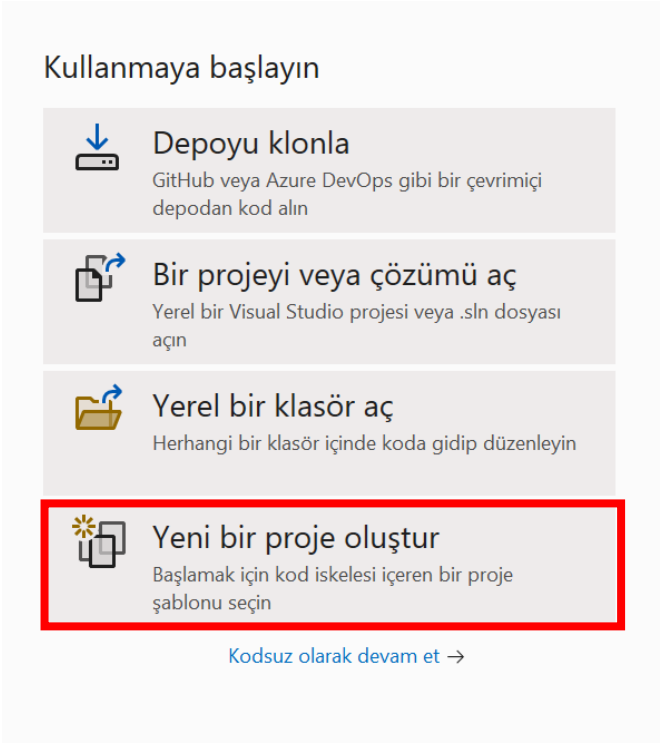
Bu bölümde MVC Yapısı Proje Üzerinde Oluşturulmuş, Default Routıng ve ViewBag Kullanımı uygulaması yapılmıştır.

1. MVC YAPISININ OLUŞTURULMASI

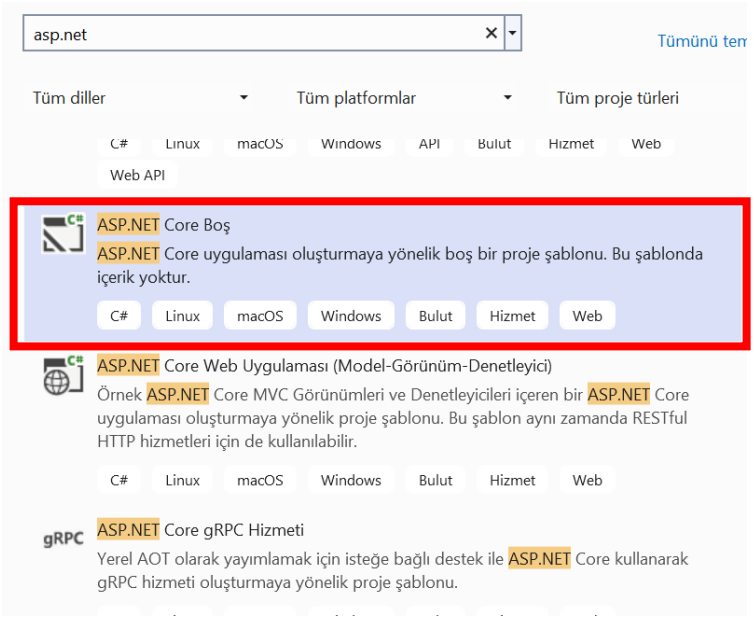
ADIM 1: Bağımsız bileşenler .net 5.0 seçilir.



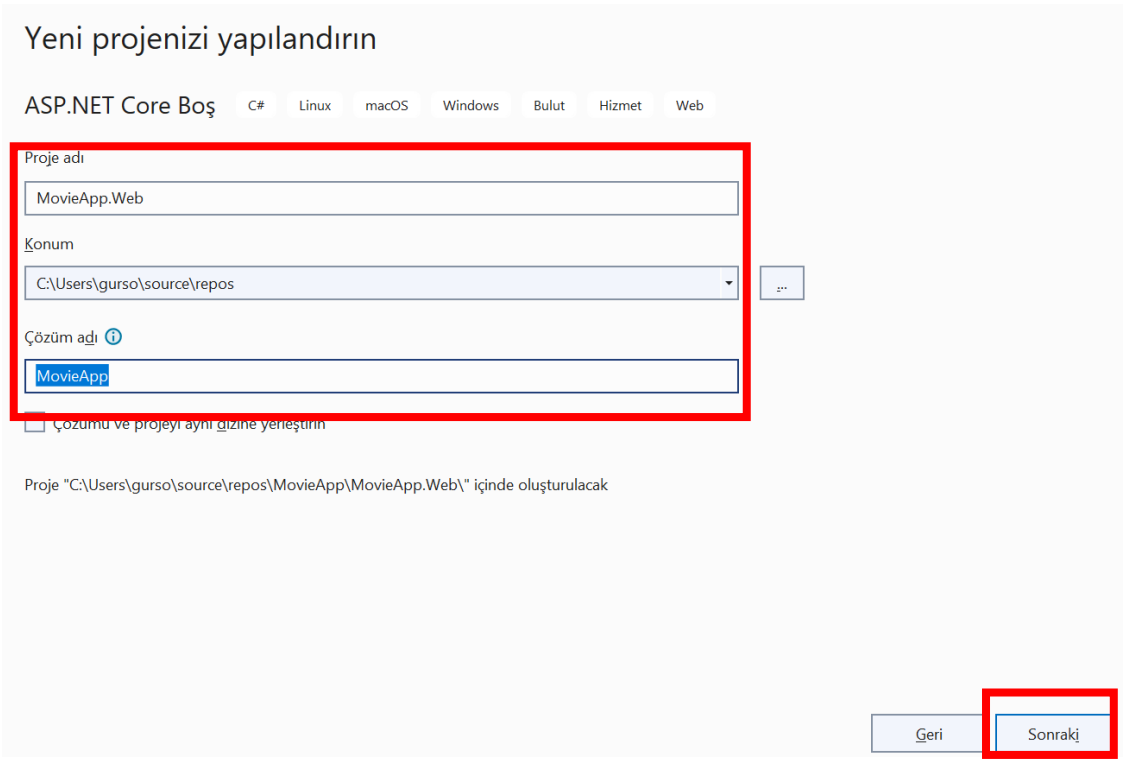
ADIM 2: Proje oluşturulur.



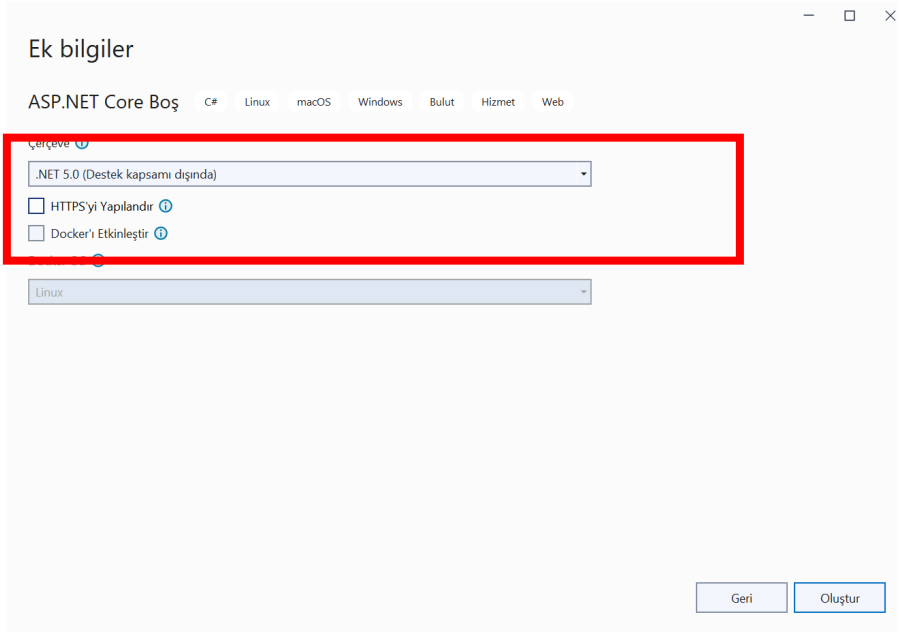
ADIM 3: ASP.Net Core Boş Proje Şablonu seçilir.



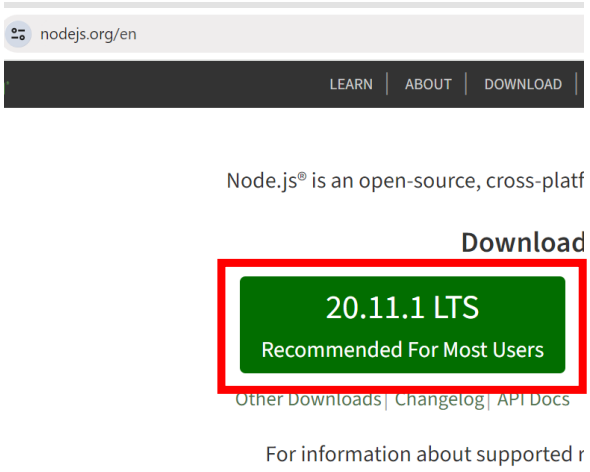
ADIM 4: Proje isimlendirilir.



ADIM 5: .NET 5.0 seçili. Tickler kapalı.



ADIM 6: Nodejs kurulumu



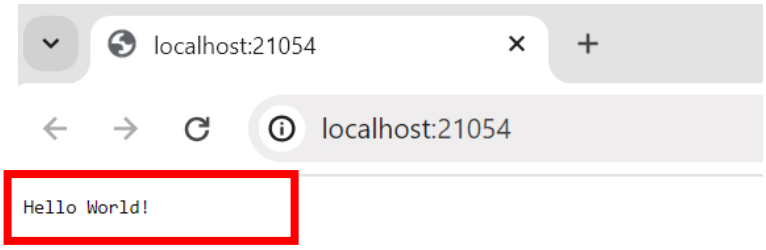
ADIM 7: Startup.cs dosyası içerisine View u görüntülemeyi sağlayacak komut eklenir.

```
namespace MovieApp.Web
{
    public class Startup
    {
        // This method gets called by the runtime. Use this method to add services to the
        // For more information on how to configure your application, visit https://go.mic
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddControllersWithViews();
        }

        // This method gets called by the runtime. Use this method to configure the HTTP r
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
```

ADIM 8: Startup.cs de default olarak bize gelen yazı “Hello World!”

```
// This method gets called by the runtime. Use this method to configure
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!");
        });
    });
}
```

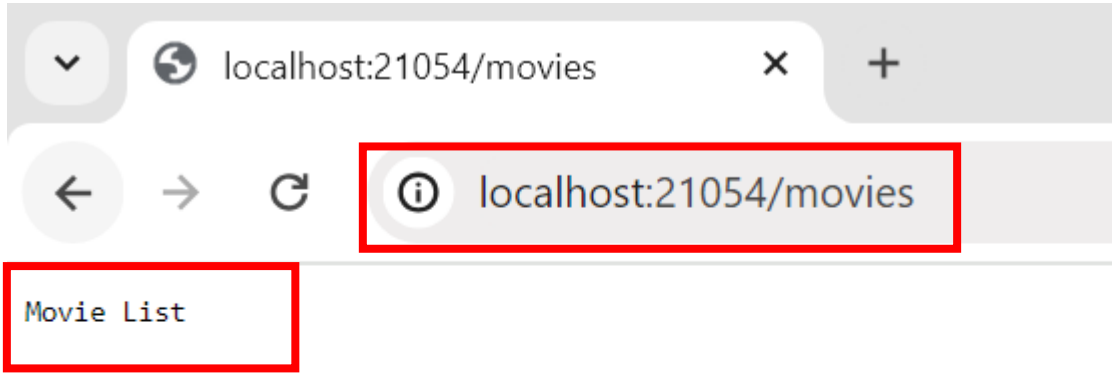
ADIM 9: Çalıştırılır.



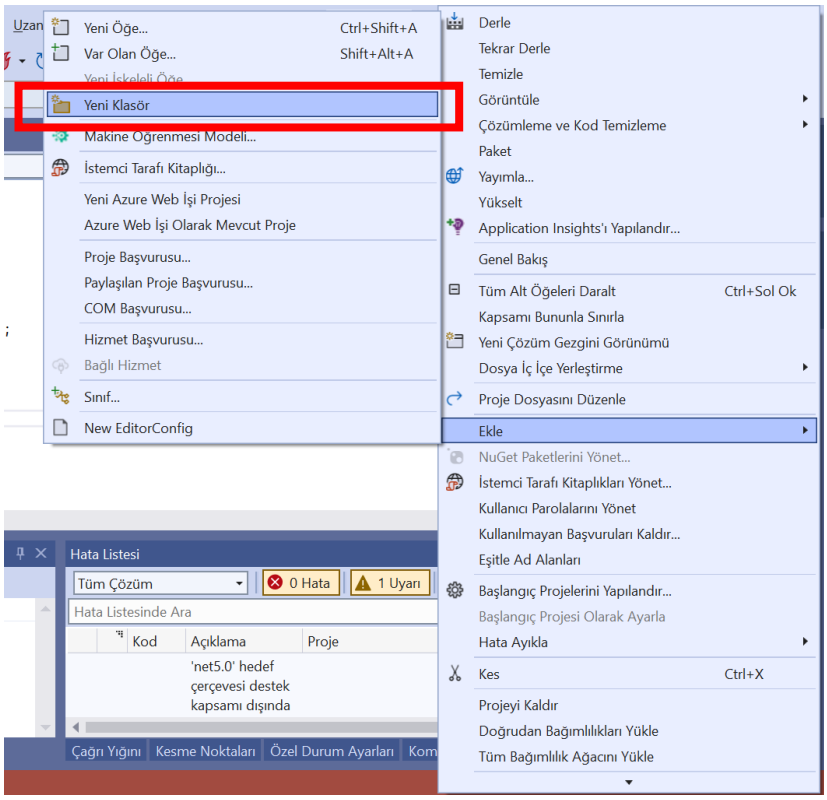
ADIM 10: Startup.cs de / ile başlayan yukarıdaki ana dizin altına /movies adında bir alt dizin ekleseydik.

```
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    app.UseRouting();

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

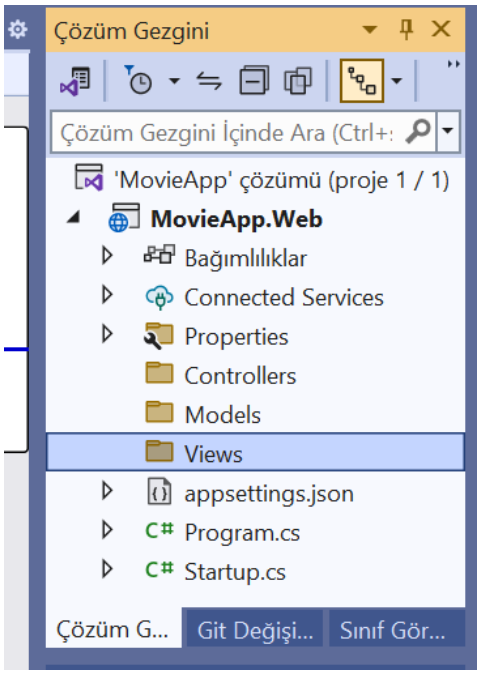
ADIM 11: Çalıştırıp url yi düzenlediğimizde karşımıza alt dizin gelir.



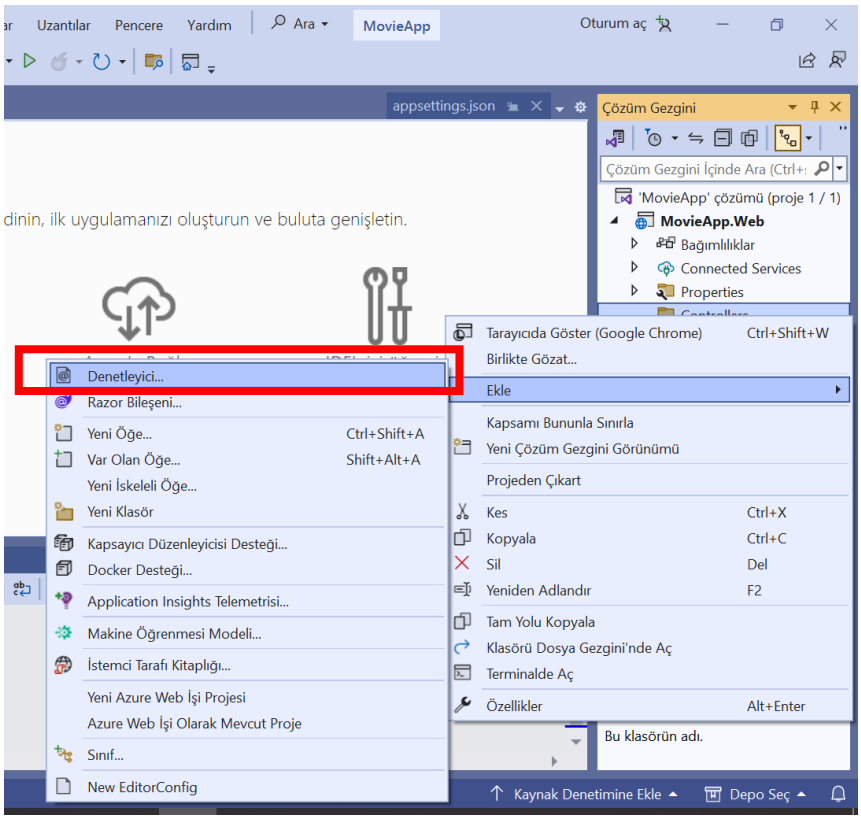
ADIM 12: Movie App te sağ tıklanarak üç yeni dosya eklenir.



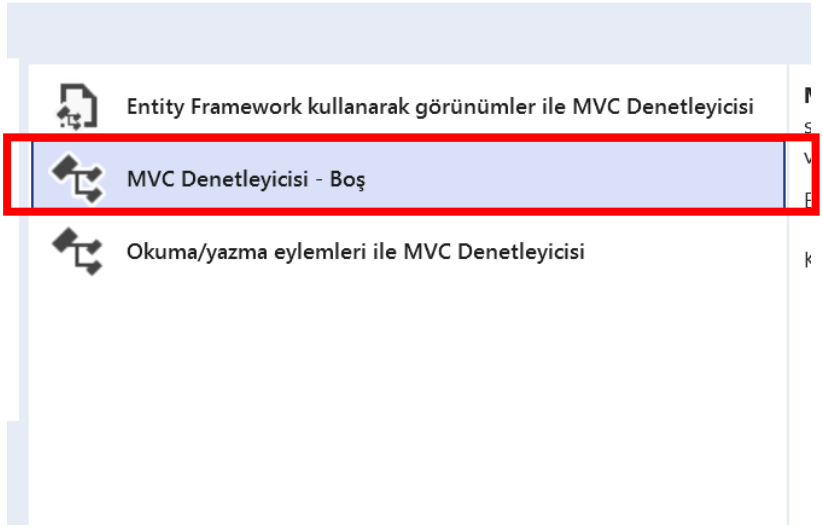
ADIM 13: Dosya isimleri Models-Views-Controllers olarak düzenlenir.



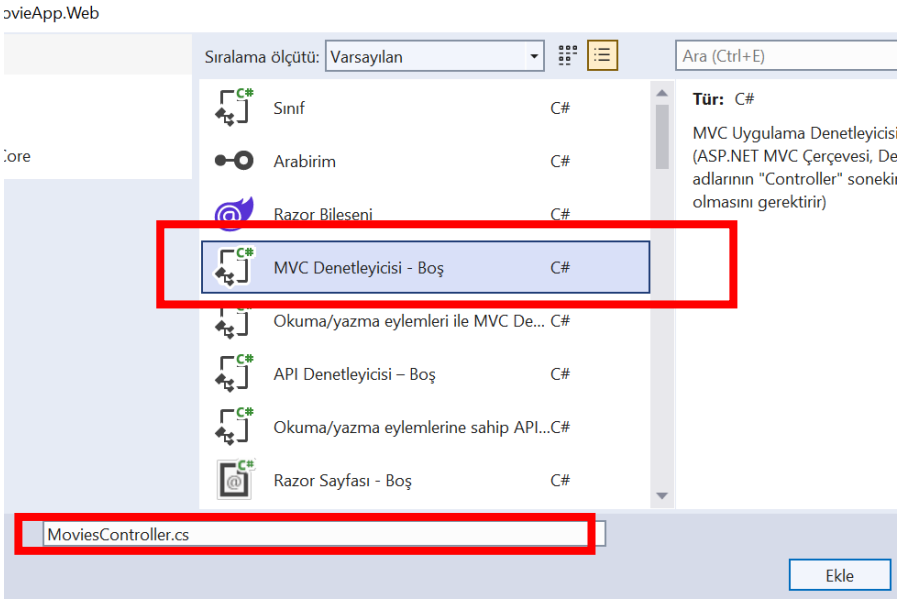
ADIM 14: Daha önce yapmış olduğumuz yönlendirmeyi controller aracılığı ile yapmak için controllers dosyası içine controller “denetleyici” ekliyoruz.



ADIM 15: MVC Denetleyicisi-Boş olarak seçilir.



ADIM 16: MVC Denetleyicisi-Boş olarak seçilir ve isimlendirilir.



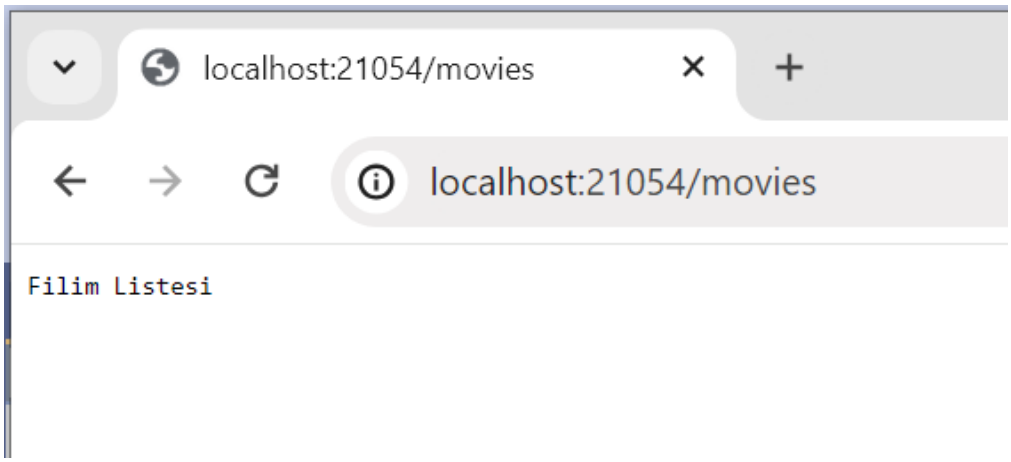
ADIM 17: Startup.cs üzerinde daha önce tanımlamış olduğumuz kodlar silinir. Sonrasında oluşturmuş olduğumuz controller aracılığıyla MapControllerRoute metodu kullanılır.

```
app.UseRouting();

app.UseEndpoints(endpoints =>
{
    endpoints.MapControllerRoute(
        name: "movieList",
        pattern: "movies",
        defaults: new { controller = "Movies", action = "List" }
        );
});
```

ADIM 18: MoviesController.cs dosyasındaki Action kısmı string değişkene göre düzenlenir.

```
namespace MovieApp.Web.Controllers
{
    public class MoviesController : Controller
    {

        public string List()
        {
            return "Film Listesi";
        }
```

ADIM 19: Çalıştırılır ve uzantı /movies olarak düzenlenir.



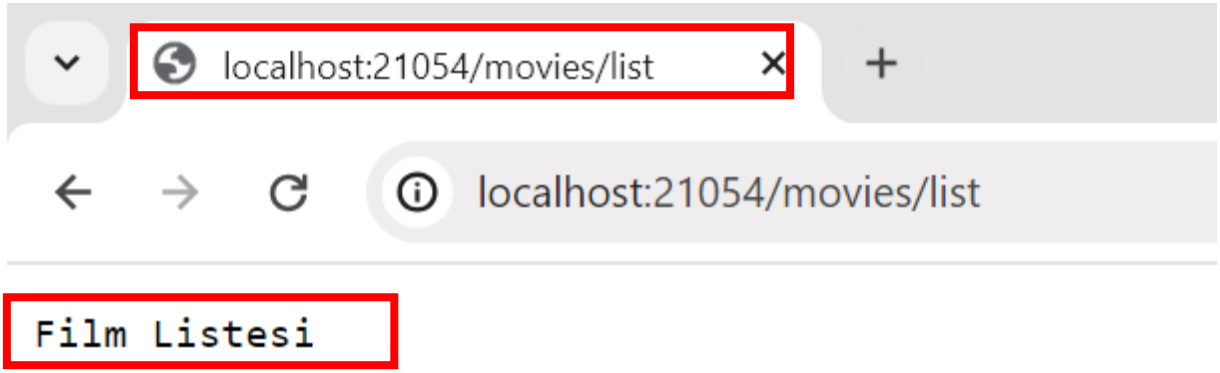
ADIM 20: Örneğin film detayını geriye döndürecek olan Details adında ikinci bir metod tanımlansın.

```
namespace MovieApp.Web.Controllers
{
    public class MoviesController : Controller
    {
        //localhost:****/movies/list
        public string List()
        {
            return "Film Listesi";
        }

        //localhost:****/movies/details
        public string Details()
        {
            return "Film Detayı";
        }
    }
}
```

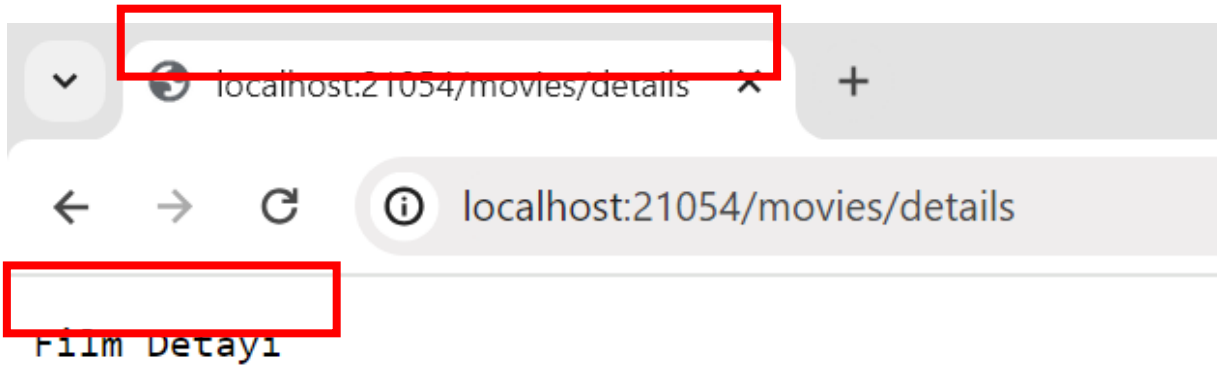
ADIM 21: ilgili erişim metoduna göre Startup.cs dosyasındaki kodlar düzenlenir.

```
app.UseEndpoints(endpoints =>
{
    endpoints.MapControllerRoute(
        name: "movieList",
        pattern: "movies/list",
        //MoviesController dosyasındaki action list metodu
        defaults: new { controller = "Movies", action = "List" }
        );

    endpoints.MapControllerRoute(
        name: "movieList",
        pattern: "movies/details",
        //MoviesController dosyasındaki action Details metodu
        defaults: new { controller = "Movies", action = "Details" }
        );
});
```

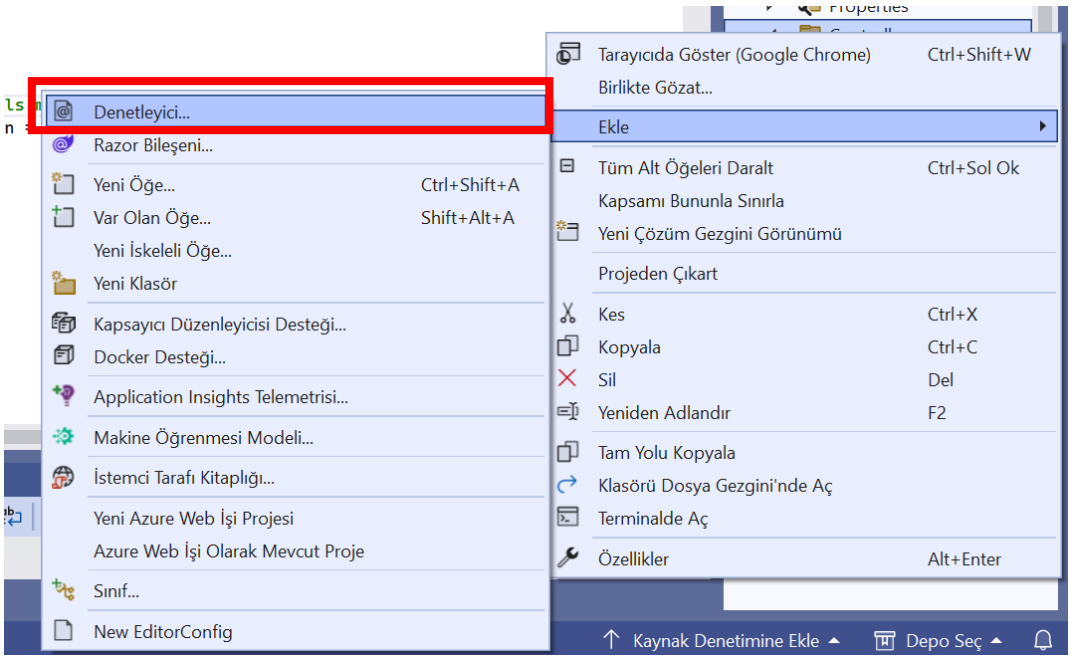
ADIM 22: URL düzenlenerek ilgili metod çağırılır.



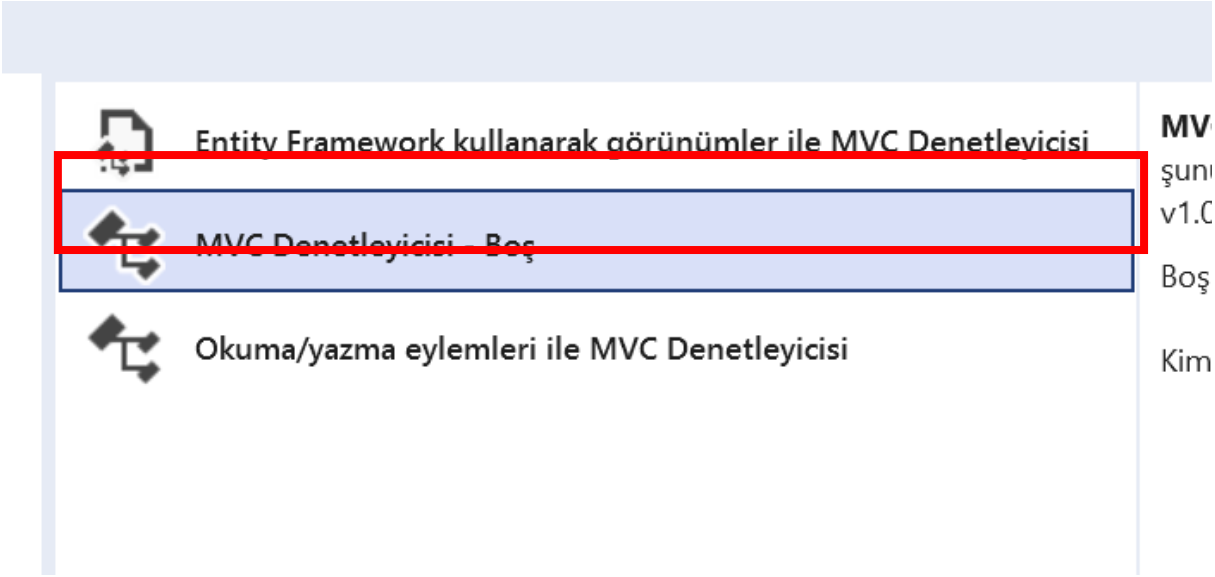
ADIM 23 : URL değiştirilir.



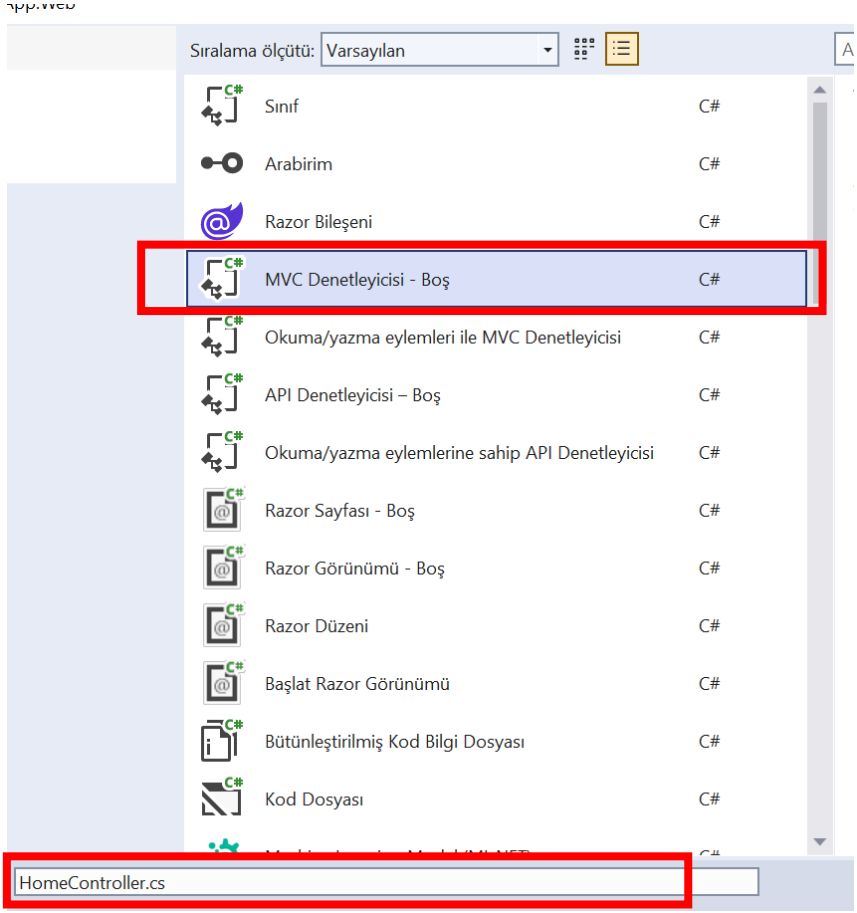
ADIM 24: Uygulamaya yeni bir Controller eklenir. Controller sağ tıkla.



ADIM 25: Boş Controller seçiyoruz.



ADIM 26: HomeController olarak isimlendiriyoruz.



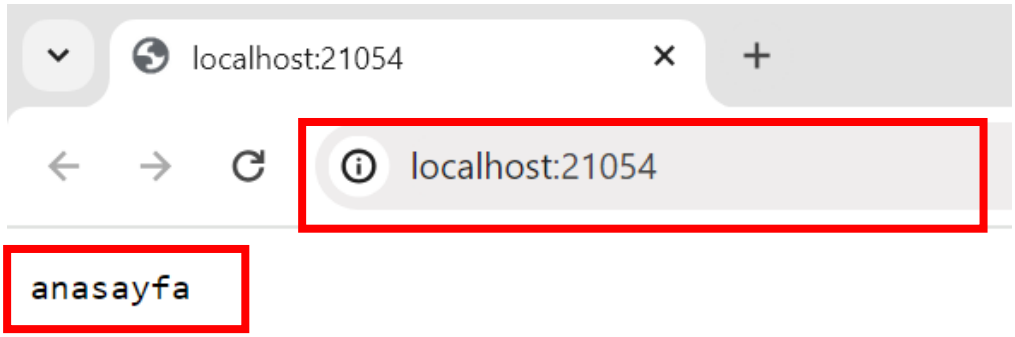
ADIM 27: HomeController.cs dosyası gerekli kodları yazılır. Anasayfa ve Hakkımızda sayfaları oluşturulur.

```
using Microsoft.AspNetCore.Mvc;

namespace MovieApp.Web.Controllers
{
    public class HomeController : Controller
    {
        public string Index()
        {
            return "anasayfa";
        }
        public string About()
        {
            return "hakkımızda";
        }
    }
}
```

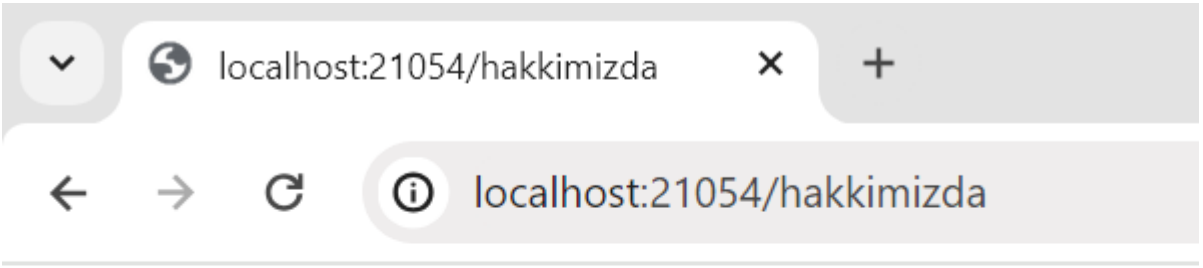
ADIM 28: Startup.cs dosyasında tüm ilgili URL uzantıları düzenlenir.

```
app.UseEndpoints(endpoints =>
{
      //localhost:****
    endpoints.MapControllerRoute(
         name: "home",
         pattern: "",
         //HomeController dosyasındaki Index
         defaults: new { controller = "Home", action = "Index" }
         );
      //localhost:****/about
    endpoints.MapControllerRoute(
        name: "about",
        pattern: "hakkimizda",
        //HomeController dosyasındaki About
        defaults: new { controller = "Home", action = "About" }
        );

    endpoints.MapControllerRoute(
        name: "movieList",
        pattern: "movies/list",
        //MoviesController dosyasındaki List
        defaults: new { controller = "Movies", action = "List" }
        );

    endpoints.MapControllerRoute(
        name: "movieDetails",
        pattern: "movies/details",
        //MoviesController dosyasındaki Details
        defaults: new { controller = "Movies", action = "Details" }
        );
});
```

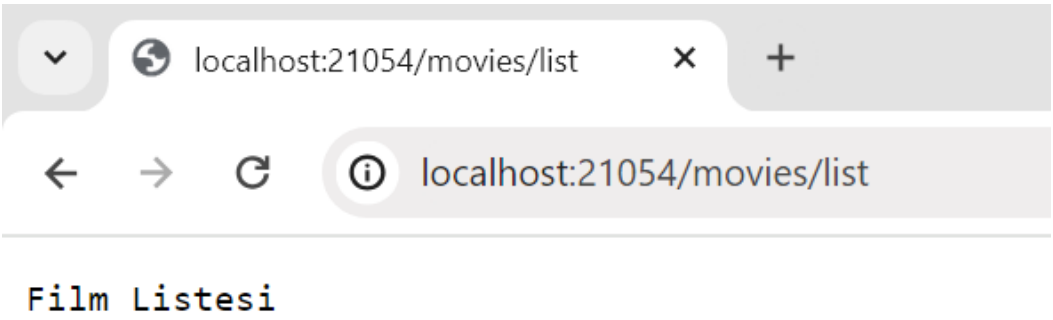
ADIM 29: Çalıştırılır.



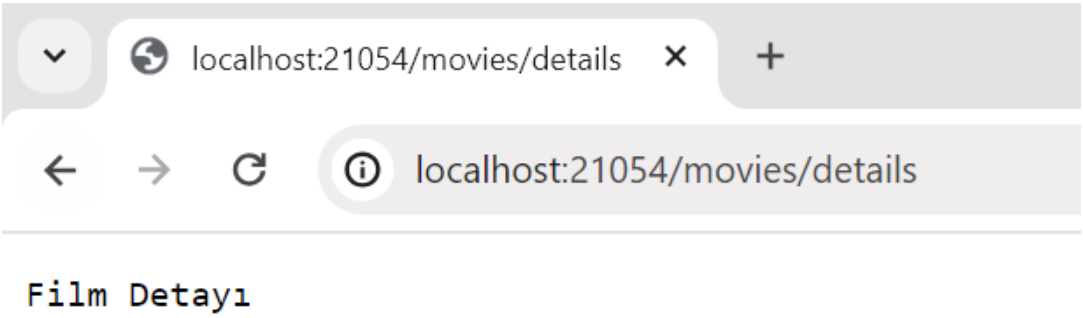
ADIM 30: URL düzenlenir



ADIM 31: URL değiştirlir



ADIM 32: URL değiştirilir.



SORU: Tek bir Routing ile list- details- Index-About sayfalarına erişim sağlansın ve açılış sayfası HomeController altındaki Index olsun.

2. DEFAULT ROUTING TANIMLAMAK

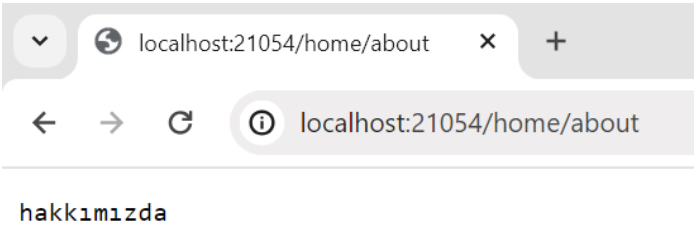
ADIM 1: Dört farklı routing bilgisi yazmak yerine tek bir routing ile tanımlanabilir.

Startup.cs | MoviesController.cs | MovieApp.Web

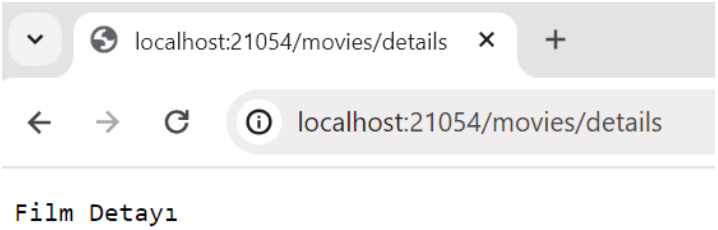
MovieApp.Web.Startup

```
app.UseRouting();

app.UseEndpoints(endpoints =>
{
    endpoints.MapControllerRoute(
        name: "default",
        pattern: "{controller}/{action}"

        );

});
}
```

ADIM 2: Çalıştırıp tek routing ile erişebildiğimizi görelim. HomeController için.



ADIM 3: Çalıştırıp tek routing ile erişebildiğimizi görelim. MoviesController için.



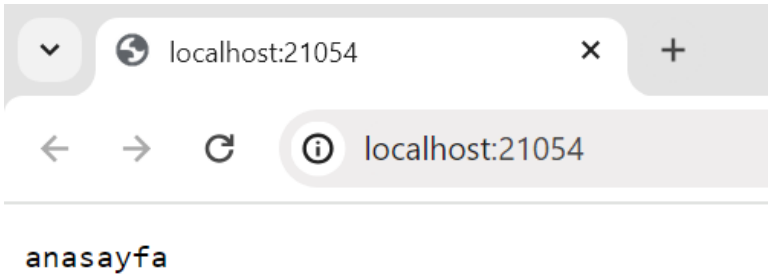
ADIM 4: Herhangi bir Controller ismi belirtmeden *Varsayılan Controller* tanımlayarak anasayfaya erişim sağlanır. Açılış sayfası HomeController altındaki Index olacak şekilde kodlar düzenlenir.

Startup.cs | MoviesController.cs | MovieApp.Web

MovieApp.Web.Startup

```
    app.UseRouting();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            name: "default",
            pattern: "{controller=Home}/{action=Index}"

            );

    });
}
```

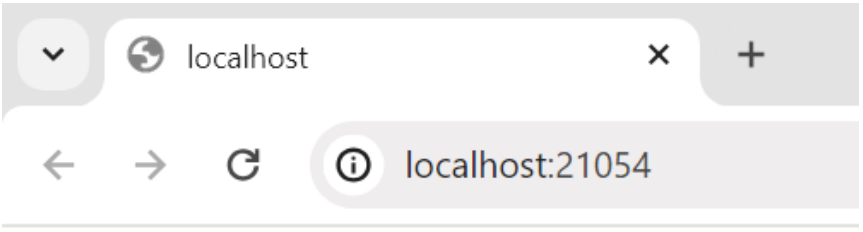
ADIM 5: Çalıştırarak kontrol edilir. Bu sayede sayfa çalıştırıldığında doğrudan Index action metodu gelir.



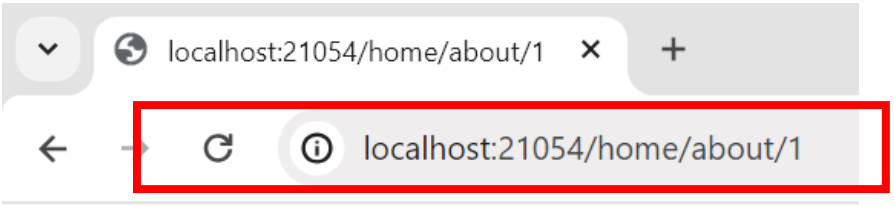
ADIM 6: Film Listesi İçerisinde bir elemana erişmek isteseydik, Kodlardaki Route bilgimiz 3 bölmeli olacak şekilde düzenlenirdi.

```
app.UseRouting();

app.UseEndpoints(endpoints =>
{
    endpoints.MapControllerRoute(
        name: "default",
        pattern: "{controller=Home}/{action=Index}/{id}"

        );

});
```

ADIM 7: Çalıştırılır. Route bilgisi üç bölmeli olarak tanımlandığından varsayılan açılış sayfasına erişilemez.

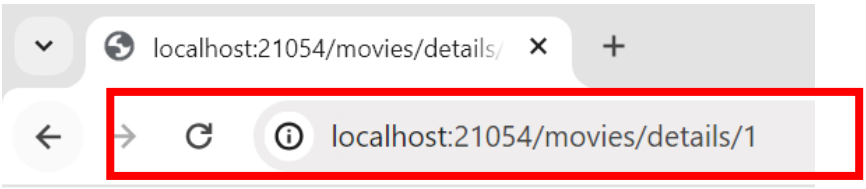


ADIM 8: URL üç bölmeli şekilde düzenlenerek tekrar çalıştırılır. **HomeController için**;




```
hakkımızda
```

ADIM 9: URL üç bölmeli şekilde düzenlenerek tekrar çalıştırılır. **MoviesController için**;

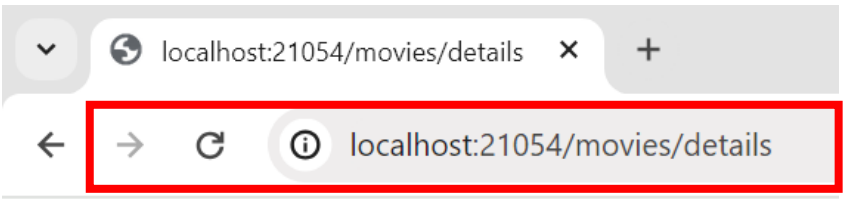



```
Film Detayı
```

ADIM 10: Üç bölmeli Rout bilgisi olmadan da sayfalara iki bölmeli URL ile erişebilmek için kodlar düzenlenir.

```
app.UseRouting();

app.UseEndpoints(endpoints =>
{
    endpoints.MapControllerRoute(
         name: "default",
         pattern: "{controller=Home}/{action=Index}/{id?}"

         );

});
```

ADIM 11: URL iki bölme olarak düzenlenerek çalıştırılır.



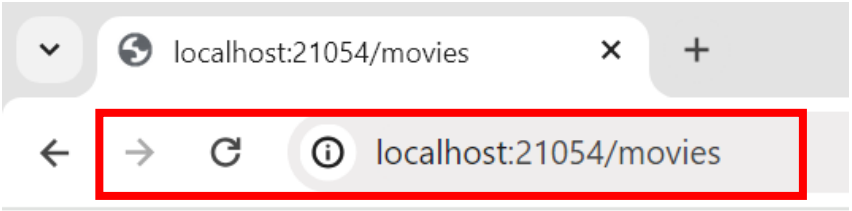

```
Film Detayı
```

SORU: Tek parametreli (bölümlü) URL ile movies e erişim sağlamak için gerekli kodları yazınız.

ADIM 12: Sadece Controller bilgisi ile URL de tek bölümlü erişim sağlamak için Movies Controller altına bir Index metodu tanımlanır. Home Controller altına Index metodu daha önce tanımlanmıştı.

```
namespace MovieApp.Web.Controllers
{
    public class MoviesController : Controller
    {

        public string Index()
        {
            return "Film Index";
        }

        //localhost:****/movies/list
        public string List()
        {
            return "Film Listesi";
        }

        //localhost:****/movies/details
        public string Details()
        {
            return "Film Detayı";
        }
    }
}
```

ADIM 13: Çalıştırıp tek parametreli URL ile Movies e erişim kontrol edilir.

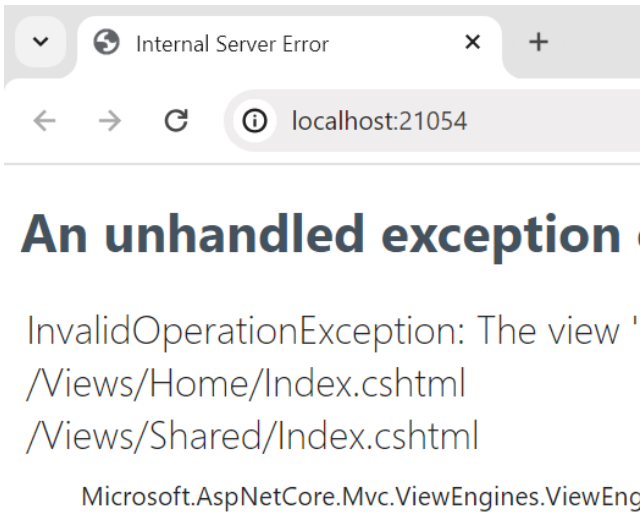


Film Index

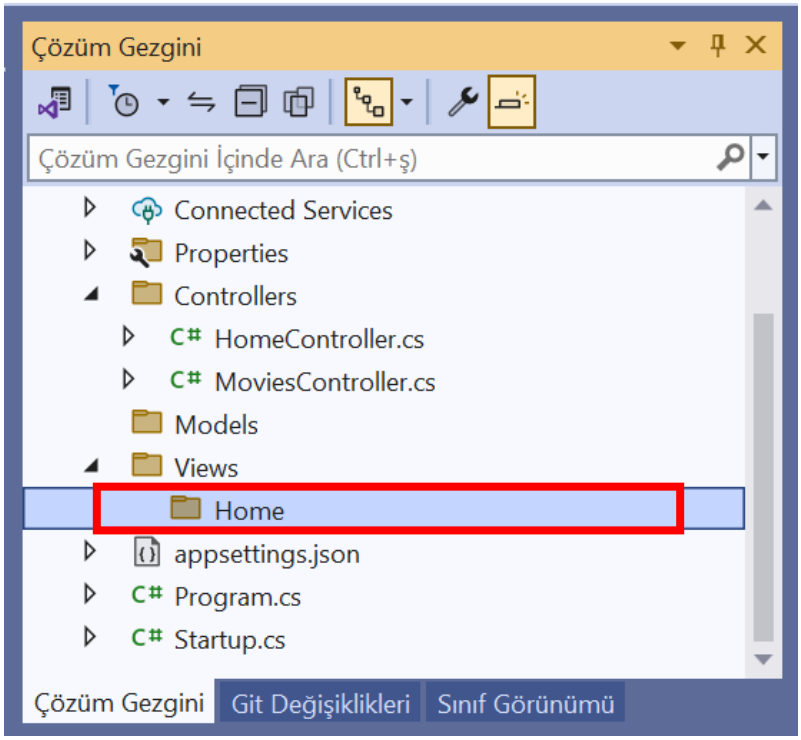
3. CONTROLLER DA STRING DEĞER YERİNE HTML SAYFASI DÖNDÜRMEK

ADIM 1: Bir HTML sayfası döndürmek için View komutu kullanılır ve metod IActionResult olarak düzenlenir.

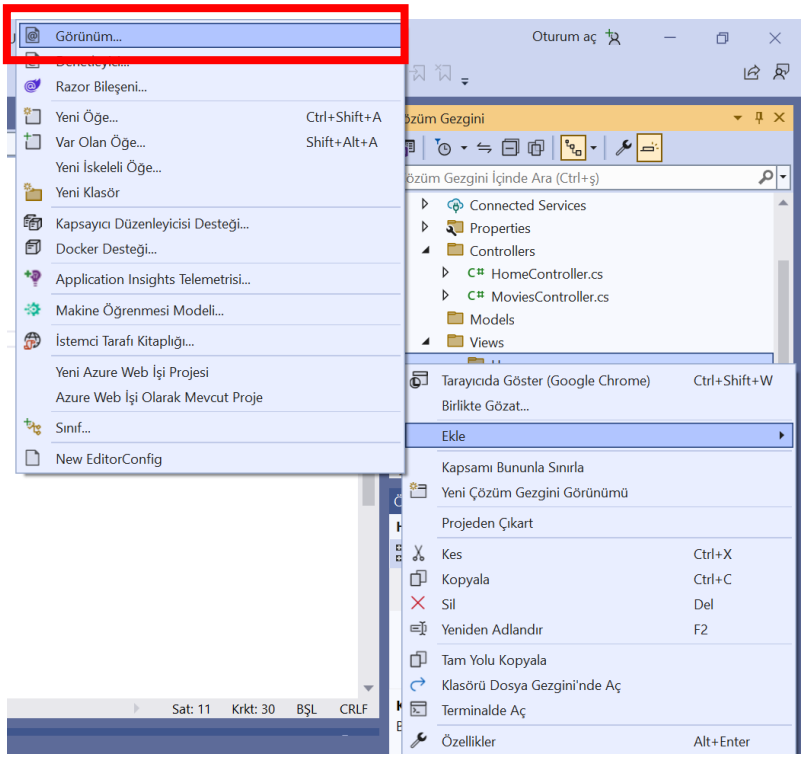
ADIM 2: Views /Home/Index dosyasını bulamadı. Bu uyarıyı almamak için Views altına Home Controller eklenmeli.



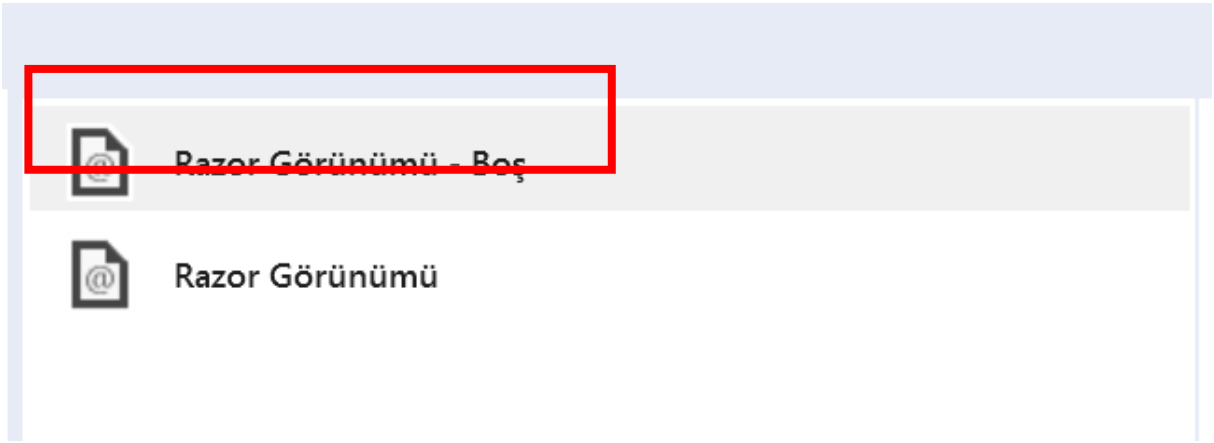
ADIM 3: Views sağ tıkla yeni dosya ekle Home olarak isimlendir.



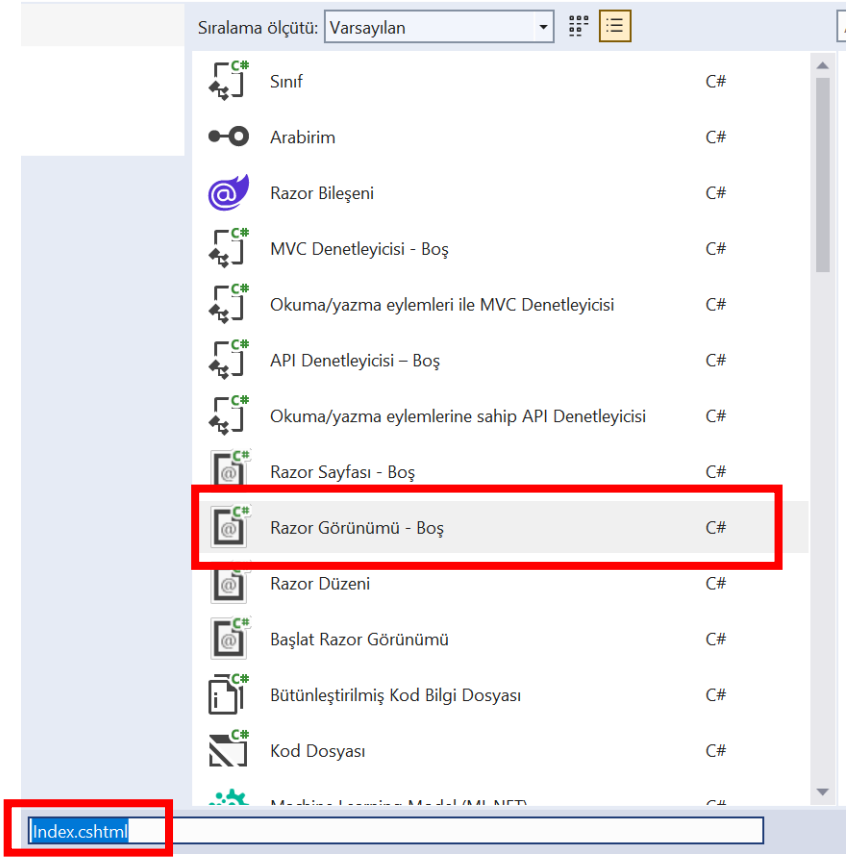
ADIM 4: Home sağ tıkla Görünüm / View ekle



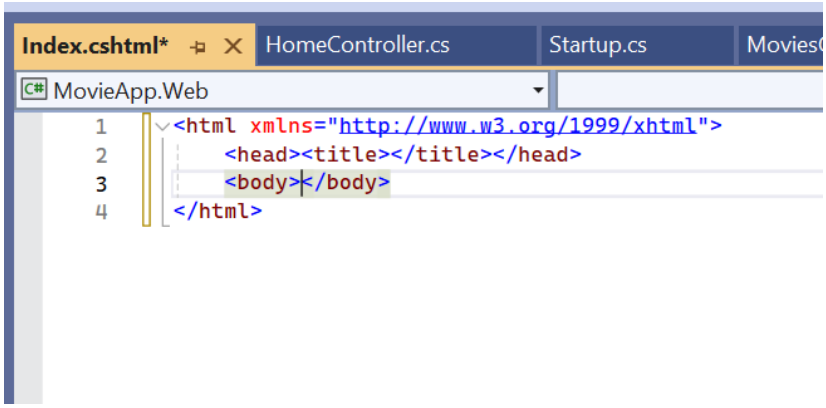
ADIM 5: Boş olan seçilir.



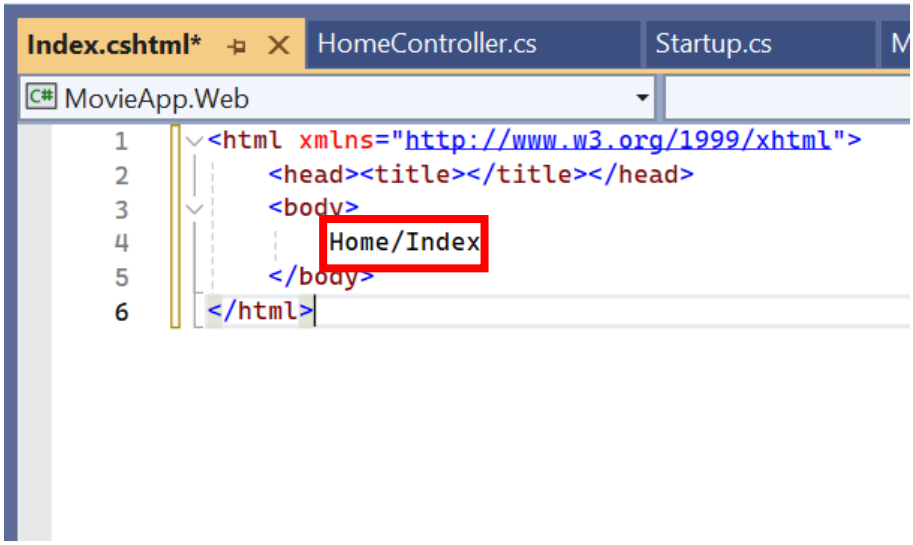
ADIM 6: İsimlendirilir.



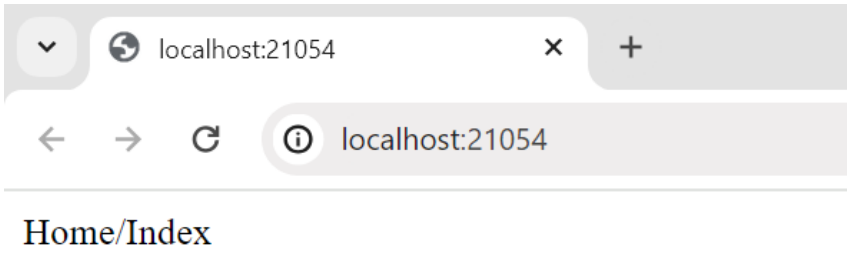
ADIM 7: İndex.cshtml içeriğini silinir html tab ile kodlar getirilir.



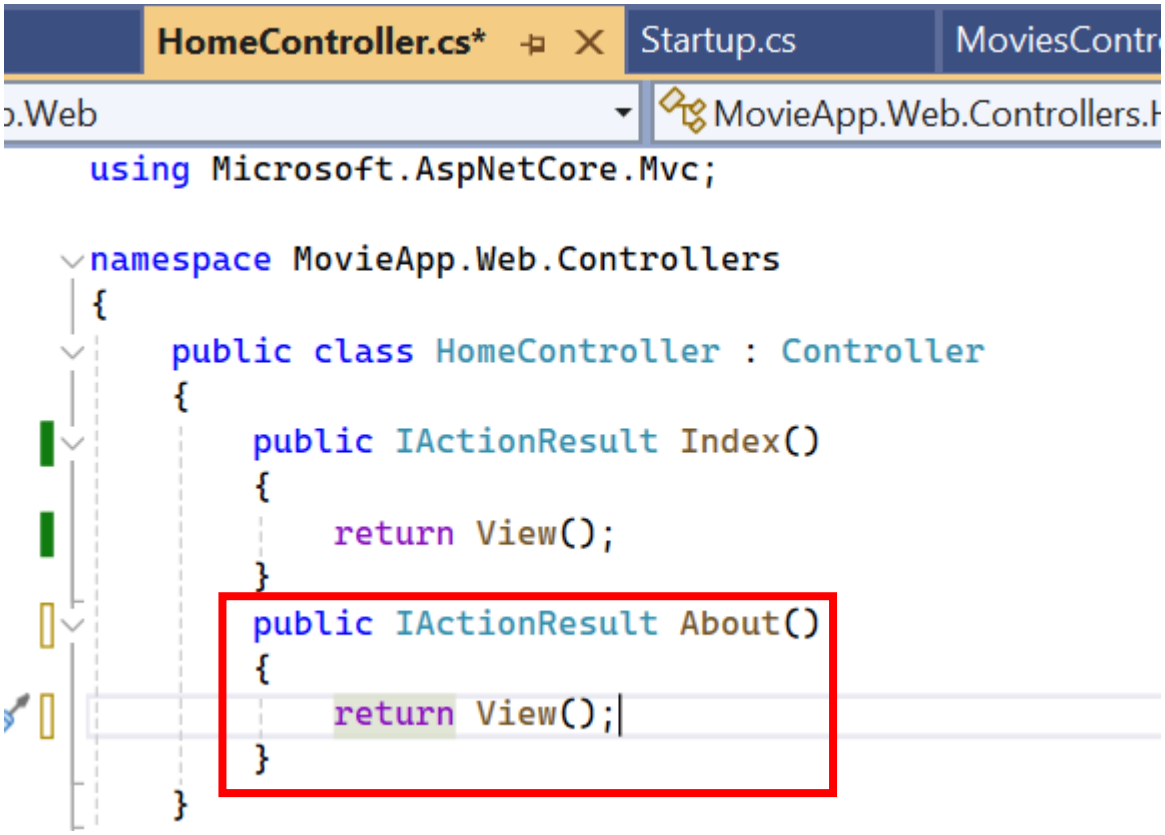
ADIM 8: Home/Index yazarak deniyoruz.



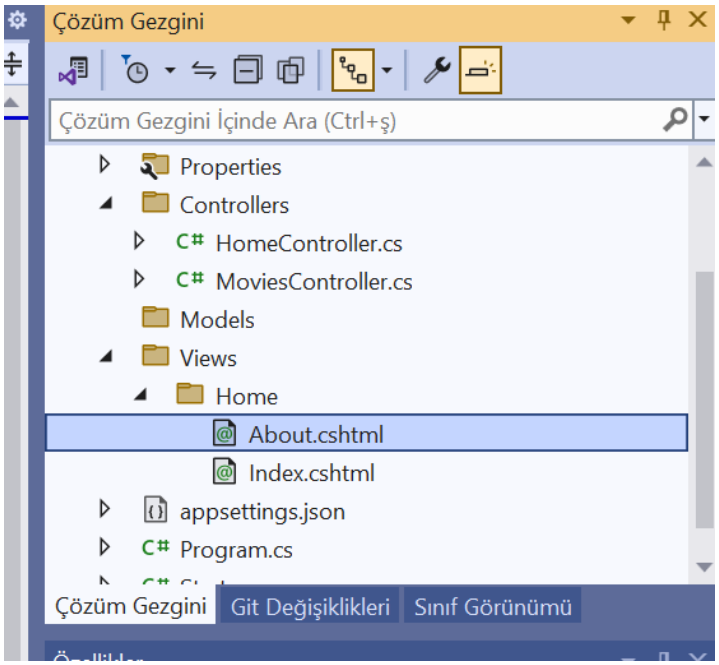
ADIM 9: HomeController içinde ActionResult metodu Index olarak düzenlenmişti. Çalıştırarak deniyoruz.



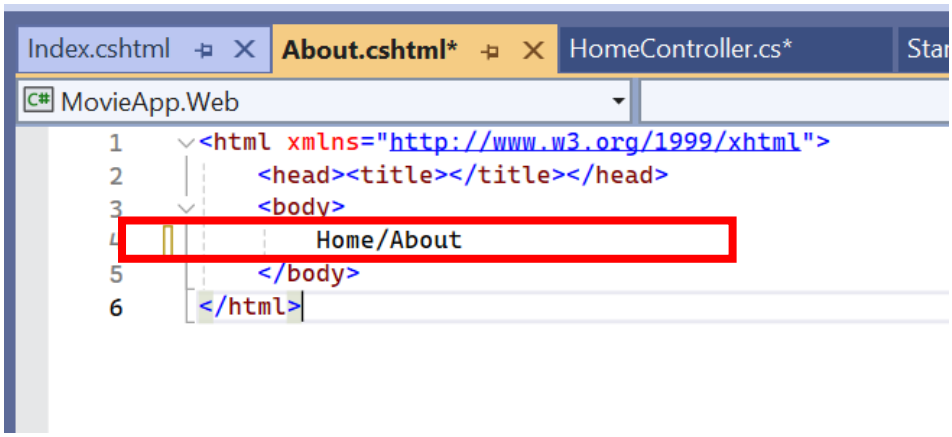
ADIM 10: Aynı işlem About için de yapılır. HomeController da kod bloğu düzenlenir.



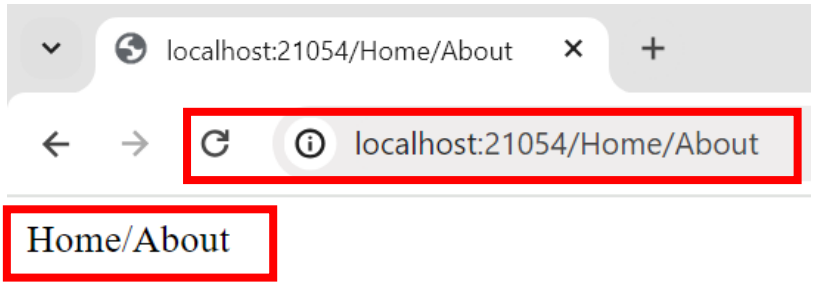
ADIM 11: Dosya eklenir ve isimlendirilir.



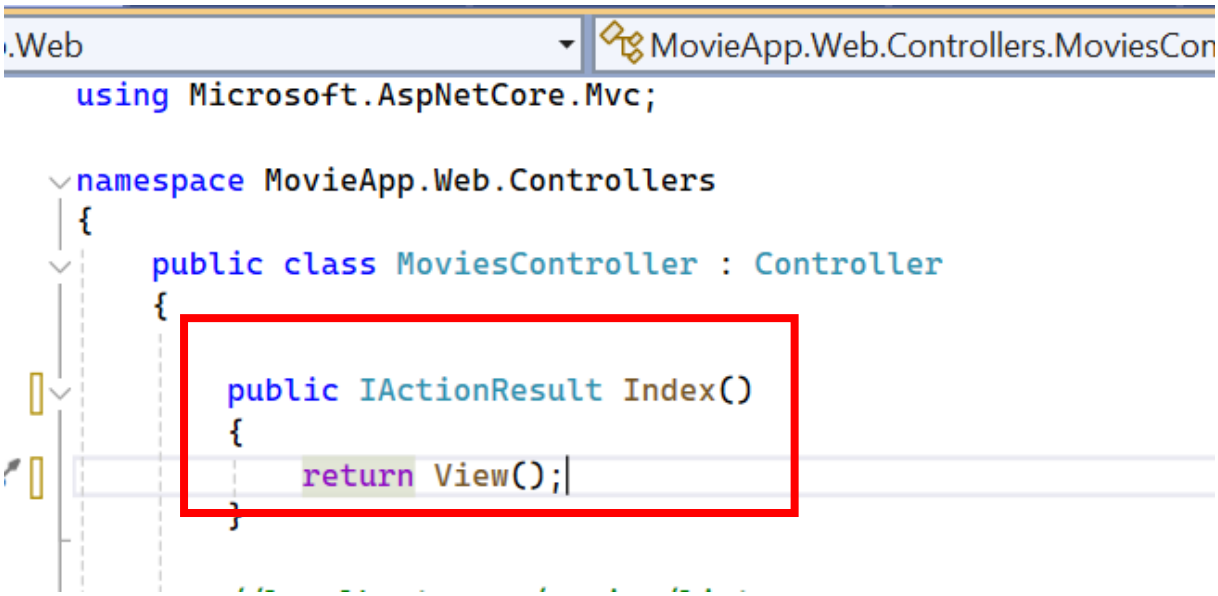
ADIM 12: Açılan dosyanın hangisi olduğunu görmek için About içindeki yazı değiştirilir.



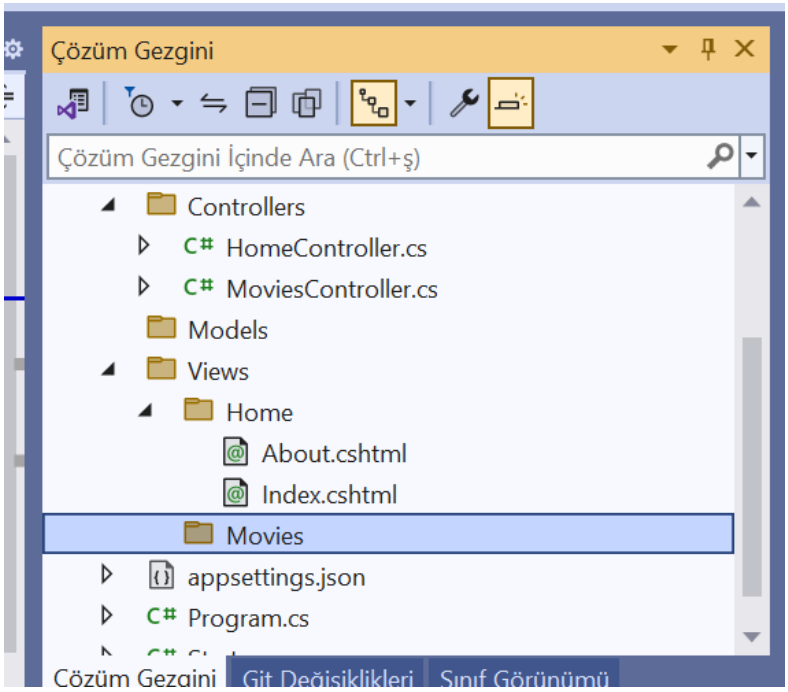
ADIM 13: Çalıştırılır. URL düzenlenerek açılış sayfası değiştirilir ve erişim kontrol edilir.



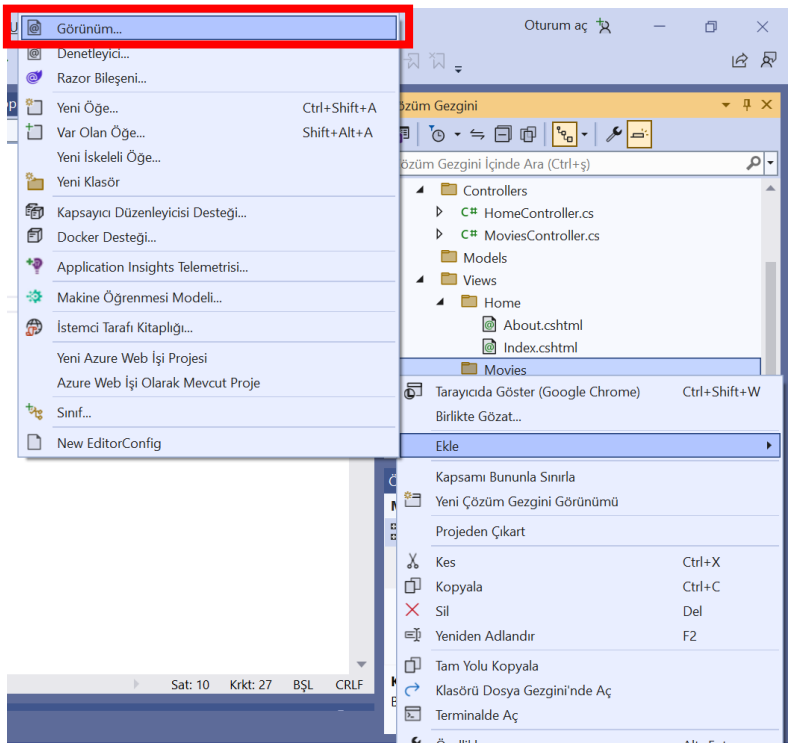
ADIM 14: MoviesControll içinde de aynı işlemler sürdürülür.



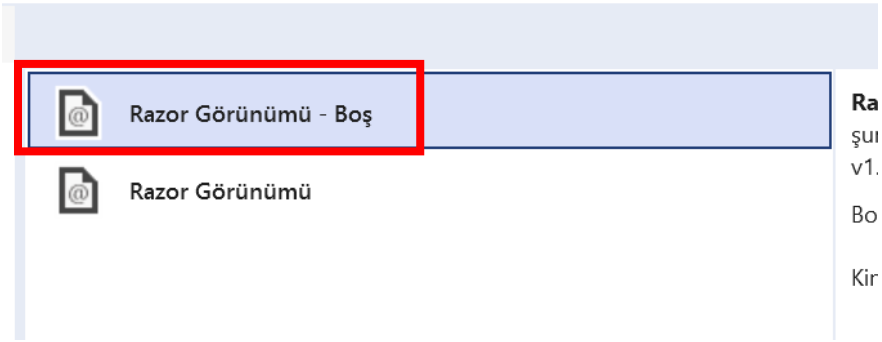
ADIM 15: Views altında yeni klasör oluşturulur Movies olarak isimlendirilir.



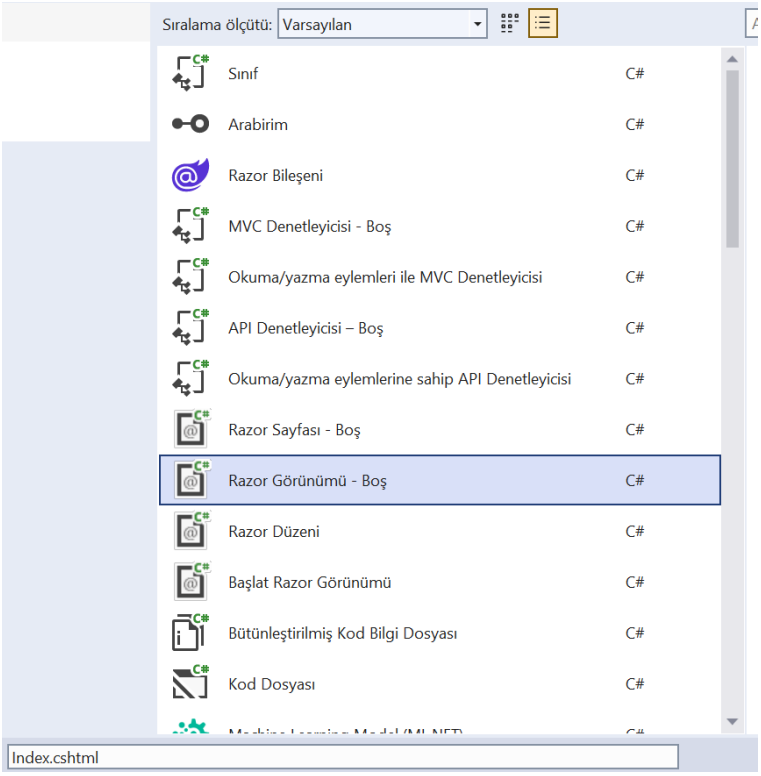
ADIM 16: Movies Klasörüne Sağ tık ile Görünüm eklenir.



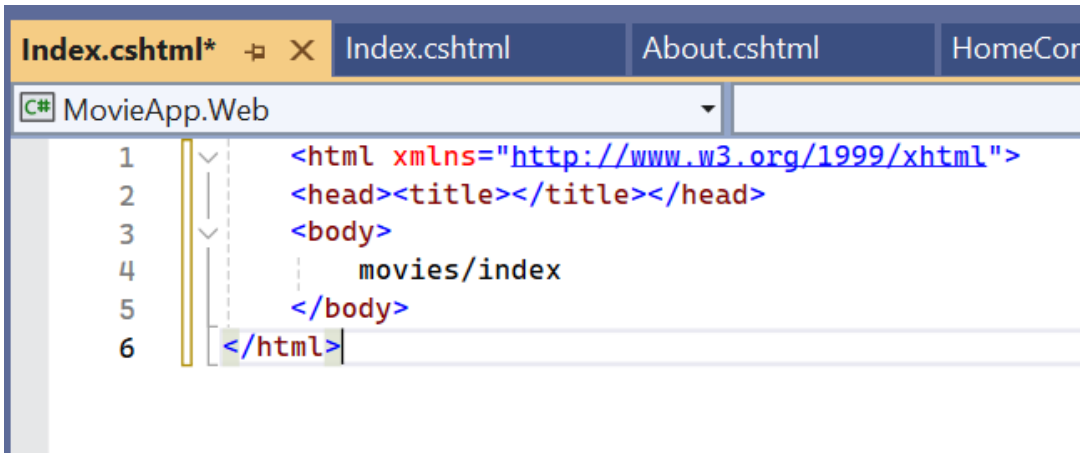
ADIM 17: Boş Görünüm Şablonu seçilir.



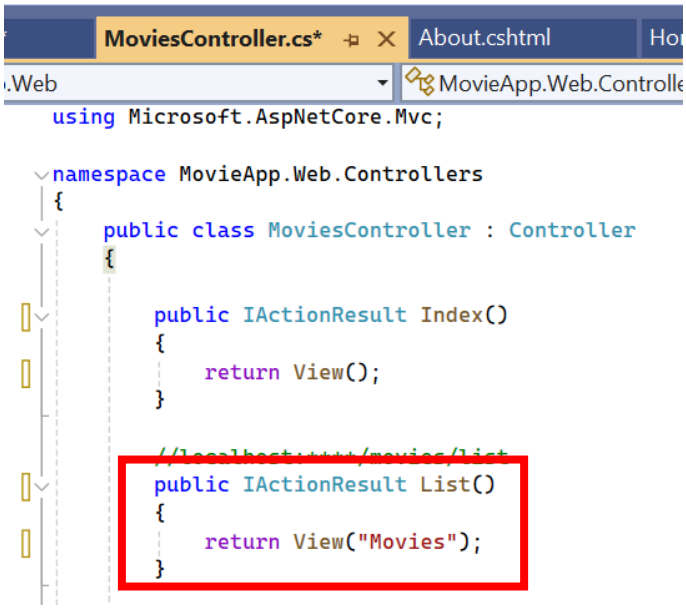
ADIM 18: Index olarak isimlendirilir.



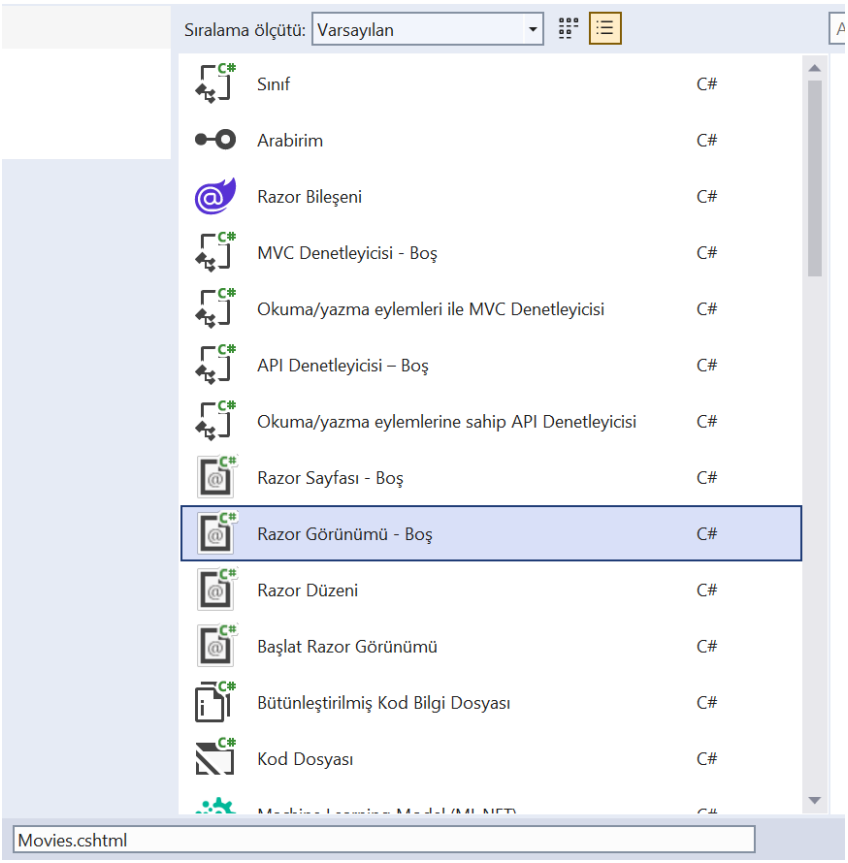
ADIM 19: Index.cshtml içindeki kodlar silinir. Html kodları tab ile getirilir. İçine açılan dosya açılışını deneyebilmek için movies/index yazılır.



ADIM 20: Movies.cshtml isminde bir dosyayı görüntüleyebilmek için MoviesController içindeki kodlar güncellenir.

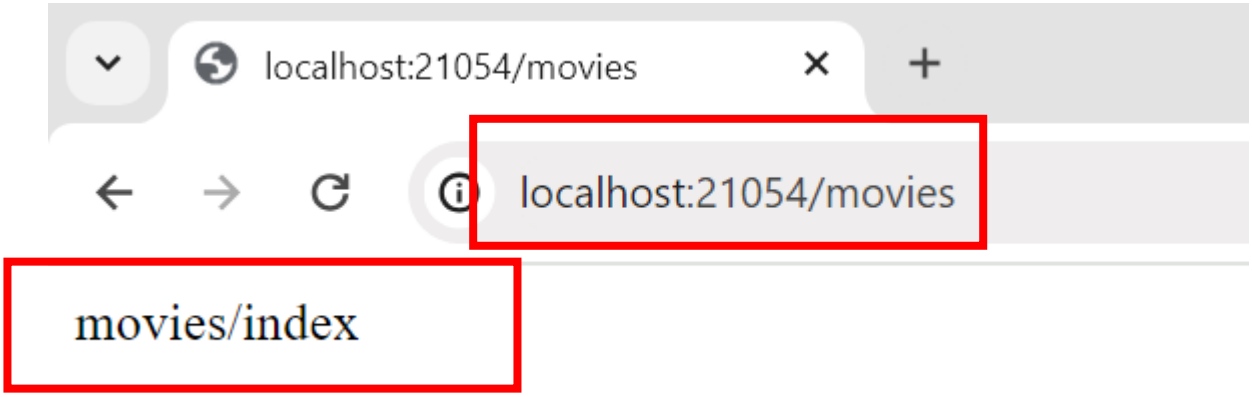


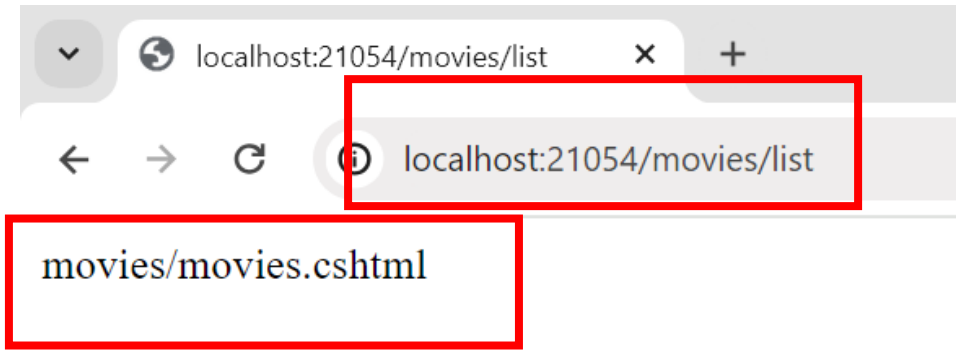
ADIM 21: Dosyayı görüntüleyebilmek için Movies Klasörü altına yeni bir görünüm eklenir. Movies olarak isimlendirilir.



ADIM 22: Kodlar temizlenir html içeriği tab tuşu yardımıyla eklenir. Html içeriğine movies/movies.cshtml yazarak çalıştırılır. (Movies Controller altındaki movies.cshtml dosyasına erişim)

ADIM 23: Çalştırılır URL düzenlenir. Movies a erişim kontrol edilir.





ÇALIŞMA SORUSU

Tek bir Routing ile list- details- Index-About sayfalarına erişim sağlanan ve açılış sayfası HomeController altındaki Index olan bir proje mimarisi oluşturunuz.

Tek parametreli (bölümlü) URL ile movies e erişim sağlamak için gerekli kodları yazınız.