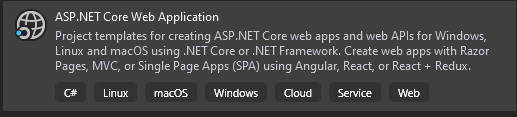
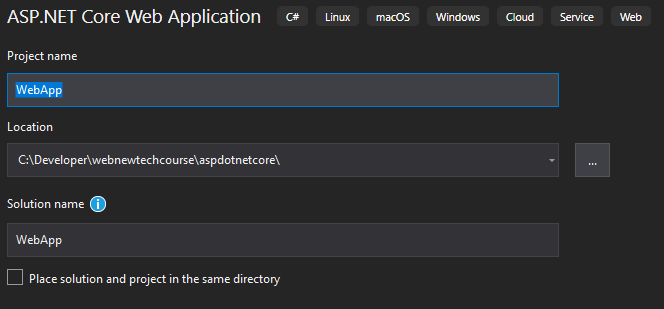
MS ASP.NET Core 3 projesi ile boş bir proje ile iskelet bir proje oluşturma adımları aşağıdaki gibidir:

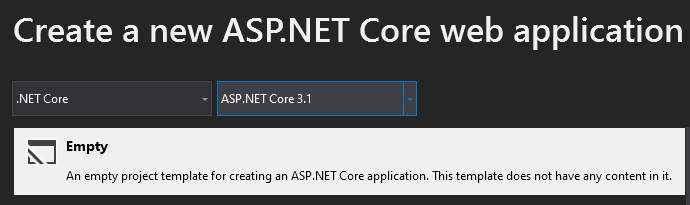
* Yeni proje kısmından aşağıdaki seçilir:



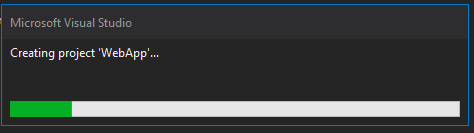
* Aşağıdaki gibi açılır ve create tuşuna basılır:



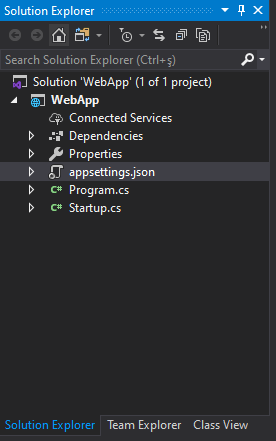
* Create tuşunu basmamızın ardından aşağıdaki gibi seçilir ve tekrar create tuşuna basılır:



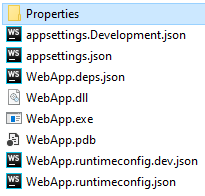
* Aşağıdaki gibi beklenir:



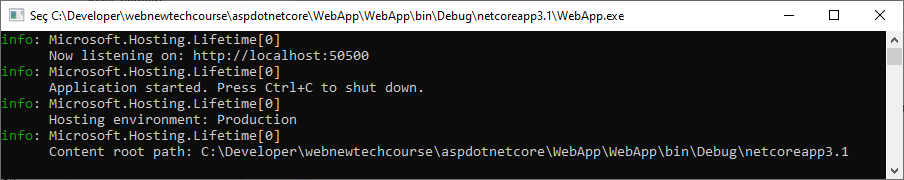
* Açılmış olan projenin parçaları aşağıdaki gibidir



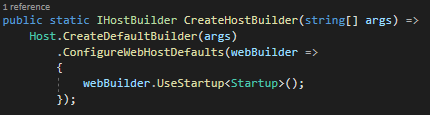
* Bu uygulamanın build edilmesi durumunda ".netcore executable file" elde edilir.



Çalıştırdığımızda ise;



* *Program.cs* dosyasında genel olarak yayınlanacak uygulamaya ilişkin konfigurasyon işlemleri yapılır. Örneğin :



Yukarıdaki *webBuilder.UseStartup<Startup>();* çağrısı startup sınıfını (*Startup.cs*) çağırır.

* Şimdi şunu ekleyelim:

*webBuilder.CaptureStartupErrors(true);*

Bu eklendiğinde uygulama başlarken de bir takım hataları yakalayıp mesaj olarak gösterebilir (konsolda).

Buraya ayrıca hangi adreslerden de erişilebileceği bilgilileri de yazılabilmektedir. Ör:

*webBuilder.PreferHostingUrls(true);*

*webBuilder.UseUrls("http://localhost:50500");*

* Ama asıl konfigurasyon startup sınıfında bulunur. Startup dosyasını ismini değiştirmek işlemi gerekli olsa bile dikkatli olunması gerekli. Birden fazla startup sınıfı olabilir. Default bir tane ile başlıyoruz.

* Startup sınıfının temel iki metodu bulunur:

*public void ConfigureServices(IServiceCollection services)*

*{ //… }*

*public void Configure(IApplicationBuilder app, IWebHostEnvironment env)*

*{ //… }*

* *ConfigureServices* metoduna kullanılacak servisler eklenir. Bu servisler örneğin, hangi arayüzü, hangi sınıfla "dependency injection" yöntemi ile kullanılabileceği belirtilir.

* *Configure* metodunda ise http isteklerine yönelik belirlemeler yapılır. Örneğin herhangi bir *exception* oluştuğunda hangi *action* metoduna yönlendirileceği belirtilir. Bu belirlemeye yönelik genel olarak tavsiye edilen sıra vardır. Bu sıra ileride ele alınacaktır.

* *Configure* metodunda aşağıdaki verilen kod ile ürün durumunda bir *exception* oluşması durumunda yönlendirilecek *action* metot yazılmıştır.

*if (env.IsDevelopment()) {*

*app.UseDeveloperExceptionPage();*

*}*

*else {*

*app.UseExceptionHandler("/Home/Error");*

*}*

* Genel programda (*program.cs*);

*app.UseStaticFiles();*

uygulamada başka statik dosyaların da kullanılabileceği belirtilmektedir.

* *app.UseRouting();* çağrısı ile yönlendirmeye ilişkin işlemlerin kullanılabileceği belirtilmektedir.

* *Configure* metodu içinde aşağıdaki;

*app.UseEndpoints(endpoints =>*

*{*

*endpoints.MapGet("/", async context =>*

*{*

*await context.Response.WriteAsync("Hello World!");*

*});*

*});*

çağrısı ile örneğin *default routing* 'in ne olabileceği belirtilebilir. Şimdi bakınız aşağıdaki örnekte Bu çağrı ile *default* yönlendirmenin *Home/Index* olduğunu yani *HomeController* sınıfındaki *Index* isimli *action* (actionResult) metoduna verilirse *id* parametreli *action* metodunun çağrısına yönlen demek istiyoruz:

*app.UseEndpoints(endpoints =>*

*{*

*endpoints.MapControllerRoute(*

*name: "default",*

*pattern: "{controller=home}/{action=index}/[id?}");*

*});*

**ANAHTAR NOTLAR;**

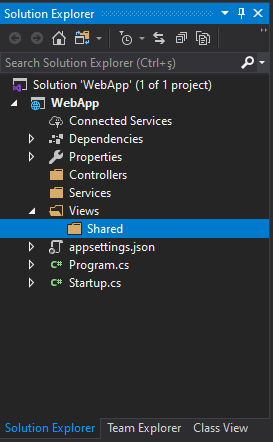
*Configure* metoduna eklenen çağrıların herbiribe *MIDDLEWARE* denmektedir. İskelet (scaffold) programda eklenen *middleware*'lerin eklendiği sırada olması tavsiye edilmektedir.

* *ConfigureServices* metoduna *controller* ve *view'*ların birlikte kullanılması için aşağıdaki

*services.AddControllersWithViews();*

çağrısının eklenmesi gerekmektedir.

* Yukarıdaki adımlardan sonra artık *Controllers*, *Views*, *Views* altında *Shared* ve *Services* dizinleri iskelet programa ekleyiniz: (WebApp çift tıklayınız, eklendiğini kod bazında görünüz)



* Bu işlemlerden sonra *Views* dizini altında *\_ViewImports.cshtml* isimli bir dosya ekleriz. Bu dosyada (*\_ViewImports.cshtml*) *view* 'lar için bir takım bildirimlerin olduğu ortak bir dosyadır. Örneğin:

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

direktifi ile *View* 'lar içerisinde "*tag helpers*" kullanılacağı belirtilmektedir. Burada ayrıca *using* direktifleri de yazılarak her v*iew* 'da *using* direktifi yapılmış duruma getirilir.

* *View* dizini altına, *view* 'lar başlatılırken, ortak kullanacağı işlemleri içeren *\_VıewStart.cshtml* isimli bir dosya eklenmiştir. Bu dosyada, her view için başlatılırken ne yapılacağı yazılmaktadır. Örneğin aşağıdaki kod ile her view'da kullanılacak ortak cshtml dosyası belirtilmektedir:

*@{*

*Layout = "~/Views/Shared/\_Layout.cshtml";*

*}*

* Eğer bir view yukarıda verilen bir layout'u kullanmak istemiyorsa, ilgili cshtml dosyasının başında yukarıdakine benzer şekilde *Layout* için ilgili değeri yada null ataması yapmalıdır.

**ANAHTAR NOT;**

*\_ViewImports.cshtml* ve *\_VıewStart.cshtml* dosyaları her *view* başında verilen sırası ile yapıştırılır biçiminde düşünülebilir.