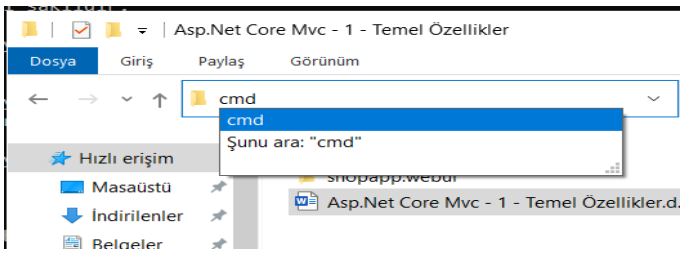
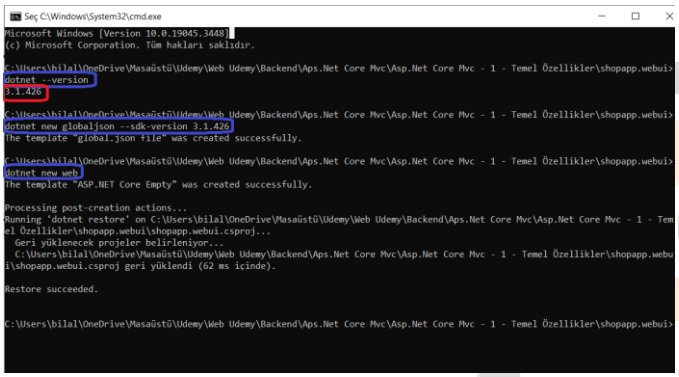
Asp.Net Core Mvc - 1 - Temel Özellikler

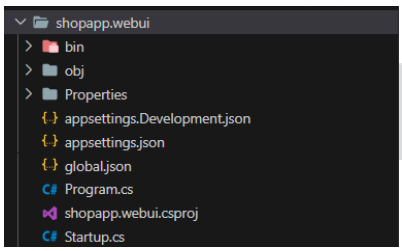
Proje Oluşturma



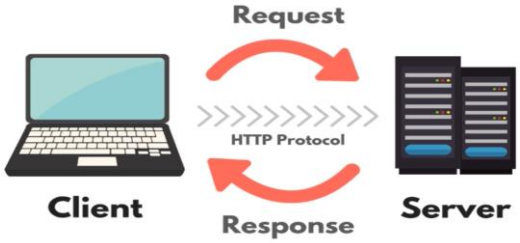
- İlk olarak projenizi tasarlamak istediğiniz klasöre gelir, yukarıda belirtilen kısmı silip cmd diyip enter diyin.



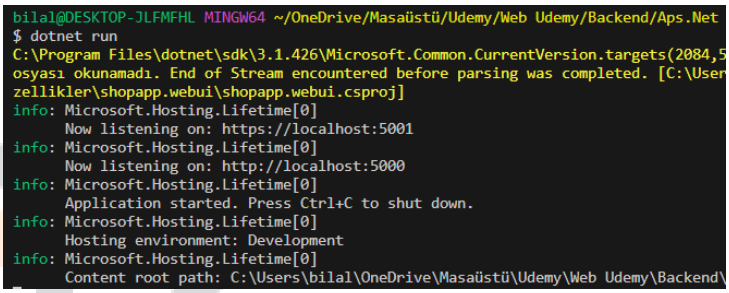
- Yukarıda ilk olarak "**dotnet --version**" yazın. Kırmızı ile gösterilen yerde size version gelecektir.
- Komutlardan devam edip "**dotnet new globaljson --sdk-version 3.1.426**" son kısımda sizde çıkan version ile kurulum yapın.
- Son olarak "**dotnet new web**" diyerek projeyi oluşturun.



Proje Çalıştırma

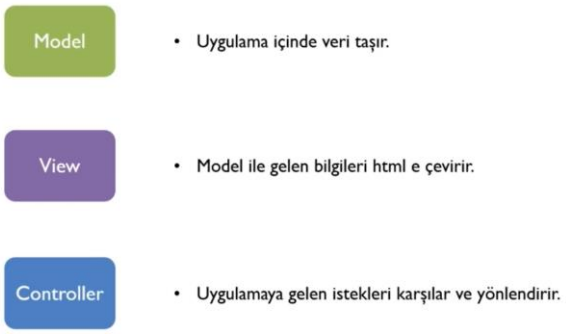


- Client server içerisinde bulunan veri tabanına bir talep gönderir(Request). Daha sonra serverdan response şeklinde nesne client'e taşınıp kullanıcıya gösterilmektedir.

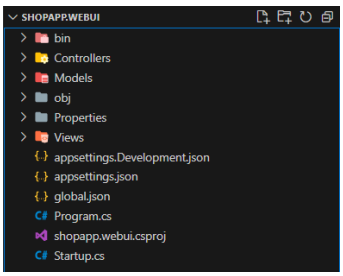


- Dotnet run dedikten sonra proje çalıştırılır.
- https://localhost:5001/ sayfasına giderek projenizi görüntüleyebilirsiniz.
- Ctrl + C tuşlarıyla projeyi durdurabilirsiniz. Tekrar çalıştırmak için dotnet run demeniz gerekmektedir.
- Dosyalarda bir değişiklik yapıldıktan sonra proje durdurulup tekrar çalıştırılır ve sayfa yenilenir.

Mvc Pattern



- Mvc .net yanında php gibi bir çok framework ile de çalıştırılabilir.
- "Bilalalgan.com/özgeçmiş" diye bir web sitesini arttığımızı düşünelim. İlk olarak controller veri tabanına istekleri iletir. Daha sonra özgeçmiş içerisindeki bilgiler model içerisinde taşınmaktadır. Taşınan veriler viev ile ekrana yazdırılmaktadır.



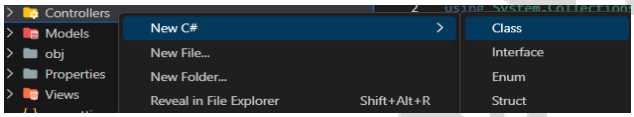
- Controllers, Models, Views klasörleri oluşturuldu.

Controller

Startup.cs

```
1 reference
public class Startup
{
    // This method gets called by the runtime. Use this method to add services to the container.
    // For more information on how to configure your application, visit https://go.microsoft.com/fwlink/?LinkID=3
    0 references
    public void ConfigureServices(IServiceCollection services)
    {
        //mvc controller
        services.AddControllersWithViews();
    }

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

        //localhost:5000/product/details
        //localhost:5000/product/details/2
        //localhost:5000/home/index
        app.UseEndpoints(endpoints =>
        {
            endpoints.MapControllerRoute(
                name: "default",
                pattern: "{controller}/{action}/{id?}"
            );
        });
    }
}
```

- 20. Satırda ConfigureServices içerisine Controllers servisi eklendi.
- 35.satırdaki UseEndPoints url istekleri nasıl işleneceğinizi belirler.
- 37. Satır ile yönlendirmeler yapılacaktır. Yani 32 33 34. Satırlardaki gibi url yerine gelen yazılara göre yönlendirme yapılacak.
- "default" isimli bir route belirlenir. Bu isim, bu yönlendirmenin adını temsil eder.
- pattern parametresi, URL'nin nasıl eşleştirileceğini belirler. {controller}/{action}/{id?} şablonu, bir URL'deki controller, action ve opsiyonel olarak bir id parametresiyle eşleşir. Örneğin, Home controller'ındaki Index action'ına gitmek için "/Home/Index" URL'si kullanılabilir. {id?} ifadesindeki ? işareti, id parametresinin opsiyonel olduğunu belirtir.



- Contollers'e classlar açalım.

Controllers/HomeController.cs

```
namespace shopapp.webui.Controllers
{
    //localhost:5000/home
    0 references
    public class HomeController:Controller
    {
        //localhost:5000/home/index
        0 references
        public string Index (){
            return "home/index";
        }

        //localhost:5000/home/about
        0 references
        public string About (){
            return "home/about";
        }
    }
}
```

- "HomeController" isminde class oluşturup enter diyin.
- 10. Satır ile Mvc içerisinden kalıtım ile Controller özelliği alıp using Microsoft.AspNetCore.Mvc eklemesini yapın.
- 13.satırda Index action metodu, "/home/index" URL'sine gelen istekleri işler. Bu URL'ye yapılan isteklerde "home/index" metni yanıt olarak döndürülür. Örneğin, " localhost:5000/home/index" adresine yapılan istek sonucunda tarayıcıda "home/index" metni görüntülenir.
- Peki /home sayfasına nasıl gidiyor diye soracak olursanız, **10. Satırdaki HomeController kısmındaki home kısmı belirler.** Yani aaController olsaydı /aa sayfasına giderdi.
- 18.satırdaki About action metoduda ındex ile aynı işlemi yapmaktadır.
- Kullanıcı "localhost:5000/home" veya "localhost:5000" adreslerine geldiğinde sayfa bulunamadı hatası almaktadır. Bunun için bir sonraki bölümde bunları düzenleyen varsayılan router işlemi yapılacaktır.
- Şimdi bir tane daha controller ekleyelim. (ProductController)

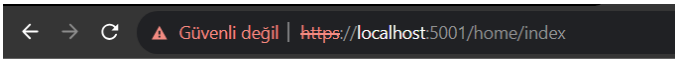
Controllers/ProductController.cs

```
namespace shopapp.webui.Controllers
{
    //localhost:5000/Product
    0 references
    public class ProductController:Controller
    {
        //localhost:5000/product/list
        0 references
        public string list (){
            return "product/list";
        }

        //localhost:5000/product/details
        0 references
        public string Details (){
            return "product/Details";
        }
    }
}
```

- HomeController gibi Controller classı içerisinde sağ tık yapıp class diyip eklenecek.
- Bu bölümde " localhost:5000/product/list" gibi sayfalara gitmektedir. Product gitmesinin sebebi 10. Satırda ProductController kısmındaki controller öncesinde "Product" belirlemektedir.
- HomeController ile aynı işlemler dönmektedir.
- Return kısımları ile sadece ekrana yazı yazdırıyor o kadar.
- Şimdi "dotnet run" diyelim.


```
product/list
```

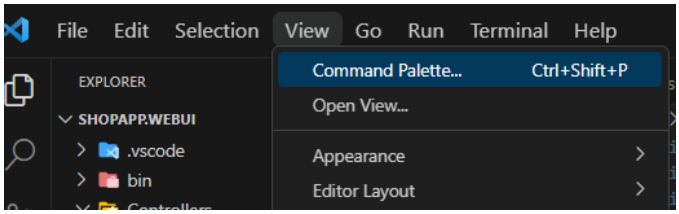

```
product/list
```



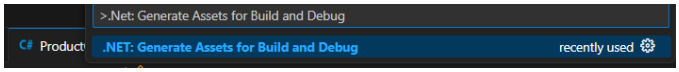

```
home/index
```

- Yukarıda görüldüğü gibi bütün yönlendirmelerimiz çalışmaktadır.
- /product/list/5 kısmında biz /5 kısmını ? işareti ile isteğe bağlı kılmıştık. O yüzden /5 olsa da olmasada bir sorun teşkil etmiyor.
- /product , /home veya / sayfaları bulunamamaktadır. Bir sonraki bölümde bu kısımlar için default bir ayar yapalım.

Default Routing



- View üzerinden command palette tıklayın.



- ".Net:Generate Assets for Build and Debug" diyip enter diyin. Artık uygulamada değişiklik yaptığımız zaman uygulamayı durdurup yeniden çalıştırmamıza gerek kalmayacak.

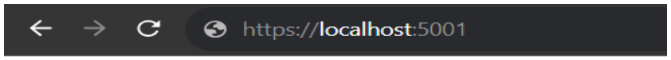
[Startup.cs](#)

```
35              app.UseEndpoints(endpoints =>
36              {
37                  endpoints.MapControllerRoute(
38                      name: "default",
39                      pattern: "{controller=Home}/{action=Index}/{id?}"
40                  );
41              });
```

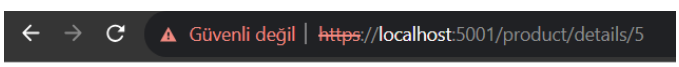
- 39.satırda güncelleme yapıldı. Controller=Home yapılırsa bir durum gelmediği takdirde Home sayfasına yönlendirilecektir. Action=ındex yaparak da bir şey gelmediğinde direk home=index'e atar.
- Lakin index yaptığımız için diğer yönlendirmeler de index metodu olması gerekiyor.

```
7   namespace shopapp.webui.Controllers
8   {
9       //localhost:5000/Product
        0 references
10      public class ProductController:Controller
11      {
            0 references
12          public string Index (){
13              return "product/index";
14          }
15
16          //localhost:5000/product/list
            0 references
17          public string list (){
18              return "product/list";
19          }
20
21          //localhost:5000/product/details
            0 references
22          public string Details (int id){
23              return "product/Details"+ id;
24          }
25      }
26  }
```

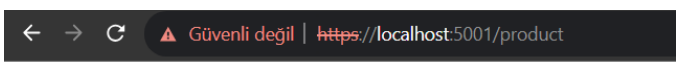
- 12. Satırdaki metot eklendi.
- 22. Satırda details içerisinde id parametresini ekledik ve /product/details/5 yazılırsa 5 numaralı ürüne gider.
- Programı çalıştırmak için "**dotnet watch run**" yazın. Değişiklik yaptığınızda durdurup çalıştırmanıza gerek kalmaz.



home/index



product/Details5



product/index

Views

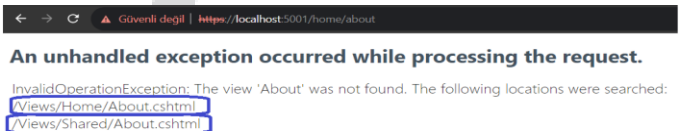
[Controllers/HomeController.cs](#)

```
7   namespace shopapp.webui.Controllers
8   {
9       //localhost:5000/home
        0 references
10      public class HomeController:Controller
11      {
12          //localhost:5000/home/index
            0 references
13          public IActionResult Index (){
14              return View();
15          }
16
17          //localhost:5000/home/about
            0 references
18          public IActionResult About (){
19              return View();
20          }
21
            0 references
22          public IActionResult Contact (){
23              return View("MyView");
24          }
25      }
26  }
```

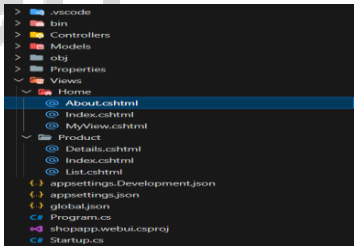
[Controllers/ProductController.cs](#)

```
7   namespace shopapp.webui.Controllers
8   {
9       //localhost:5000/Product
        0 references
10      public class ProductController:Controller
11      {
            0 references
12          public IActionResult Index (){
13              return View();
14          }
15
16          //localhost:5000/product/list
            0 references
17          public IActionResult list (){
18              return View();
19          }
20
21          //localhost:5000/product/details
            0 references
22          public IActionResult Details (int id){
23              return View();
24          }
25      }
26  }
```

- Bu şekilde bir string yerine web sayfası dönderebiliriz. Sadece IActionResult diyip View şeklinde değer döndermek.
- HomeController altındaki 23. Satırda View içerisine MyView yazdık, "/home/contact" sayfasına gidildiğinde View klasörü altında contact.cshtml araması gerekirken MyView dosyasını arayacak. Aşağı kısımlarda daha anlaşılır olacaktır.



- Bu şekilde views altında home altında bir about dosyası aramaktadır. Şimdi onları oluşturalım.

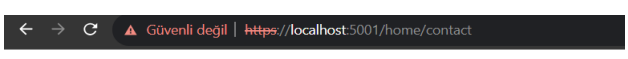


- Views içerisinde Home altında about, index eklendi.
- Vievs içerisinde Product altında Details,Index,List eklendi.

[Views/Home/MyView.cshtml](#)

```
1   <!DOCTYPE html>
2   <html leng="en">
3       <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <meta http-equiv="X-UA-Compatible" content="ie=edge">
7       <title>Home</title>
8       </head>
9       <body>
10          Aslinda Myview.cshtml dosyasi <br>
11          home/Contact
12      </body>
13  </html>
```

- Diğer web sayfaları da böyle doldurulacak. Sadece 10 ve 7. Satır değiştir. Hangi sayfada olduğun daha anlaşılır olur.



Aslinda Myview.cshtml dosyasi
home/Contact

Dinamik Veri Kullanımı

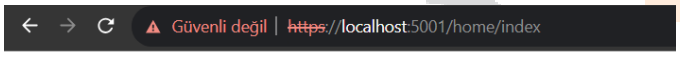
Controllers/HomeController.cs

```
13      public IActionResult Index (){
14          int saat = DateTime.Now.Hour;
15
16          ViewBag.mesaj = saat>12? "İyi Günler": "Günaydin";
17          ViewBag.username = "Bilal";
18
19          return View();
20      }
```

- 14. Satır ile şuanki saati alıp saat değişkeni içerisine atıyoruz.
- 16. Satır ile saat eğer 12 den büyükse veya eşitse iyi günler, değilse günaydın mesajını mesaj değişkenine atadık. Username bilgisini de bilal içerisine atadık.

```
13      public IActionResult Index (){
14          int saat = DateTime.Now.Hour;
15
16          ViewBag.simdikisaat = saat;
17          ViewBag.mesaj = saat>=12? "İyi Günler": "Günaydin";
18          ViewBag.username = "Bilal";
19
20          return View();
21      }
```

- 10. Satır ile @ kullanarak başında mesajı ekranda dinamik bir şekilde gösterilmektedir. Örnek ilerleyen aşamalarda kullanıcı giriş yaptığında username yerine kullanıcı adını atayacağız ve bu şekilde kullanıcı adı gelecektir.



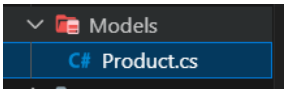
Saat : 12

İyi Günler Bilal, Nasilsin ?

Umarim Sitemizden memnunsunuzdur

- Yukarıda görüldüğü gibi saat 12 olduğundan dolayı İyi günler yazdı.

Models



- Models klasörüne sağ tık yapıp new class diyip Product yazıp enter diyelim.

Models/Product.cs

```
6   namespace shopapp.webui.Models
7   {
        1 reference
8       public class Product
9       {
            1 reference
10          public string Name { get; set; }
            1 reference
11          public int Price { get; set; }
            1 reference
12          public string Description { get; set; }
13      }
14  }
```

- Product içerisinde bu şekilde bir nesne yapısı tanımlıyoruz. Böylelikle ürünleri buradan oluşturabiliriz.

Controllers/ProductController.cs

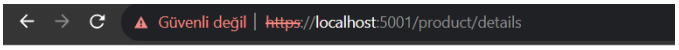
```
23      public IActionResult Details (){
24
25          var product = new Product();
26          product.Name = "Samsung S6";
27          product.Price = 6000;
28          product.Description = "Eazy Telefon";
29
30          return View(product);
31      }
```

- 25. Satır ile product nesnesi oluşturuldu.
- 26,27,28 ile isim fiyat açıklama bilgileri girildi.
- 30. Satırda return view içerisinde girerek nesneyi gönderiyoruz.

Views/Product/Details.cshtml

```
1   @model shopapp.webui.Models.Product
2
3   <!DOCTYPE html>
4   <html leng="en">
5       <head>
6       <meta charset="UTF-8">
7       <meta name="viewport" content="width=device-width, initial-scale=1.0">
8       <meta http-equiv="X-UA-Compatible" content="ie=edge">
9       <title>Product</title>
10      </head>
11      <body>
12          <p>Urun Adi: @Model.Name</p>
13          <p>Urun Fiyati: @Model.Price</p>
14          <p>Urun Aciklamasi: @Model.Description</p>
15      </body>
16  </html>
```

- 1. Satır ile model oluşturuldu ve 12,13,14. Satırlarda Model.name denildiğide ürüne ulaşıp ekranda yazdırıyoruz.

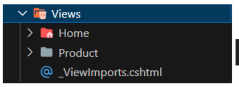


Urun Adi: Samsung S6

Urun Fiyati: 6000

Urun Aciklamasi: Eazy Telefon

View'e Veri Aktarım Yöntemleri



Views/_ViewImports.cshtml

```
@using shopapp.webui.Models
```

- İlk olarak Views içerisinde bir dosya oluşturalım ve ismini _ViewImports.cshtml koyalım ve 1. Satırdaki kodu yazalım.
- Bu dosya bizlere artık dosya Models dosyasının konumunu belirtmemize gerek kalmayacak.

Controllers/ProductController.cs

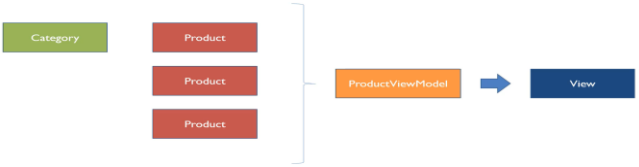
```
public class ProductController:Controller
{
    0 references
    public IActionResult Index (){
        var product2 = new Product(){
            Name = "Iphone x",
            Price = 10000,
            Description= "Denemelik telefon"
        };

        //ViewData
        ViewData["Product"] = product2;
        ViewData["Category"] = "Telefon";

        //ViewBag
        ViewBag.Product = product2;
        ViewBag.Category = "Telefon";

        //Model
        return View(product2);
    }
```

- 3 farklı gönderim yöntemi vardır. Model,Viewbag,Data.
- ViewData kullanımı 21 22. Satırlarda gösterilmiştir.
- Viewbag kullanımı 25,26. Satırlarda gösterilmiştir.
- Model için 29.satırda product2 nesnesini parametre olarak verdik.

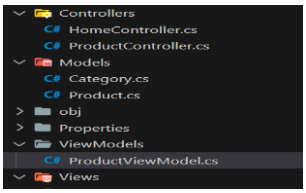
```
@model Product

<!DOCTYPE html>
<html leng="en">
    <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Product</title>
    </head>
    <body>

        @* ViewData *@
        <h1>@ViewData["Category"]</h1>
        <h4>@(((Product)ViewData["Product"]).Name)</h4>
        <h4>@(((Product)ViewData["Product"]).Price)</h4>
        <h4>@(((Product)ViewData["Product"]).Description)</h4>

        @* ViewBag *@
        <h1>@ViewBag.Category</h1>
        <h4>@ViewBag.Product.Name</h4>
        <h4>@ViewBag.Product.Price</h4>
        <h4>@ViewBag.Product.Description</h4>

        @* Model *@
        <h1>@ViewBag.Category</h1>
        <h4>@Model.Name</h4>
        <h4>@Model.Price</h4>
        <h4>@Model.Description</h4>

    </body>
</html>
```

- Html'e çekmek için ise yukarıdaki gibi yapılmaktadır.
- ViewData ile ürün çekerken (((Tip)ViewData[ürün]).Altkısım) şeklinde tanımlanır. Bu bölümün ilk başındaki _ViewImports.cshtml işlemini yapmasaydık Product yerine [shopapp.webui.Models.Product] yazmamız gerekiyordu.
- ViewBag kullanımı 20,21,22.satırlarda aşağıda gösterilmiştir.
- Model kullanımında ise 1. Satırda model Product diyip 26,27,28,29. Satırlarda model ile gösterim yapıldı. 26.satırda normale model ile Category çekemiyoruz çünkü gönderdiğimiz nesne içerisinde Category bilgileri yoktu. O yüzden viewbag ile gösterdik.

ViewModel



- Normalde product çekebiliyorduk. Şimdi bunları gruplayıp ViewModel yapıp sayfada gösterelim.



- Models altında Category.cs ve ViewModels adında bir klasör ile ProductViewModel.cs oluşturuldu.

Models/Category.cs

```
public class Category{
    1 reference
    public string Name { get; set; }
    1 reference
    public string Description { get; set; }
}
```

ViewModels/ProductViewModel.cs

```
public class ProductViewModel
{
    1 reference
    public List<Product> Products { get; set; }
    1 reference
    public Category Category { get; set; }
}
```

- Yukarıdaki category modeli oluşturduk. Product da daha önce yapmıştık. ProductViewModel modeli Product ve Category bilgilerini paketleyip göndermek için oluşturduk.

Controllers/ProductController.cs

```
public IActionResult List (){
    var products = new List<Product>(){
        new Product{Name="Iphone 8", Price= 8000, Description="iyi telefon"},
        new Product{Name="Iphone x", Price= 9000, Description="cok iyi telefon"}
    };

    var category = new Category(){Name="Telefonlar",Description="Telefon kategorisi"};

    var productViewModel = new ProductViewModel(){
        Category = category,
        Products = products
    };
    return View(productViewModel);
}
```

- 34. Satırda products nesnesi oluşturuldu.
- 39.satırda category nesnesi oluşturuldu.
- 41.satırda productViewModel nesnesi oluşturuldu ve içerisine 34 ve 39. Satırlarda oluşturduğumuz nesneleri ekledik.
- 45.satırda productViewModel nesnesini list sayfasına gönderdik.

Views/Product/List.cshtml

```
@model ProductViewModel

<!DOCTYPE html>
<html leng="en">
    <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Product</title>
    </head>
    <body>
        <h1>@Model.Category.Name</h1>
        <div>
            <p>@Model.Products[0].Name</p>
            <p>@Model.Products[0].Price</p>
            <p>@Model.Products[0].Description</p>
        </div>
        <div>
            <p>@Model.Products[1].Name</p>
            <p>@Model.Products[1].Price</p>
            <p>@Model.Products[1].Description</p>
        </div>
    </body>
</html>
```

- 1. Satır ile ProductViewModel'i çektik. 12 – 21. Satırlar arasında kullanıldığında Model içerisinden Product veya category'i çekip ona göre yerleştirdik.

Razor

Models/Product.cs

```
public class Product
{
    6 references
    public string Name { get; set; }
    6 references
    public int Price { get; set; }
    6 references
    public string Description { get; set; }
    3 references
    public bool IsApproved { get; set; }
}
```

- 13.satır eklendi. Ürünün satışta olup olmadığını döndermektedir.

Controllers/ProductController.cs

```
public IActionResult List (){
    var products = new List<Product>(){
        new Product{Name="Iphone 8", Price= 6000, Description="iyi telefon", IsApproved=false},
        new Product{Name="Iphone x", Price= 7000, Description="iyi telefon", IsApproved=true},
        new Product{Name="Iphone 11", Price= 8000, Description="iyi telefon", IsApproved=true},
        new Product{Name="Iphone 12", Price= 9000, Description="çok iyi telefon"}
    };

    var category = new Category(){Name="Telefonlar",Description="Telefon kategorisi"};

    var productViewModel = new ProductViewModel(){
        Category = category,
        Products = products
    };
    return View(productViewModel);
}
```

- 34 satırda 4 tane ürünümüz bulunmaktadır. 1 ve 4. Satırdaki ürünler active değildir. IsApproved true belirtilmediği takdirde false olarak kabul edilmektedir (38.satır).

Views/Product/List.cshtml

```
@model ProductViewModel

@{
    var popularclass =Model.Products.Count>2? "popular": " ";
    var products = Model.Products;
    var categoryName = Model.Category.Name;
}

<!DOCTYPE html>
<html leng="en">
    <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Product</title>
    <style>
        .popular{
            color: green;
            font-weight: 700;
        }
    </style>
    </head>
    <body>
    <h1 class="@popularclass">@categoryName</h1>
```

```
        @if(products.Count == 0){
            <p style="background-color: red;">Ürün Yok</p>
        }else{
            <ul>
                @foreach(var product in products){
                    @if(product.IsApproved){
                        <li>
                            <div>
                                <p>@product.Name</p>
                                <p>@product.Price</p>
                                <p>@product.Description</p>
                            </div>
                        </li>
                    }else{
                        <li style="background-color: red;">
                            Ürün Satista değil...
                        </li>
                    }
                }
            </ul>
        }
    </body>
</html>
```

- 3. Satır ile değişken tanımlanmaktadır. Yani cs dosyalarına yazdıklarımızı yazabiliriz.
- 4.satır ürün sayısı 2 den fazla olursa geriye popular string değeri dönecektir. 2 den fazla değilse boş dönecektir.
- 5.satırda products tanımlaması yapıldı, aşağıda model.products yazmamıza gerek yok.
- 6.satırda categoryname tanımlandı.
- 24. Satırda h1 classına populerclass değişkeni gelmektedir. Yani ürün sayısı 2 den fazla ise popular yazılacaktır.
- 26.satırda eğer ürün sayısı 0'a eşitse ürün yok yazmakta, 28.satırda ürün varsa else içerisinde yazıldı.
- 30.satır ile bir foreach döngüsü yazıldı ve ürünler yazdırılacak.
- 31. Satırda IsApproved true olanlar yazılacak, 39.satır ile IsApproved false olanlarda ise ürün satışta değil yazacak.

Tasarım Hazırlanması

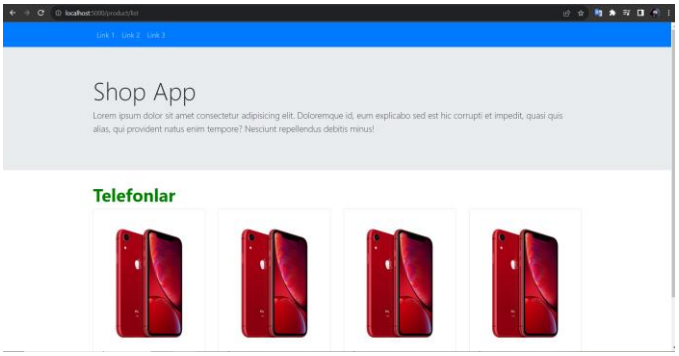
Views/Product/List.cshtml

```
@model ProductViewModel

@{
    var popularclass =Model.Products.Count>2? "popular": " ";
    var products = Model.Products;
    var categoryName = Model.Category.Name;
}

<!DOCTYPE html>
<html leng="en">
    <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha3
    <title>Product</title>
    <style>
        .popular{
            color: green;
            font-weight: 700;
        }
    </style>
    </head>
    <body>
        <div class="navbar bg-primary navbar-dark navbar-expand-sm">
            <div class="container">
                <ul class="navbar-nav">
                    <li class="nav-item">
                        <a href="#" class="nav-link">Link 1</a>
                    </li>
                    <li class="nav-item">
                        <a href="#" class="nav-link">Link 2</a>
                    </li>
                    <li class="nav-item">
                        <a href="#" class="nav-link">Link 3</a>
                    </li>
                </ul>
            </div>
        </div>
```

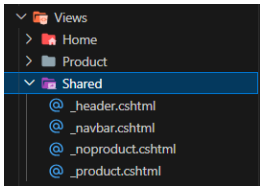
```
        <header>
            <div class="jumbotron jumbotron-fluid">
                <div class="container">
                    <h1 class="display-4">Shop App</h1>
                    <p class="lead">Lorem ipsum dolor sit amet consectetur adipisicing elit. Doloremque id, eum explicabo
                </div>
            </div>
        </header>
        <main>
            <div class="container">
                <div class="row">
                    <div class="col-md-12">
                        <h1 class="@popularclass">@categoryName</h1>
                    </div>
                </div>

                @if(products.Count ==0 )
                {
                    <div class="row">
                        <div class="col-md-12">
                            <div class="alert alert-danger">
                                Ürün yok
                            </div>
                        </div>
                    </div>
                }
                else
                {
                    <div class="row">
                        @foreach (var product in products)
                        {
                            <div class="col-md-3">
                                <div class="card">
                                    <img class="card-img-top" src="https://store.storeimages.cdn-apple.com/4668/as-images
                                    <div class="card-body">
                                        <h5 class="card-title">@product.Name</h5>
                                        <p class="card-text">@product.Description</p>
                                        <a href="" class="btn btn-primary">İncele</a>
                                    </div>
                                </div>
                            </div>
                        }
                    </div>
                }
```

```
            </div>
        </main>

    </body>
</html>
```

- Yukarıda bootsrap kütüphanesini ekleyip biraz tasarımsal olarak oynadık.
- 15.satırda bootsrap kütüphanesini ekledik.
- 25 ile 34. Satırlar arasındaki nav kısmını bütün View içerisindeki dosyalara ekledik. Bootsrap kütüphaneside aynı şekilde.
- 73.satırda fotoğraf rastgele bir siteden alınmıştır.



Partial Views



- Views içerisinde shared diye bir klasör oluşturuldu. Bu klasör içerisinde ortak sayfalarımız bulunmaktadır. Örnek navbar kısmı bütün sayfalarda olacak ve biz bir yerden değiştirdiğimizde tüm sayfalarda değişcek. Yani dinamik hale getiriyoruz.

Views/Shared/_header.cshtml

```
<header>
    <div class="jumbotron jumbotron-fluid">
        <div class="container">
            <h1 class="display-4">Shop App</h1>
            <p class="lead">Lorem ipsum dolor sit amet consectetur adipisicing elit. Doloremque id, eum explicabo sed est
        </div>
    </div>
</header>
```

Views/Shared/_navbar.cshtml

```
<div class="navbar bg-primary navbar-dark navbar-expand-sm">
    <div class="container">
        <ul class="navbar-nav">
            <li class="nav-item">
                <a href="#" class="nav-link">Link 1</a>
            </li>
            <li class="nav-item">
                <a href="#" class="nav-link">Link 2</a>
            </li>
            <li class="nav-item">
                <a href="#" class="nav-link">Link 3</a>
            </li>
        </ul>
    </div>
</div>
```

Views/Shared/_noproduct.cshtml

```
<div class="row">
    <div class="col-md-12">
        <div class="alert alert-danger">
            Ürün yok
        </div>
    </div>
</div>
```

Views/Shared/_product.cshtml

```
@model Product

<div class="card">
    <img class="card-img-top" src="https://store.storeimages.cdn-apple.com/4668/as-images.apple.com/is/iphone-xr-
    <div class="card-body">
        <h5 class="card-title">@Model.Name</h5>
        <p class="card-text">@Model.Description</p>
        <a href="" class="btn btn-primary">İncele</a>
    </div>
</div>
```

- Yukarıda sayfalarımızın bölümlerini farklı dosyalara topladık.
- _product dosyasında card içerisindeki özelliklerimiz bize dışarıdan geliyordu normalde. Şimdi 1. Satırda model Product yazdık ve modelde yerine yazdık. Bunu bir fonksiyon parametresi olarak düşünebilirsiniz, sayfamıza yerleştirdiğimizde parametre olarak göndereceğiz.

Views/_ViewImports.cshtml

```
Views > @ _ViewImports.cshtml
1   @addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
2   @using shopapp.webui.Models
```

- 1. Satıra bir kütüphane eklendi. Daha sonra kullanıldığı yer gösterilecek.

Views/Product/List.cshtml

```
24      <body>
25          @await Html.PartialAsync("_navbar")
26          <partial name="_header">
27          <main>
28              <div class="container">
29                  <div class="row">
30                      <div class="col-md-12">
31                          <h1 class="@popularclass">@categoryName</h1>
32                      </div>
33                  </div>
34
35                  @if(products.Count ==0 )
36                  {
37                      @await Html.PartialAsync("_noproduct")
38                  }
39                  else
40                  {
41                      <div class="row">
42                          @foreach (var product in products)
43                          {
44                              <div class="col-md-3">
45                                  @await Html.PartialAsync("_product",product)
46                              </div>
47                          }
48                      </div>
49                  }
50              </div>
51          </main>
52      </body>
```

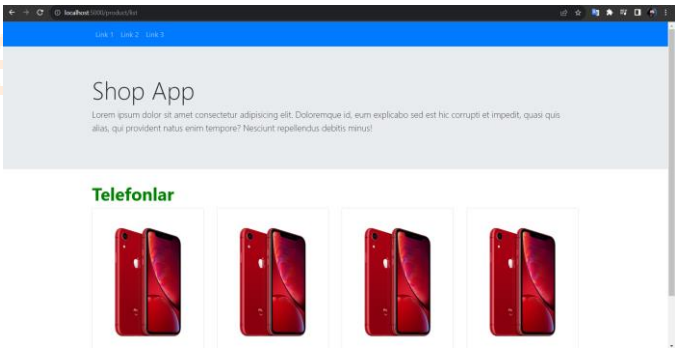
- Bu dosya içerisinde yalnızca body içerisi değiştirildi.
- 25. Satır ile navbar dosyasına ulaşıp alabiliyoruz. Asp Net Core, View klasörü içerisinde shared klasörü altında _navbar dosyasına kendisi bakıyor. Yani _navbar klasörü içerisindeki kodu getirir.
- 26.satırdaki kullanım await'e altarnatif olarak yazıldı.

  _ViewImports dosyası içerisindeki 1.satır kullanabilmemizi sağladı. Header kodlarını getirmektedir.
- 37.satır ürün olmadığı durumda alertli noproduct dosyasının kodlarını getirmektedir.
- 45.satırda product dosyası getirilmektedir. Bu dosyadan bahsederken bir parametreden bahsetmiştik. 45. Satırın ("dosya",parametre) olarak düşünebilirsiniz.

Views/Product/Details.cshtml

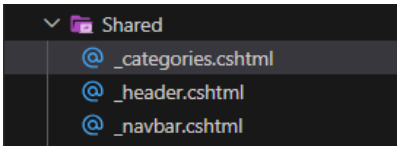
```
@model shopapp.webui.Models.Product

<!DOCTYPE html>
<html lang="en">
    <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Product</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-Vkoo8x4CGsO
    </head>
    <body>
        @await Html.PartialAsync("_navbar")
        <p>Urun Adi: @Model.Name</p>
        <p>Urun Fiyati: @Model.Price</p>
        <p>Urun Aciklamasi: @Model.Description</p>
    </body>
</html>
```

- Details sayfasındaki gibi 13.satırdaki navbar kodunu diğer sayfalara da ekleyin.



- Yukarıda görüldüğü gibi hiçbir değişiklik olmadı. Şimdi navbar üzerinde yapılacak bir değişiklik bütün sayfalarda yapılacak. Yani details içersine gidip bir şey değiştirmeme gerek yok.

Partial Views'e Veri Aktarımı



- _categories dosyasını ekleyelim.

Views/Shared/_categories.cshtml

```
@model List<Category>

<div class="list-group">
    @foreach (var category in categories){
        <a href="#" class="list-group-item list-group-item-action">
            @category.Name
        </a>
    }
</div>
```

- Category'i de bir liste haline getirdik. Product'a yaptığımız gibi.

ViewModels/ProductViewModel.cs

```
public class ProductViewModel
{
    2 references
    public List<Product> Products { get; set; }
    2 references
    public List<Category> Categories { get; set; }
}
```

- 7.satırı liste haline getirelim. Category değil Categories.

Controllers/ProductController.cs

```
public IActionResult List (){
    var products = new List<Product>(){
        new Product{Name="İphone 8", Price= 6000, Description="iyi telefon", IsApproved=false},
        new Product{Name="İphone x", Price= 7000, Description="iyi telefon", IsApproved=true},
        new Product{Name="İphone 11", Price= 8000, Description="iyi telefon", IsApproved=true},
        new Product{Name="İphone 12", Price= 9000, Description="cok iyi telefon"}
    };

    var categories = new List<Category>(){
        new Category{Name="Telefon",Description="Telefon kategorisi"},
        new Category{Name="Bilgisayar",Description="Bilgisayar kategorisi"},
        new Category{Name="Elektronik",Description="Elektronik kategorisi"},

    };

    var productViewModel = new ProductViewModel(){
        Categories = categories,
        Products = products
    };
    return View(productViewModel);
}
```

- 41.satırdaki gibi liste oluşturalım. 49. Satırı güncelleyelim.

Views/Product/List.cshtml

```
@model ProductViewModel

@{
    var popularclass =Model.Products.Count>2? "popular": " ";
    var products = Model.Products;
    var categories = Model.Categories;
}
```

```
<body>
    @await Html.PartialAsync("_navbar")
    <partial name="_header">
    <main>
        <div class="container">
            <div class="row">
                <div class="col-md-3">
                    @await Html.PartialAsync("_categories",categories)
                </div>
                <div class="col-md-9">
                    @if(products.Count ==0 )
                    {
                        @await Html.PartialAsync("_noproduct")
                    }
                    else
                    {
                        <div class="row">
                            @foreach (var product in products)
                            {
                                <div class="col-md-3">
                                    @await Html.PartialAsync("_product",product)
                                </div>
                            }
                        </div>
                    }
                </div>
            </div>
        </div>
    </main>
</body>
```

- Bu satırları güncelleyelim. Kategoriler sol tarafta 3 bölüm kapsayacak, ürünler 9 bölüm kapsayacak.

- Şimdi home anasayfamız içinde aynılarını yapalım.

Controllers/HomeController.cs

```
public IActionResult Index (){
    var products = new List<Product>(){
        new Product{Name="İphone 8", Price= 6000, Description="iyi telefon", IsApproved=false},
        new Product{Name="İphone x", Price= 7000, Description="iyi telefon", IsApproved=true},
        new Product{Name="İphone 11", Price= 8000, Description="iyi telefon", IsApproved=true},
        new Product{Name="İphone 12", Price= 9000, Description="cok iyi telefon"}
    };

    var categories = new List<Category>(){
        new Category{Name="Telefon",Description="Telefon kategorisi"},
        new Category{Name="Bilgisayar",Description="Bilgisayar kategorisi"},
        new Category{Name="Elektronik",Description="Elektronik kategorisi"},

    };

    var productViewModel = new ProductViewModel(){
        Categories = categories,
        Products = products
    };
    return View(productViewModel);
}
```

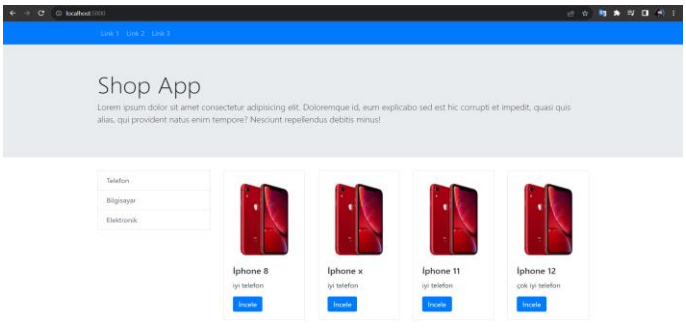
- ProductController içerisindeki index kısmını aynen alıp yapıştıralım.

Views/Home/index.cshtml

```
@model ProductViewModel

@{
    var popularclass =Model.Products.Count>2? "popular": " ";
    var products = Model.Products;
    var categories = Model.Categories;
}

<!DOCTYPE html>
<html leng="en">
    <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="
    <title>Product</title>
    <style>
        .popular{
            color: green;
            font-weight: 700;
        }
    </style>
    </head>
    <body>
        @await Html.PartialAsync("_navbar")
        <partial name="_header">
```

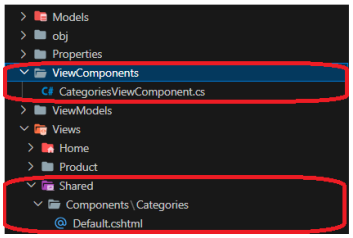
```
        <main>
            <div class="container">
                <div class="row">
                    <div class="col-md-3">
                        @await Html.PartialAsync("_categories",categories)
                    </div>
                    <div class="col-md-9">
                        @if(products.Count ==0 )
                        {
                            @await Html.PartialAsync("_noproduct")
                        }
                        else
                        {
                            <div class="row">
                                @foreach (var product in products)
                                {
                                    <div class="col-md-3">
                                        @await Html.PartialAsync("_product",product)
                                    </div>
                                }
                            </div>
                        }
                    </div>
                </div>
            </div>
        </main>
    </body>
</html>
```

- Views/Product/List.cshtml dosyasını kopyalayıp aynı şekilde yapıştıralım.
- Şimdi dikkat edilirse controllers içerisinde 2 kere aynı işlemi yaptık. Yani product veya categoriesler tekrarlandı. Bir bölüm sonra onlarıda dinamik hale getirelim.



- "/home", "/home/index", "/", "/product/list" gibi sayfalar aynı görünüm aldı. İlerde değiştirlecektir.

View Component



- ViewComponents isimde bir klasör oluşturup, sağ click ile new class oluşturup ismi CategoriesViewComponent oluşturuldu.
- Views/Shared/Components/Categories/ altında Default.cshtml dosyası oluşturuldu.

ViewComponents/CategoriesViewComponent.cs

```
namespace shopapp.webui.ViewComponents
{
    0 references
    public class CategoriesViewComponent:ViewComponent
    {
        0 references
        public IViewComponentResult Invoke(){
            var categories = new List<Category>(){
                new Category{Name="Telefon",Description="Telefon kategorisi"},
                new Category{Name="Bilgisayar",Description="Bilgisayar kategorisi"},
                new Category{Name="Elektronik",Description="Elektronik kategorisi"},

            };
            return View(categories);
        }
    }
}
```

- Category yapısı bu dosya içerisine alındı. 18.satır ile geri gönderilme yapılmaktadır.

Views/Shared/Components/Categories/Default.cshtml

```
@model List<Category>

<div class="list-group">
    @foreach (var category in Model){
        <a href="#" class="list-group-item list-group-item-action">
            @category.Name
        </a>
    }
</div>
```

- Component yapımız yukarıda görüldüğü gibidir. Bu şekilde paketlenen category yapısı site içerisinde rahat bir şekilde gerektiği yerde çağrılacaktır.

Views/Home/index.cshtml

```
<div class="container">
    <div class="row">
        <div class="col-md-3">
            @await Component.InvokeAsync("Categories")
        </div>
```

- 30.satırdaki gibi sayfaya çağrılmaktadır. Bu satır ile Views/Shared/Components/Categories/ altında Default.cshtml dosyası çağrılmaktadır.
- Klasör isimleri yukarıdaki gibi olmalıdır. 5.satırdaki categories kısmınıda silebilirsiniz.

Views/Product/List.cshtml

```
<div class="row">
    <div class="col-md-3">
        @await Component.InvokeAsync("Categories")
    </div>
```

- Product içerisindeki list.cshtml içinde aynı işlem yapılmaktadır.
- 5.satırdaki categories kısmınıda silebilirsiniz.

ViewModels/ProductViewModel.cs

```
public class ProductViewModel
{
    2 references
    public List<Product> Products { get; set; }
}
```

- Category kısmını kaldırabiliriz artık.

Controllers/HomeController.vs

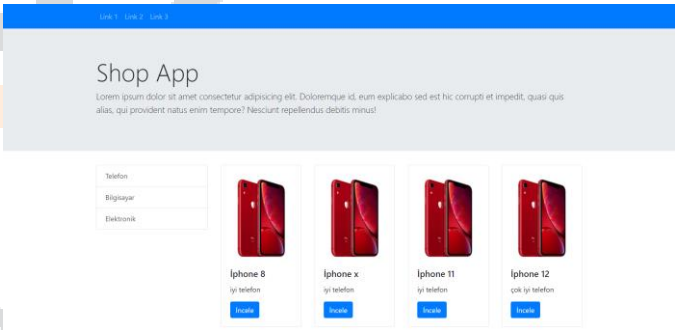
```
public class HomeController:Controller
{
    //localhost:5000/home/index
    0 references
    public IActionResult Index (){
        var products = new List<Product>(){
            new Product{Name="İphone 8", Price= 6000, Description="iyi telefon", IsApproved=false},
            new Product{Name="İphone x", Price= 7000, Description="iyi telefon", IsApproved=true},
            new Product{Name="İphone 11", Price= 8000, Description="iyi telefon", IsApproved=true},
            new Product{Name="İphone 12", Price= 9000, Description="çok iyi telefon"}
        };

        var productViewModel = new ProductViewModel(){
            Products = products
        };
        return View(productViewModel);
    }
```

- Bu kısımdan categories de kaldırıldı.
- ProductController.cs kısmından da kaldırabilirsiniz.

Views/Shared/Components/Categories/Default.cshtml

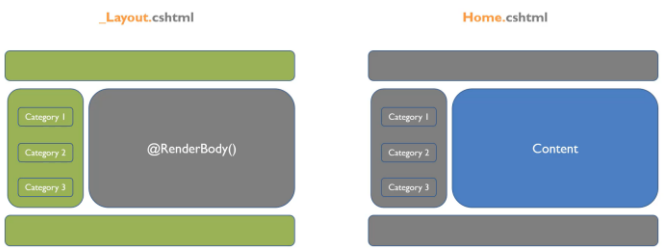
```
@model List<Category>

<div class="list-group">
    @foreach (var category in Model){
        <a href="#" class="list-group-item list-group-item-action">
            @category.Name
        </a>
    }
</div>
```

- Belirtilen yerin Model olduğuna dikkat edin.

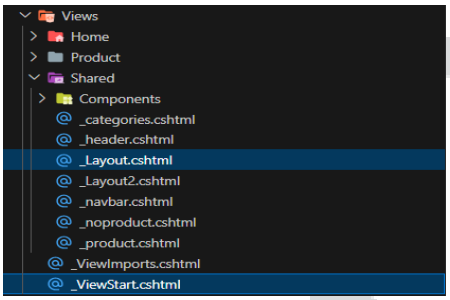


- Görüldüğü üzere bir sorun yok şuanda.
- Yani genel olarak yapılan işlem Controller içerisindeki categories yapısı CategoriesViewComponent içerisinde aktarıldı. Daha sonra category componenti default.cshtml içinde oluşturulup index.cshtml ve list.cshtml içerisine çağrıldı. Daha sonra category bilgileri silindi.

Layout



- Yukarıda bir Layout sayfası oluşturacağız. Bu sayfa projenin ana yapısını temsil edecek ve biz her sayfa için yeşil ile belirtilen yerleri değiştirmeyeceğiz. Sadece @RenderBody() kısmına Content kısımları gelcek.



- Yukarıda tanımlanan dosyalar eklendi. _Layout ve _Layout2 dosyaları Shared içerisine, ViewStart ise Views klasörü içine eklendi.

Views/Shared/_Layout.cshtml

```
<!DOCTYPE html>
<html leng="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
        <title>Product</title>
    </head>
    <body>
        @await Html.PartialAsync("_navbar")
        <partial name="_header">

        <main>
            <div class="container">
                @RenderBody();
            </div>
        </main>

    </body>
</html>
```

- Genel yapımızı bu şekilde aldık, RenderBody kısmına diğer dosyalar gelcek.

Views/Home/index.cshtml

```
@model ProductViewModel

@{
    Layout="_Layout";
}

@{
    var popularclass =Model.Products.Count>2? "popular": " ";
    var products = Model.Products;
}


<div class="row">
    <div class="col-md-3">
            @await Component.InvokeAsync("Categories")
    </div>
    <div class="col-md-9">
        @if(products.Count ==0 )
        {
            @await Html.PartialAsync("_noproduct")
        }
        else
        {
            <div class="row">
                @foreach (var product in products)
                {
                    <div class="col-md-3">
                        @await Html.PartialAsync("_product",product)
                    </div>
                }
            </div>
        }
    </div>
</div>
```

- 3. Satırdaki gibi bir layout tanımlaması yapıldığında ilgili sayfa _Layout dosyası içerisindeki RenderBody 'e yerleştiriliyor.

Views/_ViewStart.cshtml

```
@{
    Layout="_Layout";
}
```

- Bu şekilde tanımlama yapıldığında her sayfada bu tanımlama yapmaya gerek kalmayacaktır.
- Peki farklı bir _Layout nasıl oluştururum denildiğinde _Layout2 dosyasını o yüzden oluşturduk ve şimdi ayarlarına bakalım.

Views/Shared/_Layout2.cshtml

```
<!DOCTYPE html>
<html leng="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
        <title>Product</title>
    </head>
    <body style="background-color: blue;">
        @await Html.PartialAsync("_navbar")
        <partial name="_header">

        <main>
            <div class="container">
                @RenderBody();
            </div>
        </main>

    </body>
</html>
```

- Bu normalden farklı olarak body background'ı mavi yaptık. (10.str)

Views/Product/List.cshtml

```
@model ProductViewModel

@{
    Layout="_Layout2";
}

@{
    var popularclass =Model.Products.Count>2? "popular": " ";
    var products = Model.Products;
}

<div class="row">
    <div class="col-md-3">
            @await Component.InvokeAsync("Categories")
    </div>
    <div class="col-md-9">
        @if(products.Count ==0 )
        {
            @await Html.PartialAsync("_noproduct")
        }
        else
        {
            <div class="row">
                @foreach (var product in products)
                {
                    <div class="col-md-3">
                        @await Html.PartialAsync("_product",product)
                    </div>
                }
            </div>
        }
    </div>
</div>
```

- 4.satırda _Layout2 ye gönderilmiştir.

- Yukarı menü rengini değiştirmek için _navbar.cshtml sayfasına gidip bg-primary kısmını bg-danger ile değiştirin.

Views/Product/index.cshtml

```
<div class="row">
    <div class="col-md-3">
            @await Component.InvokeAsync("Categories")
    </div>
    <div class="col-md-9">
        About
    </div>
</div>
```

- Home içerisindeki About,MyView dosyalarını ve Product içerisindeki Details,İndex dosyalarını yukarıdaki gibi yapın.

Section

Views/Home/index.cshtml

```
@model ProductViewModel

@{
    var popularclass =Model.Products.Count>2? "popular": " ";
    var products = Model.Products;
}

@section MessageBox{
    <div class="alert alert-warning text-center m-0">
        You have a message..!
    </div>
}

@section Scripts{
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha38
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha
}


@if(products.Count ==0 )
{
    @await Html.PartialAsync("_noproduct")
}
else
{
    <div class="row">
        @foreach (var product in products)
        {
            <div class="col-md-3">
                @await Html.PartialAsync("_product",product)
            </div>
        }
    </div>
}
```

- 8.satır da bir section message box tanımlandı. Bu kısmı anasayfanın belirli kısımlarında göstermek için kullanacağız.
- 14.satırda scripts alanı bulunmaktadır. Bu kısım ile bootsrap scriptlerini sayfaya gönderiyoruz.

Views/Shared/_Layout.cshtml

```
<!DOCTYPE html>
<html leng="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
        <title>Product</title>
    </head>
    <body>

        @if(IsSectionDefined("MessageBox")){
            @RenderSection("MessageBox",false)
        }else{
            <div class="alert alert-success text-center m-0">
                Welcome..!
            </div>
        }

        @await Html.PartialAsync("_navbar")
        <partial name="_header">
```

```
        <main>
            <div class="container">
                @if(IsSectionDefined("Categories")){
                    <div class="row">
                        <div class="col-md-3">
                            @RenderSection("Categories",false)
                        </div>
                        <div class="col-md-9">
                            @RenderBody();
                        </div>
                    </div>
                }else{
                    <div class="row">
                        <div class="col-md-12">
                            @RenderBody();
                        </div>
                    </div>
                }
            </div>
        </main>

        <script src="https://code.jquery.com/jquery-3.4.1.min.js" integrity="sha384-J6qa4849blE2+poT4WnyKhv

        @RenderSection("Scripts", false)

    </body>
</html>
```

- 12.satırda eğer ilgili sayfada MessageBox sectionu varsa oradaki mesaj ekrana yazılır. 14.satır ile yoksa welcome yazar.
- 13.satırda RenderSection içerisinde 2. Parametre olarak false verebiliriz. Bu zorunlu olarak bütün sayfalarda yaptırmaz.
- 25.satırda eğer categories diye bir section varsa sayfayı 3'e 9 şeklinde böler, eğer yoksa 12 bölüm (Satırın hepsini) kapsar. Bundan dolayı diğer kısımlardan row col satırlarını kaldırmalıyız.
- 44.satırda jquery her sayfada kullanılabileceğinden yazıldı, 46. Satırda ekstra scriptler gönderilir ve sayfanın alt kısmına yazılır.
- Layout sayfasında ilgili RenderSection nereye yazıldığı önemlidir. O kısma ekleme yapar, lakin section olarak gönderilen sayfada section nerede yazıldığının önemi yoktur.

Views/Product/List.cshtml

```
@model ProductViewModel

@{
    var popularclass =Model.Products.Count>2? "popular": " ";
    var products = Model.Products;
}

@section Categories{
    @await Component.InvokeAsync("Categories")
}


@if(products.Count ==0 )
{
    @await Html.PartialAsync("_noproduct")
}
else
{
    <div class="row">
        @foreach (var product in products)
        {
            <div class="col-md-3">
                @await Html.PartialAsync("_product",product)
            </div>
        }
    </div>
}
```

- 8.satırda Categories sectionu gönderildiği için sayfa 3,9 bölünür.

Views/Home/About.cshtml

```
@section Categories{
    @await Component.InvokeAsync("Categories")
}

About
```

Views/Home/MyView.cshtml

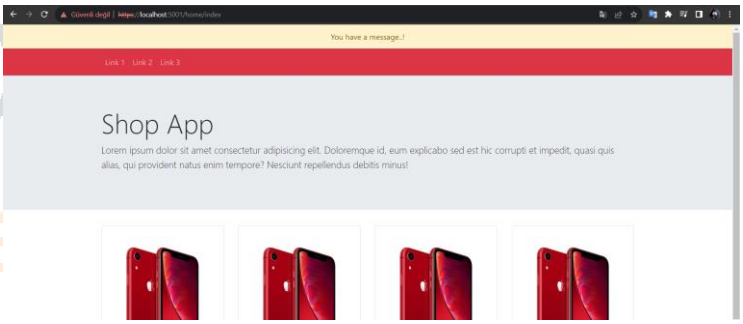
```
Views > Home > MyView.cshtml

Aslinda Myview.cshtml dosyasi <br>
home/Contact
```

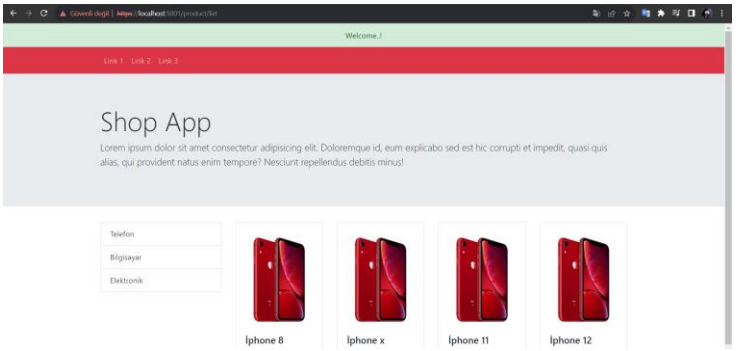
Views/Product/Details.cshtml

```
@section Categories{
    @await Component.InvokeAsync("Categories")
}

Details
```

Views/Product/Index.cshtml

```
@section Categories{
    @await Component.InvokeAsync("Categories")
}

Product Index
```

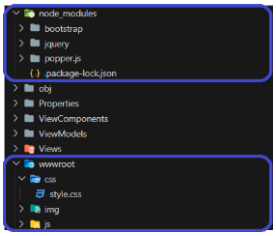




- Yukarıda home/index ve product/list sayfaları görüntülenmektedir. Yukarıdaki mesaj kısmının veya ürünlerin bulunduğu kısmın parçalı veya tam şeklinde olduğu görünebilir.

Static Files

- "**npm init –yes**" terminale girin.
- "**npm install** bootstrap@4.1.1" terminale girin.



- Terminale yukarıdaki komutları girdiğinizde node_modules klasörü yüklenecektir.
- Wwwroot diye bir klasör oluşturup içerisine css,img,js klasörlerini oluşturalım. Css içerisinde bir style.css dosyası, img içerisine 1.jpg fotoğraf dosyası yükleyelim.

wwwroot/css/style.css

```css
body {
    background-color: brown;
}
```

- Style dosyasında body kısmının background rengini kahverengi yapalım.

Startup.cs

```
26      public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
27      {
28          app.UseStaticFiles(); // wwwroot                                    1
29
30          app.UseStaticFiles(new StaticFileOptions{
31              FileProvider = new PhysicalFileProvider(
32                  Path.Combine(Directory.GetCurrentDirectory(),"node_modules")   2
33              ),
34              RequestPath="/modules"
35          });
36
37          if (env.IsDevelopment())
38          {
39              app.UseDeveloperExceptionPage();
40          }
41          app.UseRouting();
```

- Yukarıda 1.bölüm ile otomatik olarak wwwroot klasörünü arar.
- 2.bölüm ile farklı bir dosya yolu belirtmek istersek 32. Satırın sonunda klasör ismini, 34.satırla html dosyasında hangi isimle çağırmak istediğimizi yazıyoruz.

View/Shared/_product.cshtml

```
1   @model Product
2
3   <div class="card">
4       <img class="card-img-top" src="~/img/1.jpg" alt="">
5       <div class="card-body">
6           <h5 class="card-title">@Model.Name</h5>
7           <p class="card-text">@Model.Description</p>
8           <a href="" class="btn btn-primary">İncele</a>
9       </div>
10  </div>
```

- 4.satırda src kısmında yolu belirtiyoruz. Yani wwwroot dosyası içerisinde img içerisinde 1.jpg dosyasını getiriyoruz.
- ~ işareti ile img klasörü dinamik hale gelmektedir. Yani img klasörü yeri değiştirildiğinde dahi sunucu bu klasörü bulabilmekte.

View/Shared/_Layout.cshtml

```
2   <html lenng="en">
3       <head>
4           <meta charset="UTF-8">
5           <meta name="viewport" content="width=device-width, initial-scale=1.0">
6           <meta http-equiv="X-UA-Compatible" content="ie=edge">
7           <link rel="stylesheet" href="/modules/bootstrap/dist/css/bootstrap.min.css" crossorigin="anonymous">
8           <link rel="stylesheet" href="~/css/style.css">
9           <title>Product</title>
10      </head>
```

- 7.satır ile node_modules içerisinden bootstrap css dosyası getirildi
- 8.satır ile wwwroot içerisinden style.css dosyası getirildi.