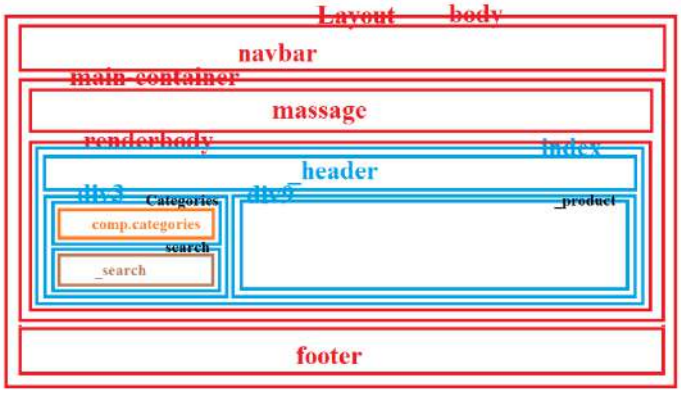
ASP.NET CORE-MVC

İçindekiler

YAYINDAN SONRA KARŞILAŞILAN SORUNLAR VE ÇÖZÜMLERİ

Tasarımlarda Sayfa Yapılarının Anlaşılması

Sayfaların içerisine yeni bir eklenti yapmaya çalıştığımızda çok sayıda View sayfasının iç içe tasarlandığını görürüz. Bu tür tasarımlarda hangi sayfa hangi sayfanının içerisinde yer alıyor ve yerleşimleri nasıldır göz önüne getirip canlandırabilmek gerekiyor. Böyle bir işi hızlandırmak için şu şekilde şablonlar tutup yerleşimlerini daha hızlı anlayabiliriz. Örneğin Home>Index.cs sayfasının yerleşimi şu şekildedir.



Asp.Net MVC AddModelError Kullanımı-(Formlarda hem üsste hemde satırda Validation kullanımı)

(https://cmengcompany.wordpress.com/2016/04/25/asp-net-mvc-addmodelerror-kullanimi/)
Asp.Net MVC'de c# tarafında post edilen modelde ,hatalı propertyler varsa bu propertylerin validationmessage() kısmında gözükmesinş istediğiniz hata mesajlarını AddModelError ile yapabilirsiniz.
Örneğin ;modelinizde DateTime türünde,"DateOfBirth" adında bir property var.Post edilen methodda bu tarih 2016 dan küçükse modeli hata mesajı ile birlikte geri gönderelim.

```
[HttpPost]
public ActionResult Create([Bind(Exclude = "EmployeeID")]EmployeeModels employeeModel)
{

    if (employeeModel.DateOfBirth.Value.Year < 2016)
    {
        ModelState.AddModelError("DateOfBirth", "2016 ve sonrasını seçiniz");
        ModelState.AddModelError("", "Lütfen hataları gideriniz.");
        return View(employeeModel);
    }
```

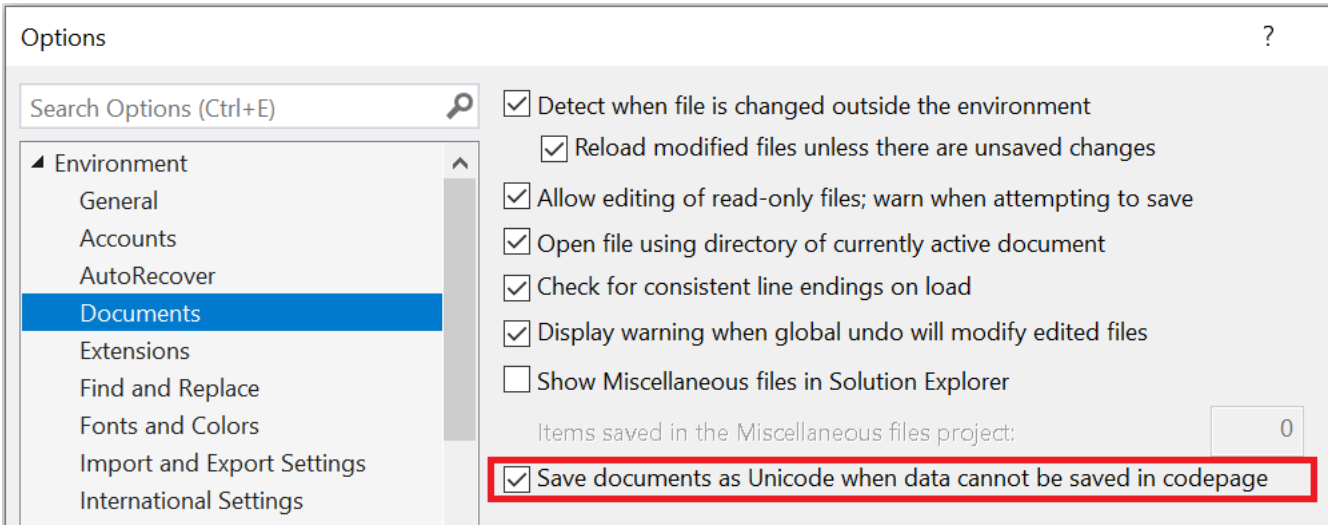
}

Yukarıda ModelState sınıfının AddModelError methodunda,ilk paremetreye property adını ,ikinci parametreye ise hata mesajını yazdık.Bu kullanımda hata mesajı view tarafında "DateOfBirth" için ValidationMessage() veya ValidationMessageFor() kullandıysak hata mesajı buralarda görüntülenecektir.

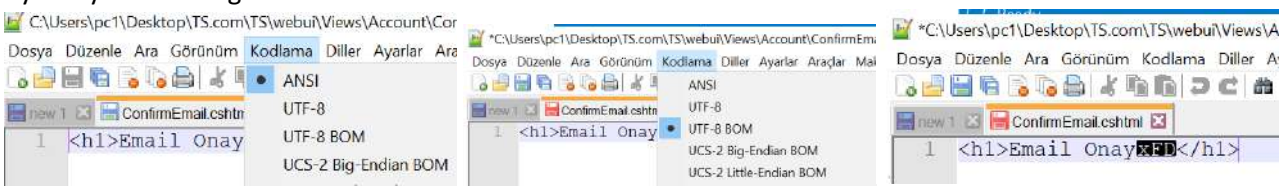
Bir alt satırda kullandığımız AddModelError methodunda ilk parametreyi boş geçtik.Böyle bir kullanımda ise yazdığımız hata mesajı,eğer view tarafında @Html.ValidationSummary ifadesini kullanmışsak, burada görüntülenecektir.

Türkçe Karakter Problemi Çözümü

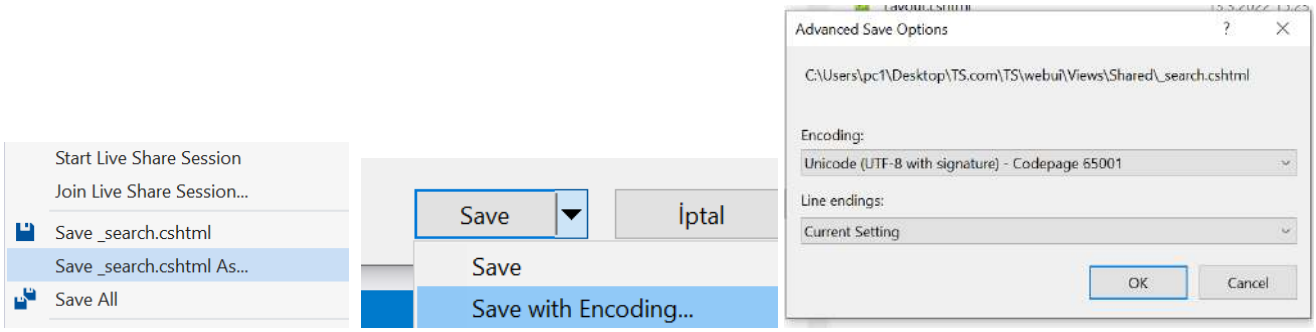
Visual Studio da daha baştan kodlama yaparken ayarlamızı UTF-8 olarak yapmamız gerekir. Bunun için Tools>Options altındaki aşağıdaki ekrandaki seçeneği kullanabiliriz.



Yada sayfalarımız ANSI olarak kaydedilmişse Not defterince (Notpad ++) açıp kodlamasını yukarıdan Utf-8 yapabiliriz. Utf-8 geçince dosyamızın içindeki Türkçe karakterlerin bozulduğunu görürüz. Bunları tek tek düzeltip öyle kaydetmemiz gerekir.



Visual Studio içinde açtığımız bir dosya Utf-8 dışında kaydedilmiş (Genellikle Windows-Türkçe seçili olabilir) bunları Save As ile alttan kodlama türünü değiştirip kaydedebiliriz.



Çoklu Dil Desteğinin Siteye Eklenmesi (Tr-En)

Controller ve Model Dosyalarında (Sınıf Dosyalarında) Dil Desteğinin Kullanılması

Günümüzde yurt dışındanda trafik almak için çoklu dil desteği neredeyse standart haline gelmiştir. Bu işlemin Asp.net core da nasıl yapıldığını öğrenelim.

ASP.NET Core içerisinde Localization desteğini (konuma ait bilgileri/Yani dil desteği) kullanabilmek için "Microsoft.AspNetCore.Mvc.Localization" sınıfına ihtiyacımız olacaktır. Bu kütüphane "Microsoft.AspNetCore.Mvc" içerisinde bulunmaktadır. Eğer eklendiyse ayrıca referans verilemesine gerek yoktur.

İlk iş olarak "**Startup.cs**" dosyası altında bulunan "**ConfigureServices**"metodu içerisinde gerekli olan sınıflarımızı kayıt edelim. Uygulama çalışmaya başladığında ilgili sınıflar burada belirtildiği şekilde oluşturulacaktır.

```
public void ConfigureServices(IServiceCollection services)
{
    //*** Çoklu dil desteği için eklendi, AddMvc() den önce eklenmiş olmalıdır.
    services.AddLocalization(options =>
    {
        // Resource (kaynak) dosyalarımızı ana dizin altında "Resources" klasorü
içerisinde tutacağımızı belirtiyoruz.
        options.ResourcesPath = "Resources";
    });

    services.AddMvc(); //***Çoklu dil desteği için eklendi
```

Kayıt işlemimizi tamamlandıktan sonra yine "**Startup.cs**" altında bulunan "**Configure**" metodu içerisinde dil desteğimizin ne şekilde çalışacağını ve hangi dillere destek vereceğimizi tanımlıyoruz. Bu metod çalışma zamanında (runtime) çalışır ve gelen HTTP istekleriyle ilgili yapılandırmalara izin verir.

Dil tanımlamalarımızı yaparken Türkçe ve İngilizce dillerine destek vereceğiz. Varsayılan dil tanımlamamız ise Türkçe olacaktır.

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env,IConfiguration
configuration,UserManager<User> userManager, RoleManager<IdentityRole> roleManager, ICartService
cartService)
{
    // Bu bölüm UseMvc()' den önce gelmelidir. Uygulamamız içerisinde destek vermemizi
istediğimiz dilleri tutan bir liste oluşturuyoruz.
    var supportedCultures = new List<CultureInfo>
    {
            new CultureInfo("tr-TR"),
            new CultureInfo("en-US"),
    };

    // SupportedCultures ve SupportedUICultures'a yukarıda oluşturduğumuz dil listesini
tanımlıyoruz. DefaultRequestCulture'a varsayılan olarak uygulamamızın hangi dil ile çalışması
gerektiğini tanımlıyoruz.
    app.UseRequestLocalization(new RequestLocalizationOptions
    {
        SupportedCultures = supportedCultures,
        SupportedUICultures = supportedCultures,
```

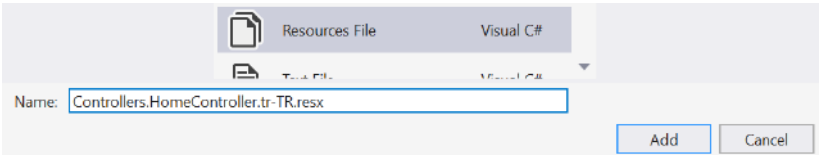
```
        DefaultRequestCulture = new RequestCulture("tr-TR")
    });
```

Startup.cs dosyası içine CultureInfo ve RequestCulture sınıfları için aşağıdaki namespace lerin ekli olduğundan emin olalım.

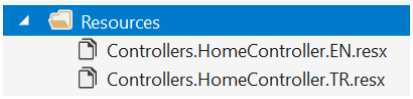
```
using System.Globalization;
using Microsoft.AspNetCore.Localization;
```

Not: Uygulamalarınızda İngilizce diline destek verecekseniz "en-US" kullanımının Amerikan İngilizcesini, "en-GB" kullanımının ise İngiliz İngilizcesini belirttiğini unutmayınız.

Dil desteğiyle ilgili tanımlamalarımızı yaptıktan sonra şimdi ilgili metinleri tutacağımız dil dosyalarımızı oluşturabiliriz. Bunun için uygulamamızın ana dizininde "**Resources**" adında yeni bir klasör oluşturuyoruz. Bu klasör adını "**Startup.cs**" dosyasındaki "**ConfigureServices**" altında belirtmiştik. İsimlendirme serbesttir. Bu klasör üzerine sağ tuşa tıklayıp "**Resources File**" ekleyelim. Bu dosyalara isim verirken hangi Controller altındaki View sayfalarında kullanılacağına göre isim verelim. Böylece isimlendirirken birbirlerinden daha bağımsız isim verebiliriz. Birinde yapacağımız bir değişiklik diğerlerini etkilememiş olur. Örneğin "Message" etiketi ile kullanacağımız bir yer için Contact sayfasında "Görüş ve Tavsiyeleriniz" yazabiliriz. aynı etiketi mail gönderme sayfası gibi bir yerde kullanırsak "Email Mesajınız" diyebiliriz.
Dil dosyalarına (Resorce) isim verirken "KlasörAdı.SinifAdi.tr-TR.resx" şeklinde isim vermeliyiz. Diğer türlü bilgileri getiren Localizer nesnesi ilgili dosyayı bulamaz.



İsimlerde biraz değişiklik yapalım. Controllers klasörü altındaki HomeController ları tek bir dosyada toplayalım.



Şimdi bu dosyaların içlerini dolduralım. Bu işlem esnasında her terim için bir Name ve birde Value değeri olacaktır. Yani kullanacağımız bir etiketin (name) içeriğin doldururken (value) TR dosyasında Türkçe karşılığını yazacağız. EN dosyasında ise ingilizce karşılığını yazacağız. Etiketlere verilen isimleri de ingilizce kullanacağız fakat etiketler büyük bir metnin daha kısa ifadesi olabilir. O nedenle EN dosyasında ikisinin de aynı olması gerekmez.

EN

Name	Value
Firstname	Firstname
Surename	Surename
Email	Email
Message	Message / Comment

TR

Name	Value
Firstname	Adınız
Surename	Soyadınız
Email	Email Adresiniz
Message	Mesaj / Görüş

Şimdi bu uygulamayı "**HomeController.cs**" altındaki Contact action metodu ve bağlı olduğu **Contact.cshtml** view sayfasında uygulayalım.

Öncelikle Controller içine Resources dosyalarından bilgileri okumak için **IStringLocalizer** sınıfını Constructor kullanarak ekleyeceğiz.

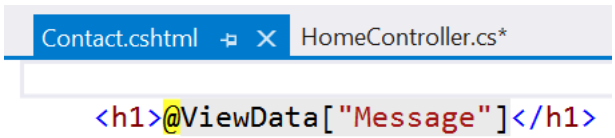
```
public class HomeController:Controller
 {
     private IProductService _productService;
     private readonly IStringLocalizer<HomeController> _localizer;
```

```
        public HomeController(IProductService productService, IStringLocalizer<HomeController>
localizer)
        {
            this._productService=productService;
            _localizer = localizer;
        }
```

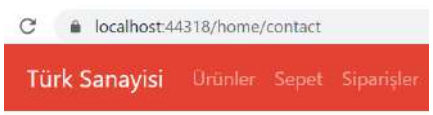
_localizer nesnemizi ürettiğimize ve bunu dependency injection yöntemi ile dolduruğumuza göre artık bu nesneyi Contact action metodu içinde kullanabiliriz.

```
public class HomeController:Controller
{
    public IActionResult Contact()
    {
        //ViewData["Message"] = _localizer["Message"]; //Bilgileri dosyadan bu şekildede
çekebiliriz.
        ViewData["Message"] = _localizer.GetString("Message");
        return View();
    }
```

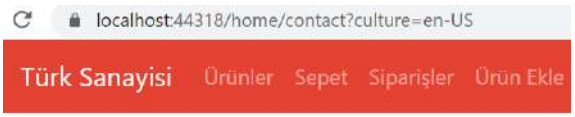
Contact.cshtml sayfasının içi



Normal şekilde çalıştırınca varsayılan dil ile geldi. Adres sonuna "?culture=En-US" şeklinde querystring eklendiğinde ingilizce versiyonları getirdi.



Mesaj / Görüş



Message / Comment

Controller altındaki sınıflara örnek olarak bir dil dosyası oluşturduk ve onu çalıştırdık. Uygulamamızın çoğu yerinde standart bir çok terim kullanırız. "Kaydet", "İptal" "Gönder" gibi uygulamamızın her alanında kullandığımız genel tanımlamalar için ise ortak bir dil dosyası (resource) kullanalım. Bunun için Models klasörü altına "SharedResources.cs" şeklinde dummy (yapmacık) bir class oluşturalım. Bu class'a karşılık yine Resources klasörü altında bir dil dosyası (resource) oluşturalım. Class "Models" klasörü altına koyduğumuzdan dil dosyamızın adını "Models.SharedResource.tr-TR.resx"şeklinde oluşturalım. Çünkü isimleri atarken önce klasör adı>Sonra sınıf adı şeklinde kullandık. Eğer farklı bir isim verirsek _lokalizer nesnesi ilgili dosyayı bulamaz.

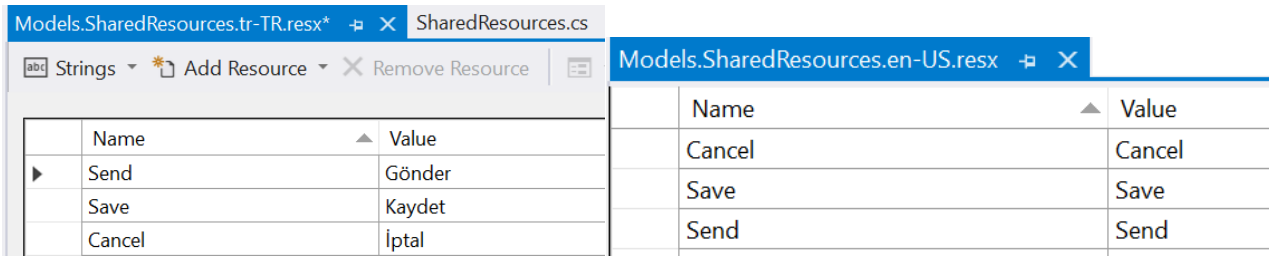
Not: Dil dosyaları içerisinde tutmuş olduğunuz metinler **HTML** formatında ise **IStringLocalizer** yerine **IHtmlLocalizer** tercih etmemiz gerekmektedir.

webui.Models> **SharedResources.cs**

```
namespace webui.Models
{
    public class SharedResources
    {
        //Bu sınıf sitenin genelinde kullanılan Dil terimlerini (Kaydet, İptal, Gönder vs)
        //kullanabilmek için dummy (yapmacık) olarak oluşturuldu.
    }
```

```
}
```

Bu sınıfa karşılık gelen Resource (dil dosyasını) oluşturalım.




```
    public class HomeController:Controller
    {
        private IProductService _productService;
        private readonly IStringLocalizer<HomeController> _localizerHomeController;
//HomeController dil seçeneklerini yüklemek için
        private readonly IStringLocalizer<SharedResources> _localizerSharedResources; //Shared
(ortak) Dil dosyalarını yüklemek için

        public HomeController(IProductService productService,
            IStringLocalizer<HomeController> localizerHomeController,
            IStringLocalizer<SharedResources> localizerSharedResources
            )
        {
            this._productService=productService;
            _localizerHomeController = localizerHomeController;
            _localizerSharedResources = localizerSharedResources;
        }

    public IActionResult Contact()
    {
        ViewData["Message"] =
_localizerHomeController.GetString("Message");//ViewData["Message"] = _localizer["Message"];
//Bilgileri dosyadan bu şekildede çekebiliriz.
        ViewData["Save"] = _localizerSharedResources.GetString("Save");
        return View();
    }
```

View dosyalarında Dil Desteğinin Kullanılması

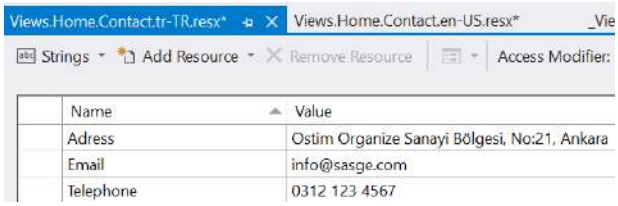
Aynı Controller sınıflarında olduğu gibi View arayüzlerinde de dil desteğini kullanabiliriz. Bunun için öncelikle "**Startup.cs**" dosyası içinde "ConfigurationServices" metodu altında AddMvc()'yi bulup sonuna aşağıdaki kodları ekliyoruz.

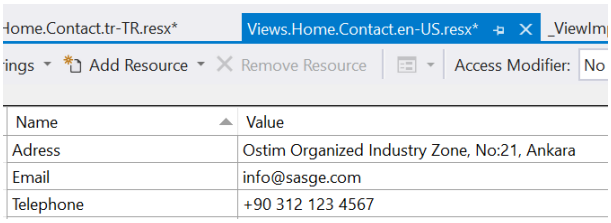
```
using Microsoft.AspNetCore.Mvc.Razor;
public class Startup
{
    public void ConfigureServices(IServiceCollection services)
    {//services.AddMvc(); //***Çoklu dil desteği için eklendi. Sadece Controllerlarda
    kullanırken bu yeterli.
    services.AddMvc().AddViewLocalization(LanguageViewLocationExpanderFormat.Suffix); //***
    çoklu dil desteği için eklendi. Hem Controller hemde View sayfalarında kullanırken bu
    şekilde kullanılacak.
```

Viewler içerisinde localizer kullanabilmek için "Views" klasörü altındaki tüm viewlerin namespacelerini tek bir yerden topladığımız "**_ViewImports.cshtml**" dosyası içine aşağıdaki "**Microsoft.AspNetCore.Mvc.Localization**" namespace'ini ekleyip inject (dependency injection) işlemi ile bir localizer nesnesi oluşturmamız gerekiyor.

```
@using Microsoft.AspNetCore.Mvc.Localization //Viewler içinde Çoklu dil desteğini kullanabilmek için eklendi
@inject IViewLocalizer Localizer //Viewler içinde çoklu dildesteğini kullanabilmek için üretilen Localizer nesnesi için inject işlemi yapılıyor.
```

Daha önce yaptığımız şekilde **Resources** klasörü altında "**Views.Home.Contact.tr-TR.resx**" ve "**Views.Home.Contact.en-US.resx**"isimlerinde iki tane dil dosyası (resource) oluşturuyoruz.





Şimdi View sayfamızın içerisinde ilgili etiketleri kullanacağımız yerleri düzenleyelim.

Views>Home>**Contact.cshmtl**

```
@*Controller altından gelen çoklu dil desteği*@
<h1>@ViewData["Message"]</h1>
@*Model altından gelen çoklu dil desteği*@
<h2>@ViewData["Save"]</h2>

@*Views içerisinde kullanılan çoklu dil desteği*@
<p>@Localizer["Address"] </p>
<p>@Localizer["Email"] </p>
<p>@Localizer["Telephone"] </p>
```

Data Annotation Yapıları Üzerinden Çoklu Dil Desteğinin Kullanılması

Data Annotations lar ile birlikte çoklu dil desteğini kullanabilmek için "**Startup.cs**" içerisindeki "**ConfigureServices**" metodunda bulunan **AddMvc()**'ye ."**AddDataAnnotationsLocalization()**" eklememiz gerekmektedir.

```
public void ConfigureServices(IServiceCollection services)
{
    //*** Çoklu dil desteği için eklendi, AddMvc() den önce eklenmiş olmalıdır.
    services.AddLocalization(options =>
    {
        // Resource (kaynak) dosyalarımızı ana dizin altında "Resources" klasorü içerisinde tutacağımızı belirtiyoruz.
        options.ResourcesPath = "Resources";
    });

    //services.AddMvc(); //***Çoklu dil desteği için eklendi. Sadece Controllerlarda kullanırken bu yeterli.
    services.AddMvc().AddViewLocalization(LanguageViewLocationExpanderFormat.Suffix); //*** çoklu dil desteği için eklendi. Hem Controller hemde View sayfalarında kullanırken bu şekilde kullanılacak.
```

```
    services.AddMvc().AddDataAnnotationsLocalization(); //*** Çoklu dil desteğini Data
Annotation larla kullanabilmek için eklendi.
```

DataAnnotations lar içinde (Formlar için oluşturulan sınıf yapıları içinde) nasıl kullanıldığına dair bir örnek buraya konulmalıdır. **** BURASI EKSİK ****

Şimdi Veritabanı üzerinden çoklu dil desteğini nasıl sağlayacağımızı inceleyelim.

Çoklu Dil Desteğinin Siteye Eklenmesi -Veritabanı Altyapısının Kullanılması(Tr-En)

Kaynak: https://www.ezzylearning.net/tutorial/asp-net-core-localization-from-database Ek kaynak: https://www.ezzylearning.net/tutorial/building-multilingual-applications-in-asp-net-core

Veritabanı üzerinden çoklu dil desteğinin sağlanması bir çok avantaj barındırabilir. Yukarıdaki gibi projeyi geliştirme aşamasında oluşturulan kaynak dosyalar kullanılırsa bu dosyaları güncellemek zordur. Her güncellemede projenin Publish edilmesine ihtiyaç olur. Bu nedenle veritabanı üzerinden çoklu dil desteğinin sağlanması daha kullanışlıdır.

Dil desteği verilerini Veritabanında tutmak, herhangi bir yerden herhangi bir zamandan güncellenmesini sağlayabilir. Basit bir arayüz kullanarak yayın esnasında düzenlemeler yapılabilir.

Ancak Veritabanından yüklenen bilgiler, yukarıda anlatılan yönteme göre daha yavaştır. Çünkü yukarıdaki yöntemde bilgiler ön belleğe yüklenir ve oradan hızlıca çalışır. Yine de Veritabanından bilgilerin getirilmesi daha avantajlıdır. Bu istikamette projemize Dil Desteğini ekleyelim.

Veritabanı Altyapısının Oluşturulması

Sayfalarda kullanılan metinleri veritabanında tutacağımızdan öncelikle tablo alt yapımızı oluşturalım. Bunun için iki temel tabloya ihtiyacımız olsun. 1 tablomuz kaç adet dil desteğini sağlayacak bunları tutan bir tablo (Languages). 2. Tablomuz ise bu tabloya bağlı olarak çalışan tüm dillerde kullanılacak olan metinleri tuttuğumuz bir tablo (StringResources). Bu tabloda her dil için kullanılacak metinlere aynı metin şeklinde anahtar bir isim verilecek. Bu isme ait value değerleride o dildeki karşılığı olacak.

Results Messages

	Id	Name	Culture
1	1	English (United States)	en-US
2	2	French (France)	fr-FR
3	3	German (Germany)	de-DE

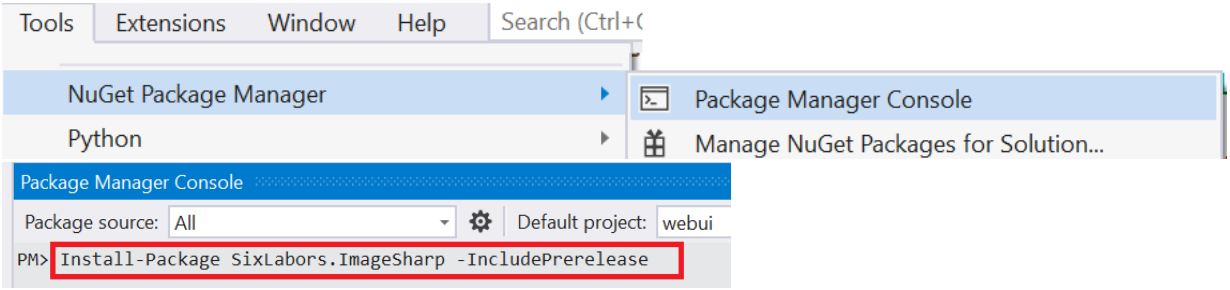
Results Messages

	Id	LanguageId	Name	Value
1	1	1	customer.page.create.title	Create Customer
2	2	1	customer.page.create.firstname	First Name
3	3	1	customer.page.create.lastname	Last Name
4	4	2	customer.page.create.title	Créer un client
5	5	2	customer.page.create.firstname	Prénom
6	6	2	customer.page.create.lastname	Nom de famille
7	7	3	customer.page.create.title	Kunden erstellen
8	8	3	customer.page.create.firstname	Vorname
9	9	3	customer.page.create.lastname	Nachname
10	10	1	general.button.create	Create
11	11	2	general.button.create	Créer
12	12	3	general.button.create	Schaffen

Resimleri Yeniden Boyutlandırıp Kaydetme (Resizing)

Öncelikle aşağıdaki Paketi PM üzerinden kuralım.

Install-Package SixLabors.ImageSharp -IncludePrerelease



Bu kütüphaneyi kurduktan sonra bir ProductEdit üzerinden resimleri yüklerken ilgili kodlar aşağıdaki şekilde kullanılacaktır.

```
using SixLabors.ImageSharp.Processing; //Resim boyutlandırma için yüklendi (Install-Package SixLabors.ImageSharp -IncludePrerelease)
using SixLabors.ImageSharp;

public async Task<IActionResult> ProductEdit(ProductModel model,int[] categoryIds,IFormFile file)
{
    //Bilgi: Asenkron programlama için şu bilgileri oku: https://atarikguney.medium.com/asenkron-asynchronous-programlama-nedir-296230121f9d
    if (ModelState.IsValid)
    {
        var entity = _productService.GetById(model.ProductId); //Veritabanındaki ürünün orjinal bilgilerini entity içine atıyor.
        if(entity==null)
        {
            return NotFound();
        }
        entity.Name = model.Name;
        entity.Price = model.Price;
        entity.Url = model.Url;
        entity.Description = model.Description;
        entity.IsHome = model.IsHome;
        entity.IsApproved = model.IsApproved;

        if(file!=null) //Eğer formdan gelen bir resim dosyası varsa
        {
            //========================================== YENİ KODLAR
            var extention = Path.GetExtension(file.FileName); //.jpg gibi Yüklenmeye çalışılan Resim dosyanın uzantısını alıyor.
            var randomName = string.Format($"{Guid.NewGuid()}{extention}"); //adlfsdlfsd.jpg şeklinde Resme rastgele bir isim atıyor, sonuna uzantısını da ekliyor. Dikkat! resim Id si kullanılmıyor. her resme farklı bir isim atanıyor.

            //Yüklenen Resimden iki tane versiyon hazırlıyor. 250x250, 750x750
            //250x250 Size*** using ifadesi ile, işimiz bitince bellekten silinecek anlamındadır.
            using (var image = SixLabors.ImageSharp.Image.Load(file.OpenReadStream())) //Dikkat: Buradaki Image sınıfı SixLabur dan gelmelidir. Drawin altında da böyle bir sınıf vardır. Karışmamalı.
            {
                var path = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot\\img\\products\\small", randomName); //C: den itibaren Resmin kaydedileceği tam fiziklsel yolu oluşturuyor. sonunda resim adı ve uzantısıda var.
                string newSize = ImageSizing(image, 250, 250); //public string ImageSizing(Image img, int MaxWidth, int MaxHeight)
                string[] sizeArray = newSize.Split(",");
                image.Mutate(x => x.Resize(Convert.ToInt32(sizeArray[0]), Convert.ToInt32(sizeArray[1]))); //Burada Resmi küçülttü.
                await image.SaveAsJpegAsync(path); //SixLabour kütüphanesi kullanılıyor.. Asenkron prgramlama. Kopyalarken bekliyor.
```

```
            }
            //750x750 Size using ifadesi ile, işimiz bitince bellekten silinecek anlamındadır.
            using (var image = SixLabors.ImageSharp.Image.Load(file.OpenReadStream())) //Dikkat:
Buradaki Image sınıfı SixLabur dan gelmelidir. Drawin altında da böyle bir sınıf vardır.
Karışmamalı.
            {
                var path = Path.Combine(Directory.GetCurrentDirectory(),
"wwwroot\\img\\products\\large", randomName);
                string newSize = ImageSizing(image, 750, 750); //public string ImageSizing(Image
img, int MaxWidth, int MaxHeight)
                string[] sizeArray = newSize.Split(",");
                image.Mutate(x => x.Resize(Convert.ToInt32(sizeArray[0]),
Convert.ToInt32(sizeArray[1]))); //Burada Resmi küçülttü.
                await image.SaveAsJpegAsync(path); //SixLabour kütüphanesi kullanılıyor..
Asenkron prgramlama. Kopyalarken bekliyor.
            }
            entity.ImageUrl = randomName;

        }

        if (_productService.Update(entity,categoryIds))
        {
                TempData.Put("message", new AlertMessage()
            {
                Title="kayıt güncellendi",
                Message="kayıt güncellendi",
                AlertType="success"
            });
            return RedirectToAction("ProductList");
        }
            TempData.Put("message", new AlertMessage()
            {
                Title="hata",
                Message=_productService.ErrorMessage,
                AlertType="danger"
            });
    }
    ViewBag.Categories = _categoryService.GetAll();
    return View(model);
}

//************************ IMAGE SIZING ****************************
public string ImageSizing(SixLabors.ImageSharp.Image img, int MaxWidth, int MaxHeight)
{
    if (img.Width > MaxWidth || img.Height > MaxHeight)
    {
        double widthRatio = (double)img.Width / (double)MaxWidth;
        double heightRatio = (double)img.Height / (double)MaxHeight;
        double ratio = Math.Max(widthRatio, heightRatio);
        int newWidth = (int)(img.Width / ratio);
        int newHeight = (int)(img.Height / ratio);
        return newWidth.ToString() + "," + newHeight.ToString();
    }
    else
    {
        return img.Width.ToString() + "," + img.Height.ToString();
    }
}
```

Entitiy Faramework Temel Veritabanı İşlemleri

Aşağıdaki program'da Entity Framework (EF) kullanarak temel Veritabanı işlemlerinin nasıl yapıldığını görebiliriz. Bu yapı içerisinde Product ve Category entityleri tanımlanmıştır. DBContext oluşturularak bu yapılar VT ye tanıtılmıştır. Daha sonra temel VT işlemleri (bulma, getirme, silme, güncelleme vs) yapılmıştır.

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Logging;

namespace ConsoleApp
{
    public class ShopContext : DbContext
    {
        public DbSet<Product> Products { get; set; }
        public DbSet<Category> Categories { get; set; }

        public static readonly ILoggerFactory MyLoggerFactory
            = LoggerFactory.Create(builder => { builder.AddConsole(); });
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder
                .UseLoggerFactory(MyLoggerFactory)
                .UseSqlite("Data Source=shop.db");
        }
    }

    public class Product
    {
        public int Id { get; set; }

        [MaxLength(100)]
        [Required]
        public string Name { get; set; }

        public decimal Price { get; set; }

    }

    public class Category
    {
        public int Id { get; set; }
        public string Name { get; set; }
    }

    class Program
    {
        static void Main(string[] args)
        {
            DeleteProduct(7);
        }

        static void DeleteProduct(int id)
        {
            using (var db = new ShopContext())
            {

                var p = new Product() { Id = 6 };

                // db.Products.Remove(p);
                db.Entry(p).State = EntityState.Deleted;
                db.SaveChanges();
```

```
            // var p = db.Products.FirstOrDefault(i=>i.Id==id);

            // if (p!=null)
            // {
            //     db.Products.Remove(p);
            //     db.SaveChanges();

            //     Console.WriteLine("veri silindi");
            // }
        }
    }

    static void UpdateProduct()
    {
        using (var db = new ShopContext())
        {
            var p = db.Products.Where(i => i.Id == 1).FirstOrDefault();

            if (p != null)
            {
                p.Price = 2400;

                db.Products.Update(p);
                db.SaveChanges();
            }
        }

        // using(var db = new ShopContext())
        // {
        //     var entity = new Product(){Id=1};

        //     db.Products.Attach(entity);

        //     entity.Price = 3000;

        //     db.SaveChanges();
        // }



        //   using(var db = new ShopContext())
        //   {
        //       // change tracking
        //       var p = db
        //                 .Products
        //                 //.AsNoTracking()
        //                 .Where(i=>i.Id==1)
        //                 .FirstOrDefault();

        //       if (p!=null)
        //       {
        //             p.Price *= 1.2m;
        //             db.SaveChanges();
        //             Console.WriteLine("güncelleme yapıldı.");

        //       }
        //   }
    }
    static void GetProductByName(string name)
    {
        using (var context = new ShopContext())
        {
            var products = context
                            .Products
                            .Where(p => p.Name.ToLower().Contains(name.ToLower()))
```

```
                        .Select(p =>
                                new {
                                    p.Name,
                                    p.Price
                                })
                        .ToList();

            foreach (var p in products)
            {
                Console.WriteLine($"name: {p.Name} price: {p.Price}");
            }

        }
    }
    static void GetProductById(int id)
    {
        using (var context = new ShopContext())
        {
            var result = context
                        .Products
                        .Where(p => p.Id == id)
                        .Select(p =>
                                new {
                                    p.Name,
                                    p.Price
                                })
                        .FirstOrDefault();



            Console.WriteLine($"name: {result.Name} price: {result.Price}");

        }
    }
    static void GetAllProducts()
    {
        using (var context = new ShopContext())
        {
            var products = context
            .Products
            .Select(p =>
                new {
                    p.Name,
                    p.Price
                })
            .ToList();

            foreach (var p in products)
            {
                Console.WriteLine($"name: {p.Name} price: {p.Price}");
            }
        }
    }
    static void AddProducts()
    {
        using (var db = new ShopContext())
        {
            var products = new List<Product>()
            {
                new Product { Name = "Samsung S6", Price=3000 },
                new Product { Name = "Samsung S7", Price=4000 },
                new Product { Name = "Samsung S8", Price=5000 },
                new Product { Name = "Samsung S9", Price=6000 }
            };
```

```
                db.Products.AddRange(products);
                db.SaveChanges();

                Console.WriteLine("veriler eklendi.");
            }
        }
        static void AddProduct()
        {
            using (var db = new ShopContext())
            {
                var p = new Product { Name = "Samsung S10", Price = 8000 };

                db.Products.Add(p);
                db.SaveChanges();

                Console.WriteLine("veriler eklendi.");
            }
        }

    }
}
```

One To Many İlişkisi Kurma

Veritabanı tabloları arasına bağlantılar kurarken Bire-Çoklu işlemleri için aşağıdaki yapı örnekleri kullanılabilir.

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Logging;

namespace ConsoleApp
{
    public class ShopContext : DbContext
    {
        public DbSet<Product> Products { get; set; }
        public DbSet<Category> Categories { get; set; }
        public DbSet<User> Users { get; set; }
        public DbSet<Address> Addresses { get; set; }
        public static readonly ILoggerFactory MyLoggerFactory = LoggerFactory.Create(builder =>
{ builder.AddConsole(); });
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder
                .UseLoggerFactory(MyLoggerFactory)

.UseMySql(@"server=localhost;port=3306;database=ShopDb;user=root;password=mysql1234;");
        }
    }

    // One to Many
    // One to One
    // Many to Many

    public class User
    {
        public int Id { get; set; }
        public string Username { get; set; }
        public string Email { get; set; }
        public List<Address> Addresses { get; set; } // navigation property
    }
```

```
    public class Address
    {
        public int Id { get; set; }
        public string Fullname { get; set; }
        public string Title { get; set; }
        public string Body { get; set; }

        public User User { get; set; } // navigation property
        public int UserId { get; set; } // int=> null, 1, 2, 3, 4
    }

    public class Product
    {
        public int Id { get; set; }

        [MaxLength(100)]
        [Required]
        public string Name { get; set; }

        public decimal Price { get; set; }

        public int CategoryId { get; set; }

    }
    public class Category
    {
        public int Id { get; set; }
        public string Name { get; set; }
    }

    class Program
    {
        static void Main(string[] args)
        {
            // InsertUsers();
            // InsertAddresses();

            using (var db = new ShopContext())
            {
                var user = db.Users.FirstOrDefault(i => i.Username == "cinarturan");

                if (user != null)
                {
                    user.Addresses = new List<Address>();

                    user.Addresses.AddRange(
                        new List<Address>(){
                            new Address(){Fullname="Çınar Turan",Title="İş adresi
1",Body="Kocaeli"},
                            new Address(){Fullname="Çınar Turan",Title="İş adresi 2
",Body="Kocaeli"},
                            new Address(){Fullname="Çınar Turan",Title="İş adresi
3",Body="Kocaeli"}
                        }
                    );

                    db.SaveChanges();
                }
            }
        }

        static void InsertUsers()
        {
            var users = new List<User>(){
```

```
                new User(){Username="sadikturan",Email="info@sadikturan.com"},
                new User(){Username="cinarturan",Email="info@cinarturan.com"},
                new User(){Username="yigitbilgi",Email="info@yigitbilgi.com"},
                new User(){Username="adabilgi",Email="info@adabilgi.com"},
            };

            using (var db = new ShopContext())
            {
                db.Users.AddRange(users);
                db.SaveChanges();
            }
        }
        static void InsertAddresses()
        {
            var addresses = new List<Address>(){
                new Address(){Fullname="Sadık Turan",Title="Ev adresi",Body="Kocaeli",UserId=1},
                new Address(){Fullname="Sadık Turan",Title="İş adresi",Body="Kocaeli",UserId=1},
                new Address(){Fullname="Yiğit Bilgi",Title="Ev adresi",Body="Kocaeli",UserId=3},
                new Address(){Fullname="Yiğit Bilgi",Title="İş adresi",Body="Kocaeli",UserId=3},
                new Address(){Fullname="Çınar Turan",Title="İş adresi",Body="Kocaeli",UserId=2},
                new Address(){Fullname="Ada Bilgi",Title="İş adresi",Body="Kocaeli",UserId=4}

            };

            using (var db = new ShopContext())
            {
                db.Addresses.AddRange(addresses);
                db.SaveChanges();
            }
        }


    }
}
```

One To One İlişkisi Kurma

Veritabanı tabloları arasında Bire-Bir ilişkisi kurarken aşağıdaki yapılar kullanılabilir.

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Logging;

namespace ConsoleApp
{
    public class ShopContext : DbContext
    {
        public DbSet<Product> Products { get; set; }
        public DbSet<Category> Categories { get; set; }
        public DbSet<User> Users { get; set; }
        public DbSet<Customer> Customers { get; set; }
        public DbSet<Address> Addresses { get; set; }
        public static readonly ILoggerFactory MyLoggerFactory = LoggerFactory.Create(builder =>
{ builder.AddConsole(); });
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder
                .UseLoggerFactory(MyLoggerFactory)

.UseMySql(@"server=localhost;port=3306;database=ShopDb;user=root;password=mysql1234;");
        }
```

```
}

// One to Many
// One to One
// Many to Many

public class User
{
    public int Id { get; set; }
    public string Username { get; set; }
    public string Email { get; set; }

    public Customer Customer { get; set; }
    public List<Address> Addresses { get; set; } // navigation property
}

public class Customer
{
    public int Id { get; set; }
    public string IdentityNumber { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public User User { get; set; }
    public int UserId { get; set; }
}

public class Supplier
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string TaxNumber { get; set; }
}

public class Address
{
    public int Id { get; set; }
    public string Fullname { get; set; }
    public string Title { get; set; }
    public string Body { get; set; }

    public User User { get; set; } // navigation property
    public int UserId { get; set; } // int=> null, 1, 2, 3, 4
}
public class Product
{
    public int Id { get; set; }

    [MaxLength(100)]
    [Required]
    public string Name { get; set; }

    public decimal Price { get; set; }

    public int CategoryId { get; set; }

}
public class Category
{
    public int Id { get; set; }
    public string Name { get; set; }
}

class Program
{
    static void Main(string[] args)
```

```
{
    using (var db = new ShopContext())
    {
        // var customer = new Customer(){
        //          IdentityNumber="12313132",
        //          FirstName="Sadık",
        //          LastName="Turan",
        //          User = db.Users.FirstOrDefault(i=>i.Id==3)
        // };

        // db.Customers.Add(customer);
        // db.SaveChanges();


        var user = new User()
        {
            Username = "deneme",
            Email = "deneme@gmail.com",
            Customer = new Customer()
            {
                FirstName = "Deneme",
                LastName = "Deneme",
                IdentityNumber = "13213132"
            }
        };

        db.Users.Add(user);
        db.SaveChanges();
    }
}

static void InsertUsers()
{
    var users = new List<User>(){
        new User(){Username="sadikturan",Email="info@sadikturan.com"},
        new User(){Username="cinarturan",Email="info@cinarturan.com"},
        new User(){Username="yigitbilgi",Email="info@yigitbilgi.com"},
        new User(){Username="adabilgi",Email="info@adabilgi.com"},
    };

    using (var db = new ShopContext())
    {
        db.Users.AddRange(users);
        db.SaveChanges();
    }
}
static void InsertAddresses()
{
    var addresses = new List<Address>(){
        new Address(){Fullname="Sadık Turan",Title="Ev adresi",Body="Kocaeli",UserId=1},
        new Address(){Fullname="Sadık Turan",Title="İş adresi",Body="Kocaeli",UserId=1},
        new Address(){Fullname="Yiğit Bilgi",Title="Ev adresi",Body="Kocaeli",UserId=3},
        new Address(){Fullname="Yiğit Bilgi",Title="İş adresi",Body="Kocaeli",UserId=3},
        new Address(){Fullname="Çınar Turan",Title="İş adresi",Body="Kocaeli",UserId=2},
        new Address(){Fullname="Ada Bilgi",Title="İş adresi",Body="Kocaeli",UserId=4}

    };

    using (var db = new ShopContext())
    {
        db.Addresses.AddRange(addresses);
        db.SaveChanges();
    }
}
```

```
    }
}
```

Many To Many İlişkisi Kurma

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Logging;

namespace ConsoleApp
{
    public class ShopContext : DbContext
    {
        public DbSet<Product> Products { get; set; }
        public DbSet<Category> Categories { get; set; }
        public DbSet<User> Users { get; set; }
        public DbSet<Customer> Customers { get; set; }
        public DbSet<Address> Addresses { get; set; }
        public static readonly ILoggerFactory MyLoggerFactory = LoggerFactory.Create(builder =>
{ builder.AddConsole(); });
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder
                .UseLoggerFactory(MyLoggerFactory)

.UseMySql(@"server=localhost;port=3306;database=ShopDb;user=root;password=mysql1234;");
        }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<ProductCategory>()
                        .HasKey(t => new { t.ProductId, t.CategoryId });

            modelBuilder.Entity<ProductCategory>()
                        .HasOne(pc => pc.Product)
                        .WithMany(p => p.ProductCategories)
                        .HasForeignKey(pc => pc.ProductId);

            modelBuilder.Entity<ProductCategory>()
                        .HasOne(pc => pc.Category)
                        .WithMany(c => c.ProductCategories)
                        .HasForeignKey(pc => pc.CategoryId);


        }
    }

    // One to Many
    // One to One
    // Many to Many

    // convention
    // data annotations
    // fluent api

    public class User
    {
        public int Id { get; set; }
        public string Username { get; set; }
        public string Email { get; set; }
```

```
        public Customer Customer { get; set; }
        public List<Address> Addresses { get; set; } // navigation property
    }
    public class Customer
    {
        public int Id { get; set; }
        public string IdentityNumber { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public User User { get; set; }
        public int UserId { get; set; }
    }
    public class Supplier
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string TaxNumber { get; set; }
    }
    public class Address
    {
        public int Id { get; set; }
        public string Fullname { get; set; }
        public string Title { get; set; }
        public string Body { get; set; }

        public User User { get; set; } // navigation property
        public int UserId { get; set; } // int=> null, 1, 2, 3, 4
    }
    public class Product
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public decimal Price { get; set; }
        public List<ProductCategory> ProductCategories { get; set; }

    }
    public class Category
    {
        public int Id { get; set; }

        public string Name { get; set; }

        public List<ProductCategory> ProductCategories { get; set; }
    }

    public class ProductCategory
    {
        public int ProductId { get; set; }
        public Product Product { get; set; }

        public int CategoryId { get; set; }
        public Category Category { get; set; }
    }


    class Program
    {
        static void Main(string[] args)
        {
            using (var db = new ShopContext())
            {

                var products = new List<Product>()
                {
```

```
                new Product(){Name="Samsung S5",Price=2000},
                new Product(){Name="Samsung S6",Price=3000},
                new Product(){Name="Samsung S7",Price=4000},
                new Product(){Name="Samsung S8",Price=5000}
            };


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


            int[] ids = new int[2] { 1, 2 };

            var p = db.Products.Find(1);

            p.ProductCategories = ids.Select(cid => new ProductCategory()
            {
                CategoryId = cid,
                ProductId = p.Id
            }).ToList();


            db.SaveChanges();

        }
    }

    static void InsertUsers()
    {
        var users = new List<User>(){
            new User(){Username="sadikturan",Email="info@sadikturan.com"},
            new User(){Username="cinarturan",Email="info@cinarturan.com"},
            new User(){Username="yigitbilgi",Email="info@yigitbilgi.com"},
            new User(){Username="adabilgi",Email="info@adabilgi.com"},
        };

        using (var db = new ShopContext())
        {
            db.Users.AddRange(users);
            db.SaveChanges();
        }
    }
    static void InsertAddresses()
    {
        var addresses = new List<Address>(){
            new Address(){Fullname="Sadık Turan",Title="Ev adresi",Body="Kocaeli",UserId=1},
            new Address(){Fullname="Sadık Turan",Title="İş adresi",Body="Kocaeli",UserId=1},
            new Address(){Fullname="Yiğit Bilgi",Title="Ev adresi",Body="Kocaeli",UserId=3},
            new Address(){Fullname="Yiğit Bilgi",Title="İş adresi",Body="Kocaeli",UserId=3},
            new Address(){Fullname="Çınar Turan",Title="İş adresi",Body="Kocaeli",UserId=2},
            new Address(){Fullname="Ada Bilgi",Title="İş adresi",Body="Kocaeli",UserId=4}

        };

        using (var db = new ShopContext())
        {
            db.Addresses.AddRange(addresses);
            db.SaveChanges();
        }
    }
```

```
    }
}
```

Ek Kaynak: https://docs.microsoft.com/en-us/ef/core/saving/related-data

Çoklu Fotograf Ekleme
Ek Kaynak: https://www.youtube.com/watch?v=lh-oOqukZus

MİGRATİON LİNKLERİ

webUI> dotnet ef migrations add AddingIdentity --context ApplicationContext
webUI> dotnet ef database update --context ApplicationContext
data> dotnet ef migrations add InitialCreate --startup-project ../webUI --context ShopContext
data> dotnet ef database update --startup-project ../webUI --context ShopContext

Var olan tabloya sütün eklerken çalışan migration
dotnet ef migrations add CreateAddressIdColumnInSuppliersTable --startup-project ../webUI --context ShopContext
Bu migration Suppliers tablosu içine yazılan adresle ilgili yabancı anahtarları aşağıdaki şekilde kullanıldı.

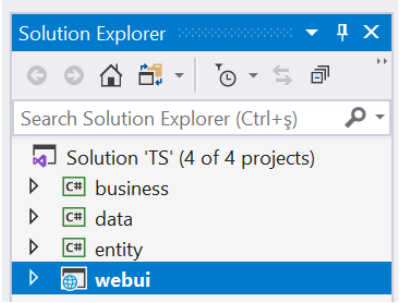
```
public int AddressId { get; set; } //Bu iki satır beraber olmak zorunda
public Address Adresses { get; set; } //Bu iki satır beraber olmak zorunda
```

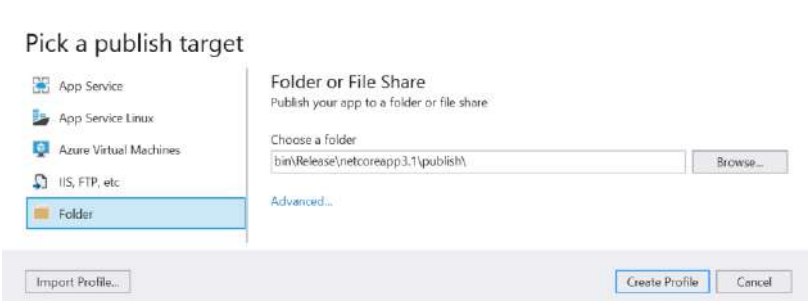
Fakat bu kodlar hata verdi.
Yeni bir Tablo Eklemek için Migration Adı: AddAdresTable
Sütun Eklemek için Migration Adı: CreateAdresColumnInMusteriTable
Sutun adını değiştirmek için Migration Adı: RenameMusteriSoyadToMusteriUnvanInMusterisTable Yalnız böyle bir uygulamada Mevcut sütunu kaldırıp Yeni bir sütün ekler. Bu durumda veriler kaybolur. Bununda yöntemi var.

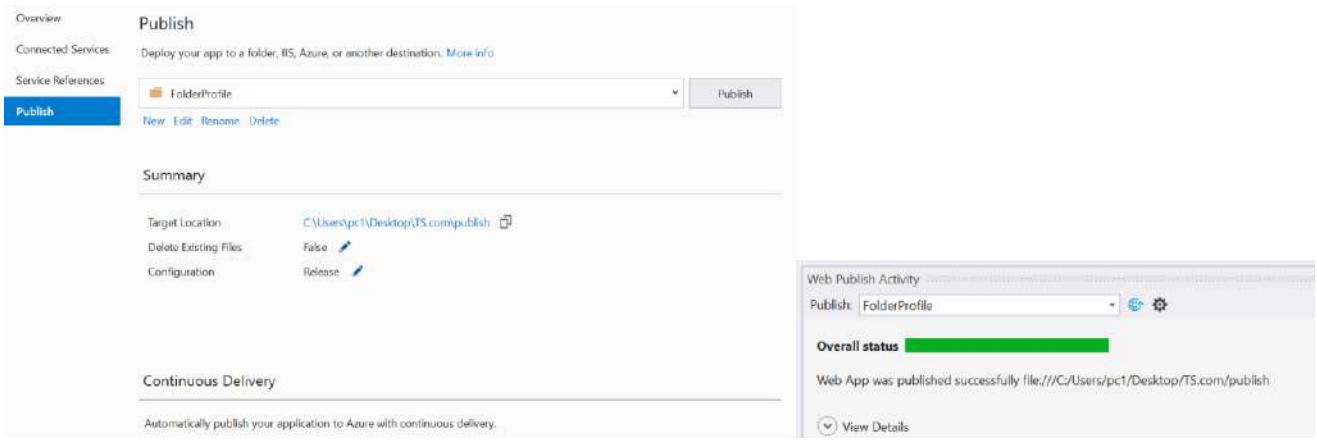
Bu komutun ne iş yaptığını bir öğren
Add-Migration InitialCreate –IgnoreChanges

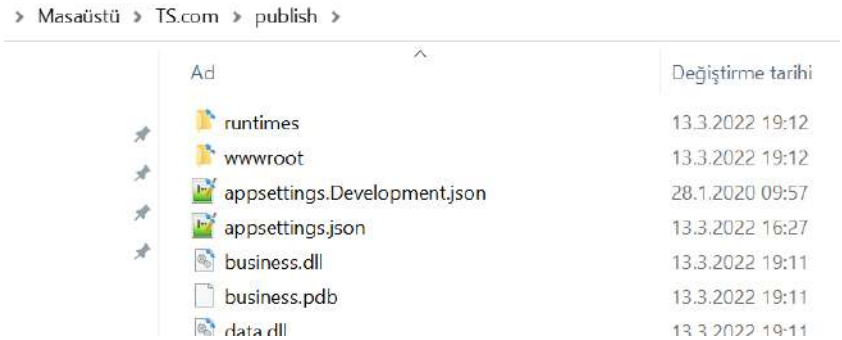
PUBLISH-SİTEYİ YAYINA ALINIRKEN DİKKAT EDİLECEK HUSUSLAR (Publish)

Publish İşleminin Visual Studio İçinden Yapılması
Projemizin üzerine Explorerdan webui projesi üzerine sağ tuşa tıklayıp publish seçelim. Açılan ekrandan Folder seçip kaydeceğimiz yeri belirleyelim. Publish dedikten sonra dosyalarımızın ilgili klasör içinde oluştuğunu görürüz.

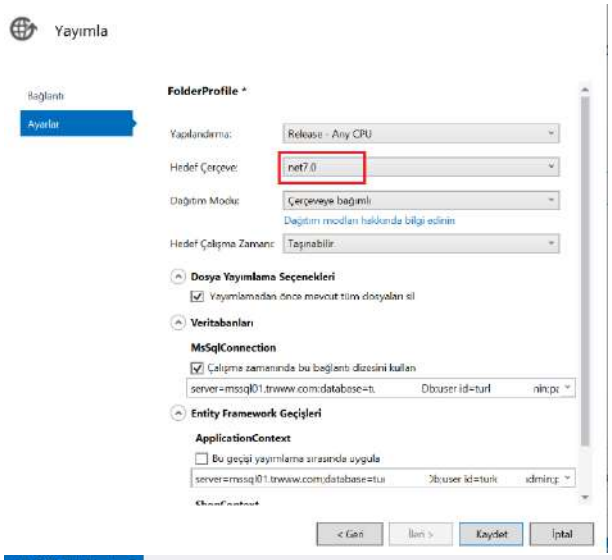


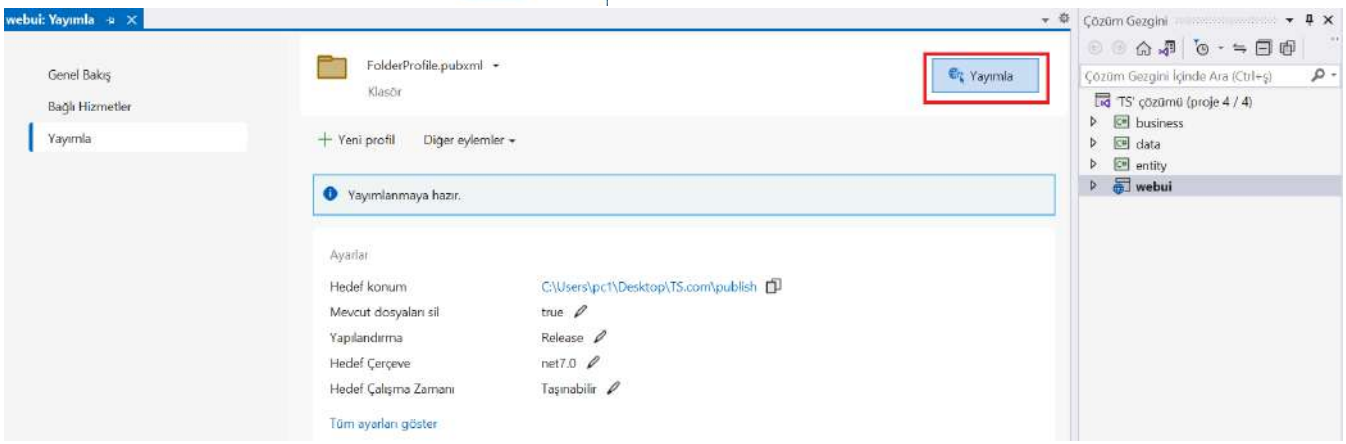






Net7 ye geçtikten sonra ayarlar şöyleydi.





1. Projeyi BUILD Et. yada ÇALIŞTIR. fotograflar gözükmemeli. Çünkü yerelde çalışılıyor.

2. APPSETTINGS.json içindeki veritabanı linkini yerelden servera al.
3. ACCOUNTCONTROLLER içindeki `Register` metodunu düzenle. 140 satırlarda yeni üyelerin kaydolurken tıkladıklarında linkin adresi var. Oranın değişmesi gerekir.. Ayrıca 260 ıncı satırlarda ResetPassword linki de değişmelidir.
4. PROGRAM.CS içindeki şu satır SERVERDA ÇALIŞIRKEN AÇIK olsa iyi olur. Migrationları her iki serverdada güncelleyerek gidilmeli. CreateHostBuilder(args).Build().Run(); //Bu kodda host u ayağa kaldırırken önce derleyecek (build). sonrada çalıştıracak (run)
5. LAUNGESETTINGS.CS içindeki şu satırı değiştir. `"ASPNETCORE_ENVIRONMENT": "Production"`
6. Veritabanındaki yeni işlemler için ayrı MİGRATİON oluştur.
7. Yüklerken WWWROOT klasörü listeden ÇIKARILSIN. Bu yüklenmesin. Upload edilmesin. Ayrıca Node_Modules klasörü de yüklenmesin (Node.js de güncelleme yapıldı ise Node Modules de yüklenmeli.).
8. SHOPCONTEXT.CS içinde en sonda bulunan 3 satır kapatılmalı. Seed dosyaları. Bu satırlar Veritabanı ilk defa kurulurken açılmalı. Aynısı STARTUP.CS içinde en sonra Identiler için Seed vardır. Orasıda kapalı olsun. Veritabanı ilk defa kurulurken açılmalı.
9. DENEME ÇALIŞTIRMASI: Veritabanı Adresi Servera ayarlı iken yerelde projeyi çalıştıralım. Resimler gelmez ama Veritabanı bilgileri serverdan gelir. Bu şekilde ise serverdada düzgün çalışacak demektir. Adress satırı yine Localhost görünür. LaungeSettings.json dosyasındaki port bilgiler serverda çalıştırılmaz.
10. SUNUCUNUN Asp desteği yerelde çalışan ile aynı olmalı. Bunu Pesk den kontrol et. Orada “barındırma ayarları” kısmından (kullanıcı girişi yap. Siteyi seç. Sol altta pleske bağlan.
11. PUBLISH çalıştırılırken dosyalardan ÜZERİNE YAZILSIN seçilsin. (Explorerdan webui projesi üzerine sağ tuşa tıklayıp publish seçelim)(Publish klasörü içindeki en son tarihli dosyaları servera atabilirsin)
12. Catch Temizlenip yenilensin.. Tarayıcı Kapatılsın..
13. Aynı Migration ı hem serverdaki VT yi hemde Yereldeki VT yi güncellemek için kullanabilirsin. Bunun için sadece AppSettings içindeki linkleri değiştirmek yeterlidir.
14. Eğer daha önceden kullanılmış bir Migrationı tekrar oluşturmak için hata veriyorsa o migrationı silebiliriz. Yalnız böyle bir durumda bir sonraki migrationlarda tekrar o tabloları oluşturmayı deneyebilir. Bence Update edilecek migrationları belirterek yapmak gerekir. Bu konuya bir bak.
15. 10)Ftp programı ile publish dosyalarını Servera atarken oradaki httpdocs klasörü altına atalım. Orada aspnet_client dosyası silinemiyor..

Çalıştırmadan önce ön belleği temizle. Bunu her zaman kullanma. Hata verirse kullan. Tarayıcı komple bir kapat olmazsa. Çünkü tüm kütüphaneler yeniden yükleniyor.

Ts> dotnet nuget locals all –clear

YAYINDAN ÖNCE ŞU DOSYALARIN İÇERİSİNİ AÇIP BİR İNCELE. Önemli değişikler içinde vardır.

Program.cs
Startup.cs
Appsettings.json (Veritabanı linki serverdaki kullanıcı bilgileri olmalı)
Data/ShopContext.cs (içerisinde sondaki 3 satırı kapat)

19.06.2023 tarihinde Güvenlik Sertifikası ile ilgili sorun esnasında aldığm notlar

Çalışan DOTNET AYARLAMA KOMUTLARI-ZOR BULDUM SAKLA
Komut ekranını aç (cmd ifasini kullan)
dotnet --info (çalışıyor ama ne işe yarar bilmiyorum)
dotnet --list-sdks (sdk ları listeler yani .Net versiyonları)
dotnet --version (Dikkat data ve webui altında ayrı ayrı dotnet (=.Net) versiyonu ayarlamak gerekiyor.)
setx DOTNET_SDK_VERSION 7.0.304 (Bu komut işe yaradımı emin değilim. Data ve diğer projeler için çalışmış olabilir. Ama webui projesinin sdk sını ancak Global.json dosyasının içinde sdk yı değiştirerek yapabildim.)

****
*SERTİFİKA SORUNUN ŞU LİNKTEKİ GİBİ YAPARAK ÇÖZDÜM. (henüz çözülmedi)
https://learn.microsoft.com/tr-tr/troubleshoot/developer/visualstudio/installation/warnings-untrusted-certificate

Burada son kısımda İÇE AKTARIM TAM ANLATAMAMIŞ. GPT ŞU Şekilde bilgi verdi ve öyle yapıldı.

Dışarı aktarılan sertifikayı belirtilen konuma aktarmanız gerekiyor: "Certificates - Current User\Trusted Root Certification Authorities\Certificates". Bu konum, Windows işletim sisteminde yer alan sertifika depolarından biridir.

Aşağıda, bu konumu nasıl bulabileceğinize dair adımları bulabilirsiniz:

Başlat menüsünü açın ve "certmgr.msc" yazarak "Sertifika Yöneticisi"ni arayın ve açın.
Sol taraftaki ağaç yapısında "Trusted Root Certification Authorities" klasörünü genişletin.
"Certificates" klasörüne sağ tıklayın ve "All Tasks" (Tüm Görevler) üzerine gelin.
Açılan alt menüden "Import..." (İçe Aktar...) seçeneğini seçin.
Dışarı aktardığınız sertifika dosyasını bulun ve "Aç" düğmesine tıklayın.
Sertifika içe aktarıldığında, "Certificate Import Wizard" (Sertifika İçe Aktarma Sihirbazı) adımlarını takip ederek sertifikayı yükleyin.
Bu adımları takip ettiğinizde, dışarı aktarılan sertifikanızı belirtilen konuma başarıyla yükleyebilirsiniz. Unutmayın, işletim sistemi veya kullanıcı izinleri nedeniyle bu konuma erişimde sorunlar yaşayabilirsiniz. Gerekirse yönetici yetkileriyle çalıştığınızdan emin olun.

********** ek bazı denediğim çalışmalar. işe yarar mı bak..

*cmd/**Services.msc** Sql hizmetlerini durdurmak için

* (SQL Server Configuration Manager) Komut Satır>**MMC** yazıp veritabanın Sertifikası ile ilgili ayarları yaptım.

* (SQL Server Management Studio) Veritabanlarını kapatmak için
SQL Server Management Studio (SSMS) gibi bir araç kullanarak SQL Server'e bağlanın.

Sol tarafta "Object Explorer" bölümünde, SQL Server'ın altında "Databases" düğmesini bulun ve tıklayın. Böylece mevcut veritabanlarınızın listesini görebilirsiniz.

*Önemli bir makale:
https://learn.microsoft.com/en-us/troubleshoot/sql/database-engine/connect/error-message-when-you-connect

Dotnet –version ile Global.json içerikleri aynı olmalıdır. Migration öncesi bu kontrolü yap.

Dotnet versiyonları görmek için her bir proje (business, data, webui vs) üzerine sağ tuşa tıkla. Proje dosyasını göster de. İçinde NuGet paketlerinin versiyonları gözükecektir.

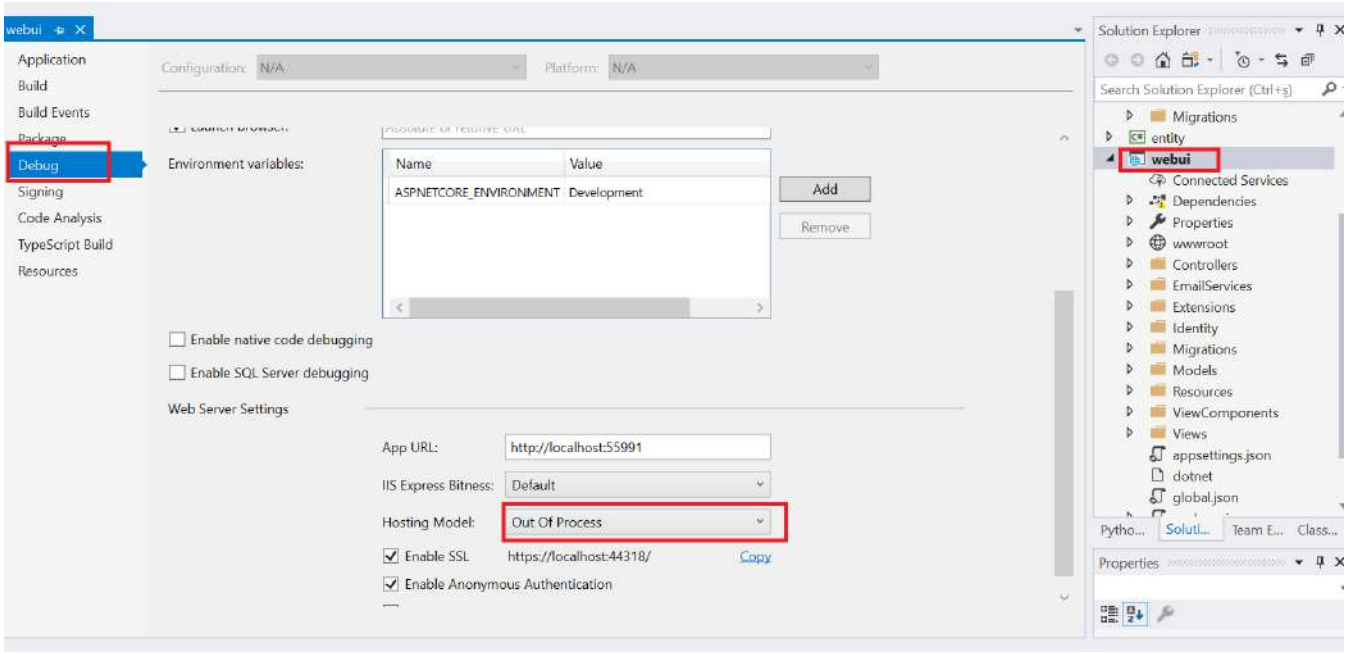
```
{
  "sdk": {
    "version": "6.0.201",
    "rollForward": "latestFeature"
  }
```

}

```
C:\Users\pc1\Desktop\TS.com\TS\webui>dotnet --version
6.0.201
```

"HTTP Error 500.0 - ANCM In-Process Handler Load Failure" Hatasını Giderme

WebUI projesi üzerine sağ tuşa tıklayıp Properties>Debug ve "Out Of Process" seçilerek giderilebilir.



Temel Ayar Dosyalarını Anlatan Doküman

https://www.c-sharpcorner.com/article/project-structure-in-asp-net-core-3-1-web-application/

.csproj File
launchSettings.json
Program.cs
Startup.cs
wwwroot Folder

InProcess and another is OutOfProcess hosting model arasındaki farkı anlatıyor.

https://www.c-sharpcorner.com/article/inprocess-hosting-model-in-asp-net-core/

Örnek Kodlar. Bir sitenin baştan sona kodları vardır.

https://github.com/elanderson/ASP.NET-Core-Basics/tree/master/ASP.NET%20Core%20Basics/src/Contacts/Controllers

Redirect Örnekleri

https://www.infoworld.com/article/3570787/how-to-redirect-a-request-in-aspnet-core-mvc.html

```
Redirect("/Author/Index");

RedirectPermanent("/Author/Index");

RedirectPermanentPreserveMethod("/Author/Index");

RedirectPreserveMethod("/Author/Index");
```

```
public RedirectResult Index()
{
    return new RedirectResult(url: "/Author/Index", permanent: true,
                              preserveMethod: true);
}


public RedirectResult Index()
{
    return Redirect("https://google.com");
}

public RedirectToActionResult Index()
{
    return RedirectToAction(actionName: "Index", controllerName: "Author");
}

public RedirectToActionResult Index()
{
    return RedirectToAction(actionName: "Privacy");
}

public RedirectToRouteResult Index()
{
    return RedirectToRoute("author");
}


var routeValue = new RouteValueDictionary
 (new { action = "View", controller = "Author" });
return RedirectToRoute(routeValue);


public IActionResult RedirectToAuthorPage()
{
    return RedirectToPage("Author");
}
```

Enum Kullanımı

```
public enum SehirPlakalari
{
    Ankara = 6, Kocaeli = 41, İzmir = 35, Adana = 1, Hatay = 31
}

string[] SehirPlakalariDizisi = Enum.Getnames(typeof(SehirPlakalari));

//Bir başka kullanımı
enum Gun { Pazartesi, Salı, Carsamba, Persembe, Cuma, Cumartesi, Pazar };
Gun secilenGun = Gun.Carsamba;
if (secilenGun == Gun.Cumartesi || secilenGun == Gun.Pazar)

//Bir başka kullanımı
enum Gun { Pazartesi = 1, Salı = 2, Carsamba = 3, Persembe = 4, Cuma = 5, Cumartesi = 6, Pazar =
7 };
int secilenGun = (int)Gun.Carsamba;
 if (secilenGun == (int) Gun.Cumartesi || secilenGun == (int) Gun.Pazar)

//Bir başka kullanımı
enum Sonuç : byte { Kaldi, Gecti };
```

- Enum içerisinde değer vermezsek, değerler 0'dan başlar ve birer birer artar.
- Enum'ların varsayılan değer "int"dir.

- Enumları; byte,sbyte, short,ushort, int, uint,long, ulong türlerin oluşturabiliriz.

```
-------
enum Renkler { Kirmizi, Yesil, Mavi };
enum Departmanlar { Yazilim, Bilgi_Işlem, Muhasebe };
enum Gunler { Pazartesi, Sali, Carsamba, Persembe, Cuma, Cumartesi, Pazar };

--------------------
enum Renkler { Kirmizi = "Kırmızı", Yesil = "Yeşil", Mavi }; // hatalı
enum Gunler { Pazartesi = 1, Sali = 2, Carsamba = 3, Persembe = 4, Cuma = 5, Cumartesi = 6,
Pazar = 7 }; // doğru

Renkler Renk = Renkler.Kirmizi;
Departmanlar Departman = Departmanlar.Muhasebe;
Gunler Gun = Gunler.Cuma;

MessageBox.Show("Renk:" + Renk.ToString() + " || Departman: " + Departman.ToString() + " || Gün:
" + Gun.ToString());

------------------
enum GecmeDurumu { Basarisiz = 0, Gecer = 45, Orta = 60, Iyi = 70, Pekiyi = 80 }

int ogrenci_not = 70;
GecmeDurumu gd = (GecmeDurumu)ogrenci_not;
MessageBox.Show(gd.ToString());
-----------
enum Gunler { Pazartesi, Sali, Carsamba, Persembe, Cuma, Cumartesi, Pazar };

string[] a = Gunler.GetNames(typeof(Gunler));
MessageBox.Show(a[0].ToString());
MessageBox.Show(Gunler.GetNames(typeof(Gunler))[3]);
```

Enum ve Dizi Kullanımı

```
@foreach (var item in Enum.GetValues(typeof(EnumBusinessType)))

   Enum.GetValues(typeof(SomeEnum))
    .Cast<SomeEnum>()
    .Select(v => v.ToString())
    .ToList();

foreach(var item in Enum.GetValues(typeof(EnumBusinessType)))
{
       var Name = item.ToString();
       var Value = Convert.ToInt32(item);
}


public class EnumModel
{
    public int Value { get; set; }
    public string Name { get; set; }
}

public enum MyEnum
{
    Name1 = 1,
    Name2 = 2,
    Name3 = 3
}

public class Test
{
    List<EnumModel> enums = ((MyEnum[])Enum.GetValues(typeof(MyEnum))).Select(c => new
EnumModel() { Value = (int)c, Name = c.ToString() }).ToList();
```

```
    // A list of Names only, does away with the need of EnumModel
    List<string> MyNames = ((MyEnum[])Enum.GetValues(typeof(MyEnum))).Select(c =>
c.ToString()).ToList();

    // A list of Values only, does away with the need of EnumModel
    List<int> myValues = ((MyEnum[])Enum.GetValues(typeof(MyEnum))).Select(c =>
(int)c).ToList();

    // A dictionnary of <string,int>
    Dictionary<string, int> myDic = ((MyEnum[])Enum.GetValues(typeof(MyEnum))).ToDictionary(k =>
k.ToString(), v => (int)v);
}
```

```
   enum Colors
    {
        Red = 1,
        Green = 2,
        Blue = 3
    };
Console.WriteLine(Enum.GetName( typeof(Colors), Colors.Green ) );
Console.WriteLine(Enum.GetName( typeof(Colors), 3 ) );
```

Çıktısı: Green, Blue

Entity Framework Dersleri –(LingQ komutları)

https://www.youtube.com/watch?v=vkwzhPTI_9A&list=PLKnjBHu2xXNNiAXIaoH9rau4IQ95b6RYu&index=15

```
dataGridView1.DataSource = db.TBLOGRENCI.Where(x => x.AD == TxtAd.Text | x.SOYAD == TxtSoyad.Text).ToList();
```

Bu sorgu öğrenciler tablosunda Ad yada Soyad sütununa bakacak ve bunların Texbox daki ifadelerle eşleşenleri liste olarak karşımıza getirecek.

```
string aranan = TxtAd.Text;
var degerler = from item in db.TBLOGRENCI
               where item.AD.Contains(aranan)
               select item;
dataGridView1.DataSource = degerler.ToList();
```

Bu sorgu öğrenciler tablosundan Ad sütünu aranan ifadeyi içereni getirir. Sonradan bulduğu değerleri de listeye çeviriyor. Bunlara LinQ sorgusu deniyor.

```
//Asc - Ascending
List<TBLOGRENCI> liste1 = db.TBLOGRENCI.OrderBy(p => p.AD).ToList();
dataGridView1.DataSource = liste1;
```

```
//Desc - Descending
List<TBLOGRENCI> liste2 = db.TBLOGRENCI.OrderByDescending(p => p.AD).ToList();
dataGridView1.DataSource = liste2;
```

Bu sorguda öğrenci tablosundaki elemanları liste1 içine atacak. Fakat elemanları getirirken OrderBy ile sıralayacak. Sıralamayı ise p parametre anlamında Ad parametresine göre yapacak. En sonda ToList() ifadesiyle sorguyu listeye dönüştürerek çalıştırmış olacak. Z den A ya doğru sıralar iken ise OrderByDescending kullanılır.

```
List<TBLOGRENCI> liste3 = db.TBLOGRENCI.OrderBy(p => p.AD).Take(3).ToList();
dataGridView1.DataSource = liste3;
```

Bu sorgu A-Z ye AD göre sıralayıp ardından ilk 3 tane kaydı listeye çevirip getirir.

```
List<TBLOGRENCI> liste4 = db.TBLOGRENCI.Where(p => p.ID == 5).ToList();
dataGridView1.DataSource = liste4;
```

Bu sorgu ID si 5 olan kişiyi getirir.

```
List<TBLOGRENCI> liste5 = db.TBLOGRENCI.Where(p => p.AD.StartsWith("a")).ToList();
dataGridView1.DataSource = liste5;

List<TBLOGRENCI> liste6 = db.TBLOGRENCI.Where(p => p.AD.EndsWith("a")).ToList();
dataGridView1.DataSource = liste6;
```

Bu sorgu AD alanı a harfi ile başlayanları getirir. Son harfi a ile bitenler ise EndsWith ile yapılır.

```
bool deger = db.TBLKULUPLER.Any();
MessageBox.Show(deger.ToString(), "Bilgi", MessageBoxButtons.OK, MessageBoxIcon.Information);
```

Buradaki Any() ifadesi değer var mı anlamındadır. Tabloda Herhangi bir değer varsa True döndürecektir.

```
int toplam = db.TBLOGRENCI.Count();
MessageBox.Show(toplam.ToString(), "Toplam Öğrenci Sayısı", MessageBoxButtons.OK, Me
```

Count() fonksiyonu ile toplam öğrenci sayısını getirmiş oluyor.

```
var toplam = db.TBLNOTLAR.Sum(p => p.SINAV1);
MessageBox.Show("Toplam Sınav1 Puanı: " + toplam.ToString());
```

Notlar tablosundaki Sınav1 sütunundaki notların toplamını getirir.

```
var ortalama = db.TBLNOTLAR.Average(p => p.SINAV1);
MessageBox.Show("1.Sınavın Ortalaması: " + ortalama.ToString());
```

Sınav1 sütünundaki notların ortalamasını getirir.

```
var enyuksek = db.TBLNOTLAR.Max(p => p.SINAV1);
MessageBox.Show("1.Sınavın En Yüksek Notu: " + enyuksek);
```

En yüksek notu getirir. Min() fonksiyonu da en küçük değeri getirir.

```
var query = from item in db.TBLNOTLAR
            select new
            {
                item.NOTID,
                item.OGR,
                item.DERS,
                item.SINAV1,
                item.SINAV2,
                item.SINAV3,
                item.ORTALAMA,
                item.DURUM
            };
dataGridView1.DataSource = query.ToList();
```

Bu sorguda Notlar tablosundan okuduğu her bir elemana item diyor. Yani foreach döngüsü gibi çalışıyor aslında. Bulduğu sürece aynı işlemi yapacaktır. Sonra her bir Item alta alanlarından hangilerini görüntüleyeceğini select ifadesiyle seçiyor. Burada tablonun sütunları Item'ın alt alanları olur. O zaman Item tablodaki her bir satırı ifade

eder. Sonradan bir alt satırda bunları listeye çevirerek görüntülüyor. Aslında sorgu o ana kadar çalışmıyor. ToList() ifadesiyle sorgu çalışmış oluyor. Burasıda önemli.

Aşağıdaki sorgu iki tabloyu JOIN işlemiyle nasıl birleştirileceğini göstermektedir.

```
var sorgu = from d1 in db.TBLNOTLAR
            join d2 in db.TBLOGRENCI
            on d1.OGR equals d2.ID
            select new
            {
                ÖĞRENCİ=d2.AD,
                SINAV1=d1.SINAV1,
                sınav2=d1.SINAV2
            };
dataGridView1.DataSource = sorgu.ToList();
```

Bu sorguda Notlar tablosu ile Öğrenciler tablosu ÖğrenciId (ÖĞR ifadesi ile gösterilmiş) ve ID sütunu üzerinden birleştiriliyor. Ortaya çıkan sütunlardan Öğrenci tablosundan AD alanını, Notlar tablosundan Sınav1 ve Sınav2 sütunlarını getiriyor. Select in içindeki sol taraftaki ifadeler sütun başlıkları oluyor. Yada değişkenler diyebiliriz.

Aşağıdaki sorgu ise 3 tane tabloyu birleştirmeye bir örnektir.

```
var sorgu = from d1 in db.TBLNOTLAR
            join d2 in db.TBLOGRENCI
            on d1.OGR equals d2.ID
            join d3 in db.TBLDERSLER
            on d1.DERS equals d3.DERSID

            select new
            {
                ÖĞRENCİ = d2.AD + " " + d2.SOYAD,
                DERS=d3.DERSAD,
                //SOYAD=d2.SOYAD,
                SINAV1 = d1.SINAV1,
                SINAV2 = d1.SINAV2,
                SINAV3 = d1.SINAV3,
                ORTALAMA = d1.ORTALAMA

            };
dataGridView1.DataSource = sorgu.ToList();
```

Where (hangisini?) Komutunun kullanımı

```
var degerler = db.TBLNOTLAR.Where(x => x.SINAV1 < 50);
dataGridView1.DataSource = degerler.ToList();
```

Notlar tablosunda Sınav1 sütunu 50 den küçük olan değerleri getirecektir.

```
var degerler = db.TBLOGRENCI.Where(x => x.AD == "ali");
dataGridView1.DataSource = degerler.ToList();
```

Öğrenciler tablosunda Ad ı Ali olanları getirir.

```
var degerler = db.TBLOGRENCI.Where(x => x.AD == textBox1.Text || x.SOYAD == textBox1.Text);
dataGridView1.DataSource = degerler.ToList();
```

Adı yada Soyadı textboxlarda yazan değerlere eşit olanları getirir.

SELECT getirme komutunun kullanımı

```
var degerler = db.TBLOGRENCI.Select(x => x);
dataGridView1.DataSource = degerler.ToList();
```

Böyle bir sorgu öğrenci tablosundaki tüm bilgileri getirir.

```
var degerler = db.TBLOGRENCI.Select(x => x.AD);
dataGridView1.DataSource = degerler.ToList();
```

Böyle bir sorgu Ad ları getirmez. Adların uzunluğunu (length) ni getirir. Her ismin kaç harf olduğunu getirir. Peki biz sadece soyadları getirmek istiyoruz. Bunu nasıl yapacağız.

```
var degerler = db.TBLOGRENCI.Select(x => new {soyadı=x.SOYAD });
dataGridView1.DataSource = degerler.ToList();
```

Sadece soyadları getirmek için Anonyms Type denilen bir değişken oluşturarak yapabiliriz. Yani bu şekilde soyad sütunundan okunan her bir ifade Soyadı isminde bir değişkenin içine atılacak ve onun içinde dışarı taşınacak. Buda tabloda sütün adı olur yada değişken olarak kullanılabilir.

soyadı
YILDIZ
ÖZTÜRK
YILDIRIM
KAYALI
SARI

```
var degerler = db.TBLOGRENCI.Select(x =>
new
{
    Ad = x.AD.ToUpper(),
    Soyadı = x.SOYAD.ToLower()
});
dataGridView1.DataSource = degerler.ToList();
```

soyadı	Ad
yıldız	ALİ
öztürk	MEHMET
yıldırım	EMEL
kayalı	AYŞE

Böyle bir soru AD büyük harfle SOYAD küçük harfle getirir. Her okunan bilginin Anonymous Type değişkeni içine atıldığını da görelim.

```
var degerler = db.TBLOGRENCI.Select(x => new
{
    Ad = x.AD.ToUpper(),
    Soyadı = x.SOYAD.ToLower()
}).Where(x=>x.Ad!="Ali");
dataGridView1.DataSource = degerler.ToList();
```

Ad	Soyadı
MEHMET	öztürk
EMEL	yıldırım
AYŞE	kayalı
VEYSEL	sarı

Burada Select sorgusu yanında Where ifadeside kullanılmış oldu. Adı ali olanların dışındaki kişileri getirecek ve bu getirme esnasında Adları büyük soyadları küçük harfle gösterecek.

```
var degerler = db.TBLNOTLAR.Select(x =>
new
{
    OgrenciAd = x.OGR,
    Ortalaması = x.ORTALAMA,
    Durumu=x.DURUM
});
dataGridView1.DataSource = degerler.ToList();
```

OgrenciAd	Ortalaması	Durumu
1	60,00	True
2	74,00	True
3	60,00	True
4	53,00	True
5	92,00	True
1	49,00	False

Bu sorguda Durum hesaplanması SQL Server içine bir sorgu yazılarak yapılmıştı. Oradaki sorgunun çıktısını sadece ekrana getiriyor. Sorgu şu şekildeydi.

```
update tblnotlar set durum=1 where ortalama>=50
update tblnotlar set durum=0 where ortalama<50
```

Bu işlemi böyle yapmayalım. Linq sorgusu içinden Türkçe ifadelerle göstererek kendimiz kodla içinde yazalım. Ama burada yinede bu sorguların çıktısı DURUM sütününu true/false olarak yazıyor oranın içeriğine göre Türkçe Geçti/Kaldı yazdırıyoruz. Durum sütunu Class içindede Bool olarak bulunuyor olması gerekir.

```
var degerler = db.TBLNOTLAR.Select(x =>
new
{
    OgrenciAd = x.OGR,
    Ortalaması = x.ORTALAMA,
    Durumu = x.DURUM == true ? "Geçti" : "Kaldı"
});
dataGridView1.DataSource = degerler.ToList();
```

OgrenciAd	Ortalaması	Durumu
1	60,00	Geçti
2	74,00	Geçti
3	60,00	Geçti
4	53,00	Geçti
5	92,00	Geçti
1	49,00	Kaldı

SelectMany Sorguları

Select Many birden fazla tablodan veri çekme ve bunları birleştirme işine yarar. Daha önce yukarıda Select ifadesi içinde birden çok alan getirmiştik. Burada bunu biraz değiştiriyoruz. SelectMany ile aynı şekilde birden fazla sütün seçebiliriz.
Önce bir ilk deneme yapalım.

```
var degerler = db.TBLNOTLAR.SelectMany(x => db.TBLOGRENCI.Where(y => y.ID == x.OGR));
dataGridView1.DataSource = degerler.ToList();
```

ID	AD	SOYAD	FOTOGRAF	TBLNOTLAR
4	AYŞE	KAYALI	TEST	
5	VEYSEL	SARI	DENEME	
1	ALİ	YILDIZ		
2	MEHMET	ÖZTÜRK		
3	EMEL	YILDIRIM		
4	AYŞE	KAYALI	TEST	

Bu sorgu Notlar tablosundan bilgileri getirirken bu tabloda ÖğrID (ÖGR yazan sutün) yazan alan ile Öğrenciler tablosunda ID yazan alanlar eşitse bu kaydı sağlayan bilgileri getirir. Fakat biz bunu biraz geliştireceğiz.

```
var degerler = db.TBLNOTLAR.SelectMany(x => db.TBLOGRENCI.Where(y => y.ID ==
  x.OGR), (x, y) => new
{
    y.AD,
    x.ORTALAMA
});
dataGridView1.DataSource = degerler.ToList();
```

AD	ORTALAMA
ALİ	60,00
MEHMET	74,00
EMEL	60,00
AYŞE	53,00
VEYSEL	92,00

Burada SelectMany içinde çok sayıda şeyi getir diye başlıyoruz içinde kullanılan (x, y) ile de x, y değişkenlerini getir fakat bunları oluşturacağım yeni şu formatta (New ifadesi) getir deyip parantez içinde x ve y satırları içindeki AD ve ORTALAMA sütunlarını getir diyoruz. Dikkat buradaki x ve y ifadeleri aslında tabloların içinden okunan satırın tamamını temsil eden bir değişkendir. Yani x ifadesi NOTLAR tablosundaki bir satırı temsil ediyor. x.ORTALAMA

dediğimizde ise bunun içindeki ORTALAMA hücresinin bilgisini getirmiş oluyor. y ise ÖĞRENCİLER tablosunu temsil ediyor. Dolayısı ile y.AD dediğimizde ise öğrenciler tablosundan alınan bir satırın içindeki AD hücresinin içeriğini getirmiş oluyor.

OrderBy ve OrderByDescending Sıralama Komutları

```
var degerler = db.TBLOGRENCI.OrderBy(x => x.ID).Take(3);
dataGridView1.DataSource = degerler.ToList();
```

ID	AD	SOYAD	FOTOGRAF	TBLNOTLAR
1	ALİ	YILDIZ		
2	MEHMET	ÖZTÜRK		
3	EMEL	YILDIRIM		

Bu ifade ÖĞRENCİLER tablosunu ID ye göre sıralar ve ilk 3 ifadeyi bize getirir. Tersten sıralama içinde OrderByDescending kullanılır.

```
var degerler = db.TBLOGRENCI.OrderByDescending(x => x.ID).Take(3);
dataGridView1.DataSource = degerler.ToList();
```

ID	AD	SOYAD	FOTOGRAF	TBLNOTLAR
10	MELİSA	YILDIZ		
9	ALİ	ASLAN		
7	CEREN	YILMAZ		

Ada göre sıralarkende aşağıdaki şekilde yapabiliriz.

```
var degerler = db.TBLOGRENCI.OrderBy(x => x.AD);
dataGridView1.DataSource = degerler.ToList();
```

ID	AD	SOYAD	FOTOGRAF	TBLNOTLAR
1	ALİ	YILDIZ		
9	ALİ	ASLAN		
4	AYŞE	KAYALI	TEST	
7	CEREN	YILMAZ		

Skip Atlama İfadesi

Aşağıdaki kullanımda ID ye göre sıralar ve ilk 5 ifadeyi atlar.

```
var degerler = db.TBLOGRENCI.OrderBy(x => x.ID).Skip(5);
dataGridView1.DataSource = degerler.ToList();
```

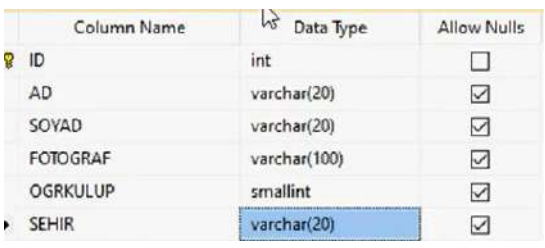
ID	AD	SOYAD	FOTOGRAF	TBLNOTLAR
7	CEREN	YILMAZ		
9	ALİ	ASLAN		
10	MELİSA	YILDIZ		

Then komutunun kullanımı

Bu komut Ad göre sıralama yaptıktan sonra bu seferde aynı isimdekileri soyada göre sıralaması gerektiğinde kullanılır.

Join birleştirme işlemleri

Tablo yapısı biraz değişsin.



Column Name	Data Type	Allow Nulls
ID	int	☐
AD	varchar(20)	☑
SOYAD	varchar(20)	☑
FOTOGRAF	varchar(100)	☑
OGRKULUP	smallint	☑
SEHIR	varchar(20)	☑

DESKTOP-07T8MF2\...- dbo.TBLOGRENCI

ID	AD	SOYAD	FOTOGRAF	OGRKULUP	SEHIR
1	ALİ	YILDIZ	NULL	1	Adana
2	MEHMET	ÖZTÜRK	NULL	1	Ankara
3	EMEL	YILDIRIM	NULL	2	İstanbul
4	AYŞE	KAYALI	TEST	3	Ankara
5	VEYSEL	SARI	DENEME	4	Ankara
7	CEREN	YILMAZ	NULL	2	İstanbul
9	ALİ	ASLAN	NULL	3	İstanbul
10	MELİSA	YILDIZ	DENEME	1	İstanbul

Şimdi bu tabloda Hangi şehirden kaç kişi olduğunu gösteren bir sorgu yazalım.

```
var degerler = db.TBLOGRENCI.OrderBy(x => x.SEHIR).GroupBy(y => y.SEHIR).
    Select(z => new { Şehir = z.Key, Toplam = z.Count() });
dataGridView1.DataSource = degerler.ToList();
```

Şehir	Toplam
Adana	1
Ankara	3
İstanbul	4

Sorgu şu şekilde çalıştı. Önce SEHİR e göre sıraladı. Ardından bu şehirleri GroupBy ile gruplandırdır. Sonra bunların içinden seçim işlemi yapıyor. Seçim yaparken ortaya çıkan listedeki satırları z değişkeni ile ifade etti. Bu z satırları içinden yeni bir seçim oluşturdu (new ifadesi). Bu seçimde z.Key ifadesi önceki şartlarımızda anahtarımız ne ise onu Şehir ifadesi içine at onunla göster demektir. Yani önceki sorgularda anahtarımız Şehir e göre sıralama ve gruplama yapmıştık o zaman Key mizi ŞEHİR dir. Bunu da Şehir= (Anonymois)ifadesi içine atıyor. Aynı zamanda Toplam= anonymous ifadesi içinde de bunların sayısını atıyor.

Min, Max vs Fonksiyonların kullanımı

```
label1.Text = db.TBLNOTLAR.Max(x => x.ORTALAMA).ToString();
```

Bu ifade en yüksek ortalamayı getirir. Aşağıdaki de en düşük sınav1 notunu getirir.

```
label1.Text = db.TBLNOTLAR.Min(x => x.SINAV1).ToString();
```

Soru: Kalan öğrenciler içinde en yüksek ortalama ile kalan öğrenciyi gösteren komutu yaz.

DESKTOP-07T8MF2\...ci - dbo.TBLURUN

Column Name	Data Type	Allow Nulls
ID	int	☐
AD	varchar(50)	☑
MARKA	varchar(50)	☑
FIYAT	int	☑
STOK	int	☑

DESKTOP-07T8MF2\...ci - dbo.TBLURUN

ID	AD	MARKA	FIYAT	STOK
1	BUZDOLABI	SIEMENS	4500	100
2	ÇAMAŞIR MAKİNESİ	SIEMENS	2000	25
3	LAPTOP	ACER	3500	40
4	BUZDOLABI	ARÇELİK	4600	30
5	BUZDOLABI	BEKO	3900	50
6	LAPTOP	TSOHIBA	5200	70

Aşağıdaki sorgu bütün stoklaı toplar, string olarak yazdırır.

```
label1.Text = db.TBLURUN.Sum(x => x.STOK).ToString();
```

Aşağıdaki sorgu Buzdolaplarının sayısını getirir. Burada FirsOrDefault ifadesi kullanılmalı. Onu bir sonraki derste görelim.

```
label1.Text = db.TBLURUN.Count(x => x.AD == "BUZDOLABI").ToString();
```

Aşağıdaki sorgu Ürün tablosundaki fiyat sütunundaki tüm bilgilerin ortalamasını getirecektir.

```
label1.Text = db.TBLURUN.Average(x => x.FIYAT).ToString();
```

Ödev: Buzdolapların ortalamasını getiren sorgu nasıl olur. Bunu da yazın.

Şimdi stokda en fazla olan ürünü getiren bir sorgu yazalım. Listeye bakarsak bu buzdolabıdır. Bunun adını getirsin.

```
label1.Text = (from x in db.TBLURUN
               orderby x.STOK descending
               select x.AD).First();
```

Ürünler tablosundaki her bir satırı tara. Bunu x ifadesi içine at. Yani ForEach gibi çalışacak. Sonra buradaki STOK sütününa göre Z-A ya sırala. Sonra bunların içinden AD sütün bilgisini getir. Buraya kadar olan kısım aslında bir listedir. Fakat biz labelda bir listeyi görüntüleyemeyiz. Dolayısı bu listedeki ilk ifadeyi getirmesi içinde .First()

komutu eklendi. Aynısının A dan Z ye sıralaması da aşağıdaki şekilde olur. Bu şekilde yazılan sorgular da Linq sorgusudur.

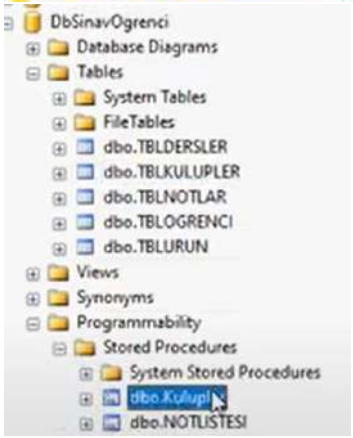
```
label1.Text = (from x in db.TBLURUN
               orderby x.STOK ascending
               select x.AD).First();
```

VT içinde Prosedure Yazımı ve Kullanımı

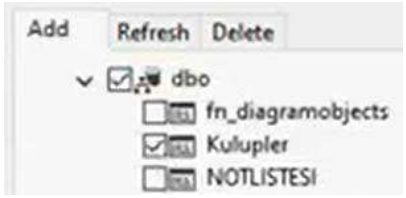
SQL Server içinde Bir Kulupler isminde bir prosedür oluşturalım. Bu prosedür Öğrenci tablosu ile Külüpler tablosunu Inner Join ile birleştirsin. Prosedürü yazdıktan sonr F5 basıp Exe haline getirelim. Artık ondan sonra bu ifade orada çalışıyor olacaktır. Aynı ekran "Execute Kulupler" yazarsak çalışacaktır. Klasörler içinde Kulupler prosedürünü görebiliriz.

```
CREATE PROCEDURE Kulupler
As
SELECT AD,SOYAD,SEHIR,KULUPAD FROM TBLOGRENCI
INNER JOIN TBLKULUPLER ON TBLOGRENCI.OGRKULUP=TBLKULUPLER.KULUPID
```

```
execute kulupler
```



VT deki bir Storeprosedure program içinde kullanabilmek için bunu Model içinde dahil edilmesi gerekiyor.



Kod içerisinde bu proseduru kullanmak için aşağıdaki gibi bir kod yazarız.

```
dataGridView1.DataSource = db.Kulupler().ToList();
```

Çıktısı bu şekilde oldu.

AD	SOYAD	SEHIR	KULUPAD
ALİ	YILDIZ	Adana	ALGORTİMA
MEHMET	ÖZTÜRK	Ankara	ALGORTİMA
EMEL	YILDIRIM	İstanbul	YAPAY ZEKA
AYŞE	KAYALI	Ankara	MAKİNE ÖĞ...

Tabi buradaki gibi kısa bir prosedurü kullanmak yerine kod içerisinde bir sorgu yazabiliriz. Ama 20 satırlık büyük bir prosedürü kod içerisinde yazmak yerine VT içinde bu prosedürü yazıp kod içerisinden bunun çağrılması daha iyi bir yol olur.

NOT: BU VİDEODAN SONRAKİ DERSLERDE MVC ANLATILIYOR. DERS NOTU OLUŞTURURKEN BURASI KULLANILABİLİR.
https://www.youtube.com/watch?v=sMJcLAki088&list=PLKnjBHu2xXNNiAXIaoH9rau4IQ95b6RYu&index=41

LinQ komutları- Entity Framework Denemeleri

Öncelikle bağlantıyı oluşturmak için şu videoyu bir incele..
https://www.youtube.com/watch?v=tK3eupNSdf0&list=PLKnjBHu2xXNNiAXIaoH9rau4IQ95b6RYu&index=4

Tüm Tabloyu Getirme

```
private void button1_Click(object sender, EventArgs e)
{
    TsDbEntities db = new TsDbEntities();
    dataGridView1.DataSource = db.Suppliers.ToList();
}
```

SupplierId	Name	Url
1	Çaytaş Makina S...	caytas
2	Baysu Ticaret	baysu

Sıralama yapma

```
    TsDbEntities db = new TsDbEntities();

    dataGridView1.DataSource = db.Suppliers.OrderBy(s=>s.Name).ToList();
```

SupplierId	Name	Url
2	Baysu Ticaret	baysu
1	Çaytaş Makina S...	caytas

ProductId=1 olan ürünü getirir

```
dataGridView1.DataSource = db.Products.Where(p => p.ProductId == 1).ToList();
```

Adı televizyon olan ürünü getirir.

```
dataGridView1.DataSource = db.Products.Where(p => p.Name == "Televizyon").ToList();
```

Adı içinde "sam" geçen ürünü yani Samsung yazan telefonu getirir.

```
dataGridView1.DataSource = db.Products.Where(p => p.Name.Contains("sam")).ToList();
```

Orders Tabloda herhangi bir Veri yoksa False döndürür.

```
bool deger = db.Orders.Any();
label1.Text = deger.ToString();
```

Tablodaki tüm bilgileri getirir.

```
var sorgu = from s in db.Suppliers select s;
dataGridView1.DataSource = sorgu.ToList();
```

Sadece Id ve Name alanını getirir.

```
var sorgu = from s in db.Suppliers
            select new
            {
                s.SupplierId,
                s.Name
            };

dataGridView1.DataSource = sorgu.ToList();
```

İki tabloyu Join ile birleştiriyor ikisindende belli alanları gösteriyor.

```
var sorgu = from s in db.Suppliers
            join p in db.Products
```

```
            on s.SupplierId equals p.SupplierId
            select new
            {
                UrunAdi = p.Name,
                FirmaAdi = s.Name
            };

dataGridView1.DataSource = sorgu.ToList();
```

Şöyle de yazılabilir.

```
dataGridView1.DataSource = (from s in db.Suppliers
                            join p in db.Products
                            on s.SupplierId equals p.SupplierId
                            select new
                            {
                                UrunAdi = p.Name,
                                FirmaAdi = s.Name
                            }).ToList();
```

LinQ Örnekleri

```
var studentsWithSameName = context.Students
                    .Where(s => s.FirstName == GetName())
                    .ToList();
```

```
-------------
var array = new[] { 5, 8, 4, 1, 2, 9 };
var filtered = array.Where(value => value > 4).Select(value => value);

foreach (var item in filtered)
{
Console.WriteLine(item);
} // bize 4'ten büyük olanları dönecektir.
```

```
int yastoplami = Data.Ogrencilertbls.Where(x => x.Sinif == "A").Sum(x => x.Yas);
    // A sınıfında olanların yaşlarının toplamını verecektir
```

```
---------------------- IQueryable  yada .AsQueryable(); işlemi
public HttpResponseMessage Get(string gender = "all", int? top = 0)
{
    IQueryable query = db.Employees;     // query'e select * from oluşturuldu ama gidip çekmiyor
kriteri verdikten sonra kriterime göre gidip çekicek
    gender = gender.ToLower();
    switch (gender)
    {
        case "all":
            break;
        case "male":
        case "female":
            query = query.Where(e => e.Gender.ToLower() == gender);
            break;
        default:
            return Request.CreateErrorResponse(HttpStatusCode.BadRequest, "geçersiz değer");
    }
    if (top > 0)
    {
```

```
        query = query.Take(top.Value); //nullable olduğundan dolayı .value diyerek değeri
alınır.
    }
    return Request.CreateResponse(HttpStatusCode.OK, query.ToList()); //ToList burada olduğundan
sorgumu atmaya burada gidiyor.
}
// Yani aslında ben query içinde sürekli kriterlerimi veriyorum ama her seferinde istek atıp
gelmiyor en son Tolist dediğim yerde son haliyle istek atıp geliyor.
```

KISACA: AsQuerible() fonksiyonu üzerine sorgu yazılabilir yani yeni kriterler eklenebilir hale getiriyor. İşlem süresince bir çok ek sorgu eklendi fakat bunlar hiçbir zaman veritabanına gidip işlem görmedi. Ne zamanki .ToList() ifadesini gördü. Bütün hepsinin etkisi bir kez veritabanına gidip gerçekleştirildi.
Aşağıdaki metodda üstteki satırda sorguda kullanılacak herşey hazırlandı fakat işlem görmedi. En sonuna AsQueryable() eklendi ve veritabanına götürmek üzere yada istenseydi üzerine ek sorgular yazılabilecek hale getirildi (ama burada yazılmadı). En son ikinci satırda ise ToList() eklenerek veritabanına tüm sorgu işlemleri gönderildi.

```
var products = ShopContext
                    .Products
                    .Where(i=>i.IsApproved && (i.Name.ToLower().Contains(searchString.ToLower())
|| i.Description.ToLower().Contains(searchString.ToLower())))
                    .AsQueryable();

                return products.ToList();
```

ViewBag, ViewData, TempData Arasındaki Farklar

Bu üç nesne arasındaki küçük ve kritik farklar vardır. Örneğin ViewBag nesnesi dynamic tipinde bir nesne olduğundan bununla alakalı hatalar compile time'da değil run time da yakalanır. Teknik anlamda ViewData nesnesinden farkı yoktur. Sözdizim olarak farklıdır.
En büyük ve önemli fark TempData ile diğer ikisi arasındadır. ViewData ve ViewBag nesnesi o anki HTTP istek içerisinde geçerlidir. Yaşam döngüsü bir sonraki isteğe kadardır. Ama TempData bir alt HTTP istek içinde geçerlidir. Yaşam döngüsü o anki ve bir sonraki HTTP istek içerinde geçerlidir. Daha iyi anlamak için farklı bir kaç örnek üzerinde görelim;

ViewData Örnek

```
//Controller Code
public ActionResult Index()
{
    List<string> Student = new List<string>();
    Student.Add("Jignesh");
    Student.Add("Tejas");
    Student.Add("Rakesh");

    ViewData["Student"] = Student;
    return View();
}
//page code
<ul>
    <% foreach (var student in ViewData["Student"] as List<string>)
        { %>
    <li><%: student%></li>
    <% } %>
</ul>
C#
```

ViewBag Örnek

```
//Controller Code
```

```
public ActionResult Index()
{
    List<string> Student = new List<string>();
    Student.Add("Jignesh");
    Student.Add("Tejas");
    Student.Add("Rakesh");

    ViewBag.Student = Student;
    return View();
}
//page code
<ul>
    <% foreach (var student in ViewBag.Student)
        { %>
    <li><%: student%></li>
    <% } %>
</ul>

//Controller Code
public ActionResult Index()
{
    List<string> Student = new List<string>();
    Student.Add("Jignesh");
    Student.Add("Tejas");
    Student.Add("Rakesh");

    TempData["Student"] = Student;
    return View();
}
//page code
<ul>
    <% foreach (var student in TempData["Student"] as List<string>)
        { %>
    <li><%: student%></li>
    <% } %>
</ul>
```

ViewData	**ViewBag**	**TempData**
Anahtar-Değer Sözlüğü koleksiyonudur	Bu bir tür nesnesidir	Anahtar-Değer Sözlüğü koleksiyonudur
ViewData bir sözlük nesnesidir ve ControllerBase sınıfının özelliğidir.	ViewBag, ControllerBase sınıfının Dynamic özelliğidir.	TempData bir sözlük nesnesidir ve controllerBase sınıfının özelliğidir.
ViewData, ViewBag'den Daha Hızlıdır	ViewBag, ViewData'dan daha yavaştır	NA
ViewData, MVC 1.0'da sunulmuştur ve MVC 1.0 ve üzeri sürümlerde mevcuttur	ViewBag, MVC 3.0'da tanıtıldı ve MVC 3.0 ve üzeri sürümlerde mevcut	TempData ayrıca MVC1.0'da tanıtılmıştır ve MVC 1.0 ve üzeri sürümlerde mevcuttur.
ViewData ayrıca .net çerçevesi 3.5 ve üzeri ile çalışır	ViewBag yalnızca .net çerçevesi 4.0 ve üstü ile çalışır	TempData ayrıca .net çerçevesi 3.5 ve üzeri ile çalışır
Numaralandırma sırasında Tür Dönüştürme kodu gerekli	Derinlemesine, ViewBag dinamik olarak kullanılır, bu nedenle numaralandırma sırasında dönüştürme yazmaya gerek yoktur.	Numaralandırma sırasında Tür Dönüştürme kodu gerekli
Yeniden yönlendirme meydana gelirse değeri null olur.	ViewData ile aynı	TempData, ardışık iki istek arasında veri iletmek için kullanılır.

Yalnızca geçerli istek sırasında bulunur.	ViewData ile aynı	TempData yalnızca mevcut ve sonraki istek sırasında çalışır

Türkçe kaynak: https://www.muratoner.net/aspnet/aspnet-mvc/aspnet-mvc-viewbag-viewdata-tempdata-ile-veri-tasima

ViewData

Controller sınıfları ile View sayfaları arasında ViewDataDictionary(ViewData) sınıfı ile taşınmaktadır. ViewData nesnelerine veri aktarıp, okuyabiliriz. Şimdi basit bir kodlar ViewData nesnesine değer aktarımını görelim.

```
ViewData["CurrentTime"] = DateTime.Now;ViewBag
```

ViewBag ise ASP.NET MVC3 te C# 4 ile gelen dynamic anahtar kelimesinin getirdiği bir yeniliktir. ViewData'nın dinamik(run time binding) halidir. Söz dizimi

```
ViewBag.CurrentTime = DateTime.Now;
```

TempData

```
TempData["CurrentTime"] = DateTime.Now;
```

Bu üç nesne arasındaki küçük ve kritik farklar vardır. Örneğin ViewBag nesnesi dynamic tipinde bir nesne olduğundan bununla alakalı hatalar compile time'da değil run time da yakalanır. Teknik anlamda ViewData nesnesinden farkı yoktur. Sözdizim olarak farklıdır.

En büyük ve önemli fark TempData ile diğer ikisi arasındadır. ViewData ve ViewBag nesnesi o anki HTTP istek içerisinde geçerlidir. Yaşam döngüsü bir sonraki isteğe kadardır. Ama TempData bir alt HTTP istek içinde geçerlidir. Yaşam döngüsü o anki ve bir sonraki HTTP istek içerinde geçerlidir. Daha iyi anlamak için farklı bir kaç örnek üzerinde görelim;

Örnek 1

Üç nesnenin örnek kullanımı ve çıktısı.

Controller

```
public ActionResult Index()
{
    ViewBag.Message1 = "ViewBag mesaj!";
    ViewData["Message2"] = "ViewData mesaj!";
    TempData["Message3"] = "TempData mesaj!";

    return View();
}
```

View

```
<h2>@ViewBag.Message1</h2>
<h2>@ViewData["Message2"]</h2>
<h2>@TempData["Message3"]</h2>
```

Çıktı

ViewBag mesaj! ViewData mesaj! TempData mesaj!

Örnek 2

TempData nesnesinin farklı özelliği.

Controller/Index

```
public ActionResult Index()
```

```
{
    ViewBag.Message1 = "ViewBag mesaj!";
    ViewData["Message2"] = "ViewData mesaj!";
    TempData["Message3"] = "TempData mesaj!";

    return RedirectToAction("About");
}
```

Contoller/About

```
public ActionResult About()
{
    var message = TempData["Message3"];

    return View();
}
```

Index metodu içerisinde oluşturduğumuz TempData About metoduna aktarılır ama diger iki nesne debug ta baktığımızda null olacaktır. TempData ile geçirdiğimiz veriyi About.cshtml içerisinde bir önceki örnekteki gibi kullanabiliriz.

TempData için sakıncalı bir durum yönlendirdiğimiz About sayfası içerisindeyken sayfası yenilediğimiz zaman veriyi kaybederiz. Bundan dolayı TempData yerine bu işlemleri daha sonra göreceğimiz ViewModel mantığıyla halletmeliyiz. İpucu olması açısından aynı işi yapan ViewModel mantığını da kodla örnekleyelim.

Controller/Index

```
public ActionResult Index()
{
    // veriyi veri tabanına kaydettikten sonra id sini alıyoruz
    var id = Repository.SaveData("foo");
    // diger sayfaya parametre olarak, veri tabanına son kaydettiğimiz kayıtın id sini
veriyoruz.
    return RedirectToAction("About", "Home", new { id = id });
}
```

Contoller/About

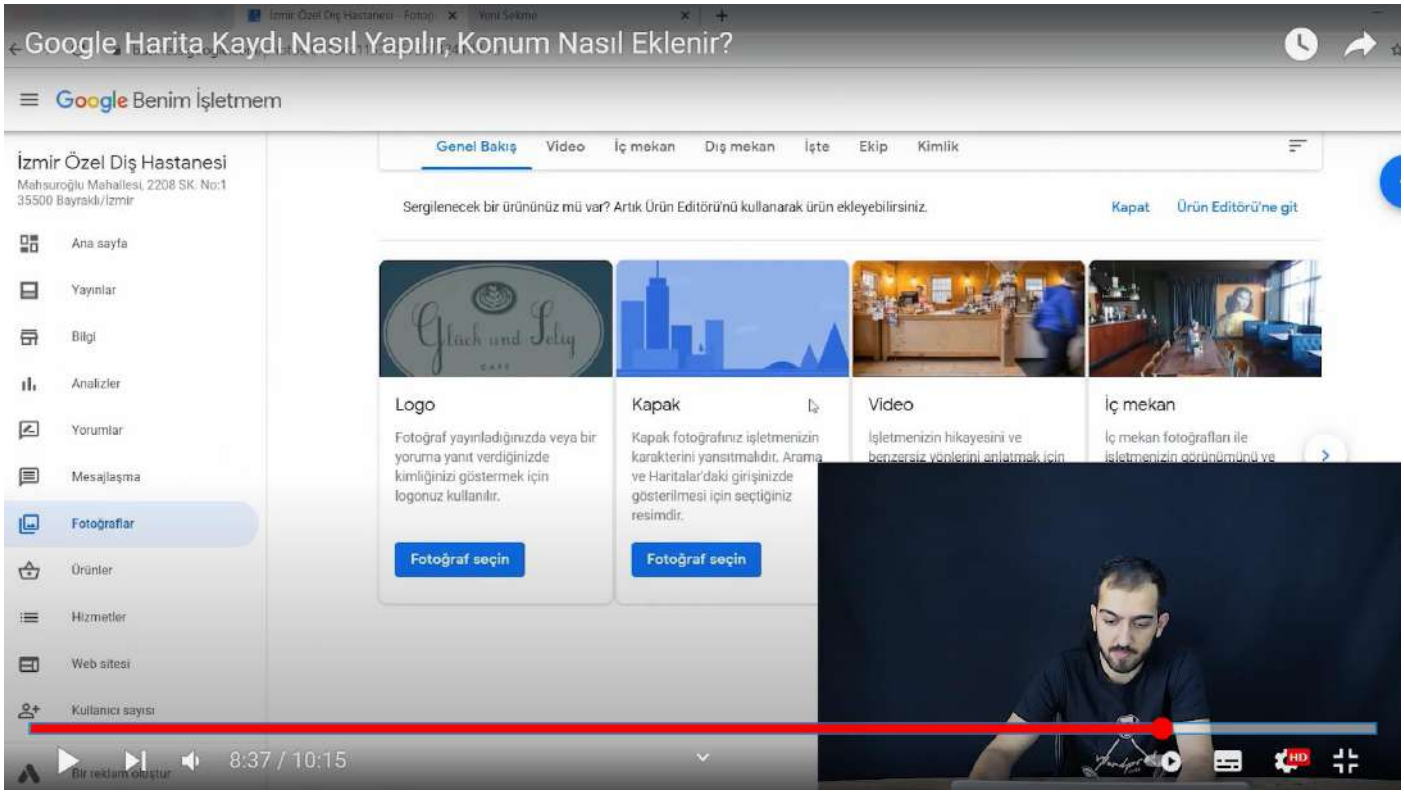
```
// Index metodundan gelen id değerini alıyoruz.
public ActionResult About(string id)
{
    // gelen id değerinden tekrar kayıtı veri tabanından çekiyoruz.
    var model = new MyViewModel
    {
        Foo = Repository.GetData(id)
    };
    return View(model);
}
```

Neden veritabanına kayıt yapıp tekrar diğer sayfadan bu kayıta ulaşmaya çalışan bir örneği, önceki TempData örneğinin alternatifi olarak verdik. Çünkü gerçek hayatta aslında bir metottan diger metoda veriyi böyle bir senaryo için aktarmak isteriz. İleride yapaçağımız örneklerle daha da açıklayıcı olacaktır.
Peki bu nesneleri ne zaman kullanmalıyız.

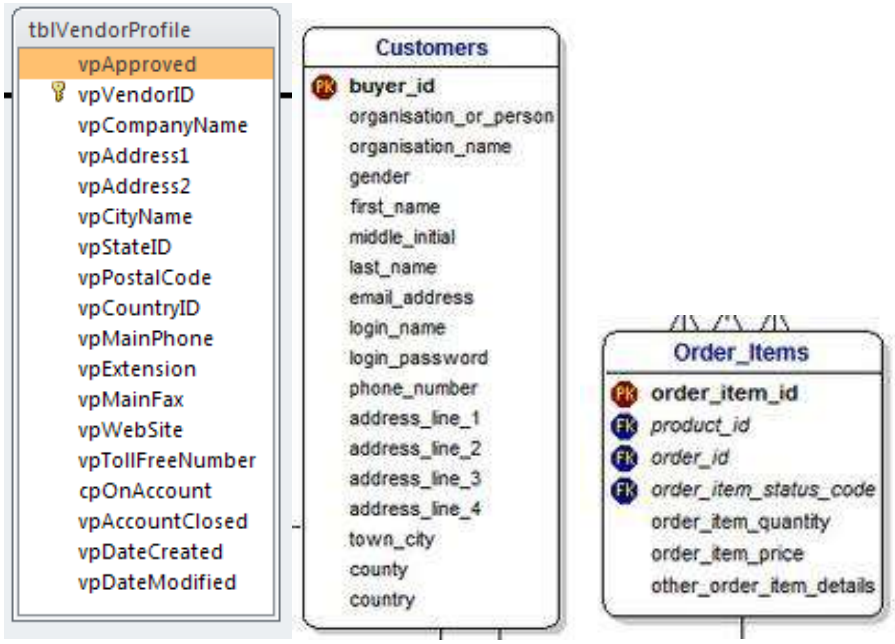
- Model sınıfımıza ait bir veriyi açılır listeden (drop down list) seçeceksek bu listenin verisi için kullanabiliriz.
- Küçük ölçekli veriler.
- Kullanıcıya verilen uyarı mesajları.
- Örneğin kullanıcı kayıt olduktan sonra kullanıcıya gösterilen profil özeti ekranı.

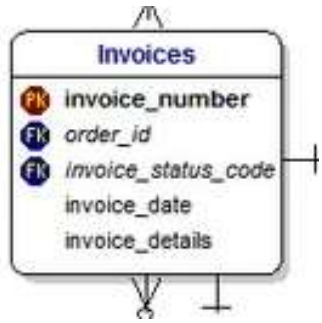
Kısacası geçici olarak tanımlayabileceğimiz küçük ölçekli verileri içeren bir çok iş için kullanabiliriz.

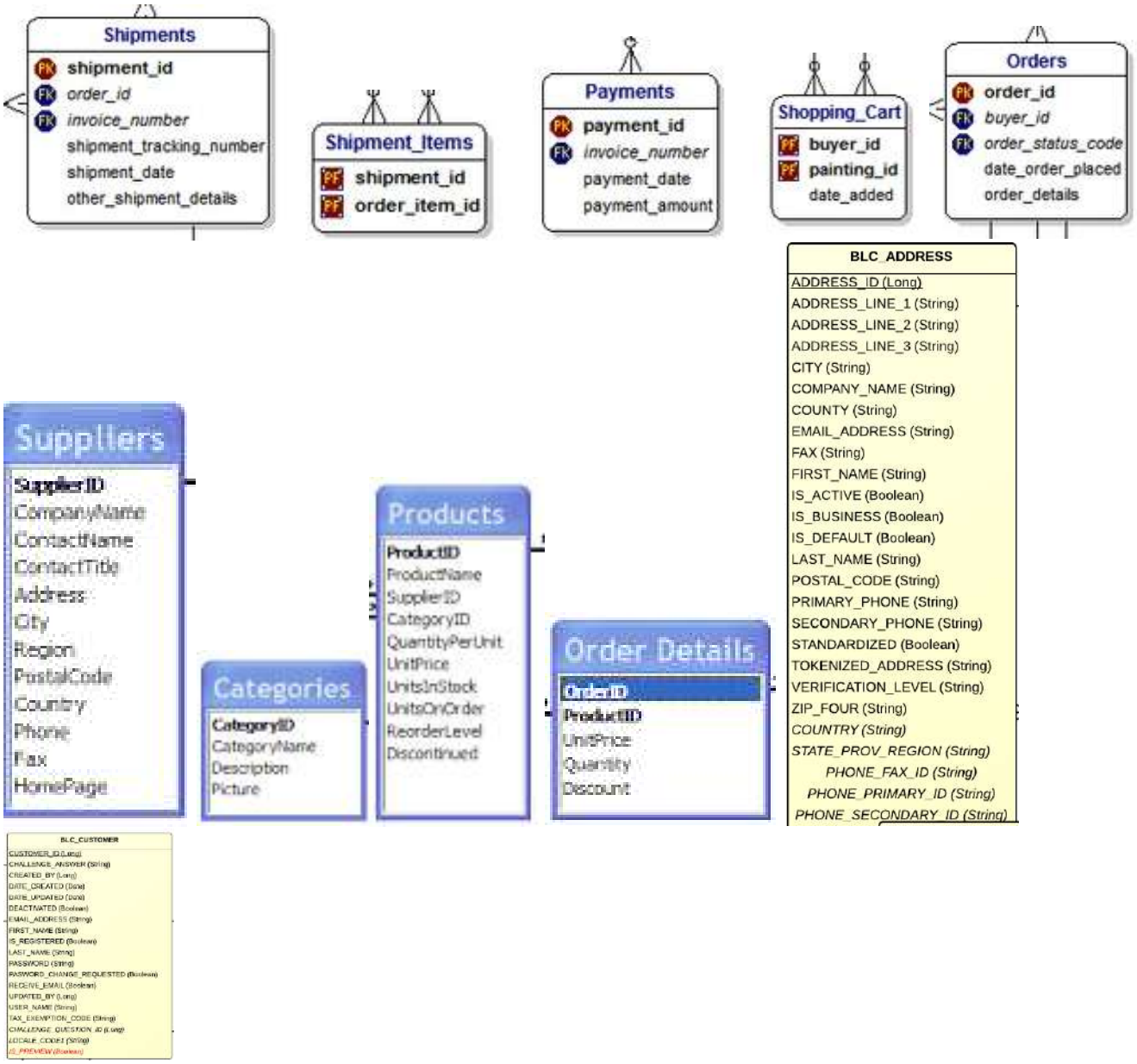
Firmalara Sayfa Oluştururken Örnek Alınabilecek Bilgiler

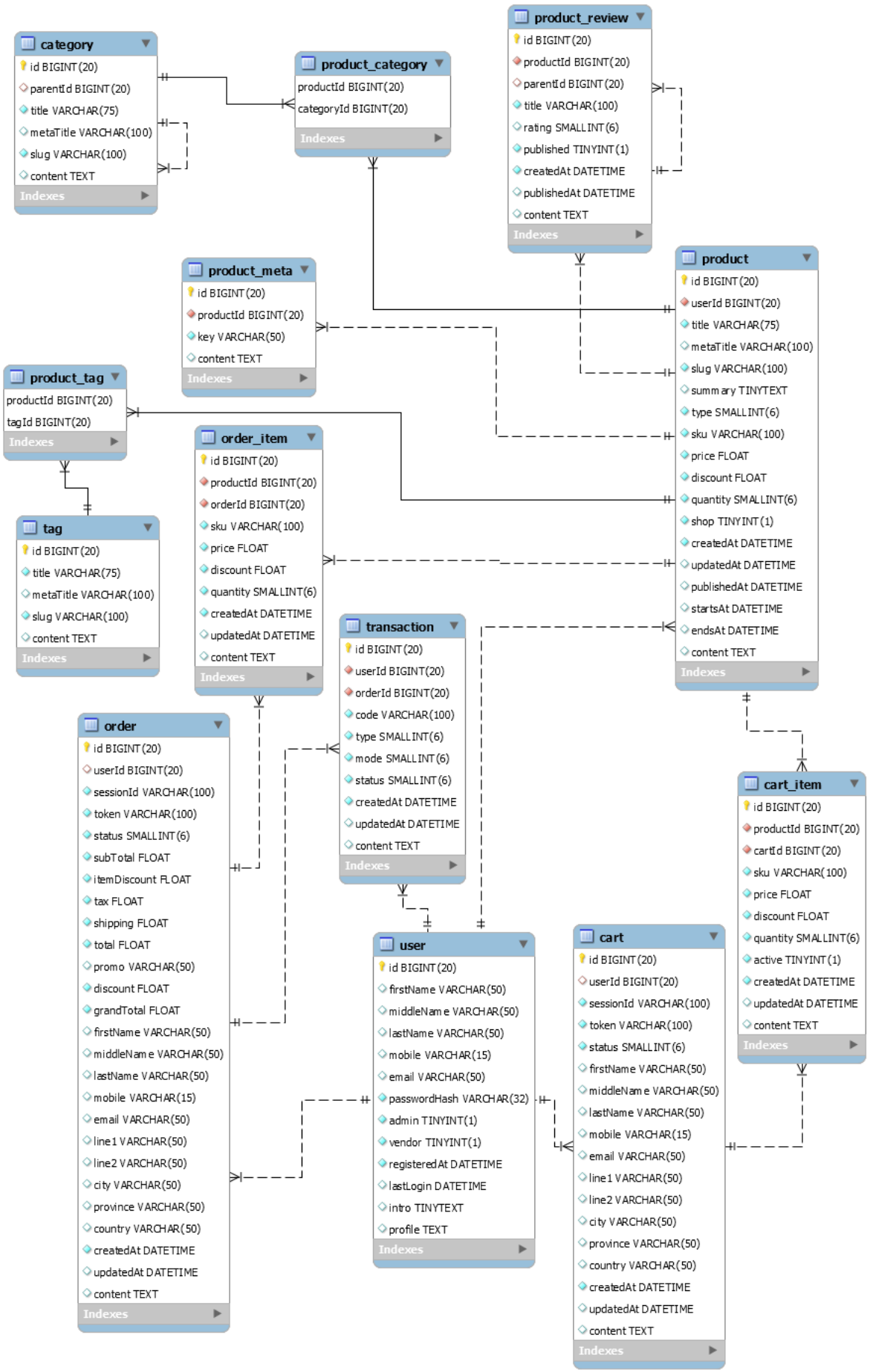


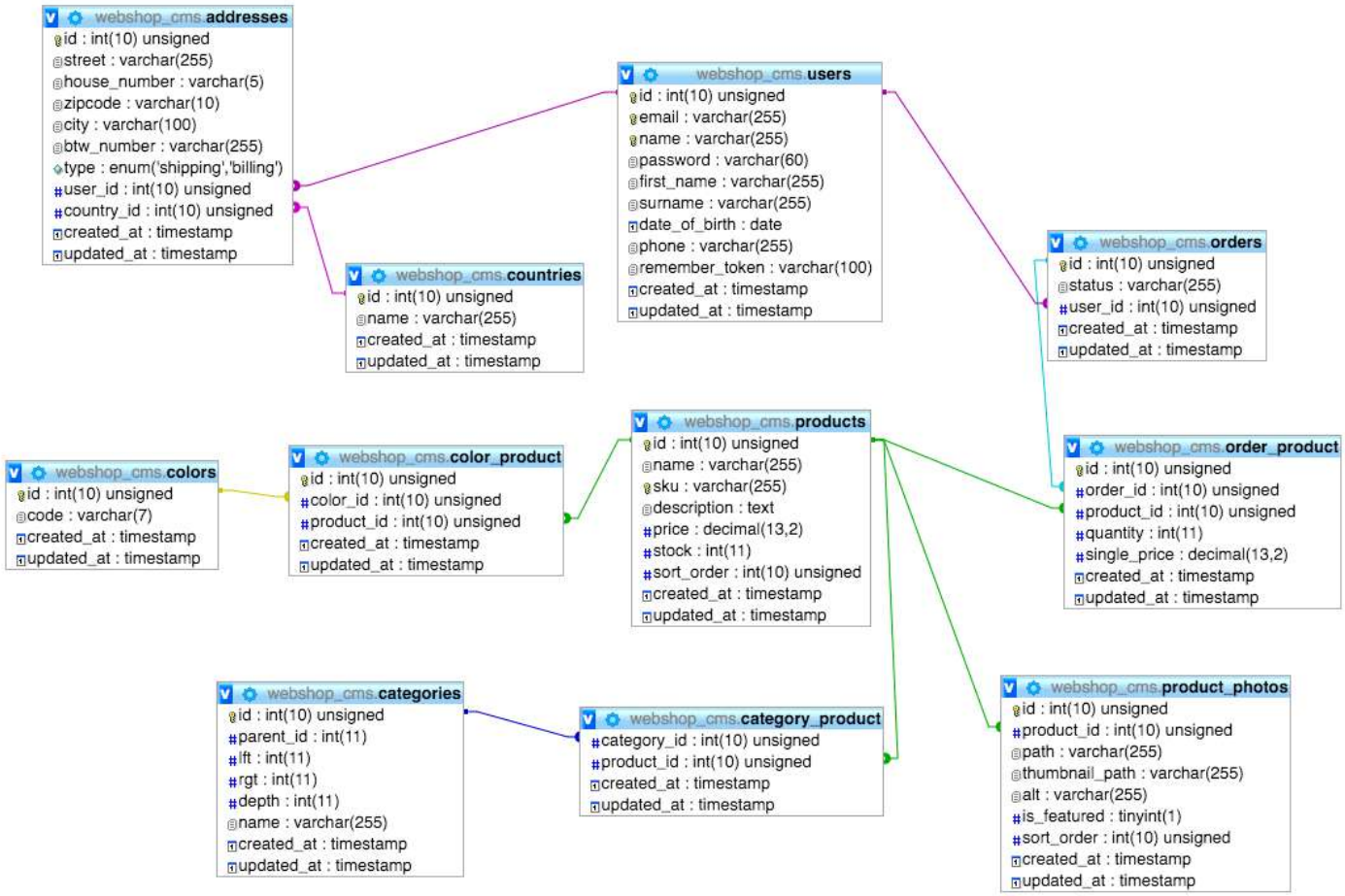
Örnek Veritabanı Haritaları

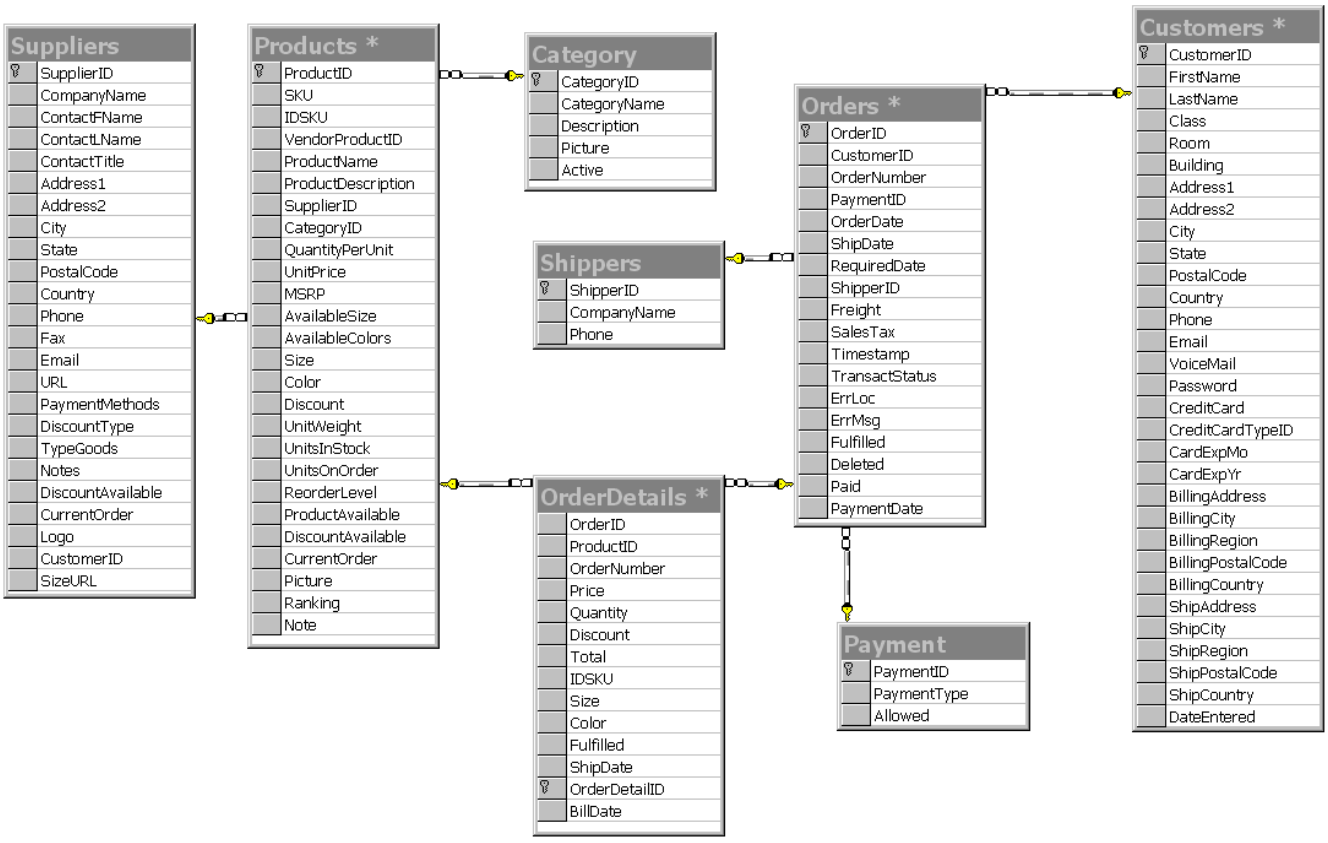





Shipments
shipment_id
order_id
invoice_number
shipment_tracking_number
shipment_date
other_shipment_details
Shipment_Items
shipment_id
order_item_id
Payments
payment_id
invoice_number
payment_date
payment_amount
Shopping_Cart
buyer_id
painting_id
date_added
Orders
order_id
buyer_id
order_status_code
date_order_placed
order_details
BLC_ADDRESS
ADDRESS_ID (Long)
ADDRESS_LINE_1 (String)
ADDRESS_LINE_2 (String)
ADDRESS_LINE_3 (String)
CITY (String)
COMPANY_NAME (String)
COUNTY (String)
EMAIL_ADDRESS (String)
FAX (String)
FIRST_NAME (String)
IS_ACTIVE (Boolean)
IS_BUSINESS (Boolean)
IS_DEFAULT (Boolean)
LAST_NAME (String)
POSTAL_CODE (String)
PRIMARY_PHONE (String)
SECONDARY_PHONE (String)
STANDARDIZED (Boolean)
TOKENIZED_ADDRESS (String)
VERIFICATION_LEVEL (String)
ZIP_FOUR (String)
COUNTRY (String)
STATE_PROV_REGION (String)
PHONE_FAX_ID (String)
PHONE_PRIMARY_ID (String)
PHONE_SECONDARY_ID (String)
Suppliers
SupplierID
CompanyName
ContactName
ContactTitle
Address
City
Region
PostalCode
Country
Phone
Fax
HomePage
Products
ProductID
ProductName
SupplierID
CategoryID
QuantityPerUnit
UnitPrice
UnitsInStock
UnitsOnOrder
ReorderLevel
Discontinued
Order Details
OrderID
ProductID
UnitPrice
Quantity
Discount
Categories
CategoryID
CategoryName
Description
Picture
BLC_CUSTOMER


category
id BIGINT(20)
parentId BIGINT(20)
title VARCHAR(75)
metaTitle VARCHAR(100)
slug VARCHAR(100)
content TEXT
Indexes
product_category
productId BIGINT(20)
categoryId BIGINT(20)
Indexes
product_review
id BIGINT(20)
productId BIGINT(20)
parentId BIGINT(20)
title VARCHAR(100)
rating SMALLINT(6)
published TINYINT(1)
createdAt DATETIME
publishedAt DATETIME
content TEXT
Indexes
product_meta
id BIGINT(20)
productId BIGINT(20)
key VARCHAR(50)
content TEXT
Indexes
product
id BIGINT(20)
userId BIGINT(20)
title VARCHAR(75)
metaTitle VARCHAR(100)
slug VARCHAR(100)
summary TINYTEXT
type SMALLINT(6)
sku VARCHAR(100)
price FLOAT
discount FLOAT
quantity SMALLINT(6)
shop TINYINT(1)
createdAt DATETIME
updatedAt DATETIME
publishedAt DATETIME
startsAt DATETIME
endsAt DATETIME
content TEXT
Indexes
product_tag
productId BIGINT(20)
tagId BIGINT(20)
Indexes
order_item
id BIGINT(20)
productId BIGINT(20)
orderId BIGINT(20)
sku VARCHAR(100)
price FLOAT
discount FLOAT
quantity SMALLINT(6)
createdAt DATETIME
updatedAt DATETIME
content TEXT
Indexes
tag
id BIGINT(20)
title VARCHAR(75)
metaTitle VARCHAR(100)
slug VARCHAR(100)
content TEXT
Indexes
transaction
id BIGINT(20)
userId BIGINT(20)
orderId BIGINT(20)
code VARCHAR(100)
type SMALLINT(6)
mode SMALLINT(6)
status SMALLINT(6)
createdAt DATETIME
updatedAt DATETIME
content TEXT
Indexes
order
id BIGINT(20)
userId BIGINT(20)
sessionId VARCHAR(100)
token VARCHAR(100)
status SMALLINT(6)
subTotal FLOAT
itemDiscount FLOAT
tax FLOAT
shipping FLOAT
total FLOAT
promo VARCHAR(50)
discount FLOAT
grandTotal FLOAT
firstName VARCHAR(50)
middleName VARCHAR(50)
lastName VARCHAR(50)
mobile VARCHAR(15)
email VARCHAR(50)
line1 VARCHAR(50)
line2 VARCHAR(50)
city VARCHAR(50)
province VARCHAR(50)
country VARCHAR(50)
createdAt DATETIME
updatedAt DATETIME
content TEXT
Indexes
user
id BIGINT(20)
firstName VARCHAR(50)
middleName VARCHAR(50)
lastName VARCHAR(50)
mobile VARCHAR(15)
email VARCHAR(50)
passwordHash VARCHAR(32)
admin TINYINT(1)
vendor TINYINT(1)
registeredAt DATETIME
lastLogin DATETIME
intro TINYTEXT
profile TEXT
Indexes
cart
id BIGINT(20)
userId BIGINT(20)
sessionId VARCHAR(100)
token VARCHAR(100)
status SMALLINT(6)
firstName VARCHAR(50)
middleName VARCHAR(50)
lastName VARCHAR(50)
mobile VARCHAR(15)
email VARCHAR(50)
line1 VARCHAR(50)
line2 VARCHAR(50)
city VARCHAR(50)
province VARCHAR(50)
country VARCHAR(50)
createdAt DATETIME
updatedAt DATETIME
content TEXT
Indexes
cart_item
id BIGINT(20)
productId BIGINT(20)
cartId BIGINT(20)
sku VARCHAR(100)
price FLOAT
discount FLOAT
quantity SMALLINT(6)
active TINYINT(1)
createdAt DATETIME
updatedAt DATETIME
content TEXT
Indexes


webshop_cms.addresses
id : int(10) unsigned
street : varchar(255)
house_number : varchar(5)
zipcode : varchar(10)
city : varchar(100)
btw_number : varchar(255)
type : enum('shipping','billing')
user_id : int(10) unsigned
country_id : int(10) unsigned
created_at : timestamp
updated_at : timestamp
webshop_cms.users
id : int(10) unsigned
email : varchar(255)
name : varchar(255)
password : varchar(60)
first_name : varchar(255)
surname : varchar(255)
date_of_birth : date
phone : varchar(255)
remember_token : varchar(100)
created_at : timestamp
updated_at : timestamp
webshop_cms.countries
id : int(10) unsigned
name : varchar(255)
created_at : timestamp
updated_at : timestamp
webshop_cms.orders
id : int(10) unsigned
status : varchar(255)
user_id : int(10) unsigned
created_at : timestamp
updated_at : timestamp
webshop_cms.products
id : int(10) unsigned
name : varchar(255)
sku : varchar(255)
description : text
price : decimal(13,2)
stock : int(11)
sort_order : int(10) unsigned
created_at : timestamp
updated_at : timestamp
webshop_cms.colors
id : int(10) unsigned
code : varchar(7)
created_at : timestamp
updated_at : timestamp
webshop_cms.color_product
id : int(10) unsigned
color_id : int(10) unsigned
product_id : int(10) unsigned
created_at : timestamp
updated_at : timestamp
webshop_cms.order_product
id : int(10) unsigned
order_id : int(10) unsigned
product_id : int(10) unsigned
quantity : int(11)
single_price : decimal(13,2)
created_at : timestamp
updated_at : timestamp
webshop_cms.categories
id : int(10) unsigned
parent_id : int(11)
lft : int(11)
rgt : int(11)
depth : int(11)
name : varchar(255)
created_at : timestamp
updated_at : timestamp
webshop_cms.category_product
category_id : int(10) unsigned
product_id : int(10) unsigned
created_at : timestamp
updated_at : timestamp
webshop_cms.product_photos
id : int(10) unsigned
product_id : int(10) unsigned
path : varchar(255)
thumbnail_path : varchar(255)
alt : varchar(255)
is_featured : tinyint(1)
sort_order : int(10) unsigned
created_at : timestamp
updated_at : timestamp


Suppliers
SupplierID
CompanyName
ContactFName
ContactLName
ContactTitle
Address1
Address2
City
State
PostalCode
Country
Phone
Fax
Email
URL
PaymentMethods
DiscountType
TypeGoods
Notes
DiscountAvailable
CurrentOrder
Logo
CustomerID
SizeURL
Products *
ProductID
SKU
IDSKU
VendorProductID
ProductName
ProductDescription
SupplierID
CategoryID
QuantityPerUnit
UnitPrice
MSRP
AvailableSize
AvailableColors
Size
Color
Discount
UnitWeight
UnitsInStock
UnitsOnOrder
ReorderLevel
ProductAvailable
DiscountAvailable
CurrentOrder
Picture
Ranking
Note
Category
CategoryID
CategoryName
Description
Picture
Active
Shippers
ShipperID
CompanyName
Phone
OrderDetails *
OrderID
ProductID
OrderNumber
Price
Quantity
Discount
Total
IDSKU
Size
Color
Fulfilled
ShipDate
OrderDetailID
BillDate
Orders *
OrderID
CustomerID
OrderNumber
PaymentID
OrderDate
ShipDate
RequiredDate
ShipperID
Freight
SalesTax
Timestamp
TransactStatus
ErrLoc
ErrMsg
Fulfilled
Deleted
Paid
PaymentDate
Payment
PaymentID
PaymentType
Allowed
Customers *
CustomerID
FirstName
LastName
Class
Room
Building
Address1
Address2
City
State
PostalCode
Country
Phone
Email
VoiceMail
Password
CreditCard
CreditCardTypeID
CardExpMo
CardExpYr
BillingAddress
BillingCity
BillingRegion
BillingPostalCode
BillingCountry
ShipAddress
ShipCity
ShipRegion
ShipPostalCode
ShipCountry
DateEntered

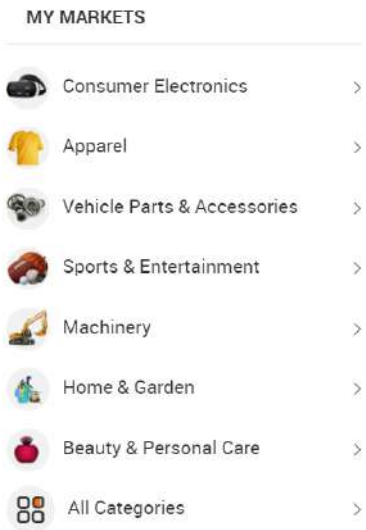
Kategoriler

Accounting
Airlines/Aviation
Alternative Dispute Resolution
Alternative Medicine
Animation
Apparel & Fashion
Architecture & Planning
Arts and Crafts
Automotive
Aviation & Aerospace
Banking
Biotechnology
Broadcast Media
Building Materials
Business Supplies and Equipment
Capital Markets
Chemicals
Civic & Social Organization
Civil Engineering
Commercial Real Estate
Computer & Network Security
Computer Games
Computer Hardware
Computer Networking
Computer Software
Construction
Consumer Electronics
Consumer Goods
Consumer Services
Cosmetics
Dairy
Defense & Space
Design
Education Management
E-Learning
Electrical/Electronic Manufacturing
Entertainment
Environmental Services
Events Services
Executive Office
Facilities Services
Farming
Financial Services
Fine Art
Fishery
Food & Beverages
Food Production
Fund-Raising
Furniture
Gambling & Casinos
Glass, Ceramics & Concrete
Government Administration
Government Relations
Graphic Design
Health, Wellness and Fitness
Higher Education
Hospital & Health Care
Hospitality
Human Resources
Import and Export
Individual & Family Services
Industrial Automation
Information Services
Information Technology and Services
Insurance
International Affairs
International Trade and Development
Internet
Investment Banking
Investment Management
Judiciary
Law Enforcement
Law Practice
Legal Services
Legislative Office
Leisure, Travel & Tourism
Libraries
Logistics and Supply Chain
Luxury Goods & Jewellery
Machinery
Management Consulting
Maritime
Marketing and Advertising
Market Research
Mechanical or Industrial Engineering
Media Production
Medical Devices
Medical Practice
Mental Health Care
Military
Mining & Metals
Motion Pictures and Film
Museums and Institutions
Music
Nanotechnology
Newspapers
Nonprofit Organization Management
Oil & Energy
Online Media
Outsourcing/Offshoring
Package/Freight Delivery
Packaging and Containers
Paper & Forest Products
Performing Arts
Pharmaceuticals
Philanthropy
Photography
Plastics
Political Organization
Primary/Secondary Education
Printing
Professional Training & Coaching
Program Development
Public Policy
Public Relations and Communications
Public Safety
Publishing
Railroad Manufacture
Ranching
Real Estate
Recreational Facilities and Services
Religious Institutions
Renewables & Environment
Research
Restaurants
Retail
Security and Investigations
Semiconductors
Shipbuilding
Sporting Goods
Sports
Staffing and Recruiting
Supermarkets
Telecommunications
Textiles
Think Tanks
Tobacco
Translation and Localization
Transportation / Trucking / Railroad
Utilities
Venture Capital & Private Equity
Veterinary
Warehousing
Wholesale
Wine and Spirits
Wireless
Writing and Editing

	n	%	*National data* %
Representation of industry categories			
Agriculture, forestry, fishing	3	0.9	4.9
Mining	32	9.9	1.0
Manufacturing	78	24.2	13.6
Electricity, gas & water	13	4.0	1.0
Construction	6	1.9	7.2
Wholesale & retail trade	17	5.3	20.6
Communications	20	6.2	1.8
Finance, property & business services	60	18.6	13.6
Public administration & defence	10	3.1	4.6
Recreation	12	3.0	2.3
Transport & storage	6	1.9	4.6
Health and community service	7	2.3	9.2
Consulting	13	4.0	not specified
Information technology	8	2.5	not specified
Education	8	2.5	7.2
Hospitality	10	3.1	not specified
Other	29	9.0	8.3
Type of employing organization			
Public sector	53	16.5	26.0
Private sector	266	82.6	73.0

Sources: ABS (1995b) Labour Force Statistics for industry categories; ABS (1995c) Employed Wage & Salary Earners, for type of employing organization.

Notes
n = 322
Missing data account for totals less than 322



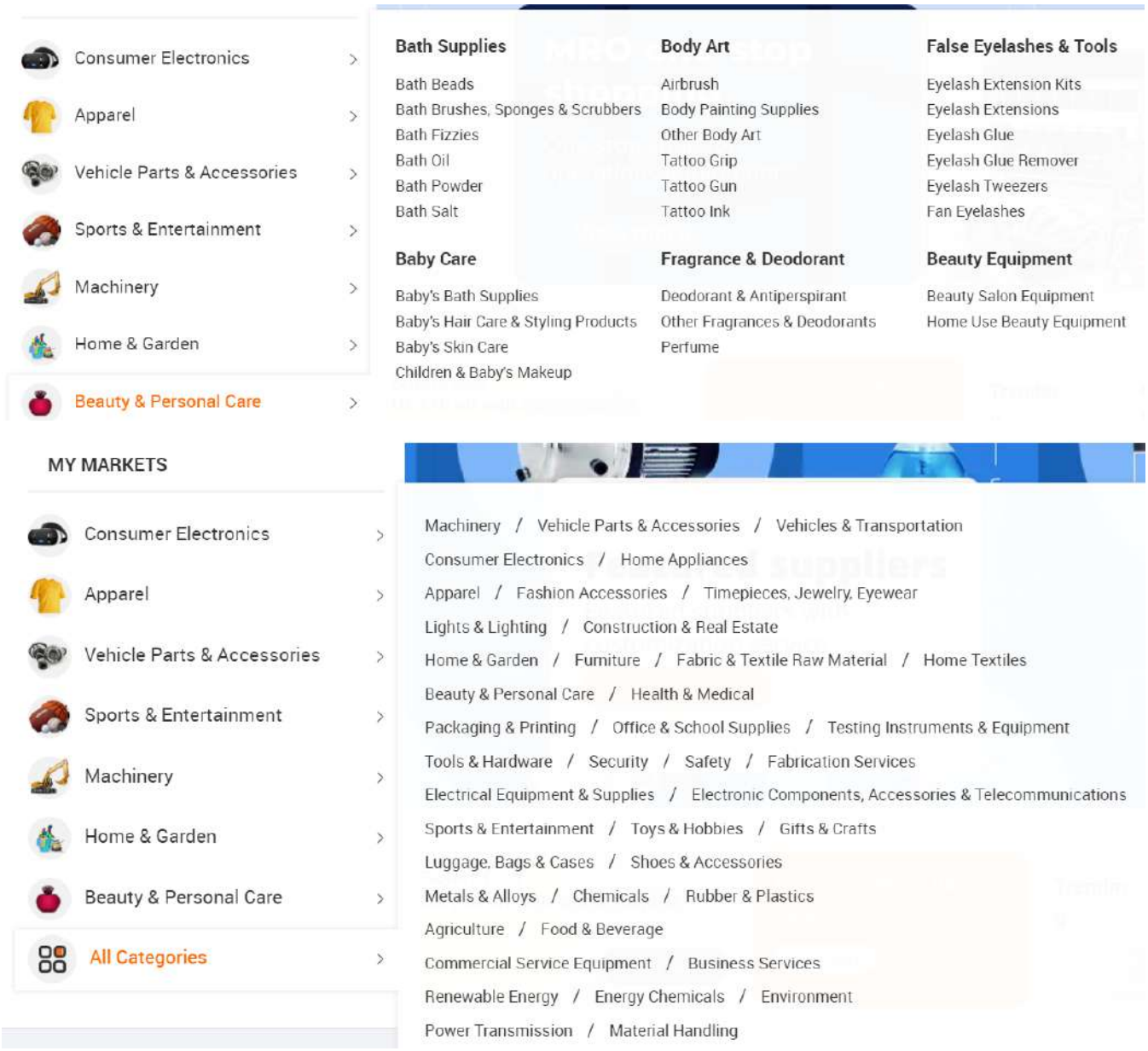

MY MARKETS
Consumer Electronics
Apparel
Vehicle Parts & Accessories
Sports & Entertainment
Machinery
Home & Garden
Beauty & Personal Care
All Categories
Camera, Photo & Accessories
Action & Sports Camera
Action & Sports Camera Accessory
Baby & Pet Monitor
Backgrounds
Battery Grip
Camera Filters
Chargers,Batteries & Power Supplies
Battery Accessories
Battery Case
Battery Charger
Charger & Adapter
Charging Stand & Holder
Charging Station & Power Station
Commonly Used Accessories & Parts
Adapters & Connectors
Audio Sound Cards & Mixers
Cable
Card Readers
Cleaners
Digital Lighters & Parts
Computer Hardware & Software
All in One Computers
Blank Disks
Computer Cases & Towers
CPUs
Desktops
Fans & Cooling
Earphone & Headphone & Accessories
Earphone Accessories
Earphones & Headphones
Gaming Earphones & Headsets
Electronic Publications
Electronic Books
Music
MY MARKETS
Consumer Electronics
Apparel
Vehicle Parts & Accessories
Sports & Entertainment
Machinery
Home & Garden
Beauty & Personal Care
All Categories
Baby Clothing
Baby bibs
Baby clothing sets
Baby dresses
Baby hoodies&sweatshirts
Baby jackets&outwears
Baby pants&shorts
Ethnic Clothing
Africa Clothing
American Clothing
Asia & Pacific Islands Clothing
European Clothing
India & Pakistan Clothing
Islamic Clothing
Garment & Processing Accessories
Badges
Boning
Braid
Buckles
Buttons
Cords
Men's Clothing
Men's Coats
Men's Down Coat
Men's Hoodies & Sweatshirts
Men's Jackets
Men's Jeans
Men's Leggings
Childrens Clothing
Boys Clothing
Family Matching Outfits
Girls Clothing
Novelty & Special Use
Costumes
Stage & Dance Wear
Uniforms
MY MARKETS
Consumer Electronics
Apparel
Vehicle Parts & Accessories
Sports & Entertainment
Machinery
Home & Garden
Beauty & Personal Care
All Categories
Automotive Parts & Accessories
Air Conditioning Systems
Auto Body Systems
Auto Brake Systems
Auto Drive Systems
Auto Electrical Systems
Auto Electronics
Aviation Parts & Accessories
Aviation Accessories
Aviation Parts
Bus Parts & Accessories
Bus Accessories
Bus Parts
ATV/UTV Parts & Accessories
Container Parts & Accessories
Go Kart & Kart Racer Parts & Accessories

****

****

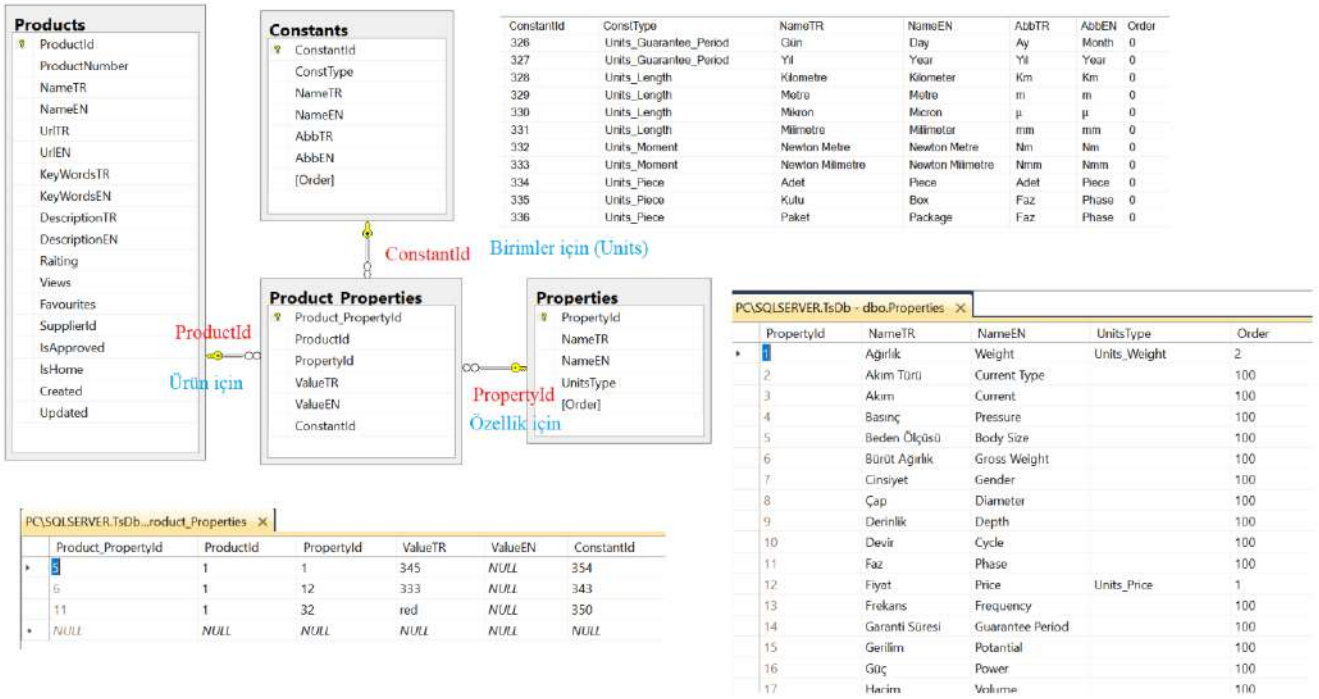
****

İNCELE : https://www.ilo.org/global/industries-and-sectors/lang--en/index.htm



SEKTÖR : https://www.globalpiyasa.com/tr/GTKategoriler.aspx

Properties Tablosunun Bağlantıları



ConstantId	ConstType	NameTR	NameEN	AbbTR	AbbEN	Order
326	Units_Guarantee_Period	Gün	Day	Ay	Month	0
327	Units_Guarantee_Period	Yıl	Year	Yıl	Year	0
328	Units_Length	Kilometre	Kilometer	Km	Km	0
329	Units_Length	Metre	Metre	m	m	0
330	Units_Length	Mikron	Micron	µ	µ	0
331	Units_Length	Milimetre	Milimeter	mm	mm	0
332	Units_Moment	Newton Metre	Newton Metre	Nm	Nm	0
333	Units_Moment	Newton Milimetre	Newton Milimetre	Nmm	Nmm	0
334	Units_Piece	Adet	Piece	Adet	Piece	0
335	Units_Piece	Kutu	Box	Faz	Phase	0
336	Units_Piece	Paket	Package	Faz	Phase	0

PC\SQLSERVER.TsDb...roduct_Properties

Product_PropertyId	ProductId	PropertyId	ValueTR	ValueEN	ConstantId
5	1	1	345	NULL	354
6	1	12	333	NULL	343
11	1	32	red	NULL	350
NULL	NULL	NULL	NULL	NULL	NULL

PC\SQLSERVER.TsDb - dbo.Properties

PropertyId	NameTR	NameEN	UnitsType	Order
1	Ağırlık	Weight	Units_Weight	2
2	Akım Türü	Current Type		100
3	Akım	Current		100
4	Basınç	Pressure		100
5	Beden Ölçüsü	Body Size		100
6	Brüt Ağırlık	Gross Weight		100
7	Cinsiyet	Gender		100
8	Çap	Diameter		100
9	Derinlik	Depth		100
10	Devir	Cycle		100
11	Faz	Phase		100
12	Fiyat	Price	Units_Price	1
13	Frekans	Frequency		100
14	Garanti Süresi	Guarantee Period		100
15	Gerilim	Potantial		100
16	Güç	Power		100
17	Hacim	Volume		100

Kurs Kaynağı: Udemy-Komple Uygulamalı Web Geliştirme Kursu-Sadık Turan