ASP.NET CORE-MVC

SİTE DETAYLARINI GELİŞTİRME

İçindekiler

SİTE DETAYLARINI GELİŞTİRME 1

Kategory Yapısının Oluşturulması 1

Örnek Kod: Div lerle tablo yapısı oluşturma 4

Örnek Kod: Acordion açılır Kategori oluşturma 4

NUGET Paketlerini Kurma 5

SDK yükleme 5

Nuget Paketlerini Yüklemek İçin 6

Access dosya bağlantısında “System.InvalidOperationException: ''Microsoft.ACE.OLEDB.12.0' sağlayıcısı yerel makine kayıtlı değil.' ”hatasının çözümü 7

SSL -Sertifkası ve Güvenli Bağlantı 7

Migration ve Veritabanı Sıfırdan Oluşturma 8

Not Defterindeki özet anlatım 11

C# Programlama Dilinde Sınıf(Class) Oluşturma 11

Constructor (Yapıcı) Metotlar 13

EF Core Migration İşlemleri 14

Identity (Kullanıcı Yetkilendirme) Framework unun Detaylandırılması 14

ÖZEL ATTRIBUTE OLUŞTURMA 14

C# Property Özellik Kullanımı (GET ve SET Nedir? Nasıl Kullanılır?) 16

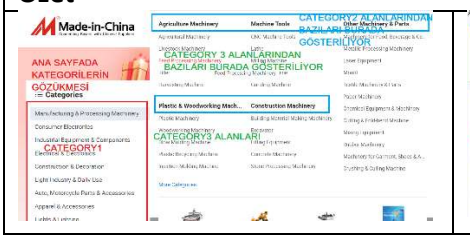
Asp.NET MVC İle Custom Error Page(Özel Hata Sayfasına Yönlendirme) 18

VERİTABANINA KLASİK OLARAK BAĞLANMA VE BİLGİLERİ GETİRME 21

Kategory Yapısının Oluşturulması

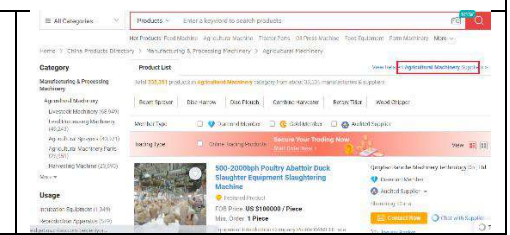
Kategori ve bunlara ait ürünleri gösterme 3 aşamalı bir sayfa düzeni ile sağlanacak.Bu hiyerarşik sayfaların adları “Ana sayfa”, “Categori Listeleme”, “Ürün Listeleme” sayfaları olarak adlandırılacak. Ana sayfada açılır menü ile 1,2 ve 3 seviye kategoriler gözükebilecek. Burada 1 seviye kategoriye tıklayınca Category listeleme sayfasına geçecek. 2 seviye kategoriye tıklama olmayacak. Sadece görüntüde gözükecek. 3 seviye kategoriye tıklanırsa o kategoriye ait ürün listeleme sayfa sistemine geçilecek.

Özet



Pop-up Ekranı

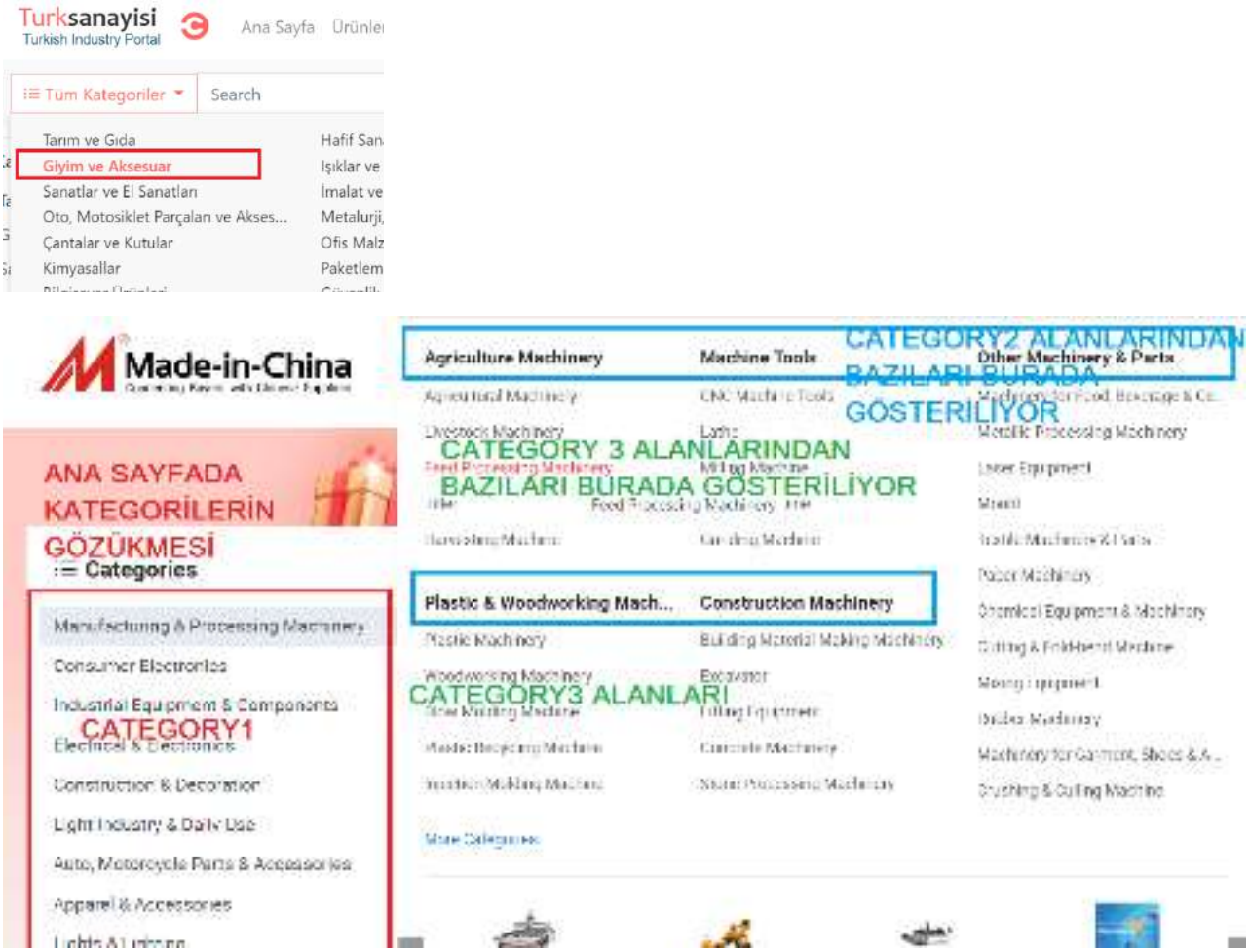
Kategori Listeleme sayfası



Ürün listeleme sayfası

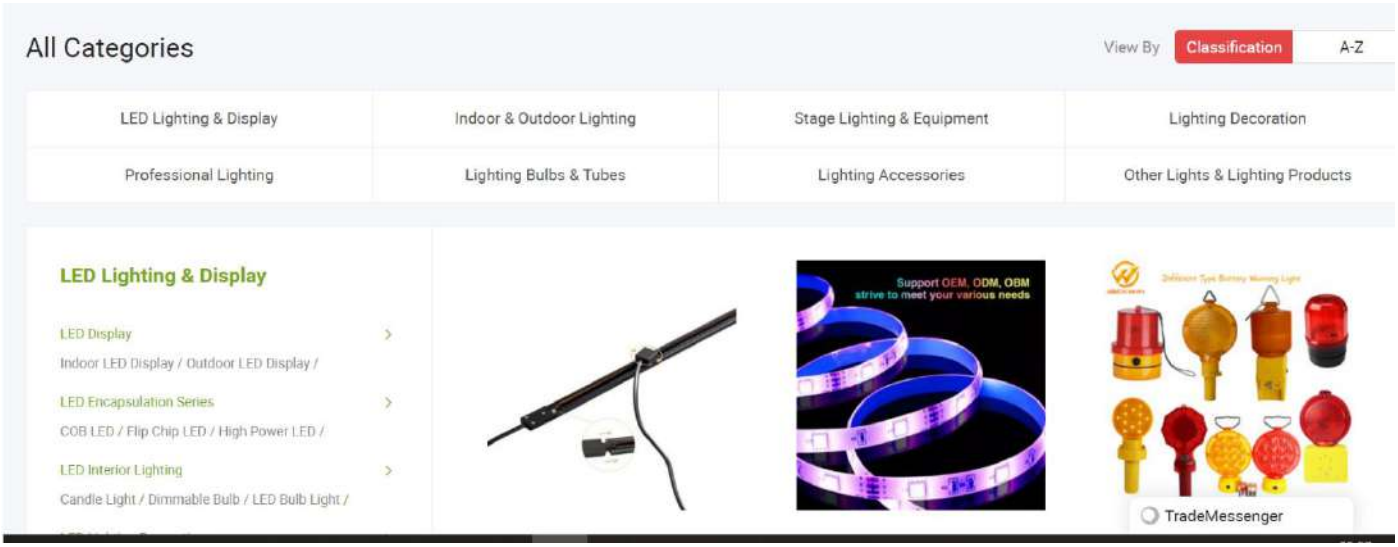
Ana sayfa-Kategori Pop-up sayfası (1,2 ve 3 kategoriler popup olarak gösterilecek)	Anasayfadan 1. Kategoriye tıklayınca burası açılacak. Bu sayfada üstte 2. Seviye kategoriler topluca gösterilecek. Aşağılarda da 2. Seviye kategoriler sayfa şeklinde gösterilecek. Her sayfada Sol taraftaki menüde üstte sabit başlık 2 seviye altta liste menüsü 3. Seviye, üzerine gelince açılan menü ise 4. Seviye olacak. 4 seviye tıklayınca yandaki Ürün Listeleme sayfası açılacak ve	bir önceki ekranda 4. Kategoriye tıklayınca açılacak. Bu sayfada üstte 5 kategoriler (anahtar kelimeler) gösterilecek. Sol tarafta üst kısımda 4. Kategoriye kadar olan üst kategorilerin hiyerarşisi gösterilecek, alt kısımlarda ise ürünlerin gruplandırılmış özellikleri olacak.

POP-UP sayfası: Ana sayfada soldaki C1 kategorilerini gösterirken tıklandığında yada üstte NavBar içinde tüm C1 kategoriler gösterilirken tıklandığında sonraki Kategori Listeleme sayfasına gidilecek.



KATEGORİ LİSTEME sayfası (CategoryList): Bu sayfa Pop-up dan 1 seviye kategoriye tıklayınca açılacak. Açılan sayfanın en üstünde 1. Seviye kategorinin adı gözükecek (Aşağıda bu yok). Daha sonra sayfanın genelinde 2-3-4 seviye kategoriler gözükecek. 2 seviyeler üstte tablo şeklinde listelenecek. Aşağılarda ise her 2 seviye kategori sayfalama şeklinde gözükecek. Sayfanın sol tarafında 3 ve 4 kategoriler menü şeklinde olacak. Sağda ise 2. Seviye kategoriyi temsil eden 4 yada 8 adet ürün listelenecek.

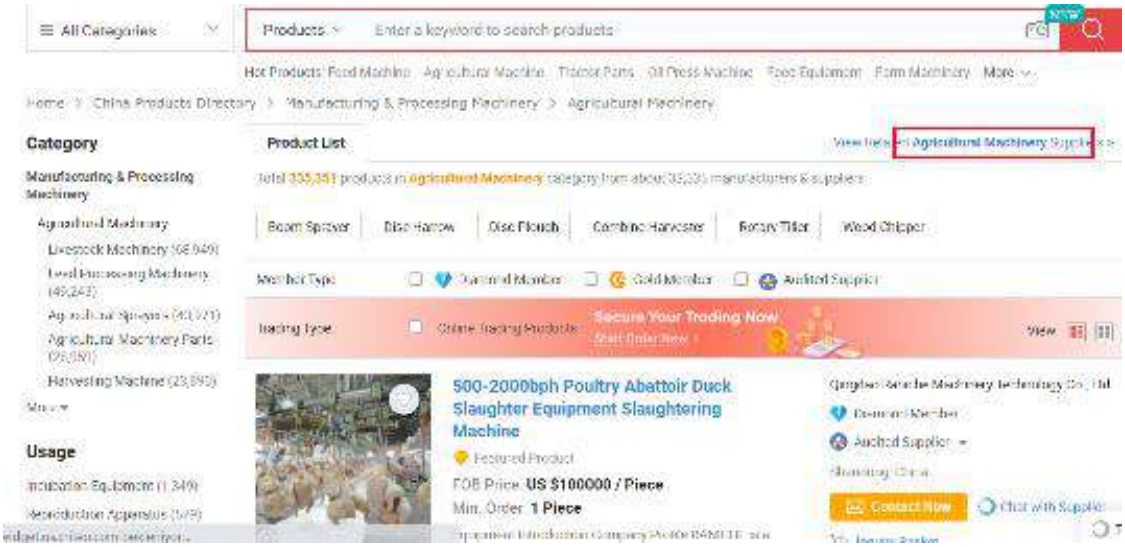

All Categories
2 SEVİYE KATEGORİLER
Agricultural & Food Machinery
3. SEVİYE KATEGORİLER
4. SEVİYE KATEGORİLER BURADA OLACAK


All Categories
View By
Classification
A-Z
LED Lighting & Display
Indoor & Outdoor Lighting
Stage Lighting & Equipment
Lighting Decoration
Professional Lighting
Lighting Bulbs & Tubes
Lighting Accessories
Other Lights & Lighting Products
LED Display
Indoor LED Display / Outdoor LED Display /
LED Encapsulation Series
COB LED / Flip Chip LED / High Power LED /
LED Interior Lighting
Candle Light / Dimmable Bulb / LED Bulb Light /
Support OEM, ODM, OBM
strive to meet your various needs
TradeMessenger


Stage Lighting & Equipment
Stage Equipment
Bubble Machine / Professional Audio /
Smoke Machine / Snow Machine /
Stage Lighting
Effect Light / LED Floor / Laser Light /
Moving Head Light / Wall Washer Light /
Led Stage
Led Head Light
Led Effect

ÜRÜN LİSTELEME sayfası (ProductList):

bir önceki ekranda 4. Kategoriye tıklayınca açılacak. Bu sayfada üstte 5 kategoriler (anahtar kelimeler) gösterilecek. Sol tarafta üst kısımda 4. Kategoriye kadar olan üst kategorilerin hiyerarşisi gösterilecek, alt kısımlarda ise ürünlerin gruplandırılmış özellikleri (Properties) olacak.



Örnek Kod: Div lerle tablo yapısı oluşturma

```
<div class="container">
@**** BU KODLARI GPT YAZDI **** divlerle tablo görünümünü elde etmek için önemli bir kod*@
@{
    var count = 0;
    var itemsPerRow = 4;
}

@foreach (var category2 in Model.Categories2)
{
    //-------her satır başladığında row ekliyor
    if (count % itemsPerRow == 0)
    {
        @:<div class="row border p-2">
        }
        //--------döngü her döndüğünde burayı ekliyor. col ekliyor.
        <div class="col border p-2">
            @category2.NameTR
        </div>
        //------ her satır sonunu gördüğünde yada tüm item sonunu gördüğünde /div ekliyor.
        if (count % itemsPerRow == itemsPerRow - 1 || count == Model.Categories2.Count - 1)
        {
        @:</div>
    }

    count++;
}
@* GPT kod sonu *@
    </div>
```

Örnek Kod: Acordion açılır Kategori oluşturma

```
<ul class="list-group">
    @foreach (var category3 in category2.Category3s)
    {
        <li class="list-group-item">
            <a class="d-flex align-items-center" data-toggle="collapse" href="#category3-
@category3.Category3Id">
                <span class="mr-auto">@GetCategoryName(category3.NameTR, category3.NameEN)</span>
                <i class="fas fa-chevron-down"></i>
            </a>
            <div class="collapse" id="category3-@category3.Category3Id">
                <div class="pl-3">
                    <ul class="list-unstyled">
                        @foreach (var category4 in category3.Category4s)
                        {
                            <li>
                                <a class="d-flex align-items-center" href="@(category4.Url)">
                                    <span>@GetCategoryName(category4.NameTR,
category4.NameEN)</span>
                                </a>
                            </li>
                        }
                    </ul>
                </div>
            </div>
        </li>
    }
</ul>
```

NUGET Paketlerini Kurma

SDK yükleme

- Komut ekranını aç (cmd ifasini kullan). Herhangi bir yerde iken yaz.

dotnet --info (yüklü SDK ları gösterir. Yani NetX.X versiyonlarını gösterir. Net6.0 yada Net7.0 gibi)

```
C:\Users\pc1>dotnet --info
.NET SDK:
 Version:   7.0.304
 Commit:    7e794e2806
```

```
.NET SDKs installed:
  6.0.201 [C:\Program Files\dotnet\sdk]
  6.0.410 [C:\Program Files\dotnet\sdk]
  7.0.304 [C:\Program Files\dotnet\sdk]
```

- Aktif çalışan SDK nın hangisi olduğunu görmek için

dotnet --version (komutu kullanılır).

```
C:\Users\pc1>dotnet --version
7.0.304
```

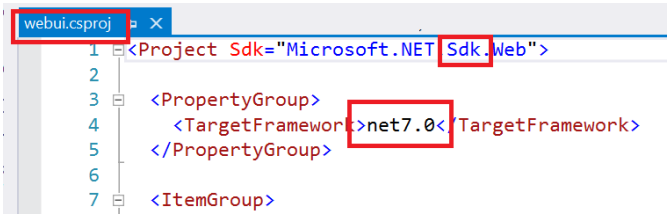
SDK yüklemenin internetten nasıl yapıldığı buraya yazılmamış..

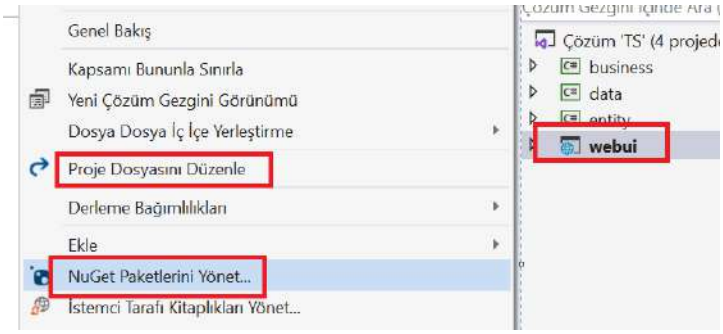
DİKKAT: En başta global.json içindeki Sdk versiyonu da değiştirilmeli. Bu Sdk internetten indirilip kurulmuş olması gerekir.

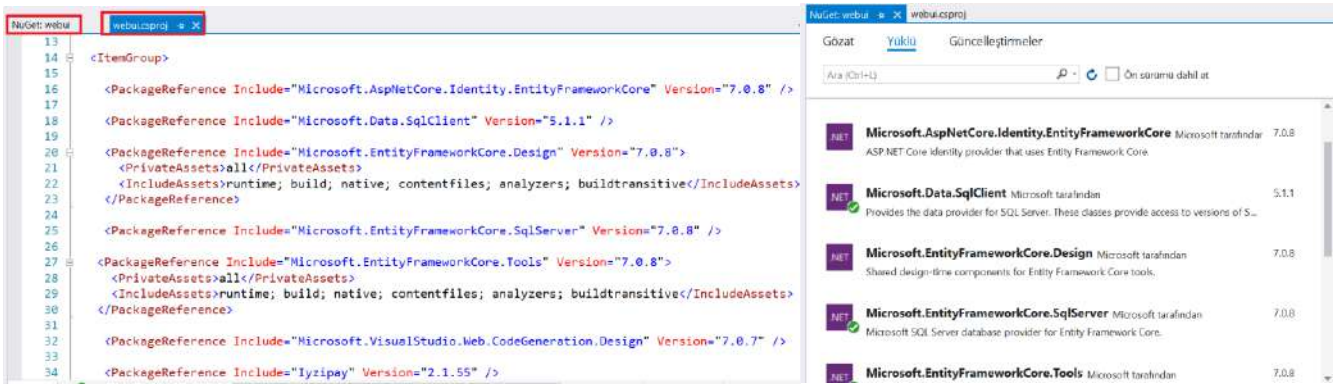
Nuget Paketlerini Yüklemek İçin

Nuget Paketlerini her bir proje için ayrı ayrı yükle. Projenin üzerine sağ tuşa tıkla. “Nuget Paket Yöneticisini” aç. Ardından “Proje Dosyasınını” da aç. Bu ikisi içindeki yüklü paketler aynı versiyon olmalı. Bazın paket yükleyince Proje dosyası içindeki versiyonu değiştirmezse elle değiştir.

Dikkat Burada 7.0 yükseltirken önce SDK nın versiyonu değiştirilmelidir. Tabii bu SDK larında öncesinde internetten indirilip bilgisayarda kurulu olması gerekir. Bu işlemler yukarıda anlatıldı.







DİKKAT: Önce Business projesi SDK ve Nuget paketleri güncellenmeli. Ardından Data projesinin kiler güncellenmeli. Diğer türlü birbirini görüyor ve diğerinin versiyonun dan dolayı sıkıntı çıkarıyor.

DİKKAT: Aşağıdaki gib Migration oluşturma esnasında bir hata alınmıştı. Şu komutu kullanınca güncelleme gerçekleşti. Visual studio dan yapamamıştım.

```
C:\Users\pc1\Desktop\TS.com\TS\data>dotnet ef migrations add InitialCreate --startup-project ../webUI --context ShopCont
ext
Build started...
Build succeeded.
The Entity Framework tools version '7.0.7' is older than that of the runtime '7.0.8'. Update the tools for the latest fe
atures and bug fixes. See https://aka.ms/AAc1fbw for more information.
Done. To undo this action, use 'ef migrations remove'

C:\Users\pc1\Desktop\TS.com\TS\dat>dotnet tool update --global dotnet-ef
'dotnet-ef' aracı, '7.0.7' sürümünden '7.0.8' sürümüne başarıyla güncelleştirildi.
```

Bu işlemden sonra Data projesinin proje dosyası içine şu satırı ekledi.

```
<PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="7.0.8">
```

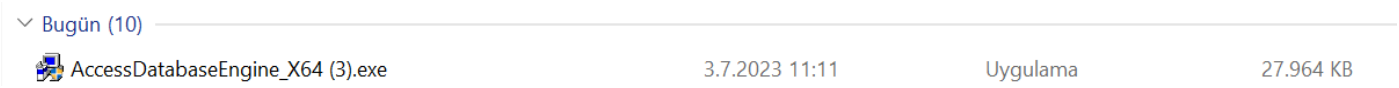
DİKKAT: Bir nuget paketini Dotnet komutları ile yüklemek için aşağıdaki gibi cmd ekranına projenin altında yazabiliriz.

```
dotnet add package Microsoft.EntityFrameworkCore.Tools
```

Access dosya bağlantısında “System.InvalidOperationException: ''Microsoft.ACE.OLEDB.12.0' sağlayıcısı yerel makine kayıtlı değil.' ”hatasının çözümü

Şu adresteki eklentiyi kurunca çalıştı

https://www.microsoft.com/en-us/download/details.aspx?id=13255



Ayrıca şu bilgiler faydalı olabilir. Bunları denedim olmamıştı.

https://www.muratoner.net/genel/c-microsoft-ace-oledb-12-0-provider-is-not-registered-on-the-local-machine-hatasi-cozumu

SSL -Sertifkası ve Güvenli Bağlantı

* cmd/mmc (Sql server configuration manager ekranını açar)

* cmd/certmgr.msc (certifikalar konsolu açar=direk certificate de yazılabilir)

* cmd/regedit (Kayıt defteri düzenleyicisi)

* cmd/services (hizmetler: sql serverları başlatma için kullan)

Çalışan DOTNET AYARLAMA KOMUTLARI-ZOR BULDUM SAKLA

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 ÇÖZDÜMmü emin değilim.. 2. MADDE

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:

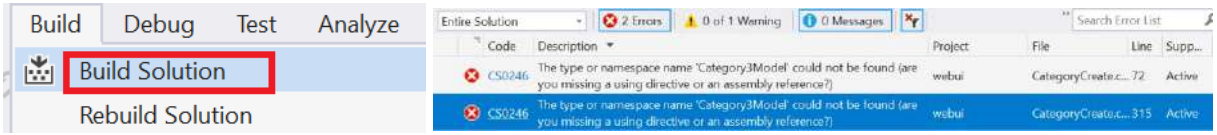
https://learn.microsoft.com/en-us/troubleshoot/sql/database-engine/connect/error-message-when-you-connect

Migration ve Veritabanı Sıfırdan Oluşturma

Daha önce çok kez sınıf yapılarını ve ilk temel bilgileri (Seed verilerini) veritabanına aktarmak için kullandığımız Migration dosyalarını oluşturalım. Ardından bu dosyaları güncelleyerek veritabanını oluşturalım.

1) Bilgisayarımızdaki SQLServer veritabanımız içindeki daha önceden oluşturduğumuz TsVT isimli veritabanımızı komple silelim.

2) Projemizde olabilecek hataları görmek ve exe dosyalarının oluşması için Build edelim. Eğer hatalar oluşmuşsa bunları düzeltelim.



3) Windos'un arama menüsünden "cmd" yazarak "Komut İstemi" siyah ekranını açalım.
4) Masaüstünde projemizin bulunduğu klasörlerden Data projesinin içerisine kadar "Cd …" şeklinde yazarak konumlanalım.
5) Buraya aşağıdaki gibi Migration oluşturma komutumuzu yazalım. Bu komut içinde başlangıç projesinin hangisi olacağını (WebUI) ve hangi Context i (ShopContext) kullanacağımızı da belirtmeliyiz.

```
C:\Users\pc1\Desktop\TS.com\TS\data>dotnet ef migrations add InitialCreate --startup-project ../webUI --context ShopContext
```

dotnet ef migrations add InitialCreate --startup-project ../webUI --context ShopContext

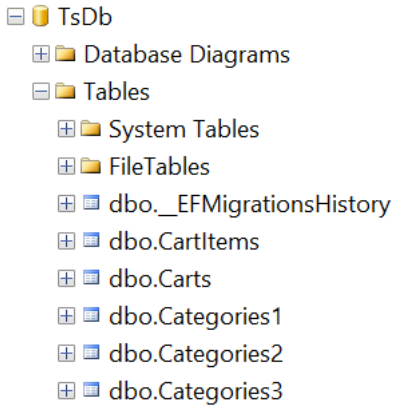
6) Migration oluşturma başarılı olduysa şu şekilde bir sonuç alırız.

7) Bu aşamada Veritabanını oluşturacak olan migration dosyalarımız oluştu fakat henüz daha SQLserver içinde veritabanı oluşmamıştır. Veritabanını oluşturmak içinde komut satırına aşağıdaki komutu yazalım.

```
C:\Users\pc1\Desktop\TS.com\TS\data>dotnet ef database update --startup-project ../webUI --context ShopContext
```

dotnet ef database update --startup-project ../webUI --context ShopContext

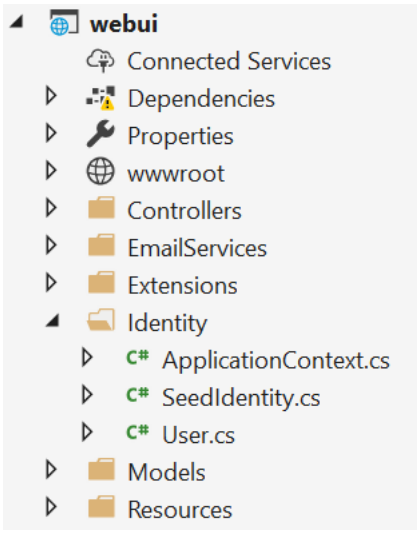
Veritabanımız oluştu fakat bunun içinde kullanıcı sınıfına ait tablolar yok. Bununla ilgili Context WebUI projesi içindeydi. Şimdi o Contexi kullanarak onunla ilgili Migrationları oluşturalım ve ona ait tablolarımızı oluşturalım.



8) Kullanıcı sınıfına ait (Identity) Migrationların ve VT tablolarının oluşturulması. Bu sınıf WebUI>Identity klasörü içindedir. Kullanılacak context in adı "ApplicationContext" dir. Bu context yapısını daha önceden projeye dahil edilen IdentityDbContext yapısından alır.


```
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

    public class ApplicationContext: IdentityDbContext<User>
```

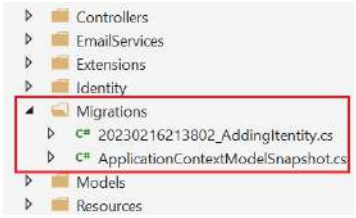
9) Bu contex e ait migrationları oluşturalım.




```
C:\Users\pc1\Desktop\TS.com\TS\webui>dotnet ef migrations add AddingItentity --context ApplicationContext
```

dotnet ef migrations add AddingItentity --context ApplicationContext

Bu işlemden sonra Migrationlarımız oluştu fakat bunlar VT içindeki tablolara henüz yansımadı.

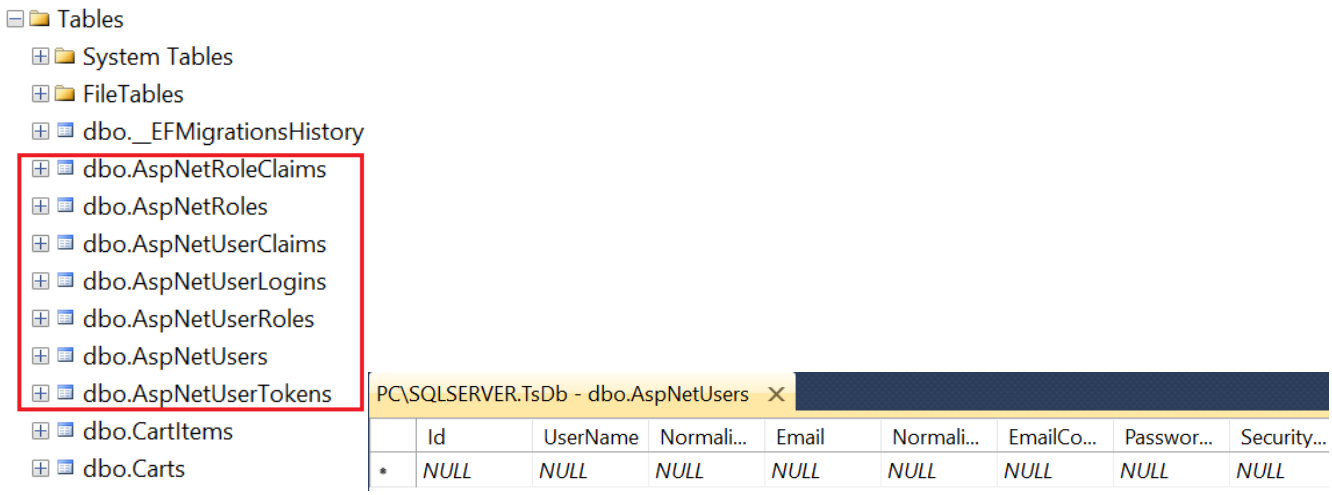


10) Identity (kullanıcıya ait) kütüphanenin tablolarının VT içinde oluşturulması. Bunun için aşağıdaki komutu çalıştırırsak ilgili tablolar oluşur ve içerisine ilk kullanıcı bilgileri aktarılır. İlk kullanıcı bilgileri AppSettings.json dosyası içinden alınır.

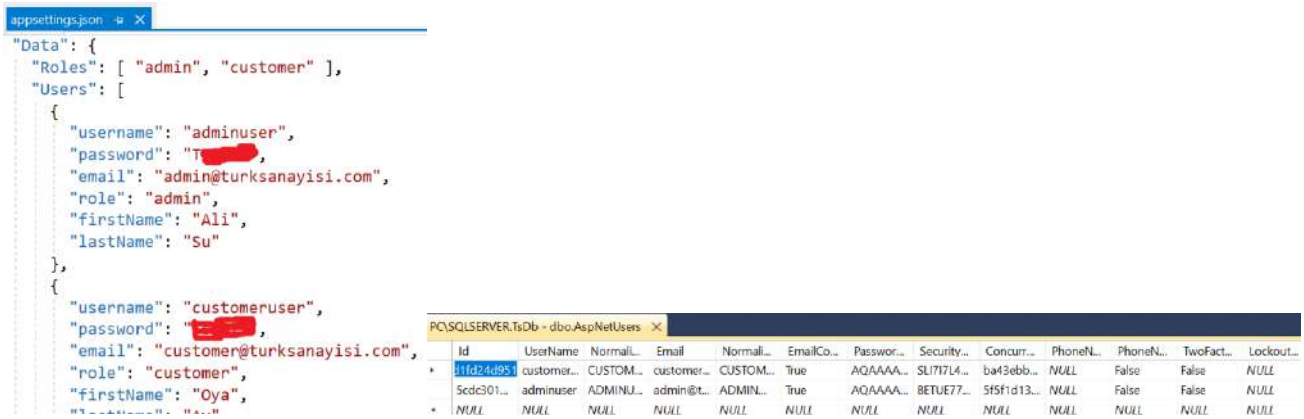
```
C:\Users\pc1\Desktop\TS.com\TS\webui>dotnet ef database update  --context ApplicationContext
```

dotnet ef database update --context ApplicationContext

Identity kütüphanesine ait tablolarımız oluştu fakat tabloların içi boştur.



11) Tabloların içine çekirdek (seed) başlangıç kullanıcı bilgilerinin AppSettings.json dosyası içinden alınıp aktarılması için programı çalıştırmamız gerekir.



Not Defterindeki özet anlatım

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

---

DİKKAT 1:ShopContext içindeki sondaki aşağıdaki 3 satırı AÇMAYI UNUTMA!... Sonrasında kapalı tut.
modelBuilder.Seed(); modelBuilder.SeedCategory(); modelBuilder.SeedLanguage();

DİKKAT 2: Startup.cs içinde en sondaki aşağıdaki satırı ilk Migration oluşturmada çalıştırmada aç. Sonrasında kapat.
SeedIdentity.Seed

DİKKAT 3: Ayrıca Program.cs nin başındaki şu satır Migration oluşturmalarda açık, yani sürekli açık
CreateHostBuilder(args).Build().MigrateDatabase().Run();

C# Programlama Dilinde Sınıf(Class) Oluşturma

Kaynak: https://www.oguzhantas.com/menu-c-dersleri

Microsoft'un geliştirdiği programlama dilleri arasında en popüleri C# olmuştur. C# programlama dilinin öne çıkmasının en önemli nedeni, daha önce geliştirilen Nesne Yönelimli (OOP:Object Oriented Programming) programlama dillerinin güzel özelliklerini bünyesinde barındırmasıdır. Özellikle yapı olarak C++ ve Java programlama dillerine çok benzeyen sınıf(class) yapısı sayesinde geniş kitlelerce hemen benimsenmiş ve popüler bir dil haline gelmiştir.

Nesne Yönelimli programlama dilleri kullanmanın en önemli avantajı tekrar kullanılabilirlik(reusability) dir. Başladığınız her projede aynı kodları defalarca yazıyorsanız ve sizin yazdığınız kodları yazılım ekibi içerisindeki diğer yazılımcılar geliştirip kullanamıyorsa, hem zaman hem maliyet açısından verimsiz bir durum söz konusudur. Nesne Yönelimli programlamanın diğer bir avantajı sürdürülebilirlik(maintainability) özelliğidir. Kaynak kodların kolayca okunabilmesi, anlaşılabilmesi gibi avantajları yanında kolayca hata bulma, optimize etme işlemleri de pratik bir şekilde yapılabilir. Genişleyebilirlik(Extensibility) de nesne yönelimli programlamanın diğer bir avantajıdır. Kodlarınıza kolayca yeni özellikler ekleyip genişletmenize olanak tanır.

Sınıfları şablonlar gibi düşünebiliriz, şablonları kullanarak nesnelerimizi oluşturuyoruz. Sınıflarımızın yaptıkları eylemlere metodlar denir. Sınıf kullanmanın en büyük artısı bir nesneyi özellik ve metotlarıyla beraber tanımlayabilmektir.

Sınıf oluştururken C# programlama dilinde aşağıdaki söz dizimini kullanıyoruz.

```
Erişim_Belirteci class Sınıf_Adı
{
}
```

Örneğin;

```
public class Insan
{
    //özellik ve metotlar
}
```

Burada Erişim_Belirteci public, private, protected gibi ifadeler olabilir. Kısaca açıklayacak olursak public kelime olarak "genel, halka açık" anlamına gelmektedir. public olarak tanımlamanan sınıflara diğer sınıflar içinden de ulaşılabilir. Anlam olarak private kelimesi "özel,gizli" anlamlarına gelir. private olarak tanımlanan metoda sadece o sınıf içinden ulaşılabilir, diğer sınıflar tarafından ulaşılamaz.

(protected ise daha sonra kalıtım bölümünde incelenecektir. ??? Bu kavramı burada açıkla)

Özellik(attribute) veya metotların önünde herhangi bir public, private, protected gibi ifade olmadığı zaman varsayılan olarak private'dır. Yani sadece o sınıf içinden ulaşılabilir.

Şimdi Visual Studio'yu açarak bir Console Application (Konsol Uygulaması) başlatalım.Şimdi bir Kutu sınıfı oluşturarak bu sınıfa ait genişlik, yükseklik ve uzunluk isimli üç özellik(attribute) tanımlayalım. Kutu sınıfımız için bir de HacimHesapla() metodu oluşturalım, bu metotun yapacağı eylem uzunluk, yükseklik ve genişlik değerlerini çarparak hacim değerini hesaplamasıdır.

```
using System;

namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            //Birinci Kutu
            Kutu kutu1 = new Kutu();
            kutu1.setGenislik(5);
            kutu1.setYukseklik(4);
            kutu1.setDerinlik(3);
            kutu1.hacimHesapla();

            //İkinci Kutu
            Kutu kutu2 = new Kutu();
            kutu2.setGenislik(10);
            kutu2.setYukseklik(2);
            kutu2.setDerinlik(4);
            kutu2.hacimHesapla();

            // Üçüncü Kutu3
            Kutu kutu3 = new Kutu();
            //Varsayılan değerleri yazacak
            kutu3.hacimHesapla();
            Console.ReadLine(); //Konsol ekranın durmasını sağlar
        }
    }

    class Kutu
    {
        double genislik;  //özellik 1
        double yukseklik; //özellik 2
        double derinlik;  //özellik 3

        //Constructor (Yapıcı) Metot, nesne oluştuğunda otomatik çalışan metot
        public Kutu()
        {
            Console.WriteLine("Kutu Oluşuyor");
            //aşağıda özelliklere varsayılan değerler atanıyor
            genislik = 6;
```

```
            yukseklik = 5;
            derinlik = 7;
        }

        public void setDerinlik(double d)
        {
            derinlik = d;  //derinlik özelliğimize atama yaptık
        }

        public void setGenislik(double g)
        {
            genislik = g; //genişlik özelliğimize atama yaptık
        }

        public void setYukseklik(double y)
        {
            yukseklik = y; //yükseklik özelliğimize atama yaptık
        }

        public void hacimHesapla()
        {
            double hacim = derinlik * yukseklik * genislik;
            Console.WriteLine("Hacim:{0}", hacim);
        }
    }
}
```

```
C:\Users\pc1\source\repos\ConsoleApp1\bin\Debug\netcoreapp3.1\ConsoleApp1.exe
Kutu Oluşuyor
Hacim:60
Kutu Oluşuyor
Hacim:80
Kutu Oluşuyor
Hacim:210
```

Burada Kutu sınıfımızla aynı isimde Kutu isimli bir metod görüyorsunuz, bu bir Yapıcı (Constructor) metotdur. Şimdi yapıcı metotları kısaca açıklayalım.

Constructor (Yapıcı) Metotlar

Sınıf ismiyle aynı olan metodlar yapıcı(constructor) metotlardır, nesne ilk oluşturulurken otomatik olarak çağrılırlar. Örneğin bir nesnenin özelliklerine varsayılan(default) değerler atamak istiyorsak kullanabiliriz. Yukarıdaki örnekte kutu3 nesnesi için genişlik, yükseklik, derinlik değerlerini atamadık. Yapıcı metot içinde tanımladığımız aşağıdaki varsayılan değerlere göre atama yapıpi kutu hacmini 6*5*7=210 olarak hesapladı.

Diğer Nesne Yönemli Makaleleri de incele

Sınıf Oluşturma

https://www.oguzhantas.com/csharp/329-c-programlama-dilinde-sinifclass-olusturma.html (Yukarıdaki döküman)

Abstract Metod

https://www.oguzhantas.com/csharp/415-c-ile-soyutabstract-metod-kullanimi.html

Threading İşlemi

https://www.oguzhantas.com/csharp/176-csharp-ile-threadingmuamele-islemleri.html

Metodlar

https://www.oguzhantas.com/csharp/112-csharp-programlama-dili-ile-metodlar-bolum-1.html

https://www.oguzhantas.com/csharp/113-csharp-programlama-dili-ile-metodlar-bolum-2.html

https://www.oguzhantas.com/csharp/118-csharp-programlama-dilinde-metodlar-bolum-3.html

https://www.oguzhantas.com/csharp/119-csharp-programlama-dili-ile-metodlar-bolum-4.html

Şifre Üretme

https://www.oguzhantas.com/csharp/79-yuksek-guvenlik-seviyesinde-sifre-uretme.html

EF Core Migration İşlemleri

Bu konuda aşağıdaki bilgileri incele, sınıf yapısını dene…

https://learn.microsoft.com/tr-tr/ef/ef6/modeling/code-first/migrations/?redirectedfrom=MSDN#customizing

Identity (Kullanıcı Yetkilendirme) Framework unun Detaylandırılması

Bu konunun detayları için aşağıdaki yazı dizisini takip et.

https://www.gencayyildiz.com/blog/asp-net-core-identity-yazi-dizisi/

Asp.NET Core Identity – Yazı Dizisi

- Asp.NET Core Identity – Nedir ve Temel Kavramlar? – I
- Asp.NET Core Identity – Identity Alt Yapısı Kurulumu – II
- Asp.NET Core Identity – Veritabanı Tablolarını İnceleyelim – III
- Asp.NET Core Identity – Kullanıcı ve Rol Modellerinde Custom Property Tanımlamak – IV
- Asp.NET Core Identity – UserManager Sınıfı İle Kullanıcı Yönetimi – V
- Asp.NET Core Identity – Şifre Validasyon Ayarları ve IPasswordValidator Arayüzü – VI
- Asp.NET Core Identity – Kullanıcı Validasyon Ayarları ve IUserValidator Arayüzü – VII
- Asp.NET Core Identity – Varsayılan Validasyon Mesajlarının IdentityErrorDescriber Sınıfı İle Özelleştirilmesi – VIII
- Asp.NET Core Identity – Cookie Bazlı Kimlik Doğrulama – IX
- Asp.NET Core Identity – Belli Sayıdaki Başarısız Girişlerde Kullanıcı Hesabını Kilitleme – X
- Asp.NET Core Identity – Şifremi Unuttum – XI
- Asp.NET Core Identity – Üye Bilgileri Güncelleme – XII
- Asp.NET Core Identity – Oturum Kapatma – XIII
- Asp.NET Core Identity – RoleManager Sınıfı İle Rol Yönetimi – XIV
- Asp.NET Core Identity – Sayfaları Yetkilendirme (Authorization) – XV
- Asp.NET Core Identity – Politika Bazlı Kimlik Doğrulama – XVI
- Asp.NET Core Identity – Claim Bazlı Kimlik Doğrulama – XVII
- Asp.NET Core Identity – Facebook Login – XVIII
- Asp.NET Core Identity – Google Login – XIX
- Asp.NET Core 3.1 ile Token Bazlı Kimlik Doğrulaması ve Refresh Token Kullanımı(JWT) *Not : Bu makale direkt olarak Identity mekanizmasıyla ilgili olmasada farklı bir kimlik doğrulama mekanizması konusunu kapsadığı için ilgili yazı dizisine dahil edilmiştir.*
- Asp.NET Core Identity – Two Factor Authentication Nedir?
- Asp.NET Core – Google & Microsoft Authenticator İle Two Factor Authentication
- Asp.NET Core Identity – Kullanıcının Cookie Değerini Yenileme

ÖZEL ATTRIBUTE OLUŞTURMA

```
using System;

namespace ConsoleApp2
{
    class Program
    {
        static void Main(string[] args)
```

```
        {
            AlisVeris alisVeris = new AlisVeris(150);
            AlisVeris alisVeris2 = new AlisVeris(10);
            Console.Read();
        }
    }

    //************************** PROPERTY İÇİN ÖZEL ATTRIBUTE KULLANIMI ************

    class AlisVeris
    {

        [Kontrol(Para = 100)] //BUNA ATTRIBUTE DENİR: kendi oluşturduğumuz Kontrol attribute içinde
Para propertisi var. Onu 100 olarak atıyor.
        public int classFiyat { get; set; } //BUNA PROPERTY DENİR: Bu property Alışveriş class ına
ait biz ekledik.

        public AlisVeris(int Fiyat)
        {
            this.classFiyat = Fiyat; //fonksiyon giren Fiyat bilgisi class seviyesinde tanımlanan
Fiyatın içine atılıyor. Buradaki this. ifadesi class seviyesindeki tanımlamayı gösterir.

            //AlisVeris sınıfının Property (özelliklerini) tarayacak.
            foreach (var property in typeof(AlisVeris).GetProperties())  //typeof ifadesi ile
Alisverisin ne tür olduğu bilgisini okuyor ondan sonra onun hakkında bilgileri getiriyor. Sonundaki
Properties ile onun özelliklerini okuyor.
            {
                //Alışveriş classına ait Property (özellik) lerin Attribute (öznitelik) lerini
getiriyor
                foreach (var attr in property.GetCustomAttributes(false))
                {
                    //Alışveriş class'ının Fiyat adındaki kendi oluşturduğumuz Propertisine gelince
işlem yapacak
                    if (property.Name == "classFiyat")
                    {
                        KontrolAttribute kontrolAttribute = (KontrolAttribute)attr;

                        if ((int)property.GetValue(this) > kontrolAttribute.Para)
                            Console.WriteLine("Alışveriş yapıldı.");
                        else
                            Console.WriteLine("Para eksik");
                    }
                }
            }
        }
    }

    [AttributeUsage(AttributeTargets.Property)]
    class KontrolAttribute : Attribute
    {
        public int Para { get; set; }
    }

    //************************** METOD İÇİN ÖZEL ATTRIBUTE KULLANIMI ************
    class OrnekClass
    {
        [Deneme()] //KontrolAttribute özniteliğini (attribute) çağırıyor
        public void OrnekMetod()
        {

        }
    }

    //Burada Kontrol isminde bir Attribute class oluşturuluyor. Bu class Attribute sınıfından
türetiliyor.
    //AttributeUsage ile bu attribute in hangi işlemlerde kullanılacağı belirlenmiş oluyor.
    //Örneğin burada metodlar için kullanılacağını AttributeTargets ile ifade etmiş oluyor.
    [AttributeUsage(AttributeTargets.Method)]
```

```
    class DenemeAttribute : Attribute
    {
        public DenemeAttribute()
        {
            Console.WriteLine("Deneme Attribute tetiklenmiştir.");
        }
    }
}
```

C# Property Özellik Kullanımı (GET ve SET Nedir? Nasıl Kullanılır?)

https://alkanfatih.com/c-property-ozellik-kullanimi-get-ve-set-nedir-nasil-kullanilir/#:~:text=%C3%96zellik%20metotlar%C4%B1%20GET%20ve%20SET,anahtar%20kelime%20ile%20de%20olu%C5%9Fturabiliriz.

C# da özellikleri, metotların ve sınıfların görünürlüklerini yönetmek için kullanırız. Kısaca örneklemek gerekirse bir sınıf içerisinden farklı bir sınıf içerisinde ki nesneye ulaşmak istiyorsak özellik metotlarını kullanmalıyız. Özellik metotları GET ve SET anahtar kelimesinden oluşan iki kod bloğundan oluşurlar. GET metodu veri okunduğu zaman, SET metodu ise veri yazıldığı zaman (yani değer ataması yapıldığı zaman) yürütülür. Özellik olarak bu iki anahtar kelimeyi aynı anda kullanabildiğimiz gibi, tek anahtar kelime ile de oluşturabiliriz. Örneğin sadece GET metodu ile oluşturduğumuz özellik sadece okunabilir, SET metodu ile oluşturduğumuz özellik ise sadece yazılabilir bir hal alır. Her iki anahtar kelimeyle oluşturduğumuz özellik ise hem okunabilir hem de yazılabilir özelliğe dönüşür.

Kısaca özetlemek gerekirse, Bir Class içerisinde bulunan bazı alanlara her zaman ulaşmak gerekmez. Çünkü bir nesneyi sürekli ulaşılabilir hale getirmek bilinçsiz kullanım, veri kaybı ve güvenliği gibi sorunları ortaya çıkartır. Zaten nesneleri tanımlarken varsayılan değer olarak “Private” erişim belirleyicisi olarak tanımlanması da bu tip gerekçelerden kaynaklanır. “Public” erişim belirleyicisi ise tamamen açık hale getirir. İşte tam bu noktada nesnelerimizin erişimini yönetmek için “Property” kavramı devreye girer. “Property” yani Özellik metotları nesnelerimiz üzerinde kontrollü kullanım sağlar.

Örnek Uygulama – 1

```
public class Personel
{
    public static int PersonelSayısı;
    private static int say;
    private string isim;

    // Okuma ve Yazma Özelliği
    public string Isım
    {
        get { return isim; }
        set { isim = value; }
    }

    // Sadece Okunabilir Özelliği
    public static int Say
    {
        get { return say; }
    }

    // Yapıcı metodumuz.
    public Personel()
    {
        // Personel Sayısını Hesapla
```

```
            say = ++PersonelSayısı;
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Personel.PersonelSayısı = 100;

            Personel prsnl = new Personel();
            prsnl.Isım = "Fatih Alkan";

            Console.WriteLine("Personel Sayısı: {0}", Personel.PersonelSayısı);
            //Çıktı: 101
            Console.WriteLine("Personel Ismi: {0}", prsnl.Isım);
            //Çıktı: Fatih Alkan
        }
    }
```

Örnek Uygulama – 2 – Set Metodu ile

```
class Urun
    {
        private string urunad;
        private string urunkod;
        private double urunfyt;
        //Değer alıp gönderen metodumuz.
        public string UrunAd
        {
            get { return urunad; }
            set { urunad = value; }
        }
        //100-999 arasında değer üretip kullanıcıdan alınan değerle birleştirip
        ürün kodu set ediyoruz
        public string UrunKod
        {
            get { return urunkod; }
            set
            {
                Random rnd = new Random();
                urunkod = value.ToString() + rnd.Next(100, 999).ToString();
            }
        }
        //Kullanıcıdan alınan veriyi yuvarlayıp set ediyoruz.
        public double UrunFyt
        {
            get { return urunfyt; }
            set
            {
                urunfyt = Math.Round(value, 1);
            }
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Urun urn = new Urun();
            urn.UrunAd = "Kursun Kalem";
```

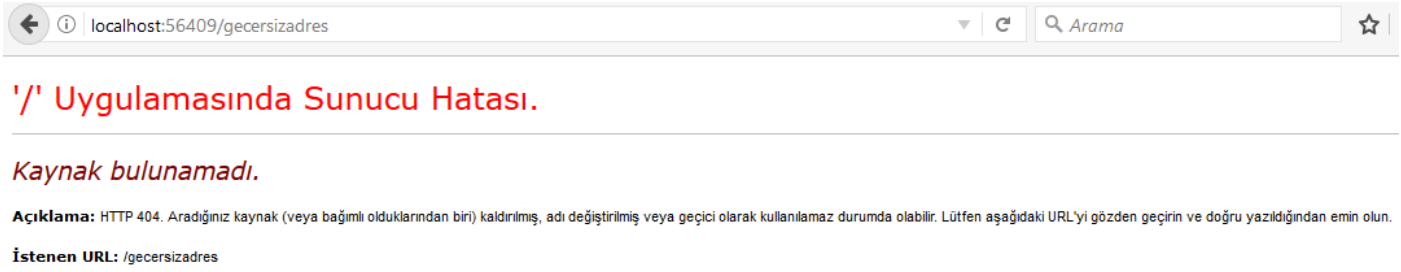
```
            urn.UrunKod = "KK";
            urn.UrunFyt = 10.786;

            Console.WriteLine("Ürün Adı: {0} - Ürün Kodu: {1} - Ürün Fiyatı: {2}",
urn.UrunAd, urn.UrunKod, urn.UrunFyt);
            //ÇIKTI: Ürün Adı: Kursun Kalem
            //ÇIKTI: Ürün Kodu: KK918
            //ÇIKTI: Ürün Fiyat: 10.8
        }
    }
```

Asp.NET MVC İle Custom Error Page(Özel Hata Sayfasına Yönlendirme)

https://www.gencayyildiz.com/blog/asp-net-mvc-ile-custom-error-pageozel-hata-sayfasina-yonlendirme/

Asp.NET MVC mimarisiyle inşa edilen projelerde beklenmedik olası hatalar alındığında yahut yapılandırılmamış, bozuk adreslere kullanıcılar tarafından talepte bulunulduğunda öncelikle yapmamız gereken sistem tarafından fırlatılan hatayı kullanıcıya göstermeksizin kullanıcıyı özel bir sayfaya yönlendirmektir. İşte bu içeriğimizde bu işlemin nasıl yapılacağını ele alacak ve dikkat edilmesi gereken noktaları vurgulayacağız.

Şimdi düşünün; üzerinde çalıştığımız projede kullanıcı tarafından geçersiz bir adrese talepte bulunulduğu vakit nasıl bir sonuçla karşılaşılmaktadır?



Görüldüğü üzere “404 hatası” dediğimiz hatayla karşılaşılmaktadır. Bu ve buna benzer birçok olası hata doğurabilecek durumlar için aşağıdaki talimatları izleyerek özel hata sayfanıza yönlendirmede bulunabilirsiniz.

Öncelikle yapmanız gereken, hatayı karşılayacak olan “Error(Controller).cs” isminde bir controller oluşturmanızdır.(*Controller’ın adını konuyla alakadar olmak şartıyla istediğiniz şekilde belirtebilirsiniz.*)

```
01public class ErrorController : Controller
02{
03    public ActionResult PageError()
04    {
05        Response.TrySkipIisCustomErrors = true;
06        return View();
07    }
08    public ActionResult Page404()
09    {
10        Response.StatusCode = 404;
11        Response.TrySkipIisCustomErrors = true;
12        return View("PageError");
13    }
14    public ActionResult Page403()
15    {
16        Response.StatusCode = 403;
          Response.TrySkipIisCustomErrors = true;
```

```
        return View("PageError");
    }
    public ActionResult Page500()
    {
        Response.StatusCode = 500;
        Response.TrySkipIisCustomErrors = true;
        return View("PageError");
    }
}

```

Yukarıdaki kod bloğunda görüldüğü gibi “Error(Controller).cs” içerisine aklıma gelen tüm hataları karşılamayı sağlayacak action metotları oluşturdum.(*Tabi ki de siz daha fazlasını oluşturabilirsiniz.*)

Tüm actionlar “PageError” sayfasına dönüş yapacağı için “PageError.cshtml” sayfasınıda aşağıdaki gibi oluşturuyorum.

```
<h2>Sayfa bulunamadı...</h2>
<div>
    Hata Kodu : @Response.StatusCode
</div>
<div>
    Hata Sebebi : @ViewBag.Kaynak
</div>
```

Bu işlemden sonra ana dizindeki Web.config dosyasını açınız ve <system.web>…</system.web> nodeları içerisine aşağıdaki kalıbı yerleştiriniz.

```

.
.
.
  <system.web>
    <customErrors mode="On">
      <error statusCode="404" redirect="/Error/Page404"/>
    </customErrors>
    <compilation debug="true" targetFramework="4.6.1"/>
    <httpRuntime targetFramework="4.6.1"/>
  </system.web>
.
.
.
```

Dikkat ederseniz burada “customErrors” nodeu ile özel hata kontrolünü devreye sokmuş bulunuyoruz. “error” nodu ile de “statusCode” u 404 olan hata için “/Error/Page404” sayfasına yönlendirmede bulunuyoruz.

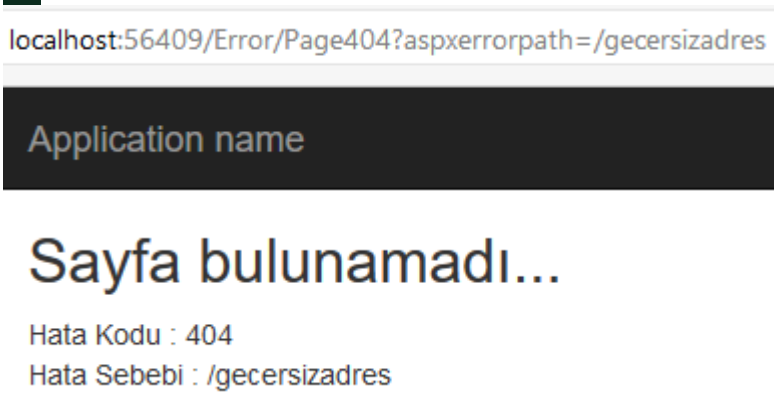
Bu işlemden sonra projemizi derleyip çalıştıralım ve bilinçli olarak 404 hatasını alarak süreci yeniden gözlemleyelim.

Videodan da gördüğünüz üzere olası bir hata durumunda Web.config’te yaptığımız düzenleme devreye girmekte ve verilen adrese ilgili kullanıcı yönlendirilmektedir.

Burada dikkatinizi bir hususa çekmek istiyorum. Videoda da görüldüğü üzere yönlendirme yapılırken “*http://localhost:****/Error/Page404?aspxerrorpath=/gecersizadres*” şeklinde karışık ve MVC’nin yapısına uymayan bir tarzda adres oluşturulmuştur. Adresteki “aspxerrorpath” parametresi hangi değerin 404 hatasına sebep olduğunu göstermektedir. Eğer ki bu adresi ve gelen “aspxerrorpath” parametresini yönetmek istiyorsanız aşağıdaki gibi bir çalışma yapmanız yeterli olacaktır.

```
    public class ErrorController : Controller
    {
.
```

```
.
.
        public ActionResult Page404(string aspxerrorpath)
        {
            if (!string.IsNullOrEmpty(aspxerrorpath))
                ViewBag.Kaynak = aspxerrorpath;
            Response.StatusCode = 404;
            Response.TrySkipIisCustomErrors = true;
            return View("PageError");
        }
.
.
.
    }
```



Velhasıl, oluşan olası hataları bu şekilde Web.config ile yönlendirebilmekteyiz. Ayriyetten olası hataları, “Global.asax” dosyasıylada yönlendirebiliriz. “Global.asax” dosyası, Asp.NET mimarisinde etkili bir hata yönetimi(Error Handling) sunmaktadır. Uygulamadaki tüm request(talep) ve response(cevap)ların yönetildiği ve bunlara ait eventların yakalandığı yer “Global.asax” dosyasıdır. İlgili dosya içerisinde “Application_Error()” metodu, uygulamanın herhangi bir yerinde herhangi bir kaynaktan dolayı meydana gelen hataları yakalamakla mükelleftir. Şimdi gelin birde olayı bu yapıyla ele alalım.

Öncelikle RouteConfig dosyasına aşağıdaki gibi bir route şeması ekleyelim.

```

    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
.
.
.
            routes.MapRoute(
              name: "Hata",
              url: "hata/{kod}",
              defaults: new { controller = "Error", action = "Page404", kod = UrlParameter.Optional
          );
.
.
.
        }
    }
```

Bu route şeması birazdan “Global.asax” dosyasından yapacağımız yönlendirmenin adres şeması olacaktır.

```
public class MvcApplication : System.Web.HttpApplication
{
```

```
    protected void Application_Start()
    {
        AreaRegistration.RegisterAllAreas();
        RouteConfig.RegisterRoutes(RouteTable.Routes);
    }

    protected void Application_Error(object sender, EventArgs e)
    {
        Exception ex = Server.GetLastError();
        if (ex is HttpException && ((HttpException)ex).GetHttpCode() == 404)
        {
            Response.Redirect("hata/404");
        }
    }
}
```

Evet… Görüldüğü üzere olası hata alındığı an Application_Error() metodu devreye girecek ve ilgili hatanın kodu denetlenip kullanıcı yukarıda oluşturduğumuz routea uygun adrese yönlendirilecektir.

VERİTABANINA KLASİK OLARAK BAĞLANMA VE BİLGİLERİ GETİRME

```
public class HomeController : Controller
{
    // GET: Home
    public IActionResult Index()
    {
        List<SelectListItem> items = PopulateFruits();
        return View(items);
    }

    [HttpPost]
    public IActionResult Index(string[] fruit)
    {
        ViewBag.Message = "Selected Items:\\n";
        List<SelectListItem> items = PopulateFruits();
        foreach (SelectListItem item in items)
        {
            if (fruit.Contains(item.Value))
            {
                item.Selected = true;
                ViewBag.Message += string.Format("{0}\\n", item.Text);
            }
        }

        return View(items);
    }

    private static List<SelectListItem> PopulateFruits()
    {
        string constr = @"Data Source=.\SQL2019;Initial Catalog=AjaxSamples;Integrated
Security=true";
        List<SelectListItem> items = new List<SelectListItem>();
        using (SqlConnection con = new SqlConnection(constr))
        {
            string query = " SELECT FruitName, FruitId FROM Fruits";
            using (SqlCommand cmd = new SqlCommand(query))
            {
                cmd.Connection = con;
                con.Open();
                using (SqlDataReader sdr = cmd.ExecuteReader())
                {
                    while (sdr.Read())
                    {
```

```
                        items.Add(new SelectListItem
                        {
                            Text = sdr["FruitName"].ToString(),
                            Value = sdr["FruitId"].ToString()
                        });
                    }
                }
                con.Close();
            }
        }
        return items;
    }
}
```

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