|  |  |
| --- | --- |
| EnTİTY fRAMEWOK  CORE | Berkay Akar |

İçindekiler

[ENTİTY FRAMEWORK VE ORM NEDİR ? 2](#_Toc98707995)

[KURULUM 4](#_Toc98707996)

[Context Sınıfının Oluşturulması 7](#_Toc98707997)

[Veritabanını Oluşturma Ve ilk Migration İşlemi 8](#_Toc98707998)

[VERİTABANI İŞLEMLERİ 9](#_Toc98707999)

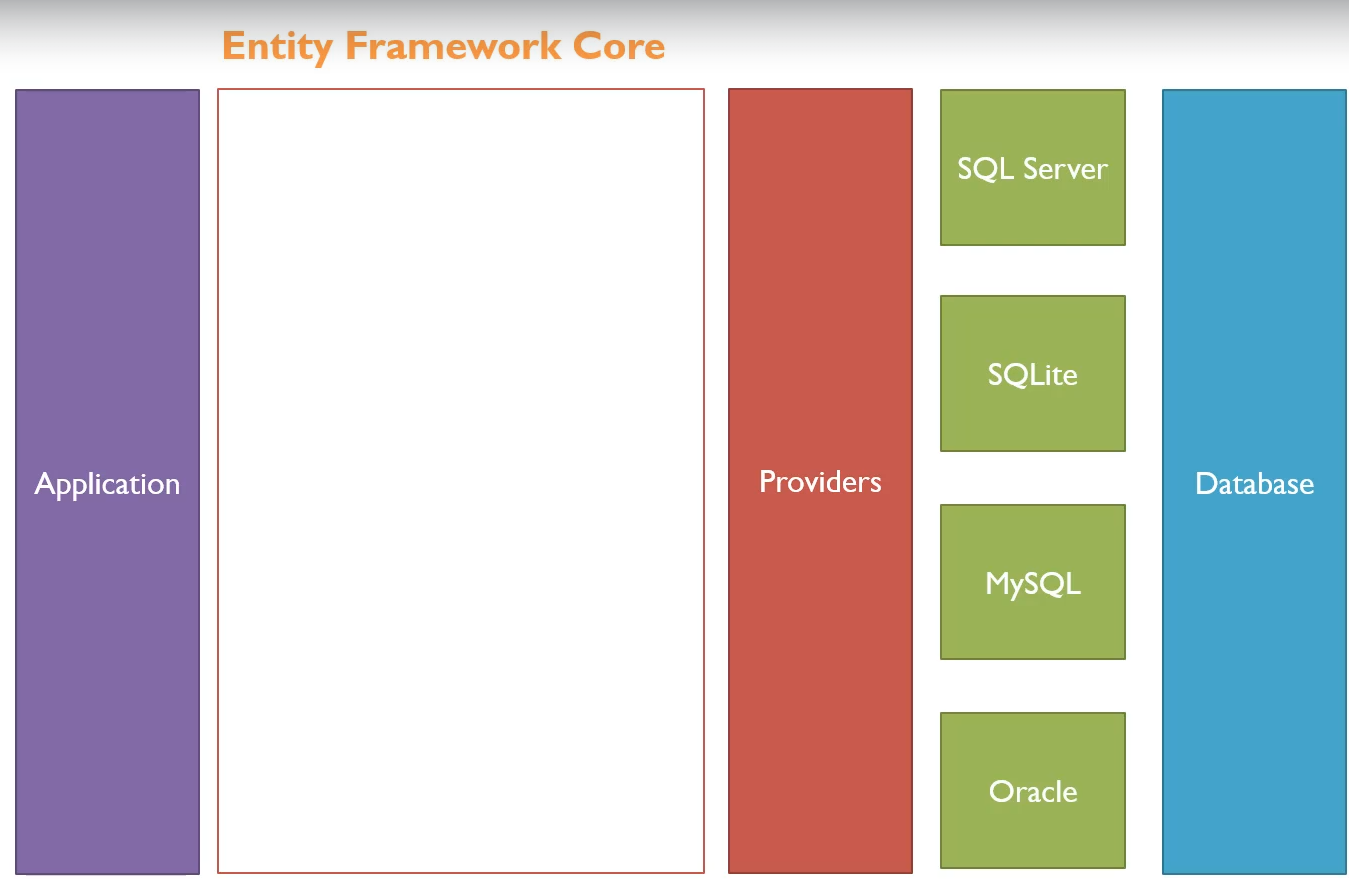
[Veritabanına veri ekleme : 9](#_Toc98708000)

[VERİTABANINDAN KAYIT GETİRME 11](#_Toc98708001)

# ENTİTY FRAMEWORK VE ORM NEDİR ?

Bir uygulamada veritabanına erişim kurarken veri tabanına erişim için bir Provider (driver) kurmamız gerekmektedir. ADO.NET teknolojilerinden de hatırlanabileceği gibi ilk olarak bir connection nesnesi, Beraberinde bir command nesnesi ve özellikleri verilerek bu işlem gerçekleştiriliyordu. Bu işlem için bir sql sorgusu yazmamız gerekmekteydi. Bağlantıyı isteğe bağlı olarak Connected ve disconnected olarak tanımladıktan sonra model dosyaları doldurularak uygulama içerisinde kullanılabilmekteydi. Bu işlem Mapping olarak adlandırılmaktadır. Mappingten kastımız sql içerisinden dönen verinin uygulamada kullanılabilir bir veri tipine göre eşleştirilerek kullanılması sürecidir.

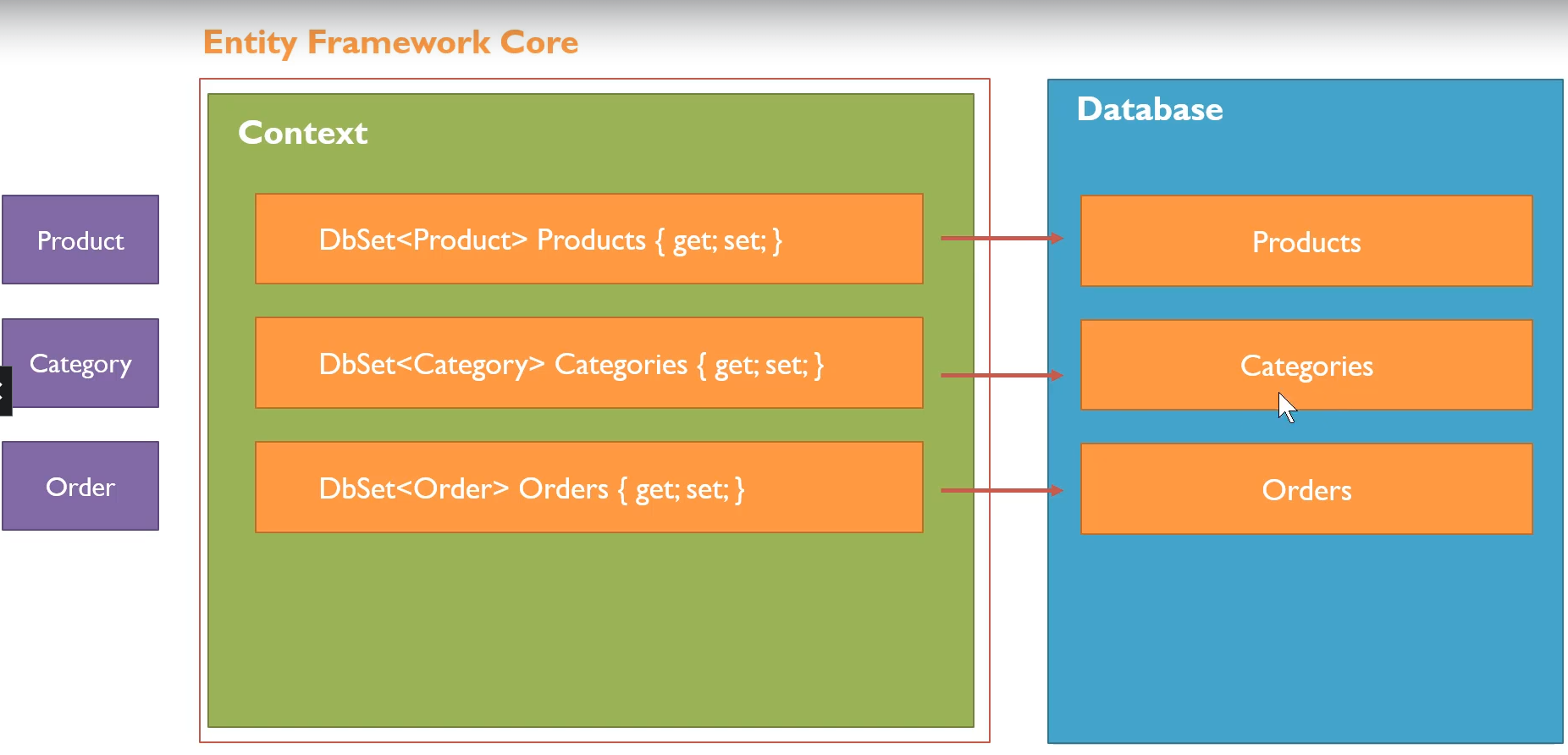
Bu süreci daha kolay olarak gerçekleştiren araçlar üretilmiştir. Bu araçlar Object Relaiton Mapping (**ORM**) olarak adlandırılmaktadır. ORM Birçok Veritabanı Yönetim Sistemini desteklemektedir.



Uygulama ile Entity Framewok ilerişime geçebilmesi için

1. Entity Framework Core
2. Hangi Veri Tabanı Yönetim Sistemi İle Çalışılacaksa onun bir Provider I

Yüklenmesi gerekmektedir.



# KURULUM

Yukarıda işleyiş bahsedildiği üzere hedef veri tabanı belirlenerek bu veritabanına uygun provider kurulumu gerçekleştirilmeldir. Microsoft sitesi veya Nuget Package Manager bu işi sağlayabilen servislerdir.

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure Providerlar üzerinden örnek

Örnek oluşturulabilmesi açısından işleyişe SQLİTE database I için gerekli provider tedariği ile başlıyoruz. ORM yapısındaki kodlama merkezi bir yönetim oluşturduğu için sadece provider değişiminde tüm db lere dönüşümü sağlanabilmektedir.

Bir console uygulaması açalım

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Uygulama 5.0 da oluşturulduğundan ötürü paketleri yüklerken uyumluluklarını göz önüne alarak indiriniz.

metin, ekran görüntüsü, ekran içeren bir resim

Açıklama otomatik olarak oluşturuldu

Paket kurulumunu lütfen ilgili projenin dizinine gerçekleştiriniz. Solution üzerine paket kurulması durumunda tüm proje boyunca paketler yüklenmiş olacak ve bu projenin hatalı üretim (deising patternlerde uyumsuzluk ve güvensizlik ), fazla yer kaplaması gibi çeşitli problemleri doğruacaktır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Core yazmayan (en üstteki) .NetFramework için gerekli pakettir.

metin, ekran görüntüsü, ekran, iç mekan içeren bir resim

Açıklama otomatik olarak oluşturuldu

Provider da budur.

Kurulacak bir diğer kütüphane ise db-first, migration komutlarının içinde yer aldıgı EntityFrameworkCore.Tools kütüphanesidir.

metin, ekran görüntüsü, ekran, siyah içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure Tools Kütüphanesi ve içeriğindeki kodlar

Bir sonraki aşamada Veritabanı tablolarını kodlamaya başlamak için bir adet Entities isimli bir klasör oluşturup içerisine tablolarımızı DataAnnodation komutları ile işlemeye başlayacağız.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Entitiler burada sınıflar şeklinde oluşturularak ve üzerinden migration denen bir yöntem ile veritabanı tablolarına dönüştürülecektir.

Tablo oluştururken dikkat edilmesi gereken özellik ise her tablonun bir primary key özelliğinin bulunması zorunluluğudur.

public int ID { get; set; }

public decimal Price { get; set; }

Alanları bu şekilde belirlenmiş olan bir sınıfta eğer alan isimlerinden birisi sadece Id veya ClassAdıId şeklinde tanımlanırsa hiçbir data annodation kullanmadan veritabanı bunu primary key olarak algılar. Fakat farklı bir isim koyarsak üzerine gelerek [Key] attributunu eklememiz gerekecektir.

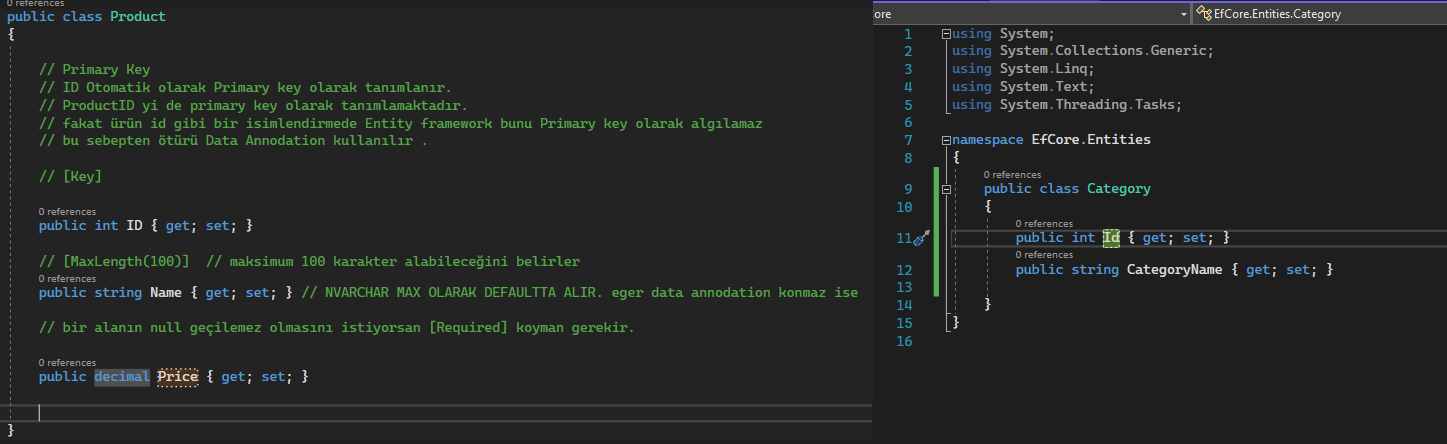


Figure Entitiler Bu şekilde tanımlandı

# Context Sınıfının Oluşturulması

Bu aşamada bir adet Context isimli sınıf açılır. Bu sınıfın bir Context olduğunun algılanması için DbContext sınıfından bir kalıtım verdirtilmesi gerekmektedir. Eklenen verilerin Çalıştırılabilmesi için tabloların tek tek Dbset<sınıf> formatında bir tanımlama ile birlikte bulunulması gerekmektedir. Bu property lere verilecek her isim Veritabanı kısmında tablo ismine karşılık gelineceğini unutmayalım. Beraberinde bir diğer husus veritabanını nerede oluşturulacağıdır. İsteğe bağlı olarak ConnectionString tagı ile proje üzerinde , Appconfigte veya DbContext ten kalıtım alan yerde OnConfigüring komutu ezdirilerek connection stringi tanınabilir. OptiunBuilder den sonra UseHangidbtipi() fonksiyonu ile (Provider olarak kurulan kütüphane hangisi ise onun methodu burada belirlenir. ) bu şekilde işe başlanır .

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

# Veritabanını Oluşturma Ve ilk Migration İşlemi

Veritabanı kısmında (Code kısımıda fiziksel veritabanı değil) her bir değişiklikte değişikliğin veritabanına işlenebilmesi için Migraton oluşturma ve bu migration ile veritabanına update ederek kullanmamız gerekmektedir.

Son aşamada 4 adet paket kurulu olması gerekmektedir.

* EntityFrameworkCore (ef için)
* EntityFrameworkCore.Sqlite (hedef provider için)
* EntityFramewok.Tools (update,migration gibi yapılar için )
* EntityframeworkCore.Deisinger (Dbset ve Context sınıfı için)

Bu aşamadan sonra yapılması gereken add-migration migrationismi , update-database ile veri tabanını update ederek kullanılabilir bi r db oluşturmaktadır.

# Veritabanı İşlemleri

## Veritabanına Veri Ekleme :

using (var context = new SQLİTECONTEXT())

{

var p = new Product { Name = isim , Price = fiyat };

context.Products.Add(p);

context.SaveChanges();

Console.WriteLine("İşlem başarılı");

ProductGetir();

}

SQLİTECONTEXT sınıfı içerisinde kalıtım yolu ile DbContext sınıfından Dispose method gelidiği için var değişkeni ile kullanılabilmektedir. Temel ekleme, güncelleme, silme ve görüntüleme işlemlerinde kendi Context sınıfımızdan üretilmiş bir context . bunun bir tabosu . method ismi ve parametresi verilmelidir. İşlem bittikten sonra ise context.saveChanges ile verilen işi veritabanına comitleyerek eşleyebilmekteyiz.

Bir context e birden fazla veri aynı anda atılmak istenirse listelerden de tanıgımız metho d olan addrange kullanılabilmektedir .

using (var context = new SQLİTECONTEXT())

{

List<Product> productlistesi = new List<Product>() {

new Product { Name = "samsung s6", Price = 2500 },

new Product { Name = "samsung s7", Price = 3500 },

new Product { Name = "samsung s8", Price = 4500 },

};

context.Products.AddRange(productlistesi);

context.SaveChanges();

Console.WriteLine("İşlem başarılı");

Yürütülen sorguların SQL karşılıklarını ve en son çalıştırılan sorgunun sql karşılığını entity frameworkte alabilmemiz mümkündür. Bu işlemin türkçesine loglama Entity Frameworkte ki kaşılığına logging denmektedir. Bu iş için Eklenmesi gereken bir kütüphanemiz mevcuttur.

metin, ekran görüntüsü, ekran, siyah içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bir sonraki aşamada ise DbContext sınıfımızdan kalıtım almış olan SQLİTECONTEXT sınıfı içerisinde eklenmesi gereken bir ILogger ekleyelim .

public static readonly ILoggerFactory loggerFactory = LoggerFactory.Create(builder => { builder.AddConsole(); });

Override ettiğimiz onconfiguring kısmındaki Provider ın önüne bir değişken daha eklenerek sonuç bitirilir.

Son hali aşağıdaki şekilde verilmiştir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu işlemden sonra yapılabilecek her türlü insert update delete işleminde bize hedef makinede nasıl bir sql çalıştırdığını gösterecektir. Soldaki kodun çıktısı sağ kısımda verilmiştir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

## Veritabanindan Kayıt Getirme

Bir context Referansı oluşturularak bu context üzerinden tablomuzu vermeil ve üzerinden bir LİNQ çalıştırarak tablolarımıza erişebilmekteyiz. Kısacası :

using (var context = new SQLİTECONTEXT())

{

var tablo = context.Products.ToList();

Console.WriteLine("ÜRÜNLER TABLOSU :");

foreach (var item in tablo)

{

Console.WriteLine("ID : "+item.ID + " İSİM :"+item.Name+" FİYAT : "+item.Price);

}

}

Kodun çıktısı Aşağıda verilmiştir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu