SUNAN: ÖZGE AKDAŞ

ENTİTY FRAMEWORK

NELER KONUŞACAĞIZ

SUNUMDA ÖNE ÇIKANLAR

- **Entity framework nedir**
- **Entity framework neden kullanılır**
- **Avantaj ve dezavantajları**
- **Katmanlı mimaride kurulumu**

ENTİTY FRAMEWORK NEDİR?

Entity Framework ORM(Object Relational Mapping) araçlarından biridir.

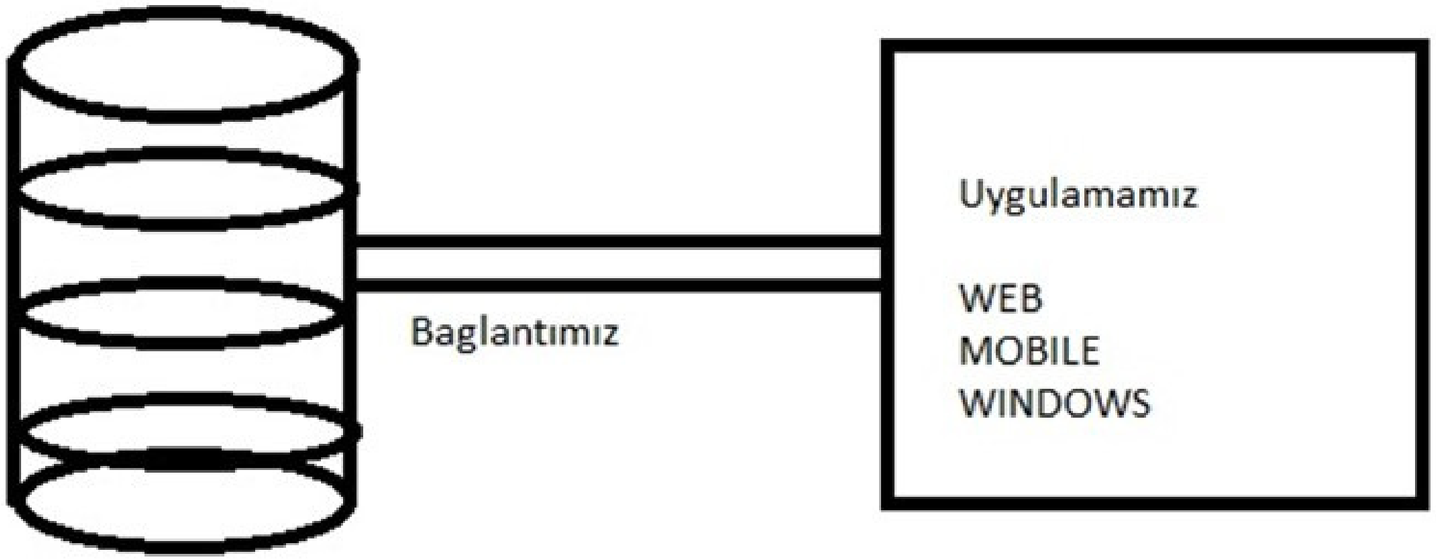
- Yazılım geliştirenler için database üzerinde yer alan verilere erişim sağlama, verileri depolama gibi işlemleri yapabilmemizi sağlayan bir ADO.NET mekanizmasıdır.
- Query yazmak için LINQ kullanılır.

NASIL ÇALIŞIR?

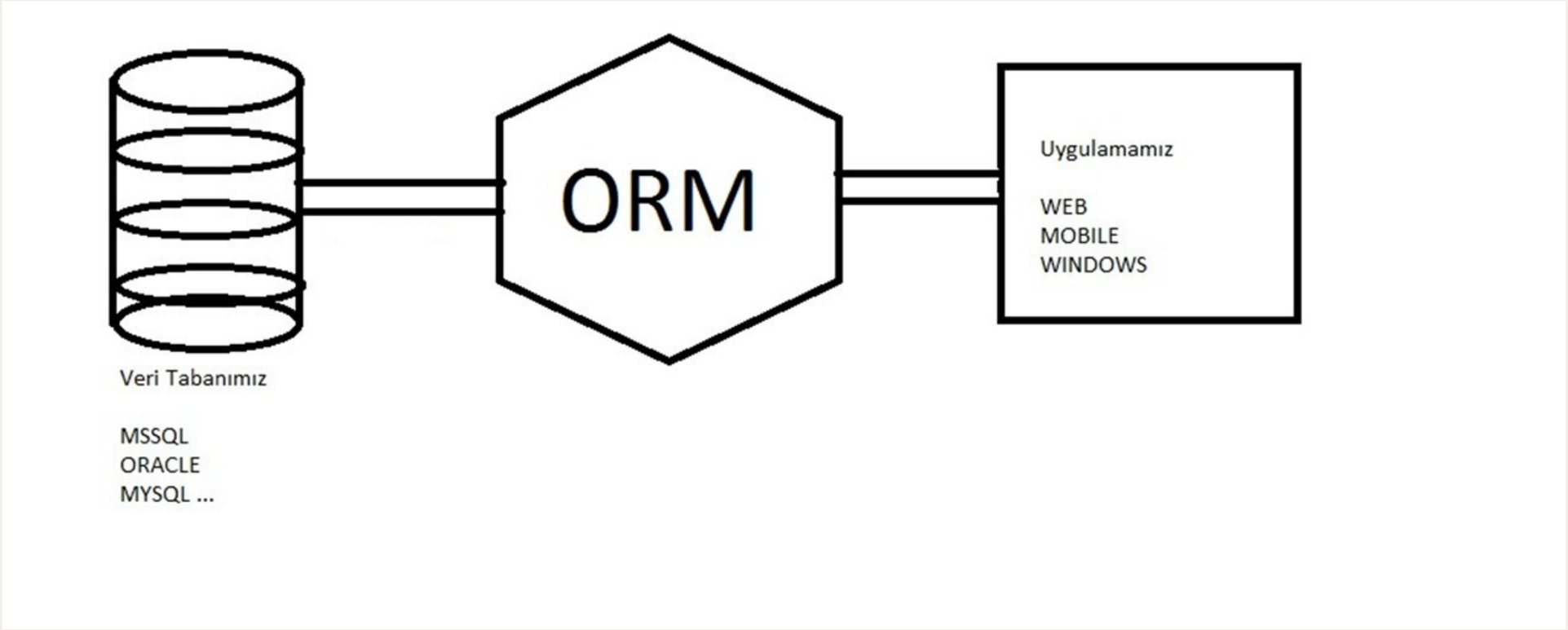
Entity Framework .NET içerisinde yer alan List, Directory gibi LINQ temelli çalışır.
Veritabanı işlemlerinin yapıldığı sınıf da yer alan özelliklere değerler Add, AddRange gibi metotlarla eklenir.
İşlemin veritabanına yansıması için DbContext sınıfında yer alan SaveChanges metodunun çalıştırılması gerekir.

ORM NEDİR DERSEK:

İlişkisel veritabanı ile nesneye yönelik programlama(OOP) arasında bir köprü görevi gören araçtır. Bu köprü, ilişkisel veritabanındaki bilgilerimizi yönetmek için nesne modellerimizi kullandığımız bir yapıdır.


Uygulamamız
WEB
MOBILE
WINDOWS
Baglantımız
Veri Tabanımız
MSSQL
ORACLE
MYSQL ...

ORM yani Object Relational Mapping, veri tabanınızdaki tabloları Class’lara, kolonları Property’lere, tabloların içindeki kayıtları da Object’lere dönüştüren ve tüm bu dönüşün sonucunda oluşan Class’lar ve objeler üzerinden veri tabanı işlemlerinizi yapmayı sağlayan bir teknolojidir. Yani kısaca tanımlamak gerekirse Database modelini uygulama modelindeki nesnelere uyarlar.



Kısaca veritabanına bizim nesnelerimizi bağlayan ve bizim için veri alışverişini yapan Microsoft tarafından geliştirilmiş bir framework’tür.

NEDEN KULLANILIR?

- Karmaşık veritabanı sorguları geliştirmeyi daha da zor hale getirir.

  ORM araçlarının temel kullanım nedeni bu ve bunun gibi sorunları ortadan kaldırmaktır.

- Entity Framework gibi ORM araçları farklı veri sağlayıcıları (SQL Server, MySQL, SQLite gibi) için aynı komutları kullanarak işlem yapmaya imkan verir

- Veritabanına dayalı modeller oluşturabilir veya tam tersi yapılabilir.
- Tekrarlanan işlemlerden kurtuluruz.
- Daha güvenlidir.
- Çapraz platformlar için uygundur.
- Açık kaynaklıdır.
- Veri tabanı ile ilişkilerin manuel yürütülmesine gerek yoktur.

Entity Framework ile 4 farklı yöntem kullanarak proje geliştirebilirsiniz. Bu yöntemler;

- Model First (Yeni Veritabanı Oluşturma Visual Studio İle)
- Database First (Var Olan Veritabanını Kullanma)
- Code First (Yeni Veritabanı Kod Yazarak)
- Code First(Var Olan Veritabanını Kullanma)

MODEL FIRST

Bu yöntem Visual Studio üzerinde boş bir model dosyası (.edmx) eklenerek veritabanını bu model üzerinde tasarlanabilmesine olanak sağlıyor. Derleme adımında verilen script dosyası veritabanını oluşturur.

Bu yaklaşım, sıfırdan bir veritabanı oluşturarak projeye başlanması gereken durumlar için ideal olabilmektedir. Kod yazmadan ya da serverda fiziksel database ile ilgilenmeden, tamamen entity model üzerinden veritabanı oluşturmak için kullanılan yaklaşımdır. Entity user interfacesi üzerinden modellediğiniz verilerinizi generate ettiğinizde hem database tarafında hem de kod tarafında otomatik olarak oluşurlar.

DATABASE FIRST

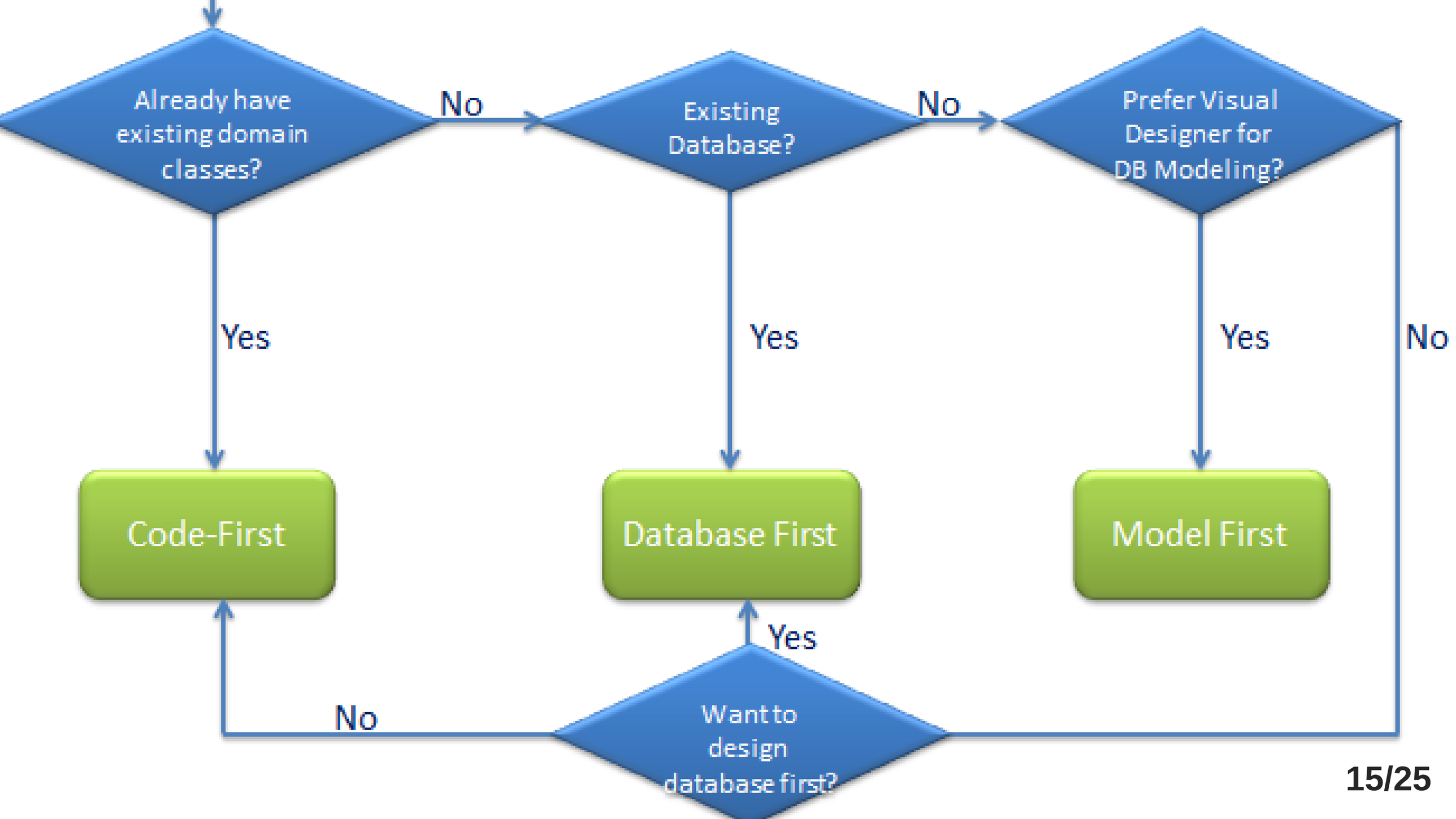
Bu yöntem önceden oluşturulmuş olan veritabanını projeye model olarak bağlayarak gerekli classlarımız Entity Framework tarafından oluşturulmaktadır.

CODE FIRST(KOD YAZARAK)

Bu yöntem classlarımızı visual studio ortamında oluşturmaya başlayarak gerçekleştirdiğimiz bir yöntemdir. Veritabanımız bu classlardan türetilmektedir.Burada Mapping işlemleri yazılımcı tarafından classlar oluşturulurken Attribute'lar sayesinde yapılabilmektedir. Bu arada Mapping işlemi kısaca tablolarımızdaki kısıtlarımızı belirlediğimiz olaydır. Attribute'ların yanında farklı yöntemlerlede bu işlemleri gerçekleştirebilmekteyiz. Örnek vermek gerekirse Fluent Api veya Fluent Validation gibi araçlar Mapping işlemleri için popüler olarak kullanılmaktadır.

CODE FIRST(VAR OLAN VERİTABANINI KULLANMA)

Bu yöntemde de classlar ve mapping kodları yazılımcı tarafından oluşturulmaktadır. Veritabanımız classlarımızın ve modellemenin durumuna göre güncellenmektedir.


Already have existing domain classes?
No
Existing Database?
No
Prefer Visual Designer for DB Modeling?
Yes
Yes
Yes
No
Code-First
Database First
Model First
Yes
No
Want to design database first?

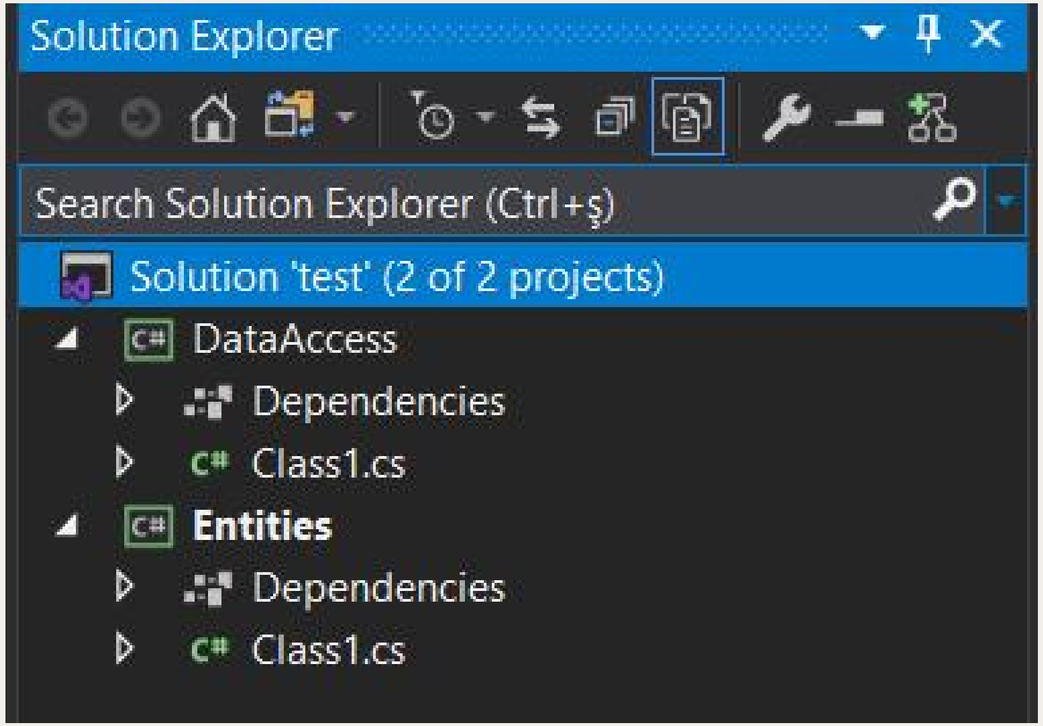
AVANTAJLARI

- OOP olarak kod geliştirmemize olanak tanır.
- Hiçbir SQL bilgisi olmayan bir kimse veritabanı işlemlerini **EF** ile gerçekleştirebilir.
- Herhangi bir veritabanına bağımlılık yoktur.
- Projelerinize büyük bir esneklik sağlamaktadir.
- Yazılım geliştirme zamanını azaltır.
- Yazılım geliştirme maaliyetini azaltır.

DEZAVANTAJLARI

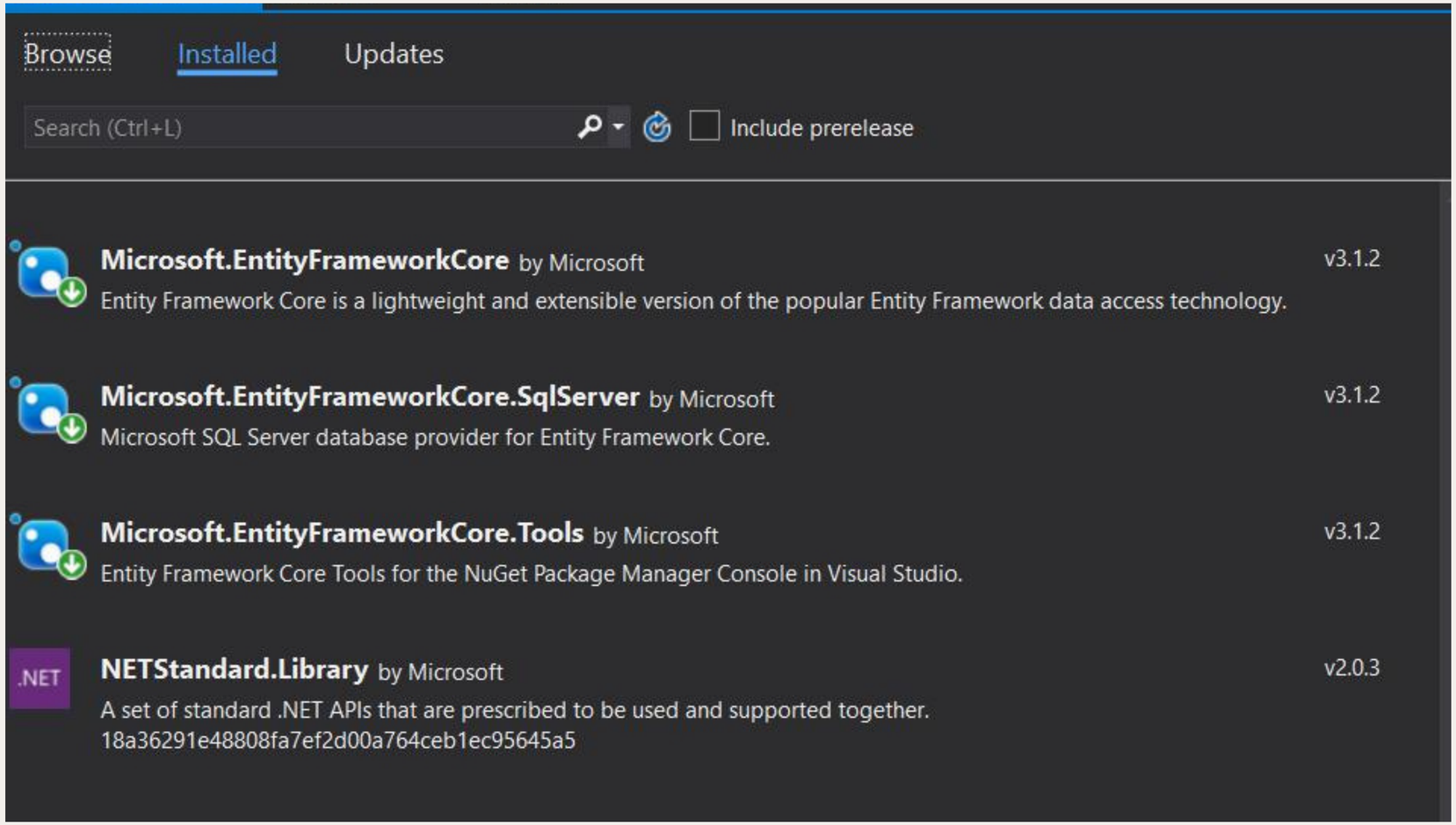
- En büyük sorunumuz performans.
- Veritabanından veri alış-verişi yapılacağı zaman kontrol bizde değil Entity Frameworktedir.
- Syntax'ı yeni kullanacak kişiler için karmaşık gelebilmektedir.

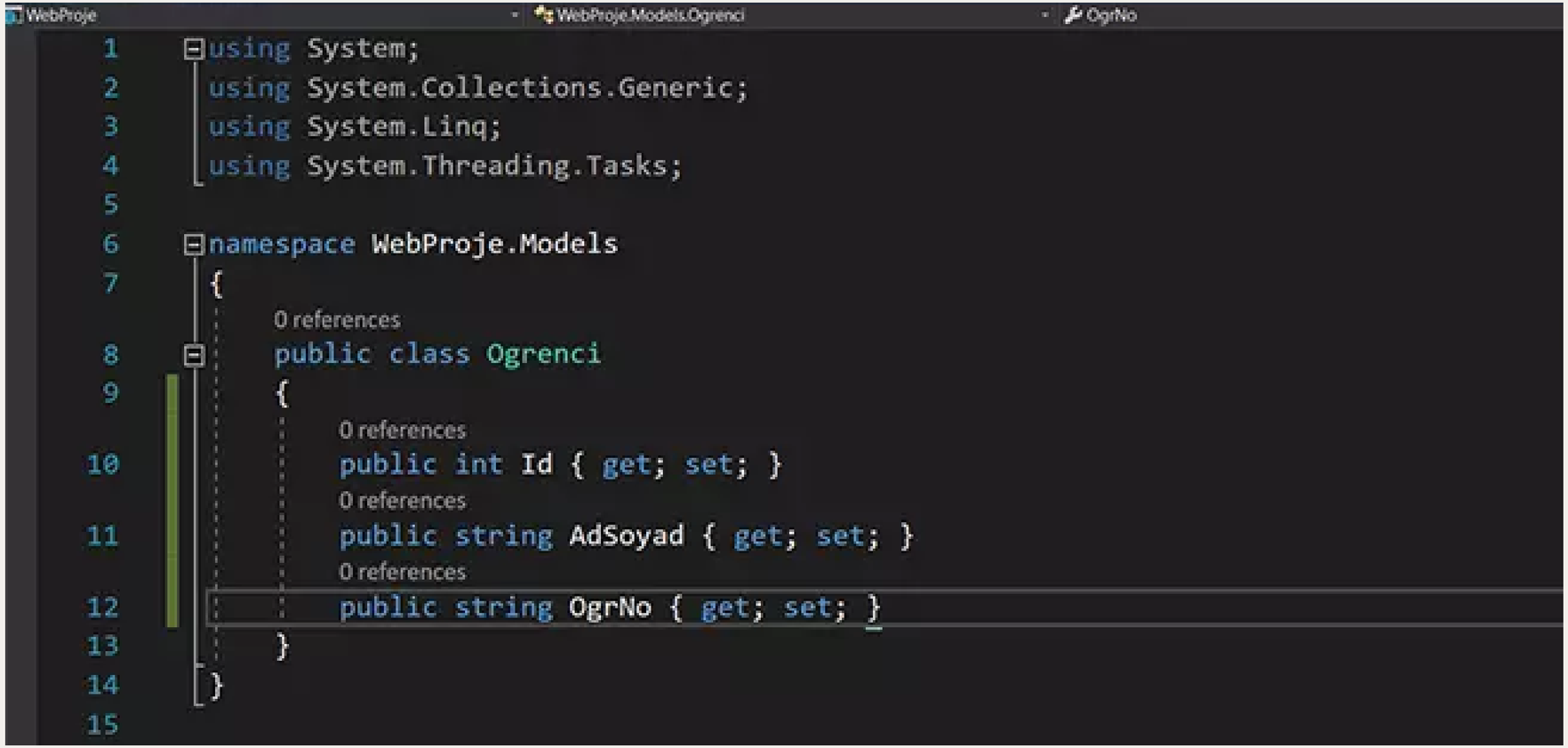
Katmanlı mimaride .Net Core için Entity Framework Core nugetini code first olarak nasıl ekleyecebileceğinizi göstereceğim.



Entities katmanı bizim için veritabanı tablolarımızın nesnelerini tutacak.

DataAccess katmanı da bizim veritabanı bağlantı contextimizi tutacak katman olacak. Ama öncelikle DataAccesse sağ tık yapıp Manage Nuget Packages diyerek nuget penceresini açıyoruz. Ve aşağıdaki paketleri kuruyoruz.





Önce tabloyu temsil eden sınıf oluşturduk.

```
ing System.Collections.Generic;
ing System.Linq;
ing System.Threading.Tasks;

mespace WebProje.Models

    5 references
    public class OrnekDbContext:DbContext
    {
        0 references
        public OrnekDbContext(DbContextOptions<OrnekDbContext> options) : base(options)
        {

        }
        0 references
        public DbSet<Ogrenci> Ogrencis { get; set; }
    }
```

Veritabanı ile bağlantı kuracağımız context sınıfını oluşturduk. Migration yapıları sayesinde de Visual Studio da yaptığımız değişiklikleri fiziksel veritabanında görebiliyoruz.

DbContext bir classtır ve Entity Framework'un olmazsa olmazıdır. DBContext veritabanımızla uygulamamız arasında sorgulama, güncelleme, silme gibi işlemleri yapmamız için olanak sağlar. Yani veritabanı içinde yer alan verilerimizle alakalı olarak her türlü süreçte iletişimimizi sağlayan bir classtır.

DbContext bize ne sağlar?

Database bağlantısının yönetimi,
Modellerimiz ve database ilişkilerinin yönetimi,
Database sorguları yönetimi,
Database veri kaydetme işlemleri,
Değişikliklerin izlenebilmesi,
Transaction (işlem) yönetimi,
Caching (Önbellek işlemleri)

Code First yapısında bir veritabanı dizayn ettik ve sıra bu veritabanını oluşturmaya geldi.Oluşturduğumuz bu yapı, hangi Server'da hangi isimde veritabanı oluşturacağını bilmiyor. "appsettings.json" dosyasının içine gerekli tanımlamamızı yapacağız.

DİNLEDİĞİNİZ İÇİN TEŞEKKÜRLER....