

ENTITY FRAMEWORK

Entity Framework Nedir?

Entity Framework, .Net platformunda kullanılan ORM (Object Relational Mapping) araçlarından biridir.

ORM (Object Relational Mapping) Nedir?

ORM (Object Relational Mapping) ise veritabanı ile nesneye yönelik programlama (OOP) arasındaki ilişkiyi kuran teknolojidir.

Yani Entity Framework, nesne tabanlı programlamada veritabanındaki tablolara uygun nesnelere oluşturma tekniğidir.

ORM için; Veritabanımızda yaratmış olduğumuz her bir nesneye karşılık kod tarafında bir nesne oluşturan programlardır diyebiliriz. Bu programlar **code generation** veya **shema generation** tekniği kullanarak bizim yazmamız gereken kodu otomatik üretiyor veya tam tersinde bizim yazdığımız kod şablonuna uygun database şemasını oluşturuyor.

Günümüzde kullanılan birçok ORM aracı bulunmaktadır. Örneğin; Java tabanlı olarak Hibernate, Flex'de Athena Framework, Delphi' de ECO gibi. Entity Framework ise Microsoft tarafından geliştirilen .Net tabanlı bir ORM aracıdır.

Neden ORM?

Veritabanı işlemleri ile ilgili kod yazımı en aza ineceğinden minimum zamanda maksimum iş çıkartmayı sağlar.

OOP düzeninde kod yazmayı sağlar.

Veritabanı olarak esnek yapıya sahiptir. Örneğin yazdığımız proje MSSQL ile çalışmakta ve birden Oracle' a geçmeniz istendi. Bunu yapabilmek için birçok ayar gerekirken Entity Framework ile direkt geçiş yapabilirsiniz.

Veritabanı bağımlılığı yoktur. Yani EF' yi oluşturmadan önce veritabanı tablo ve kolonlarını oluşturmalısınız gibi bir kural yoktur. Siz EF ile modellemeyi yaparken olmayan tabloları ve kolonları sizin yerinize açacaktır.

Maintenance (Bakım) daha kolaydır.

Her ne kadar EF kodlama olarak zaman kazandırsa da performans olarak **ADO.Net kadar hızlı değildir**. Bu nedenle yazılan her proje Entity ile olmak zorunda değildir ve/veya projenin tamamında entity kullanmak zorunda olmak sizin için dezavantaj olabilir.

Entity Framework; .Net 3.5 ile beraber Vs 2008 sp1 ile gelmiştir. Entity Framework ile 4 farklı yöntem ile proje geliştirilebilir. Bu yöntemler;

- Database First (Existing Database)
- Model First (New Database)
- Code First (New Database)
- Code First (Existing Database)

Database First (Önce Veritabanı) : Bu yöntemde hali hazırda var olan veritabanı projeye model dosyası ile bağlanır ve gerekli class' lar EF tarafından üretilir. İleride veritabanına yapılacak eklemelerde mevcut version Ef 6 da sorunlar yaşatabilir

Model First (Önce Model) : Bu yöntemde Visual Studio üzerinde boş bir model dosyası (.edmx) eklenerek veri tabanı bu model üzerinde tasarlanır. Derleme adımında verilen script dosyası ile veritabanı oluşturulur.

Code First (Önce Kod – Yeni Veritabanı) : Bu yöntemde classlar ve mapping kodları yazılımcı tarafından oluşturulur. Daha sonra veri tabanı bu class' lardan türetilir.

Code First (Önce Kod – Var olan Veritabanı) : Bu yöntemde de classlar ve mapping kodları yazılımcı tarafından oluşturulur. Veritabanı class' ların ve modellemenin durumuna göre tekrardan şekillenebilir.

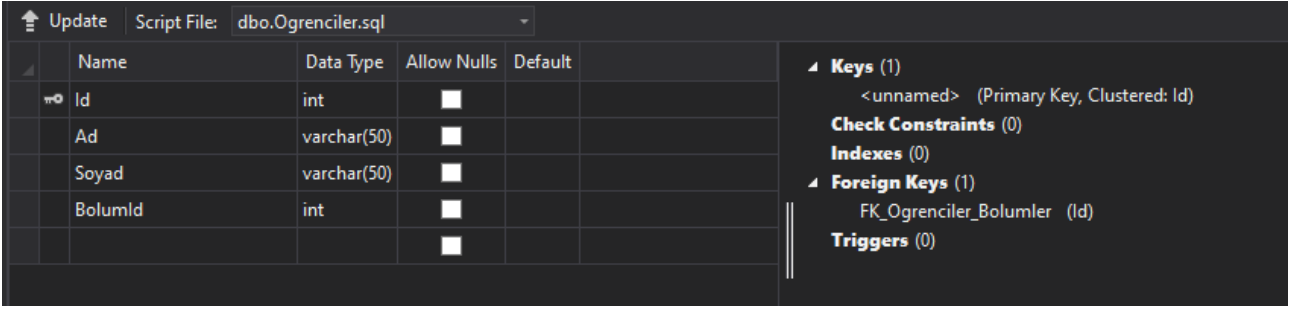
Kullanılan Yazılımlar

Dokümanların devamında yapılacak örneklerde aşağıda verilen yazılımlar kullanılmıştır.

- Visual Studio 2015
- SQL Server 2014 Express
- SQL Server Management Studio

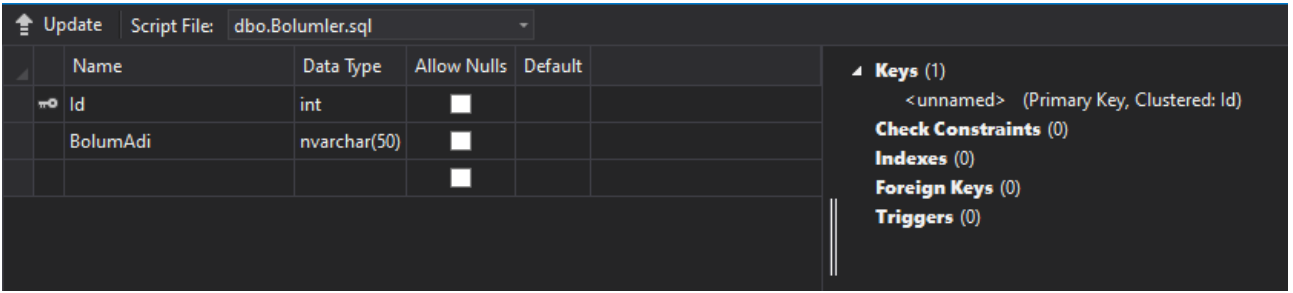
ÖNCE VERİTABANI (DB FIRST)

Önce Veritabanı yaklaşımına ait örneklerde aşağıdaki veritabanı yapısı kullanılacaktır. Bu yapıda; Öğrencilerin Ad, Soyad ve Bölüm Kodlarını tutacak **Oğrenciler** tablosu



Name	Data Type	Allow Nulls	Default
Id	int	■	
Ad	varchar(50)	■	
Soyad	varchar(50)	■	
BoluId	int	■	

Bölümlerin isim bilgisini tutan **Bolumler** tablosu



Name	Data Type	Allow Nulls	Default
Id	int	■	
BoluAdi	nvarchar(50)	■	

olmak üzere 2 tablo yer almaktadır. **Kişiler** tablosundaki **BolumKodu** ile **Bolumler** tablosundaki **Id** alanları arasında **İkincil Anahtar (Foreign Key)** ilişkisi bulunmaktadır.

Oğrenciler tablosunu oluşturmak için gerekli kod aşağıda verilmiştir.

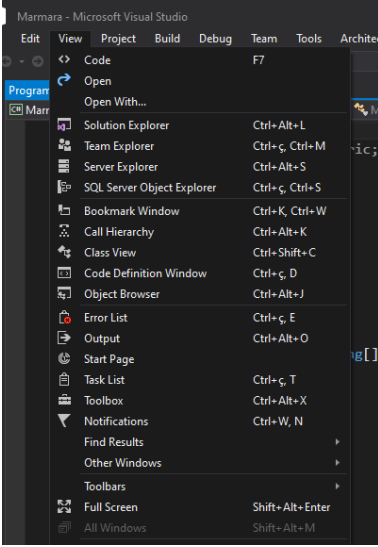
```
1 CREATE TABLE [dbo].[Oğrenciler] (  
2     [Id] INT IDENTITY (1, 1) NOT NULL,  
3     [Ad] VARCHAR (50) NOT NULL,  
4     [Soyad] VARCHAR (50) NOT NULL,  
5     [BoluId] INT NOT NULL,  
6     PRIMARY KEY CLUSTERED ([Id] ASC),  
7     CONSTRAINT [FK_Oğrenciler_Bolumler] FOREIGN KEY ([BoluId]) REFERENCES [dbo].[Bolumler] ([Id])  
8 );  
9  
10
```

Bolumler tablosunu oluşturmak için gerekli kod aşağıda verilmiştir.

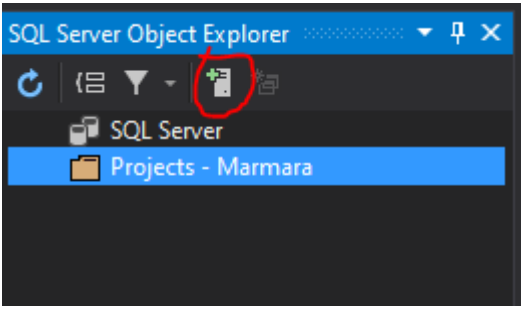
```
1 CREATE TABLE [dbo].[Bolumler] (  
2     [Id] INT IDENTITY (1, 1) NOT NULL,  
3     [BoluAdi] NVARCHAR (50) NOT NULL,  
4     PRIMARY KEY CLUSTERED ([Id] ASC)  
5 );  
6
```

Visual Studio Altında Veritabanı ve Tabloların Oluřturulması

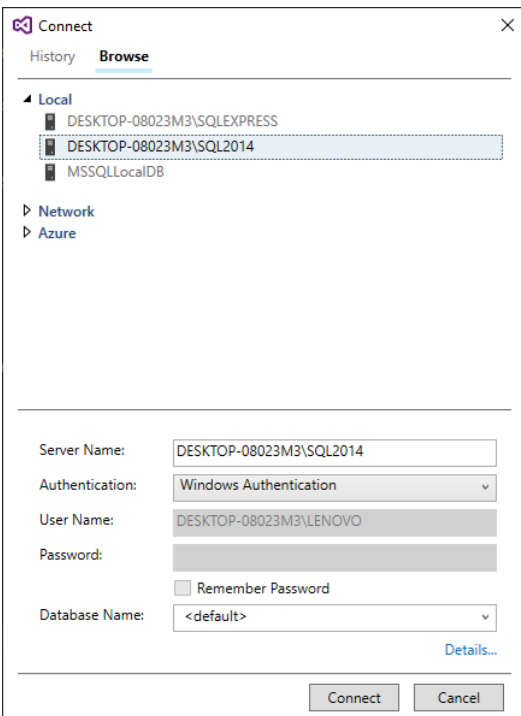
Visual Studio programını aarak View menüsünden SQL Server Object Explorer penesini aın



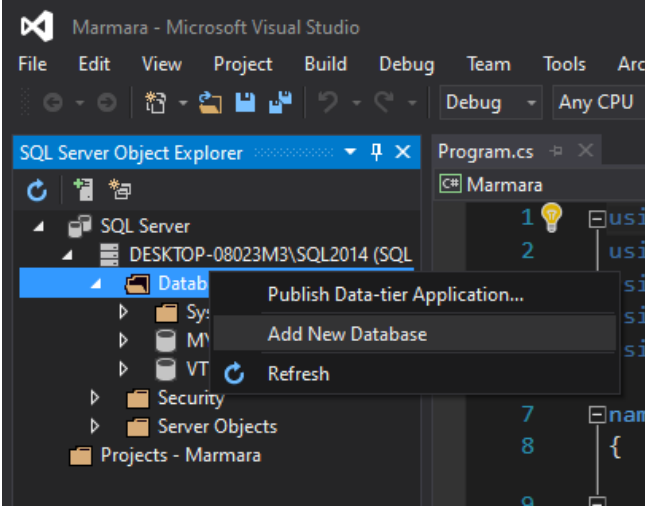
SQL Server Object Explorer panelinde yer alan **Add SQL Server** butonuna tıcklayınız.



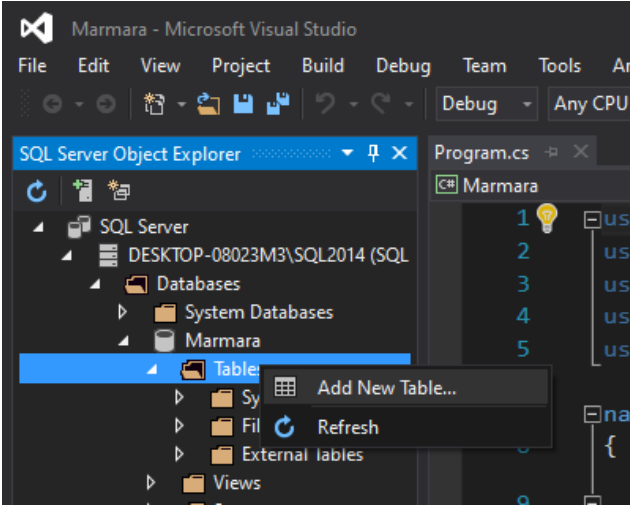
Aılan pencerede bilgisayarınızda ykl olan SQL serverlar listelenecektir. alıřmak istediėiniz sever adını listeden seerek servera baėlanmak iin gerekli kullanıcı adı ve řifre bilgilerini girerek connect butonuna basınız.



Database sekmesinde **farenin sağ tuşuna** basarak açılan menüden **Add New Database** seçeneğine tıklayınız. Veritabanına bir isim veriniz. (Örnekler Marmara Veritabanı üzerinde yapılacaktır.)



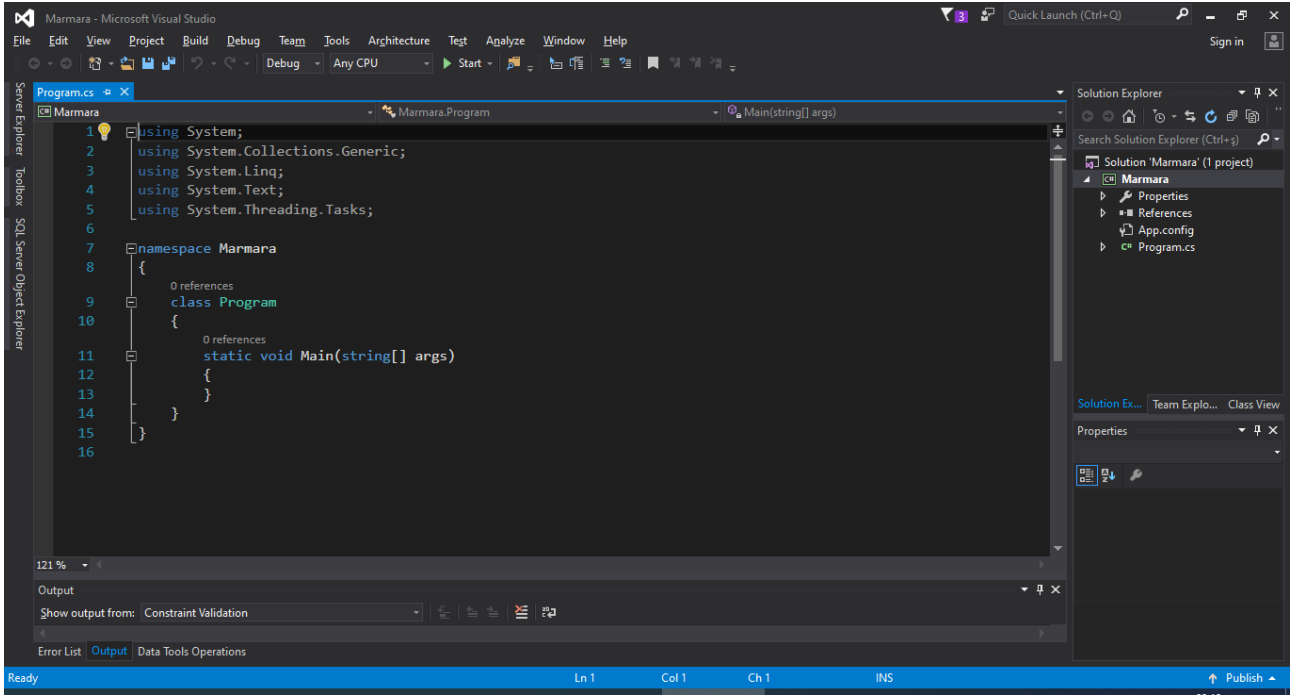
Oluşturduğunuz veritabanı altındaki tablolar sekmesinde **farenizin sağ tuşuna** basarak açılan menüden **Add New Table** seçeneğine tıklayınız.



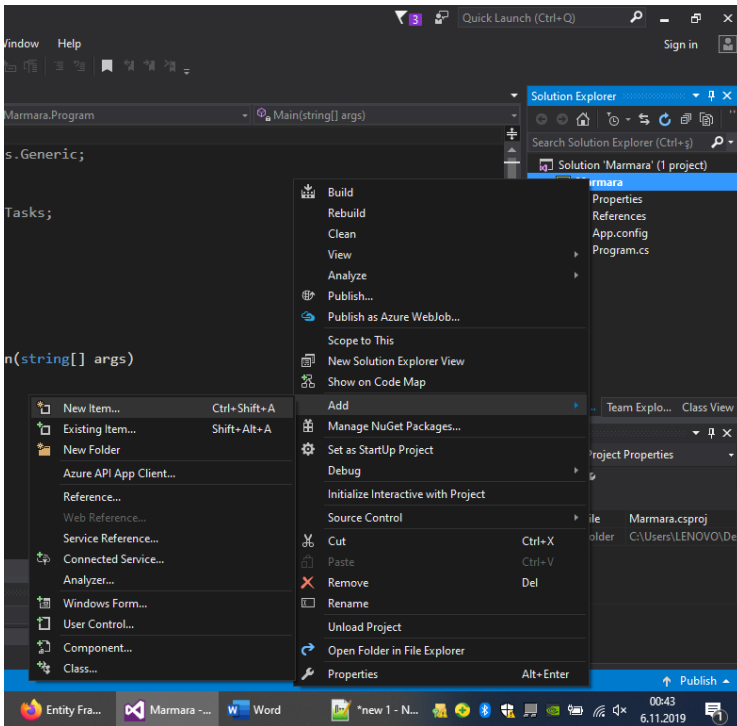
Yukarıda verilen **Oğrenciler** ve **Bolumler** tablolarını oluşturunuz.

Veritabanının Projeye Dahil Edilmesi

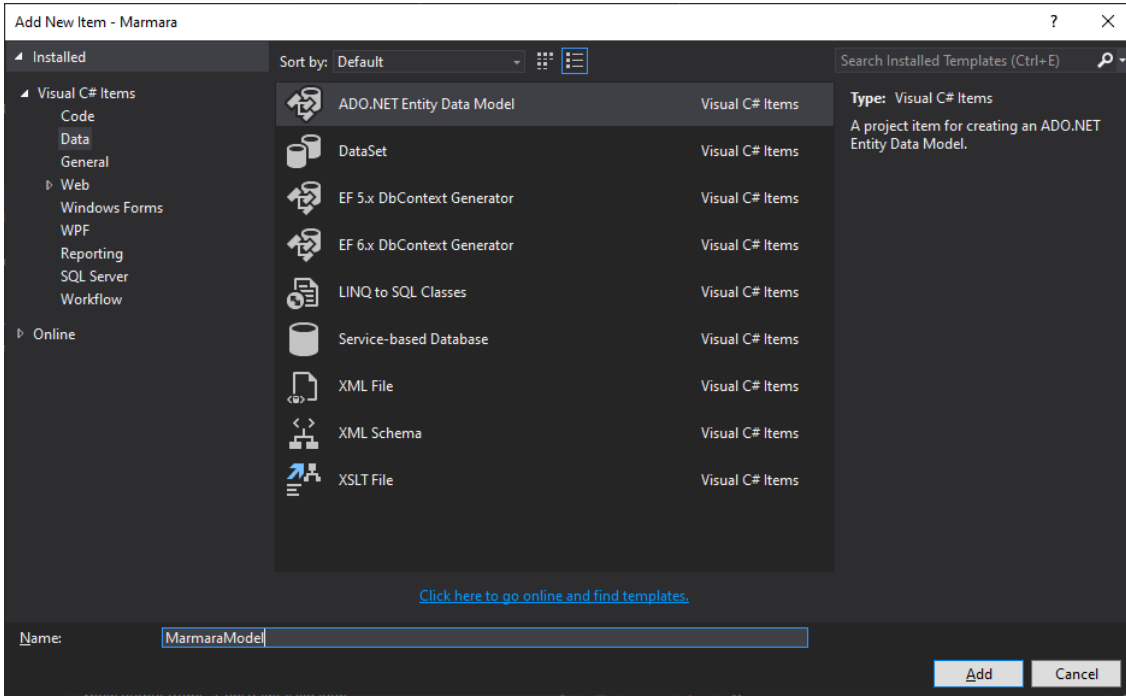
Boş bir C# konsol uygulaması açın



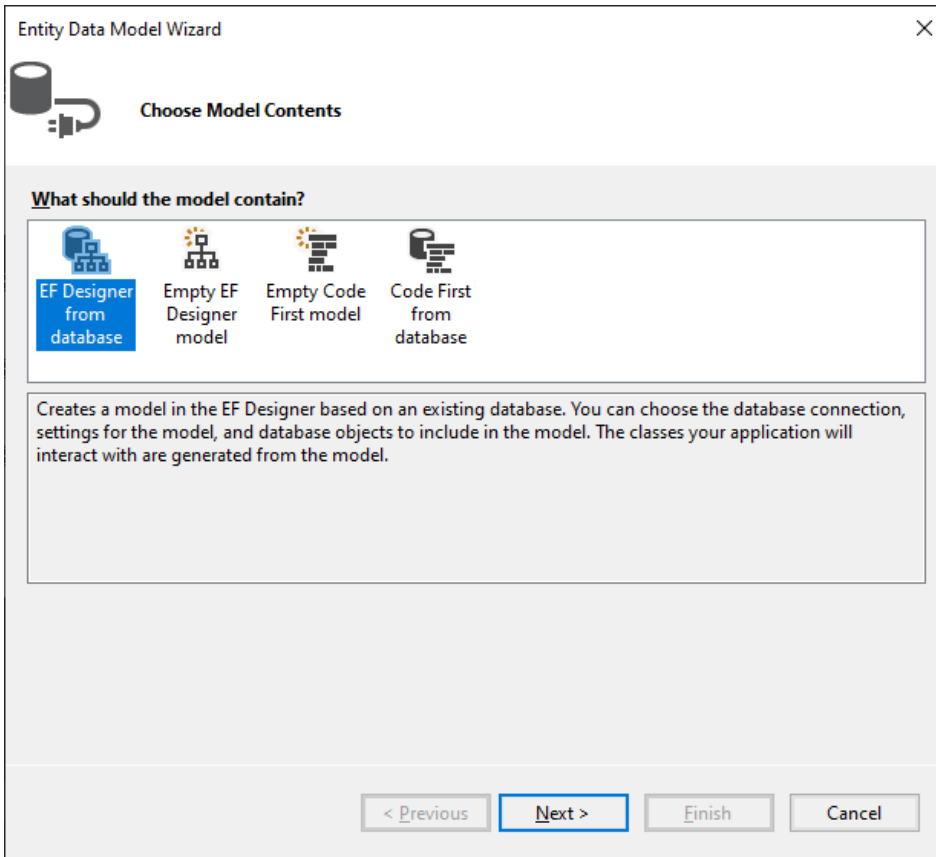
Solution Explorer panelinde yer alan proje adımızın (Marmara) üzerinde **farenin sağ tuşuna** basın. Açılan menüden sırasıyla Add > New Item seçeneğini seçiniz.



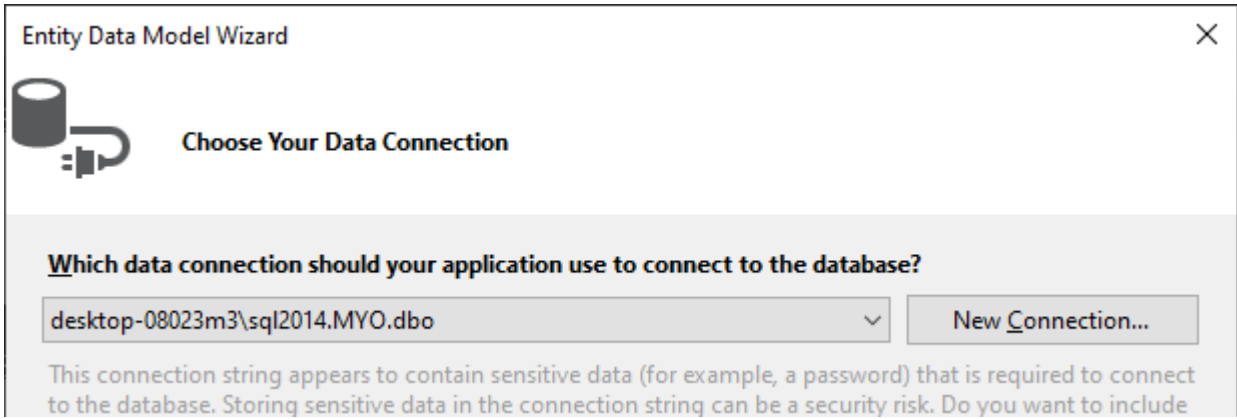
Açılan yeni pencerede **Data** alanında yer alan **ADO.NET Entity Data Model** seçeneğini seçerek alt kısımda yer alan **Name** kısmında modelimizin ismini veriniz. (Örneklere Marmara Model olarak geçecektir) **Add** butonuna basarak devam edin.



Açılan pencerede yer alan Entity Framework yöntemlerinden **DBFirst** yöntemi olan **EF Designer from Database** seçeneğini seçerek devam edelim.

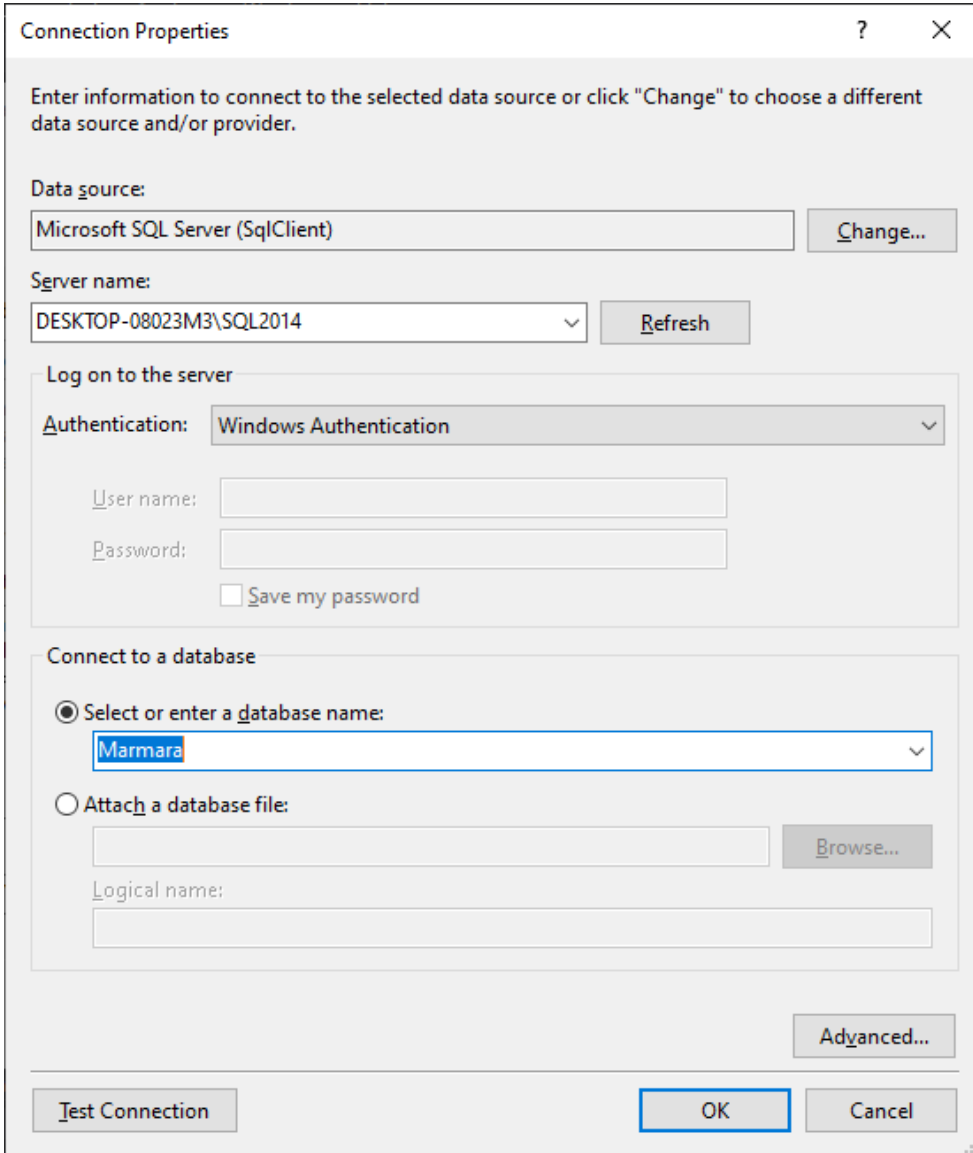


Gelen sekmede **New Connection** diyerek hangi veritabanı ile çalışacağımız hakkında gerekli ayarlamaları yapalım.



The dialog box is titled "Entity Data Model Wizard" and "Choose Your Data Connection". It asks the user to select a data connection. The selected connection is "desktop-08023m3\sql2014.MYO.dbo". A warning message states: "This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include".

Açılan pencerede Data Source, Server Name seçtikten sonra Server üzerinde oturum açarak (Authentication) **Connect to a database** kısmında listelenen veritabanları arasından kullanmak istediğiniz veritabanını seçiniz.



The dialog box is titled "Connection Properties". It contains the following fields and options:

- Data source:** Microsoft SQL Server (SqlClient) [Change...]
- Server name:** DESKTOP-08023M3\SQL2014 [Refresh]
- Log on to the server:**
 - Authentication:** Windows Authentication
 - User name:** []
 - Password:** []
 - Save my password
- Connect to a database:**
 - Select or enter a database name: Marmara
 - Attach a database file: [] [Browse...]
 - Logical name:** []

Buttons: [Test Connection], [Advanced...], [OK], [Cancel]

Bu aşamada Entity ayarlarımızın kayıt edileceği ismin oluşturulduğunu (MarmaraEntities) göreceksiniz.

Entity Data Model Wizard

Choose Your Data Connection

Which data connection should your application use to connect to the database?

desktop-08023m3\sql2014.Marmara.dbo New Connection...

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

No, exclude sensitive data from the connection string. I will set it in my application code.

Yes, include the sensitive data in the connection string.

Connection string:

```
metadata=res://*/MarmaraModel.csd|res://*/MarmaraModel.ssd|
res://*/MarmaraModel.msl;provider=System.Data.SqlClient;provider connection string="data
source=DESKTOP-08023M3\SQL2014;initial catalog=Marmara;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework"
```

Save connection settings in App.Config as:

MarmaraEntities

< Previous Next > Finish Cancel

Entity Framework versiyonunu seçiyoruz.

Entity Data Model Wizard

Choose Your Version

Which version of Entity Framework do you want to use?

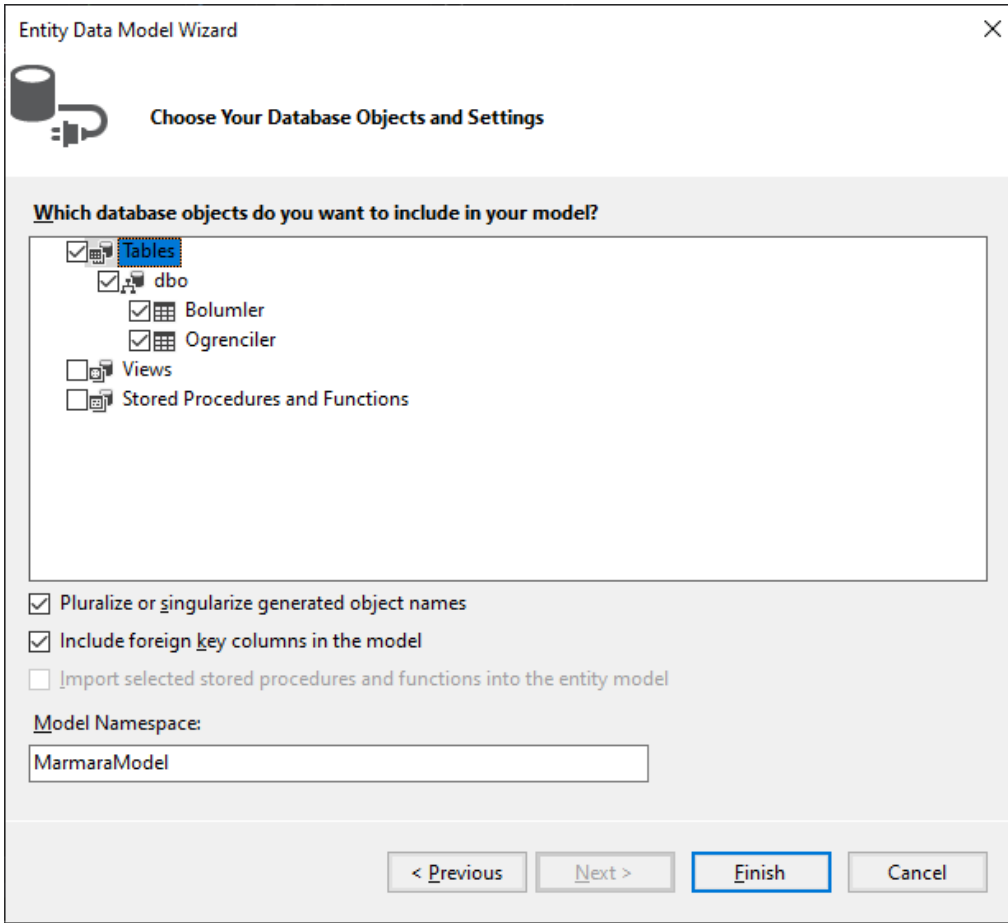
Entity Framework 6.x

Entity Framework 5.0

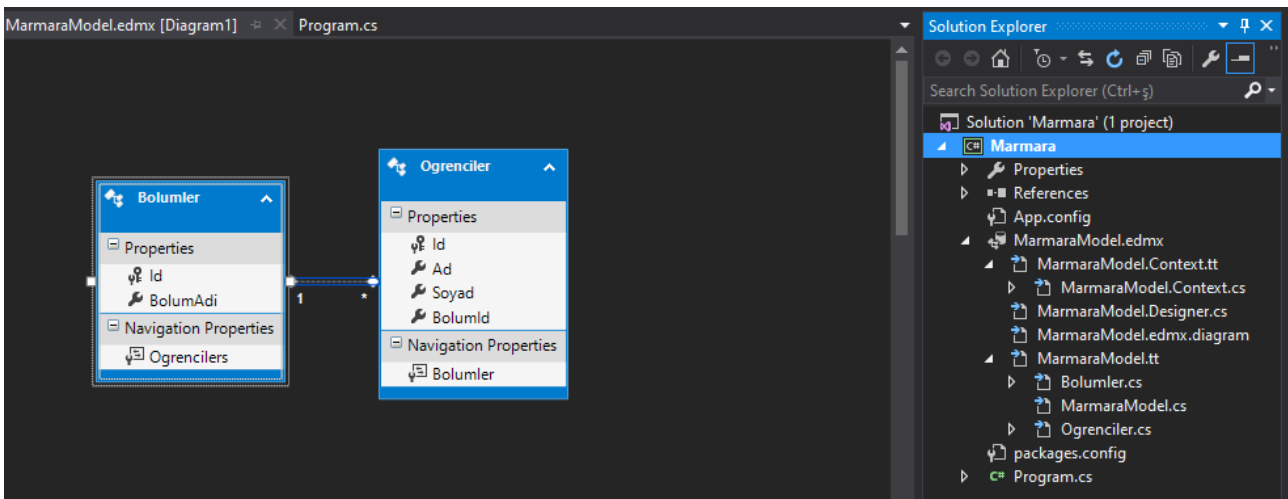
i It is also possible to install and use other versions of Entity Framework.
[Learn more about this](#)

< Previous Next > Finish Cancel

Projeye veritabanı üzerinde yer alan hangi nesnelerin (tablolar, fonksiyonlar, kayıtlı prosedürler vb.) dahil edileceğini seçtikten sonra **Finish** diyerek veritabanı bağlantısını projeye eklemiş oluyoruz.



Hazırladığımız veritabanı Entity Framework yardımı ile projemize eklenmiş oldu. MarmaraModel.edmx dosyası bize tablolarımız ve aralarındaki ilişkileri diyagram şeklinde gösterir. Ayrıca Solution Explorer pençesine bakacak olursanız Bolumler.cs ve Ogrenciler.cs sınıflarının da oluşturulmuş olduğunu göreceksiniz.



CRUD (Create – Read – Update – Delete) İşlemleri

Veritabanına Kayıt Ekleme

Veritabanına kayıt yapabilmek için öncelikle hangi tabloya kayıt eklenecekse o class türünden yeni bir nesne oluşturulur. Daha sonra bu nesneye özellik bilgileri eklenir ve **context** nesnesinin **Add** metodu kullanılarak veritabanına eklenir. **Context** nesnesinin **SaveChanges** metodu kullanılmadan yapılan değişiklikler veritabanı üzerinde uygulanmaz.

```
//Öğrenci Ekleme
Ogrenciler yeniogrenci = new Ogrenciler
{
    Ad = "Bekir",
    Soyad = "Kurt",
    BoluId = 1
};
context.Ogrenciler.Add(yeniogrenci);
context.Ogrenciler.Add(new Ogrenciler { Ad = "Kurt", Soyad = "Bekir", BoluId=2 });
context.SaveChanges();
```

Veritabanında Varolan Kayıtları Listeleme

Veritabanında var olan kayıtlar üzerinde işlem yapabilmek için **Context** nesnesinden faydalanılır. Belirli bir tablodaki tüm verileri göstermek için bir foreach döngüsü kurularak tüm veriler sırası ile ekrana getirilebilir.

```
//Listeleme
List<Ogrenciler> tumogrenciler = context.Ogrenciler.ToList();
tumogrenciler.ForEach(xyz => Console.WriteLine("Ad : {0}", xyz.Ad));
Console.ReadKey();
```

veya

```
foreach(var ogrenci in context.Ogrenciler)
{
    Console.Write("Öğrenci Adı : {0}", ogrenci.Ad);
    Console.WriteLine("Öğrenci Soyadı : {0}", ogrenci.Soyad);
}
Console.ReadLine();
```

Veritabanında Varolan Belirli Kayıtları Listeleme

Veritabanındaki tablolarda yer alan tüm verileri listelemek yerine belirli bir özelliği olan verileri görüntülemek istersek (Bölüm ID si 1 olan bölümde okuyan öğrenciler gibi) context nesnesi ile veritabanına erişirken bu parametreyi de belirtmemiz gerekir.

```
//Ogrenci Listele
Bolumler bolumbul = context.Bolumlers.FirstOrDefault(r => r.Id==1);
List < Ogrenciler > listelenecekler = bolumbul.Ogrencilers.ToList();
listelenecekler.ForEach(g => Console.WriteLine(g.Ad));
Console.ReadKey();
```

Veri Güncelleme

Veritabanında yer alan bir kayda eriştikten sonra o kayıt üzerinde düzenleme (güncelleme) yapmak çok kolaylaşır. Örneğin Adı “Ahmet” olan kaydın adını ve soyadını değiştirmek istersek yapmamız gereken Ahmet e ait bilgileri context üzerinden gerekli nesneye aktarmaktır.

```
//Güncelleme
Ogrenciler ogrencibul = context.Ogrenciler.FirstOrDefault(r => r.Ad == "Ahmet");
ogrencibul.Ad = "Kurt";
ogrencibul.Soyad = "Bekir";
context.SaveChanges();
```

Veri Silme

Veritabanında yer alan bir kayda eriştikten sonra o kayıt üzerinde düzenleme (silme) yapmak çok kolaylaşır. Öğrenciler tablosundaki id si 1 olan kaydı silmek istersek, o kayda ait bilgilere context üzerinden erişerek silebiliriz.

```
//Silme
Ogrenciler ogrencibul = context.Ogrenciler.FirstOrDefault(r => r.Id == 1);
context.Ogrenciler.Remove(ogrencibul);
context.SaveChanges();
```

IQueryable Kullanımı

IQueryable kullanılırken sorgu veri kaynağı (SQL Server) için oluşturulur. Sorgular sadece **ToList** veya **Foreach** gibi metotlar çağırılarak çalıştırılır. Her seferinde veritabanına erişmeye gerek kalmadan **Where**, **OrderBy** veya **Select** gibi ifadeler kullanılan sorgular yazmamızı sağlar.

Öğrenciler tablosunda BolumID değeri 1 olan kayıtları soyadına göre listelemek istersek gerekli sorguyu IQueryable ile hazırlayıp ToList metodu ile listeleyebiliriz.

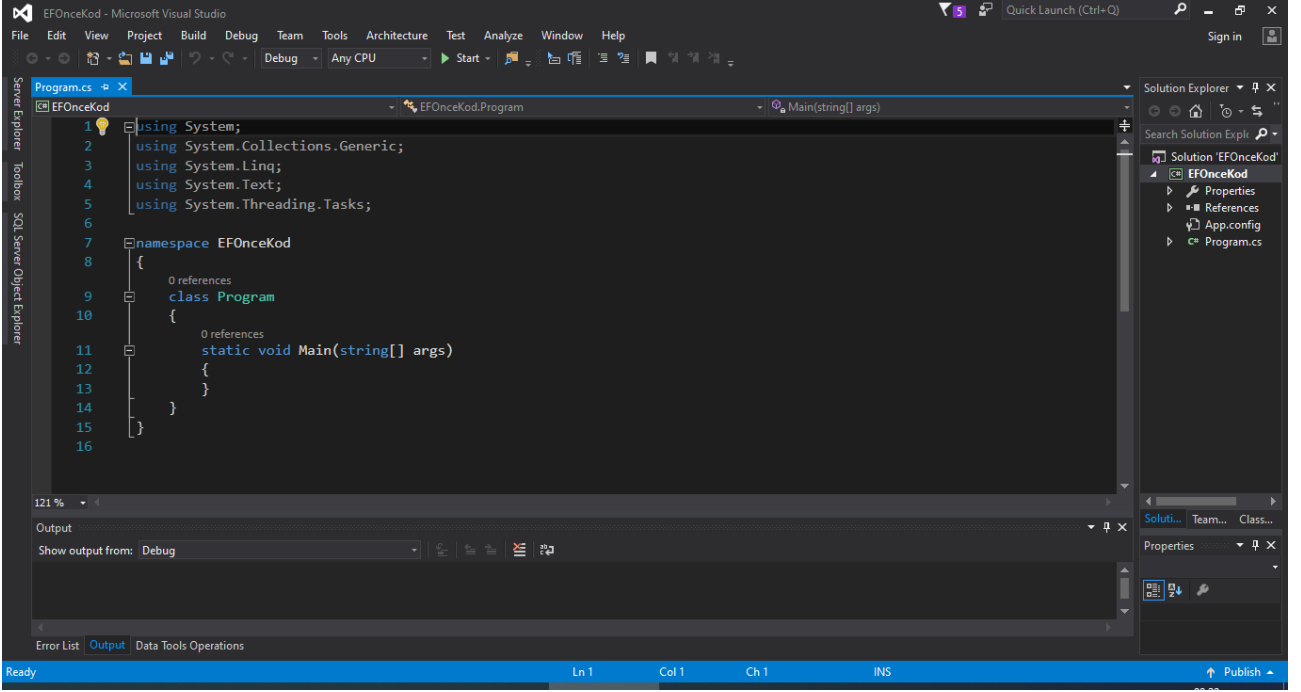
```
IQueryable<Ogrenciler> sorgu = context.Ogrenciler
    .Where(p => p.BolumId == 1)
    .OrderBy(p => p.Soyad);
//Sorgu hazır olmasına rağmen veritabanına erişmez.
//List komutu ile birlikte veri tabanına erişim sağlanır.
List<Ogrenciler> listelenecekler = sorgu.ToList();
listelenecekler.ForEach(p => Console.WriteLine(p.Ad));

Console.ReadKey();
```

ÖNCE KOD (Code First) Yöntemi

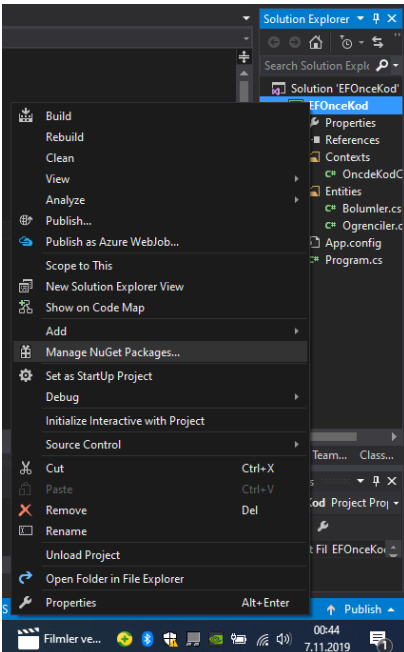
Bu yöntemde classlar ve mapping kodları yazılımcı tarafından oluşturulur. Daha sonra veri tabanı bu class' lardan türetilir.

Bunun için öncelikle Visual Studio programında boş bir konsol uygulaması açıyoruz



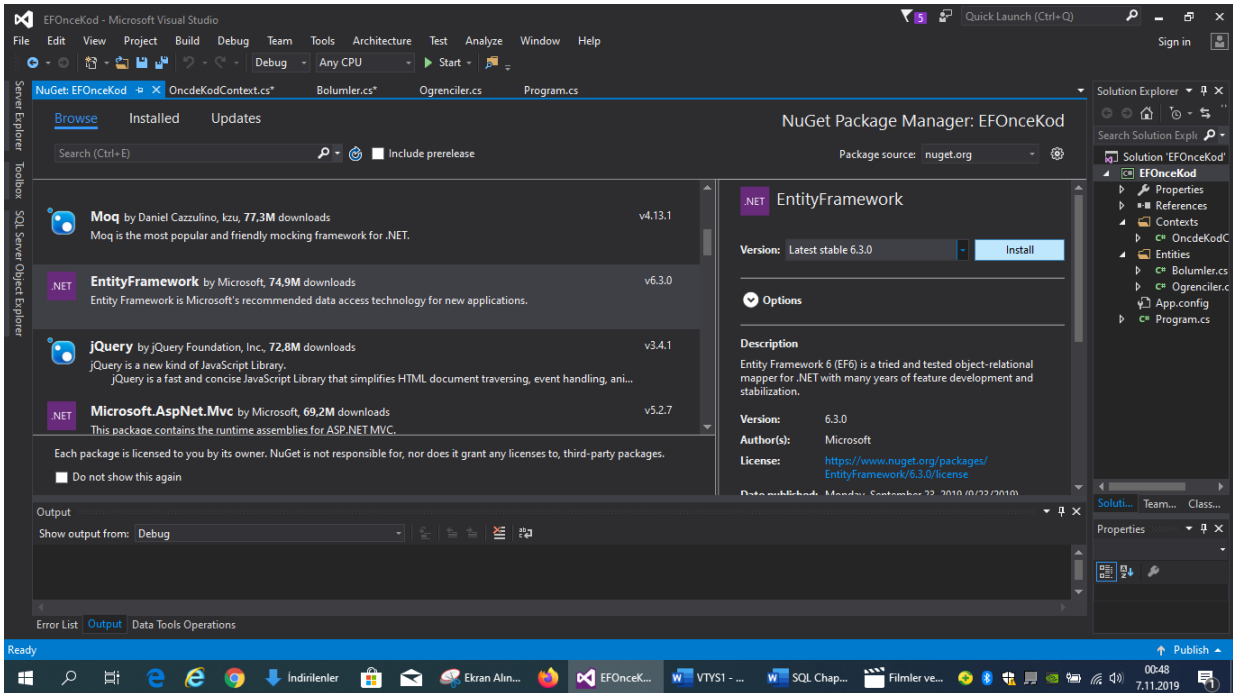
Daha sonra Veritabanında yer almasını istediğimiz her bir tablo için bir sınıf (class) tanımlaması yapmamız gerekir. Tüm bu sınıf tanımlamalarına daha kolay ulaşabilmek ve karmaşanın önüne geçmek için **Entities** isiminde bir klasör oluşturarak sınıf tanımlamalarını bu klasör altında oluşturuyoruz. (DB First örneğine benzerlik olması açısından bu örnekte aynı veritabanının oluşturulması sağlanacaktır) Bizde **Ogrenciler** ve **Bolumler** tabloları için onlara karşılık gelen sınıf tanımlamalarını yapıyoruz.

Ayrıca Veritabanı ile program arası veri alışverişi için gerekli olan **Context** nesnesini / nesnelarini tanımlamak için **Contexts** isimli bir klasör oluşturarak bu klasör içinde her bir contex im için bir sınıf tanımlaması yapıyoruz.

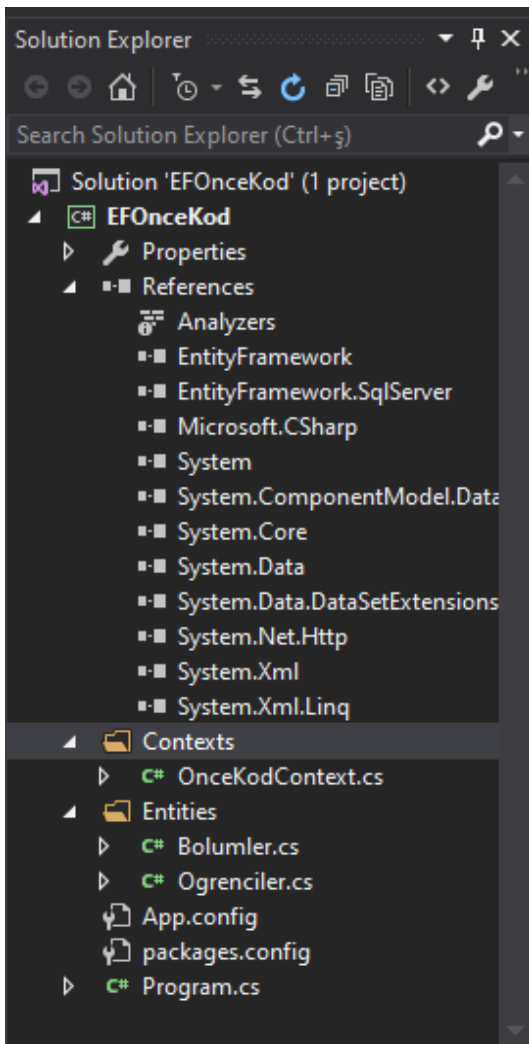


Bu işlemleri yapabilmemiz için öncelikle projemize Entity Framework ü dahil etmemiz gerekiyor. Bunun için proje ismi üzerinde **sağ tuşa** tıklayarak açılan listeden **Manage NuGet Packages** seçeneğine tıklıyoruz

Açılan pencerede **Entity Framework** seçeneğini bularak **INSTALL** butonuna basarak projeye dahil ediyoruz.



Entity Framework de projeye eklendikten sonra Solution Explorer paneli soldaki şekilde görünür. Entities ve Contexts sınıflarının içeriği sağ tarafta verilmiştir.



```
public class Ogrenciler
{
    0 references
    public int ID { get; set; }
    0 references
    public string Ad { get; set; }
    0 references
    public string Soyad { get; set; }
    0 references
    public int BolumID { get; set; }
}
```

```
public class Bolumler
{
    0 references
    public int ID { get; set; }
    0 references
    public string BolumAdi { get; set; }
    0 references
    public List<Ogrenciler> Ogrenciler { get; set; }
}
```

```
class OnceKodContext:DbContext
{
    0 references
    public DbSet<Ogrenciler> Ogrenciler { get; set; }
    0 references
    public DbSet<Bolumler> Bolumler { get; set; }
}
```

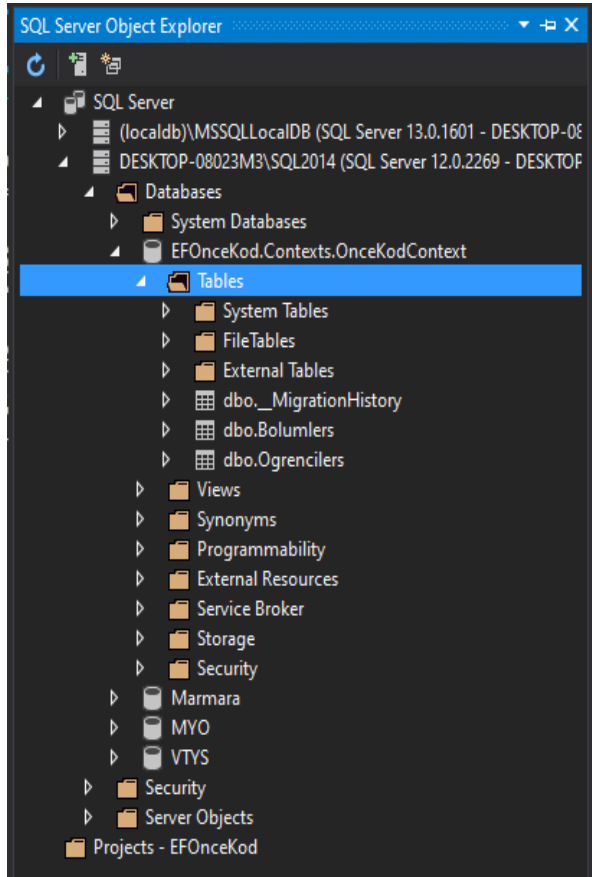
Veritabanının Oluşturulması

Kod kısmında veritabanının oluşturulması için tüm tanımlar yapıldıktan sonra. Context nesnesinin bir defa çağırılması sonucu veritabanı otomatik olarak oluşur.

Veritabanı üzerindeki tüm kayıtları görüntüleyen bir program yazıp bunu çalıştırdığımızda program veritabanını bulamadığı için otomatik olarak oluşturur.

```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        using (var OnceKodContext = new OnceKodContext())
        {
            foreach (var ogrenci in OnceKodContext.Ogrenciler)
            {
                Console.WriteLine("Öğrenci Adı:{0}", ogrenci.Ad);
                Console.ReadLine();
            }
        }
    }
}
```

Program çalıştırıldıktan sonra programın namespace i ne ise o isimde bir veritabanı dosyası oluşturur.



Name	Data Type	Allow Nulls	Default
ID	int	<input type="checkbox"/>	
Ad	nvarchar(MAX)	<input checked="" type="checkbox"/>	
Soyad	nvarchar(MAX)	<input checked="" type="checkbox"/>	
BolumID	int	<input type="checkbox"/>	
Bolumler_ID	int	<input checked="" type="checkbox"/>	

Name	Data Type	Allow Nulls	Default
ID	int	<input type="checkbox"/>	
BolumAdi	nvarchar(MAX)	<input checked="" type="checkbox"/>	