

Entity Framework *(Varlık-İskelet)*

Arif GÜNEL



- *Entity Framework Microsoft tarafından geliştirilen ve yazılım geliştiricilerin katı sql sorguları yazmalarını ortadan kaldırarak bir ORM (Object Relational Mapping-Nesne İlişkisel Eşleştirme) imkanı sağlayan framework'tür.*
- *ORM ise ilişkisel veri tabanı yönetim sistemlerine direkt olarak müdahale yerine nesneler aracılığı ile müdahale edilmesini sağlayan bir köprüdür diyebiliriz*

- *Piyasada bir çok ORM Framework'leri bulunmaktadır. Örnek olarak; DataObjects.Net, NHibernate, OpenAccess, SubSonic etc. Entity Framework vs.*
- *Entity framework'ün aslında temel amacı uygulama geliştiricinin data işlemleri ile çok haşır neşir olmadan uygulama tarafına odaklanmasını sağlamaktır.*

Entity Framework ' ü kullanmanın avantajları ;

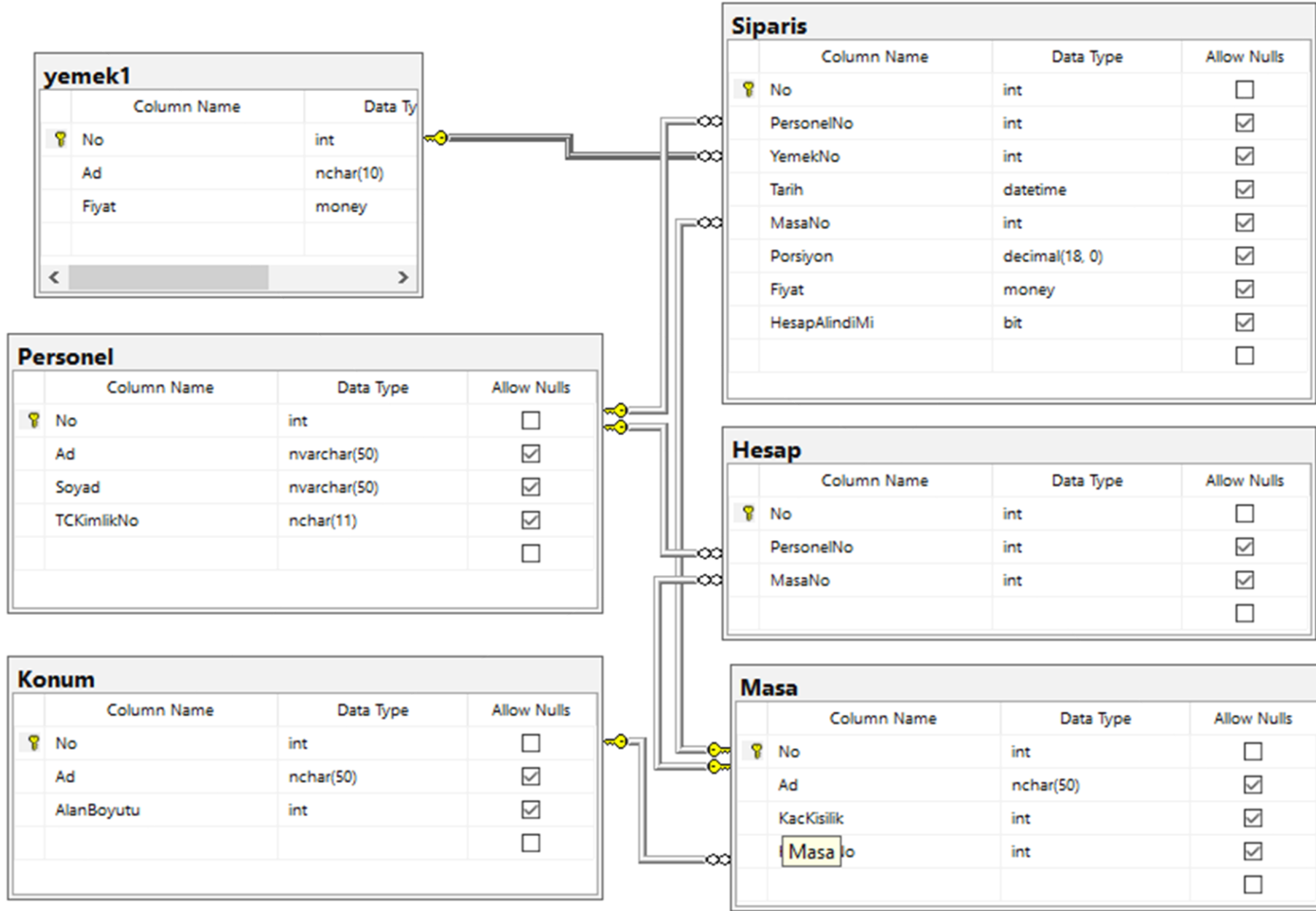
- *CRUD (Create , Read , Update , Delete) işlemleri ile uzun , karışık ve zahmet verici SQL kodlarından kurtulmamızı sağlar.*
- *Veri tabanına bağımlılığı ortadan kaldır.*
- *Veri tabanı işlemlerinde nesneye yönelik kod yazmamızı sağlar.*
- *Daha sade ve zahmetsiz SQL sorguları sayesinde veri tabanı performansını artırır.*
- *Yazılım geliştiricinin data işlemleri ile haşır neşir olmadan sadece uygulama üzerinde odaklanmasına olanak sağlar. Hiçbir SQL bilgisi olmayan bir kimse veri tabanı işlemlerini EF ile gerçekleştirebilir.*
- *Kod yazma süresini kısaltarak daha az zamanda daha çok iş yapmayı sağlar.*

Bir örnek üzerinde işleyelim

- Uzun bir örnek olacağı için veri tabanı hazırlayalım bunun üzerinde işlemleri yapalım.

Ornek: Veri tabanına veri giriři

- 1.ADİM : Veritabanı oluřturma
- İlk olarak bir veri tabanı oluřturalım.
- Management Studio kullanılarak Lokanta isminde bir veri tabanı oluřturalım. Diyagram görünümündeyken Yemek isminde bir tablo oluřturalım.



*Diyagram
görünümündeyken;*

- *Yemek*
- *Masa*
- *Konum*
- *Hesap*
- *Personel*
- *Görev*
- *Personel Görev*
- *Sipariş*

*isimlerinde
tablolarımızı
oluşturalım.*

2.ADIM : Form oluşturma

Yemek			
	Column Name	Condensed Type	Nullable
🔑	No	Yemek	No
	Ad	nvarchar(50)	Yes
	Fiyat	money	Yes

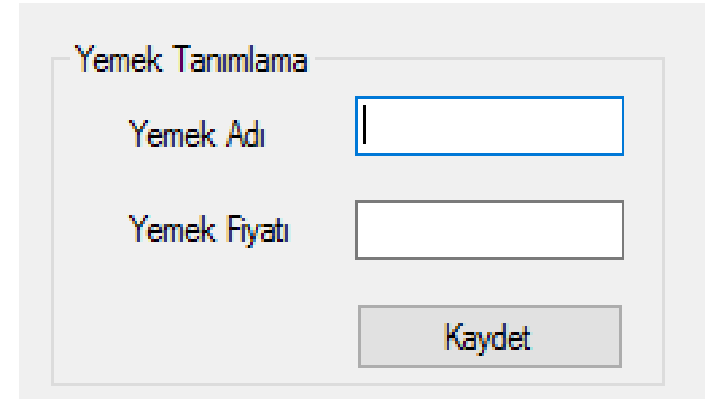
- İlk olarak veri ekleme işlemlerimizi yapmak için form tasarımı ile başlayalım.
- İlk olarak **Yemek** tablosuna bir şeyler ekleyebileceğimiz form yapalım.
- No alanı otomatik olarak geldiğinden no alanı hariç Ad ve Fiyat alanları için veri girişi ve eski kayıtlarda güncelleme yapacağımız `textBox`' ları ilave ediyoruz.

Formda kullanılan kontroller

- 1 adet **groupBox** ilave edelim

(Text Yemek Tanımlama)(içine koyduğumuz kontrolleri gruplamaya yarıyor)

- 2 adet **Label** (text: Yemek Adı, Yemek Fiyatı-TextBoxları isimlendirmek için)
- 2 adet **TextBox** (name tbYemekAdi,tbYemekFiyati,)
- 1 adet **Button** (name btnYemekKaydet, text Kaydet)

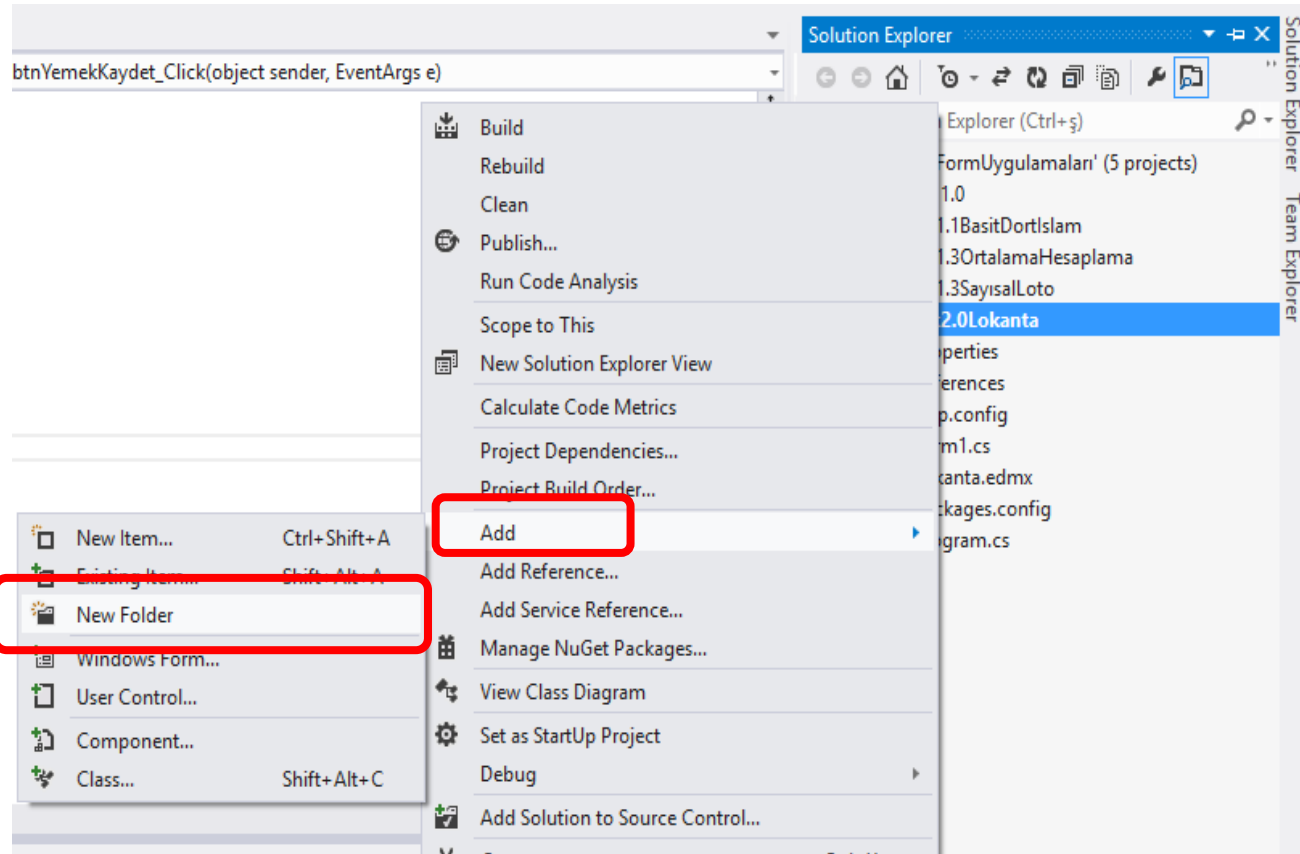


The screenshot shows a Windows Form titled "Form1". Inside the form, there is a group box titled "Yemek Tanımlama". Within this group box, there are two labels: "Yemek Adı" (Food Name) and "Yemek Fiyatı" (Food Price). Each label is followed by a text box for input. Below these text boxes, there is a button labeled "Kaydet" (Save).

- *Button'a tıklayarak Click olayını yazalım. Butona tıklandığında TextBox'taki bilgileri veri tabanına kaydetmek istiyoruz.*
- *Bu kısımda Entity Framework devreye giriyor. Entity Framework kullanılarak veri tabana veri girişi, veri okuma ve sorgu işlemleri gerçekleştireceğiz.(Ado.net kullanmayacağız)*

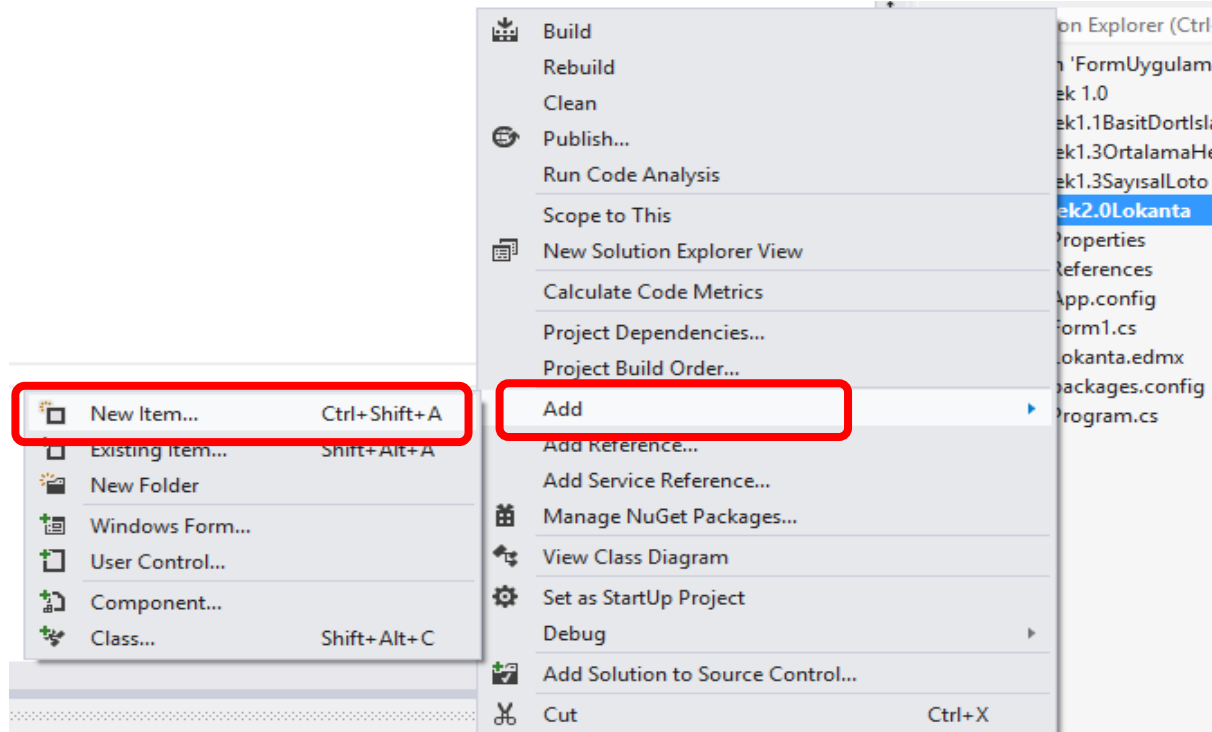
- *Bunun için Entity Framework DLL 'ini dahil etmemiz gerekiyor. Entity Framework bir ORM aracıdır. ORM araçlarının amacı ;veri tabanında tablolar şeklinde bulunan verilerimizi classlara dönüştürerek(sınıf yapısına) projemize dahil etmemizi sağlayan bir araçtır.Örneğin veri tabanındaki Yemek isimli tabloyu Yemek.cs dosyası olarak projemize dahil ediyor.Konum.cs, masa.cs gibi her biri bir class olarak projeye dahil ediyor.*

- *Entitiy Framework veri tabanının bir modelini class yapısı olarak oluşturacağından dosyalar için bir klasör oluşturalım. Proje üzerinde sağ tıklayıp **Add** deyip **New Folder**'ı seçiyoruz.İsmi DataModel diyelim. Bu klasör içerisinde Veri tabanının bir modelini kod tarafında bize oluşturacak.*

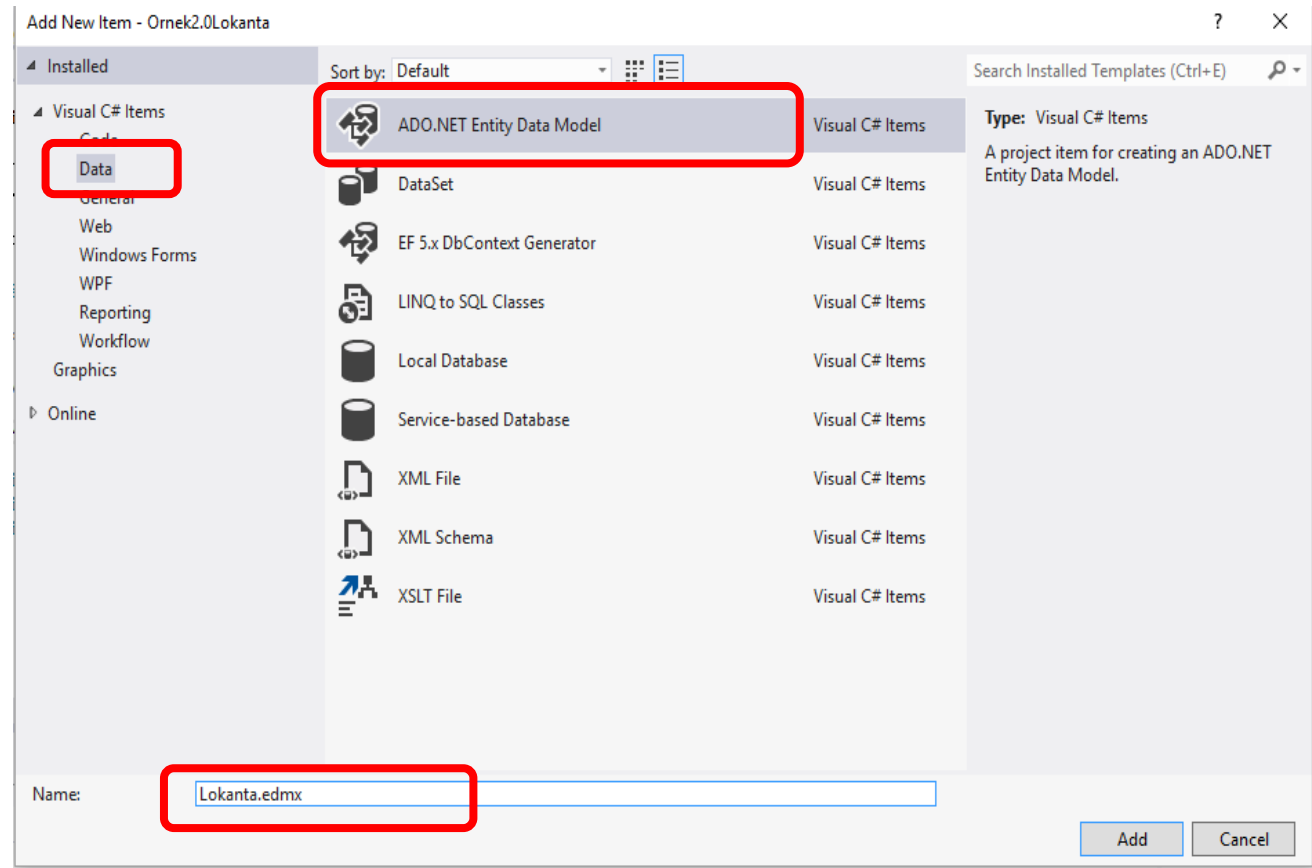


Entity Framework oluşturmak;

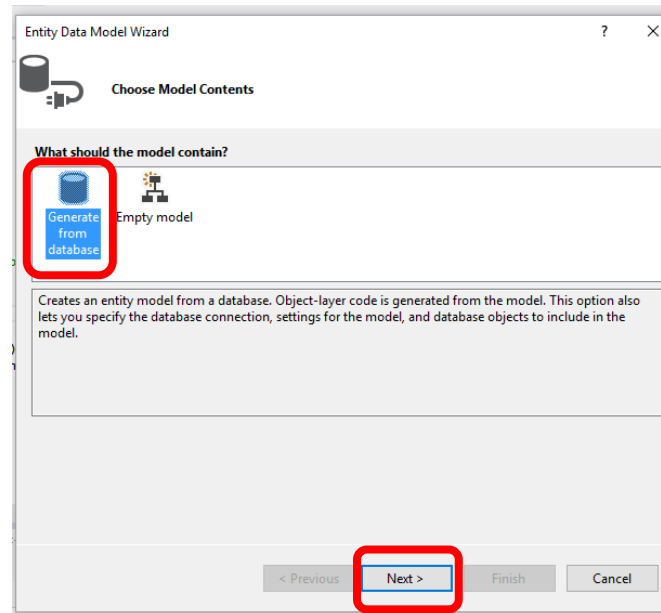
- *Proje üzerinde sağ tıkla ---> Add ---> New Item*



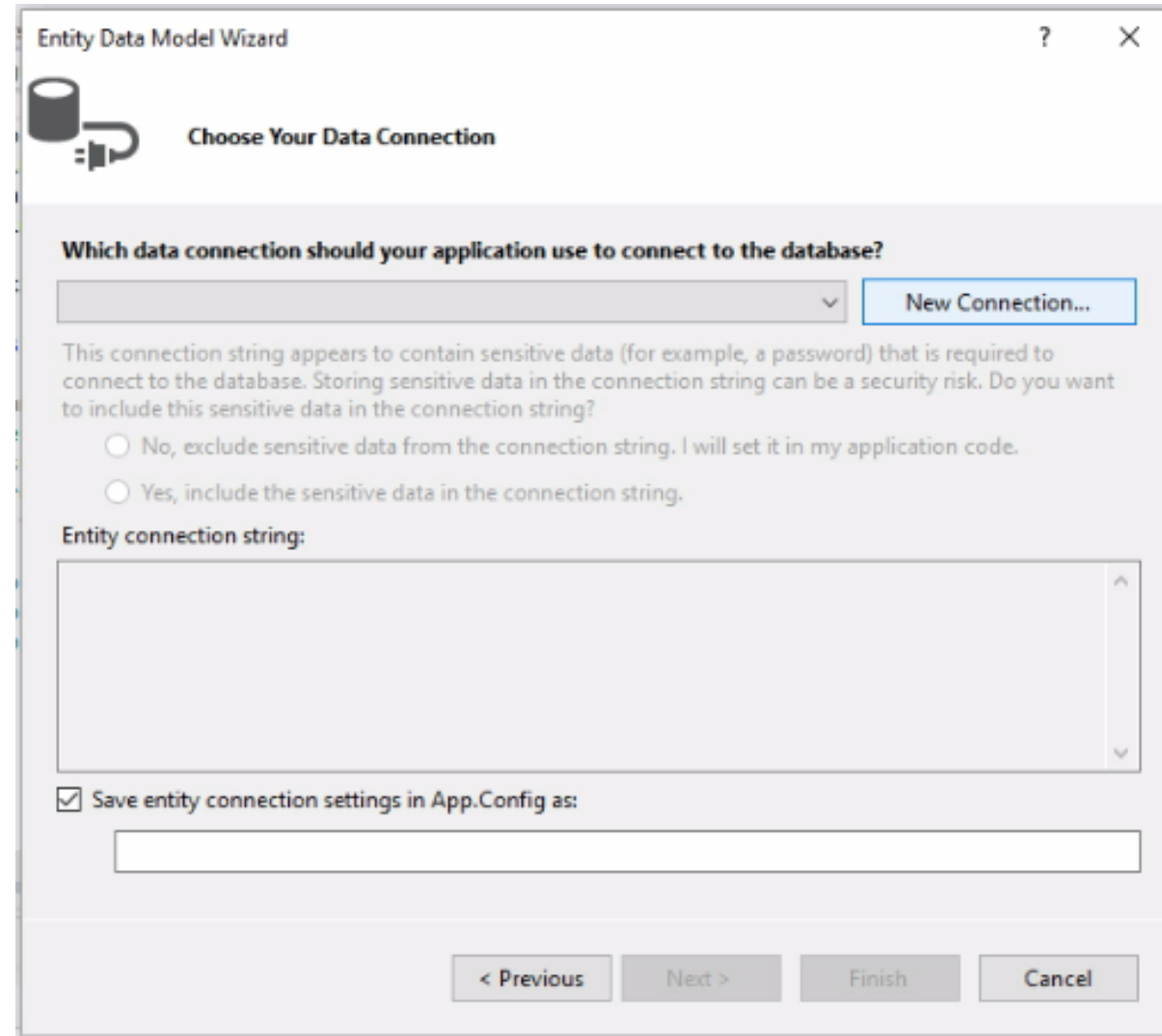
- *Data -- → ADO.NET Entity Data Model . Buradan Name kısmına isim veriyoruz.*
- *.edmx entity framework'ün dosya uzantısı.*



- Gelen pencereden veri tabanını seçtirecek. Bir veri tabanımız olduğundan modelin veri tabanından oluşmasını istiyoruz.
- Generate(oluşturmak) From Database seçip Next diyoruz



- *Bağlanmak istediğimiz veri tabanı bilgilerini soruyor.*
- *New Connection(yeni bağlantı) diyoruz.*



The image shows a screenshot of the 'Entity Data Model Wizard' dialog box, specifically the 'Choose Your Data Connection' step. The window has a title bar with a question mark and a close button. Below the title bar is a header area with a database icon and the text 'Choose Your Data Connection'. The main content area contains a question: 'Which data connection should your application use to connect to the database?'. Below this question is a dropdown menu and a 'New Connection...' button. A warning message follows: '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?'. There are two radio buttons: 'No, exclude sensitive data from the connection string. I will set it in my application code.' and 'Yes, include the sensitive data in the connection string.'. Below the radio buttons is a text box labeled 'Entity connection string:'. At the bottom, there is a checkbox labeled 'Save entity connection settings in App.Config as:' with an empty text box next to it. The bottom of the dialog has four buttons: '< Previous', 'Next >', 'Finish', and 'Cancel'.

Entity Data Model Wizard

Choose Your Data Connection

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

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.

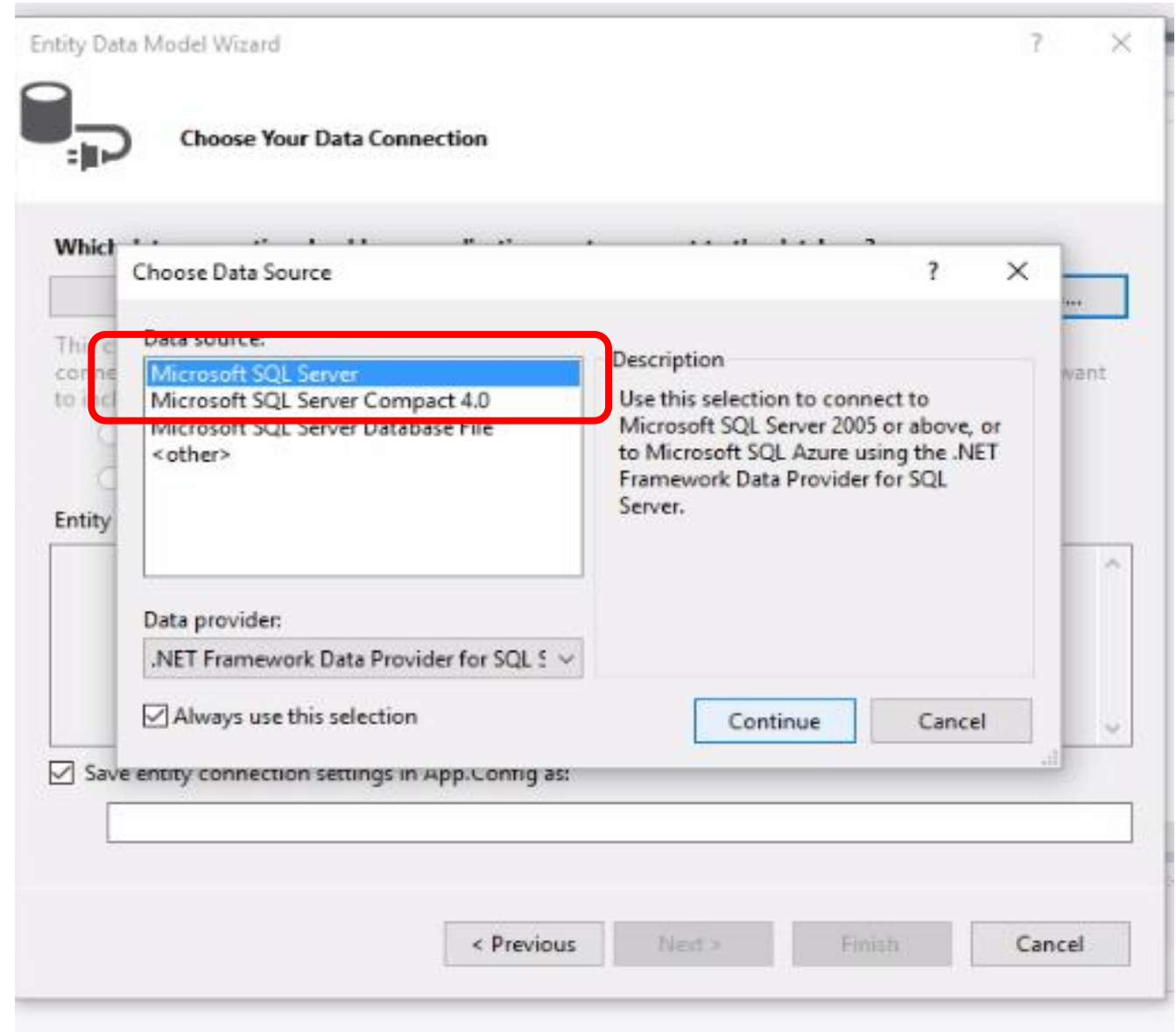
☐ Yes, include the sensitive data in the connection string.

Entity connection string:

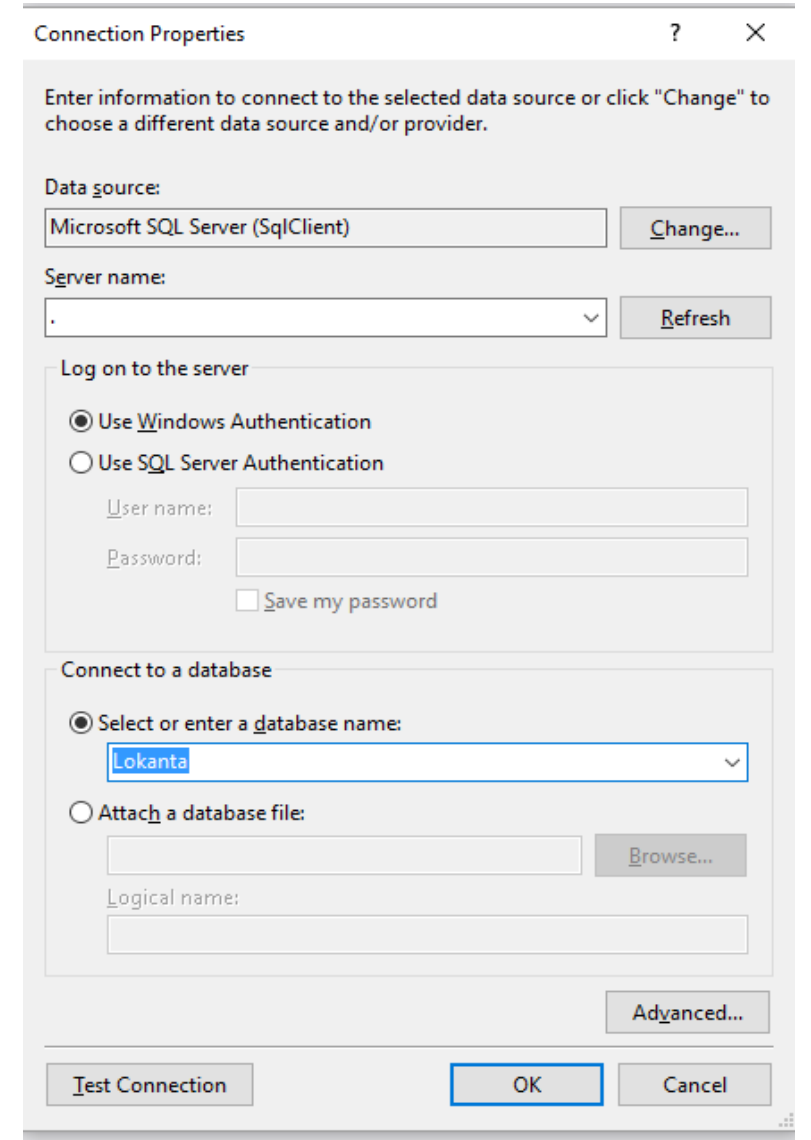
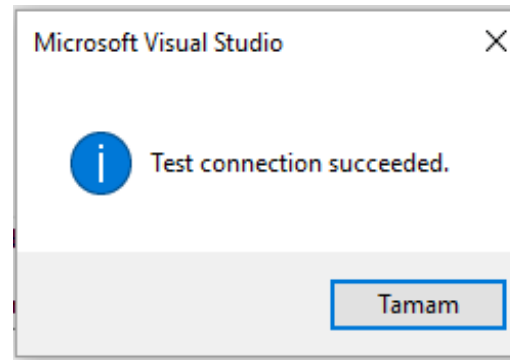
☒ Save entity connection settings in App.Config as:

< Previous Next > Finish Cancel

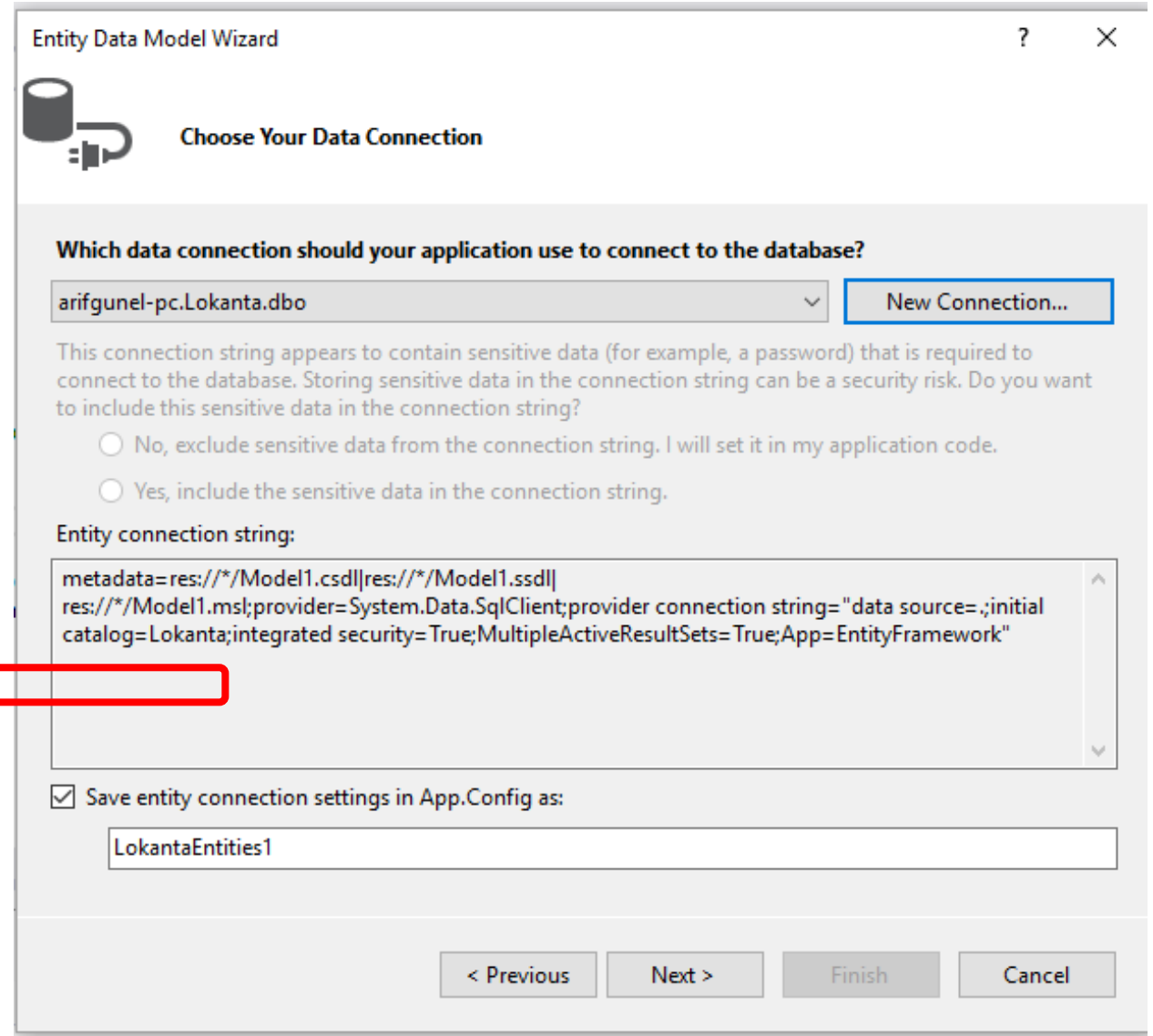
- *Microsoft SQL Server*



- Burada lokal bilgisayarda çalıştığımızdan Server Name kısmına “.” (lokal host) yazıyoruz. Uzak bir veri tabanı olsaydı ip adresini yazacaktık. SqlExpress’ lerde SQLEXPRESS uzantılı yazıyoruz.
- **Windows Authentication** kalabilir uzak bilgisayar olsaydı SQL Server Authentication deyip user name password verecektik.
- **“Connect to a database name”** kısmından model oluşturmak istediğimiz veri tabanımızı seçiyoruz
- **Test connection** diyoruz bağlantı başarılı derse Ok diyoruz



- **“Save entity connection settings in App.Config as:” bağlantı cümleciğini config dosyasına kaydet seçeneği işaretlememiz gerekiyor.**



The image shows the 'Entity Data Model Wizard' dialog box, specifically the 'Choose Your Data Connection' step. The window has a title bar with a question mark and a close button. Below the title bar is a header area with a database icon and the text 'Choose Your Data Connection'. The main content area contains the question 'Which data connection should your application use to connect to the database?'. Below this is a dropdown menu showing 'arifgunel-pc.Lokanta.dbo' and a 'New Connection...' button. A text block explains that the connection string may contain sensitive data and asks if the user wants to include it. There are two radio buttons: 'No, exclude sensitive data from the connection string. I will set it in my application code.' (selected) and 'Yes, include the sensitive data in the connection string.' Below this is a text box labeled 'Entity connection string:' containing the following text: 'metadata=res://*/Model1.csdl|res://*/Model1.ssdl|res://*/Model1.msl;provider=System.Data.SqlClient;provider connection string="data source=.;initial catalog=Lokanta;integrated security=True;MultipleActiveResultSets=True;App=EntityFramework"'. A red rectangle highlights the 'Save entity connection settings in App.Config as:' checkbox, which is checked. Below the checkbox is a text box containing 'LokantaEntities1'. At the bottom of the dialog are four buttons: '< Previous', 'Next >', 'Finish', and 'Cancel'.

Entity Data Model Wizard

Choose Your Data Connection

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

arifgunel-pc.Lokanta.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.

Entity connection string:

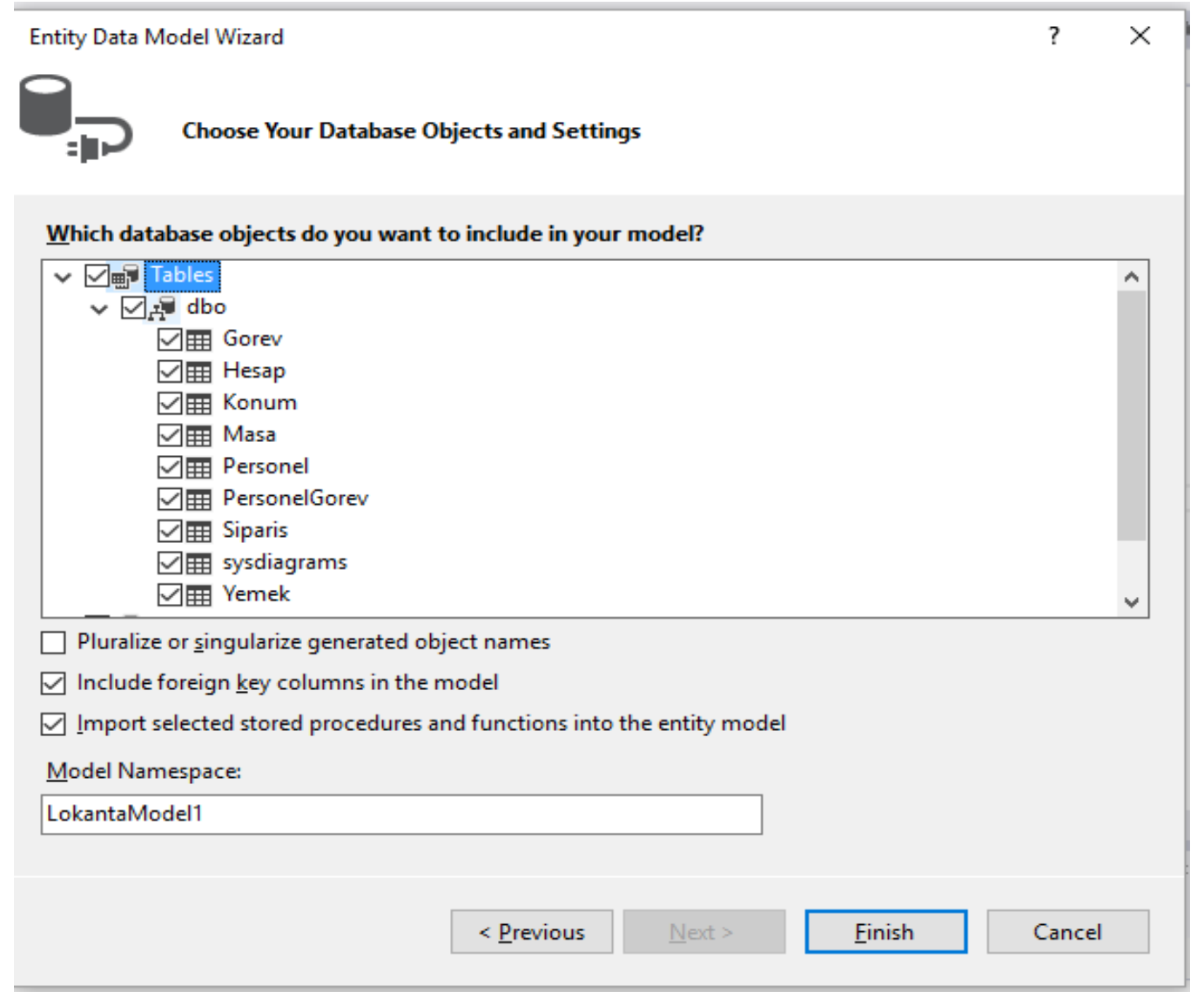
metadata=res://*/Model1.csdl|res://*/Model1.ssdl|res://*/Model1.msl;provider=System.Data.SqlClient;provider connection string="data source=.;initial catalog=Lokanta;integrated security=True;MultipleActiveResultSets=True;App=EntityFramework"

☒ Save entity connection settings in App.Config as:

LokantaEntities1

< Previous Next > Finish Cancel

- Gelen pencerede hangi tablolardan modele dahil edilmesini soruyor, tüm tabloları seçelim. (sys.diyagramı seçmeyelim) “**Finish**” diyoruz



The image shows the 'Entity Data Model Wizard' window, specifically the 'Choose Your Database Objects and Settings' step. The window has a title bar with a question mark and a close button. Below the title bar is a database icon and the text 'Choose Your Database Objects and Settings'. The main area is titled 'Which database objects do you want to include in your model?'. It contains a tree view with 'Tables' selected, showing a list of tables under the 'dbo' schema: Gorev, Hesap, Konum, Masa, Personel, PersonelGorev, Siparis, sysdiagrams, and Yemek. All tables are checked. Below the tree view are three checkboxes: 'Pluralize or singularize generated object names' (unchecked), 'Include foreign key columns in the model' (checked), and 'Import selected stored procedures and functions into the entity model' (checked). At the bottom, there is a 'Model Namespace' field with the text 'LokantaModel1'. The bottom right corner has four buttons: '< Previous', 'Next >', 'Finish' (highlighted with a blue border), and 'Cancel'.

Entity Data Model Wizard

Choose Your Database Objects and Settings

Which database objects do you want to include in your model?

Tables

dbo

- Gorev
- Hesap
- Konum
- Masa
- Personel
- PersonelGorev
- Siparis
- sysdiagrams
- Yemek

☐ Pluralize or singularize generated object names

☒ Include foreign key columns in the model

☒ Import selected stored procedures and functions into the entity model

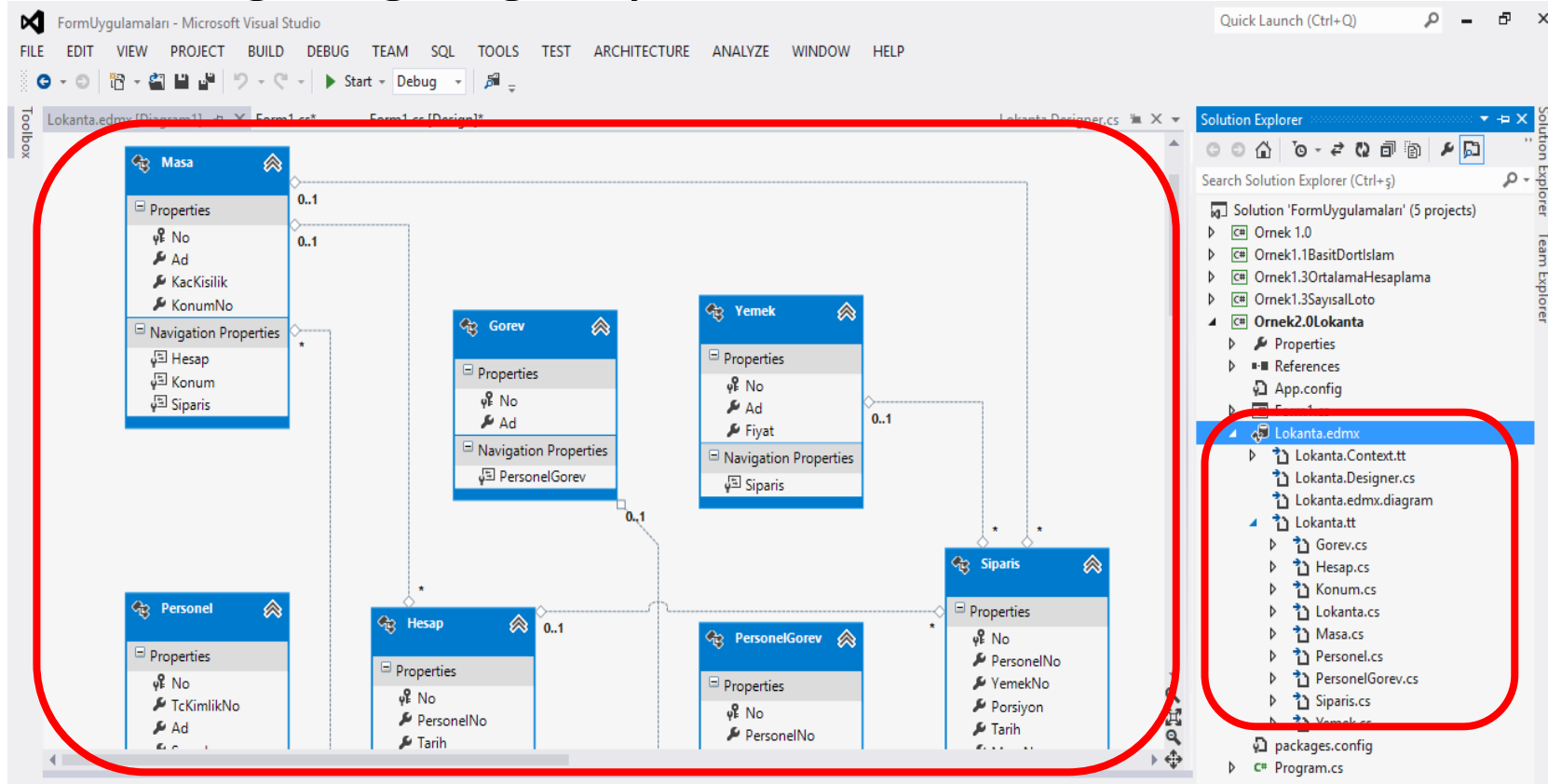
Model Namespace:

LokantaModel1

< Previous Next > Finish Cancel

Bundan sonra data model projemize yüklenecektir.

- *Baktığımızda tablolar birer .cs dosya şekline tabloların arasındaki ilişkilerin bile geldiğini görüyoruz.*



- *Diyagram oluştuktan sonra dosyalara derleme işlemini gerçekleştirelim **ctrl + shift+ B** ile ayarlarımız kayıt edilsin.*
- *Bundan sonra yine asıl işimiz form kısmı ile olacak. **References** dosyasına baktığımızda **Entity Framework** dosyası geldiğini görüyoruz.*

- **App.config** dosyasına (config dosyamız ayarlar dosyamız) bir takım eklemeler yapıldığını görürüz burada;
- **connectionString** bilgisi önemlidir, bağlanılacak veri tabanına ait bilgileri saklar Entity Framework buradaki bilgileri kullanarak bağlantı sağlıyor.
- **data source** sonrası yazan veri kaynağı yani veri tabanının nerede olduğu bilgisi, “.”(nokta) lokal host olduğunu verir dışarıdan bir sunucu olsaydı burada ip adresi yazacaktı.
- **Initial catalog** bağlanılan veri tabanının ismini veriyor.
- **integrated security** Windows Authentication ile bağlandığımızdan her hangi bir parola kullanmadığımızı gösteriyor.False yaparsak user ve password gerekecek.Uzak bir bilgisayara bağlanırken gerekli.

App.config dosyası:

```
<connectionStrings>
```

```
  <add name="LokantaEntities"  
connectionString="metadata=res://*/Lokanta.csdl|res://*/L  
okanta.ssdl|res://*/Lokanta.msl;provider=System.Data.SqlC  
lient;provider connection string=&quot;
```

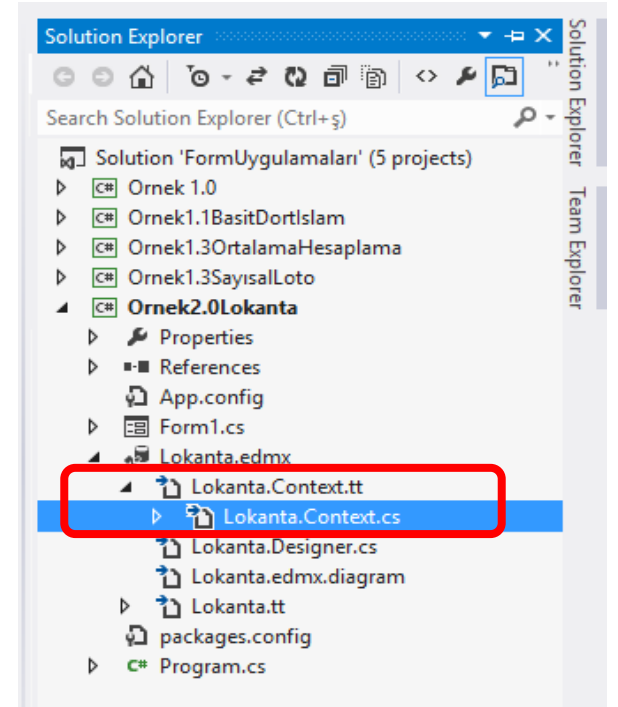
```
data source=.;initial catalog=Lokanta;integrated  
security=True;
```

```
MultipleActiveResultSets=True;App=EntityFramework&quot;;"  
providerName="System.Data.EntityClient" />
```

```
</connectionStrings>
```

Button Click olayına geri dönelim;

- Butona bastığımızda yemeklerin kaydedilmesini istiyorduk.
- Entity Framework ile veri tabanına kayıt eklemek istiyorsak yapacağımız ilk şey model oluşturmaktır.
- Bunun için **LokantaEntities** sınıfını kullanıyoruz bu sınıftan nesne türetiyoruz. Bunu Lokanta.Context dosyasında görebiliriz



- *Veri tabanının bir modelini oluşturalım. Bu sayede veri tabanının tüm tablolarının kullanımını sağlıyoruz.*

```
LokantaEntities model = new LokantaEntities();
```

- *Veri tabanındaki yemek tablosu c# tarafında birer nesne olarak oluştuğundan var olan yemek sınıfından bir nesne türeteceğiz;*

```
Yemek yeniYemek = new Yemek();
```

- *Tablodaki alanlara hangi kısımlardaki verilerin aktarılacağını gösteriyoruz.*
- *Yeniyemek. Dediğimizde veri tabanındaki yemek tablosunun tüm alanlarına erişebiliyoruz.*

```
yeniYemek.Ad = tbYemekAdi.Text;
```

```
yeniYemek.Fiyat = Convert.ToDecimal(tbYemekFiyati.Text);
```

- *Modelimize yeni nesnemiz olan yeniYemek nesnemizi ADD metodu kullanarak ekliyoruz;*

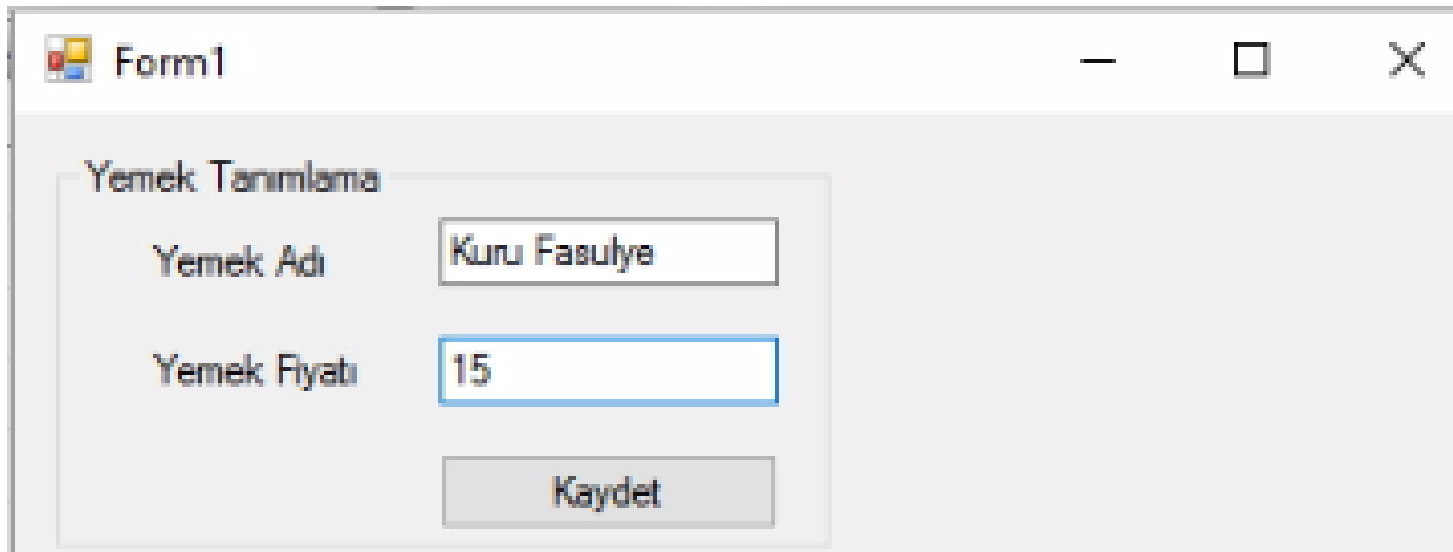
```
model.Yemek.Add(yeniYemek);
```

- Yapılan değişiklikler bellekte tutulmaktadır, bunları kalıcı olarak veri tabanına kaydetmek için;

`model.SaveChanges();`

- *SaveChanges metodu çağrılmaz ise veri tabanına kayıt yapılmaz*

- ***Çalışmasına bakalım;***



Form1

Yemek Tanımlama

Yemek Adı: Kuru Fasulye

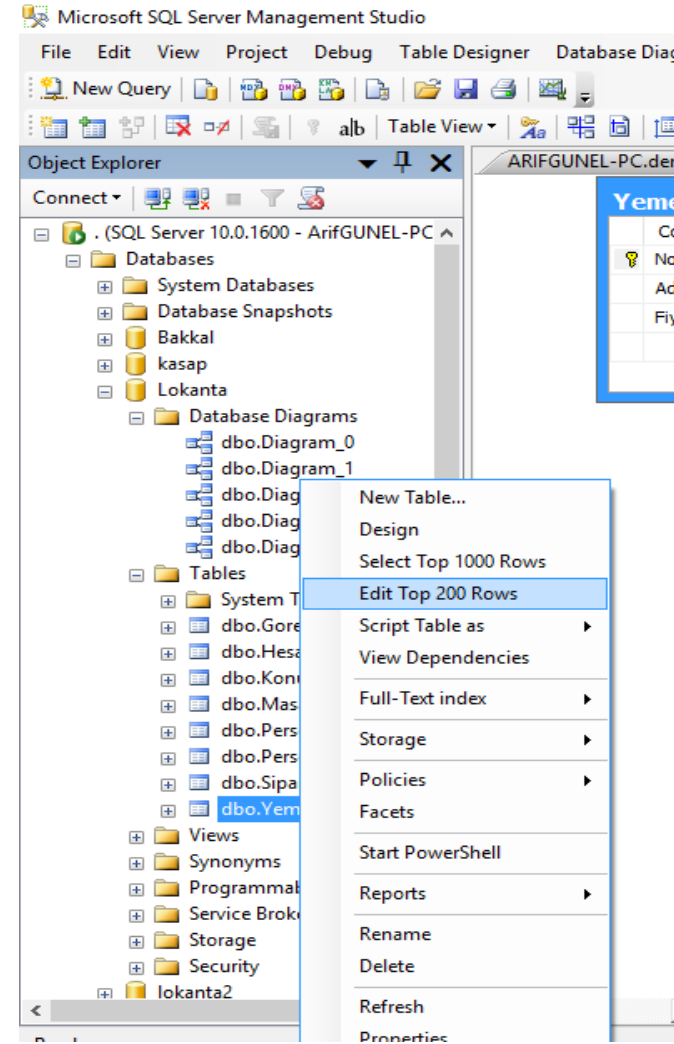
Yemek Fiyatı: 15

Kaydet

ARIFGUNEL-PC.Lokanta - dbo.Yemek			
	No	Ad	Fiyat
►	1	Taze Fasulye	7,0000
	2	Adana	9,0000
	3	Urfa	9,0000
	4	İskender	12,0000
	5	Beyti Kebap	12,0000
	6	Patıcanlı Kebap	12,0000
	7	Köfte	10,0000
	8	Karışık Izgara	15,0000
	9	Bilecik Usulü	19,0000
	10	Künefe	12,0000
	11	Kuru Fasulye	15,0000
*	NULL	NULL	NULL

- Şimdi veri tabanımıza gidelim ve Yemek tablomuzu açalım var olan kayır sayısına bakalım

ARIFGUNEL-PC.Lokanta - dbo.Yemek			
	No	Ad	Fiyat
►	1	Taze Fasulye	7,0000
	2	Adana	9,0000
	3	Urfa	9,0000
	4	İskender	12,0000
	5	Beyti Kebap	12,0000
	6	Patlıcanlı Kebap	12,0000
	7	Köfte	10,0000
	8	Karışık Izgara	15,0000
	9	Bilecik Usulü	19,0000
	10	Künefe	12,0000
	11	Kuru Fasulye	15,0000
*	NULL	NULL	NULL

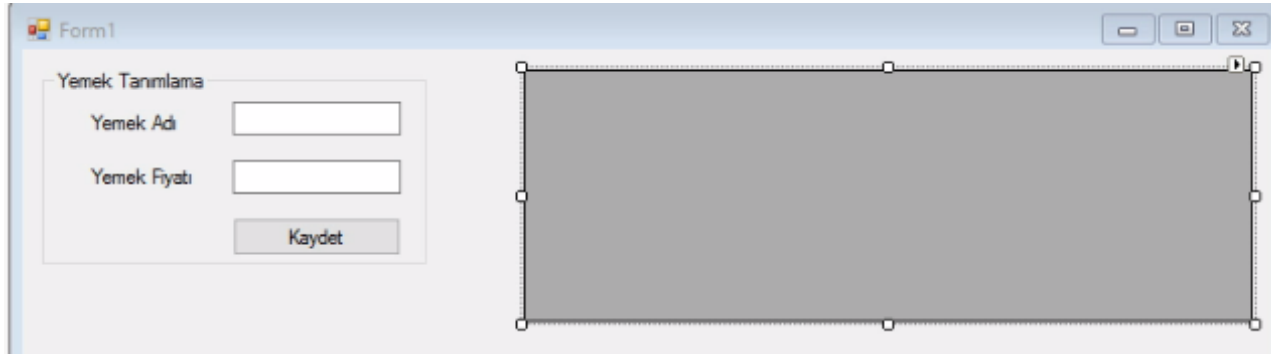


Sonra form üzerinde yeni yemek girişi yapalım tekrar veri tabanındaki tabloya baktığımızda yemeğin eklenmiş olduğunu görürüz.

- *Bu işlemlerin sonunda veri tabanına çok kolay bir şekilde var olan nesneler üzerinden bağlantı sağlayabiliyoruz.*
- *Entity Framework bize sağladığı, yemek ekleyeceksek yemek nesnesi üzerinden bunu yapmak. Veri tabanına veri eklemek.*

Ornek 2.1 Linq Sorguları(tablodan sorgu)

- Şimdi link sorguları kullanarak formumuz yüklenirken var olan yemekleri bir form kontrolünde yüklenmesine bakalım.
- 1 adet DataGridView(nem dgvYemekListe) (veri tabanından geçilen bir listeyi görüntülemek için kullanılıyor.)



GridView kontrolü database' den çekilen verilerin veya geçici olarak oluşturulan dataların saklandığı verilerin gösterilmesine yarayan bir Windows kontrolüdür. Tablo halinde verilerin görülmesini sağlar.

- *Form yüklenirken var olan kayıtların dgvYemekListe isimli DataGridView'e yüklenmesini istiyoruz.*
- *Form'a çift tık ile **FormLoad** olayı geliyor.*

```
(private void Form1_Load(object sender, EventArgs e)
```

- *Veri tabanından yükleme yaparken yine model oluşturacağız;*

```
LokantaEntities model = new LokantaEntities();
```

- *Var türünden bir değişken tanımlıyoruz.*

Var türünden değişken demek, değişkenin tipinin karşısındaki veri tipine göre belirleniyor. Yani var' ın karşısına ne yazıyorsam değişken tipi ona dönüşüyor.

Bunun bir yemek listesi olmasını istediğimiz için Linq sorgusu yazıyoruz.

Linq sorguları SQL sorgularına benzer;

Yemek tablosunu sorguluyorum, yemek tablosu k harfini temsil ediyor.

```
var yemekListe = (from k in model.Yemek // yemek tablosu sorgulama k yemek  
tablosunu temsil ediyor.
```

```
select k).ToList(); // Select k demek tüm alanlar getir  
anlamında,ToList listeye dönüştürüyor.
```

- *Data source özelliği ile veri kaynağı olarak YemekListe isimli listeyi belirtiyoruz.*

```
dgvYemekListe.DataSource = YemekListe;
```

Çalışmasına bakalım

Form1

Yemek Tanımlama

Yemek Adı

Yemek Fiyatı

Kaydet

No	Ad	Fiyat	Siparis
1	Taze Fasulye	7,0000	
2	Adana	9,0000	
3	Urfa	9,0000	
4	İskender	12,0000	
5	Beyti Kebap	12,0000	

Ekleme: Gelen tablonun sadece belli alalarının görölmesi istendiğinde;

- Bazı uygulamalarda veri tabanındaki bir tablonun tüm alanlarının gelmesi istenmez sadece belli alanların görölmesi istenebilir.
- Bu durumda `linq` sorgusunda değışiklik yapıyoruz.
- Daha önce yazdığımız “`select k`” kavramı tüm tablonun gelmesini sağlıyordu. Biz bunu “`select new`” diyerek “`new`” anahtar kelimesiyle istediğimiz alanları yazacağız.

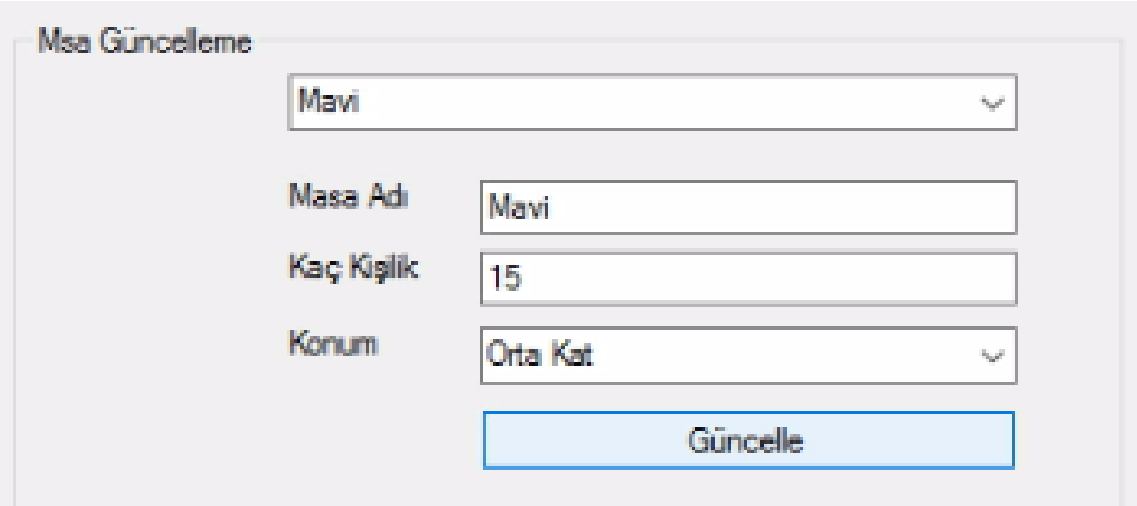
```
var liste2 = (from k in model.Yemek
              select new
              {
                  YemekAdı = k.Ad,
                  YemekFiyatı = k.Fiyat
              }).ToList();
```

YemekAdı	YemekFiyatı
Taze Fasulye	7,0000
Adana	9,0000
Urfa	9,0000
Iskender	12,0000
Beyti Kebap	12,0000
Patlıcanlı Kebap	12,0000

- Data source'yi değıştirelim.Yeni yazdığımız Liste2 veri kaynağı olarak gösterelim.
- ```
dgvYemekListe.DataSource = liste2;
```

## ***Form Güncelleme-Veri tabanındaki Var Olan Kayıta Değişiklik Yapma***

- *Seçmiş olduğumuz bir masanın adını güncelleme işlemi yapalım. Sıfırdan bir masa eklemeyeceğiz var olan masaların bilgilerinde değişiklik-güncelleme yapacağız.*
- *ComboBox ile **masayı seçip** textBox'lar ile **adının, kaç kişilik olduğunu ve konumunu güncelleyeceğiz.***



Masa Güncelleme

Mavi

Masa Adı Masa

Kaç Kişilik 15

Konum Orta Kat

Güncelle

- **Kontrolleri ekleyelim;**
- *1 adet GroupBox (Text Masa Güncelle)(Var olan masalar burada listelenecek, seçilen masanın özellikleri aly kısımda gelecek.)*
- *1 adet ComboBox(name cbMasaListe): Bu comboBox bize veri tabanından o an var olan tüm masa kayıtlarını listelenmiş olarak getirecek ve hangi masada değişiklik-güncelleme yapmak istiyorsak gelen masalardan o masayı seçeceğiz.Bu yüzden bu comboBox, form yüklenirken var olan masalar gelecek.*

- *Seçilen masanın bilgileri(ad, kaç kişilik ve konum) alt kısımdaki kontrollere yüklenecek.*
- *2 adet TextBox(name tbMasaAdi2,tbKacKisilik2)*
- *1 adet ComboBox(name **cbKonum2**)(Konum bilgisi var olan konumlardan geleceğinden dolayı comboBox ile yapmak zorundayız)*



## ***FormLoad olayına gidelim;***

- *Masa Liste isimli ComboBox'a masa isimlerini yükleyelim; Linq sorgusu ile yapalım.*
- *Tüm Masaları sorgulamak istiyorsak **Entity Framework sorgusu** olan **ToList** linq sorgusu ile bu şekilde yapabiliriz.*

```
var MasaListe = model.Masa.ToList();
```

- *Dönen listedeki Ad alanının DisplayMember görünen alan olarak gösterelim;*

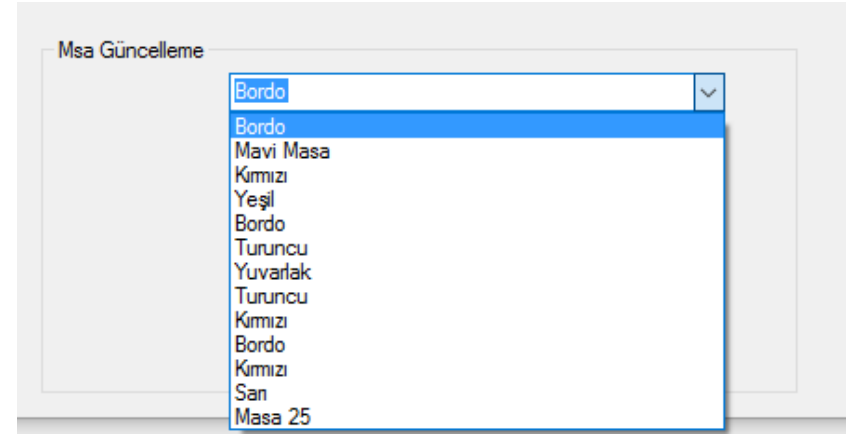
```
cbMasaListe.DisplayMember = "Ad";
```

- *Dönen listedeki No alanının ValueMember görünen alan olarak gösterelim;*

```
cbMasaListe.ValueMember = "No";
```

- *Veri kaynağı da yukarıda çağırdığımız MasaListe olacak. Veri tabanından çektiğimiz listeyi buraya atacağız.*

```
cbMasaListe.DataSource = MasaListe;
```



- *İkinci olarak ComboBox' a konumları yükleyelim. Form yüklenirken gelsin. Bu yüzden forma çift tıklayıp yazalım.*
- *Burada daha önce veri tabanından konumları bir liste halinde çekip başka bir comboBox' a yüklemiştik aynı değişkeni şimdiki comboBox'ımıza veri kaynağı olarak gösterelim;*
- *Not: Bir önceki örneğe devam ediyorsak metin kutusu içerisindeki kodları yazmamıza gerek yok;*

```
LokantaEntities model = new LokantaEntities();
var konumlar = (from k in model.Konum.ToList()
 select new
 {
 No = k.No,
 Ad = k.Ad
 }).ToList();
```

```
cbKonum2.DisplayMember = "Ad";
```

```
cbKonum2.ValueMember = "No";
```

```
cbKonum2.DataSource = konumlar;
```

```
var liste2 = (from k in model.Yemek
 select new
 {
 No=k.No,
 YemekAdı = k.Ad,
 YemekFiyatı = k.Fiyat
 }).ToList();
var konumlar = (from k in model.Konum.ToList()
 select new
 {
 No = k.No,
 Ad = k.Ad
 }).ToList();
```

```
cbKonum.DisplayMember = "Ad";
cbKonum.ValueMember = "No";
cbKonum.DataSource = konumlar;
```

```
cbKonum2.DisplayMember = "Ad";
cbKonum2.ValueMember = "No";
cbKonum2.DataSource = konumlar;
```

```
dgvYemekListe.DataSource = liste2;
```

- Formumuz yüklenirken ilgili nesnelere veri tabanından bilgiler çekilip yüklenecek. Çalıştırıp görelim;

Msa Güncelleme

Bordo

Masa Adı Bordo

Kaç Kişilik 10

Konum Bahçe1

Bahçe1  
Giris Katı  
Vip  
Orta Kat

- Şimdiye kadar yaptıklarımızı özetleyecek olursak; Masa listesi ve konumların listesi ilgili comboBoxlara yüklendi.

Form1

Yemek Tanımlama

Yemek Adı

Yemek Fiyatı

Kaydet

| No | YemekAdı     | YemekFiyatı |
|----|--------------|-------------|
| 1  | Taze Fasulye | 7,0000      |
| 2  | Adana        | 9,0000      |
| 3  | Urfa         | 9,0000      |
| 4  | İskender     | 12,0000     |

Masa Tanımlama

Masa Adı

Kaç Kişilik

Konum Bahçe1

Kaydet

Msa Güncelleme

Bordo

Masa Adı

Kaç Kişilik

Konum Bahçe1

## ***Devam edelim;***

- *Şimdi güncelleme yapmak istediğim masanın bilgileri görmek istiyorum.*

*Bunun için bilgileri değiştirilmek istenen masa comboBox' tan seçilince; ilgili 2 adet textbox ve 1 adet comboBox' a seçilen masanın bilgilerinin gelmesini yapalım.*

- *Bunu ComboBox'ın **SelectedIndexChanged** olayı ile yapıyoruz. Bu comboBox'ın varsayılan olayıdır.*
- *ComboBox'ta bir değer değiştiğinde işlemin tetiklenmesi için.*
- *cbMasaListe isimli ComboBox'a çift tıklıyoruz*

```
private void cbMasaListe_SelectedIndexChanged(object sender, EventArgs e)
```

- *Not:Sender MasaListe isimli combobox'ı temsil ediyor, e ise olayı tetikleyen argümanları temsil ediyor*

- İlk olarak veri tabanının Entity modelini oluşturuyoruz.

```
LokantaEntities model = new LokantaEntities();
```

- ComboBoxta masayı seçtiğimizde seçilen masanın No alanının bilgisini almamız gerekiyor.
- Bu yüzden masanın “no” değerini alıyoruz. Bunu bize **SelectedValue** özelliği veriyor. ComboBoxta o an seçilen masanın no bilgisini veriyor. Text bilgisi değil arka tarafta kullanılacak olan no bilgisini veriyor.

```
string secilenMasaNo = cbMasaListe.SelectedValue.ToString();
```

- *Seçilen MasaNo değeri string türünden bir değer bunu sorguda kullanacağımız için int(tam sayı) türüne dönüştürelim.*

```
int MasaNo = Convert.ToInt32(secilenMasaNo);
```

- *Seçilen masayı sorgulamak için linq sorgusu yerine lambda ifadesini(=>) kullanalım. FirstOrDefault diye bir metot var p masayı temsil ediyor veri tabanındaki masa tablosunun No alanı ile int olarak tanımladığımız değişkeni eşitliyoruz.*

```
var SeciliMasa = model.Masa.FirstOrDefault(p => p.No == MasaNo);
```

- *Şimdi işlem yapılmak(güncellemek-değiştirmek) istenen Masa nın bilgilerinin Textboxlara ve ComboBox'lara atayalım;*

```
tbKacKislik2.Text = SeciliMasa.KacKisilik.ToString();
```

*(veri tabanındaki tablodaki bu alan int türünden olduğu içinToString ile dönüştürerek textBox'a atıyoruz.)*

```
tbMasaAd2.Text = SeciliMasa.Ad;
```

- *ComboBox' a değer atayalım*

```
cbKonum2.SelectedIndex = cbKonum2.FindString(SeciliMasa.Konum.Ad)
```



***Çalıştırıp bakalım;***

Msa Güncelleme

Bordo

Masa Adı Bordo

Kaç Kişilik 10

Konum Bahçe1

## **Seçilen değeri veri tabanının yazmak için button koyalım;**

- 1 adet Button ekleyelim.(name **btnMasaGuncelle**, text Güncelle)

*Butona çift tıklayıp kodları yazalım.*

- *Yine veri tabanının modelini oluşturarak başlayalım.*

```
LokantaEntities model = new LokantaEntities();
```

- *Seçili Masa Numarasına burada da ihtiyacımız var. ComboBox'dan seçili olanı alıp tam sayıya dönüştürüyoruz.*

```
int MasaNo = Convert.ToInt32(cbMasaListe.SelectedValue);
```

- *Güncel bilgileri veri tabanına yazmak için FirstOrDefault metodunu kullanarak değişkene atıyoruz.*

```
var seciliMasa = model.Masa.FirstOrDefault(p=>p.No==MasaNo);
```

- *Değişken üzerinden textBox ve comboBox'lardaki verileri veri tabanına gönderiyoruz*

```
seciliMasa.Ad = tbMasaAd2.Text;
```

```
seciliMasa.KacKisilik = Convert.ToInt32(tbKacKislik2.Text);
```

```
seciliMasa.KonumNo = Convert.ToInt32(cbKonum2.SelectedValue);
```

- *Değişiklikleri veri tabanına kalıcı olarak kaydediyoruz.*

```
model.SaveChanges();
```

***Çalıştırıp görelim;***

Form1

Yemek Tanımlama

Yemek Adı

Yemek Fiyatı

Kaydet

| No | YemekAdı     | YemekFiyatı |
|----|--------------|-------------|
| 1  | Taze Fasulye | 7.0000      |
| 2  | Adana        | 9.0000      |
| 3  | Urfa         | 9.0000      |
| 4  | Iskender     | 12.0000     |

Masa Tanımlama

Masa Adı

Kaç Kişilik

Konum

Kaydet

Masa Güncelleme

Masa Adı

Kaç Kişilik

Konum

Güncelle

## Özetleyelim:

- ComboBox'ın SelectedIndexChanged olayı üzerinden önce modeli oluşturduk, seçilen masanın nosunu form üzerinden elde ettik. Bu no ile veri tabanından sorgu ile gerçek tablodan veriyi nesne olarak elde ettik. (Var değişkeni üzerine FirstOrDefault ile bunu yapıyoruz)

*Daha sonra veri tabanından elde ettiğimiz alanları textboxlardan ve comboBox kontrollerden doldurduk.*

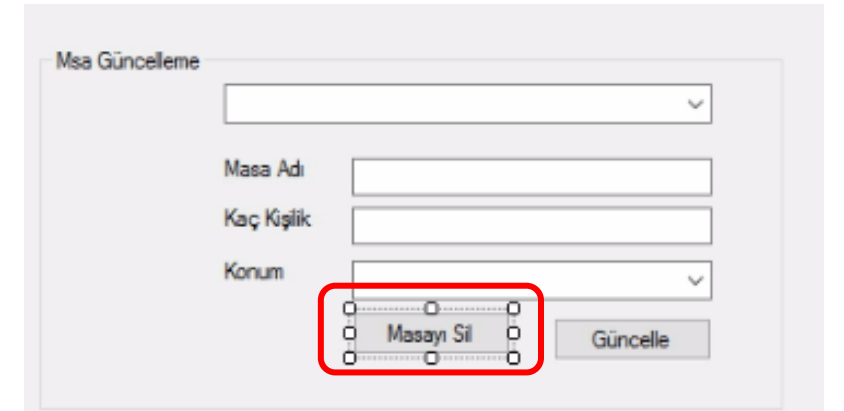
- ComboBox için SelectedIndex özelliği ile bilgiyi doldurduk. Bunu yaparken seçili olan masanın konumunun adı ne ise bunu bul ve onu seçili hale getir(FindString) dedik.
- FindString metodu ilgili comboBox'da ilgili textler(string'ler) üzerinde arama yapar bulunan textin indexini bize döndürür. Bizde `cbKonum2.SelectedIndex = cbKonum2.FindString(SeciliMasa.Konum.Ad)` bu kodda seçili masanın konum adı üzerinden arama yapıp eşleşeni buluyor ve comboBoxa yükler.

*Daha sonra güncelle butonu altında;*

- Veri tabanının modelini oluşturduktan sonra seçili masa nosunu elde ettik, bunun üzerinden textBox ve comboBox üzerindeki bilgileri aldık ve seciliMasa nesnesini güncelledik savechanges metodu ile de yapılan değişiklikler veri tabanına kaydedildi.

## Kayıt Silme İşlemi

- Veri tabanından kayıt silerken silinecek verinin **ilişkili kayıtlarının olmaması** gerekiyor. İlişki varsa veri tabanı seviyeli hata verecek ve silme işlemi gerçekleşmeyecektir. Veri tabanı seviyeli hata verir.
- Bu veri tabanındaki veri güvenliği kavramı ile ilgilidir. İlişki halinde bulunan tablolarda bir tablodaki veri diğer tabloda kullanılıyorsa silme işlemine izin vermeyerek bu bilgiler diğer tabloda var diye mesaj verir.
- 1 adet buton ekliyoruz(name btnMasaSil, text Masayı Sil)



The screenshot shows a web form titled "Masa Güncelleme". It contains three input fields: "Masa Adı", "Kaç Kişilik", and "Konum". Below these fields, there is a button labeled "Masayı Sil" which is highlighted by a red rectangular box. To the right of this button is another button labeled "Güncelle".

## **Butona çift tıklayalım;**

- *İlk olarak Entity modeli oluşturalım*

```
LokantaEntities model = new LokantaEntities();
```

- *Sonra seçili olan masa numarasına ihtiyaç var, bunu da Convert ile dönüştürerek ComboBox' tan elde ediyoruz*

```
int MasaNO = Convert.ToInt32(cbMasaListe.SelectedValue);
```

- *Masa nesnesi oluşturup, seçili olan masayı veri tabanından çekiyoruz. Bunu yaparken Linq sorgusu yerine Entity Framework'un bize hazır olarak sunduğu metotlar ile yapabiliyoruz.*
- **FirstOrDefault** isimli metot veri tabanından tek bir metodu nesne olarak çekip bize döndürüyor. Bu metoda bir şart belirliyoruz. "P" masayı temsil ediyor. Masa üzerinden **FirstOrDefault** metodunu çağırıyoruz. Tek bir kayıt döndürmek için  $p \Rightarrow p.No$  kalıbını kullanabiliriz. Bir nevi filtreleme yapıyoruz. Herhangi bir tabloyu sorgulayıp tek bir kayıt döndürmek için bu FirstOrDefault metodunu kullanabiliriz.

```
var SeciliMasa = model.Masa.FirstOrDefault(p=>p.No==MasaNO);
```

- *Şimdi seçtiğimiz kaydı veri tabanından sileceğiz. **Remove** metodu ile yapıyoruz. Remove metodu masa türünden bir nesne(yani seciliMasa) ile çalışır. Bizde bu nesneyi veri tabanından **FirstOrDefault** metodu ile elde ettiğimiz seçili Masa nesnesidir.*
- *Eğer ilişki kayıt durumunu kontrol etmeyecek olsak alt satırı yazıp birde veri tabanına kayıt satırı ile bitirecektik;*

```
model.Masa.Remove(SeciliMasa);
```

```
model.SaveChanges();
```

**Fakat** veri tabanında silinmek istenen kaydın **ilişki** durumu varsa silme işlemini yapamayacağından dolayı bizde bu durumu **if** yapısı ile bir şart oluşturalım.



- Eğer Masa tablosundan silinecek olan kaydın Sipariş tablosunda ve Hesap tablosunda kaydı varsa silme işlemini gerçekleştirmeyecek. (İki tabloda(sipariş ve hesap) da kaydın olup olmadığı kontrolünü count ile yapıyoruz, kaydı varsa 0' dan farklı bir sayı olacaktır)

Ve ekrana mesaj kutusu ile bilgilendirme yapalım. MessageBox sınıfını Show metodunu kullanalım.

```
if (SeciliMasa.Siparis.Count==0 && SeciliMasa.Hesap.Count==0)
{
 model.Masa.Remove(SeciliMasa);
 model.SaveChanges();
 MessageBox.Show("Masa kaydı başarıyla silindi", "", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
else
{
 MessageBox.Show("Masa Silinemez. Masaya bağlı sipariş veya hesap kaydı var.", "", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
```

***Çalıştırıp bakalım;***

Msa Güncelleme


Bordo

Masa Adı Bordo

Kaç Kişilik 10

Konum Bahçe1

Masayı Sil Güncelle

 Masa Silinemez. Masaya bağlı sipariş veya hesap kaydı var.

Tamam

Msa Güncelleme


Masa 25

Masa Adı Masa 25

Kaç Kişilik 15

Konum Giriş Katı

Masayı Sil Güncelle

 Masa kaydı başarıyla silindi

Tamam

### ***Kısaca özetleyelim;***

- Silme işleminde de veri tabanının modeli ve sonrasında silinecek olan masanın numarasına ihtiyaç var. Bu numarayı form kısmından alıp bu **no** üzerinden veri tabanının ilgili tablodan FirstOrDefault metodu ile veri tabanından sorgulayıp çekiyoruz ve bir masa nesnesi elde ediyoruz/oluşturuyoruz.*
- Elde ettiğimiz nesne ile ilgili sipariş ve hesap kaydının olup olmama durumuna göre(count) silme(kayıt yoksa count==0, remove metodu ile) ve ekrana bilgi yazdırma işlemlerinin **if** şartı ile yapıyoruz.*

```
private void btnMasaSil_Click(object sender, EventArgs e)
{
 LokantaEntities model = new LokantaEntities();
 int MasaNO = Convert.ToInt32(cbMasaListe.SelectedValue);
 var SeciliMasa = model.Masa.FirstOrDefault(p=>p.No==MasaNO);
 if (SeciliMasa.Siparis.Count==0 && SeciliMasa.Hesap.Count==0)
 {
 model.Masa.Remove(SeciliMasa);
 model.SaveChanges();
 MessageBox.Show("Masa kaydı başarıyla silindi", "", MessageBoxButtons.OK, MessageBoxIcon.Information);
 }
 else
 {
 MessageBox.Show("Masa Silinemez. Masaya bağlı sipariş veya hesap kaydı var.", "", MessageBoxButtons.OK, MessageBoxIcon.Error);
 }
}
```

