



ADO.NET Uygulamaları

Öğr.Gör.Erkan HÜRNALI

Uygulamamız

MainWindow

Lütfen bir il adı girin:

Adana	Akyurt
Adıyaman	Altındağ
Afyonkarahisar	Ayaş
Ağrı	Bala
Amasya	Beypazarı
Ankara	Çamlıdere
Antalya	Çankaya
Artvin	Çubuk
Aydın	Elmadağ
Aksaray	Etimesgut
Ardahan	Evren
	Gölbaşı
	Güdül

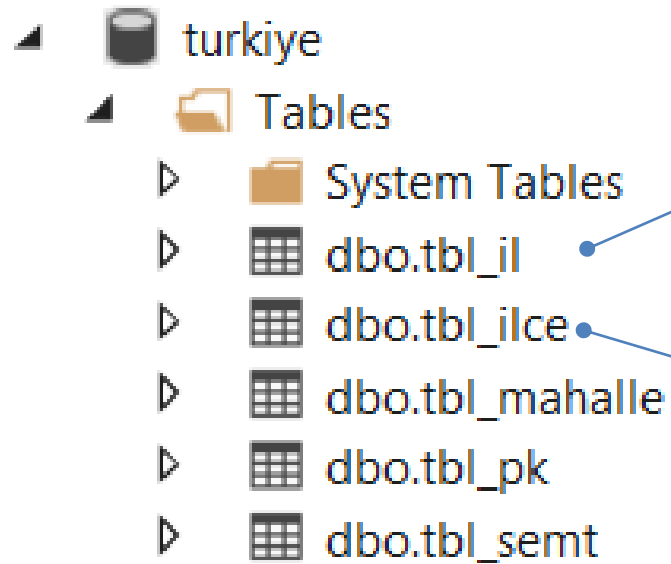
Lütfen bir il adı girin:

 →

Lütfen bir il adı girin:

Adana	Ankara
Adıyaman	Antalya
Afyonkarahisar	
Ağrı	
Amasya	
Ankara	
Antalya	
Artvin	
Aydın	
Aksaray	
Ardahan	

Veritabanı



	Name	Data Type	Allow Nulls	Default
PK	il_id	smallint	<input type="checkbox"/>	
	il_ad	nvarchar(16)	<input checked="" type="checkbox"/>	
			<input type="checkbox"/>	

	Name	Data Type	Allow Nulls	Default
PK	ilce_id	smallint	<input type="checkbox"/>	
	il_id	smallint	<input type="checkbox"/>	
	ilce_ad	nvarchar(32)	<input checked="" type="checkbox"/>	

Kodlama

```
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    // 1. aşama: Bağlantı kurulur
    string cs = @"Data Source=(localdb)\Projects;Initial Catalog=turkiye;Integrated ➤
        Security=True";
    baglanti = new SqlConnection(cs);
    //ya da baglanti.ConnectionString = cs;
}
```

Kodlama

```
private void tbArama_TextChanged(object sender, TextChangedEventArgs e)
{
    baglanti.Open(); //Bağlantılı (connected) veritabanı

    //2. Aşama: Çalıştırılacak komutlar oluşturulur (sql komutları)
    //SqlCommand komut = new SqlCommand("select * from tbl_il where il_ad like '"+p+"'", baglanti);
    SqlCommand komut = new SqlCommand("select * from tbl_il where il_ad like @ilAdi", baglanti);
    komut.Parameters.AddWithValue("@ilAdi", tbArama.Text + "%");// Sql Injection'dan korur

    //3.Aşama: Sorgu sonuçları değerlendirilir
    SqlDataReader dr = komut.ExecuteReader();
    lbIller.Items.Clear();
    while (dr.Read())
    {
        //string satir = string.Format("Plaka={0,2},\tŞehir={1}", dr[0], dr[1]);
        //lbIller.Items.Add(satir);

        lbIller.Items.Add(dr["il_ad"]);
    }
    baglanti.Close();
}
```

Kodlama

Adana
Adıyaman
Afyonkarahisar
Ağrı
Amasya
Ankara
Antalya
Artvin
Aydın
Aksaray
Ardahan



```
private void lbIller_SelectionChanged(object sender,
    SelectionChangedEventArgs e)
{
    if (lbIller.SelectedItem != null)
    {
        ilceleriSuz(lbIller.SelectedItem.ToString());
    }
    else
    {
        lbIlceler.Items.Clear();
    }
}
```



Akyurt
Altındağ
Ayaş
Bala
Beypazarı
Çamlıdere
Çankaya
Çubuk
Elmadağ
Etimesgut
Evren
Gölbaşı

Kodlama

```
private void ilceleriSuz(string p)
{
    baglanti.Open();

    2.Aşama

    3.Aşama

    baglanti.Close();
}
```

Sorgu cümleleri

Akyurt
Altındağ
Ayaş
Bala
Beypazarı
Çamlıdere
Çankaya
Çubuk
Elmadağ
Etimesgut
Evren
Gölbaşı

Kodlama

```
//2. Aşama: Çalıştırılacak komutlar oluşturulur (sql komutları)
//SqlCommand komut = new SqlCommand("select * from tbl_il where il_ad like '"+p
    +"%", baglanti);
//SqlCommand komut = new SqlCommand("select ilce_ad from tbl_il A inner join
    tbl_ilce B on A.il_id = B.il_id where il_ad = '"+p+"'" order by B.ilce_ad",
    baglanti); //bu yöntem SqlInjection'a açık olmuş olur

SqlCommand komut = new SqlCommand("select ilce_ad from tbl_il A inner join
    tbl_ilce B on A.il_id = B.il_id where il_ad = @ilAdi order by B.ilce_ad",
    baglanti);

//komut.Parameters.AddWithValue("@ilAdi", p+"%"); bu yanlış olur
komut.Parameters.AddWithValue("@ilAdi", p);
```


Kodlama

```
//3.Aşama: Kayıtlar Listelenir
SqlDataReader dr = komut.ExecuteReader();
lbIlceler.Items.Clear();
while (dr.Read())
{
    //string satir = string.Format("Plaka={0,2},\tŞehir={1}", dr[0], dr[1]);
    //lbIller.Items.Add(satir);

    lbIlceler.Items.Add(dr[0]);
}
```

Örnek Sorgular

```
select * from tbl_il  
select * from tbl_il where il_ad like 'A%'  
select * from tbl_il where il_ad like '_A%'
```

```
--select * from tbl_ilce where il_ad='Ankara'    !!!YANLIŞ!!!  
--select * from tbl_ilce where il_id='Ankara'    !!!YANLIŞ!!!  
select * from tbl_ilce where il_id='6'
```

```
select * from tbl_ilce where il_id=(select il_id from tbl_il where  
    il_ad='Ankara')
```

```
select * from tbl_il,tbl_ilce    -- tbl_il :81kayıt    tbl_ilce: 960kayıt  
select ilce_ad from tbl_il A inner join tbl_ilce B on A.il_id = B.il_id where  
    il_ad = 'Ankara' order by B.ilce_ad
```

Havada Kalmasin ☺



Teşekkürler...



*bize
katlandığınız
için!*

LinqToSqlClasses ve Entity Framework Uygulamaları*

Öğr.Gör.Erkan HÜRNALI

*Çeşitli kaynaklardan derlenmiştir.

LINQ To SQL Classes

Çeşitli yazılım geliştirme platformları tarafından üretilmiş ilişkisel nesne haritalaması (Object / Relational Mapping – O/RM) araçları mevcuttur, bunlardan birkaçı aşağıda verilmiştir;

- .NET platformu: NHibernate, **DLINQ**, EntityFramework
- JAVA platformu için : JPA, Hibernate, TopLink, iBatis
- Açık Kaynaklı (Open Source) : Business Logic Toolkit, Euss, LLBLGen

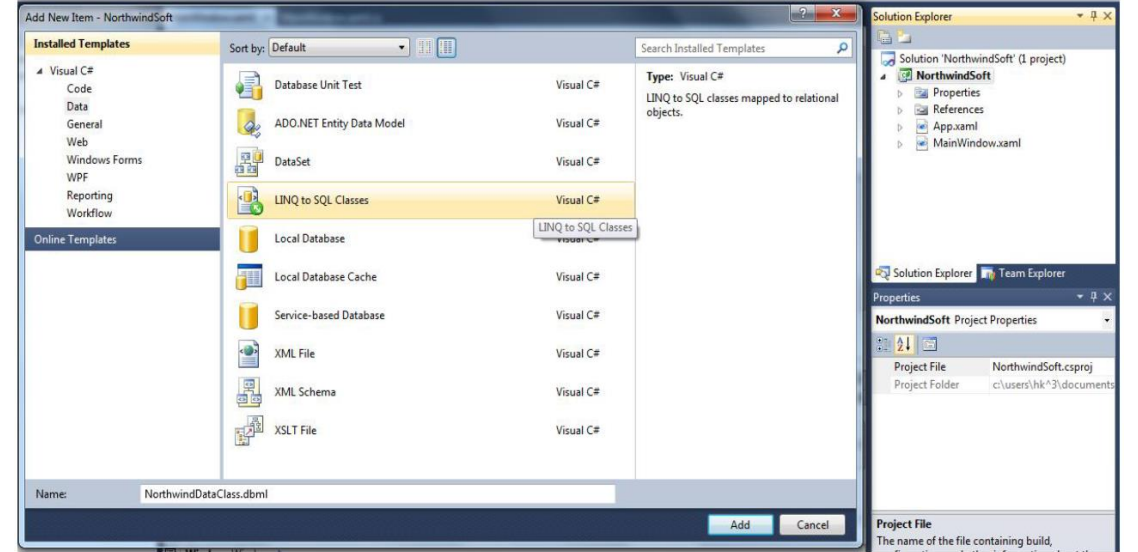
Bu öğrenme faaliyetinde .Net için DLINQ(Database Language Integrated Query) ORM aracını kullanacaksınız.

DLINQ ile Veri Baęlama

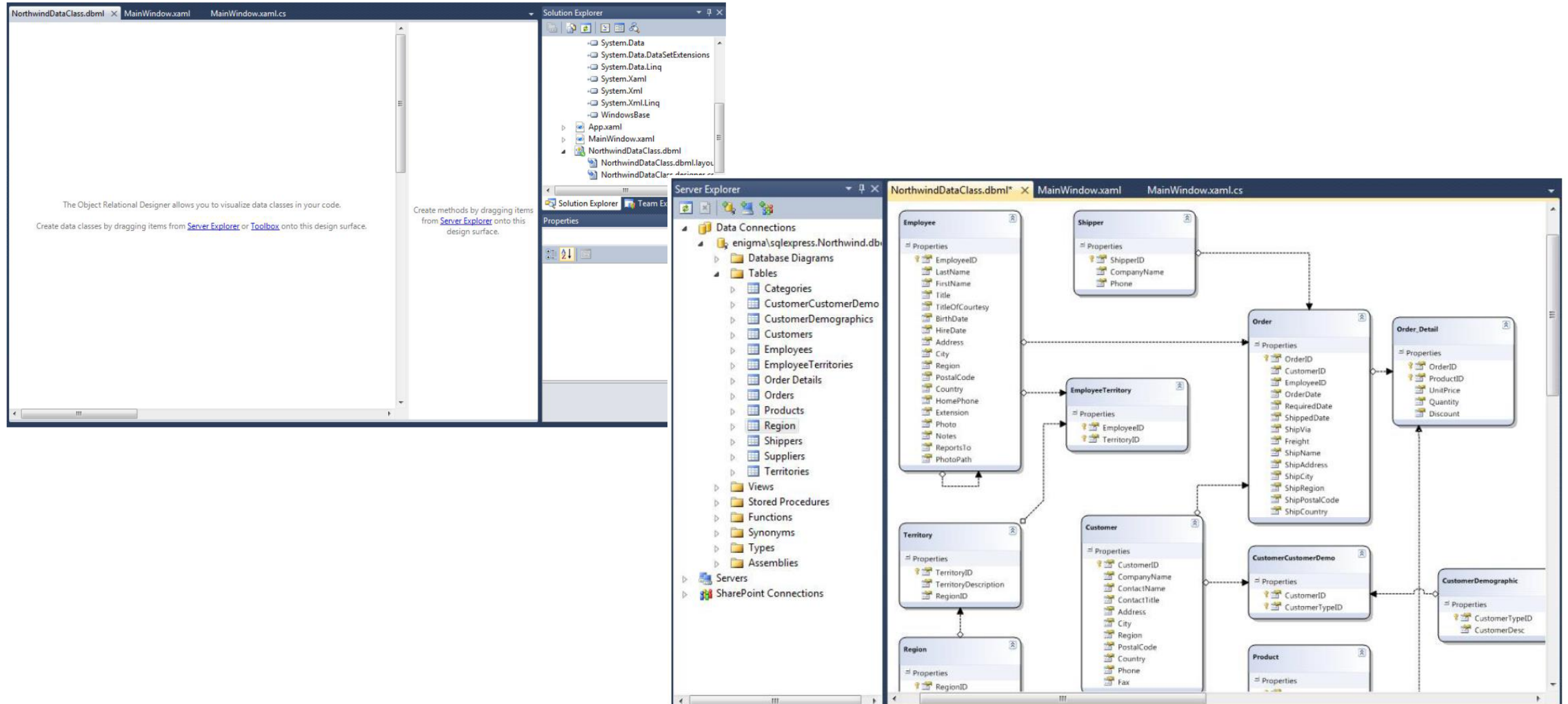
DLINQ size veri tabanı varlık nesnelerini oluřturmada, kullanmada, seçme, ekleme, silme, güncelleme işlemlerini birer metot olarak sunmada ve bütün bunları basit bir sihirbaz yardımı ile grafik ortamda hızlı olarak gerçekleřtirmenize imkan veren küçük ve orta ölçekli yazılım projeleriniz için bir O/RM tasarım aracı sunar.

DLINQ'nun veri erişim katmanı (data access layer) olarak kullanmak için ařağıdaki işlem adımlarını takip ediniz.

- Öncelikle Visual Studio ortamında yeni bir WPF uygulaması (WPF Application) oluřturunuz.
- Solution Explorer paneli içinde uygulama adı üzerinde fare saę tuř→Add→New Item seçimini yapınız.
- Ekranaya gelen “New Item” penceresinden “LINQ to SQL Classes” seçiniz. Name kısmına üretilecek .dbml uzantılı dosyaya vermek istediğiniz ismi yazınız. Bu uygulamada “NorthwindDataClasses” örnek ismi verilmiřtir. Ekle (Add) butonuna basınız.



DLINQ ile Veri Bağlama

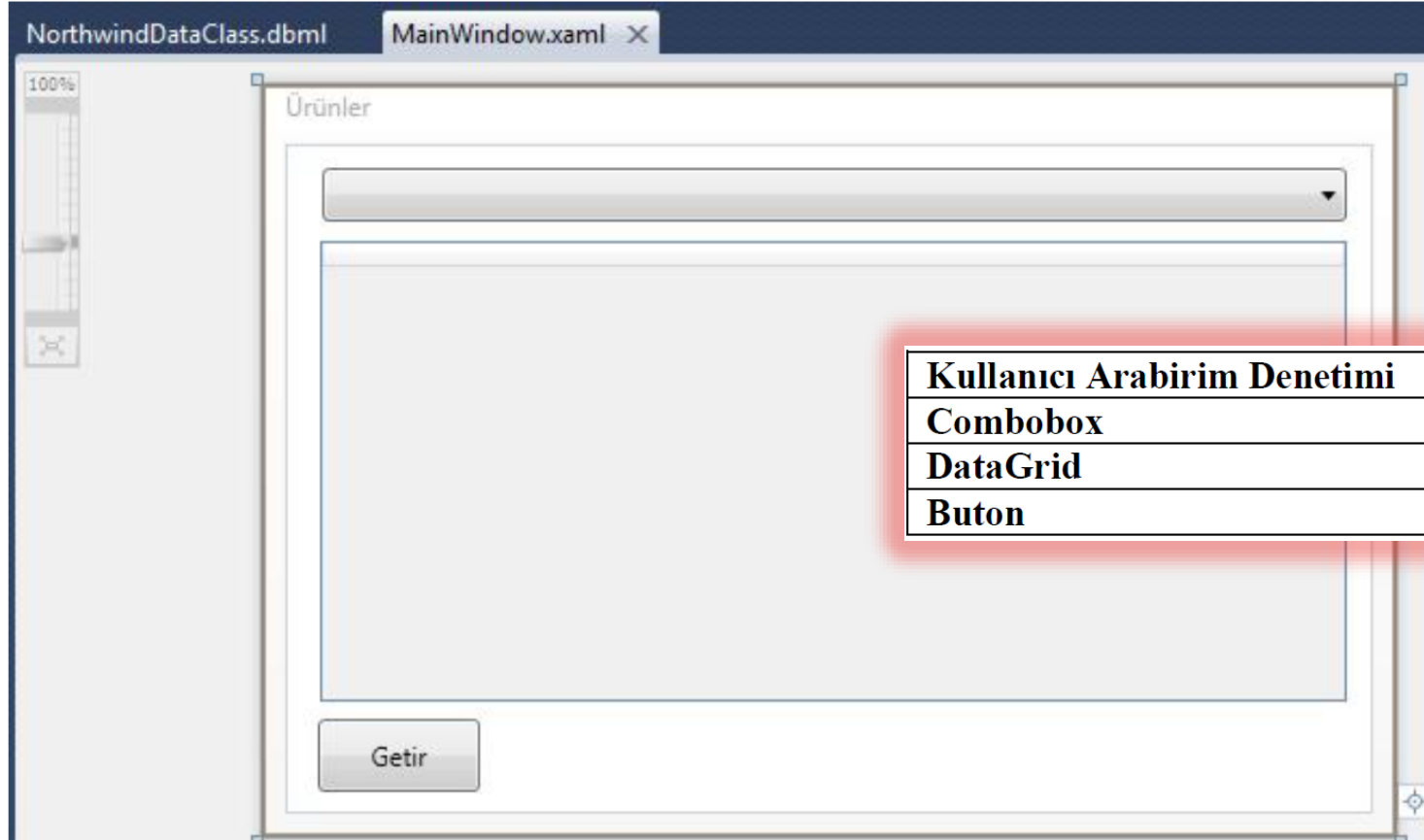


DLINQ ile Veri Bağlama

Sunum katmanında kullanılacak WPF kütüphanesinde bulunan kullanıcı arabirim denetimleri (Textbox, listbox, combobox, datagridview vs.) bu tip dinlemeleri gerçekleştirerek içeriği bakımından bağlı (data binding) bulundukları özelliklerin taşıdığı güncel bilgileri gösterebilirler. Aynı şekilde kullanıcı tarafından bu arabirim denetimlerinde yapılmış değişiklikler de nesneler tarafından dinlenerek gereken değişimi yapılarındaki alanlara uygulayabilirler.

WPF'in en güçlü taraflarından biri olarak görülen bu mekanizmaya veri bağlama (Data Binding) adı verilir ve tamamen "INotifyPropertyChanged" ve "INotifyPropertyChanged" arayüzlerine borçludur.

Tasarım Zamanı



Kullanıcı Arabirim Denetimi	İsim (Name) Özelliği
Combobox	cmbKategori
DataGrid	dgUrunler
Buton	btnGetir

XAML

```
<Window x:Class="NorthwindSoft.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Ürünler" Height="350" Width="525"
        xmlns:my="clr-namespace:NorthwindSoft">

    <Window.Resources>
        <my:PriceConverter x:Key="ConvertResource" />
    </Window.Resources>
    <Grid>
        <ComboBox ItemsSource="{Binding}" DisplayMemberPath="CategoryName"
        IsSynchronizedWithCurrentItem="True"
        SelectionChanged="cmbKategori_SelectionChanged" Height="25"
        HorizontalAlignment="Left" Name="cmbKategori" Width="474" />
```

```
<DataGrid AutoGenerateColumns="False" Height="214" HorizontalAlignment="Left"
Margin="16,45,0,0" Name="dgUrunler" VerticalAlignment="Top" Width="475">
    <DataGrid.Columns>
        <DataGridTextColumn Header="Ürün Adı" Binding="{Binding
Path=ProductName}" />
        <DataGridTextColumn Header="Fiyat" Binding="{Binding Path=UnitPrice,
Converter={StaticResource ConvertResource}}" />
        <DataGridTextColumn Header="Miktar" Binding="{Binding
Path=UnitsInStock}" />
        <DataGridTextColumn Header="Birimdeki Miktar" Binding="{Binding
Path=QuantityPerUnit}" />
        <DataGridCheckBoxColumn Header="Tükendi" Binding="{Binding
Path=Discounted}" />
    </DataGrid.Columns>
</DataGrid>
<Button Content="Getir" Height="34" HorizontalAlignment="Left"
Margin="15,267,0,0" Name="btnGetir" VerticalAlignment="Top" Width="75"
Click="btnGetir_Click" />
</Grid>
</Window>
```

XAML

Yukarıda bulunan xaml kodunu incelemeniz uygulamanızın nasıl çalıştığını anlamanız için önemlidir. İlk olarak “Combobox” tanımlamasını ele alalım;

```
<ComboBox ItemsSource="{Binding}" DisplayMemberPath="CategoryName"
IsSynchronizedWithCurrentItem="True"
SelectionChanged="cmbKategori_SelectionChanged" ...
```

“Combobox” bileşeninin “ItemsSource” özeliği ile bir veri kaynağına bağlandığını, gösterimi yapılacak alanın “DisplayMemberPath” ile kategori adı olacağı bilgisi verilmiştir.

Veri kaynağının ne olduğu bilgisi kod sayfasında verilecektir. “IsSynchronizedWithCurrentItem” özelliğinin “true” olarak ayarlanması “combobox” ile seçilen öğenin (SelectedValue) ile eşleşmesini garanti eder.

“SelectionChanged” olayının “cmbKategori_SelectionChanged” metoduna bağlanmıştır. Kategori seçimi her değiştiğinde gösterilecek ürünlerin güncellenmesi işlemini kod tarafında gerçekleştirilmelidir.

“DataGrid” bileşeni satır ve sütunlardan oluşan bir ızgara kontrolüdür. Her sütun kendi başlığına sahiptir. Sütunlarda kendisine bağlı bulunan tablonun kolonları satırlarında ise kayıtları göstermekle görevlidir.

“DataGrid” üzerinde ürünler tablosu yer alacaktır. “AutoGenerateColumns” özelliğinin “false” bırakılmasının sebebi kolon yapısının nasıl şekilleneceği (template) hemen altında oluşturulmuş olmasıdır. Aksi halde kolonları bağlandığı tabloya göre kendisi otomatik olarak tanımlayacak ve bağlayacaktır.

```
<DataGrid AutoGenerateColumns="False" ....
  <DataGrid.Columns>
    <DataGridTextColumn Header="Ürün Adı" Binding="{Binding
Path=ProductName}"/>
    <DataGridTextColumn Header="Fiyat" Binding="{Binding Path=UnitPrice,
Converter={StaticResource ConvertResource}}"/>
    <DataGridTextColumn Header="Miktar" Binding="{Binding Path=UnitsInStock}"/>
    <DataGridTextColumn Header="Birimdeki Miktar" Binding="{Binding
Path=QuantityPerUnit}"/>
    <DataGridCheckBoxColumn Header="Tükendi" Binding="{Binding
Path=Discounted}"/>
  </DataGrid.Columns>
</DataGrid>
```

“DataGrid” kontrolünün dört adet “text” tipi bir adet “checkbox” tipi bileşenden mevcut olacağı “<DataGrid.Columns>” içinde belirtilmiştir.

Kolonların başlık bilgileri “Header” özelliği ile atanmaktadır. “Binding” parametresi ile yolun (path) bağlandığı tablonun hangi alanını göstereceği bilgisi verilmiştir.

Fiyat kolonunda farklı olarak “Converter” parametresinin olduğu görülür. Tanımında statik kaynaklardan “ConvertResource” anahtar adına (key) sahip kaynaktan elde edilmesi gerektiği ifade edilmiştir. Bu kaynak incelendiğinde;

```
..... xmlns:my="clr-namespace:NorthwindSoft">
<Window.Resources>
  <my:PriceConverter x:Key="ConvertResource" />
</Window.Resources>
```


ValueConverter

“my” isim alanı takısı ile projemiz isim alanında bulunacak olan “PriceConverter” isimli sınıfını göstermektedir. Bir bileşene ait veri bağlamada “Converter” bilgisinin verilmesi, alan gösteriminde kaynak bilgi olduğu türde değil başka bir türe dönüştürülerek yapılacağı anlamını taşır. Uygulamada fiyat (UnitPrice) kolonu gösterimi yapılırken kaynaktan gelecek olan veri “PriceConverter” sınıfı tarafından işlenecek ve bilgi dönüşüme uğratılacaktır. Bu sınıfı aşağıdaki gibi düzenleyip projenize ekleyiniz.

```
using System;
using System.Windows.Data;
namespace NorthwindSoft
{
    class PriceConverter:IValueConverter
    {
        public object Convert(object value, Type targetType, object parameter,
            System.Globalization.CultureInfo culture)
        {
            if (value != null)
            {
                return String.Format("{0:C}", value);
            }
            return "";
        }

        public object ConvertBack(object value, Type targetType, object parameter,
            System.Globalization.CultureInfo culture)
        {
            throw new NotImplementedException();
        }
    }
}
```

“PriceConverter” sınıfını özel yapan “IValueConverter” arayüzü ile gerçekleştirilmiş olmasıdır. Bu arayüz kendi içinde iki adet metot barındırır. “Convert” metodu kaynaktan gelen verinin hedefe hangi tipe ve nasıl dönüştürüleceği, “ConvertBack” ise bunun tam tersi işleve sahip olmak için gövde tanımların yapılması gerekmektedir.

Uygulamamız içinde “decimal” tipte gelecek olan fiyat bilgisini hedefe (arabirim denetimine) TL para birimi şeklinde virgülden sonra 2 haneli olacak şekilde formatlanarak gönderilmesini istediğimizden “Convert” metodunda gelen değerin (metodun value parametresi ile) öncelikle boş (null) değer içerip içermediğine bakılıp ardından metinsel desen formatını uygulayıp geriye (return) döndürülür. Bu tip dönüşümlere ihtiyaç duyacağınız farklı kolonlar varsa her biri için ayrı ayrı dönüşüm sınıflarını “IValueConverter” arayüzünden bildirilmiş olma şartıyla tanımlamalısınız.

Code Behind

Formda yer alan buton nesnemiz fare (mouse) ile tıklandığında kod sayfasında bulunan “btnGetir_Click” metodunu çalıştıracaktır. “MainWindow.xaml.cs” dosyasında bulunan kodu aşağıdaki gibi düzenleyiniz.

```
NorthwindDataClassDataContext context;  
BindingList<Product> productsInfo;  
Category cat;  
  
private void btnGetir_Click(object sender, RoutedEventArgs e)  
{  
    // Getir butonuna tıklandığında çalışacak methoddur.  
    // Veri tabanı örnekleme oluşturulmaktadır.  
    context = new NorthwindDataClassDataContext();  
    // Combobox'ın bağlanacağı içerik Kategori tablosu olarak belirlenir  
    cmbKategori.DataContext = context.Categories;  
}
```

```
private void cmbKategori_SelectionChanged(object sender, SelectionChangedEventArgs e)  
{  
    // Combobox seçimi değiştiğinde çalışacak methoddur.  
    // Combobox'da seçili öge Kategori türüne çevrilip cat değişkeninde tutulur.  
    cat = (Category)cmbKategori.SelectedItem;  
    // cat nesnesi üzerinden ürünlerin bulunduğu Products tablosu  
    // kayıt listesi liste değişkenine verilir.  
    IList liste = ((IListSource)cat.Products).GetList();  
    // liste nesnesi Product tipinden BindingList nesnesine dönüştürülür.  
    productsInfo = (BindingList<Product>)liste;  
    // Datagrid'in bağlanacağı içerik Kategoriden seçilmiş ürünler olarak  
    // BindingList nesnesi olarak belirlenir  
    dgUrunler.ItemsSource = productsInfo;  
}
```

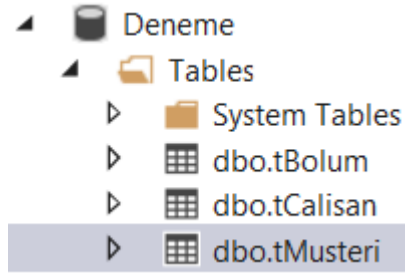
Sonuç

Veri tabanına bağlanmak ve kategori ile ürünler tablosunu elde etmek için “NorthwindDataClassDataContext” sınıfından “context” isimli nesne örnekleme yapılmıştır. “cmbKategori” isimli “combobox” bileşeninin sahip olacağı verileri “DataContext” özelliği (property) ile “context” nesnesi içinde yer alan Kategori (Categories) varlık sınıfı ile elde edecektir. Çalışma anında formda yer alan “getir” butonuna tıklandığında kategori listesi “combobox” bileşenine yüklenmiş olacaktır.

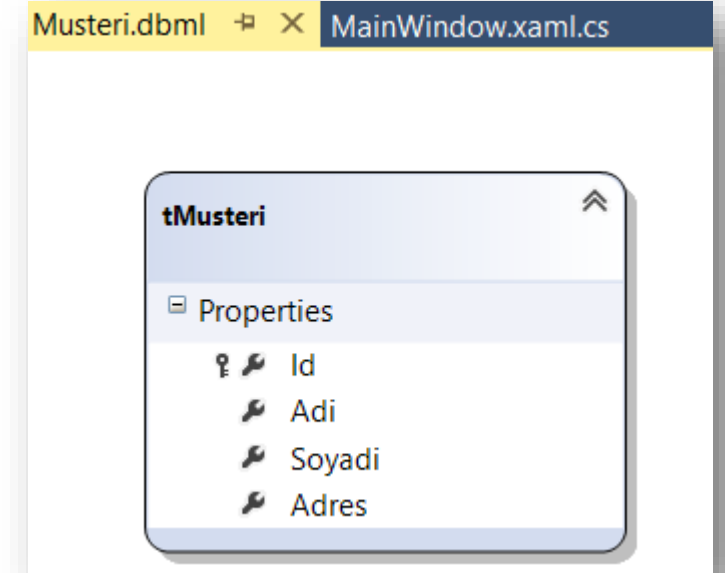
“Combobox” listesinde bir seçim değişikliği meydana geldiğinde “cmbKategori_SelectionChanged” metodu çalışacaktır. Öncelikle Kategori tipinde olan “cat” değişkeni “combobox” bileşeninin seçili ögesini işaret etmektedir. Böylelikle seçilen kategori bilgisi elde edilmiş olur. İlişkisel nesne haritası (O/RM) çıkarılırken kategori (category) ile ürün (product) tablosu arasındaki ilişki sonucu kategori varlık sınıfı içinde products varlık seti (EntitySet) tanımlaması yapılmıştır. Bu tanımlama sayesinde bir kategoriye ait ürün varlıkları sorgulayabilirsiniz. Kategori seçimine bağlı elde edilen ürün listesi “IList” tipinde “liste” isimli nesneye verilebilmektedir. Son olarak “liste” nesnesi “products” tipinde “BindingList” türüne çevrilerek “datagrid” bileşeninin veri kaynağının işaret edildiği “ItemSource” özelliğine bağlanmıştır. Bu sayede daha önce “xaml” tarafında “DataGrid” için “ItemSource={Binding}” ile veri kaynağı belirlenmiş olur.

Ürün Adı	Fiyat	Miktar	Birimdeki Miktar	Tükendi
Queso Cabrales	21,00 TL	22	1 kg pkg.	<input type="checkbox"/>
Queso Manchego La Pastora	38,00 TL	86	10 - 500 g pkgs.	<input type="checkbox"/>
Gorgonzola Telino	12,50 TL	0	12 - 100 g pkgs	<input type="checkbox"/>
Mascarpone Fabioli	32,00 TL	9	24 - 200 g pkgs.	<input type="checkbox"/>
Geitost	2,50 TL	112	500 g	<input type="checkbox"/>
Raclette Courdavault	55,00 TL	79	5 kg pkg.	<input type="checkbox"/>
Camembert Pierrot	34,00 TL	19	15 - 300 g rounds	<input type="checkbox"/>
Gudbrandsdalsost	36,00 TL	26	10 kg pkg.	<input type="checkbox"/>
Flotemysost	21,50 TL	26	10 - 500 g pkgs.	<input type="checkbox"/>
Mozzarella di Giovanni	34,80 TL	14	24 - 200 g pkgs.	<input type="checkbox"/>

Arama Uygulaması



dbo.tMusteri [Data]				
yeniKayit.xaml.cs				
yeniKayit.xaml				
Max Rows: 1000				
	Id	Adi	Soyadi	Adres
	1	Ahmet	Çokçalışır	Cumhuriyet ...
	2	Mehmet	Hiçdurmaz	Karanfil Sok.
	3	Hasan	Yorulmaz	TOKİ
	4	Hüseyin	Hepkoşar	Susam Sok.
	1002	Ayşe	Kulakasmaz	Çıkmaz Sok.
	3002	Ali	Hepyürür	Elmadağ
	3003	Veli	Çokçalışır	Hasanoğlu
*	NULL	NULL	NULL	NULL



Müşteri Adı:

a



Müşteri Adı:

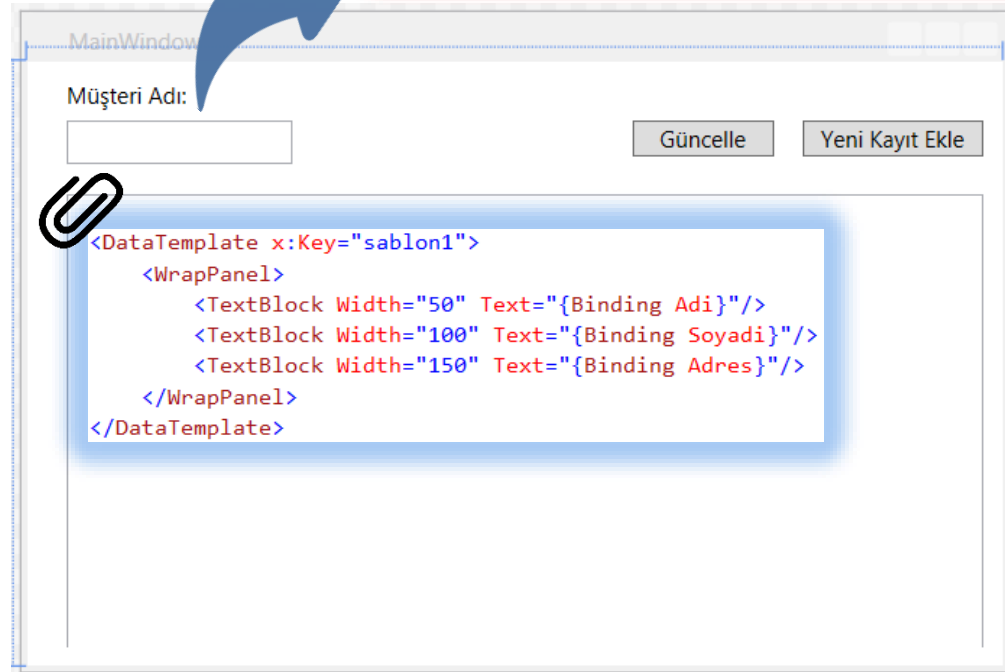
ah

Ahmet	Çokçalışır	Cumhuriyet Cd.
Ayşe	Kulakasmaz	Çıkmaz Sok.
Ali	Hepyürür	Elmadağ

Ahmet	Çokçalışır	Cumhuriyet Cd.
-------	------------	----------------

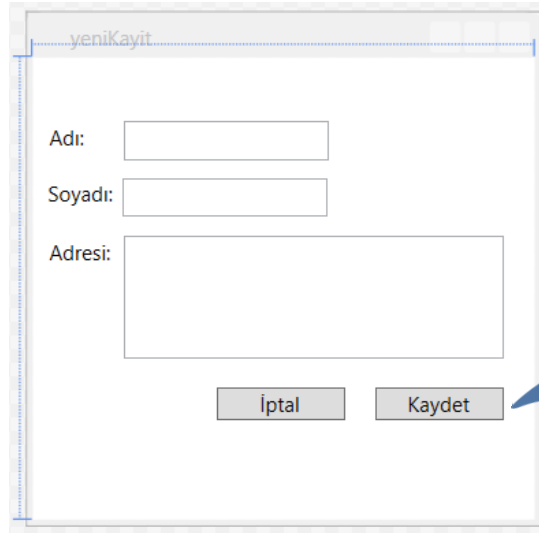
Arama Uygulaması – Hepsi bu kadaaaar😊

```
private void TextBox_TextChanged(object sender, TextChangedEventArgs e)
{
    lb1.ItemsSource = m.tMusteris.Where(x => x.Adi.StartsWith(tb1.Text));
}
```



```
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    m = new MusteriDataContext();
}
```

Kaydetme Uygulaması – Yine hepsi bu kadar😊



InsertAllOnSubmit()

```
private void btnKaydet_Click(object sender, RoutedEventArgs e)
{
    MusteriDataContext m = new MusteriDataContext();
    tMusteri x = new tMusteri();    //Yeni bir müşteri tanımla
    x.Adı = tbAdi.Text;             //Müşterinin özelliklerini belirle
    x.Soyadı = tbSoyadi.Text;
    x.Adres = tbAdresi.Text;

    m.tMusteris.InsertOnSubmit(x);  //Müşteriyi kaydet
    m.SubmitChanges();              //Değişiklikleri güncelle

    this.Close();                  //Bu window'u kapat
}
```

Pekiye! Ama Nasıl ?

List<T> Yapısı

```
List<Personel> emyoPersonel = new List<Personel>();

Personel p1 = new Personel();
p1.Adi = "Ahmet";
p1.Soyadi = "Çokçalışır";

emyoPersonel.Add(p1);

Personel p2 = new Personel();
p2.Adi = "Mehmet";
p2.Soyadi = "Hiçdurmaz";

emyoPersonel.Add(p2);

emyoPersonel.Add(new Personel { Adi = "Hasan", Soyadi = "Yorulmaz" });
emyoPersonel.Add(new Personel { Adi = "Ayşe", Soyadi = "Hepkoşar" });

dg1.ItemsSource = emyoPersonel;
lb1.ItemsSource = emyoPersonel;
```

class Personel

{

4 references

public string Adi { get; set; }

4 references

public string Soyadi { get; set; }

}

Adi	Soyadi	
Ahmet	Çokçalışır	
Mehmet	Hiçdurmaz	
Hasan	Yorulmaz	
Ayşe	Hepkoşar	

DataGrid

VeritabaniGiris.Personel
VeritabaniGiris.Personel
VeritabaniGiris.Personel
VeritabaniGiris.Personel
{NewItemPlaceholder}

ListBox



DataTemplate

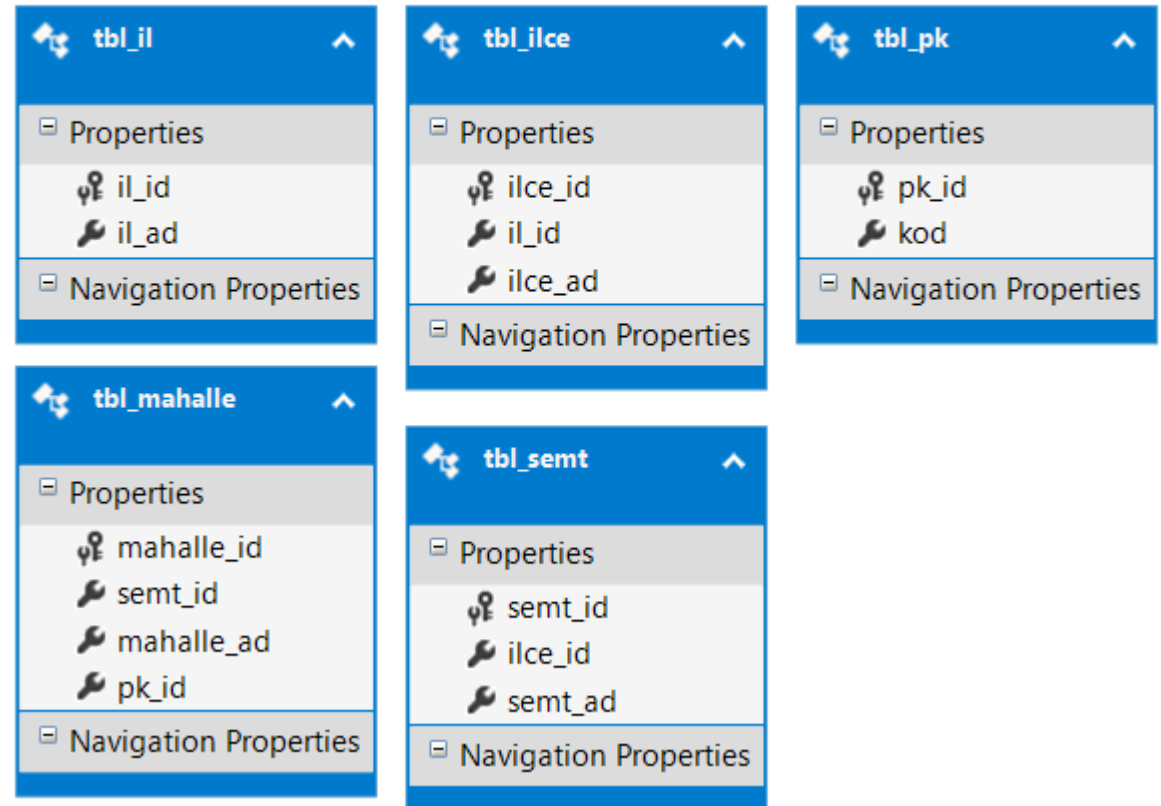
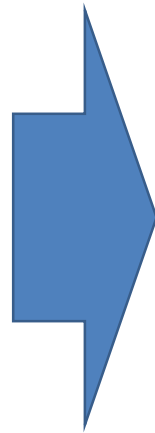
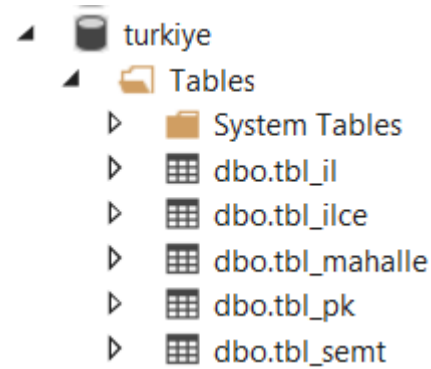
Entity Framework

Turkiye.edmx [Diagram1]*

dbo.tbl_il [Data]

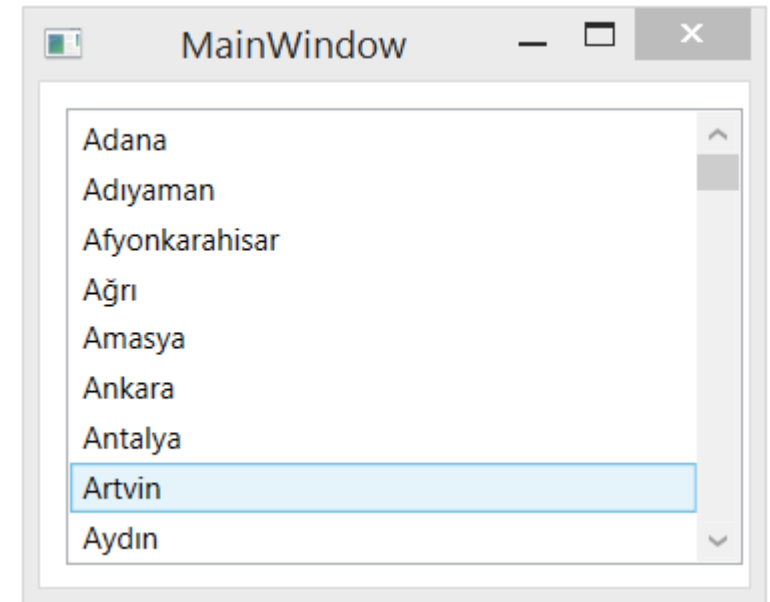
MainWindow.xaml

MainV



Entity Framework

```
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    turkiyeEntities t = new turkiyeEntities();
    foreach (tbl_il x in t.tbl_il)
    {
        lb1.Items.Add(x.il_ad);
    }
}
```



LINQ to SQL Classes - Entity Framework

Kavram	EF 4.0	LINQ to SQL
Sql Server Harici Veritabanı Desteđi	Var	Yok gibi
Doğrudan veritabanı bağlantısı	Yok	Var
Çoklu tablodan kalıtım(Multiple Table Inheritance)	Var	Yok
Birden fazla tablodan tek bir Entity üretmek	Var	Yok
Conceptual Schema Definition Language(CSDL)	Var	Yok
Storage Schema Definition Language(SSDL)	Var	Yok
Mapping Schema Language(MSL)	Var	Yok
Lazy Loading	Var	Var
Stored Procedures	Var	Var

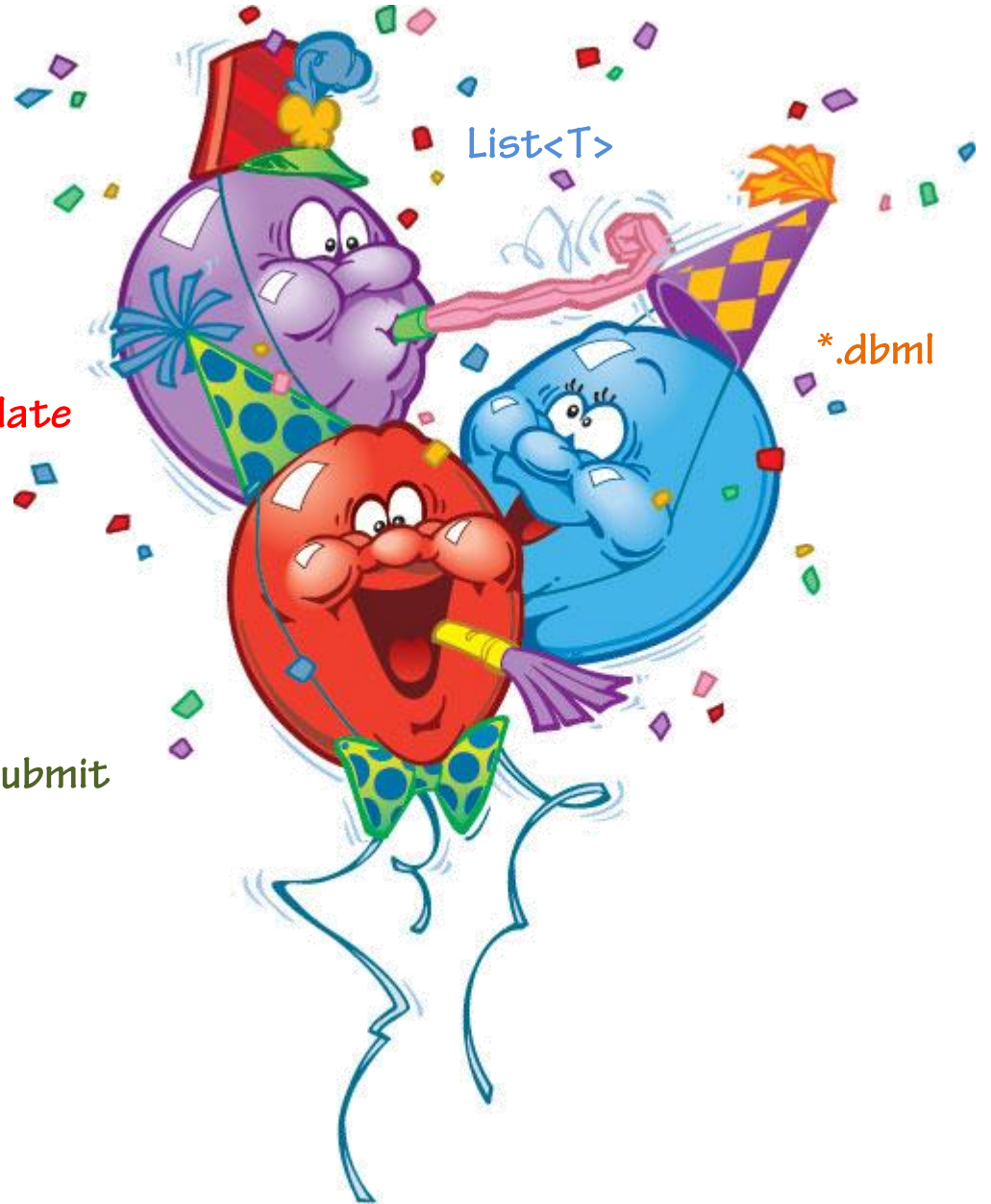
Havada Kalmasin ☺

`DataTemplate`

`InsertOnSubmit`

`List<T>`

`*.dbml`



Teşekkürler...



*bize
katlandığınız
için!*
