

**T.C.
MİLLÎ EĞİTİM BAKANLIĞI**

BİLİŞİM TEKNOLOJİLERİ

NESNE TABANLI PROGRAMLAMADA VERİ YÖNETİMİ

Ankara, 2013

- Bu modül, mesleki ve teknik eğitim okul/kurumlarında uygulanan Çerçeve Öğretim Programlarında yer alan yeterlikleri kazandırmaya yönelik olarak öğrencilere rehberlik etmek amacıyla hazırlanmış bireysel öğrenme materyalidir.
- Millî Eğitim Bakanlığınca ücretsiz olarak verilmiştir.
- PARA İLE SATILMAZ.

İÇİNDEKİLER

AÇIKLAMALAR	ii
GİRİŞ	1
ÖĞRENME FAALİYETİ-1	3
1. VERİ SORGULAMA	3
1.1. Northwind Veri Tabanı	4
1.2. Veri Tabanı Oluşturma.....	4
1.3. Veri Tabanıyla Bağlantı Oluşturma	5
1.4. Verileri Sorgulama.....	9
1.5. Bağlantıyı Kapatma.....	13
1.6. DLINQ Kullanarak Veri Tabanını Sorgulama	14
1.7. DataContext Sınıfı.....	18
UYGULAMA FAALİYETİ	25
ÖLÇME VE DEĞERLENDİRME	26
ÖĞRENME FAALİYETİ-2	27
2. VERİ İŞLEMLERİ.....	27
2.1. DLINQ İle Veri Bağlama.....	27
2.2. DLINQ İle Veri Güncelleme.....	43
2.3. Veri Ekleme ve Silme	44
UYGULAMA FAALİYETİ	46
ÖLÇME VE DEĞERLENDİRME	47
MODÜL DEĞERLENDİRME	48
CEVAP ANAHTARLARI	49
KAYNAKÇA	50

AÇIKLAMALAR

ALAN	Bilişim Teknolojileri
DAL/MESLEK	Veri Tabanı Programcılığı
MODÜLÜN ADI	Nesne Tabanlı Programlamada Veri Yönetimi
MODÜLÜN TANIMI	Veri tabanında bilgiyi sorgulama ve üzerinde işlem yapma becerilerinin kazandırıldığı bir öğrenme materyalidir.
SÜRE	40/32
ÖN KOŞUL	Nesne Tabanlı Programlamada Windows Uygulamaları modülünü başarmış olmak
YETERLİK	Veriyi yönetmek
MODÜLÜN AMACI	Genel Amaç Bu modül ile gerekli ortam sağlandığında, veriyi yönetebileceksiniz. Amaçlar <ol style="list-style-type: none">1. Veri tabanında bilgiyi sorgulayabileceksiniz.2. Veri üzerinde işlem yapabileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	Ortam: Bilgisayar laboratuvarı Donanım: Bilgisayar, programlama yazılımı
ÖLÇME VE DEĞERLENDİRME	Modül içinde yer alan her öğrenme faaliyetinden sonra verilen ölçme araçları ile kendinizi değerlendireceksiniz. Öğretmen modül sonunda ölçme aracı (çoktan seçmeli test, doğru-yanlış testi, boşluk doldurma, eşleştirme vb.) kullanarak modül uygulamaları ile kazandığınız bilgi ve becerileri ölçerek sizi değerlendirecektir.

GİRİŞ

Sevgili Öğrenci,

Günümüzde pek çok orta ve büyük çaplı kurumsal uygulamalar, projeler, teknoloji yatırımları zamanla büyüyen iş hacmi ile birlikte beraberinde büyük miktarda verilerin işlenebilmesini de gerektirmektedir. Birikecek olan bu veri ambarlarının bir şekilde depolanması ve gerektiğinde yazılımlar tarafından ele alınarak kullanıcının talep ettiği bir biçimde işletilmesi iş dünyasının en büyük gereksinimlerini oluşturmaktadır. Aynı zamanda bu gereksinimlere paralel olarak zamana karşı olan rekabet, kurumsal ihtiyaçların gün geçtikçe değişimi, gelişimi, hiç umulmayan amaçlar için kullanılması, istenen çözümler gibi beklentilere yanıt verecek yazılımların üretilmesi bilişim teknolojilerinin en fazla üretken olduğu konuları arasındadır.

Bu modülde .NET Framework çatısı altında veri erişim tekniklerini, bu tekniklerin uygulanabilmesine olanak sağlayan ADO.NET ve ilişkisel nesne haritası (object / relational mapping) olan DLINQ (Linq to Sql) ve araçlarını öğreneceksiniz. Bu sayede veri tabanında bulunan verileri, uygulamanızın iş mantığı üzerinde yönetecek ve sunum katmanında kullanıcınızın hizmetine sunabileceksiniz.

ÖĞRENME FAALİYETİ-1

AMAÇ

Veri tabanında ihtiyacınıza yönelik bilgi sorgulayabileceksiniz.

ARAŞTIRMA

- Veri bazlı yazılım projeleri hangi sektörlerde daha fazla kullanılmaktadır? Araştırınız.
- İlişkisel veri tabanı yönetim sistemleri ve birlikte çalıştıkları platformlar nelerdir? Araştırınız.
- Veriyi organize etmek için kullanılan yöntem ve teknikler nelerdir? Araştırınız.

1. VERİ SORGULAMA

Günümüz uygulamaları ister masaüstü ister web tabanlı olsun mutlaka bir veri tabanı (database) yazılımı kullanmak zorundadırlar. Çünkü çağımızda bilgi ve bilgiye erişim çok önemli bir rol oynamaktadır. Bir amaç etrafında verileri topluyoruz, sorguluyoruz, bir sonuca ulaşmak için onları işleyip raporlar elde ediyoruz. Çünkü geleceği tahmin etmek istiyoruz. Bu sebepten dolayı bilgisayar programları da, çeşitli bilgileri saklamak, düzenlemek ve tekrar saklamak zorundadır.

Uygulama geliştiriciler ise bu veriler etrafında her zamankinden daha hızlı, esnek, ölçeklenebilir, yüksek erişilebilirlik değerlerine sahip ve performanslı çözümler oluşturmak durumundadırlar. Hazırladıkları yazılımların günün şartlarına uygun olması, kurumsal platformlarda aranan daha fazla esneklik ve daha yüksek performans özellikleri gösteren mimariye sahip olması ilk akla gelen “olmazsa olmaz” şartlardan sadece birkaçıdır. Bunlara ek olarak hazırlanan uygulamanın günümüz standartlarına hitap etmesi iyi bir yazılım olma adına önemli bir gereksinimdir.

İşte bu gereksinimlerin karşılığı olarak .Net uygulama geliştiricilere ADO.NET adı verilen ve uygulamanız ile sahip olduğunuz verilerin saklandığı veri tabanı sisteminize erişimde bulunmanıza ve üzerinde işlem yapmanıza olanak sağlayan bir teknoloji sunmaktadır. ADO.NET (ActiveX Data Object) eski nesil ADO veri erişim teknolojisinin .NET için yeniden tasarlanmış halidir. .NET Framework’un bir parçası olarak 1.0 sürümünden beri önemli bir yere sahiptir.

Veri erişim yöntem ve teknikleri değişen teknoloji ve yeni nesil programlama paradigmaları ile birlikte sürekli olarak değişim göstererek MS SQL Server, Oracle, MySQL gibi pek çok ilişkisel veri tabanı yönetim sistemine (RDBMS) bağlanabilmeyi ve nesne

yönelimli mimarisi ile verilerinizi uygulamanız etrafında kolayca yönetebilmeyi ve bütün bunları XML standartlarında gerçekleştirebilmeyi mümkün kılmaktadır.



Resim 1.1: ADO.NET

Geçmişteki veri erişim teknolojileri ile ODBC (Open Database Connectivity), DAO (Data Access Objects), RDO (Remote Data Objects), OLE DB (Object Linking and Embedding DataBase), ADO (ActiveX Data Objects), teknolojilerini sunan Microsoft günümüz gereksinimlerini karşılamakta ADO.NET’i uygulama geliştiricilerin kullanımına sunmaktadır. Ancak uygulamaların giderek artan kod satırları, Windows uygulamaları, web servisleri, web uygulamaları gibi konularda programcılara daha üstün hizmet vermek ve daha hızlı üretimlerde bulunmalarını sağlamak maksadıyla uygulama geliştiricilere ADO.NET temelinde .NET kodunu SQL deyimlerine dönüştürebilen DLINQ (Linq to SQL) ve sonrasında veri tabanınızı nesne yönelimli olarak yönetebilmenize olanak sağlayan “EntityFramework” çatısını uygulama geliştiricilere bir araç olarak sunmaktadır.

1.1. Northwind Veri Tabanı

Veri bazlı uygulamalar geliştirmede deneysel amaçlı ele alınmış “Northwind” şirketi müşterilerine yiyecek ve içecek ürünleri satan hayali bir şirkettir. “Northwind” veri tabanı ise şirketin sattığı ürünler (products), müşteriler (customers), müşterilerin verdiği siparişler (orders), faturalar, tedarikçiler (suppliers), ürünlerin teslim edilmesinde kullanılan nakliyeciler (suppliers) ve şirket için çalışan personel (employees) bilgilerinin tutulduğu ilişkili tabloları (table) görünümüleri (view) ve saklı yordamları (stored procedure) barındırır.

Veri erişimi ve sorgulama tekniklerini öğreneceğiniz bu modülde uygulama örneklerimiz için “Northwind” veri tabanını kullanacaksınız.

1.2. Veri Tabanı Oluşturma

Northwind veri tabanı için gerekli dosyaları;
<http://www.microsoft.com/en-us/download/details.aspx?id=23654> adresinden temin edebilirsiniz.

Yüklemesi yapılan “SQL2000SampleDb.msi” isimli programı çalıştırdığınız takdirde bilgisayarınızın “C:\SQL Server 2000 Sample Databases” klasörü altında veri tabanı

dosyalarının oluşturulduğunu göreceksiniz. Bu aşamadan sonra yapılması gereken bilgisayarınızda yüklü bulunan SQLServer örneğine (instance) Northwind veri tabanını kayıt etmek olacaktır. Gereken işlem adımları;

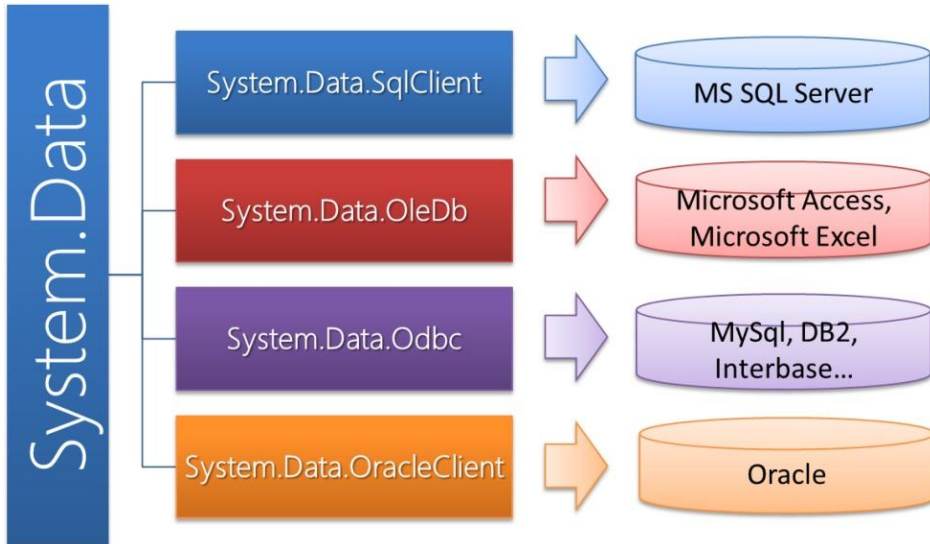
- Windows Başlat menüsü altında “cmd” yazarak konsol ortamına geçiniz.
- Konsol ekranında “>cd SQL Server 2000 Sample Databases” komutunu veriniz.
- “>osql -E -S ComputerName\InstanceName -i instnwnd.sql” komutunu uygulayınız. Komut içinde geçen “ComputerName” kurulumun yapıldığı bilgisayar adı “InstanceName” ise bilgisayarınızda kurulu olan SQL Server’ın adı olarak yazmalısınız.

1.3. Veri Tabanıla Bağlantı Oluşturma

Veri tabanına bağlanmada ADO.NET size “System.Data” isim alanı içinde bir takım hizmet sınıfları sunmaktadır. “System.Data” isim alanı aynı zamanda oldukça geniş bir sağlayıcı kümesini de içinde barındırır. Bu sayede farklı platformlara ait (Oracle, MySQL, Access vb.) veri kaynaklarına bağlanabilme imkânı sağlar. Böylece farklı veri kaynaklarını tek bir uygulamada toplayabilir ve hatta birbirleri ile konuşurabilirsiniz.

Projenizde hangi veri tabanı ile bağlantıya geçmek istiyorsanız ilgili sağlayıcı isim alanını üzerinde bulunan sınıf hizmetlerini kullanmalısınız.

ADO.NET uygulama geliştiricilere 4 farklı isim alanı farklı veri kaynaklarına bağlanabilme imkanı sunmaktadır.



Resim 1.2: ADO.NET veri sağlayıcıları (Data Providers)

➤ **System.Data.SqlClient**

MS SQL Server ile çalışabilmek için yazılmış tiplerin yer aldığı bu isim alanı, SQL Server 7.0 ve sonraki versiyonları ile kullanılabilir.

➤ **System.Data.OleDb**

MS SQL Server 6.5 ve önceki sürümleri, Microsoft Access, Oracle, Microsoft Excel dosyaları gibi OleDb arayüzüne sahip tüm sistemler ile bağlantı kurmak için gerekli hizmet sınıflarını barındırmaktadır.

➤ **System.Odbc**

ODBC (Open Database Connectivity) standartlarını destekleyen ve ODBC sürücüsü bulunan sistemlere bağlantı kurmayı sağlayan hizmet sınıflarını barındırmaktadır.

➤ **System.Data.Oracle**

.Net 2.0'dan sonra Oracle veri yönetim sistemine daha güçlü destek sağlanmıştır. Ancak Oracle veri sağlayıcısını kullanabilmek için öncelikle "System.Data.OracleClient" referansının proje referansları arasına dahil edilmesi gerekmektedir.

Hangi ilişkisel veri tabanı yönetim sistemi ile bağlantıya geçmek ve üzerinde işlem yapmak istiyorsanız öncelikle ilgili veri sağlayıcısına (provider) ait isim alanını kod satırlarınızın başında "using" deyişi ile dahil etmelisiniz.

Örnek 1.1. MS SQL Server üzerinde planladığınız bir veri tabanı ile uygulama geliştirmek istiyorsunuz. Projede kullanılacak referans bildirimlerini gerçekleştiriniz.

```
using System;
using System.Data;
using System.Data.SqlClient;//SQL Server için veri sağlayıcı isim alanı

namespace Proje1
{
    class Program
    {
        static void Main(string[] args)
        {
            // Uygulama Kodları
        }
    }
}
```

Örnek 1.2. MS Access üzerinde planladığımız bir veri tabanı ile uygulama geliştirmek istiyorsunuz. Projede kullanılacak referans bildirimlerini gerçekleştiriniz.

```
using System;
using System.Data;
using System.Data.OleDb; // MS Access için veri sağlayıcı sınıfları isim alanı

namespace Proje1
{
    class Program
    {
        static void Main(string[] args)
        {
            // Uygulama Kodları
        }
    }
}
```

Yukarıdaki örneklerde her iki farklı sağlayıcı için de ortak olarak “System.Data” isim alanı (namespace) bildiri yapılması dikkatinizi çekmiştir. Bunun sebebi uygulama kodları içinde bağlantının yapılması ve veri tabanı yönetim sistemine sorgu iletilmesinde kullanılacak hizmet sınıflarının yazımında bu isim alanı içinde yer alan bir takım öğelerin kullanılacak olmasıdır. Hangi veri sağlayıcı ile çalışırsanız çalışın mutlaka “System.Data” isim alanını da projenize referans vermeyi unutmayınız.

İlgili sağlayıcıya ait referanslarının bildiriminden sonra veri tabanına bağlantı işlemi için uygulama kodunuzu oluşturmaya başlayabilirsiniz. İlk olarak düşünmeniz gereken bağlantının kurulması ve açılması aşamalarıdır.

Veri kaynağı, uygulamanın çalıştığı makinede olabileceği gibi ağ(intranet, internet) üzerinde bulunan uzak bir sunucu da olabilir. Bu sebeple uygulamanız ile veri tabanı arasındaki köprünün kurulabilmesi için bir bağlantı (connection) sınıfına ihtiyacınız olacaktır. Bu sınıf “System.Data” isim alanına ait “dbConnection” sınıfından türetilmiş olup hangi platformda bulunan veri kaynağına bağlanmak istenirse o platformu destekleyen sağlayıcı hizmete ait isim alanı içindeki bağlantı sınıfı “new” anahtar kelimesi ile örneklenip kullanılmalıdır.

Örnek 1.3. MS SQL Server’ a bağlanmak için gereken nesne örneğini oluşturunuz.

```
SqlConnection baglanti=new SqlConnection();
```

Örnek 1.4. Access veri tabanına bağlanmak için gereken nesne örneğini oluşturunuz.

```
OleDbConnection baglanti=new OleDbConnection();
```

Bağlanmak istediğiniz veri tabanı yönetim sistemine ait ilgili sağlayıcı (provider) üzerinden bağlantı nesnesinin örneklenmesi ilk aşamadır. Ancak bu bağlanma işlemi için yeterli değildir. Bu bağlantının hangi makinede, hangi veri tabanına, hangi güvenlik ayarları ile yapılacağını belirtmesi gereklidir. Bu bilgi iki şekilde belirtilebilir.

- xxxConnection sınıfı üzerinde örnekleme yapılırken kurucu parametresi olarak verilebilir.
- Bağlantı sınıfı üzerinden elde edilen nesnenin .ConnectionString isimli string türünden parametresi ile bildirilebilir.

Bağlantı bilgisi metinsel (string) türünden olup parametreleri ve buna bağlı belirli kurallara göre yazılır. Bağlantı bilgisinde kullanılan parametreler Tablo 1.1’de açıklanmıştır.

Parametre	Tanımı
Provider	Sağlayıcı ismi (Yalnızca OleDb kullanır.)
ConnectionTimeout	Bağlantı için en fazla bekleme süresi
Initial Catalog	Veri tabanı adı
Datasource	SQL Server kurulumu adı veya MSAccess için dosya adı
Password	Login parolası
UserId	Login adı
Integrated Security	SQL Server’a Windows hesabı ile giriş yapılacağını belirtir
WorkStationId	WorkStation veya client adı
Mode	Veri tabanına yalnızca okunulabilir (read-only) veya yazma (write) modunda bağlanmayı sağlar. SQL Server bağlantılarında kullanılmaz.

Tablo 1.1: Bağlantı bilgisi için kullanılan parametreler

Örnek 1.5. PC isimli makinede bulunan SQLEXPRESS isimli SQL Server kurulumunda yer alan Northwind veri tabanına Windows kimlik doğrulaması ile bağlantı kurunuz.

```
SqlConnection baglanti=new SqlConnection("Data Source=PC\\SQLEXPRESS; Initial Catalog=Northwind; Integrated Security=true");
```

veya;

```
SqlConnection baglanti=new SqlConnection();  
baglanti.ConnectionString="Data Source=PC\\SQLEXPRESS; Initial Catalog=Northwind;Integrated Security=true";
```

Örnek 1.6. 192.168.1.100 ip adresinde bulunan SQLWeb isimli SQLServer kurulumunda yeralan “Northwind” veri tabanına “Mixed Mode” kimlik doğrulaması ile kullanıcı adı “admin” şifresi “root” olarak belirlenmiş bağlantıyı kurunuz.

```
SqlConnection baglanti=new SqlConnection("Data  
Source=192.168.1.100\\SQLWeb;Initial Catalog=Northwind; Username=admin;  
Password=root");
```

Örnek 1.7. Bilgisayarın D lokal sürücüsünün Data klasörü altında bulunan MS Access 2010 ortamında hazırlanmış “Samples” isimli veri tabanı dosyası ile bağlantı kurmanız gerekmektedir. Bağlantı nesnesini hazırlayınız.

```
SqlConnection baglanti=new SqlConnection("Provider=Microsoft.ACE.OLEDB.12.0;  
DataSource=C:\\Data\\Samples.accdb");
```

Not: Bundan sonraki örnek uygulamalarımız için PC isimli makinede bulunan SQLEXPRESS isimli SQL Server’da kurulmuş olan “Northwind” veri tabanına Windows kimlik doğrulaması ile erişilecektir.

1.4. Verileri Sorgulama

Daha önceden veri tabanına kurulmuş olan bağlantı üzerinden verileri çekmek, eklemek, silmek, güncellemek gibi sorgulama işlemlerinizi yürütmek amacıyla ilgili sağlayıcıya ait “Command” sınıfını kullanmanız gerekmektedir. Bu sınıf ile SQL sorgularınızı veri tabanına yollayabilir, veri tabanı üzerinde yer alan saklı yordamlarınızı (Stored Procedure) çalıştırabilir, parametrik yapıdaki sorgularınız için parametre tanımlaması yapabilirsiniz.

“Command” sınıfına ait en çok kullanılan özellik ve metodlar Tablo 1.2’de inceleyebilirsiniz.

Sınıf Üyesi	Tanım
CommandText	SQL ifadesinin veya saklı yordam (stored procedure) isminin yazıldığı özelliktir.
CommandType	“CommandText” özelliğine atanan değer bir SQL ifadesi mi yoksa bir saklı yordam mı olacağını belirtilmesini sağlayan özelliktir.
Connection	Bağlantının hangi nesne üzerinden sağlanacağını belirtilmesini sağlayan özelliktir.
ExecuteReader()	Geriye kayıt kümesi döndürmesi beklenen seçme (SELECT) türünde sorguların yürütülmesini sağlayan metoddur. Metoddan dönen kayıt kümesinin sağlayıcının tipinde bir “DataReader” nesnesine verilmesi gerekmektedir.

ExecuteNonQuery()	Veri ekleme (INSERT), silme (DELETE), güncelleme (UPDATE) sorgularının yürütülmesini sağlayan metoddur. Bu tip sorguların yürütülmesi sonucu geriye bundan etkilenen kayıt sayısı döneceğinden bu bilginin “int” tipi bir değişkene eşitlenmesi gerekmektedir.
ExecuteScalar()	Geriye tek bir değer döndüren seçme sorgularının (SELECT) yürütülmesini sağlayan metoddur.
ExecuteXMLReader()	Sadece “SqlCommand” sınıfında (class) bulunan bu metod; SQL Server üzerinden gelen verinin XML (eXtensible Markup Language) formatında alınabilmesini sağlar.
Parameters	Parametrik sorgular için parametre listesini taşır.

Tablo 1.2: Command sınıfına ait özellik ve metodlar

Örnek 1.8. “Northwind” şirketi için çalışan personelin adı, soyadı ve telefon bilgilerini raporlamanız istenen uygulamayı gerçekleştiriniz.

Örnek senaryonun hayata geçirilmesi için aşağıdaki işlem adımlarını uygulayınız.

- Öncelikle yeni bir konsol uygulaması oluşturunuz.
- “Northwind” veri tabanı SQL Server üzerinde kurulu olduğundan uygulamada kullanacağınız hizmet sınıfları için “Program.cs” dosyasına, “System.Data” ve “System.Data.SqlClient” isim alanı bildirimlerini yazınız.

```
using System;
using System.Data;
using System.Data.SqlClient;
```

Uygulama kodları için ilk başlangıcınız bağlantı nesnesinin örnekleme ve bağlantı metninin verilmesi olacaktır.

```
static void Main(string[] args)
{
    // Bağlantı nesnesinin örnekleme
    SqlConnection baglanti = new SqlConnection();
    // Bağlantı bilgisinin bağlantı nesnesine atanması
    baglanti.ConnectionString = "Data Source=PC\\SQLEXPRESS;Initial
        Catalog=Northwind; Integrated Security=true;";
```

- Çalışan personelin kayıtlı olduğu “Employees” tablosundan raporlanması istenen bilgilerin getirilebilmesi için bir seçme sorgusu (SELECT) uygulanmalıdır. Sorgumuzu hazırlamak üzere “Command” sınıfı örnekleme ihtiyacı olacaktır. Oluşturulan nesne örneğine sıra ile önce bağlantı nesnesinin bildirimi, sorgu metninin tipi ve seçme sorgusu metni bildirilmelidir.

```
// SQL sorgusu için Command nesnesinin örneklenmesi
SqlCommand komut = new SqlCommand();
// Sorgu nesnesine bağlantı bilgisinin atanması
komut.Connection = baglanti;
// SQL sorgusunun komut bilgisi olarak atanması
komut.CommandText = "SELECT FirstName,LastName,HomePhone FROM
Employees";
// SQL sorgusunun text bilgisi olduğunun bildirilmesi
komut.CommandType = CommandType.Text;
```

- “komut” Nesnesi ile yürütülmek istenen SQL sorgusundan beklenen bir kayıt kümesidir. Bu nedenle Tablo 1.2’de belirtildiği üzere komutun veri tabanı yönetim sistemine uygulatılmasında “ExecuteReader()” metodunu kullanılmalıdır. Diğer komut yürütme metodlarından farklı olarak bu metodun geri dönüş tipi bir “DataReader” nesnesidir. “DataReader” nesnesi “ExecuteReader()” metodundan dönen kayıt kümesi üzerinde tek yönlü (forward only) ve yalnızca okunulabilir (read only) olarak dolaşmanıza imkan veren bir bileşendir. Burada dikkat edilmesi gereken nokta “komut” nesnesine ait yürütme metodu çağrılmadan önce bağlantının açılmış olması gerekliliğidir.

```
// Geriye dönmesi beklenen kayıt kümesinin alınabilmesi için hazırlanmış
// DataReader nesnesi
SqlDataReader rd;
// Bağlantının açılması
baglanti.Open();
// Sorgunun yürütülmesi, geriye dönen kayıtların DataReader nesnesine verilmesi
rd = komut.ExecuteReader();
```

- Sorgu sonucu gelen kayıt kümesinin her bir satırını “DataReader” nesnesinin Read() metodu çağrıldıkça ancak okunulabilir. “Read()” metodu başarılı ise “True”, başarısız ise “False” değerini döndürür. Genellikle okumalar bu özellikten faydalanılarak bir (Do While Loop) döngüsü ile gerçekleştirilir. Döngü “Read()” metodundan dönen değer True olduğu sürece devam ettirilir. Döngü bloku altında okunan her bir kayıt hücrelerine erişilerek gereken raporlamayı yapabilirsiniz. Döngü bitiminden hemen sonra açık olan bağlantı ve veri okuyucusu (DataReader nesnesi) “Close()” metodu ile kapatılmalıdır.

```

// DataReader nesnesi okunulabildiği sürece döngünün sağlanması
while (rd.Read())
{
// Her kayıt okunduğunda kayıta ait bilgilerin ekrana yazdırılması
Console.WriteLine(rd["FirstName"].ToString() + " " + rd["LastName"].ToString() + "-
->" + rd["HomePhone"].ToString());
}
// DataReader nesnesinin kapatılması
rd.Close();
// Bağlantının kapatılması
baglanti.Close();
}
}
}

```

Programın çalışma sürecinde oluşabilecek bir takım özel istisna durumları ile karşılaşabilir. SQL Sunucusunun yanıt vermemesi, aşırı yoğunluk yüzünden bağlantının zaman aşımına düşmesi, ağ alt yapısında meydana gelebilecek bağlantı kopuklukları gibi olasılıklar izlenmediği takdirde oluşturduğunuz yazılımın hata üretmesine neden olacaktır. Uygulamanız dışında meydana gelebilecek bu özel durumlar uygulamanız tarafından istisna durum denetlemesi ile kontrol altına alınabilir. Oluşabilecek erişim problemlerine karşı programınızın kırılması yerine uygun hata mesajları vermek, kullanılmış açık kalmış sistem kaynaklarını geriye vermek yerinde olacaktır.

Hata denetimi yapılmış haliyle yukarıda ele alınan örneğimizin tam kodları aşağıda verilmiştir.

```

using System;
using System.Data;
using System.Data.SqlClient;

namespace MyApp
{
class Program
{
static void Main(string[] args)
{
SqlConnection baglanti = new SqlConnection();
try
{
baglanti.ConnectionString = "Data Source=PC\\SQLEXPRESS;Initial
Catalog=Northwind;Integrated Security=true;";
SqlCommand komut = new SqlCommand();
komut.Connection = baglanti;
}
}
}
}

```

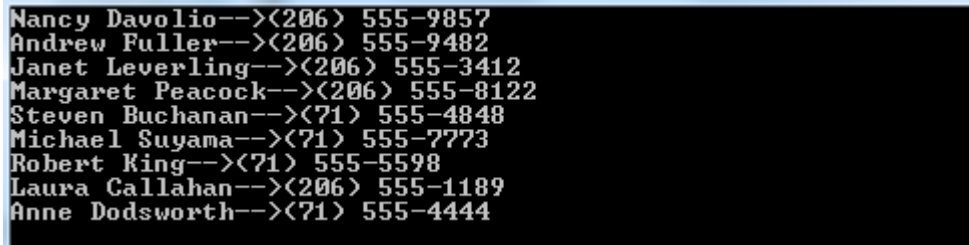


```

        komut.CommandText = "SELECT *FROM Employees";
        komut.CommandType = CommandType.Text;
        SqlDataReader rd;
        baglanti.Open();
        rd = komut.ExecuteReader();
        while (rd.Read())
        {
            Console.WriteLine(rd["FirstName"].ToString() + " " +
rd["LastName"].ToString() + "-->" + rd["HomePhone"].ToString());
        }
        rd.Close();
    }
    catch (SqlException exc)
    {
        Console.WriteLine("Üzgünüm şimdilik kayıtlarınıza erişemiyorum.");
    }
    finally
    {
        baglanti.Close();
    }
}
}
}

```

Uygulamanın çalıştırılması oluşan rapor listesi Resim 1.1'de gösterilmiştir.



```

Nancy Davolio-->(206) 555-9857
Andrew Fuller-->(206) 555-9482
Janet Leverling-->(206) 555-3412
Margaret Peacock-->(206) 555-8122
Steven Buchanan-->(71) 555-4848
Michael Suyama-->(71) 555-7773
Robert King-->(71) 555-5598
Laura Callahan-->(206) 555-1189
Anne Dodsworth-->(71) 555-4444

```

Resim 1.3: Northwind şirketi çalışan personel telefon rehberi

1.5. Bağlantıyı Kapatma

Açılmış bir bağlantı işlem bittiğinde muhakkak kapatılmalıdır. Bunun sebebi veri tabanının açık bırakılan her bağlantı için kendi sistemi üzerinde ek sistem kaynağı tüketimini gerçekleştiriyor olmasıdır. Söz konusu veri tabanı üzerinde birden fazla uygulama ağ (network) üzerinden işlem yapmaya çalışması ve her biri için açık bırakılan bağlantılar doğal olarak sisteme ek yük getirecek ve bu da sunucunun sorgulama için kullanacağı önemli sistem kaynaklarını aynı zamanda bağlantı trafiğine de harcamasını gerektirecektir.

Sorgunuzu yürütmeden hemen önce veri tabanı açılmalı ilgili verilerin alınmasından hemen sonra kapatılması doğru davranış olacaktır.

“Command” sınıfına ait yürütme metodlarına “System.Data.CommandBehaviour” parametresi verilebilir. Bu parametre “DataReader” nesnesi kapandığında kullandığı bağlantıyı otomatik olarak kapatılacağı bilgisi verir. Bunu bağlantı nesnesine ait “Close()” metodu kullanmaktan farklı bir yol olarak düşünebilirsiniz. Kullanım şekli;

```
rd = komut.ExecuteReader(CommandBehavior.CloseConnection);
```

1.6. DLINQ Kullanarak Veri Tabanını Sorgulama

LINQ (Language Integrated Query) “IEnumerable” arayüzüne sahip numaralandırılabilir koleksiyon tiplerini hafızada sorgulamak amacıyla SQL benzeri söz dizimine sahip bir programlama tekniği sunar. Dile entegre edilmiş sorgu anlamını taşır.

Programlama dili tasarımcıları bu konuda daha az miktarda kod ve daha kolay okunulabilir bir tarzda yazım imkanı sunan LINQ projesini geliştirmiştir. LINQ gerek ana fikri gerekse kullandığı dili SQL’e oldukça benzer ve aynı avantajlar sunar.

DLINQ (Database LINQ) yada diğer adı LINQ to SQL (SQL için LINQ) SQL ile veri tabanı üzerinde yapabileceğiniz sorgulama işlemlerini LINQ kodu olarak .NET ortamında düzenlemenize imkan tanır.

Language INtegrated Query to SQL mimarisi ile varlık tipleri (Entity Types) üzerinden sorgular çalıştırılabilir. Basit anlamda, nesnelere (Objects) üzerinde uygulanabilen LINQ sorguları, SQL tarafına ulaştıklarında ise bildiğiniz sorgu ifadelerine (Query Expressions) dönüşmektedir.

ADO.NET teknolojisi üzerine inşa edilen DLINQ ile SQL Server üzerindeki veri tabanı varlıklarınızı sorgulamada “Connection”, “Command”, “DataReader” gibi veri erişim için kullanılan sınıflardan sizi soyutlar. Bu tip ayrıntılar tamamen .NET üzerinde DLINQ tarafından çözümlenir. DLINQ ile amaçlanan veri bazlı uygulamalarda programcının üretkenliğini daha fazla arttırmaktır.

DLINQ ile çalışabilmeniz için ilk olarak veri tabanı varlıklarınızın programlama ortamında nasıl ifade edildiklerini anlamanız gerekmektedir.

Örnek 1.9. “Northwind” veri tabanı üzerinde yer alan ürünler (Products) tablosuna ait bir varlık sınıfını oluşturunuz.

```
using System;
using System.Data.Linq;
using System.Data.Linq.Mapping;

namespace LINQ2SQLOrnek
{
    [Table(Name="Products")]
    class Products
    {
        [Column(IsPrimaryKey=true,CanBeNull=false)]
        private int _ProductID;

        public int ProductID
        {
            get { return _ProductID; }
            set { _ProductID = value; }
        }

        [Column(CanBeNull=false)]
        private string _ProductName;

        public string ProductName
        {
            get { return _ProductName; }
            set { _ProductName = value; }
        }

        [Column]
        private int? _SupplierID;

        public int? SupplierID
        {
            get { return _SupplierID; }
            set { _SupplierID = value; }
        }

        [Column(DbType="money")]
        private decimal? _UnitPrice;

        public decimal? UnitPrice
        {
            get { return _UnitPrice; }
        }
    }
}
```

```
        set { _UnitPrice = value; }  
    }  
}
```

Yukarıda ele alınan “product” isimli varlık sınıfı, uygulamalarımız için veri tabanından gelecek olan ürün bilgilerini ifade etmek için kullanılacaktır. Sınıf yapısına dikkat edilecek olursa normal bir sınıftan farklı olarak sınıf tanımlamasının ve özelliklerinin hemen üzerinde nitelikleri (attributes) görebilirsiniz. Burada bulunan niteliklerin görevi DLINQ için veri tabanı varlıkları ile program öğeleriniz arasındaki eşleştirmeyi sağlamaktır. Kullanmış olduğunuz bu nitelikleri kısaca tanıyalım;

- **[Table]** : Sınıfın tablo olduğu nitelmesini yapar. Kullanılan Name parametresi sembolize ettiği tablonun adıdır.
- **[Column]** : Varlık sınıfına ait özelliğin (property) tablo içinde bir alan olduğu nitelmesini yapar. Kendi içinde pek çok parametresi vardır.
 - **IsPrimaryKey** : Alanın birincil anahtar olduğu bilgisini verir.
 - **DbType** : Alanın sahip olduğu veri türü bilgisini verir.
 - **CanBeNull** : Alanın boş değer girişine izin verilip verilmeyeceği bilgisini verir.

Nitelikleri kullanmamanız durumunda DLINQ, varlık sınıfı ismini ve özellik isimlerini eşleştirmede dikkate alacaktır.

Örnek 1.10. “Northwind” veri tabanı üzerinde yer alan müşteriler (Customer) tablosuna ait bir varlık sınıfını oluşturunuz.

```
using System;
using System.Data.Linq;
using System.Data.Linq.Mapping;
namespace LINQ2SQLOrnek
{
    [Table(Name="Categories")]
    class Kategori
    {
        private int _CategoryID;
        [Column(IsPrimaryKey = true, CanBeNull = false, DbType = "Int NOT NULL IDENTITY")]
        public int CategoryID
        {
            get { return _CategoryID; }
            set { _CategoryID = value; }
        }

        private string _CategoryName;
        [Column(DbType = "NVarChar(15) NOT NULL", CanBeNull = false)]
        public string CategoryName
        {
            get { return _CategoryName; }
            set { _CategoryName = value; }
        }

        private string _Description;
        [Column(DbType="NText")]
        public string Description
        {
            get { return _Description; }
            set { _Description = value; }
        }

        private System.Data.Linq.Binary _Picture;

        [Column(DbType="Image")]
        public System.Data.Linq.Binary Picture
        {
            get { return _Picture; }
            set { _Picture = value; }
        }
    }
}
```

1.7. DataContext Sınıfı

“DataContext” sınıfı uygulamanızın çatısı altında tanımladığımız varlık sınıfları ile veri tabanında yer alan tablolarınız arasındaki bağlantıları yöneterek ve veri akımlarını her iki yapı arasında sağlamakla görevli hizmet sınıfıdır. “System.Data.Linq” isim alanında yer alır.

Örnek 1.11. “Northwind” şirketinde fiyatı 10 birim üzeri olan ürünleri DLINQ kullanarak raporlamasını yapınız.

Uygulamanın DLINQ ile gerçekleştirilmesi için aşağıdaki işlem adımlarını takip ediniz.

- Öncelikle DLINQ kullanımı için ihtiyacımız olan “System.Data.Linq” ve “System.Linq” isim alanı bildirimlerini yapınız.

```
using System;
using System.Data.Linq;
using System.Linq;

namespace LINQ2SQLOrnek
{
    class Program
    {
        static void Main(string[] args)
        {
            // Uygulama Kodları
        }
    }
}
```

- “Northwind” tablosunda bulunan “products” tablosuna ait veriler raporlanacağından projenize yeni bir sınıf ekleyiniz ve “Products” varlık sınıfını örnek 1.9’da görüldüğü gibi oluşturmaya unutmayınız.
- “Main” bloku altında öncelikle “DataContext” sınıfını bağlantı bilgisi verilerek örnekleyiniz. Beklediğiniz verileri yapısında tutacak “Table” tipinden hafızaya alabilmek için “NorthwindContext” nesnesinin “GetTable()” metodunu kullanınız.

```
static void Main(string[] args)
{
    DataContext NorthwindContext = new DataContext("Data
Source=.\SQLExpress;Initial Catalog=Northwind; Integrated Security=true;");

    Table<Product> products = NorthwindContext.GetTable<Product>(); } }
```

- Elde etmek istediğiniz sorguyu ürün tablosu (products) nesnesine uygulayınız.

```
static void Main(string[] args)
{
    DataContext NorthwindContext = new DataContext("Data
Source=.\SQLExpress;Initial Catalog=Northwind; Integrated Security=true;");

    Table<Product> products = NorthwindContext.GetTable<Product>();

    var sorgu = from p in products
                where p.UnitPrice>10
                select p;
}
```

- Sorgu sonucunda “product” koleksiyonu elde edildiğinden “foreach” döngüsü ile her bir kayda tek tek ulaşabilirsiniz.

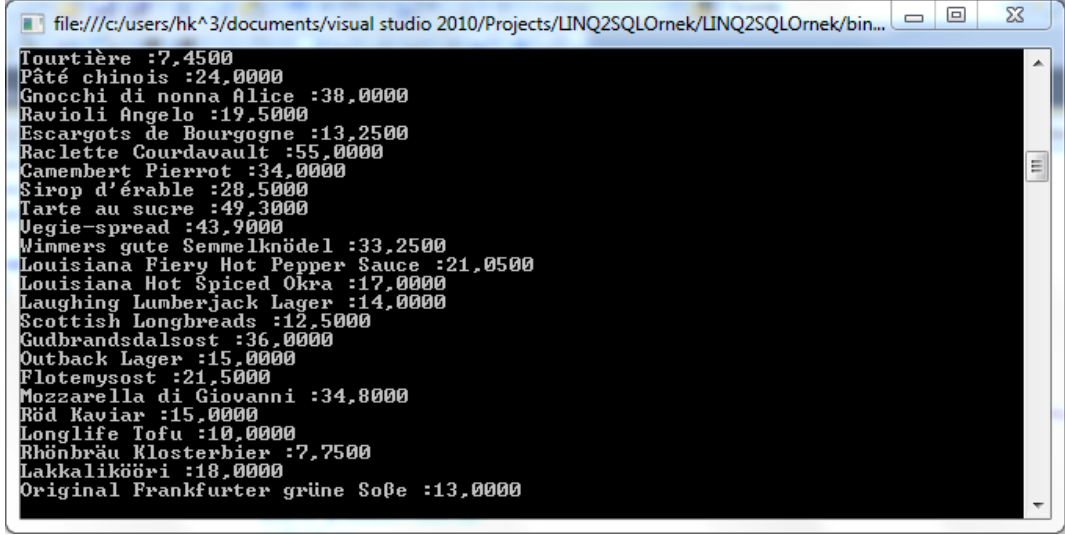
```
using System;
using System.Data.Linq;
using System.Linq;

namespace LINQ2SQLOrnek
{
    class Program
    {
        static void Main(string[] args)
        {
            DataContext NorthwindContext = new DataContext("Data
Source=.\SQLExpress;Initial Catalog=Northwind; Integrated Security=true;");

            Table<Product> products = NorthwindContext.GetTable<Product>();
            var sorgu = from p in products
                        where p.UnitPrice>10
                        select p;

            foreach (Product urun in products)
            {
                Console.WriteLine(urun.ProductName+" :"+urun.UnitPrice);
            }
        }
    }
}
```

Program çalıştırıldıktan sonra elde edeceğimiz ekran çıktısı resim 1.4'te gösterilmiştir.



```
file:///c:/users/hk^3/documents/visual studio 2010/Projects/LINQ2SQLOrnek/LINQ2SQLOrnek/bin...
Tourtière :7,4500
Pâté chinois :24,0000
Gnocchi di nonna Alice :38,0000
Ravioli Angelo :19,5000
Escargots de Bourgogne :13,2500
Raclette Courdavault :55,0000
Camembert Pierrot :34,0000
Sirop d'érable :28,5000
Tarte au sucre :49,3000
Veggie-spread :43,9000
Wimmers gute Semmelknödel :33,2500
Louisiana Fiery Hot Pepper Sauce :21,0500
Louisiana Hot Spiced Okra :17,0000
Laughing Lumberjack Lager :14,0000
Scottish Longbreads :12,5000
Gudbrandsdalsost :36,0000
Outback Lager :15,0000
Flotemysost :21,5000
Mozzarella di Giovanni :34,8000
Röd Kaviar :15,0000
Longlife Tofu :10,0000
Rhönbräu Klosterbier :7,7500
Lakkalikööri :18,0000
Original Frankfurter grüne Soße :13,0000
```

Resim 1.4: Program sonuç ekranı

“foreach” Döngüsü ile kayıtlar üzerinde hareket etmeden evvel “DataContext” nesnesi sizin adınıza bağlantıyı açar ve LINQ ile talep edilen veriler için arka planda SQL sorgusunu veri tabanına iletir, sonuç kümesini getirir. Döngü bitiminde bağlantısını otomatik olarak kapatır.

Örnek 1.12: “Northwind” şirketinde biten ürünlerin tedarikçi şirket isim ve telefon bilgilerini listeleyiniz.

- “Product” varlık sınıfı oluşturunuz.

```
using System.Data.Linq.Mapping;

namespace LINQ2SQLOrnek
{
    [Table(Name = "Products")]
    public class Product
    {
        private int _ProductID;

        [Column(IsPrimaryKey = true, CanBeNull = false)]
        public int ProductID
        {
            get { return _ProductID; }
            set { _ProductID = value; }
        }
    }
}
```



```

private string _ProductName;

[Column(CanBeNull = false)]
public string ProductName
{
    get { return _ProductName; }
    set { _ProductName = value; }
}

private int? _SupplierID;

[Column]
public int? SupplierID
{
    get { return _SupplierID; }
    set { _SupplierID = value; }
}

private short? _UnitsInStok;

[Column(DbType = "smallint")]
public short? UnitsInStock
{
    get { return _UnitsInStok; }
    set { _UnitsInStok = value; }
}
}
}

```

- “Suppliers” varlık sınıfını oluşturunuz.

```

using System.Data.Linq.Mapping;

namespace LINQ2SQLOrnek
{
    [Table(Name="Suppliers")]
    class Suppliers
    {
        int _SupplierID;

        [Column(IsPrimaryKey=true,CanBeNull=false)]
        public int SupplierID
        {
            get { return _SupplierID; }
            set { _SupplierID = value; }
        }
    }
}

```

```

    }
    string _CompanyName;

    [Column(DbType="NVarChar(15)")]
    public string CompanyName
    {
        get { return _CompanyName; }
        set { _CompanyName = value; }
    }
    string _Phone;

    [Column(DbType = "NVarChar(15)")]
    public string Phone
    {
        get { return _Phone; }
        set { _Phone = value; }
    }
}
}
}

```

- “Program.cs” dosyasını aşağıdaki gibi düzenleyiniz.

```

using System;
using System.Data.Linq;
using System.Linq;

namespace LINQ2SQLOrnek
{
    class Program
    {
        static void Main(string[] args)
        {
            DataContext NorthwindContext = new DataContext("Data
Source=.\SQLExpress;Initial Catalog=Northwind; Integrated Security=true;");

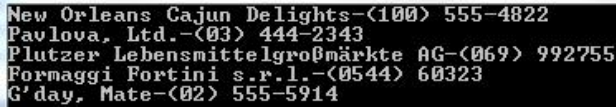
            // Varlık tabloları
            Table<Product> products = NorthwindContext.GetTable<Product>();
            Table<Suppliers> suppliers = NorthwindContext.GetTable<Suppliers>();

            // LINQ sorgusu
            var sorgu = from p in products
                join s in suppliers
                on p.SupplierID equals s.SupplierID
                where p.UnitsInStock==0
                select s;

```

```
// Kayıtların getirilmesi
foreach (Suppliers tedarikci in sorgu)
{
    Console.WriteLine(tedarikci.CompanyName+"-"+tedarikci.Phone);
}
Console.ReadLine();
}
}
```

Program kodunu incelediğinizde veri tabanını senaryo gereği sorgulayabilmek için gereken “Product” ve “Suppliers” varlık sınıflarımız oluşturulmuştur. Bu örneğimizde asıl ilgilenmeniz gereken LINQ sorgusunun işlevidir. Bu yüzden tablolara ait alan tanımlarının bir kısmı yapılmıştır. Program içerisinde öncelikle “DataContext” nesnesi bağlantı bilgisi ile birlikte oluşturulmuştur. “NorthwindContext” nesnesinin “GetTable()” metodları ile varlık sınıflarımız veri tabanı tabloları ile eşleştirilmiştir. Ardında yer alan LINQ sorgusunda “Table” tipinde olan “products” ile “suppliers” tablosu join deyimi ile birbirlerine on deyiminden sonra gelen ifade olduğu gibi “SupplierID” alanı üzerinden ilişki kurulmuştur. Amacımız “product” tablosunda “UnitsInStok” değeri sıfır olan kayıtlar üzerinde “suppliers” (tedarikçi) bilgisine ulaşmaktır. Bu kayıt kümesi “select” deyimi ile var tipinde bulunan sorgu değişkenine verilmektedir. “sorgu” değişkeni üzerinde “foreach” döngüsü ile kayıtlar tek tek okunarak istenilen bilgiler kullanıcıya raporlanmaktadır. Program çalıştırdıktan sonra Resim 1.5’teki ekran çıktısını almalısınız.



```
New Orleans Cajun Delights-(100) 555-4822
Pavlova, Ltd.-(03) 444-2343
Plutzer Lebensmittelgroßmärkte AG-(069) 992755
Formaggi Fortini s.r.l.-(0544) 60323
G'day, Mate-(02) 555-5914
```

Resim 1.5: Program sonuç ekranı

Uygulama üzerinde gerek varlık sınıflarının tanımlamaları, gerekse verinin getirilmesinde kullanılan yöntemler “DataContext” sınıfının metodları yardımı ile yapılmaktadır ve pek bir değişiklik göstermezler. Asıl değişiklik sizin varlık sınıfları üzerinden veri tabanına uygulatacağınız sorgulardır. LINQ burada operasyonel anlamda çok fazla işi sizin yerinize çözmektedir. LINQ deyimlerini ve yazımı belirli bir mantığa dayanır ve bu SQL deyimlerinde olduğu gibi sorgu yorumuna dayalıdır.

Tablo 1.3'te LINQ için kullanılan operatörler ve açıklamaları yer almaktadır.

Operatör	Açıklama
Where	Kısıtlama operatörleri
Select/SelectMany	Seçme operatörleri
Take/Skip/ TakeWhile/SkipWhile	Bölümleme operatörleri
Join/GroupJoin	Birleştirme operatörleri
OrderBy/ThenBy/OrderByDescending/ ThenByDescending	Sıralama operatörleri
Reverse	Ters çevirme operatörü
GroupBy	Gruplama operatörleri
Distinct	Ayıklama operatörü
AsEnumerable	IEnumerable<T> tipine dönüştürme operatörü
ToArray/ToList	Array veya List tipine dönüştürme operatörleri
SequenceEqual	Equality operator checking pairwise element equality
First/FirstOrDefault/Last/LastOrDefault /Single/SingleOrDefault	Eleman operatörleri
DefaultIfEmpty	Element operator replacing empty sequence with default-valued singleton sequence
Range / Repeat	Üretim operatörleri
Any/All	Ölçüm operatörleri
Contains	İçerme operatörü
Count/LongCount	Sayma fonksiyonu operatörleri
Sum/Min/Max/Average	Gruplama fonksiyonu operatörleri

UYGULAMA FAALİYETİ

Aşağıdaki işlem basamaklarını takip ederek faaliyeti gerçekleştiriniz.

İşlem Basamakları	Öneriler
<p>Berlin’de yaşayan müşterilerin şirket isimleri, adres, telefon ve fax bilgilerini listeleyiniz.</p> <ul style="list-style-type: none">➤ İsim alanları (namespace) bildirimini yapınız.➤ SqlConnection nesnesi oluşturunuz.➤ SqlCommand nesnesi oluşturunuz.➤ SqlCommand nesnesinin .CommandText, Connection, CommandType özelliklerini veriniz.➤ SqlDataReader nesnesi hazırlayınız.➤ Bağlantıyı açınız.➤ Komutu yürütünüz.➤ While döngüsü altında DataReader nesnesinden kayıtları raporlayınız.➤ Bağlantıyı kapatınız.	<ul style="list-style-type: none">➤ Northwind veri tabanında “customer” tablosu müşterilerin kayıtlı olduğu tablodur. Uygulama faaliyetinin konusu gereği “Customer” tablosunu sorgulayınız.➤ Yürütülecek komut metni “SELECT CompanyName, Address, Phone FROM Customer WHERE City=’Berlin’” şeklinde olmalıdır.➤ SqlDataReader nesnesini “new” yapıcısı ile oluşturmanıza gerek yoktur.➤ Komut yürütümünde ExecuteReader() metodunu uygulayınız.➤ Uygulama kodlarını hata denetim mekanizması altında denetleyiniz.➤ Uygulama için Örnek 1.8’den faydalanabilirsiniz.
<p>Berlin’de yaşayan müşterilerin şirket isimleri, adres, telefon ve fax bilgilerini DLINQ kullanarak listeleyiniz.</p> <ul style="list-style-type: none">➤ Customer varlık sınıfını oluşturunuz.➤ DataContext nesnesini oluşturunuz.➤ “Customer” tablosunu context nesnesinin GetTable() metodu ile oluşturunuz.➤ LINQ sorgusunu yazınız.➤ Döngü altında sorgu sonucunu ekrana raporlayınız.	<ul style="list-style-type: none">➤ Customer tablosu içinde mutlaka Address, Phone, CompanyName özelliklerini tanımlayınız.➤ Bağlantı bilgisini context nesnesi örneklerken kurucu parametresi olarak vermeyi unutmayınız.➤ LINQ sorgunuz; var q=from c in Customer where c.City==’Berlin’ select c; şeklinde düzenleyiniz.➤ Uygulama faaliyeti için Örnek 1.12’den faydalanabilirsiniz.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyunuz ve doğru seçeneği işaretleyiniz.

1. Aşağıdakilerden hangisi veri erişim teknolojilerinden birisi değildir?
A) ADO
B) OLE DB
C) RDO
D) DAC
2. Aşağıdakilerden hangisi Northwind için doğrudur?
A) Sanal bir şirkete ait verileri barındıran bir veri tabanıdır
B) Veri erişim türüdür
C) Veri tabanı bağlantı nesnesidir
D) Hiçbiri
3. Access ile oluşturulmuş bir veri tabanına bağlantı kurabilmek için aşağıdaki sınıflardan hangisini kullanmamız gerekir?
A) System.Odbc
B) System.Data.SqlClient
C) System.Data.OleDb
D) System.Data.Oracle
4. Aşağıdakilerden hangisi bağlantı bilgisinde kullanılan parametrelerden birisi değildir?
A) ConnectionTimeout
B) Properties
C) Provider
D) UserId
5. Aşağıdakilerden hangisi sorgu sonucu kayıt kümesi bekleniyorsa yürütülmesi gereken metoddur?
A) ExecuteScalar()
B) ExecuteNonQuery()
C) ExecuteReader()
D) ExecuteXMLReader()

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme faaliyetine geçiniz.

ÖĞRENME FAALİYETİ-2

AMAÇ

Veri tabanında ihtiyacınıza yönelik işlemleri yapabileceksiniz.

ARAŞTIRMA

- Çeşitli yazılım geliştirme platformları için üretilmiş ORM araçlarını araştırınız.
- ORM Araçlarının program yazarına sunduğu avantaj ve dezavantajlar nelerdir? Araştırınız.
- GUI (Graphical User Interface) için veriyi göstermek için kullanılan arabirim denetimleri nelerdir? Araştırınız.

2. VERİ İŞLEMLERİ

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.

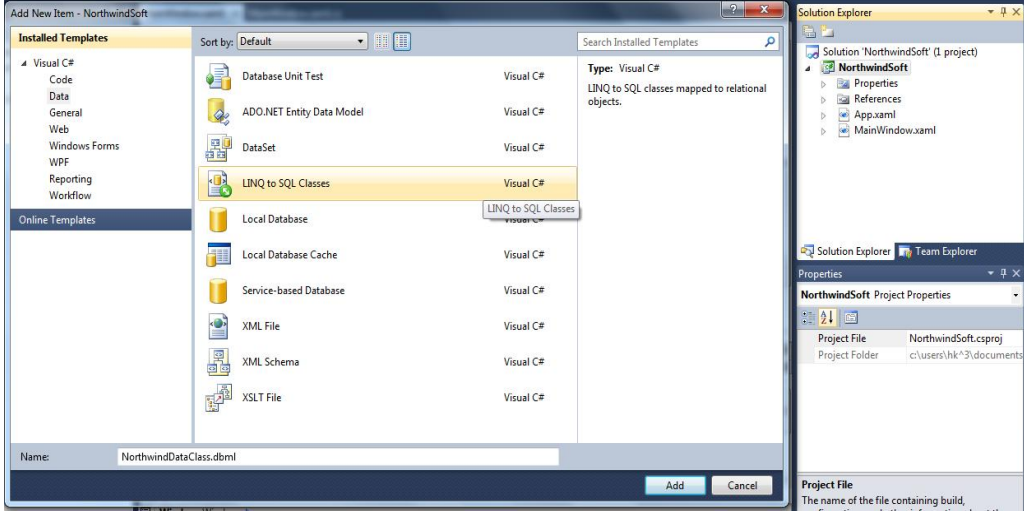
2.1. DLINQ İle Veri Bağlama

DLINQ size veri tabanı varlık nesnelərini oluşturmada, kullanmada, seçme, ekleme, silme, güncelleme işlemlerini birer metot olarak sunmada ve bütün bunları basit bir sihirbaz yardımı ile grafik ortamda hızlı olarak gerçekleştirmenize 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 aşağıdaki işlem adımlarını takip ediniz.

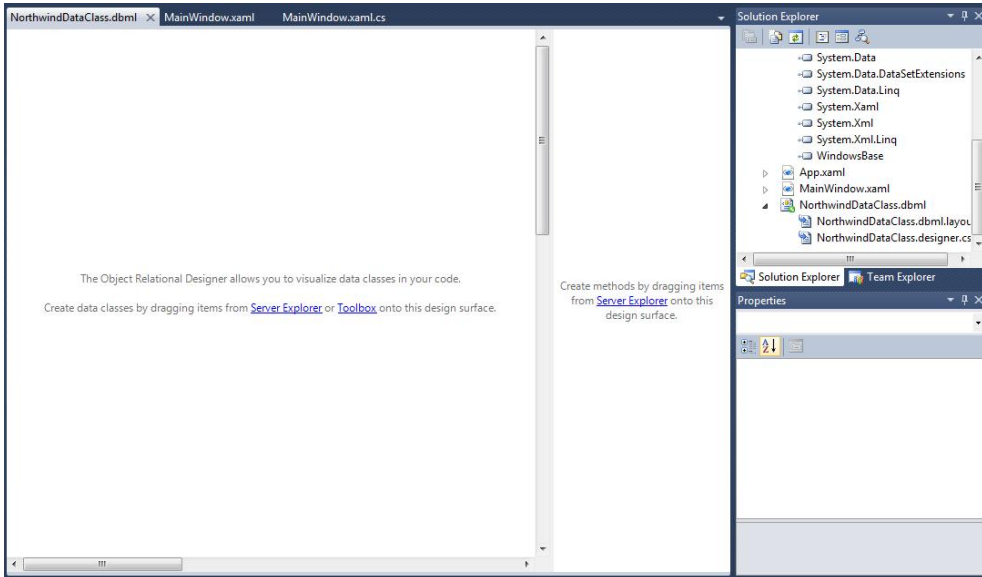
- Öncelikle Visual Studio ortamında yeni bir WPF uygulaması (WPF Application) oluşturunuz.

- Solution Explorer paneli içinde uygulama adı üzerinde fare sağ tuş→Add→New Item seçimini yapınız.
- Ekranı gelen “New Item” penceresinden “LINQ to SQL Classes” seçiniz. Name kısmına üretilecek .dbml uzantılı dosyaya vermek istediğiniz ismi yazınız. Bu uygulamada “NorthwindDataClasses” örnek ismi verilmiştir. Ekle (Add) butonuna basınız.



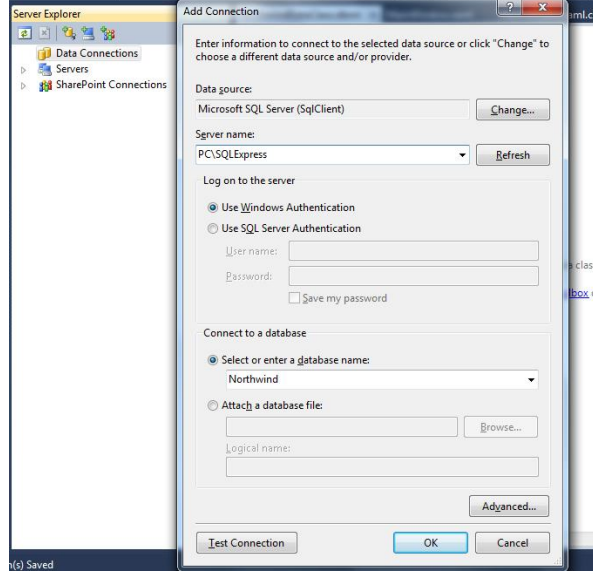
Resim 2.1: LINQ to SQL Class ekleme işlemi

- Ekranı “NorthwindDataClass.dbml” dosyasına ait içi şu an için boş olan bir grafik tasarım penceresi Resim 2.2’de görüldüğü gibi gelecektir.



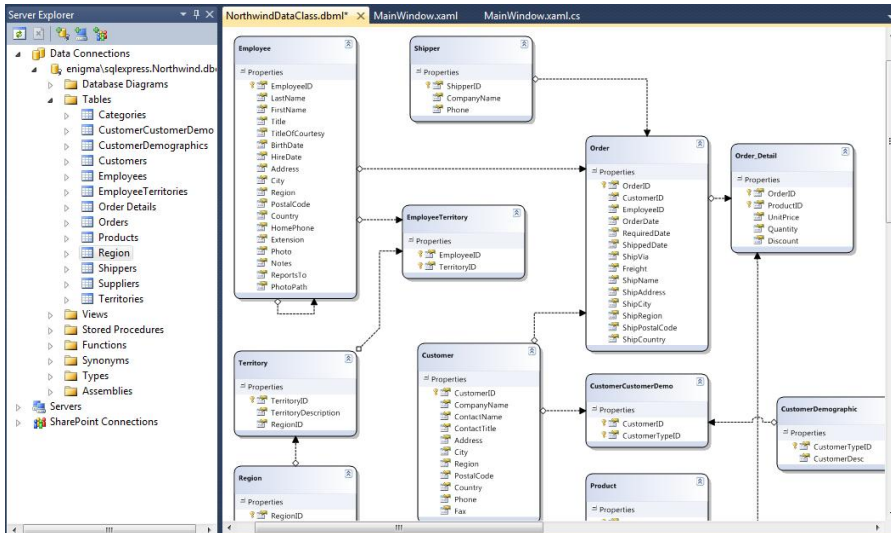
Resim 2.2: “NorthwindDataClass.dbml” dosyası görünümü

- Northwind veri tabanı tablolarını grafik ekran üzerinde oluşturmak için öncelikle sunucu ile bağlantı kurmalısınız. “Server Explorer” panelinde “Data Connections” öğesi üzerine Mouse sağ tuş→”Add Connection” seçimini yapınız. Resim 2.3’te görülen pencere ekrana gelecektir. Sunucu adı (Server Name) ve veri tabanı adı (database name) bilgilerinizi girdikten sonra “Tamam” (Ok) butonuna basınız.



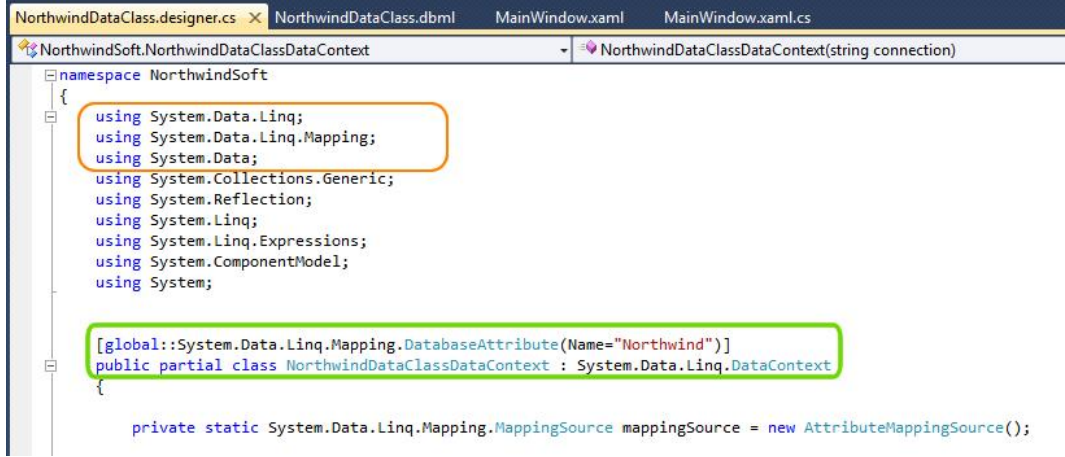
Resim 2.3: “Add Connection” penceresi

- Server Explorer penceresinden ulaştığımız “Northwind” veri tabanı tablolarını uygulamanızın “NortwindDataClass.dbml” dosyasına içine sürükleyiniz. Veri tabanı şemasını Resim 2.4’te olduğu gibi oluşturulduğunu gözlemleyiniz.



Resim 2.4: Veri tabanı şeması

Yukarıdaki işlem adımlarının ardından “NorthwindDataClass.dbml” dosyasını kaydetmeyi unutmayınız. Bütün bu işlem adımları altında Resim 2.5’te görülen “NorthwindDataClass.designer.cs” kaynak kod dosyasının otomatik olarak oluşumunu sağlamak amacıyla yapılmıştır. Bu dosyayı açıp kodları incelediğinizde;



```
namespace NorthwindSoft
{
    using System.Data.Linq;
    using System.Data.Linq.Mapping;
    using System.Data;
    using System.Collections.Generic;
    using System.Reflection;
    using System.Linq;
    using System.Linq.Expressions;
    using System.ComponentModel;
    using System;

    [global::System.Data.Linq.Mapping.DatabaseAttribute(Name="Northwind")]
    public partial class NorthwindDataClassDataContext : System.Data.Linq.DataContext
    {
        private static System.Data.Linq.Mapping.MappingSource mappingSource = new AttributeMappingSource();
    }
}
```

Resim 2.5: “NorthwindDataClass.designer.cs” kaynak kodu

DLINQ için isim alan bildirimlerinin yapıldığını, “Northwind” veri tabanı isimlendirme nitelemesinin hemen altında “DataContext” sınıfından kalıtılmış olarak oluşturulan “NorthwindDataContext” isimli bir sınıfın oluşturulduğunu görebilirsiniz. Bu sınıf “Northwind” veri tabanının şablonudur.

“NorthwindDataContext” sınıf yapısında bulunan bileşenleri Resim 2.6’da inceleyiniz.



```
public System.Data.Linq.Table<Region> Regions...
public System.Data.Linq.Table<Product> Products...
public System.Data.Linq.Table<Order> Orders...
public System.Data.Linq.Table<Order_Detail> Order_Details...
public System.Data.Linq.Table<EmployeeTerritory> EmployeeTerritories...
public System.Data.Linq.Table<Employee> Employees
{
    get
    {
        return this.GetTable<Employee>();
    }
}
public System.Data.Linq.Table<Customer> Customers...
```

Resim 2.6: “NorthwindDataContext” sınıf yapısı

Veri tabanı tablolarının “DataContext” sınıfı üzerinden “GetTable()” metodu ile alınmış olduğunu gözlemleyiniz.

Resim 2.7’de varlık sınıflarının tanımlandığı kodları inceleyiniz. Sınıfın sahip olduğu özellikler veri tabanı tablosunun alanları şeklinde hazırlanmış, gereken nitelermeler yapılmıştır. O/RM aracını kullanmadan önce bu sınıfların yazılması programcının yapması gereken işler arasında olduğunu bir önceki öğrenme faaliyetinde öğrendiniz. Bu aracın ne kadar fazla iş yükünü sizin adınıza yaptığı ortadadır.

```

NorthwindDataClass.designer.cs X NorthwindDataClass.dbml MainWindow.xaml MainWindow.xaml.cs
NorthwindSoft.Product
    }
    [global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.Products")]
    public partial class Product : INotifyPropertyChanging, INotifyPropertyChanged
    {
        private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);
        private int _ProductID;
        private string _ProductName;
        private System.Nullable<int> _SupplierID;
        private System.Nullable<int> _CategoryID;
        private string _QuantityPerUnit;
        private System.Nullable<decimal> _UnitPrice;
    }

```

Resim 2.7: “NorthwindDataContext”e ait bir varlık sınıfı oluşumu

Burada varlık sınıfının “INotifyPropertyChanging” ve “INotifyPropertyChanged” arayüzlerinden uygulanmış olmasının bir anlamı vardır. Her iki arayüz, Resim 2.8’de görüldüğü gibi varlık sınıfı içine birer olay (event) tanımlaması dahil ederler.

```

public event PropertyChangingEventHandler PropertyChanging;
public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()
{
    if ((this.PropertyChanging != null))
    {
        this.PropertyChanging(this, emptyChangingEventArgs);
    }
}

protected virtual void SendPropertyChanged(String propertyName)
{
    if ((this.PropertyChanged != null))
    {
        this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
    }
}

```

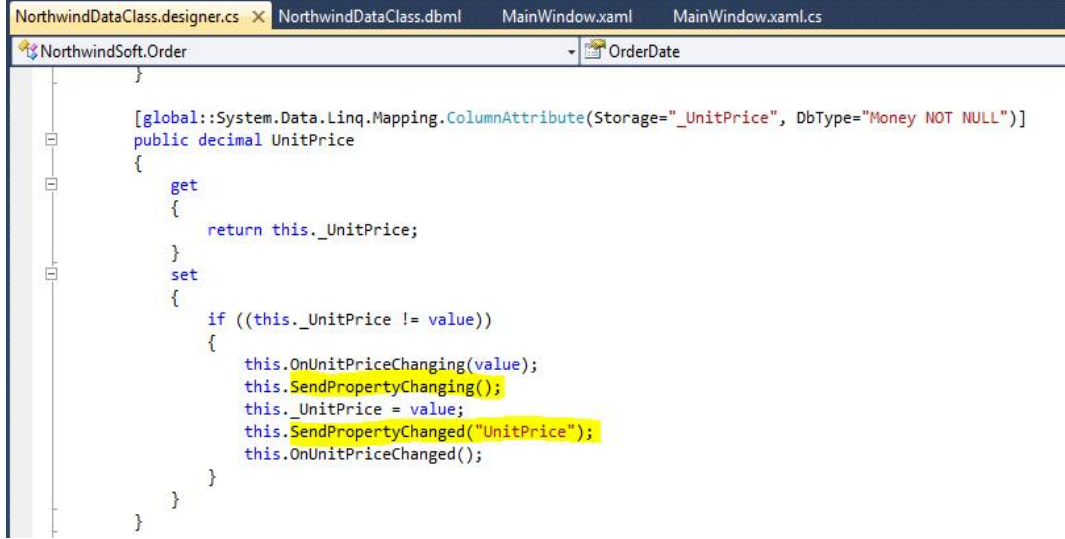
Resim 2.8: Özellik değişimini bildiren olay tanımları

Eğer varlık nesnesinin sahip olduğu bir değer değişirse ki bu kullanıcı tarafından da yapılabilir veya veri kaynağından da gelebilir, bu değişimi dinleyen olay (event) tanımları, ilgili özelliğin (property) “set” blokunda Resim 2.9’da görüldüğü gibi uyarılır. Bu olaylara abone olmuş diğer sınıflar bu uyarıları dinler ve değişimlerden haberdar olurlar.

Sunum katmanında kullanılacak WPF kütüphanesinde bulunan kullanıcı arabirim denetimleri (Textbox, listbox, combobox, datagridview vs.) bu tip dinlemeleri gerçekleştirerek içeriği bakımından bağlı (data binding) buldukları özelliklerin taşıdığı

güncel bilgileri gösterebilirler. Aynı şekilde kullanıcı tarafından bu arabirim denetimlerinde yapılmış değişiklikler de nesnelere 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.



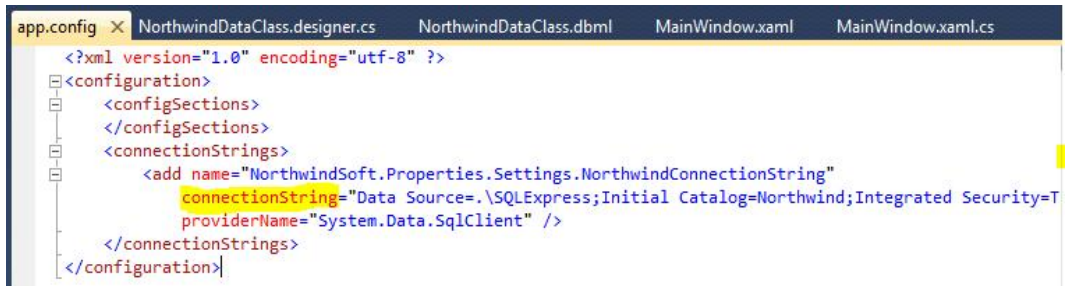
```
NorthwindDataClass.designer.cs | NorthwindDataClass.dbml | MainWindow.xaml | MainWindow.xaml.cs
NorthwindSoft.Order | OrderDate

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="_UnitPrice", DbType="Money NOT NULL")]
public decimal UnitPrice
{
    get
    {
        return this._UnitPrice;
    }
    set
    {
        if ((this._UnitPrice != value))
        {
            this.OnUnitPriceChanging(value);
            this.SendPropertyChanging();
            this._UnitPrice = value;
            this.SendPropertyChanged("UnitPrice");
            this.OnUnitPriceChanged();
        }
    }
}
```

Resim 2.9: PropertyChanged ve PropertyChanging olaylarının başlatılması

Bu konu hakkında daha fazla bilgi için "Nesne tabanlı programlamada temsilciler ve olaylar" modülünü tekrar inceleyiniz. Her iki arayüzü kendi varlık sınıflarınıza da uygulayabilirsiniz.

"NorthwindDataContext" sınıfı veri kaynağına bağlantı bilgisini sınıf dosyasında değil uygulama ayarlarının "xml" formatında saklandığı "app.config" isimli özel bir dosyada saklanmaktadır. Resim 2.10'da bu dosyanın yapısını görmekteyiz. Çok değerli olan bağlantı bilgisi "config" dosyasında saklanması güvenlik açısından doğru bir davranış olacaktır. Web ortamında bu tip bilgiler "web.config" dosyasında korunurlar.



```
app.config | NorthwindDataClass.designer.cs | NorthwindDataClass.dbml | MainWindow.xaml | MainWindow.xaml.cs

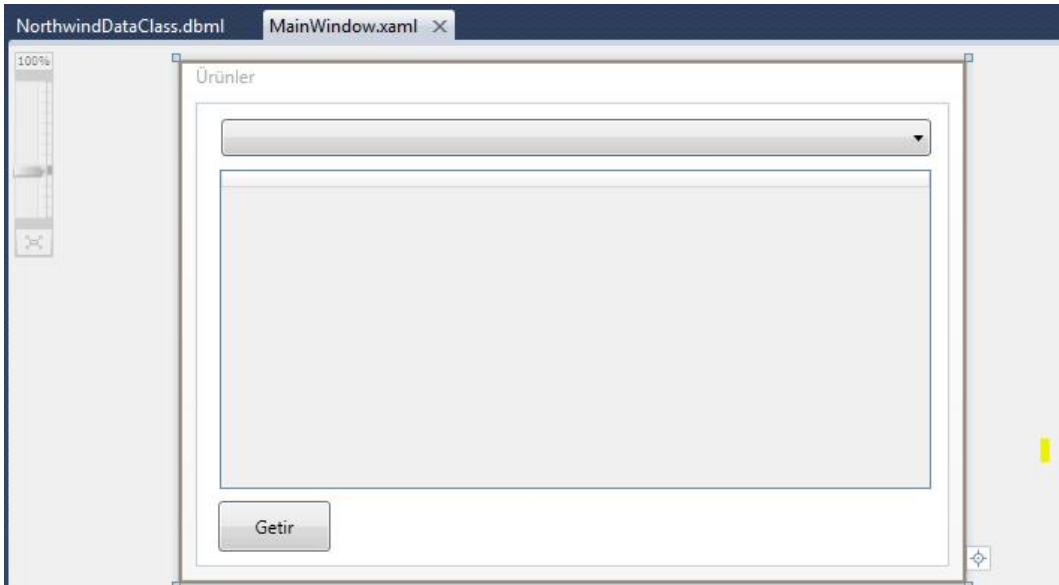
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
  </configSections>
  <connectionStrings>
    <add name="NorthwindSoft.Properties.Settings.NorthwindConnectionString"
        connectionString="Data Source=.\SQLEXPRESS;Initial Catalog=Northwind;Integrated Security=TRUE"
        providerName="System.Data.SqlClient" />
  </connectionStrings>
</configuration>
```

Resim 2.10: app.config dosyası içeriği

Artık sunum katmanı için uygulamanın kullanıcı arabirimini tasarlayabilirsiniz. Bunun için yapmanız gereken uygulama içinde bulunan "MainWindow.xaml" dosyasını açın ve form içine Tablo 2.1'de verilen arabirim denetimlerini Resim 2.11'de gösterildiği gibi yerleştiriniz.

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

Tablo 2.1: Sunum katmanı için denetim listesi



Resim 2.11: Ekran tasarımı

Resim 2.11'de bulunan ekran tasarımı için formun "xaml" kodu aşağıda verildiği gibi düzenleyiniz.

```
<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" />
    </Grid>
</Window>
```

```

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>

```

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” bileşenin “ItemSource” 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 işlemi kod tarafında gerçekleştirilmelidir.

```
<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” 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” 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>
```

“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 metod 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.

```
<Button Click="btnGetir_Click" Content="Getir"...
```


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.

```
using System.Collections;
using System.ComponentModel;
using System.Windows;
using System.Windows.Controls;
namespace NorthwindSoft
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();

            NorthwindDataClassDataContext context;
            BindingList<Product> productsInfo;
            Category cat;

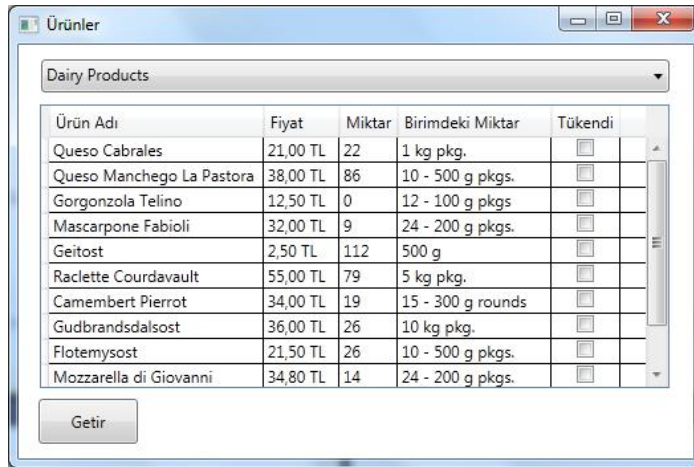
private void btnGetir_Click(object sender, RoutedEventArgs e)
        {
            // Getir butonuna tıklandığında çalışacak metoddur.
            // Veri tabanı örnekleme oluşturulmaktadır.
            context = new NorthwindDataClassDataContext();
            // Combobox'ın bağlanacağı içerik Kategori tablosu olarak belirlenir.
            cmbKategori.DataContext = context.Categories;
        }

private void cmbKategori_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            // Combobox seçimi değiştiğinde çalışacak metoddur.
            // Combobox'da seçili öğe 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;
        }
    }
}
```

Veri tabanına bağlanmak ve kategori ile ürünler tablosunu elde etmek için “NorthwindDataClassDataContext” sınıfından “context” isimli nesne örnekleme yapılmıştır. “cmbKategori” isimli “combobox” bileşenin 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 tıkladığında kategori listesi “combobox” bileşenine yüklenmiş olacaktır.

“Combobox” listesinde bir seçim değişikliği meydana geldiğinde “cmbKategori_SelectionChanged” metodu çalışacaktır. Öncelikle Kategori tipinde olan “cat” değişkeni “combobox” bileşenin seçili öğesini 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” bileşenin veri kaynağının işaret edildiği “ItemSource” özelliğine bağlanmıştır. Bu sayede daha önce “xaml” tarafında “DataGrid” için “ItemSource={Binding}” ile veri kaynağı belirlenmiş olur.

Uygulamayı çalıştırınız ve kategori seçimi yaptığınızda Resim 2.12’de olduğu gibi ilgili ürünlerin geldiğini gözlemleyiniz.



Resim 2.12: Uygulamanın çalışma anı ekran görüntüsü

Örnek 2.1. Müşteri listesi üzerinde arama yapılması ve bulunan müşterinin satın aldığı ürünleri gösteren uygulamayı gerçekleştiriniz.

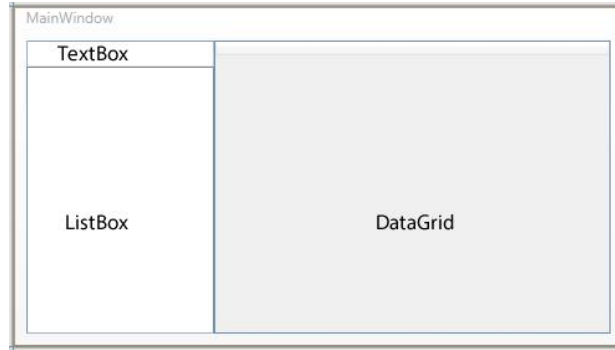
Uygulamayı gerçekleştirmek için aşağıdaki işlem adımlarını gerçekleştiriniz.

- Yeni bir WPF uygulaması oluşturunuz.
- Form tasarımında kullanılacak bilşen listesi Tablo 2.2’de verilmiştir.

Kullanıcı Arabirim Denetimi	İsim (Name) Özelliği
TextBox	txtAra
DataGrid	dgUrun
ListBox	IstMusteri

Tablo 2.2: Uygulama bileşen listesi

- Resim 2.13’te verilen ekran tasarımını gerçekleştiriniz.



Resim 2.13: Ekran tasarımı

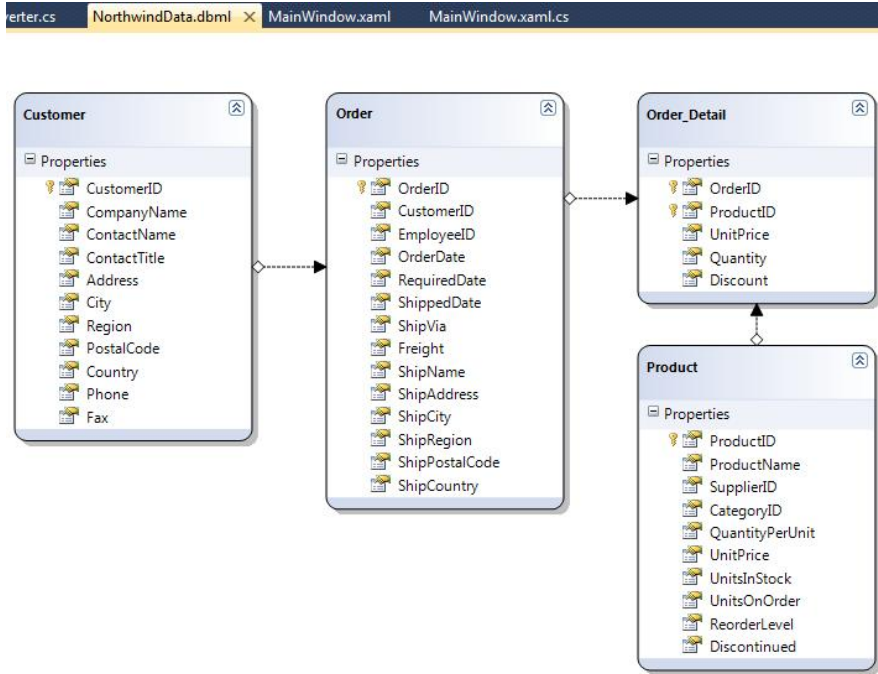
Ekran tasarımını gösteren xaml kodları aşağıda verilmiştir.

```
<Window x:Class="BulGetir.MainWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="MainWindow" Height="275" Width="492">
  <Grid>
    <Grid.ColumnDefinitions>
      <ColumnDefinition Width="162*" />
      <ColumnDefinition Width="341*" />
    </Grid.ColumnDefinitions>
    <StackPanel Name="stackPanel1">
      <TextBox x:Name="txtAra" KeyUp="txtAra_KeyUp"></TextBox>
      <ListBox x:Name="IstMusteri" ItemsSource="{Binding}"
        DisplayMemberPath="CompanyName"
        MouseLeftButtonUp="IstMusteri_MouseLeftButtonUp" Height="290"></ListBox>
    </StackPanel>
    <DataGrid AutoGenerateColumns="True" Grid.Column="1" Margin="0"
      Name="dgUrunler"/>
    </Grid>
  </Window>
```

“txtAra” isimli bileşenin “KeyUp”, olay (event) bildiriminin amacı aranmak istenen müşteri ismi yazılırken listbox içeriğinin güncellenmesini sağlamaktır. Günümüz internet

arama motorlarında oldukça popüler olan bu yaklaşım için olaya (event) bağlanan “txtAra_KeyUp” metodunda yazılacaktır. Aynı şekilde “ListBox” için “MouseLeftButtonUp” olay bildirimi ile liste kutusundan seçilen müşteriye satışı yapılan ürün listesi “DataGrid” içinde güncellenecektir.

- Veri katmanının oluşturulması için uygulamanıza “LINQ to SQL” bileşeni ekleyiniz. Bunun için yapmanız gereken “Solution Explorer” panelinde proje ismi üzerinde Mouse sağ tuş→Add→New Item ile açılan diyalog penceresinde “LINQ to SQL” seçimini yapınız. Oluşturulmak istenen “dbml” uzantılı dosyaya isim veriniz (Uygulama için “NorthwindData.dbml” ismi tercih edilmiştir).
- Uygulama için müşteri (customer), ürünler (product), satışlar (orders) ve satış detaylarının yer aldığı (order details) tablolarını server explorer panelinden sürükleyip bırakınız. Elde edeceğiniz şema resim 2.14’te gösterildiği gibidir.



Resim 2.14: “NortwindData.dbml” dosyası

- “MainWindow1.xaml.cs” dosyasına uygulama kodunu oluşturmaya başlayabilirsiniz. İlk olarak “NorthwindDataDataContext” sınıfından “new” yapıcısı ile context nesnesini örnekleyiniz.

```
NorthwindDataDataContext context = new NorthwindDataDataContext();
```

- Uygulama ilk açıldığında liste kutusunda (listbox) müşteri listesinin hazır olması gerekmektedir. Bundan dolayı form yüklendiğinde (Window Loaded) liste kutusunun veri bağlaması gerçekleştirilmiş olmalıdır.

```
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    lstMusteri.ItemsSource = context.Customers;
}
```

- Uygulama ekranında yer alan metin kutusuna (TextBox) kullanıcının girdiği ifade ile başlayan kayıtları getirmek üzere LINQ sorgusunu oluşturup sorgu sonucunu liste kutusunun (ListBox) "ItemSource" özelliğine bağlayınız. Böylelikle liste kutusunun veri kaynağını belirlemiş olacaksınız.

```
private void txtAra_KeyUp(object sender, KeyEventArgs e)
{
    var sorgu1 = from c in context.Customers
                where c.CompanyName.StartsWith(txtAra.Text)
                orderby c.CompanyName
                select c;
    lstMusteri.ItemsSource = sorgu1;
}
```

- Liste kutusundan bir seçim yapıldığında müşteri kimlik bilgisi üzerinden (CustomerID) satış (Orders), satış detayları (Order Details) ve ürün bilgilerinin (Product) getirilmesi gerekmektedir. Bütün bu ilişkili tablolardan sorgu sonucu olarak ekrana yalnızca getirmek istediğiniz bilgileri seçmek için LINQ sorgusunda "select" deyiminden sonra "new" deyimi ile sorgunuzu sadeleştiriniz. Elde edilen sorgu sonucu "DataGrid" bileşeninin "ItemSource" özelliğine bağlayınız.

```
private void lstMusteri_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)
{
    Customer secilen = (Customer)lstMusteri.SelectedValue;
    var sorgu2 = from o in context.Orders
                join d in context.Order_Details
                on o.OrderID equals d.OrderID
                where o.CustomerID == secilen.CustomerID
                select new { d.Product.ProductName, d.UnitPrice, d.Quantity };

    dgUrunler.ItemsSource = sorgu2;
}
```

Uygulamanın tüm kodu aşağıda verildiği gibidir.

```
using System.Linq;
using System.Windows;
using System.Windows.Input;

namespace BulGetir
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();

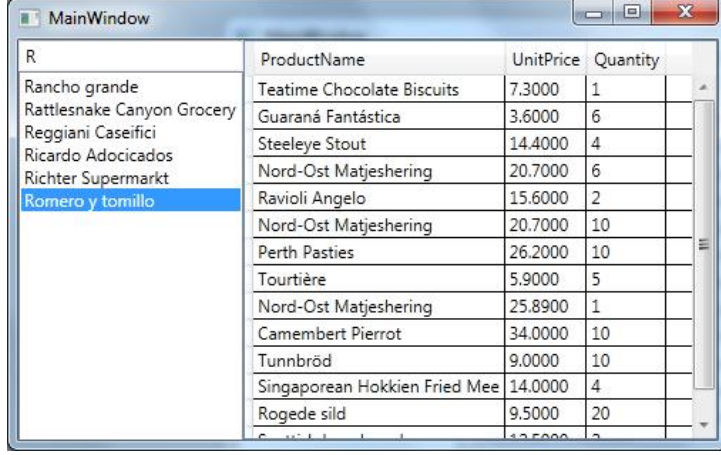
            NorthwindDataDataContext context = new NorthwindDataDataContext();

            private void txtAra_KeyUp(object sender, KeyEventArgs e)
            {
                var sorgu1 = from c in context.Customers
                    where c.CompanyName.StartsWith(txtAra.Text)
                    orderby c.CompanyName
                    select c;
                lstMusteri.ItemsSource = sorgu1;
            }

            private void Window_Loaded(object sender, RoutedEventArgs e)
            {
                lstMusteri.ItemsSource = context.Customers;
            }

            private void lstMusteri_MouseLeftButtonUp(object sender, MouseButtonEventArgs
e)
            {
                Customer secilen = (Customer)lstMusteri.SelectedValue;
                var sorgu2 = from o in context.Orders
                    join d in context.Order_Details
                    on o.OrderID equals d.OrderID
                    where o.CustomerID == secilen.CustomerID
                    select new { d.Product.ProductName, d.UnitPrice, d.Quantity };
                dgUrunler.ItemsSource = sorgu2;
            }
        }
    }
}
```

- Uygulamayı çalıştırınız ve Resim 2.15’te gösterildiği gibi test ediniz.



R	ProductName	UnitPrice	Quantity
Rancho grande	Teatime Chocolate Biscuits	7.3000	1
Rattlesnake Canyon Grocery	Guaraná Fantástica	3.6000	6
Reggiani Caseifici	Steeleye Stout	14.4000	4
Ricardo Adocicados	Nord-Ost Matjeshering	20.7000	6
Richter Supermarkt	Ravioli Angelo	15.6000	2
Romero y tomillo	Nord-Ost Matjeshering	20.7000	10
	Perth Pasties	26.2000	10
	Tourtière	5.9000	5
	Nord-Ost Matjeshering	25.8900	1
	Camembert Pierrot	34.0000	10
	Tunnbröd	9.0000	10
	Singaporean Hokkien Fried Mee	14.0000	4
	Rogede sild	9.5000	20

Resim 2.15: Uygulama testi

2.2. DLINQ İle Veri Güncelleme

Varlık nesnelere uygulamanın çalışma sürecinde hafızada yer alır. Bunlar üzerinde yapmış olduğunuz değişiklik veri tabanı tarafından algılanmaz. Yapılan değişikliklerin veri tabanı tarafına geçirilmesini istiyorsanız veri tabanı nesnesi “SubmitChanges()” metodunu çağırmanız yeterlidir. Bu metod kayıtların veri tabanı üzerinde güncellenebilmesi için SQL sorgusu oluşturur ve veri tabanına uygular.

Örnek 2.2. 1 numaralı ürünün adını “Chains1” olarak güncelleyiniz.

```
NorthwindDataClassDataContext context = new NorthwindDataClassDataContext();
Product urun = (from p in context.Products
                where p.ProductID == 1
                select p).Single<Product>();

urun.ProductName = "Chains1";
context.SubmitChanges();
```

Öncelikle “NorthwindDataClassDataContext” sınıfı üzerinden “context” nesnesi örnekleme yapmalısınız. Ardından güncellenmek istenen kayıt bilgisi LINQ sorgusu ile getirilmelidir. Ürün numarası (ProductID) 1 olan kaydı (Single) “Product” tipindeki ürün nesnesine verilmiştir. Sonuçta elde edilen ürünün ismi (ProductName) “Chains1” olarak yeniden düzenlenmiş ve yapılan güncelleme işlemi “SubmitChanges()” metodu ile context nesnesi aracılığı ile veri tabanına uygulanmıştır.

Örnek 2.3. “Tokyo Traders” şirketine ait ürünlerin stok miktarını 0 olarak gerçekleştiriniz.

```
NorthwindDataClassDataContext context = new NorthwindDataClassDataContext();
    var sorgu=from s in context.Suppliers
        join p in context.Products
        on s.SupplierID equals p.SupplierID
        where s.CompanyName=="Tokyo Traders"
        select p;

    foreach(Product pr in sorgu)
    {
        pr.UnitsInStock = 0;
    }
    context.SubmitChanges();
```

Bu tip bir sorguda birden fazla ürünün değeri güncellenecektir. LINQ sorgusu 2 tablo üzerinden yapılmaktadır. Dolayısıyla “join” deyimi ile “Suppliers” (Tedarikçiler) ve “Product” (Ürünler) tablosu arasında “SupplierID” alanı üzerinden eşleşen (equals) kayıtlar sorgu içine alınmıştır. “Foreach” döngüsü ile elde edilen sonuç kümesine ait kayıtlar tek tek hareket ederek üzerinde gereken değişiklik yapılmış, “context” nesnesinin “SubmitChanges()” metodu ile veri tabanı üzerine uygulanmıştır.

“DataContext” nesnesi üzerinde yapılan ve veri tabanına güncellenmemiş değişiklikleri eski haline getirmek için “Refresh()” metodunu aşağıda verildiği şekilde kullanabilirsiniz.

```
context.Refresh(RefreshMode.OverwriteCurrentValues, context.Products);
```

Refresh() metodu kendine iki farklı parametre alır. Birincisi RefreshMode numaralandırma tipidir. “OverwriteCurrentValues” değeri varlık nesnelere üzerindeki değişiklikleri iptal eder. Diğer bir parametresi ise hangi varlık sınıfının bu iptali yapacağı belirtir.

2.3. Veri Ekleme ve Silme

DLINQ ile varlık koleksiyonu üzerine yeni bir öge ekleyebilir veya çıkarabilirsiniz. Yeni bir öge eklemek için ilgili koleksiyonun “InsertOnSubmit()” metodu çıkarmak için “Remove()” metodu kullanılır.

Örnek 2.4. “Oruvar Exp” adında yeni bir nakliyeci kaydı yapınız.

```
NorthwindDataClassDataContext context = new NorthwindDataClassDataContext();
    Shipper nakliye = new Shipper() { CompanyName = "Oruvar Exp",
        Phone = "(505) 555-7675" };
    context.Shippers.InsertOnSubmit(nakliye);
    context.SubmitChanges();
```


Eklenecek olan kayıt ait olacağı varlığın tipinde bir örneği “new” yapıcısı ile oluşturulur. Varlık nesnesine ait özelliklere (property) yeni kayıt bilgileri verilir. İlgili varlık koleksiyonunun “InsertOnSubmit()” metoduna parametre verilmek suretiyle ekleme işlemi gerçekleştirilir.

Örnek 2.5 : “Oruvar Exp” adındaki nakliyecinin kaydını siliniz.


```
NorthwindDataClassDataContext context = new NorthwindDataClassDataContext();
Shipper nakliye = (from s in context.Shippers
                  where s.CompanyName == "Oruvar Exp"
                  select s).Single<Shipper>();
context.Shippers.DeleteOnSubmit(nakliye);
context.SubmitChanges();
```

Öncelikle silinecek kayıt varlık setinden LINQ sorgusu ile elde edilir. İlgili varlık koleksiyonunun “DeleteOnSubmit()” metoduna parametre olarak verilmek suretiyle silme işlemi gerçekleştirilir.

Ekleme ve silme işlemlerinin ardından “SubmitChanges()” metodu “context” nesnesine uygulanarak değişiklikler veri tabanına uygulanır.

UYGULAMA FAALİYETİ

Aşağıdaki işlem basamaklarını takip ederek faaliyeti gerçekleştiriniz.

İşlem Basamakları	Öneriler
<ul style="list-style-type: none">➤ Kategorilerin listesi üzerinde seçim yapıldığında ekrana detaylarını getiren uygulamayı gerçekleştiriniz.➤ Yeni bir WPF uygulaması oluşturunuz.➤ “LINQ to SQL” bileşenini projeye ekleyiniz.➤ Uygulama arayüzünü tasarlayınız.➤ DataContext nesnesini örnekleyiniz.➤ Uygulama penceresi açıldığında kategori varlık sınıfını (Categories) “DataGrid” bileşenine bağlayınız.➤ Formda bulunan diğer kullanıcı arabirim denetimlerine veri bağlama (Data binding) işlemini yapınız.➤ Uygulamayı test ediniz.	<ul style="list-style-type: none">➤ Uygulama arayüzü  <ul style="list-style-type: none">➤ Kategori ismini görüntüleyecek metin kutusunun veri bağlamasını “<TextBox Text="{Binding Path=CategoryName}"...” şeklinde yapınız.➤ Kategori açıklamasını görüntüleyecek metin kutusunun veri bağlamasını “<TextBox Text="{Binding Path=Categorydescription}"” şeklinde yapınız.➤ Ürün resmini görüntüleyecek veri bağlamasını “<Image Source="{Binding Path=Picture}"” şeklinde yapınız.➤ Metin ve resim bileşenlerini kapsayan “grid” nesnesine “<Grid x:Name="grid1">” şeklinde isim veriniz.➤ Uygulama penceresi yüklendiğinde “DataGrid” ve “Grid” bileşenlerinin veri kaynağı olarak “context” nesnesinin “category” varlık sınıfını bağlayınız.➤ “dataGrid1.ItemsSource=context.Categories;➤ grid1.DataContext =context.Categories;”

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyunuz ve doğru seçeneği işaretleyiniz.

1. Aşağıdakilerden hangisi ORM aracı değildir?
A) EntityFramework
B) DLINQ
C) Server Explorer
D) PA
2. Aşağıdakilerden hangisinde DLINQ için söylenemez?
A) Program yazarının iş yükünü hafifletir.
B) .NET kodunu SQL deyimlerine dönüştürür.
C) En performanslı veri erişim tekniğidir.
D) ORM aracıdır.
3. Aşağıdakilerden hangisi LINQ sorgusuna ait deyim değildir?
A) for
B) from
C) where
D) select
4. Aşağıdakilerden hangisi güncelleme metodudur?
A) DeleteOnSubmit()
B) SaveChanges()
C) InsertOnSubmit()
D) SubmitChanges()
5. Aşağıdakilerden hangisi "DataGrid" bileşenin veri kaynağı bildirilen özelliğidir?
A) DisplayMemberPath
B) ItemSource
C) DataSource
D) DataBinding
6. Aşağıdaki DLINQ sorgularından hangisi yanlıştır?
A) From o in context.Product where o.ProductID==1 select;
B) From Customer Where Customer.CustomerID==1 Select;
C) from s in context.Suppliers join p in context.Products on s.SupplierID equals p.SupplierID selectp;
D) from c in q.ToSequence() select new MyType { Name = DoNameProcessing(c.ContactName), Phone = DoPhoneProcessing(c.Phone) };

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise "Modül Değerlendirme"ye geçiniz.

MODÜL DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyunuz ve doğru seçeneği işaretleyiniz.

1. Aşağıdakilerden hangisi uygulamaya ait ayarların XML formatında saklandığı dosya adıdır?
A) app.config
B) app.xml
C) web.config
D) web.xml
2. Aşağıdaki DLINQ sorgularından hangisi yanlış yazılmıştır?
A)

```
from c in db.Customers
where c.City == "London"
select new MyType(c.ContactName, c.Phone) into x
orderby x.Name
select x;
```


B)

```
from c in db.Customers
where c.City == "London"
select { c.ContactName, c.Phone };
```


C)

```
from context in db.Customers, o in db.Orders
where c.CustomerId == o.CustomerID && c.City == "London"
select o;
```


D)

```
db.Customers.First(c => c.CustomerID == id);
```
3. Aşağıdakilerden hangisi dönüştürme işlemi metot arayüzlerini yapısında tutar?
A) IValueConverter
B) INotifyPropertyChanged
C) IComparer
D) IDataReader
4. Aşağıdaki metotlardan hangisi varlık sınıflarını yineler?
A) SubmitChanges()
B) LoadContext()
C) SaveChanges()
D) Refresh()
5. Aşağıdakilerden hangisi varlık anlamındadır?
A) Attribute
B) Interface
C) Entity
D) DataContext

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki modüle geçmek için öğretmeninize başvurunuz.

CEVAP ANAHTARLARI

ÖĞRENME FAALİYETİ-1'İN CEVAP ANAHTARI

1	D
2	A
3	C
4	B
5	C

ÖĞRENME FAALİYETİ-2'NİN CEVAP ANAHTARI

1	C
2	C
3	A
4	D
5	B
6	B

MODÜL DEĞERLENDİRMENİN CEVAP ANAHTARI

1	A
2	B
3	A
4	D
5	C

KAYNAKÇA

- Michaelis, Marc, **Essential C# 4.0**, Addison-Wesley /US, 2010.
- Sharp, John, **Microsoft Visual C# 2008**, Microsoft Press / US, 2008.
- Fabio Claudio Ferracchiati, **LINQ for Visual C# 2008**, Apress / US, 2008.