ADO.NET ve DATASET

Öğr. Gör. Emine TUNÇEL
Kırklareli Üniversitesi Pınarhisar Meslek Yüksekokulu

DATASET Kavramı

- Uygulamanızda, veriler için, bir veritabanı modeli oluşturmanızı sağlayan class gruplarıdır.
- Veritabanı kavramını hatırlayacak olursak, bir veritabanında tablolar, kayıtlar(sıra, rows), sütunlar, alanlar, tablolar arası ilişkiler, özel anahtarlar vs. vardı.
- İşte Dataset bunların hepsini uygulamanızda veritabanından bağımsız olarak oluşturmanızı sağlar.
- Bir veritabanı ile bağlantıya geçerek, buradaki bütün verileri uygulamanıza DATASET sayesinde aktardıktan sonra artık, veritabanı ile ilginiz kalmaz.
- Veriler sunucunun hafızasında saklanır. Hafızadaki bu verileri dilediğiniz gibi kullanabilirsiniz. Görüntüleyebilir, değiştirebilir, silebilirsiniz vs.
- Eğer isterseniz veritabanındaki verileri uygulamanızdaki son durumdaki veriler ile güncelleyebilirsiniz.

DATASET Kavramı

- Dataset'e ait bütün classlar .NET Frameworkünde System.Data namaspace i altında toplanmıştır. Sayfalarınızda Dataset ile ilgili işlemler yapmak istiyorsanız öncelikle bu Namespace i import etmelisiniz.
- **<%@ Import Namespace="System.Data" %>**
- Dataset iki yöntemle oluşturulup kullanılabilir:
- Birinci yöntemde Tablolar, Row'lar ve veritabanı ile ilgili diğer kavramlar en baştan progmatik olarak tasarlanarak boş bir dataset oluşturulur
- İkinci yöntemde ise, bir veritabanı ile bağlantı kurularak buradaki veri olduğu gibi Dataset e aktarılır

Programatik Olarak, En Baştan Bir Dataset Oluşturmak

- Örneğimizde hiçbir veritabanına bağlanmadan bir dataset tasarlayacak ve bu Dataset de bir veri tablosu oluşturacağız.

```
using System.Data;

public partial class DatasetOlusturma : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        DataSet ds = new DataSet("Site");
        DataTable tablom = new DataTable("Musteriler");
        tablom.Columns.Add("Ad",System.Type.GetType("System.String"));
        tablom.Columns.Add("Soyad", System.Type.GetType("System.String"));
        tablom.Columns.Add("Meslek", System.Type.GetType("System.String"))
        tablom.Columns.Add("Yas", System.Type.GetType("System.Int32"));
        ds.Tables.Add(tablom);
```

Programatik Olarak, En Baştan Bir Dataset Oluşturmak

- Öncelikle System.Data namespace'ini uygulamamıza dahil ediyoruz.
- `DataSet ds = new DataSet("Site");` ile Dataset nesnesini oluşturup bunu bir değişkene (örneğimizde bu değişken ds) atıyoruz. ds değişkeni artık bir Dataset'i temsil etmektedir. Bu değişkenen parantez içinde bir isim veriyoruz. Örneğimizde bu isim Site
- Dataset oluşturulduktan sonra, bu Dataset de kullanılacak tablo ya da tablolar oluşturulmalı.
- `DataTable tablom = new DataTable("Musteriler");` bildirimi ile tablomuzu oluşturuyoruz. Yine parantez içinde tablonun adını Musteriler olarak belirledik
- Ardından Musteriler tablosunun Sütun isimlerini ve tiplerini belirtiyoruz.
- Ve oluşturduğumuz tablomuzu `ds.Tables.Add(tablom);` bildirimi ile datasetimize ekliyoruz

Verileri Girmek

- Dataset ve bu Dataset de bir tablo oluşturulduktan sonra artık, bu tablomuza verileri girebiliriz.
- Bunun için öncelikle tablomuzda gireceğimiz veriler için bir satır oluşturmalıyız.
- Sütunlarımız mevcut ama henüz bir satır oluşturmadık.
- Aşağıdaki kodları sayfamıza ekliyoruz:

```
DataRow satir = tablom.NewRow();//Yeni bir satır oluşturduk
satir["Ad"] = "Emine";// Verilerimizi girdik
satir["Soyad"] = "TUNÇEL";
satir["Meslek"] = "Öğretim ELemanı";
satir["Yas"] = "28";
tablom.Rows.Add(satir);//oluşturduğumuz satır değişkenini tablomuza ekledik
```

Verileri Girmek

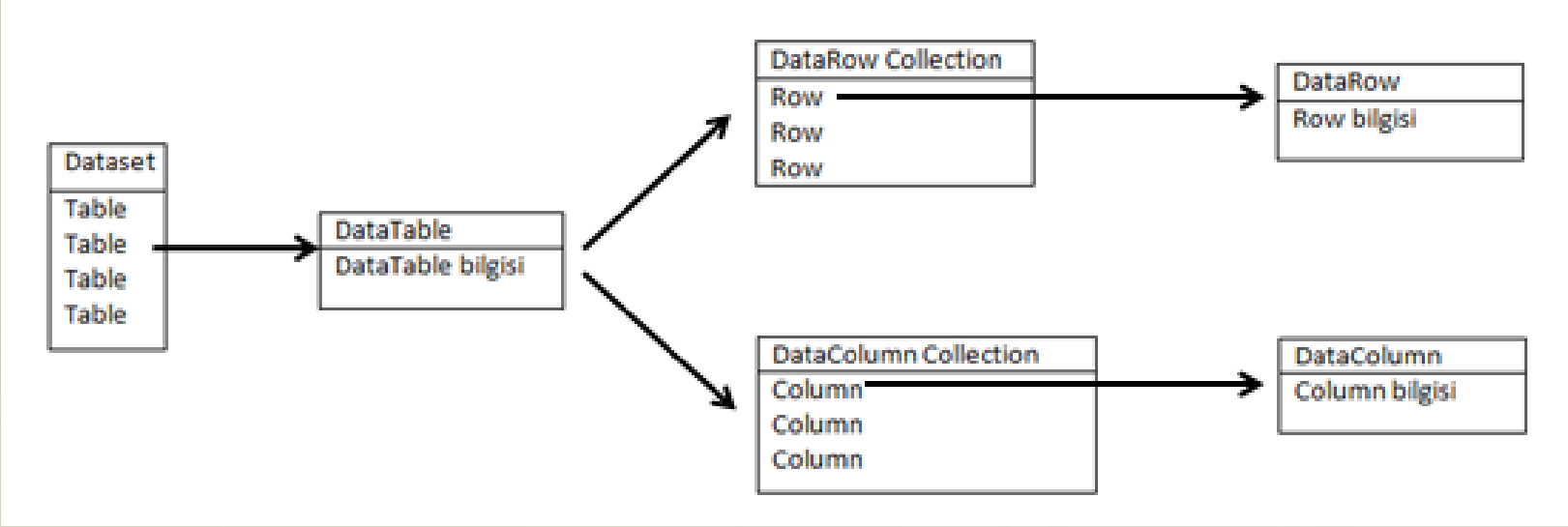
- Tablodaki sütunlar sıfırdan başlayarak numaralandırılır. Buna göre aşağıda verilen iki tanımlama da kullanılabilir

```
DataRow satir = tablom.NewRow();
satir["Ad"] = "Emine";
satir["Soyad"] = "TUNÇEL";
satir["Meslek"] = "Öğretim ELemanı";
satir["Yas"] = "28";
tablom.Rows.Add(satir);
```

```
DataRow satir1 = tablom.NewRow();
satir1[0] = "Ayşe";
satir1[1] = "Demir";
satir1[2] = "Doktor";
satir1[3] = "32";
tablom.Rows.Add(satir1);
```

Verilerin Elde Edilmesi

- Veriler ile işlem yapabilmek için, içerisinde bir çok kavramın tanımlandığı Dataset modeli ve hiyerarşisi iyice anlaşılmalıdır.
- Dateset içinde tablolar, sütunlar ve sıralar hiyerarşik bir yapıda tutulur.



Verilerin Elde Edilmesi

- Örneğin herhangi bir satırın herhangi bir sütunundaki veriyi elde etmek için aşağıdaki gibi bir notasyon yazılır :
- ds.Tables["tablo_ismi"].Rows[Satır_İndeksi][Sutun_Indeksi]
- Örneğimize uyarlarsak;
- ds.Tables["Musteriler"].Rows[1][0]
- Veya
- ds.Tables["Musteriler"].Rows[1]["Yas"]
- Şeklinde bir ifade "ds" datasetimizdeki tablolardan, Musteriler isminde olanının sıraları içinden (rows) 1.sırayı ve bu sıradaki "Yas" sütununu veya "0" nolu (bu "ad" sütununun indeksidir) sütunu getirir.
- Böylece ulaşmak istediğimiz veriyi noktasal olarak tanımladık
- Noktasal notasyonu bir değişkene atamak da mümkündür
- Örneğin;
- DataRow satir2 = ds.Tables["Musteriler"].Rows[1];
- Şeklinde bir ifade müşteriler tablosundaki 2.satırı (sıra numaralarının 0'dan başladığını unutmayın) bir değişkene atar. Bu değişken tipini atanacak değerin yapısına göre tanımlamalıyız. Atayacağımız değer bir sıra olduğuna göre satır2 değişken tipini DataRow olarak tanımlamalıyız

Verilerin Elde Edilmesi

- Artık satir2 değişkenimiz bir sırayı temsil etmektedir
- Bu sıradan herhangi bir sütundaki veriyi elde etmek için sütun ismi veya indeksini belirtmeliyiz.
- satir2["Yas"]
- Veya
- Satir2[3]
- Noktasal notasyonları değişkenlere atayarak işlem yapmak kolaylık sağlar. Ancak değişken tip tanımlamaları çok dikkatli yapılmalıdır
- Bazı örnekler:
- DataRowCollection siralar;
- siralar = ds.Tables["Musteriler"].Rows;
- Response.Write(siralar[1]["Meslek"]);
- Müşteriler tablosundaki bütün sıralar, DataRowCollection yani sıra koleksiyonu tipinde bir değişkene atanıyor. Artık tablomuzdaki sıraları bu değişken temsil ediyor
- Bu sıradan veri elde etmek için satır ve sütun indekslerini girdik

Verilerin Elde Edilmesi

- Başka bir örnek;
- DataTable tablo_degiskenim;
- tablo_degiskenim=ds.Tables["Musteriler"];
- Response.Write(tablo_degiskenim.Rows[0][2]);
- Bu sefer tablomuzu, DataTable tipinde bir değişkene atıyoruz
- Bu değişkenimizden her hangi bir veriyi elde etmek için, hiyerarşiye uyup, noktasal notasyonda önce Rows yani satırları belirtmeliyiz. Daha sonra bu satırlardan hangisini ve hangi sütundaki veriyi talep ettiğimizi yazmalıyız

AutoIncrement

- Kelime anlamı otomatik artış olan AutoIncrement veritabanlarında, otomatik numaralandırma yapmak için kullanılır.
- Bunun için oluşturacağımız sütunlara AutoIncrement özelliği atamalıyız.
- Böylece bu sütun, her yeni kayıt için farklı bir numara atayacaktır.
- Böyle bir indeksleme her kayıtın veya sıranın yegane (eşsiz) olmasını sağlayacaktır
- Genellikle özel anahtarlar bu sütuna atanır, çünkü bu sütundaki alanların tekrarlanma olasılığı yoktur.

AutoIncrement

- AutoIncrement özelliğini uygulamamızda sütunları oluştururken atayabiliriz.

```
tablom.Columns.Add("Musteri_Id", System.Type.GetType("System.Int32"));
tablom.Columns["Musteri_Id"].AutoIncrement = true;
```

- Müşteriler tablomuza integer verileri kabul eden Musteri_Id sütununu ekledik ve otomatik numaralandırma özelliğini aktif hale getirdik.
- AutoIncrement'in başlangıç sayısı ve artış adımları tarafımızdan belirlenebilir.
- **AutoIncrementSeed** ile numaralandırmanın hangi değerden başlayacağı,
- **AutoIncrementStep** ile de her adımda ne kadar artacağı belirlenir

```
tablom.Columns["Musteri_Id"].AutoIncrementSeed = 10;
tablom.Columns["Musteri_Id"].AutoIncrementStep = 5;
```

RelationShips (İlişkilendirmeler)

- Temel olarak ilişkilendirme, iki tablo arasında, tabloların birer sütunu ile bağlantı oluşturmaktır.
- İki tablonun, bağlantıyı oluşturacak sütunlarının benzer verileri taşıması gerekir.
- En çok kullanılan ilişkilendirme biçimi Master-Detay ilişkilendirmesidir.
- Master tablo farklı kayıtları tutar, detay tablosu ise master tablodaki her bir kayıt için diğer alanlardan oluşan detay kayıtlarını barındırır. Bu iki tablo birbiriyle ortak bir sütun ile bağlantılandırılır.

RelationShips (İlişkilendirmeler)

- Aşağıdaki gibi iki tablo düşünelim:

Musteriler: Query(emine\sqlexpress.Site)

Musteri_İd	Ad	Soyad	Meslek	Yas
1	Ayşegül	Demirkol	Mühendis	30
2	Denizhan	Akkaya	Muhasebeci	26
3	Gönül	Güdü	Rehber	34
4	Hakan	alirıza	Mühendis	34
5	Levent	Ayaz	Aşçı	28
6	Serdar	Benli	Pazarlamacı	32

AlısVeris: Query(emine\sqlexpress.Site)

Musteri_Numa...	Urun	Adet	Fiyat
1	Kitap	2	250
1	CD	4	30
2	Poster	5	150
1	Kaset	1	20

- Alışveriş tablosu Müsteriler tablosunu detaylandırıyor.

RelationShips (İlişkilendirmeler)

- DataSet de tablolar arasındaki ilişkilendirmeyi **DataRelation** class'ı yapar.
- Bir DataRelation nesnesinin iki ana üyesi vardır. Bunlar **ParantColumn** ve **ChildColumn**'dur.
- Örneğin Müşteriler ve AlışVeriş olmak üzere iki tablomuz olduğunu varsayalım. Bu iki tablo arasında ortak olarak kullanılan alan Müşteri_ID alanı olmalıdır.
- Buna göre Müşteriler tablosundaki Musteri_ID sutunu ParentColumn, AlışVeriş tablosundaki Müsteri_ID sutunu ise ChildColoumn olacaktır

```
DataColumn parent = ds.Tables["Musteriler"].Columns["Musteri_ID"];
DataColumn child = ds.Tables["AlisVeris"].Columns["Musteri_ID"];

//İlişkiyi oluşturalım
DataRelation dr = new DataRelation("iliski_ismi", parent, child);
//Dataset nesnesini bu ilişkiden haberdar etmek için aşağıdaki gibi eklenmelidir
ds.Relations.Add(dr);
```

RelationShips (İlişkilendirmeler)

```
DataColumn parent = ds.Tables["Musteriler"].Columns["Musteri_ID"];
DataColumn child = ds.Tables["AlisVeris"].Columns["Musteri_ID"];

//İlişkiyi oluşturalım
DataRelation dr = new DataRelation("iliski_ismi", parent, child);
//Dataset nesnesini bu ilişkiden haberdar etmek için aşağıdaki gibi eklenmelidir
ds.Relations.Add(dr);
```

- Yukarıda tanımlanan ilişkinin anlamı şudur :
- Her bir müşterinin çok sayıda siparişi bulunmaktadır.
- Bu ilişkiden faydalanarak her bir Müşteri kaydına ait Child kayıtları elde etmek mümkündür.
- Bu kayıtları elde etmek için DataRow sınıfının **GetChildRows()** metodu kullanılmaktadır.
- Aynı şekilde AlışVeriş tablosundaki her siparişin hangi müşteriye ait olduğunu temsil eden Parent Row kaydını yine DataRow sınıfının **GetParentRow()** isimli metodu ile elde edebiliriz.

Primary Key (Özel Anahtar)

- Bu tanım, kayıtların tek ve eşsiz olmasını sağlar.
- Bir veya daha fazla alan için tanımlanabilir.
- Özel anahtar atanan alan ya da alanlara girilen veriler, başka kayıtlarda tekrarlanamaz.
- Örneğin ***AD*** alanına atayacağımız özel anahtardan sonra aynı ada sahip iki kayıt bulunamaz
- Ya da ***AD*** ve ***SOYAD*** alanlarına birlikte atayacağımız özel anahtardan sonra aynı ada ve soyada sahip iki kayıt bulunamaz. Ama, kayıtlarda tek başına Ad ve Soyad aynı olabilir.

Primary Key (Özel Anahtar)

- Özel anahtar bir tablo için ***PrimaryKey*** ifadesi ile oluşturulur
- ds.Tables[0].PrimaryKey = anahtar;
- Şeklinde bir ifade "*PrimaryKey*" i belirler.
- Burada "anahtar" ifadesi bir değişken olup, özel anahtar atayacağımız tablonun sütunlarını barındırır. Birden fazla sütuna özel anahtar atamak mümkün olduğuna göre "PrimaryKey " bir diziyi temsil eder.
- Dolayısıyla " anahtar " değişkeninin tanımı da bir dizi şeklinde olmalıdır. Şimdi " Anahtar " değişkenini tanımlayalım:

```
        DataColumn [] anahtar = {ds.Tables[0].Columns[1],
ds.Tables[0].Columns[2] };
        ds.Tables[0].PrimaryKey = anahtar;
```

Veritabanından Dataset'e Veri Yükleme

- Bir veritabanındaki tabloyu tüm verisi ile DataSet'e aktarabilirsiniz.
- Veri bir kere DataSet'e aktarıldıktan sonra, artık veritabanı ile alakanız kalmaz.
- Veriler, DataSet ile progmatik olarak yönetilebilir.
- Uygulamanız sonunda, eğer son durumu veritabanına geri yüklemek isterseniz, tekrar veritabanı ile bağlantıya geçip güncelleme yapabilirsiniz.
- Veritabanı ile uygulama arasındaki bu ilişkiyi **DataAdapter** düzenler.

DATAADAPTER SINIFI

- Verinin getirilmesi ve iletilmesi gibi görevleri otomatik olarak gerçekleştirir.
- Bu amaçla SELECT, INSERT, DELETE ve UPDATE komutlarını kullanır.

```
SqlConnection baglanti = new SqlConnection(@"Data Source=Emine\SQLEXPRESS;Initial Catalog=Site;Integrated Security=True;Pooling=False");
SqlDataAdapter da = new SqlDataAdapter("Select * from Musteriler", baglanti);
```

- Yukarıda ki bildirim ile Musteriler tablosunun tüm kayıtlarını elde etmek için gerekli adaptör tanımlandı
- Ancak henüz veriler fiziksel olarak elde edilmedi.
- Şimdi yapmamız gereken elde etme metodunu tanımlamak
- Bu metot uygulamamız için Dataset

DATASET

- Kodlarımızda

```
DataSet ds = new DataSet();
```

ifadesi ile bir dataset oluşturuyoruz

- Ancak bu Dataset in içi henüz boş
- Şimdi, da olarak isimlendirdiğimiz Data Adapter nesnesinde tanımlanmış veri yapısını Fill() metodunu kullanarak Dataset'e aktarıyoruz

```
da.Fill(ds, "Ds_Musteriler");
```

- Bu bildirim ile veritabanındaki Müşteriler tablosundan DataAdapter sayesinde elde edilen tüm kayıtlar ds şeklinde tanımlanmış Dataset'e, aktarılıyor.
- Bu bildirimde ki Ds_Musteriler tablonun Datasetteki ismidir.

DATASET

- Uygulamamızda bir Gridview kontrolü kullanarak Dataset deki verileri görüntüleyebiliriz.

```
    protected void Page_Load(object sender, EventArgs e)
    {
        SqlConnection baglanti = new SqlConnection(@"Data Source=Emine\SQLEXPRESS;Initial
Catalog=Site;Integrated Security=True;Pooling=False");
        SqlDataAdapter da = new SqlDataAdapter("Select * from Musteriler", baglanti);
        DataSet ds = new DataSet();
        da.Fill(ds, "Ds_Musteriler");
        GridView1.DataSource = ds;
        GridView1.DataBind();
    }
```

DataSet Güncellemeleri

- Dataset nesnesine, yeni bir kayıt eklenmesi, mevcut kaydın silinmesi ya da güncellenmesi durumunda yapılan bu değişikliklerin veritabanına da yansıtılması gerekir.
- Dataset de yapılan değişiklikler veritabanına **DataAdapter** nesnesinin **Update** metodu kullanılır.
- Dataset nesnesine aşağıdaki verilerin aktarıldığını varsayalım:

(Row State)	Kişi_ID	Ad	Soyad	Meslek	Yas
Unchanged	1	Emine	TUNÇEL	Öğretim Elm	29
Unchanged	2	Aylin	CAN	Muhasebeci	32

- Uygulamamızda id'si 2 olan kişinin Yaş bilgisini 35 olarak değiştirdik diyelim.
- Bu değişiklik sonucunda DataRow.RowState özelliği Unchanged 'den Modied' a döner. Yani Dataset içeriği şu şekilde olur.

(Row State)	Kişi_ID	Ad	Soyad	Meslek	Yas
Unchanged	1	Emine	TUNÇEL	Öğretim Elm	29
Modified	2	Aylin	CAN	Muhasebeci	32

DataSet Güncellemeleri

- DataAdapter nesnesinin Update metodu çalıştırdığında, Update metodu her satırı denetler.
- İlk satır için veritabanına hiçbir Sql komutu gönderilmez, çünkü RowState özelliğinin değeri “Unchanged” olarak kalmıştır.
- Buradaki RowState özelliği DataSet’ i oluşturan DataTable nesnesinin her DataRow nesnesinde bulunan bir özelliktir.
- Ancak ikinci satır için, Update metodu otomatik olarak DataAdapter nesnemizde tanımladığımız komutu arar ve onu çağırır.
- Bu komutu veritabanı gönderip orada işlettirir.
- Update metodu RowState özelliğindeki duruma göre : RowState özelliği değişen satırlar için Modified”, dataset’ e yeni eklenen satırlar için “Added” ve silinen satırlar için “Deleted” değerlerini alır
- DataAdapter nesnesinden Modified satırlar için UpdateCommand, Added satırlar için InsertCommand ve Deleted satırlar içinse DeleteCommand’ da tanımlanmış Sql komutlarını veritabanına gönderir ve değişikliklerin orada da olmasını sağlar. İşte bütün bu özelliklerin çalışmasını sağlayan Update metodudur.

DataSet Güncellemeleri

- DataAdapter nesnesinin Update metodu çalıştırıldığında, Update metodu her satırı denetler.
- İlk satır için veritabanına hiçbir Sql komutu gönderilmez, çünkü RowState özelliğinin değeri “Unchanged” olarak kalmıştır.
- Buradaki RowState özelliği DataSet’ i oluşturan DataTable nesnesinin her DataRow nesnesinde bulunan bir özelliktir.
- Ancak ikinci satır için, Update metodu otomatik olarak DataAdapter nesnemizde tanımladığımız komutu arar ve onu çağırır. Bu komutu veritabanı gönderip orada işlettirir.
- RowState özelliği değişen satırlar için Modified”, dataset’ e yeni eklenen satırlar için “Added” ve silinen satırlar için “Deleted” değerlerini alır
- DataAdapter nesnesinden Modified satırlar için UpdateCommand, Added satırlar için InsertCommand ve Deleted satırlar içinse DeleteCommand’ da tanımlanmış Sql komutlarını veritabanına gönderir ve değişikliklerin orada da olmasını sağlar. İşte bütün bu özelliklerin çalışmasını sağlayan Update metodudur.

DataSet Güncellemeleri

```
    protected void Page_Load(object sender, EventArgs e)
    {
        SqlConnection baglanti = new SqlConnection
(WebConfigurationManager.ConnectionStrings["baglantim"].ConnectionString);
        DataSet ds = new DataSet();
        SqlDataAdapter da = new SqlDataAdapter("select * from Musteriler",
baglanti);
        da.Fill(ds, "Musteriler");
        da.UpdateCommand=new SqlCommand("update Musteriler set
Ad=@ad,Soyad=@soyad,Meslek=@meslek,yas=@yas where Musteri_Id=@id", baglanti);
        da.UpdateCommand.Parameters.Add("@ad",SqlDbType.VarChar,20,"Ad");
        da.UpdateCommand.Parameters.Add("@soyad", SqlDbType.VarChar, 20, "Soyad");
        da.UpdateCommand.Parameters.Add("@meslek", SqlDbType.VarChar, 30,
"Meslek");
        da.UpdateCommand.Parameters.Add("@yas", SqlDbType.Int,0, "Yas");
        da.UpdateCommand.Parameters.Add("@id", SqlDbType.Int, 0, "Musteri_Id");
        ds.Tables[0].Rows[0][1] = "Menekşe";
        da.Update(ds, "Musteriler");
    }
```

DataSet Güncellemeleri

- Öncelikle veritabanı bağlantısını oluşturuyoruz
- SqDataAdapter nesnesini tanımlıyoruz ve bu nesne aracılığılya elde ettiğimiz verileri Dataset nesnesine aktarıyoruz.
- Dataset nesnesinde tutulan veriler üzerinde güncelleme işlemi yapacağımız için DataAdapter nesnesinin UpdateCommand özelliğine Update SQL komutunu aktarıyoruz.
- **Dataset nesnesinde ne tür değişiklik yapılacaksa, uygun özellikler kullanılmalı. Veri güncelleme için UpdateCommand özelliği, Veri ekleme için InsertCommand özelliği, Veri silme için ise DeleteCommand özelliği**
- Update sql komutunda kullandığımız parametre tanımlamalarını yapıyoruz.
- Dataset nesnesinde gerekli değiişikliği yapıp, Update metodunu çalıştırıyoruz.
- Update komutu tablodaki tüm satırları kontrol ediyor ve RowState özelliği Modified olan satır için DataAdapter da tanımlanmış UPDATE sorgusunu çalıştırıyor
- Dataset nesnesinde yapılan değişikliklerin veritabanına da yansıtılması sağlanıyor

DataSet Güncellemeleri

- Veri Ekleme-InsertCommand Özelliği

```
        SqlConnection baglanti = new SqlConnection(WebConfigurationManager.ConnectionStrings
["baglantim"].ConnectionString);
        DataSet ds = new DataSet();
        SqlDataAdapter da = new SqlDataAdapter("select * from Musteriler", baglanti);
        da.Fill(ds, "Musteriler");
        da.InsertCommand = new SqlCommand("insert into Musteriler values
(@ad,@soyad,@meslek,@yas) ", baglanti);
        da.InsertCommand.Parameters.Add("@ad", SqlDbType.VarChar, 20, "Ad");
        da.InsertCommand.Parameters.Add("@soyad", SqlDbType.VarChar, 20, "Soyad");
        da.InsertCommand.Parameters.Add("@meslek", SqlDbType.VarChar, 30, "Meslek");
        da.InsertCommand.Parameters.Add("@yas", SqlDbType.Int,0, "Yas");
        //Dataset nesnesine yeni kayıt ekleyelim
        DataRow yeni_Satir = ds.Tables[0].NewRow();
        yeni_Satir[1]="Aylin";
        yeni_Satir[2]="Aylin";
        yeni_Satir[3]="Aylin";
        yeni_Satir[4]=22;
        ds.Tables[0].Rows.Add(yeni_Satir);
        //Güncelleme komutunu verelim
        da.Update(ds,"Musteriler");
    }
```

DataSet Güncellemeleri

- Veri Ekleme-DeleteCommand Özelliği

```
        SqlConnection baglanti = new SqlConnection
(WebConfigurationManager.ConnectionStrings["baglantim"].ConnectionString);
        DataSet ds = new DataSet();
        SqlDataAdapter da = new SqlDataAdapter("select * from Musteriler",
baglanti);
        da.Fill(ds, "Musteriler");
        da.DeleteCommand = new SqlCommand("delete  from Musteriler where Ad=@ad and
 Soyad=@soyad", baglanti);
        da.DeleteCommand.Parameters.Add("@ad", SqlDbType.VarChar, 20, "Ad");
        da.DeleteCommand.Parameters.Add("@soyad", SqlDbType.VarChar, 20, "Soyad");

        //Dataset nesnesinden biy kaydı silelim
        ds.Tables[0].Rows[7].Delete();

        //Güncelleme komutunu verelim
        da.Update(ds,"Musteriler");
    }
```

- KAYNAK
- C# ile ASP.NET
- Zafer Demirkol