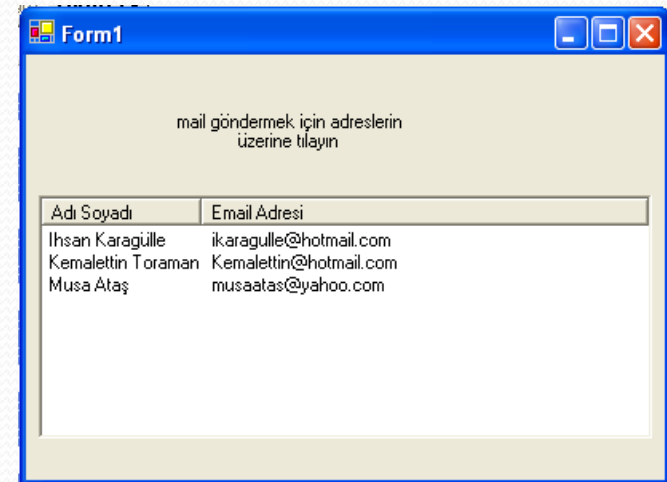
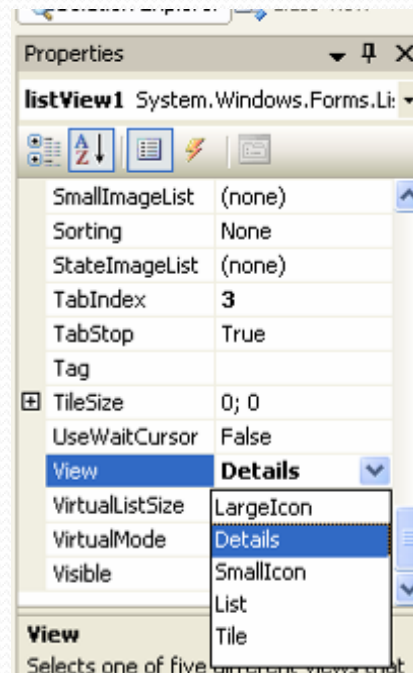
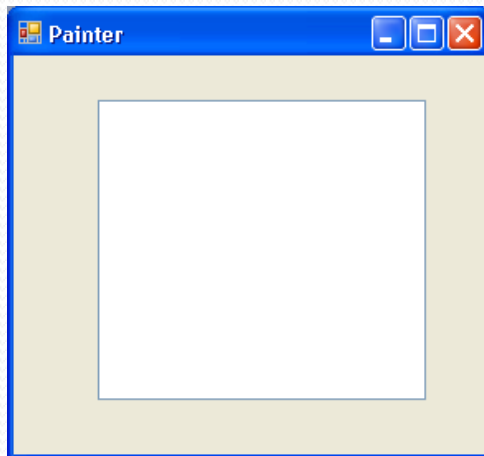


VISUAL STUDIO.NET ve FORM UYGULAMALARI

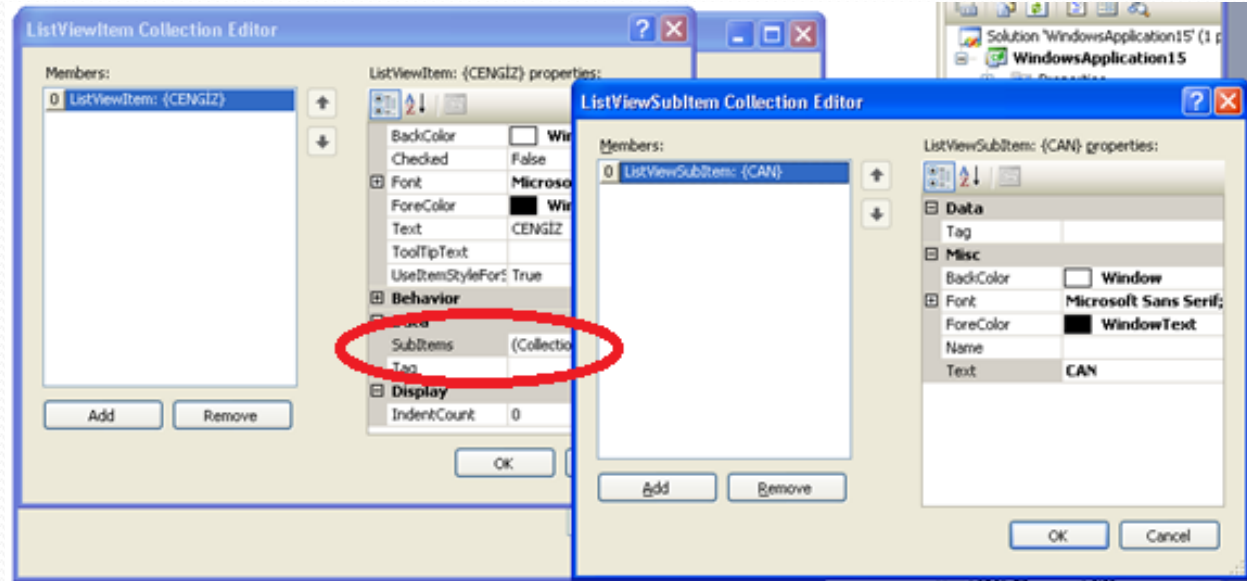
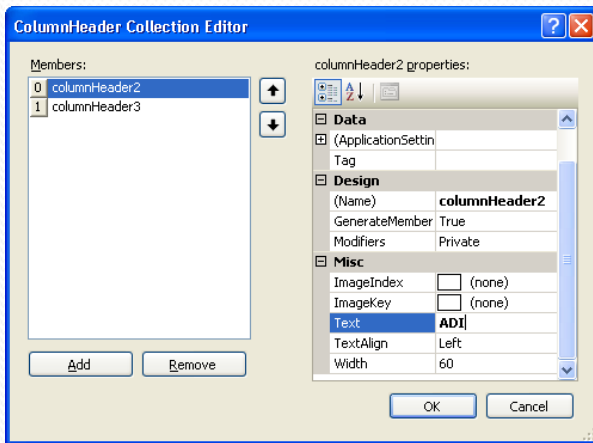
• Listview Kontrolleri

- **Listview** gelişmiş bir listeleme kontrolüdür. Listbox'ta olduğu gibi içine elemanlar eklenebilir, her elemana bir resim verilebilir ve listedeki elemanlar farklı biçimlerde listelenebilir. Form uygulamasına Listview özelliği eklendikten sonra **Properties-View** kısmından Details seçeneğini seçerek oluşturacak sütunlar görüntülenir.



VISUAL STUDIO.NET ve FORM UYGULAMALARI

- Listview içerisindeki verileri sütunlar halinde gruplamak için **Properties/Columns (Collection)** seçeneği seçilir ve sütun ifadeleri girilir.
- Sütunlar içerisinde görülecek ifadeler içinde **Items (Collection)** seçeneğini seçilip ifadeler girilir. Burada dikkat edilmesi gereken birinci sütuna gelecek ifadeyi ekledikten sonra bu pencerede iken diğer sütunlara gelecek ifadeler için **SubItems (Collection)** seçeneğini kullanmaktır.



VISUAL STUDIO.NET ve FORM UYGULAMALARI

- Listview içerisine eklemeler kod ortamında da gerçekleştirilebilir.




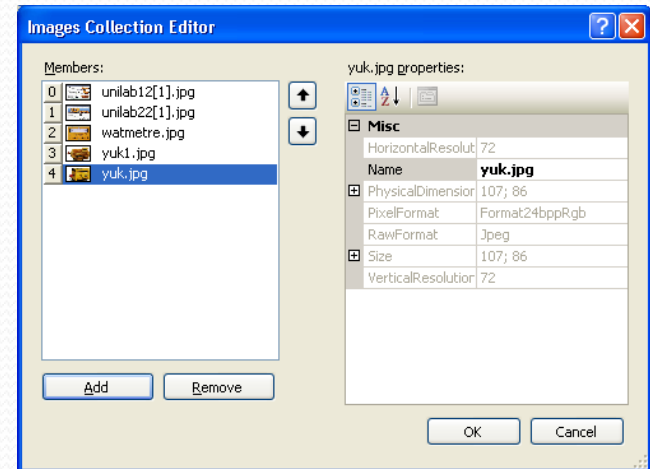
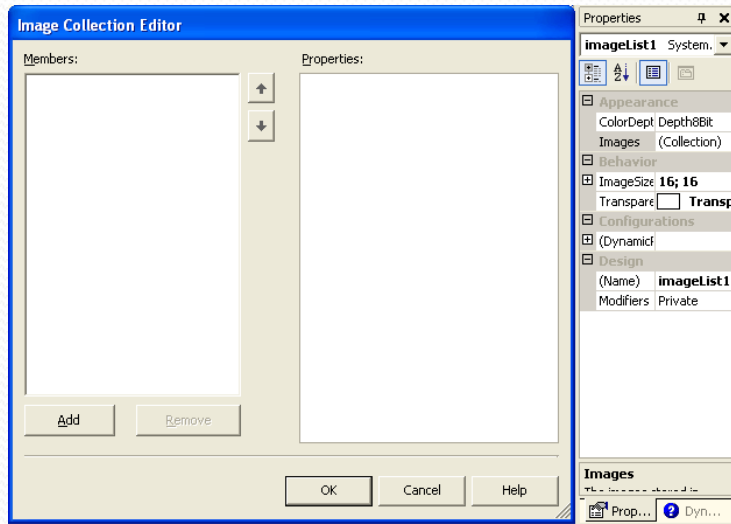
ADI	SOYADI
CENGİZ	CAN

- `private void Form1_Load(object sender, System.EventArgs e)`
- `{`
- `listView1.Activation = ItemActivation.OneClick;`
- `// tek tıklamada aktif hale getir.`
- `listView1.Columns.Add("ADI", -2, HorizontalAlignment.Left);`
- `// Adı sütun başlığı, -2, Girilen ifadenin tam olarak sığması için gereken ifade, sola hizalama`
- `listView1.Columns.Add("SOYADI",-2,HorizontalAlignment.Left);`
- `listView1.Items.Add("CENGİZ");`
- `listView1.Items[0].SubItems.Add("CAN");`
- `listView1.View = View.Details;`
- `listView1.Sorting = SortOrder.Ascending;// Sıralama }`

VISUAL STUDIO.NET ve FORM UYGULAMALARI

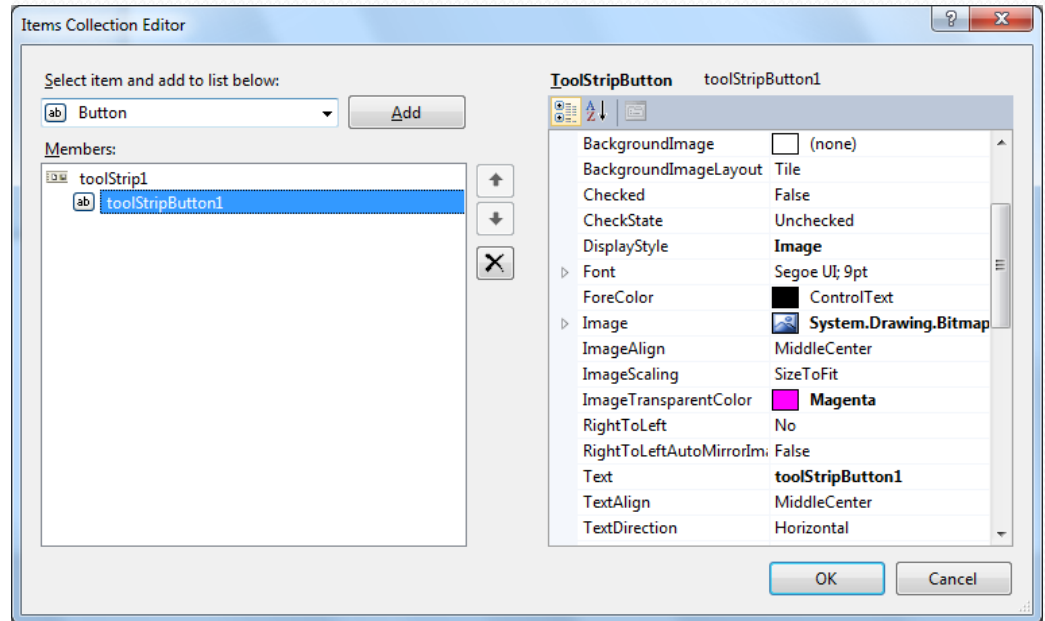
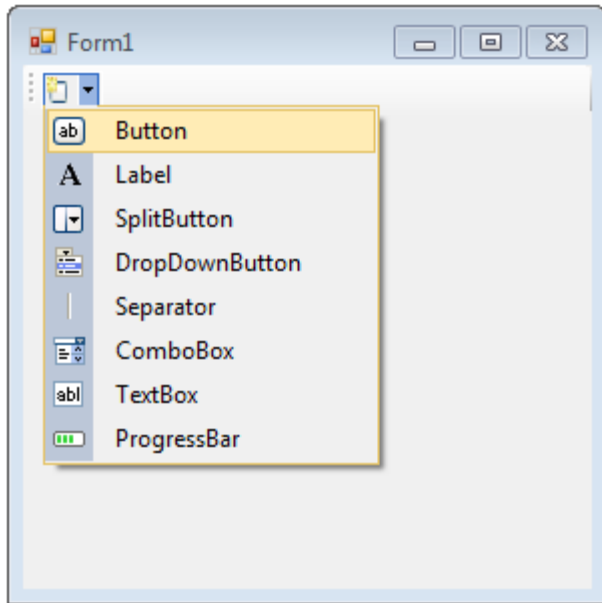
• Imagelist Kontrolleri

- Birden fazla resmi bir arada tutmak için kullanılan bir kontroldür. Daha çok ListView, ToolBar, TreeView gibi birden fazla resmin bulunabileceği kontrollerdeki resimleri bir arada tutmak için kullanılır.
- Bu kontrolü form üzerine yerleştirdikten sonra Properties penceresindeki Images özelliğinin yanında bulunan  düğmesine tıklayarak resimleri belirleyebilirsiniz.



VISUAL STUDIO.NET ve FORM UYGULAMALARI

- **ToolStrip Kontrolleri**
- Araç çubukları için hazır düğmeler sunan bir kontroldür. ToolStrip kontrolü içine düğmeler ekleyip üzerlerine resim yerleştirilebilir veya yazı yazılabilir.

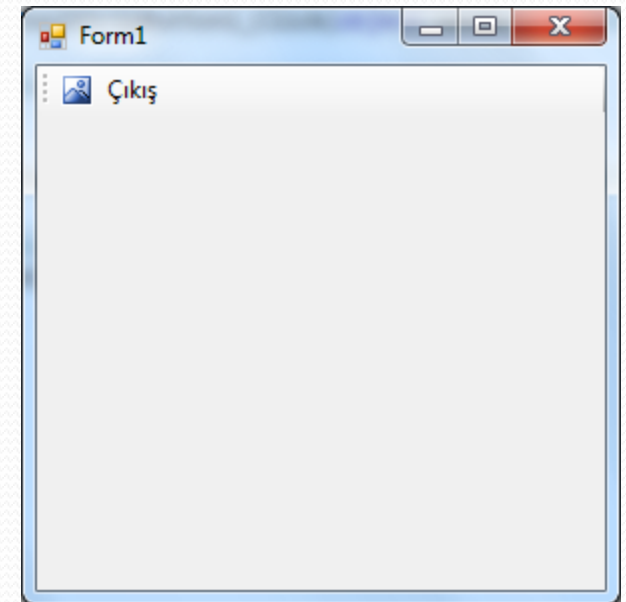


VISUAL STUDIO.NET ve FORM UYGULAMALARI

- **ToolStrip Kontrolleri**

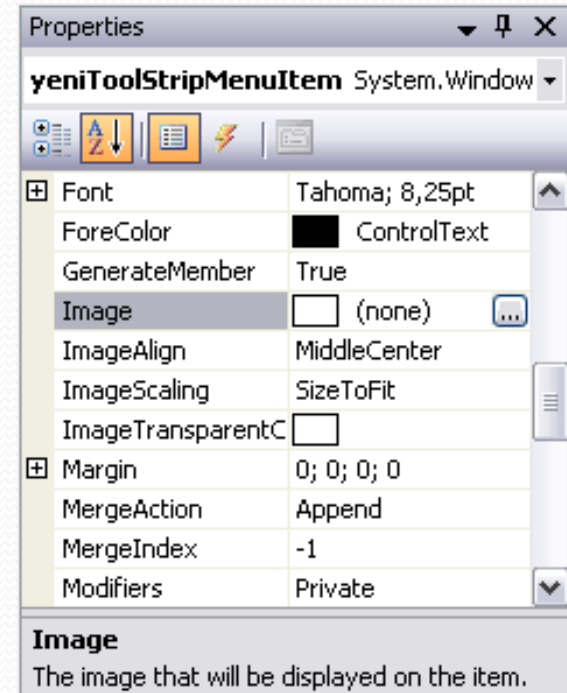
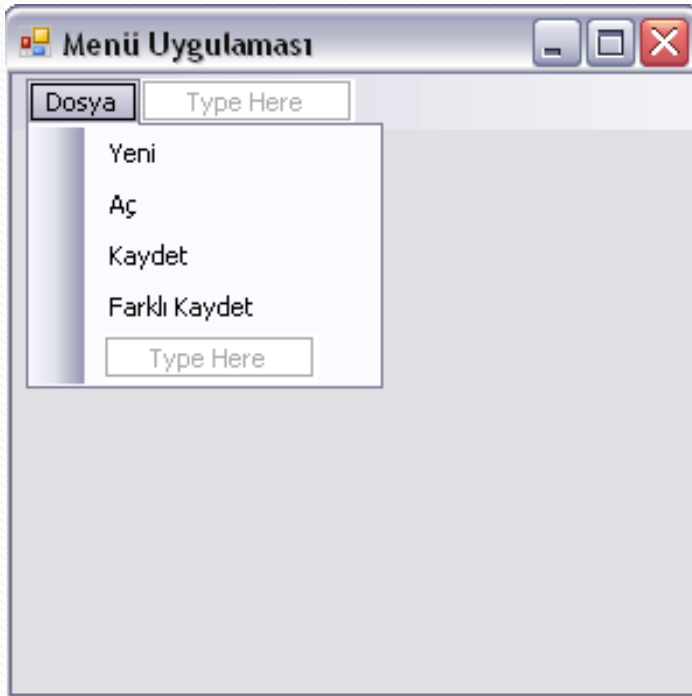
- `private void toolStripButton1_Click(object sender, EventArgs e)`
- `{ MessageBox.Show("Toolsript Buton 1 e tıklandı"); }`

- `private void toolStripLabel1_Click(object sender, EventArgs e)`
- `{ MessageBox.Show("Toolsript Label 1 e tıklandı");`
- `this.Close(); }`



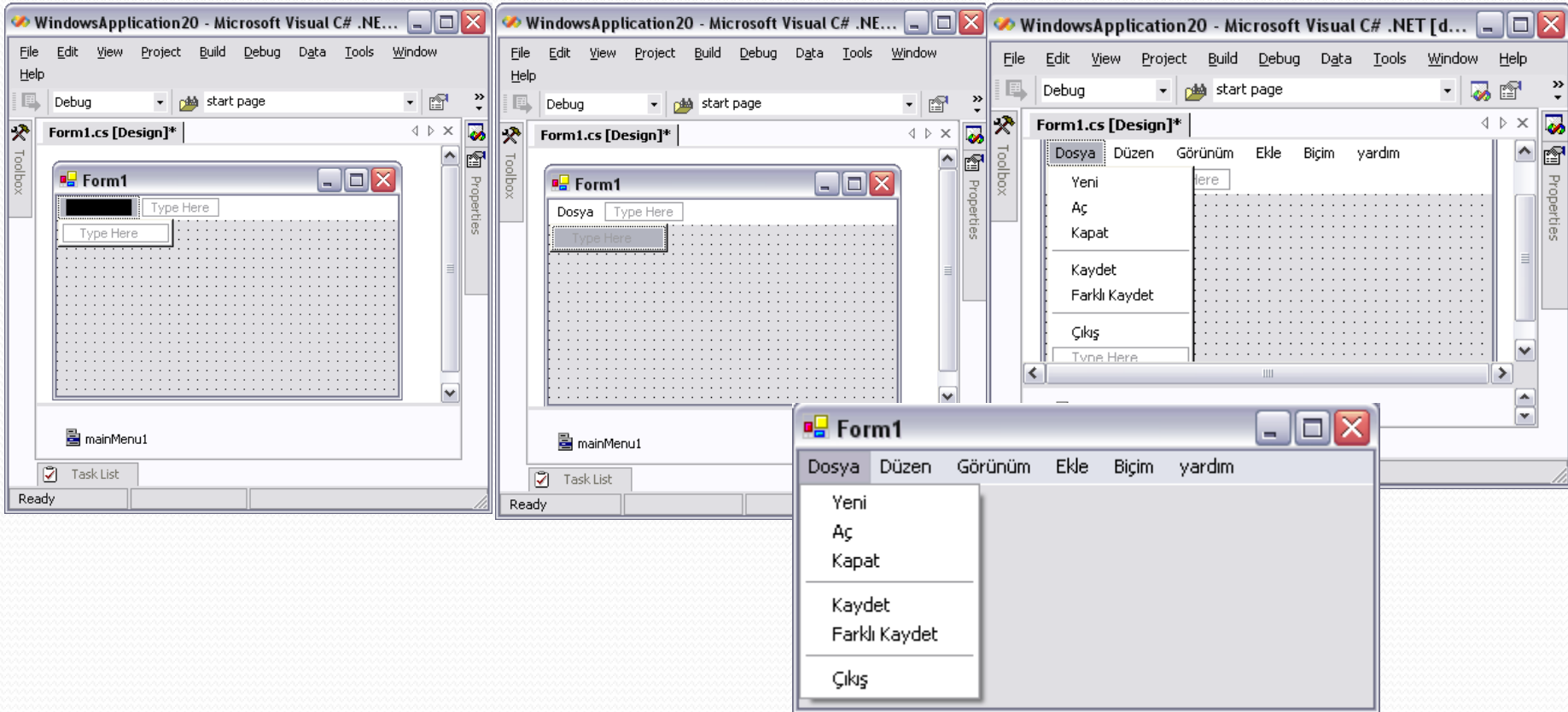
VISUAL STUDIO.NET ve FORM UYGULAMALARI

- **Menü Oluşturmak -MenuStrip**
- Araç kutusundan (ToolBox) MenuStrip ögesini seçip forma eklenerek menü oluşturulur. Menu isimleri *Type Here* yazan kutucuğa yazılır. Alt menü oluşturmak için *Type Here* ögesinin sağındaki ok işarete tıklanır ve aynı işlem tekrar edilir.



VISUAL STUDIO.NET ve FORM UYGULAMALARI

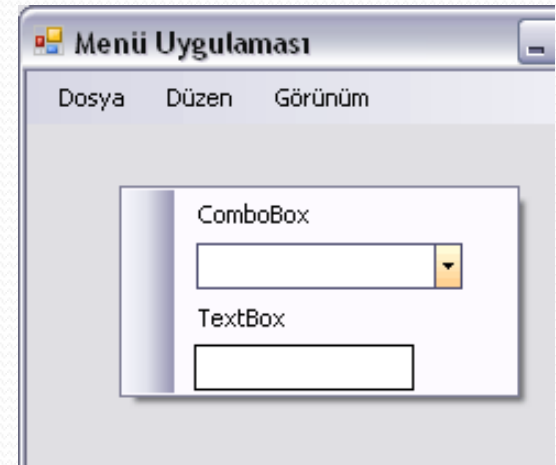
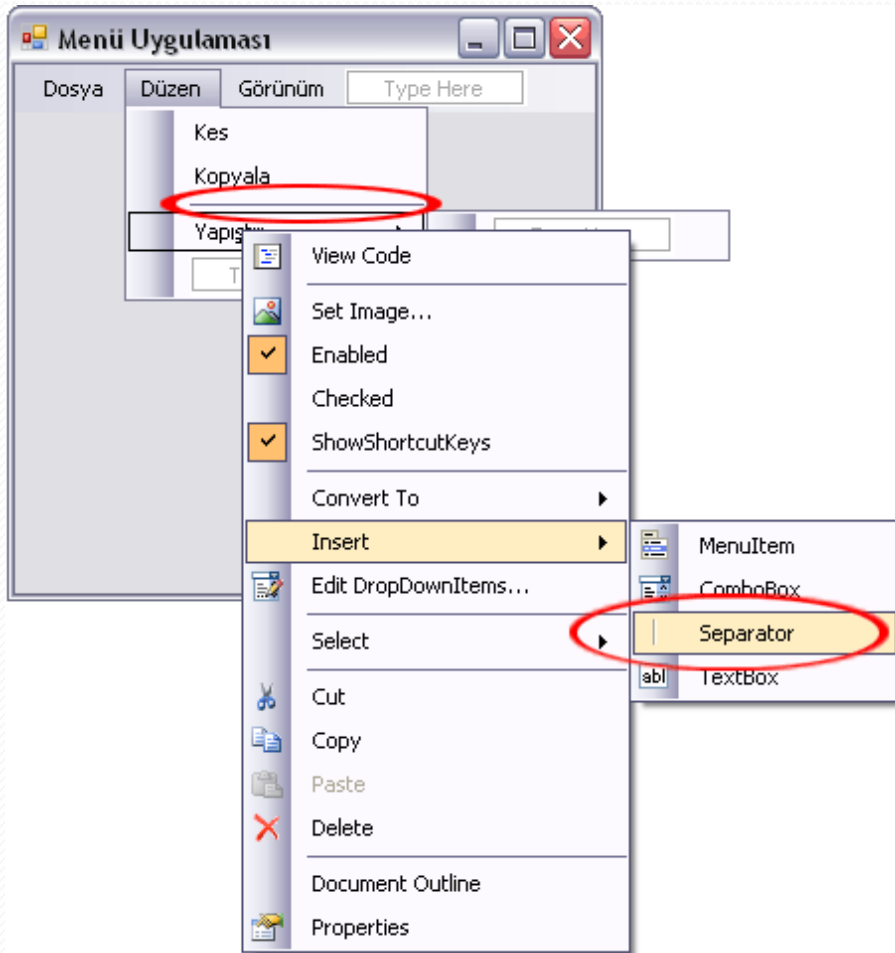
- **MenuStrip**



- `private void yeniToolStripMenuItem_Click(object sender, EventArgs e)`
- `{MessageBox.Show("Yeni Komutunu verdiniz"); }`

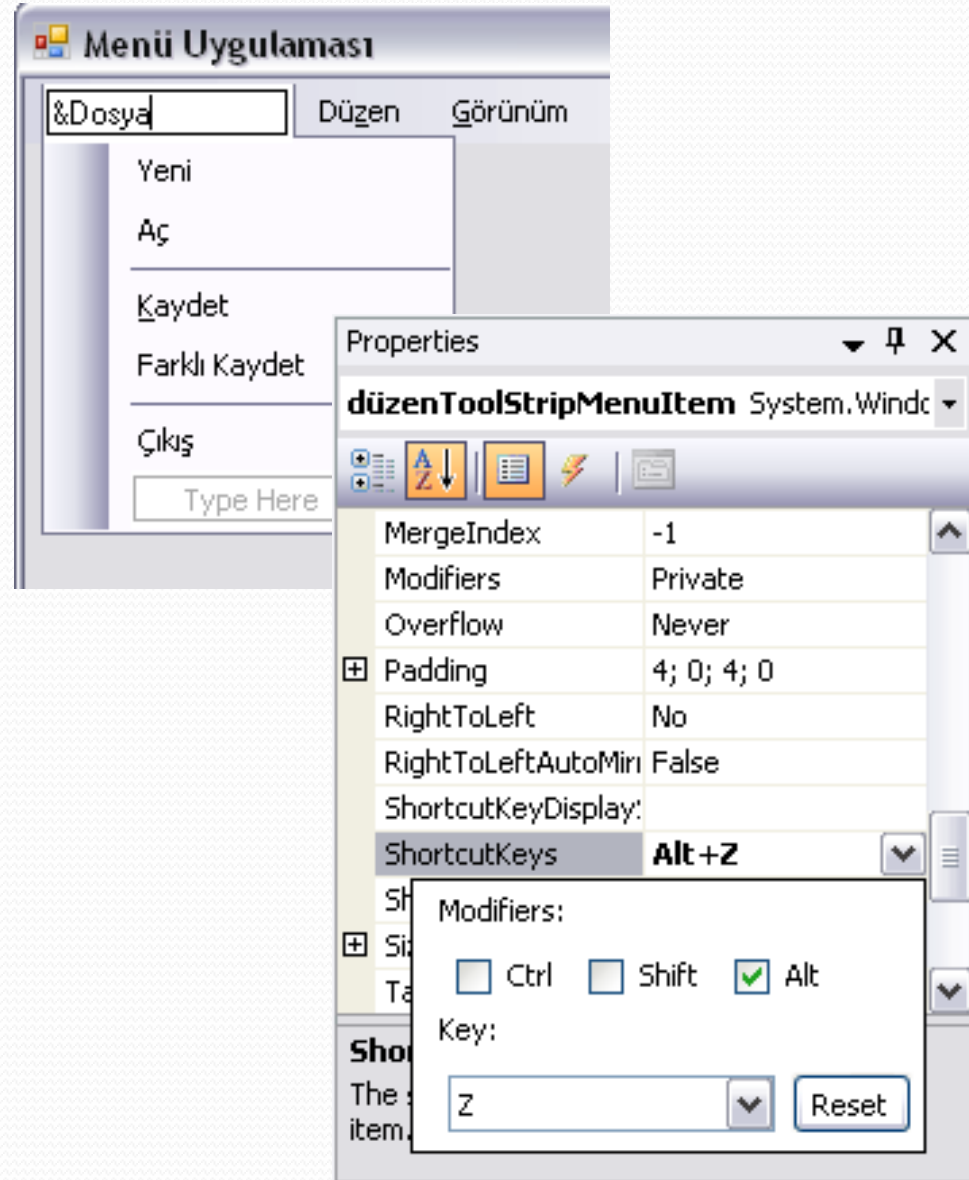
VISUAL STUDIO.NET ve FORM UYGULAMALARI

- **MenuStrip**
- Menü öğelerine ayırıcı çubuk (seperator bar) eklemek için,



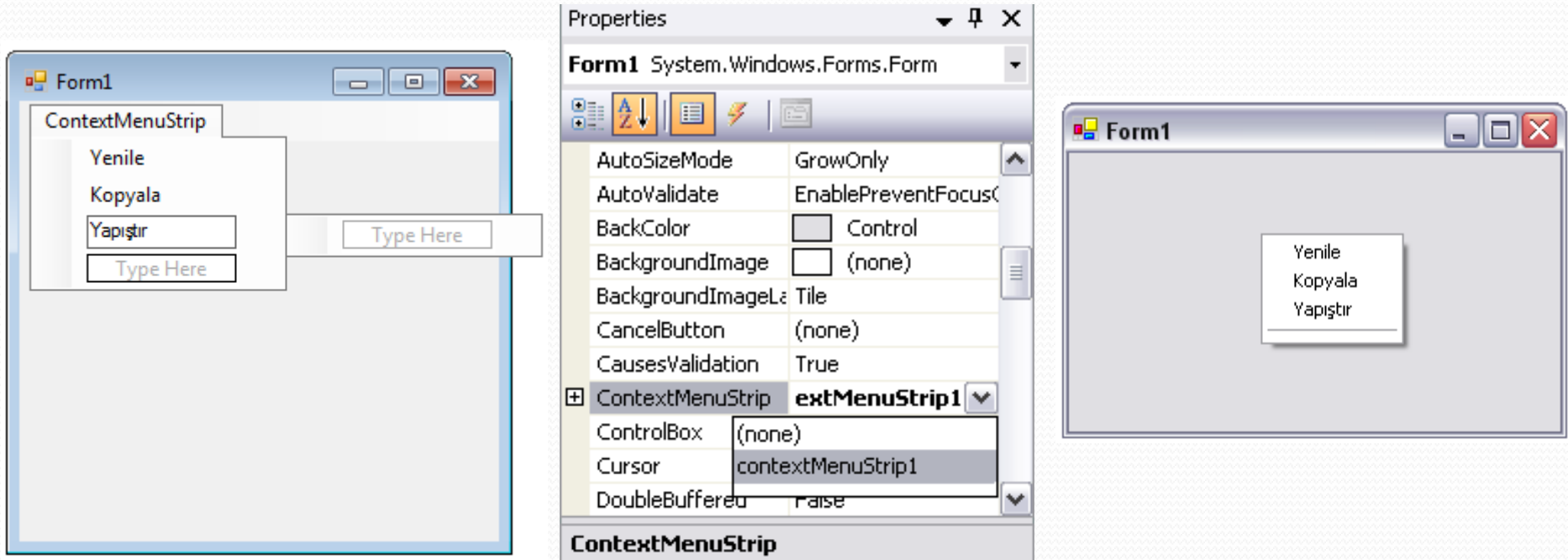
VISUAL STUDIO.NET ve FORM UYGULAMALARI

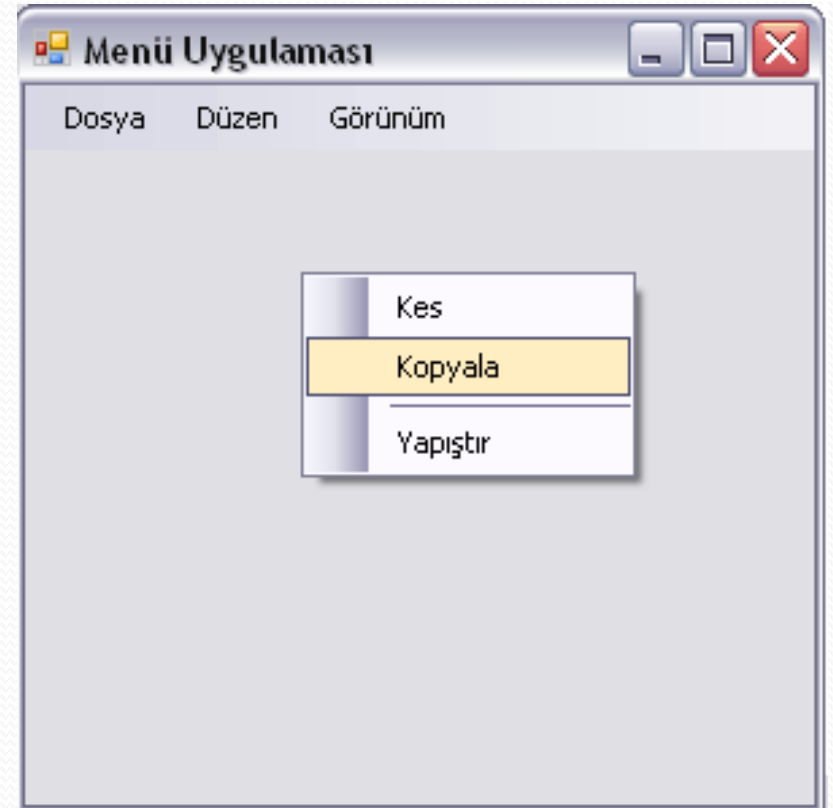
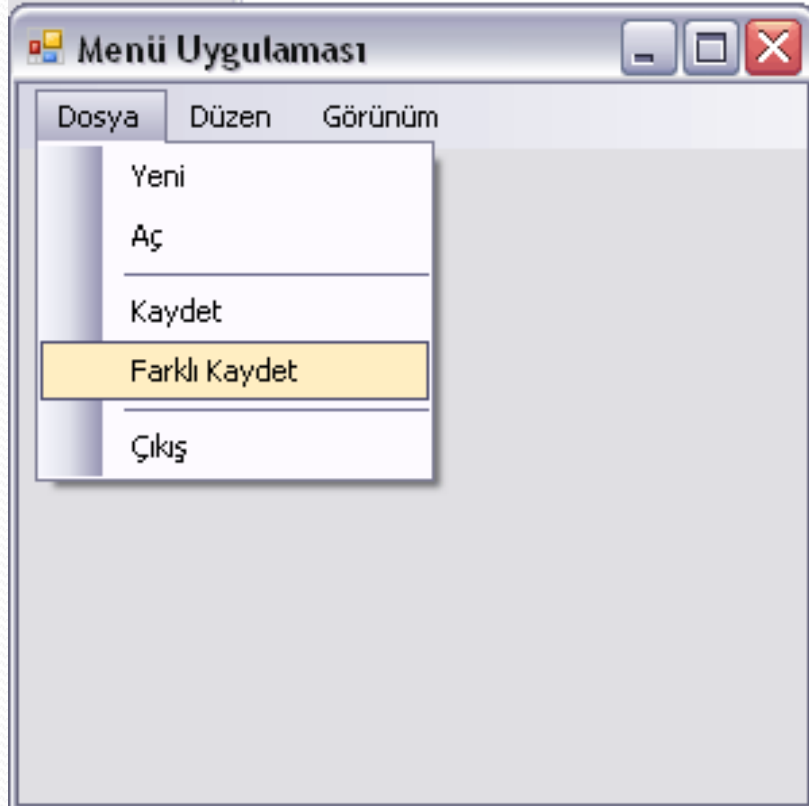
- **Menü Erişimi:**
- **& (Ampersand)** işareti eklenen karaktere ALT+karakter ile klavyeden erişebilirsiniz (ALT+D)
- **Kısa yol Tuşları :**
- **Properties-ShortcutKeys** özelliği ile menü ve altında bulunan elamanlara farklı kısa yollar ataya bilirsiniz.



VISUAL STUDIO.NET ve FORM UYGULAMALARI

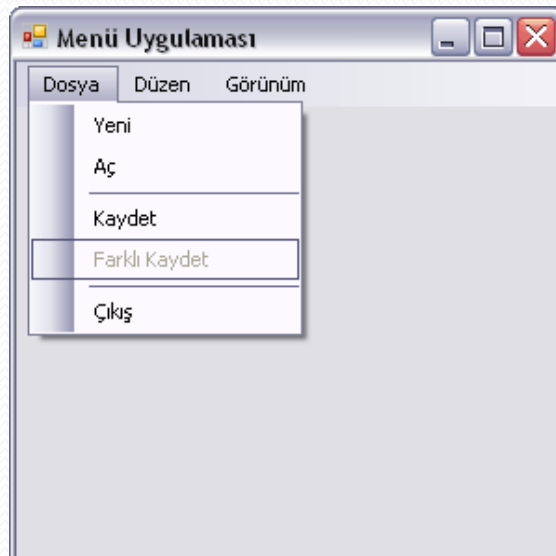
- **Menü Oluşturmak -ContextMenuStrip**
- İçerik menüleri genellikle, öge üzerinde sağ tık yapıldığında açılan menüler için kullanılır. İçerik menüsünü oluşturduktan sonra kullanabilmek için formun özelliklerinde bunun tanımlanması gerekir.





VISUAL STUDIO.NET ve FORM UYGULAMALARI

- **Çalışma Zamanında Menüleri Değiştirmek**
- Menüleri çalışma zamanındaki şartlara bağlı olarak dinamik bir şekilde yönetilebilir. Örneğin programın belirli bir görevi yerine getiremediği zamanlarda, çağrılan bir komutla bu menü pasif duruma getirilebilir veya gizlenebilir. Çalışma zamanında bir menü öğesi eklenebilir veya başka bir menü öğesiyle birleştirilebilir.
- **Aktif/Pasif Yapma**
 - `private void Form1_Load(object sender, EventArgs e)`
 - `{ farklıKaydetToolStripMenuItem.Enabled = false; }`



VISUAL STUDIO.NET ve FORM UYGULAMALARI

- **Menü Öğelerini Gizleme**

- `private void Form1_Load(object sender, EventArgs e)`
- `{düzenToolStripMenuItem.Visible = false;}`



- **Menüleri Kopyalama**

- Oluşturulan menü özellikleri içerik menüsüne veya içerik menüsü menüye kopyalanabilir. Örneğin dosya menüsüne ait içeriğin contex menüye kopyalanması.
- `ContextMenu CMenu = new ContextMenu();`
- `CMenu.MenuItems.Add(DosyaMenuItem.CloneMenu());`

VISUAL STUDIO.NET ve FORM UYGULAMALARI

- **Menüleri Birleştirme**

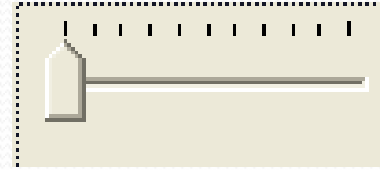
- Tek menü içerisinde birden fazla menü gösterilmesi için *MergeMenu* özelliği kullanılır.
- `private void Form1_Load(object sender, EventArgs e)`
 - `{ dosyaMenuItem.MergeMenu(myContextMenu) ; }`

- **Menü Öğeleri Ekleme**

- Çalışma anında dinamik olarak bir menü öğesine yeni bir öğe eklemek için menüye ait yapıcı metot yeni değer ile çağrılır.
- `MenuItem myItem = new MenuItem(" Pencere");`
- Bu program kodunda ise, *MenuItem* 'e Pencere adında yeni bir öğe eklenmiştir.

VISUAL STUDIO.NET ve FORM UYGULAMALARI

- **TrackBar Kontrolleri**



- Bu kontrol, **ScrollBar** kontrollerine benzer yapıdadır ancak kontrol, bir cetvel biçiminde olduğu için, üzerinde durulan pozisyon görsel olarak takip edilebilir. Kontrolün, kaydırma çubuklarından bir farkı da üzerine odaklanabilir olmasıdır. Dolayısıyla kontrolün **Value** değeri klavyede bulunan yukarı, aşağı, sağ, sol okları ve **PageUp**, **PageDown** düğmeleri ile değiştirilebilir.
- **TrackBar** kontrolünün birçok özelliği **ScrollBar** kontrollerinin özellikleriyle aynıdır. Fakat kontrolü daha esnek hale getiren birkaç özelliği vardır.

VISUAL STUDIO.NET ve FORM UYGULAMALARI

- **TrackBar Özellikleri**

Özellik	Açıklama
TickStyle	Kontrolün değerini gösteren çizgilerin pozisyonunu belirler
TickFrequency	Çizgiler arasında kalan değerlerin sayısını belirler
Orientation	Kontrolün yönünün yatay veya dikey olmasını sağlar.

VISUAL STUDIO.NET ve FORM UYGULAMALARI

- `private void Form6_Load(object sender, EventArgs e)`
- `{`
- `trackBar1.Minimum = 5;`
- `trackBar1.Maximum = 55;`
- `}`
- `private void trackBar1_Scroll(object sender, EventArgs e)`
- `{`
- `textBox1.Font = new Font("Arial", trackBar1.Value);`
- `}`



VISUAL STUDIO.NET ve FORM UYGULAMALARI

- **TabControl Kontrolleri**
- **TabControl** nesnesi, içinde sekme sayfaları tutan yapıdır. Bu sayfalar, **TabPage** nesneleri olarak oluşturulup yapılandırıldıktan sonra **TabControl** nesnesinin **TabPage** koleksiyonuna eklenir. Ekleme işlemi, **Properties** paneli ile tasarım anında da yapılabilir.

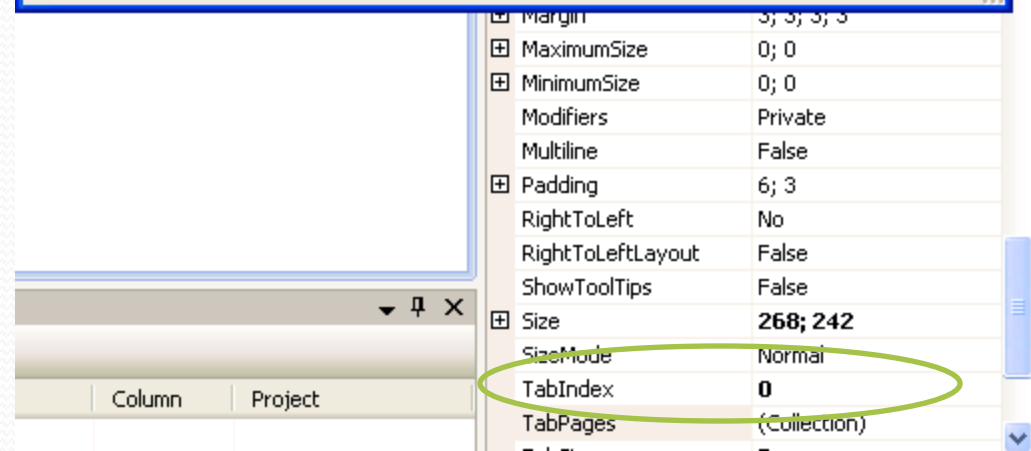
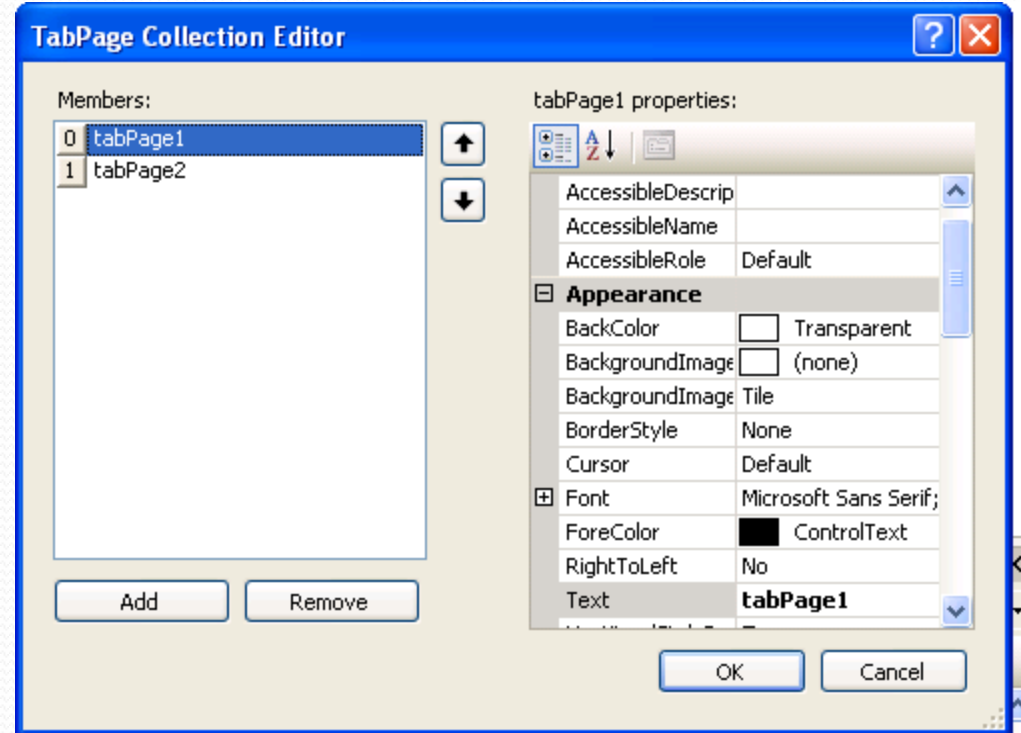


VISUAL STUDIO.NET ve FORM UYGULAMALARI

Özellik	Açıklama
HotTrack	Fare ile sekme sayfalarının üzerine gelindiğinde, isimlerinin görsel olarak değişmesini belirler
ItemSize	Sekme sayfalarının boyutunu belirler
Multiline	Eklenen sekmelerin birden fazla satırda üst üste gözükmelerini belirler
ShoWToolTips	Fare sekme sayfalarının üzerindeyken bilgi mesajının gösterilmesini belirler
SelectedTab	Seçilen sekme sayfasını belirler
SelectedIndex	Seçilen sekme sayfasının indisini belirler
TabCount	Sekme sayısını belirler
TabPage	Kontrolün içinde bulunduğu sekme sayfalarının koleksiyonudur.

VISUAL STUDIO.NET ve FORM UYGULAMALARI

- **TabControl** nesnesine **TabPage** sayfaları eklemek için tasarım anında **TabPage Collection Editor** penceresinden yararlanılabilir.



VISUAL STUDIO.NET ve FORM UYGULAMALARI

● **TabPage Özellikleri**

- Sekme sayfaları, normal form tasarımları gibi kontroller eklenerek yapılır .
- **TabPage** kontrolü **Panel** kontrolünden türer ve **Panel** kontrolünün tüm özelliklerini alır.

Özellik	Açıklama
ToolTipText	Bu özelliğin değeri, fare sayfanın üzerindeyken, bilgi mesajı olarak gösterilir. Ait olduğu TabControl nesnesinin ShowToolTip özelliği True olmalıdır.

The screenshot shows a Windows application window titled 'Form7'. It features a tabbed interface with three tabs: 'Çıkarma', 'Toplama', and 'Bölme'. The 'Çıkarma' tab is currently selected. Inside this tab, there are three text input fields. The first is labeled '1.Sayı', the second '2.Sayı', and the third 'Sonuç'. To the right of these fields is a button labeled 'Hesapla'.

VİSUAL STUDIO.NET ve FORM UYGULAMALARI



- **Timer Kontrolleri**

- Zaman değeri ayarlanabilen sayaçtır.
- Bir Windows sayacını temsil eder. Sayaç çalışmaya başladığı zaman, belirli zaman aralıklarında Tick olayı gerçekleşir. Timer kontrolünün **Interval** değeri, Tick olayının kaç milisaniyede bir gerçekleşeceğini belirler. Örneğin Interval değeri 2000 olan bir sayaç, Tick olayında yazılan kodları iki saniyede bir çalıştıracaktır.
- Sayacı başlatmak için kontrolün **Start** metodu, durdurmak için ise **Stop** metodu kullanılır. Enabled özelliği, sayacın aktif olup olmadığını belirler.

VISUAL STUDIO.NET ve FORM UYGULAMALARI

- **Timer Özellikleri**

Özellik	Açıklama
Enabled	Kontrolün aktif olup olmadığını belirler.
Interval	Sayacın hangi zaman aralığında bir çalışması gerektiğini belirler. Milisaniye cinsindedir.

- **Timer Olayları**

Olay	Açıklama
Tick	Interval özelliğinde belirtilen zaman değeri geçtiğinde gerçekleşir.

- **Timer Metotları**

Metot	Açıklama
Start	Sayacı başlatır
Stop	Sayacı durdurur

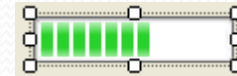
VISUAL STUDIO.NET ve FORM UYGULAMALARI

- `private void btnBasla_Click(object sender, System.EventArgs e)`
- `{ // Sayaç 5 saniyede bir çalışacak`
- `timer1.Interval = 5000;`
- `timer1.Start(); }`
- `private void timer1_Tick(object sender, System.EventArgs e)`
- `{ MessageBox.Show("Sayaç çalışıyor..."); }`
- `private void btnDur_Click(object sender, System.EventArgs e)`
- `{ timer1.Stop(); }`

- Uygulama: Kronometre yapınız

VISUAL STUDIO.NET ve FORM UYGULAMALARI

- **ProgressBar**



- Yapılan işlemlerin ilerleyişini gözlemeyi sağlar.
- Maksimum ve minimum değerleri arasındaki pozisyonu gösterir.

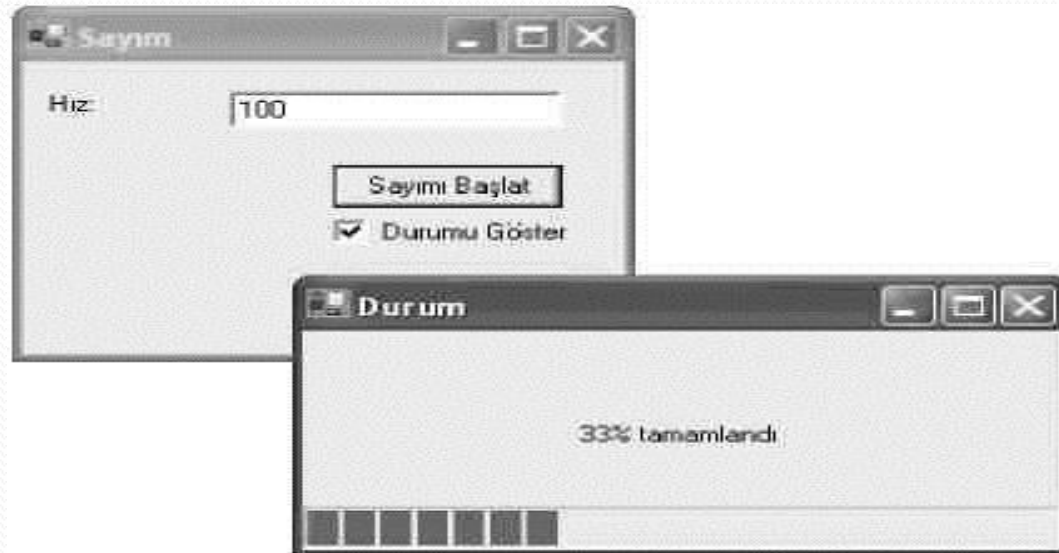
- **ProgressBar Özellikleri**

-

Özellik	Açıklama
Minimum	Kontrolün alabileceği minimum değer belirler
Maximum	Kontrolün alabileceği minimum değer belirler
Value	Kontrolün verilen değer aralığındaki pozisyonunu belirler

VISUAL STUDIO.NET ve FORM UYGULAMALARI

- **Örnek: ProgressBar** bir sayım işleminde kalan durumu göstermek için kullanılabilir. **ProgressBar** ile durumun gösterileceği ayrı bir form eklenir. Burada sayma işleminin hızı için bir **Timer** bulunur. Sayaç her işlediğinde yeni değer **ProgressBar** kontrolünde gösterilir.



VISUAL STUDIO.NET ve FORM UYGULAMALARI

- `public int kalan;`
- `private void Durum_Load(System.Object sender, System.EventArgs e)`
- `{kalan = ProgressBar1.Maximum; Timer1.Start(); }`
- `private void Timer1_Tick(System.Object sender, System.EventArgs e)`
- `{`
- `if (kalan == 0) { Timer1.Stop(); this.Close(); }`
- `int aralik = ProgressBar1.Maximum -ProgressBar1.Minimum;`
- `int oran = (aralik - kalan) / aralik * 100; Label1.Text = oran + "% tamamlandı";`
- `ProgressBar1.Value = ProgressBar1.Maximum -kalan; kalan -= 1;`
- `}`
- `private void Form1_Load(System.Object sender, System.EventArgs e)`
- `{ CheckBox1.Checked = true; }`
- `private void btnBaslat_Click(System.Object sender, System.EventArgs e)`
- `{`
- `Durum frmDurum = new Durum();`
- `frmDurum.Timer1.Interval = TextBox1.Text;`
- `if (CheckBox1.Checked) { frmDurum.ShowDialog();}`
- `}`

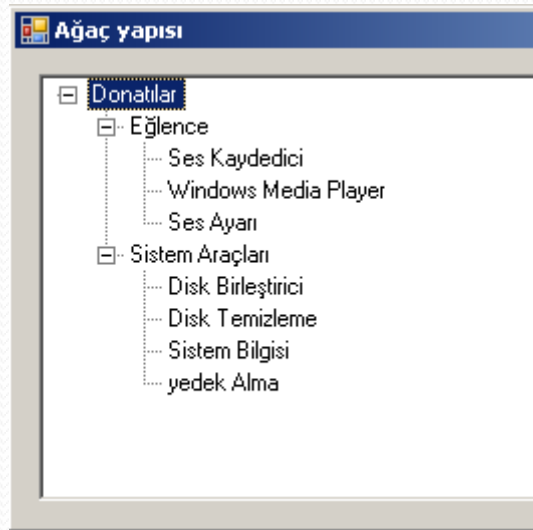
VISUAL STUDIO.NET ve FORM UYGULAMALARI

- **Treeview Kontrolleri**

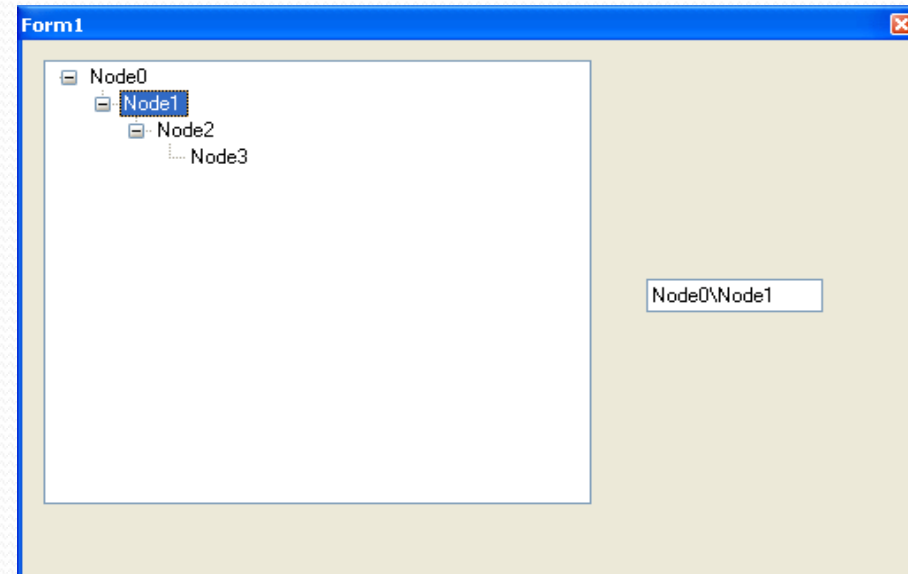
- Windows'un kullandığı ve ağaç yapısı olarak adlandırılan bir yapı içerir. Kullanıcıların bazı detayları daha net anlayabilmesi ve görebilmesi için oluşturulur. (Bu kontrole ait, silme ekleme, seçme vs. özellikler notlarda mevcuttur. Bu kısımlar sizin tarafınızdan incelenecektir.)
- **Özellikleri:**
- **treeView1.Nodes:** treeView satır işlemleri için kullanılır.
- **treeView1.Nodes.Add():** Ağaç yapısına düğüm noktası eklemek için kullanılır. Daha sonra bu düğüm noktalarına alt bileşenler eklenir.
- Properties / Nodes (Collections) kısmından eklenebileceği gibi kodlarla ekleme yapılabilir.

VISUAL STUDIO.NET ve FORM UYGULAMALARI

- `private void Form1_Load(object sender, EventArgs e)`
- `{ treeView1.Nodes.Add("Donatılar");`
- `treeView1.Nodes[0].Nodes.Add(new`
`TreeNode("Eğlence"));`
- `treeView1.Nodes[0].Nodes[0].Nodes.`
`Add(new TreeNode("ses kaydedici"));`
- `}`



- **SelectedNode** : Bu özellik ile seçilen elaman ile ilgili işlemler yapılabilir.
- `private void treeView1_AfterSelect`
`(object sender, TreeViewEventArgs e)`
- `{ textBox1.Text =`
`treeView1.SelectedNode.FullPath;`
`}`

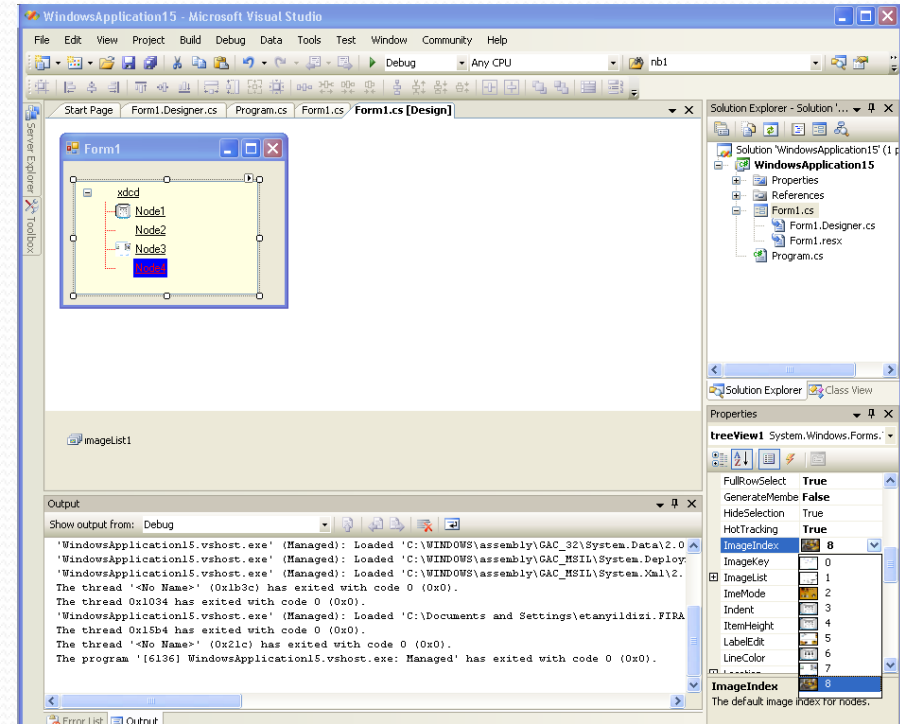


VISUAL STUDIO.NET ve FORM UYGULAMALARI

- **treeView1.CheckBoxes:** true değeri aktarılırsa her satırın başına checkbox eklenir. Kutucukların işaretlenmesi ile aktif hale gelir.
- **private void Form1_Load(object sender, System.EventArgs e)**
- **{**
- **treeView1.CheckBoxes = true;**
- **}**



- ImageList veya image özelliği ile ağacın düğümlerine resim eklenebilir.



VISUAL STUDIO.NET ve FORM UYGULAMALARI

- **NumericUpDown**
- Bu kontrol kullanıcının, sayısal bir değeri girmesini veya yukarı aşağı okları ile seçmesini sağlar.
- **NumericUpDown Özellikleri**

Özellik	Açıklama
Hexadecimal	Sayıların on altılık tabanda görüntülenmesini belirler.
Increment	Aşağı yukarı oklar kullanıldığında, sayıların artma ve azalma adımlarını belirler.
Maximum	Kontrolde gösterilen sayıların alabileceği maksimum değeri belirler.
Minimum	Kontrolde gösterilen sayıların alabileceği minimum değeri belirler.
ThousandSeparators	Sayıların basamak ayracını gösterilmesini belirler.
Value	Kontrolü gösterdiği sayı değerini belirler.
ReadOnly	True değerini alırsa kullanıcının giriş yapmasını engeller.

VISUAL STUDIO.NET ve FORM UYGULAMALARI

- NumericUpDown Olayları**

Olay	Açıklama
ValueChanged	Aşağı yukarı oklar kullanıldığında, sayıların artma ve azalma adımlarını belirler.

- NumericUpDown Metotları**

Metot	Açıklama
DownButton	Aşağı düğmesine basar ve sayı değerini düşürür.
UpButton	Yukarı düğmesine basar ve sayı değerini artırır.

VISUAL STUDIO.NET ve FORM UYGULAMALARI

- **Örnek:** Alarm kurarken, tarih ve zaman değerlerinin ayarlanması NumericUpDown kontrolü ile yapılabilir.



The screenshot shows a Windows Form titled "Form4" with a blue title bar and standard Windows window controls (minimize, maximize, close). The form has a light yellow background and a blue border. Inside the form, there is a section titled "Tarih - Saat Ayarla" (Date - Time Set) in blue text. Below this title, there are two rows of controls. The first row is for the date, with labels "Tarih" and three NumericUpDown controls showing the values 26, 3, and 2009. The second row is for the time, with labels "Saat" and two NumericUpDown controls showing the values 19 and 36. At the bottom of the form, there is a text label "Alarm Saati : 26.3.2009 19:36" and a button labeled "Alarm Kur".

VISUAL STUDIO.NET ve FORM UYGULAMALARI

- string tarih; string zaman;
- private void Form4_Load(object sender, EventArgs e)
- { nAy.Minimum = 1; nGun.Minimum = 1;
- nYil.Minimum=1; nYil.Maximum = 9999;
- nAy.Maximum = 12; nGun.Maximum = 31;
- nSaat.Minimum = 0; nDakika.Minimum = 0;
- nSaat.Maximum = 23; nDakika.Maximum = 59;
- nYil.Value = DateTime.Now.Year; nAy.Value = DateTime.Now.Month;
- nGun.Value = DateTime.Now.Day; nSaat.Value = DateTime.Now.Hour;
- nDakika.Value = DateTime.Now.Minute;
- }
- private void button1_Click(object sender, EventArgs e)
- { tarih = nGun.Value + "." + nAy.Value + "." + nYil.Value+" ";
- zaman = nSaat.Value + ":" + nDakika.Value;
- label3.Text = "Alarm Saati :"+tarih + zaman;
- }

VISUAL STUDIO.NET ve FORM UYGULAMALARI

- **DomainUpDown Kontrolü**
- NumericUpDown kontrolü ile aynı yapıdadır ancak sayısal değerler yerine Object tipinde değerler tutar. Bu değerler kontrolün Items koleksiyonunda tutulur. Kontrol, bu özelliği ile liste kutusuna benzemektedir.
- **DomainUpDown Özellikleri**

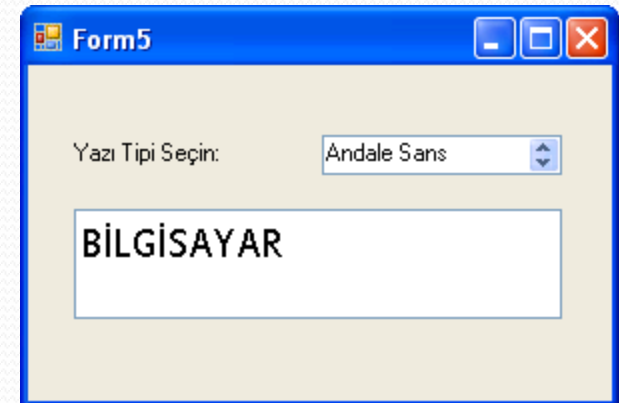
Özellik	Açıklama
Items	Kontrolün öğelerinin tutulduğu dinamik bir listedir
SelectedItem	Kontrolde seçilen öğeyi tutar
Wrap	Liste sonuna gelindiğinde baştaki veya sondaki öğeye geri dönülmesini belirler.

- **DomainUpDown Olayları**

Olay	Açıklama
SelectedItemChanged	Kontrolde seçilen öğe değiştiği zaman gerçekleşir.

VISUAL STUDIO.NET ve FORM UYGULAMALARI

- Örnek: Metin kutularının değiştirilmek istenen yazı tipleri DomainUpDown kontrolünde tutulabilir.
- `private void Form5_Load(object sender, EventArgs e){`
- `for (int i = 10; i < 100; i++)`
- `{domainUpDown1.Items.Add(System.Drawing.FontFamily.Families[i].Name);}`
- `domainUpDown1.Wrap = true; }`
- `private void domainUpDown1_SelectedItemChanged(object sender,`
`EventArgs e)`
- `{if (domainUpDown1.SelectedIndex>=0)`
- `{ textBox1.Font=new Font(Convert.ToString`
`(domainUpDown1.SelectedItem),15);`
- `}`
- `}`



VISUAL STUDIO.NET ve FORM UYGULAMALARI

- **HScrollBar / VScrollBar Kontrolleri**
- **Horizontal - Vertical ScrollBar** kontrolleri, sayısal bir değer taşıyan kaydırma çubuklarıdır. Tuttukları değerlerin sayısal olması bakımından **NumericUpDown** kontrolüne benzer.
- Bu kontroller, üzerlerinde kaydırma çubukları olmayan kontroller üzerinde kullanılabilir. Örneğin bir **ListBox**, **Panel** gibi kontrollerin kendi **ScrollBar** kontrolleri vardır. **TextBox** kontrolünün de ilgili özellikleri ayarlanarak yatay ve dikey **ScrollBar** kontrolleri gösterilebilir.

VISUAL STUDIO.NET ve FORM UYGULAMALARI

- ScrollBar Özellikleri**

Özellik	Açıklama
Value	Kaydırma çubuğunun pozisyonuna göre alınan değeri tutar.
SmallChange	Kontrolü, üstündeki oklar ile kaydırıldığı zaman eklenecek ya da çıkartılacak değeri tutar.
LargeChange	Kontrolü, kaydırma çubuğundaki boşluğa tıklanarak kaydırıldığında zaman eklenecek ya da çıkartılacak değeri tutar.
Minimum	Value özelliğinin alabileceği değeri minimum tutar
Maximum	Value özelliğinin alabileceği maksimum değeri tutar

- ScrollBar Olayları**

Olay	Açıklama
Scroll	Çubuklar kaydırıldıkları zaman gerçekleşir.
ValueChanged	Kod ile ya da çubuklar kaydırılınca Value özelliği değiştiği zaman gerçekleşir.

VISUAL STUDIO.NET ve FORM UYGULAMALARI

- **Örnek:** Bir **ComboBox** kontrolünün öğelerini listelemek için, aşağıya doğru bir kaydırma çubuğu görüntülenir. Ancak listedeki bazı elemanların kontrole sığmıyorsa, çalışma anında bu kontrolün genişliği artırılabilir.
- ```
private void Form1_Load(System.Object sender, System.EventArgs e)
```
- ```
{
```
- ```
hsGenislik.Maximum = ComboBox1.Width * 2;
```
- ```
hsGenislik.Value = ComboBox1.Width;
```
- ```
}
```
- ```
private void hsGenislik_Scroll( System.Object sender,
```

```
System.Windows.Forms.ScrollEventArgs e )
```
- ```
{ ComboBox1.Width = hsGenislik.Value; }
```





# Veritabanı Uygulamaları

## ADO.NET

# VERİTABANI UYGULAMALARI

- **Veri Merkezli Uygulamalar:**
- Veri Depolama, Bağlantılı (Connected) ve Bağlantısız (Disconnected) Veri Ortamları, Veri Erişim Yöntemleri olarak ele alınmaktadır.
- **Veri Depolama**
- Veriye erişmek için çeşitli veri depolama yöntemleri geliştirilmiştir.
  - Yapısal Olmayan Yöntem
  - Yapısal Yöntem
  - Hiyerarşik Yöntem
  - İlişkisel Veritabanı Yöntemi
  - Nesne Yönelimli Veri Tabanı Yöntemi

# Veri Depolama

- **Yapısal Olmayan:** Bu yöntem ile depolanan veriler için belirli bir sınıflandırma ve sıralama yoktur. Veriler düz bir şekilde kaydedilir. Örneğin basit not dosyaları.
- **Yapısal:** Bu yöntem ile depolanan veriler çeşitli gruplara ayrılarak saklanır fakat bu gruplar arasında bir alt-üst ayrımı yapılmaz. Örneğin virgülle ayrılmış dosyalar (csv), Excel belgeleri.
- **Hiyerarşik:** Hiyerarşik depolama yöntemini ağaç yapısına benzetebiliriz. Bu yöntemde veriler çeşitli kategorilere bölünerek depolanır. Her bir kategorinin içerisinde alt kategorilerde olabilir. Örneğin XML (eXtensible Markup Language) dosyalar.
- **İlişkisel Veritabanı:** İlişkisel veritabanlarında veriler tablolar üzerinde depolanır. Tablo içerisindeki her bir satır kaydı, her bir sütun ise veriyi ifade eder. Örneğin SQL Server, Oracle, Access.
- **Nesne Yönelimli Veritabanı:** En gelişmiş veri depolama yöntemidir. Bu yöntemde veriler; ihtiyaca göre gruplandırılarak, nesneler içerisinde saklanır. Örneğin Versant, AOL.
- **ADO.NET bu depolama tekniklerinin tümünü destekler.**

# Bağlantılı ve Bağlantısız Veri Ortamları

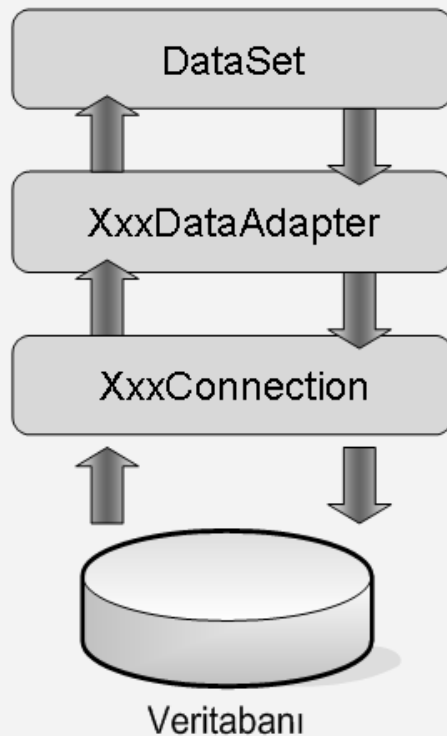
- **Bağlantılı (Connected) Veri Ortamları**
- Bağlantılı veri ortamları, uygulamaların veri kaynağına sürekli bağlı kaldığı ortamlardır. Bu ortamlarda veri alma ve değiştirme işlemleri uygulama ile veri kaynağı arasında bağlantı kurulduktan sonra gerçekleştirilir. Bağlantılı veri ortamlarında, veri işlemleri gerçekleştiği sürece bağlantı açık kalır.
- **Avantajları:**
  - En güvenli veri ortamı.
  - Erişimler eş zamanlı.
- **Dezavantajları:**
  - Sabit bir ağ bağlantısının olması gerekir.
  - Ağ trafiğinin yoğunluğunu artırır.

# Bağlantılı ve Bağlantısız Veri Ortamları

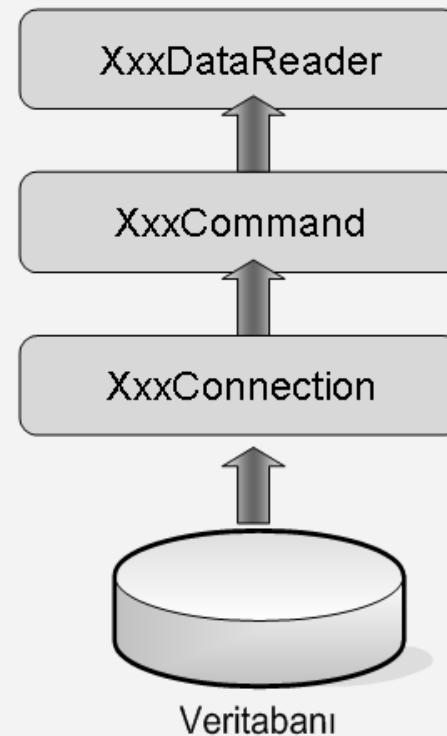
- **Bağlantısız (Disconnected) Veri Ortamları**
- Bağlantısız veri ortamı, uygulamanın veri kaynağına sürekli bağlı kalmadığı veri ortamıdır. Bağlantı, veri alış verişi yapılırken açılır, işlem bittikten sonra kapatılır.
- **Avantajları:**
  - Taşınabilen aygıtlarla (Laptop, Pocket PC) girilen veriler, istenilen zamanda veri ortamlarına aktarılabilir.
  - Uygulama performansını artırır.
- **Dezavantajları:**
  - Verinin güncelliği sağlanmalıdır.
  - Veri çakışması önlenmelidir.

## Bağlantılı ve Bağlantısız Veri Ortamları

### Bağlantısız (Disconnected) Veri Ortamları



### Bağlantılı (Connected) Veri Ortamları



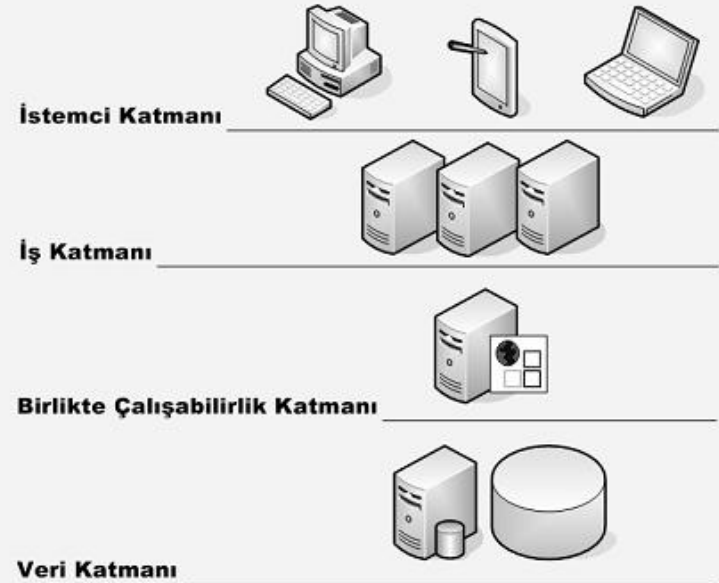
# Veri Erişim Yöntemleri

- Veriye erişmek için pek çok yöntem geliştirilmiştir. Bu yöntemlerin bazılarında amaç **yerleşim (veriyi saklama)**, bazılarında ise **paylaşım** olmuştur.
- Amacın veriyi saklamak olduğu durumlarda paylaşım konusunda çözüm aranmış, amacın veriyi birçok kullanıcı arasında paylaşmak olduğu durumda ise ana verinin nerede saklanacağı konusunda çözüm yolları aranmıştır.
- Kullanıcı sayısının ve verinin boyutunun artmasıyla, veri erişimi için bilinen modeller de oldukça gelişmiştir. Birebir veri paylaşımı yerine, internet üzerinden çoklu kullanıcı desteğine açık veri erişim modelleri geliştirilmiştir. Günümüzde gelinen son nokta ise, her an her yerden veriye kolayca erişmemizi sağlayan **XML Web Servis** modelidir.
- Veri merkezli uygulamalar geliştirmek için veri erişim modelleri kullanılır.
- Bir veri erişim modelinde ki mantıksal her birime **katman (tier)** denir.
- Veri merkezli bir uygulamada katman sayısı makine sayısına bağlı değildir. Katman sayısını veri erişim modelindeki düzeyler belirler.

# Veri Eriřim Yöntemleri

- **İstemci Katmanı (Client tier):** Sunum ya da kullanıcı servis katmanı olarak da bilinir. Bu katman kullanıcı ara yüzünü içermektedir.
- **İř Katmanı (Business tier):** Bu katman, uygulamanın veri kaynağı ile etkileřen bölümüdür.
- **Birlikte çalışabilirlik Katmanı (Interoperability tier):** Platform ve dilden bağımsız, her tür veriye etkileřim sağılayan katmandır. Bu katmana herhangi bir işletim sistemi üzerinde bulundurulabilen XML Web Servislerini örnek verebiliriz.
- **Veri Katmanı (Data tier):** Veriyi içeren katmandır.

## Veri Eriřiminde Katmanlar





# ADO.NET

- **ADO (ActiveX Data Objects)**, farklı veri kaynaklarına hızlı ve güvenli erişim için Microsoft tarafından geliştirilen nesne modelidir.
- **ADO.NET** ise ADO teknolojisinin en yeni versiyonudur. ADO ile aynı programlama modelini kullanmamakla birlikte, ADO modelinden gelen pek çok çözüm yolunu da beraberinde getirir.
- ADO.NET nesne modeli iki ana bölümden oluşmaktadır.
  - **DataSet Sınıfları**
  - **.NET Veri Sağlayıcı Sınıfları**

## ADO.NET Nesne Modeli

- **DataSet Sınıfları:**

- Çevrimdışı ortamlar için veri depolama ve yönetme işlemlerini sağlar. DataSet sınıfları veri kaynağından bağımsız her tür uygulama ve veritabanı için kullanılabilir. Özellikle İlişkisel Veritabanı, XML ve XML Web Servisleri üzerinden veri çekmek için kullanılır.

- **.NET Veri Sağlayıcı Sınıfları:**

- .NET veri sağlayıcı sınıfları, farklı türdeki veritabanlarına bağlanmak için kullanılır. Bu sınıflar sayesinde istenilen türdeki veri kaynağına kolayca bağlantı kurulabilir, veri çekilebilir ve gerekli güncelleme işlemleri yapılabilir. ADO.NET nesne modeli, aşağıdaki veri sağlayıcı sınıflarını içerir: **SQL Server** .NET Veri Sağlayıcısı, **OLE DB** .NET Veri Sağlayıcısı, Diğer .NET Veri Sağlayıcıları

## ADO.NET Veri Sağlayıcıları

- NET veri sağlayıcıları, **System.Data** isim alanı içinde tanımlanmıştır.
- **SQL Server .NET:** SQL Server 7.0 ve SQL Server 2000 veritabanlarına hızlı bağlantı sağlar. SQL Server bağlantı nesneleri **System.Data.SqlClient** isim alanında bulunur.
- **OLE DB .NET:** SQL Server 6.5 ve daha öncesi sürümlerine, Oracle, Sybase, DB2/400 ve Microsoft Access veri tabanlarına bağlantı kurmayı sağlar. OLE DB bağlantı nesneleri **System.Data.OleDb** isim alanında bulunur.
- **ORACLE .NET:** Oracle veritabanlarına bağlantı için tasarlanmış veri sağlayıcısıdır. Oracle bağlantı nesneleri **System.Data.OracleClient** isim alanında bulunur.
- **ODBC .NET:** Diğer veritabanlarını destekleyen genel bir veri sağlayıcıdır. ODBC bağlantı nesneleri **System.Data.ODBC** isim alanında bulunur.

## ADO.NET Veri Sağlayıcıları

- Her bir veri sağlayıcısı içerisinde, birçok bağlantı nesnesi bulunur.
  - **Connection**: Bağlantı kurmak için kullanılır.
  - **Command** : Veritabanına sorgu yollamak için kullanılır.
  - **DataReader** : Çevrim içi bağlantı ile sadece veri okuma.
  - **DataAdapter** : Çevrim dışı bağlantılarda veri işleme nesnesi.
- Veri Kaynaklarına Bağlanmak
  - Veri Sağlayıcı Seçmek
  - Bağlantı oluşturmak
  - Bağlantı Yönetimini seçmek

## Veri Sağlayıcı Seçmek

- Uygulama ile veritabanı arasında bağlantı kurmak ve kurulan bağlantı üzerinden kayıtları almak, değiştirmek ve silmek için veri sağlayıcılarını kullanır.
- Microsoft .NET Framework, veritabanları ile bağlantı kurmak için farklı veri sağlayıcılarını destekler.
  - SQL Server .NET
  - OLEDB .NET
  - ODBC .NET

## Veri Sağlayıcı Seçmek

- **System.Data.SqlClient** isim alanı içerisinde çevrimiçi bağlantılar geliştirmek için **SqlConnection**, **SqlCommand**, **SqlDataReader** sınıfları kullanılır.
- **SqlConnection**; MS SQL Server üzerinde bağlantı açmak ve kapatmak için kullanılan sınıftır.
- **SqlCommand**; MS SQL Server üzerinde Stored Procedure (Saklı Yordamlar) veya SQL Cümleleri çalıştırmak için kullanılan sınıftır.
- **SqlDataReader**; MS SQL Server üzerinde SqlCommand ile çalıştırılan SELECT sorguların sonuçlarını geri döndürmek için kullanılan sınıftır.

## Veri Sağlayıcı Seçmek

- **System.Data.SqlClient** isim alanı içerisinde çevrimdışı bağlantılar geliştirmek için **SqlConnection**, **SqlDataAdapter**, **DataSet** sınıfları kullanılır.
- 
- **SqlConnection**; MS SQL Server üzerinde bağlantı açmak ve kapatmak için kullanılan sınıftır.
- **SqlDataAdapter**; MS SQL Server'dan çekilen verileri DataSet içerisine ve DataSet'e çevrimdışı eklenmiş verileri MS SQL Server'a aktarmak için kullanılan sınıftır.
- **DataSet**; SqlDataAdapter nesnesinden gelen kayıtları çevrimdışı depolamak ve yönetmek için kullanılan sınıftır. DataSet tüm veri sağlayıcı sınıflar için ortaktır.

## Veri Sağlayıcı Seçmek

- **System.Data.OleDb** isim alanı içerisinde çevrimiçi bağlantılar geliştirmek için **OleDbConnection**, **OleDbCommand**, **OleDbDataReader** sınıfları kullanılır.
- 
- **OleDbConnection**; Access veya diğer veritabanları üzerinde bağlantı açmak ve kapatmak için kullanılan sınıftır.
- **OleDbCommand**; Access veya diğer veritabanları üzerinde Stored Procedure (Saklı Yordamlar) veya SQL Cümleleri çalıştırmak için kullanılan sınıftır.
- **OleDbDataReader**; Access veya diğer veritabanları üzerinde OleDbCommand ile çalıştırılan SELECT sorguların sonuçlarını geri döndürmek için kullanılan sınıftır.



## Veri Sağlayıcı Seçmek

- **System.Data.OleDb** isim alanı içerisinde çevrimdışı bağlantılar geliştirmek için **OleDbConnection**, **OleDbDataAdapter** sınıfları kullanılır.
- 
- **OleDbConnection**; Access veya diğer veritabanları üzerinde bağlantı açmak ve kapatmak için kullanılan sınıftır.
- **OleDbDataAdapter**; Access veya diğer veritabanlarından çekilen verileri DataSet içerisine ve DataSet'e çevrimdışı eklenmiş verileri ilgili veritabanına aktarmak için kullanılan sınıftır.

## Bağlantı Oluşturmak

- Bağlantı cümlesi, veri kaynağına bağlanmak için gerekli bilgileri tutar.
- **Provider:** Sadece OleDbConnection nesnelerinde kullanılır. Bağlantı sağlayıcısının ismini tutar.
- **ConnectionTimeout veya Connect Timeout :** Veritabanı bağlantı için beklenmesi gereken maksimum saniye sayısıdır. Varsayılan değer 15 saniye dir.
- **Initial Catalog:** Veri tabanı adı
- **Data Source:** Veri tabanı için dosya adı.
- **Password (pwd):** Hesap bağlantı şifresi
- **User Id (uid):** Hesap kullanıcı ismi

## Bağlantı Oluşturmak

- **Integrated Security veya Trusted Connection:** Bağlantının güvenli olup olmadığını belirten parametredir.
- **Persist Security Info:** Varsayılan değeri false olur. Bu durumda güvenlik için hassas bilgileri geri döndürmez. True olursa, servere bağlantılarda kullanıcı adı ve şifresi istenir.
- **WorkstationID (wid) :** Workstation veya client(istemci) adını belirtir.
- **Packet Size:** Client(istemci)-server(sunucu) arası veri transferinde kullanılan paketlerin boyutunu belirtir.
- **Mode :** Veritabanını Read-only(Sadece okunur) ya da Write(Yazılabilir) modunu belirtir. SQL Server bağlantılarında kullanılmaz.

## Bağlantı Oluşturmak

- **Provider** parametresinin Access, SQL Server ve Oracle veri tabanlarına göre alacağı değerler;
  - 
  - Tür
  - **SQLOLEDB**
  - **MSDAORA**
  - **Microsoft.Jet.OLEDB.4.0**
- | Açıklama                                  |
|-------------------------------------------|
| SQL Server için Microsoft OLE DB Provider |
| ORACLE için Microsoft OLE DB Provider     |
| Microsoft Jet için OLE DB Provider        |

## Bağlantı Oluşturmak

- **Ms Access ile OLEDB Bağlantı Cümleleri**
- **Access'e Bağlantı:**
- "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=DB\_Name.mdb;"
- **Access'e Çalışma Grubu dosyası üzerinden Bağlantı:**
- "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=Db\_Name.mdb; Jet OLEDB:System Database=Db\_Name.mdw"
- **Access'e Parola Korumalı Bağlantı:** "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=Db\_Name.mdb; Jet OLEDB:Database Password=sifreniz"
- **Network'teki Access'e Bağlantı:** "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=\\Server\_Name\Share\_Name\Share\_Path\Db\_Name.mdb"
- **Remote Server(UzakServer) üzerindeki bir Access'e Bağlantı:** "Provider=MS Remote; Remote Server=http://Your-Remote-Server-IP; Remote Provider=Microsoft.Jet.OLEDB.4.0; Data Source=Db\_Name.mdb"

## Bağlantı Oluşturmak

- **Ms Access 2007 ile OLEDB Bağlantı Cümleleri**
- **Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\myFolder\myAccess2007file.accdb;Persist Security Info=False;**
- **Database password**
- **Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\myFolder\myAccess2007file.accdb;Jet OLEDB:Database Password=MyDbPassword;**

## Bağlantı Oluşturmak

- **SQL Server ile ODBC Bağlantı Cümleleri**
- **SQL Server sunucusuna SQL Authentication (Kimlik Doğrulama) ile bağlanmak:**  
"Driver={SQL Server}; Server= Server\_Name; Database=Db\_Name; Uid=Username;  
Pwd= sifreniz;"
- **SQL Server sunucusuna Windows Authentication ile bağlanmak:**
- "Driver={SQL Server}; Server= Server\_Name; Database=DB\_Name;  
Trusted\_Connection=yes;"

## Bağlantı Oluşturmak

- **SQL Server ile Sql Server Bağlantı Cümleleri**
- **SQL Server sunucusuna Authentication ile bağlanmak**
- "Data Source=\_Server\_Name;Initial Catalog=Db \_Name;User SQL Id=Username;Password=sifreniz;"
- **SQL Server sunucusuna SQL Authentication ile bağlanmak**
- "Server= Server\_Name;Database=Db\_Name;User ID=Username;Password=sifreniz;Trusted\_Connection=False"
- **SQL Server sunucusuna Windows Authentication ile bağlanmak**
- "Data Source= Server\_Name;Initial Catalog=Db\_Name;Integrated Security=SSPI;"
- **SQL Server sunucusuna SQL Authentication ile bağlanmak**
- "Server=Server\_Name;Database=Db\_Name;Trusted\_Connection=True;"



## Bağlantı Yönetimi

- Bağlantı cümlesini oluşturduktan sonra, bağlantıyı açmak ve kapamak için **Connection** sınıfının iki önemli metodu kullanılır.
- **Open**
- **Close** (Bağlantı nesnesinin **Dispose** metodu da bağlantıyı kapatmak için kullanılabilir. )
- Örnekte Northwind.mdb isimli Access veritabanı üzerinde, Open ve Close metotlarının kullanımı gösterilmektedir.
- `cnNorthwind.ConnectionString = @"Provider=Microsoft.Jet. OLEDB.4.0;Data Source=C:\Samples\Northwind.mdb";`
- `cnNorthwind.Open();` //Bağlantıyı açmak, Veritabanı işlemleri bu arada gerçekleştirilir.
- `cnNorthwind.Close();` //Bağlantıyı kapatmak

## Bağlantı Yönetimi

- `System.Data.OleDb.OleDbConnection cnNorthwind;`
- **try** { `cnNorthwind = new System.Data.OleDb.OleDbConnection();`
- `cnNorthwind.ConnectionString = @"Provider=Microsoft.Jet. OLEDB.4.0;Data Source=C:\Samples\Northwind.mdb";`
- `cnNorthwind.Open(); // Veritabanı işlemleri gerçekleştirilir.`
- `}`
- **catch** (`InvalidOperationException XcpInvOp`) {
- `// İlk önce bu tipte hata yakalanır.`
- `MessageBox.Show("Önce veri tabanı bağlantısını kapatın");`
- `//Hata Mesajının içeriğini görmek için kullanılır.`
- `MessageBox.Show(XcpInvOp.ToString()); }`
- **catch** (`Exception Xcp`) { `//Diğer hatadan farklı bir tipte hata burda yakalanır.`
- `MessageBox.Show(Xcp.ToString()); }`
- **finally** { `cnNorthwind.Close(); //ya da cnNorthwind.Dispose(); }`

## Bağlantı Durumlarını Kontrol Etmek

- Bağlantı sınıfının durumu hakkında bilgi almak için, bağlantı sınıfının **State** özelliği kullanılır.
- | İsim              | Açıklama                                                       | Değeri |
|-------------------|----------------------------------------------------------------|--------|
| <b>Broken</b>     | Yalnızca, açık bir bağlantının kopup tekrar bağlanıldığı durum | 16     |
| <b>Closed</b>     | Bağlantı kapalı                                                | 0      |
| <b>Connecting</b> | Veri kaynağına bağlanma aşamasında                             | 2      |
| <b>Executing</b>  | Bağlantı üzerinden bir komutu çalıştırılıyor                   | 4      |
| <b>Fetching</b>   | Bağlantı üzerinden veri çekiliyor                              | 8      |
| <b>Open</b>       | Bağlantı açık                                                  | 1      |
- `private void BaglantiAc(OleDb.OleDbConnection con)`
  - `{ //Connection, sadece kapalı ise açılacak`
  - `If (con.State == ConnectionState.Closed)`
  - `{ con.Open(); }`
  - `}`

## Bağlantılı (Connected) Veritabanı İşlemleri

- Bağlantılı veri ortamları ile veritabanı üzerinde, gerekli tüm veritabanı işlemleri yapılabilir (Veritabanından tek değer çekme, Sadece okunabilir kayıt kümeleri döndürme, Kayıt ekleme, Kayıt silme, Kayıt güncelleme)
- **Command ile Çalışmak**
- Command, veritabanı üzerinde Stored Procedure (Saklı Yordam) ve Sorgu çalıştırmak için kullanılır. Command Nesneleri ile veritabanı tablolarında; sorgu, ekleme, silme ve güncelleme işlemleri yapılabilir. Command Nesnelerinin özellikleri aşağıda belirtilmiştir.
- **Name:** Command nesnesinin kod içerisindeki ismidir.
- **Connection:** Command nesnesinin hangi Connection üzerinde çalışacağını belirler.
- **CommandType:** Çalıştırılacak komutun türünü belirtir. Text, Stored Procedure ve TableDirect olmak üzere üç değeri vardır. TableDirect, SQL Server tarafından desteklenmez.
- **CommandText:** Stored Procedure adını veya Sorgu cümlesini tutar.
- **Parameters:** İsteğe bağlı parametrelerin kullanımı

## Bağlantılı (Connected) Veritabanı İşlemleri

- **Command ile Çalışmak**
- Command özelliklerine değer girildikten sonra, Command'ı çalıştırmak için
- Command sınıfı metotlarından uygun olan seçilir.
- **ExecuteScalar:** Çalıştırılan Command nesnesinden geriye tek değer döndürmek için kullanılır.
- **ExecuteReader:** Çalıştırılan Command nesnesinden geriye kayıt kümesi döndürmek için kullanılır.
- **ExecuteNonQuery:** Command Nesnesi üzerinde veri güncelleme değiştirme ve silme işlemleri yapmak için kullanılır. Bu işlemin sonucunda etkilenen kayıt sayısı geriye döndürür.
- **ExecuteXmlReader:** Çalıştırılan Command Nesnesinden geriye XML döndürmek için kullanılır. Sadece SQL Server 7.0 ve sonraki versiyonları için kullanılır.

## Bağlantılı (Connected) Veritabanı İşlemleri

- **Command ile Çalışmak**
- Command, kod içerisinden veya ToolBox üzerinden oluşturulabilir. Bu yöntemler ile kullanılan veritabanına göre, OleDbCommand veya SqlCommand nesneleri oluşturulur.
- //Access Veritabanına bağlanmak için Command tanımlanır.
- `System.Data.OleDb.OleDbCommand cmd =new System.Data.OleDb.OleDbCommand();`
- //Command Sınıfının CommandText özelliğine üniversiteler tablosu
- `cmd.CommandText="select * from üniversiteler";`
- 
- //Command Sınıfının Connection özelliğine aktif connection aktarılır
- `cmd.Connection=conn;`
- //Command Sınıfına, sorgu cümlesi yazılacağını belirler.
- `cmd.CommandType=CommandType.Text;`

## Bağlantılı (Connected) Veritabanı İşlemleri

- **Command ile Geriye Değer Döndürmek**
- OleDbCommand veya SqlCommand nesnesi ile geriye değer döndürmek için, **ExecuteScalar** metodu kullanılır.
- Örnekte OleDbCommand nesnesinin ExecuteScalar metodu ile Universiteler tablosundaki toplam kayıt sayısı geri döndürülmektedir.
- `System.Data.OleDb.OleDbConnection conn = new System.Data.OleDb.OleDbConnection(@"provider = Microsoft.JET.OLEDB.4.0; Data source=..\universiteler.mdb");`
- `System.Data.OleDb.OleDbCommand cmd = new System.Data.OleDb.OleDbCommand("select count(*) from" + "universiteler", conn);`
- `conn.Open();`
- `MessageBox.Show(cmd.ExecuteScalar.ToString());`

## Bağlantılı (Connected) Veritabanı İşlemleri

- **Command ile Geriye Değer Döndürmek**
- ExecuteScalar metodu ile geriye değer döndürmek için, sadece **Sum, Min, Max** veya **Count** gibi fonksiyonlar kullanılmaz. Aynı zamanda Select cümlesi veya Stored Procedure ile geriye tek değer döndürülebilir. Örnekte Ürün Tablosundaki Stok Miktarı SqlCommand nesnesi ile geriye döndürülmektedir.
- `string sql = "SELECT StokMiktari FROM Urun WHERE UrunID="+ "@UrunID";`
- `System.Data.SqlClient.SqlCommand cmUrun = new System.Data.SqlClient.SqlCommand(sql,cnAlisveris);`
- `System.Data.SqlClient.SqlParameter prmID = new System.Data.SqlClient.SqlParameter("@UrunID",System.Data.SqlDbType.Int, 4);`
- `cmUrun.Parameters["@UrunID"].Value = 42;`
- `cnAlisveris.Open();`
- `int adet = Convert.ToInt32(cmUrun.ExecuteScalar());`
- `cnAlisveris.Close(); MessageBox.Show("Quantity in stock: " + adet.ToString());`



## Bağlantılı (Connected) Veritabanı İşlemleri

- **Command ile Geriye Değer Döndürmek**
- OleDbCommand veya SqlCommand nesnesi ile geriye kayıt döndürmek için, **ExecuteReader** metodu kullanılır. ExecuteReader ile dönen kayıtlar DataReader nesnesine aktarılır.
- Örnekte Ürün Tablosundaki tüm ürünler, OleDbDataReader ile form üzerindeki
- ListBox kontrolüne eklenir.
- `System.Data.OleDb.OleDbCommand cmUrun = new System.Data.OleDb.OleDbCommand ("SELECT UrunAdi, StokMiktari" + "FROM Urun", cnAlisveris); cnAlisveris.Open();`
- `System.Data.OleDb.OleDbDataReader rdrUrun;`
- `rdrUrun = cmUrun.ExecuteReader(CommandBehavior.CloseConnection);`
- `while (rdrUrun.Read()) {`
- `listBox1.Items.Add(rdrUrun.GetString(0)+" - "+ rdrUrun.GetInt16(1)); }`
- `rdrUrun.Close();`

## Bağlantılı (Connected) Veritabanı İşlemleri

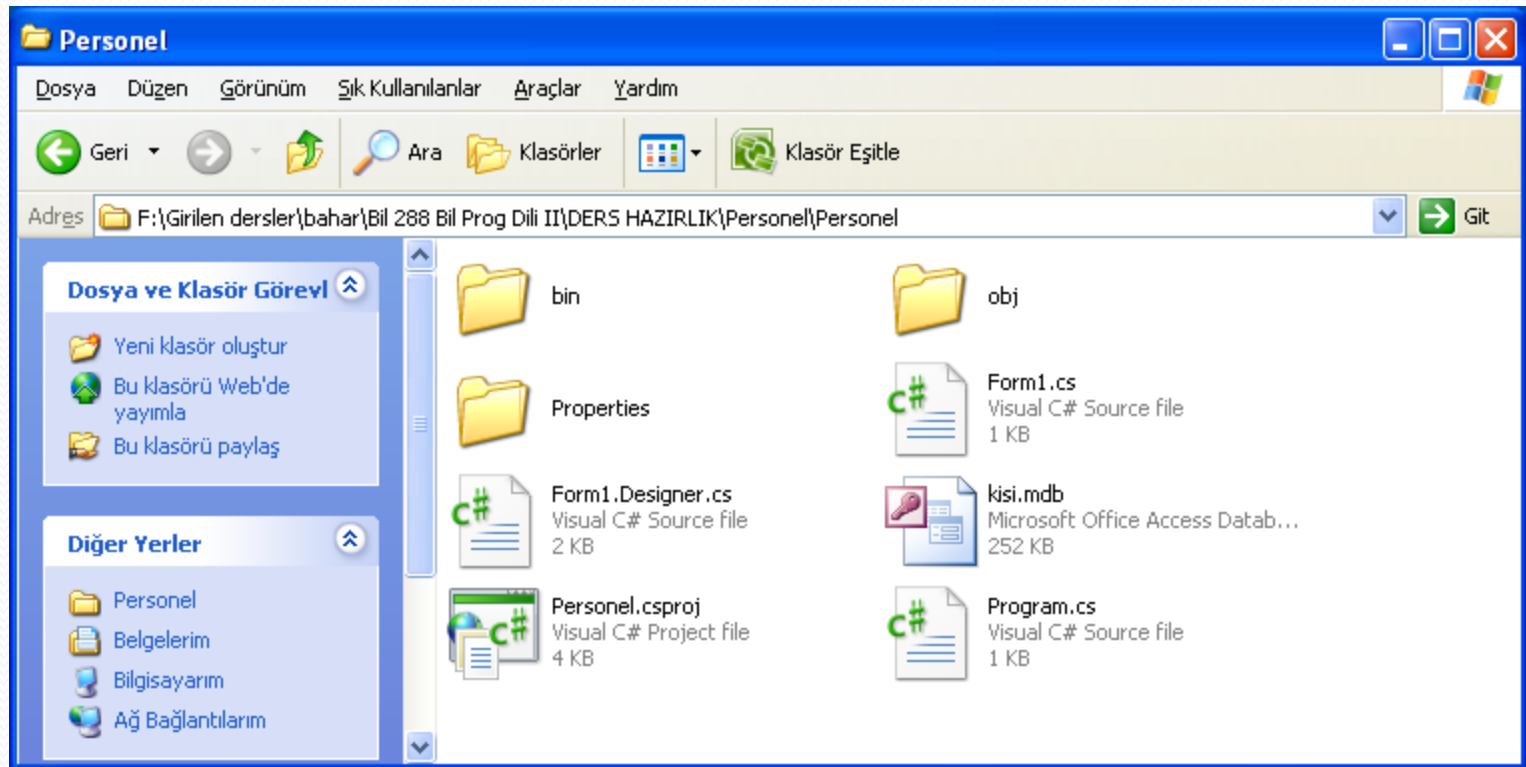
- **Command ile Kayıt Döndürmeyen Sorgular Çalıştırmak**
- Command ile veritabanı yapısında değişiklik yapılabilir (Tablo, View ve Stored Procedure oluşturmak, değiştirmek ve silmek), güvenlik seçenekleri ayarlanabilir (Tablo ve View izinleri) ve veritabanı içerisindeki veri değiştirilebilir (Kayıt ekleme, silme ve güncelleme). OleDbCommand veya SqlCommand nesnesi ile bu tür işlemlerin yapılabilmesi için, **ExecuteNonQuery** metodu kullanılır.
- ExecuteNonQuery metodu ile **INSERT, UPDATE ve DELETE** sorguları çalıştırılabilir.

# Örnek

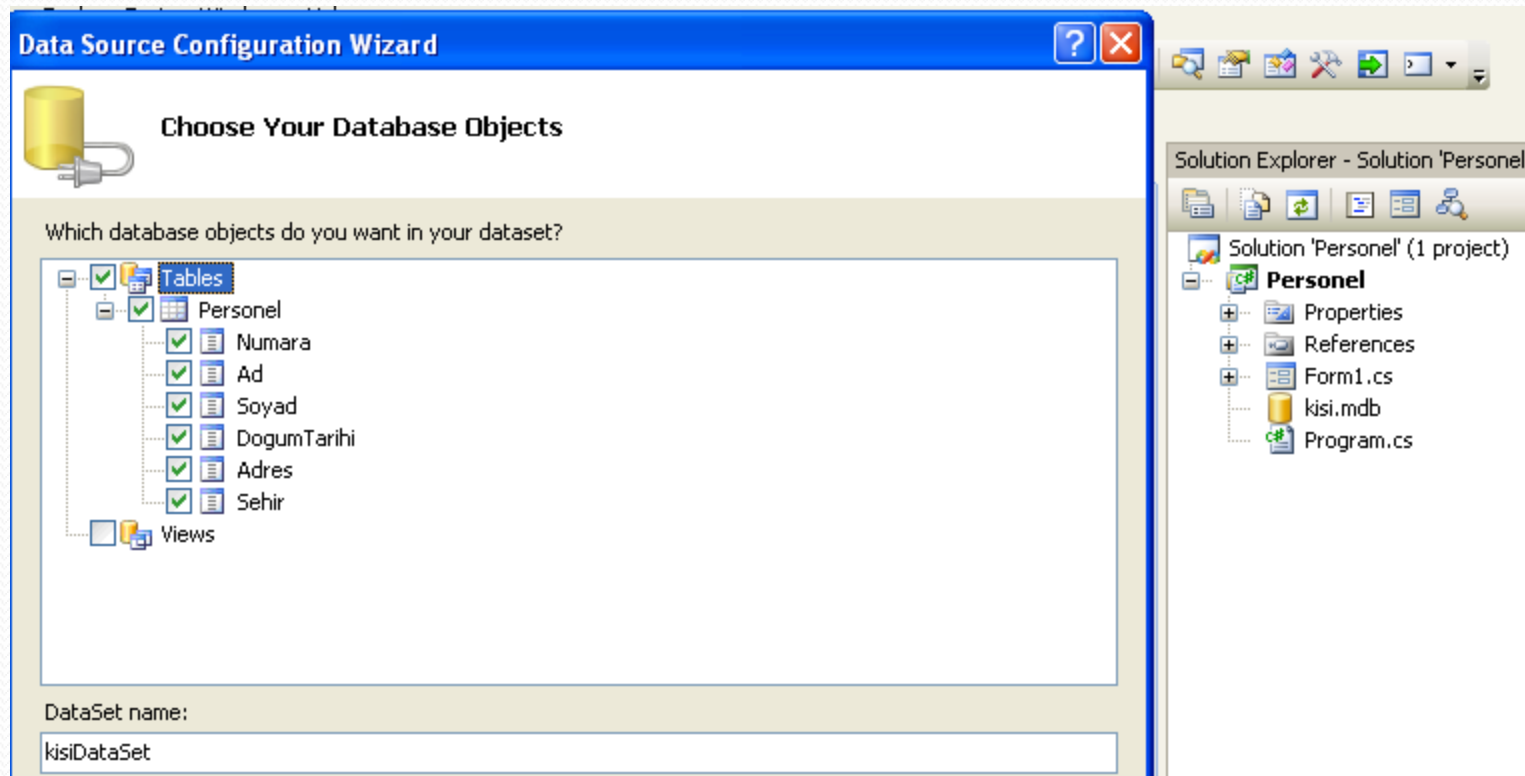
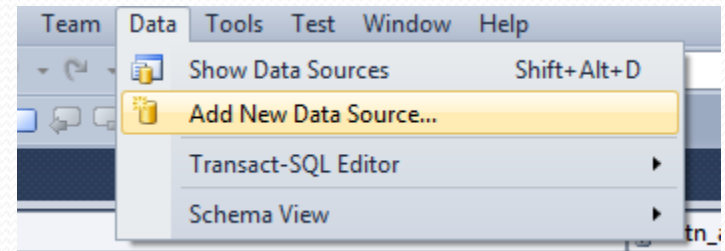
- **Veritabanının oluşturulması**
- Bu uygulamada kullanılacak Personel tablosu için bir veritabanı oluşturulması gerekir.
- 1. Microsoft Access ile “kisi.mdb” isminde bir veritabanı oluşturun.
- 2. Veritabanına Personel isminde bir tablo ekleyin ve tabloda belirtilen kolonları ekleyin. (Dikkat tablo ve alan adlarının tamamı büyük harf olmayacak bağlantıda hata verebilir.)

| Personel |             |               |
|----------|-------------|---------------|
|          | Alan Adı    | Veri Türü     |
| 🔑        | Numara      | Otomatik Sayı |
|          | Ad          | Metin         |
|          | Soyad       | Metin         |
|          | DogumTarihi | Tarih/Saat    |
|          | Adres       | Metin         |
|          | Sehir       | Metin         |

- **Kontrollerin eklenmesi**
- Personel isminde yeni bir Windows projesi açın.



- **Kontrollerin eklenmesi**



## Örnek

- **Kontrollerin eklenmesi**
- Personel isminde yeni bir Windows projesi açın. Aşağıdaki özellikleri girin.

| Kontrol – Kontrol İsmi | Özellik       | Değer        |
|------------------------|---------------|--------------|
| TextBox – txtAd        | BorderStyle   | FixedSingle  |
| TextBox – txtSoyad     | BorderStyle   | FixedSingle  |
| TextBox – txtDTarihi   | BorderStyle   | FixedSingle  |
| TextBox – txtSehir     | BorderStyle   | FixedSingle  |
| TextBox – txtAdres     | BorderStyle   | FixedSingle  |
|                        | Multiline     | True         |
|                        | ScrollBars    | Vertical     |
| ComboBox – cbNo        | DropDownStyle | DropDownList |
| Button – btnYeni       | Text          | Yeni         |
| Button – btnİptal      | Text          | İptal        |
| Button – btnKaydet     | Text          | Kaydet       |
| Button – btnSil        | Text          | Sil          |

# Örnek

- Kontrollerin eklenmesi

The screenshot shows a Windows application window titled "Form1". The form contains the following controls:

- No'ya Göre Ara:** A text box with a dropdown arrow.
- Ad:** A text box.
- Soyad:** A text box.
- Doğum Tarihi:** A text box.
- Adres:** A large text area with a vertical scrollbar.
- Şehir:** A text box.
- Buttons:** Four buttons at the bottom: "Yeni" (New), "İptal" (Cancel), "Kaydet" (Save), and "Sil" (Delete).

## Örnek

- **Kodların yazılması**
- Personel tablosu üzerinde işlem yapılması için veritabanına bağlantı açılması gerekir. Bu bağlantı için gereken Connection String ifadesinin merkezi bir yerden alınması, değişiklik durumunda kolaylık sağlayacaktır. Veritabanı işlemleri son olarak debug klasöründe oluşturulacaktır.

```
-using System.Data.OleDb;
|
} namespace Personel
{
} public partial class Form1 : Form
 {
 OleDbConnection conn= new OleDbConnection ("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=kisi.mdb");
```

- using System.Data.OleDb;
- namespace Personel
- { public partial class Form1 : Form
- { OleDbConnection conn= new OleDbConnection ("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=kisi.mdb");



## Örnek

- `public void Kaydet()`
- `{`
- `OleDbCommand kaydet = new OleDbCommand("INSERT INTO`  
`Personel(Ad,Soyad,DogumTarihi,Adres,Sehir) VALUES ("`
- `+ txtAd.Text + "," + txtSoyad.Text + "," + txtDTarihi.Text + "," + txtAdres.Text + ","`  
`+ txtSehir.Text + ")")", conn);`
- `try {`
- `conn.Open();`
- `kaydet.ExecuteNonQuery(); }`
- `catch (Exception ex) { MessageBox.Show(ex.Message); }`
- `finally`
- `{ conn.Close(); }`
- `}`

## Örnek

- `public void Sil(int ID) {`
- `OleDbCommand sil = new OleDbCommand("DELETE FROM Personel WHERE Numara=" + ID + "", conn);`
- `try`
- `{`
- `conn.Open();`
- `sil.ExecuteNonQuery();`
- `}`
- `catch(Exception ex) { MessageBox.Show(ex.Message); }`
- `finally { conn.Close(); }`
- `}`
- `public void Temizle()`
- `{ txtAd.Text = ""; txtSoyad.Text = "";`
- `txtAdres.Text = ""; txtSehir.Text = ""; txtDTarihi.Text = "";`
- `txtAd.Focus();`
- `}`

## Örnek

- `public void IDDoldur() {`
- `cbNo.Items.Clear();`
- `OleDbCommand veri = new OleDbCommand("SELECT Numara FROM Personel ORDER BY Numara", conn);`
- `OleDbDataReader oku = null;`
- `try {`
- `conn.Open();`
- `oku=veri.ExecuteReader();`
- `while(oku.Read())`
- `{ cbNo.Items.Add(oku.GetInt32(0));        }`
- `}`
- `catch (Exception ex)        {        MessageBox.Show(ex.Message);        }`
- `finally        {        oku.Close();        conn.Close();        }`
- `}`

## Örnek

- `public void IDyeGoreFormDoldur(int ID) {`
- `OleDbCommand veri = new OleDbCommand("SELECT Numara, Ad, Soyad, DogumTarihi, Adres, Sehir FROM Personel where Numara="+ID+"", conn);`
- `OleDbDataReader oku=null;`
- `try {`
- `conn.Open();      oku=veri.ExecuteReader();`
- `if (oku.Read()) {`
- `txtAd.Text = oku["Ad"].ToString();`
- `txtSoyad.Text = oku["Soyad"].ToString();`
- `txtAdres.Text = oku["Adres"].ToString();`
- `txtSehir.Text = oku["Sehir"].ToString();`
- `txtDTarihi.Text = oku["DogumTarihi"].ToString();      }`
- `}`
- `catch (Exception ex)      {      MessageBox.Show(ex.Message);      }`
- `finally      {      oku.Close();      conn.Close();      }`
- `}`

## Örnek

- `public bool Kontrol() {`
- `if (txtAd.Text == "")`
- `{ MessageBox.Show("Adı Giriniz");`
- `txtAd.Focus(); return false;`
- `}`
- `else if (txtSoyad.Text == "")`
- `{ MessageBox.Show("Soyadı Giriniz");`
- `txtSoyad.Focus();`
- `return false;`
- `}`
- `else if (txtDTarihi.Text == "")`
- `{`
- `MessageBox.Show("Doğum Tarihini Giriniz");`
- `txtDTarihi.Focus();`
- `return false;`
- `}`
- `else if (txtAdres.Text == "")`
- `{`
- `MessageBox.Show("Adresi Giriniz");`
- `txtAdres.Focus();`
- `return false;`
- `}`
- `else if (txtSehir.Text == "")`
- `{`
- `MessageBox.Show("Şehiri Giriniz");`
- `txtSehir.Focus();`
- `return false;`
- `}`
- `else { return true; }`
- `}`

## Örnek

- private void btnKaydet\_Click(object sender, EventArgs e)
- { if(Kontrol() == true)
- {
- Kaydet();
- btnYeni.Enabled = true;
- btnKaydet.Enabled = false;
- btnIptal.Enabled = false;
- IDDoldur();
- cbNo.SelectedIndex =
- cbNo.Items.Count - 1;
- }
- }

- private void btnYeni\_Click(object sender, EventArgs e) {
- Temizle();
- btnYeni.Enabled = false;
- btnKaydet.Enabled = true;
- btnIptal.Enabled = true;
- cbNo.SelectedIndex = -1; }
- private void btnIptal\_Click(object sender, EventArgs e) {
- Temizle();
- btnYeni.Enabled = true;
- btnKaydet.Enabled = false;
- btnIptal.Enabled = false;
- cbNo.SelectedIndex = 0; }

## Örnek

- `private void btnSil_Click(object sender, EventArgs e)`
- `{`
- `if(MessageBox.Show(cbNo.SelectedItem + " nolu kaydı silmek istiyor musunuz?",`  
`this.Text, MessageBoxButtons.YesNo, MessageBoxIcon.Question,`  
`MessageBoxDefaultButton.Button2) == DialogResult.Yes)`
- `{`
- `Sil(Convert.ToInt32(cbNo.SelectedItem));`
- `IDDoldur();`
- `cbNo.SelectedIndex = cbNo.Items.Count - 1;`
- `}`
- `}`
- `private void Form1_Load(object sender, EventArgs e)`
- `{      IDDoldur();          cbNo.SelectedIndex = 0;      }`
- `private void cbNo_SelectedIndexChanged(object sender, EventArgs e)`
- `{      IDyeGoreFormDoldur(Convert.ToInt32(cbNo.SelectedItem));      }`

## Bağlantısız (Disconnected) Veritabanı İşlemleri

- Bağlantısız veri ortamları, uygulamaların veritabanından bağımsız çalıştığı ortamlardır.
- Veritabanı sunucusunun uzak olması, veri işlemlerinin uzun sürmesi ve mobil çalışma ihtiyacı, bağlantısız veri ortamlarına olan ihtiyacı artırmıştır.
  - **DataAdapter** nesnesi, **DataSet** nesne modeli, **DataTable** nesne modeli
- **DataAdapter nesnesi:** Connection, Command ve DataReader sınıflarını kullanarak, verilerin DataSet'e doldurulmasını ve DataSet de yapılan değişikliklerin veri tabanına kaydedilmesini sağlar.



## Bağılantısız (Disconnected) Veritabanı İşlemleri

- **DataSet ve DataTable Oluşturmak**
- Veri kaynağından DataAdapter ile çekilen verilerin, çekirdek belleğe atılan kopyası DataSet içerisinde saklanır. DataSet ile bu veriler üzerinde gerekli düzenlemeler yapıldıktan sonra, veriler aynı DataAdapter ile veritabanına aktarılır.
- **DataSet**, Sanal bir veritabanı yapısını temsil eder. DataTable nesnelerinden oluşur. Bu tablolar arasında ilişkiler tanımlanabilir. DataSet'i oluşturan nesneler: **DataTable**, **DataColumn**, **DataRow**, **DataRelation** nesneleridir.

## Bağılantısız (Disconnected) Veritabanı İşlemleri

- **DataTable** : Veritabanı tablolarını temsil eder. DataColumn, DataRow nesnelerinden oluşur. Primary Key alanı tanımlanabilir.
- **DataColumn**: DataTable nesnelerini oluşturmak için gereken kolonları temsil eder.
- **DataRow** : DataTable nesneleri için veri satırlarını temsil eder.
- **DataRelationship** : Tablolar arasındaki ilişkileri temsil eder.
- **DataView** : DataTable nesneleri üzerinde filtreleme, veri güncellemeleri işlemleri yapmak için kullanılır.

## Bağlantısız (Disconnected) Veritabanı İşlemleri

- Örnekte ds ismindeki DataSet nesnesinin tüm tablo, ilişki ve verileri ds\_yeni ismindeki DataSet nesnesinin içerisine aktarılmıştır.
  - `DataSet ds = new DataSet("Yeni DataSet");`
  - `DataSet ds_yeni;`
  - `ds_yeni= ds.Copy(); //veya dsCopy = ds.Clone();`
- DataSet sınıfının Tables koleksiyonu ile DataSet içerisine bir veya birden çok DataTable eklenebilir. Örnekte dtKitaplar isminde yeni bir DataTable oluşturulmaktadır.
  - `DataTable dtKitaplar = new DataTable("Kitaplar");`
- Oluşturulan tabloyu DataSet içerisine eklemek için DataSet nesnesinin Tables koleksiyonu kullanılır.
- `Ds.Tables.Add(dtKitaplar);`

## Bağlantısız (Disconnected) Veritabanı İşlemleri

- DataTable nesnesinin içerisine kolon eklenebilir. Örnekte dtKitaplar isimindeki DataTable nesnesinin içerisine, yenild isminde yeni bir kolon eklenmektedir. Yeni kolon eklemek için, DataTable nesnesinin Columns koleksiyonu kullanılır.
  - DataColumn colKitapId = dtKitaplar.Columns.Add("yenild");
- Örnekte DataTable nesnesi için Ucret, KDV ve Tutar isiminde 3 adet kolon oluşturulmuştur. Örnekteki KDV kolonu, Ucret kolonun %17 değeri üzerinden hesaplanır. Tutar kolonu ise Ucret ve KDV değerinin toplamı ile hesaplanır.
  - DataColumn colUcret = new DataColumn("Ucret");
  - DataColumn colKdv = new DataColumn("KDV");
  - colKdv.Expression = "Ucret \* 0.17";
  - DataColumn colTutar = new DataColumn("Tutar");
  - colTutar.Expression = "Ucret + KDV";

## Bağılantısız (Disconnected) Veritabanı İşlemleri

- **DataAdapter ile kayıtları DataSet'e doldurmak**
- Örnekte OleDbDataAdapter ile çekilen veriler, ds ismindeki DataSet nesnesine aktarılır. DataSet içerisindeki veriler, DataGridView ile ekranda gösterilir.
  - OleDbConnection conn = new OleDbConnection ("provider = " + microsoft.jet.oledb.4.0; data source=C:\Stok.mdb");
  - OleDbDataAdapter da = new OleDbDataAdapter("select \* from kitaplar", conn);
  - DataSet ds = new DataSet();
  - **da.Fill(ds,"Kitaplar");**
  - DataGridView1.DataSource= ds.Tables["Kitaplar"];
- Fill metodu ile belirli kayıt aralığı DataSet içerisine aktarılabilir. Örnekte da isimli DataAdapter ile çekilen ilk altı kayıt, Kitaplar tablosuna aktarılır.
  - **da.Fill(ds, 0, 5, "kitaplar");**

## Bağılantısız (Disconnected) Veritabanı İşlemleri

- DataSet üzerinde yapılan değişiklikleri veri kaynağına aktarmak için, DataAdapter sınıfının Update metodu kullanılır. DataAdapter nesnesinin DeleteCommand, UpdateCommand ve InsertCommand nesneleri içinde tutulan sorgular ile güncelleme işlemi gerçekleştirilir. Örnekte Sipariş tablosundaki tüm değişiklikler veri kaynağına aktarılmaktadır.
  - **da.Update(ds, "siparisler");**
- Örnekte Dataset içerisindeki kitap\_baslik kolonun değeri, TextBox kontrolünün **Text** özelliğine aktarılır.
  - **TextBox1.Text = ds.Tables["kitaplar"].Rows[2].Item["kitap\_baslik"];**
- Örnekte ComboBox ve ListBox kontrolünün **DataSource** ve **DisplayMember** özellikleri kullanılmaktadır.
  - **ComboBox1.DataSource = ds.Tables["kitaplar"];**
  - **ComboBox1.DisplayMember = ds.Tables["kitaplar"].Columns["kitap\_baslik"].ToString();**
  - **ListBox1.DataSource = ds.Tables["kitaplar"];**
  - **ListBox1.DisplayMember = ds.Tables["kitaplar"].Columns["kitap\_baslik"].ToString();**

## Bağlantısız (Disconnected) Veritabanı İşlemleri

- Örnekte TreeView kontrolüne veri bağlamak için, TreeNode nesnesinin **Text** özelliği kullanılır.
  - **TreeView1.Nodes[0].Text = ds.Tables["kitaplar"].Rows[1].Item["kitap\_baslik"];**
- Örnekte DataSet nesnesinden gelen veriler ListView ve CheckedListBox kontrollerine aktarılmıştır.
  - **int count ;**
  - **Count = ds.Tables["kitaplar"].Columns.Count();**
  - **for (int i=0;i< count;i++)**
  - **{ ListView1.Items.Add(ds.Tables["kitaplar"].Rows[i][0].ToString()); }**
  - **for (int i=0;i<count;i++) {**
  - **CheckedListBox1.Items.Add(ds.Tables["kitaplar"].Rows[i][0].ToString()); }**
-

## Bağılantısız (Disconnected) Veritabanı İşlemleri

- **DataRow** ile DataTable içerisindeki kayıtlar değiştirilebilir. DataRow nesnesi ile satır düzenleme işlemleri için aşağıdaki metodlar kullanılır.
- **BeginEdit**, veriyi düzenlerken oluşabilecek olayları askıya alır. Veriyi düzenlemek için Items koleksiyonu kullanılır.
- **EndEdit** metodu ile, askıya alınan olaylar yeniden aktif edilir.
- **CancelEdit** metodu ile değişikliklerden ve askıya alınan olaylardan vazgeçilir.



## Bağılantısız (Disconnected) Veritabanı İşlemleri

- Örnekte DataTable içerisindeki dördüncü kayıt için güncelleme işlemi yapılmıştır.
  - **DataRow drNew = dtKitaplar.Rows[3];**
  - **drNew.BeginEdit();**
  - **drNew["kitap\_baslik"] = "yeni hayat"; drNew["kitap\_yazar"] = "can düNDAR";**
  - **drNew.EndEdit();**
- DataRow ile DataTable içerisindeki belirli bir satır silinebilir. Örnekte DataTable içerisindeki dördüncü kayıt silinmiştir.
  - **DataRow drSil = dtKitaplar.Rows[3];**
  - **dtKitaplar.Rows.Remove(drSil);**
  - DataRow nesnesinin Delete metodu kullanılarak aktif kayıt silinebilir.
  - **drSil.Delete();**

# Bağılantısız (Disconnected) Veritabanı İşlemleri

- **Windows Form ile Kayıt Üzerinde Hareket Sağlamak**
- Verileri düzenlemeden önce, hangi veri üzerinde düzenleme yapılacağının tespit edilmesi gerekir. DataSet, DataTable veya DataView ile kayıtlar üzerinde hareket sağlayan nesneye **CurrencyManager** denir.
- DataSet içinde çoklu veri kaynağı tutulabildiği için, birden fazla CurrencyManager nesnesi içerebilir.
- Belirli bir satıra gidebilmek için, CurrencyManager nesnesinin **Position** özelliği kullanılır.

## Bağlantısız (Disconnected) Veritabanı İşlemleri

- Örnekte dtKitaplar tablosunun kayıtları arasında ilk, son, önceki ve sonraki
- satıra hareket sağlanmıştır.
  - **CurrencyManager cmKitaplar = new CurrencyManager();**
  - **private void Form1\_Load() {**
  - **txtKitapAdi.DataBindings.Add("Text", dtKitaplar, "kitap\_baslik");**
  - **cmKitaplar = (CurrencyManager)BindingContext[dtKitaplar];**
  - **cmKitaplar.Position = 0; }**
  - **private void btnMoveNext()**
  - **{If (cmKitaplar.Position != cmKitaplar.Count) { cmKitaplar.Position+= 1;}}**
  - **private void btnMoveFirst() { cmKitaplar.Position = 0; }**
  - **private void btnMovePrevious() {**
  - **If (cmKitaplar.Position != 0) { cmKitaplar.Position -= 1; } }**
  - **private void btnMoveLast() { cmKitaplar.Position = cmKitaplar.Count-1; }**