

NESNE TABANLI PROGRAMLAMA II

Öğr. Gör. Hasan SANCAK

@BizimDigital

İÇİNDEKİLER

PictureBox Kontrolü	3
Çoklu Form Uygulamaları	8
RichTextBox Kontrolü	14
Menü Tasarımı	22
Dosyalama	35
FileStream Sınıfı	41
Ado.NET	59
Veri Ortamları	64
Access Veri Tabanı Uygulama	73
SQL Server Veri Tabanı Uygulama	79
Raporlama	87
Referanslar	97
Geçmiş Sorular	114
Kaynaklar	118

PictureBox Kontrolü

Resim görüntülemeyi sağlar. Form üzerinde bir resim görüntülemek için kullanılır.

PictureBox Özellikleri

Özellik Değer Tipi Açıklama

Image Image Kontrolün resim kaynağını belirler

SizeMode PictureBoxSizeMode Kontrolün, resmi nasıl görüntüleyeceğini belirler.

AutoSize değeri, kontrolün büyüklüğünü resmin büyüklüğüne göre ayarlar.

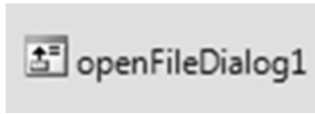
CenterImage değeri, resmi kontrolün ortasına gelecek şekilde ayarlar.

Normal değeri, kontrolün sol üst köşesine göre konumlandırır.

StretchImage değeri, resmi kontrolün büyüklüğüne göre boyutlandırır ve resmin tam görünmesini sağlar.

OpenFileDialog Kontrolü

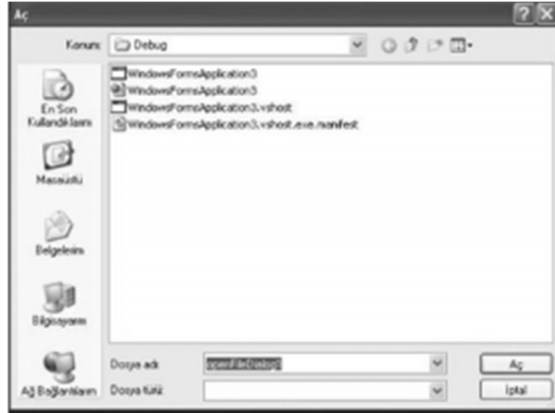
Bu kontrol sayesinde kullanıcılar windowsun kullandığı ortak iletişim formlarını kullanarak dosya yükleme işlemlerini gerçekleştirebilirler.



openFileDialog.ShowDialog()

openFileDialog penceresinin açılmasını sağlayan koddur. Bir butonun click olayına yazıldığında dosya windowsun bilindik dosya açma ekranı karşımıza çıkar.

```
private void button1_Click(object sender, EventArgs e)
{
    openFileDialog1.ShowDialog();
}
```



openFileDialog1.Title()

openFileDialog penceresinin başlığını belirler. openFileDialog1.ShowDialog() komutundan önce kullanılmalıdır. Çünkü pencere açıldıktan sonra Title' a değer atanırsa formun başlığı değişmeyecektir.

```
private void button1_Click(object sender, EventArgs e)
{
    openFileDialog1.Title = "Lütfen Dosya Seçiniz";
    openFileDialog1.ShowDialog();
}
```

openFileDialog1.Filter()

openFileDialog penceresinde sadece belirli uzantılı dosyaların listelenmesini istenirse bu kod kullanılmalıdır.

```
private void button1_Click(object sender, EventArgs e)
{
    openFileDialog1.Title = "Lütfen Dosya Seçiniz";
    openFileDialog1.Filter = " (*.jpg)|*.jpg| (*.png)|*.png";
    openFileDialog1.ShowDialog();
}
```

Burada sadece jpg ve png uzantılı dosyaların listelenmesi istenmiştir.

openFileDialog1.FilterIndex()

Filtreleme yaparken varsayılan olarak hangisinin kullanılacağını belirler. Mesela openFileDialog penceresi ilk açıldığında varsayılan olarak belirlenen değer jpg ise önce jpg resimler çıkar.

```
private void button1_Click(object sender, EventArgs e)
{
    openFileDialog1.Title = "Lütfen Dosya Seçiniz";
    openFileDialog1.Filter = " (*.jpg)|*.jpg| (*.png)|*.png";
    openFileDialog1.FilterIndex = 1;
    // varsayılan olarak jpg uzantılarını göster
    openFileDialog1.ShowDialog();
}
```

openFileDialog1.InitialDirectory

openFileDialog penceresinin varsayılan olarak açılması istenilen klasörün yolunu belirler.

```
private void button1_Click(object sender, EventArgs e)
{
    openFileDialog1.Title = "Lütfen Dosya Seçiniz";
    openFileDialog1.Filter = " (*.jpg)|*.jpg| (*.png)|*.png";
    openFileDialog1.FilterIndex = 1;
```

```
        openFileDialog1.InitialDirectory="C:Documents and
        Settings\\Desktop\\blog";
        openFileDialog1.ShowDialog();
    }
```

openFileDialog1.Multiselect

True değeri aktarılsa openFileDialog penceresinde shift tuşuna basarak birden fazla dosya seçilebilir.

```
private void button1_Click(object sender, EventArgs e)
{
    openFileDialog1.Title = "Lütfen Dosya Seçiniz";
    openFileDialog1.Filter = " (*.jpg)|*.jpg| (*.png)|*.png";
    openFileDialog1.FilterIndex = 1;
    openFileDialog1.InitialDirectory="C:\\Documents and
    Settings\\Desktop\\blog";
    openFileDialog1.Multiselect = true;
    openFileDialog1.ShowDialog();
}
```

openFileDialog1.FileNames

Seçilen dosyanın yolunu belirler. Mesela seçilen dosyanın yolunu bir textbox a yazdıralım.

```
private void button1_Click(object sender, EventArgs e)
{
    openFileDialog1.Title = "Lütfen Dosya Seçiniz";
    openFileDialog1.Filter = " (*.jpg)|*.jpg| (*.png)|*.png";
    openFileDialog1.FilterIndex = 1;
    openFileDialog1.InitialDirectory="C:\\Documents and
    Settings\\Desktop\\blog";
    openFileDialog1.Multiselect = true;
    openFileDialog1.ShowDialog();
}
```

```

string str = openFileDialog1.FileName;
textBox1.Text = str;
}

```

Uygulama: Dosya aç butonuna basılınca dosya açma penceresinden seçilen resim pictureBox kontrolüne yüklenecek dosya yolu görüntülenecek.



```

private void button1_Click(object sender, EventArgs e)
{
    openFileDialog1.ShowDialog();
    string dosya;
    dosya=openFileDialog1.FileName;
    textBox1.Text = dosya;
    pictureBox1.Image = Image.FromFile(textBox1.Text);
    pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;
}

```

Aynı uygulamaya bir combobox kontrolü ekleyip bu kontrolden seçilen resmin pictureBox kontrolüne nasıl yerleşeceğini seçip resmin belirtilen şekilde görüntülenmesini sağlamak için



formu ve kodu aşağıdaki şekilde değiştirebiliriz.

```
private void button1_Click(object sender, EventArgs e)
{
    openFileDialog1.ShowDialog();
    string dosya;
    dosya=openFileDialog1.FileName;
    textBox1.Text = dosya;
    pictureBox1.Image = Image.FromFile(textBox1.Text);
    pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;
}
private void cmbResim_SelectedIndexChanged(object sender,
EventArgs e)
{
    switch (cmbResim.SelectedIndex)
    {
        case 0: pictureBox1.SizeMode = PictureBoxSizeMode.AutoSize;
        break;
        case 1: pictureBox1.SizeMode = PictureBoxSizeMode.CenterImage;
        break;
        case 2: pictureBox1.SizeMode = PictureBoxSizeMode.Normal;
        break;
        case 3: pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;
        break;
    }
}
```

Çoklu Form Uygulamaları

Windows uygulamaları, kullanıcı ile iletişimi Form nesneleri ile sağlar. Formlar, görünüm özellikleri, pencere stili değiştirilerek ve üzerine kontroller eklenerek özelleştirilir. Ayrıca birden çok form nesnesi kullanılarak, uygulamalar zenginleştirilir.

Birden Fazla Form Oluşturmak

Windows uygulamaları birden fazla form nesnesinden oluştuğu için, projelere form eklemek her zaman gereklidir. Bir Windows projesine yeni bir form eklemek için:

1. Solution Explorer panelin den projeye sağ tıklayarak ya da Project menüsünden Add Windows Form komutu seçilir.
2. Çıkan menüden Windows Form öğesinin seçili olduğu kontrol edilir ve bir isim verilerek form eklenir.

Başlangıç formlarının ayarlanmasının yanı sıra, uygulamada bir formdan başka bir formun açılması ve ayarlanması sık karşılaşılan bir durumdur. Form nesneleri, System.Windows.Forms namespace içinde bulunan Form sınıfından türemiş sınıflardır. Dolayısıyla yeni bir Form oluşturmak için, istenen Form sınıfından bir nesne oluşturulması yeterlidir.

```
Form1 yeniForm = new Form1( );
```

Yeni oluşturulan formların gösterilmesi, formun Show ve ShowDialog metotları ile yapılır. ShowDialog metodu, form gösterildikten sonra, kapanana kadar diğer formlara erişimi engeller. ShowDialog metodundan sonra yazılan kodlar, form kapandıktan sonra çalıştırılır.

```
Form1 yeniForm = new Form1( );  
yeniForm.ShowDialog( );
```

ShowDialog ile gösterilen formlar, hangi durum ile kapandıklarını belirten bir DialogResult sonucu döndürürler. Bu kullanım MessageBox.Show hazır fonksiyonu ile aynıdır.

```
Form1 frm = new Form1( );  
if (frm.ShowDialog == DialogResult.Yes)  
{  
    // Verileri kaydet }  
}
```

Formun hangi diyalog sonucu ile döneceğini, üzerindeki Button kontrollerinin DialogResult özelliği ile belirlenir. Eğer düğmenin bu özelliği Yes olarak ayarlanmışsa, Form bu düğmeye basılıp kapandığı zaman, DialogResult.Yes değerini döndürür.

Main yordamı bütün uygulamaların giriş noktasıdır. Windows uygulamalarında formlar yüklenmeden önce o form içinde tanımlı Main yordamı çalıştırılır. Bu Main yordamında Application sınıfı başlangıç formunu Run metodu ile yükler. Application sınıfı, .NET Framework çatısında, uygulamaları başlatmak, yönetmek ve sonlandırmak için kullanılır.

Projenin özelliklerinden başlangıç nesnesi Sub Main olarak ayarlanırsa, uygulama çalıştığı zaman tüm projede Main yordamı arar. Windows uygulamaları geliştirirken Main yordamı yazılırsa başlangıç formunun da bu yordam içinde belirtilmesi gerekir. Bu yordam bir modülün içinde tanımlanabilir.

```
public static void Main()  
{  
    Application.Run( new Form1( ) );  
}
```

Application sınıfının Run metodu, parametre olarak başlangıç formu ister. Uygulama başladığı zaman hangi formun çalışması isteniyorsa, bu formdan oluşturulup parametre olarak verilir. New anahtar kelimesi, sınıfları oluşturmak için kullanılır.

Herhangi bir uygulamayı sonlandırmak için ise Application sınıfının Exit metodu kullanılabilir.

```
Application.Exit( ) ;
```

Form Özellikleri:

<u>Özellik</u>	<u>Değer Tipi</u>	<u>Açıklama</u>
AcceptButton	Button	Form üzerinde Enter tuşuna basıldığı zaman “tıklanacak” Button kontrolü
CancelButton	Button	Form üzerinde Esc tuşuna basıldığı zaman “tıklanacak” Button kontrolü
Opacity	Double	Formun şeffaflık oranı (0 -1 arası)
MaximizeBox	Boolean	Ekranı Kapla düğmesinin görünürlüğü
MaximizeBox	Boolean	Simge Durumunda Küçült düğmesinin görünürlüğü
ControlBox	Boolean	Close, Maximize ve Minimize düğmelerinin tümünün görünürlüğü
StartPosition	FormStartPosition	Form açıldığı zaman, ekran üzerindeki konumu
TopMost	Boolean	Formun tüm pencerelerin üzerinde görünmesi
FormBorderStyle	FormBorderStyle	Formun kenar stili
MaximumSize	Size	Formun alabileceği maksimum büyüklük
MinimumSize	Size	Formun alabileceği minimum büyüklük

Form Olayları:

<u>Olay</u>	<u>Açıklama</u>
Click	Form üzerine tıklandığı zaman gerçekleşir
Closing	Form kapanmadan hemen önce gerçekleşir

Closed	Form kapandıktan sonra gerçekleşir
Load	Form yüklenirken gerçekleşir
KeyDown	Form üzerindeyken bir tuşun basılması ile gerçekleşir
KeyUp	Basılan tuşun bırakılması ile gerçekleşir

Form Metotları:

Metot Açıklama

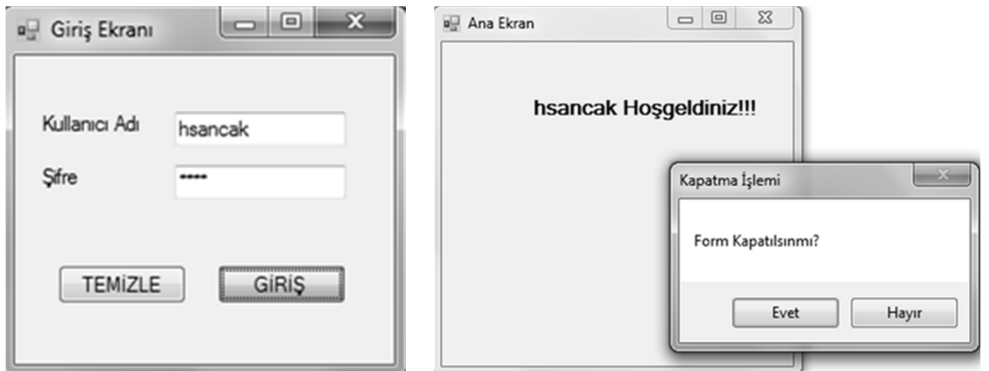
Hide	Formu Visible özelliğini False yaparak, gizler
Close	Formu kapatır. Eğer form başlangıç formuysa uygulama sonlanır
Show	Formu gösterir . Hide ile gizlenmişse, Visible özelliği True yapılır.
ShowDialog	Formu diyalog kutusu olarak gösterir.

Ayrıca aktif olan forma ait metotları kullanmak için **this** metodu kullanılabilir.

Örneğin aktif olan bir formu kapatmak için

this.Close();

Uygulama:Giriş ekranından doğru kullanıcı adı ve şifre yazılınca Ana ekranı açan ve giriş formundaki kullanıcı adı verisini aktaran program.



Giriş Formu Kodları

```
// Temizle Butonu
private void button1_Click(object sender, EventArgs e)
{
    textBox1.Clear();
    textBox2.Clear();
}

public static string veri; //Veriyi aktaracak değişken

//Giriş Butonu
private void button2_Click(object sender, EventArgs e)
{
    if (textBox1.Text == "hsancak" && textBox2.Text == "1234")
    {
        Form2 anafrm = new Form2();
        veri = textBox1.Text;
        anafrm.Show();
        this.Visible = false;
    }
    else
    {
        MessageBox.Show("İsim veya Parola Yanlış");
        textBox1.Clear();
        textBox2.Clear();
    }
}
```

Ana Form Kodları

```
private void Form2_Load(object sender, EventArgs e)
{
    label1.Text = Form1.veri + " Hoşgeldiniz!!!";
}
```

```

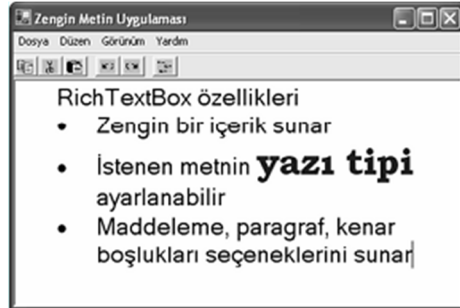
// ALT + F5 Tuşlarına basılınca form kapanma işlemi başlar
private void Form2_KeyDown(object sender, KeyEventArgs e)
{
    if (e.Alt == true && e.KeyCode == Keys.F5)
        this.Close();
}

// Formun Kapanması olayı
private void Form2_FormClosing(object sender,
FormClosingEventArgs e)
{
    DialogResult s;
    s = MessageBox.Show("Form Kapatılınsınımı?", "Kapatma
İşlemi", MessageBoxButtons.YesNo);
    if (s == DialogResult.Yes)
    {
        e.Cancel = false;
        Form1 grsfrm = new Form1();
        grsfrm.Visible = true;
    }
    else
        e.Cancel = true;
}

```

RichTextBox Kontrolü

TextBox kontrolünden daha gelişmiş özelliklere sahiptir. Seçilen yazının rengi, yazı tipi değiştirilebilir. Madde işaretleri kullanılabilir. Satır başlarındaki boşluklar ayarlanabilir.



Normal bir metin kutusundan daha gelişmiş özelliklere sahip bir kontroldür.

TextBox kontrolünde yazının yazı tipi, büyüklüğü gibi ayarlar yapılabilir. Ancak sadece seçilen yazının rengi, yazı tipi, satır başı genişliği, madde işaretleri kullanımı gibi ayarlar yapmak mümkün değildir. RichTextBox kontrolü, bu tip zengin özelliklerin kullanılmasını sağlar.

RichTextBox Özellikleri

RichTextBox kontrolü kullanıcıya birçok seçenek sunar, dolayısıyla tasarım ve çalışma anında erişilebilen birçok özelliği bulunur.

Tasarım anında ulaşılabilecek özellikler:

<u>Özellik</u>	<u>Değer Tipi</u>	<u>Açıklama</u>
ZoomFactor	Single	Metnin büyüklüğünü belirler. 1 – 64 arası bir değer alır.
WordWrap	Boolean	Uzun yazıların bir sonraki satıra geçerek görüntülenmesini sağlar
DetectUrls	Boolean	Bağlantı olarak girilen yazıların LinkLabel şeklinde algılanmasını belirler
Lines	String()	Satırları String dizisi olarak tutar
BulletIntend	Integer	Satırların madde işaretinden kaç piksel açıkta duracağını belirler
AcceptsTab	Boolean	Tab tuşunu bir karakter olarak algılanmasını, dolayısıyla bu tuşa basıldığında kontrolden çıkılmasının engellenmesini belirler
ShowSelectionMargin	Boolean	Satır başındaki boşluğun gösterilmesini belirler

RightMargin Integer Satırların maksimum uzunluğunu piksel cinsinden belirler.

Çalışma anında ulaşılabilecek özellikler:

<u>Özellik</u>	<u>Değer Tipi</u>	<u>Açıklama</u>
Capture	Boolean	Kontrol içine yazı yazarken farenin gizlenmesini belirler
UndoActionName	String	En son yapılabilecek Undo işleminin ismini tutar
RedoActionName	String	Undo işlemi yapıldıktan sonra, en son yapılabilecek Redo işleminin ismini tutar.
SelectedText	String	Seçilen metni belirler
SelectionBullet	Boolean	Seçilen satırın madde işaretli olarak görüntülenmesini belirler
SelectionAlign	Boolean	Seçilen satırın hizalanmasını belirler
SelectionColor	Color	Seçilen metnin rengini belirler
SelectionFont	Font	Seçilen metnin yazı tipini belirler
SelectionIntend	Integer	Seçilen satırın, sol kenara olan uzaklığını belirler
SelectionLength	Integer	Seçilen metnin uzunluğunu belirler

RichTextBox Metotları

<u>Metot</u>	<u>Açıklama</u>
Find	Metin kutusu içinde, parametre olarak verilen bir yazıyı arar. Yazıyı ilk gördüğü yerin indisini döndürür.
LoadFile	Bir dosyadan alınan metni yükler
SaveFile	Parametre olarak verilen konumdaki dosyaya, metni yazar. Dosyanın rtf veya doc uzantılarında kaydedilmesi, zengin içeriğin görüntülenmesi açısından önemlidir.
Undo	Yapılan işlem geriye alınır
Redo	Geri alınan işlem tekrar yapılır

RichTextBox Olayları

<u>Olay</u>	<u>Açıklama</u>
TextChanged	Metin kutusundaki yazı değiştiği zaman gerçekleşir
LinkClicked	Metin içindeki bir bağlantıya tıklandığı zaman gerçekleşir

SaveFileDialog Kontrolü

Bu kontrol sayesinde kullanıcılar windowsun kullandığı ortak iletişim formlarını kullanarak dosya kaydetme işlemlerini gerçekleştirebilirler. Windowsun dosya kaydet penceresini görüntüler. Bir çok özelliği OpenFileDialog kontrolü ile aynıdır.



```
private void button1_Click(object sender, EventArgs e)
{
    saveFileDialog1.Title = "Lütfen Dosya Seçiniz";
}
```


```

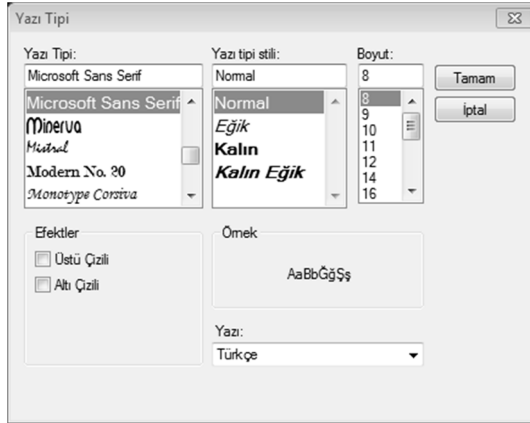
saveFileDialog1.InitialDirectory="C:\\Documents and
Settings\\Desktop\\blog";
saveFileDialog1.ShowDialog( );
string str = saveFileDialog1.FileNames;
textBox1.Text = str;
}

```

FontDialog Kontrolü

Windowsun kullandığı ortak iletişim formlarını kullanarak font paleti işlemlerini gerçekleştirebilir. Windowsun yazı tipi penceresini görüntüler.

 fontDialog1



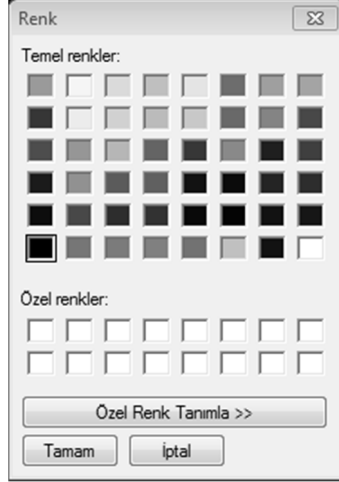
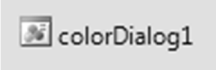
```

private void button1_Click(object sender, EventArgs e)
{
// Yazı Tipi Penceresinde Renk Paletini de Görüntüler
fontDialog1.ShowColor=true;
// Yazı Tipi Penceresini Görüntüler
fontDialog1.ShowDialog();
// Pencereden Seçilen Yazı Tipi Font Özelliğinde Tutulur.
richTextBox1.SelectionFont = fontDialog1.Font;
}

```

ColorDialog Kontrolü

Windowsun kullandığı ortak iletişim formlarını kullanarak renk paleti işlemlerini gerçekleştirebilir. Windowsun renk penceresini görüntüler.



```
private void button1_Click(object sender, EventArgs e)
{
    // Renk Penceresini Görüntüler
    colorDialog1.ShowDialog();
    // Pencereden Seçilen Renk Color Özelliğinde Tutulur.
    richTextBox1.SelectionColor = colorDialog1.Color;
}
```

Uygulama: RichTextBox ta seçilen metnin yazı tipi ve rengi düzenleme programı.



```

// Font Butonu
private void button1_Click(object sender, EventArgs e)
{
    fontDialog1.ShowDialog();
    richTextBox1.SelectionFont = fontDialog1.Font;
}

// Renk Butonu
private void button2_Click(object sender, EventArgs e)
{
    colorDialog1.ShowDialog();
    richTextBox1.SelectionColor = colorDialog1.Color;
}

```

Uygulama: Basit Not defteri uygulaması. (Uygulamada OpenFileDialog, SaveFileDialog, FontDialog ve ColorDialog kontrolleri de kullanılmıştır.



```

// Font Butonu
private void button1_Click(object sender, EventArgs e)
{
    fontDialog1.ShowDialog();
}

```

```

        richTextBox1.SelectionFont = fontDialog1.Font;
    }

    // Renk Butonu
    private void button2_Click(object sender, EventArgs e)
    {
        colorDialog1.ShowDialog();
        richTextBox1.SelectionColor = colorDialog1.Color;
    }

    // Dosya Aç Butonu
    private void button3_Click(object sender, EventArgs e)
    {
        string dosya;
        openFileDialog1.ShowDialog();
        dosya = openFileDialog1.FileName;
        richTextBox1.LoadFile(dosya,
            RichTextBoxStreamType.PlainText);
    }

    // Dosya Kaydet Butonu
    private void button4_Click(object sender, EventArgs e)
    {
        string dosya;
        saveFileDialog1.ShowDialog();
        dosya = saveFileDialog1.FileName;
        richTextBox1.SaveFile(dosya);
    }

    // Arama Butonu
    private void button5_Click(object sender, EventArgs e)
    {
        string aranan;
        aranan = textBox1.Text;
        int sonuc = richTextBox1.Text.IndexOf(aranan, 0);
    }

```

```
        if (sonuc == -1)
            MessageBox.Show("Aranan Kelime Yok");
        else
        {
            richTextBox1.Focus();
            richTextBox1.Select(sonuc, aranan.Length);
        }
    }

    // Metni Boyutlandır Uygula Butonu
    private void button6_Click(object sender, EventArgs e)
    {
        if (radioButton1.Checked)
            richTextBox1.ZoomFactor += 0.10f;
        if (radioButton2.Checked)
            richTextBox1.ZoomFactor -= 0.10f;
    }
}
```

Menü Tasarımı

Windows uygulamalarında en çok kullanılan tasarım araçları menülerdir. Dosya, düzen, görünüm gibi menüler neredeyse tüm Windows uygulamalarında, belli başlı işlerin yapılmasında kullanıcıya kolay erişim sağlar.

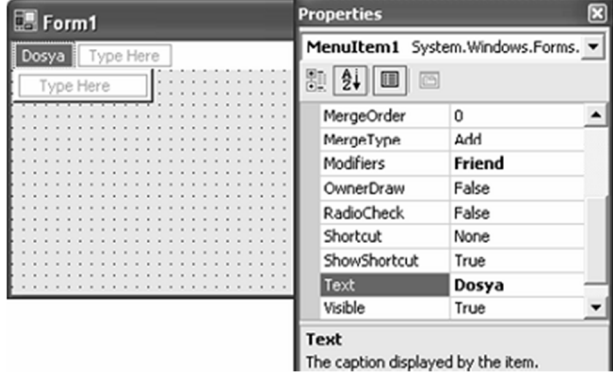
Uygulamalarda, menülerde tanımlanan işlemlere görsel kısa yollar sunulur. Bu işlem araç kutuları ile sağlanır.

Menüler

Windows uygulamalarında kullanılan iki tip menü vardır. MainMenu, formların başında duran sabit menüdür. ContextMenu, fare ile sağ tıklandığında çıkan menüdür.

MainMenu (MenuStrip) Kontrolü

Formların başında duran menüdür. MenuItem nesnelерinden oluşur. Menü öğelerine kısa yollar atanabilir.



Windows uygulamasına bir menü eklemek için, Toolbox panelinden bir MenuStrip kontrolünü forma sürükleyin. Eklenen menü bir bileşen olarak formun alt bölümünde gözükecektir. Ancak üstüne gelindiğinde formun başlığının hemen altında belirir. Menü öğesi eklemek veya ismini değiştirmek için üstüne gelinir ve başlık yazısı yazılır. Properties panelinde bu menünün MenuItem olarak eklendiği görülür.

Menüye MenuItem eklendiğinde hemen altında ve yanında, menü eklemek için bir yer açılır. Bu açılan yere de menü ismi girip, alt menü öğeleri oluşturulabilir. Menü öğelerine basıldığı zaman bir işlemin gerçekleşmesi için, kontrole çift tıklanarak bu öğenin Click olayına geçilir. Çalıştırılmak istenen kodlar buraya yazılır.

```
private void dosyaToolStripMenuItem_Click(object sender,
EventArgs e)
{
    // Dosya Menüsüne ait kodlar buraya ...
}
```

Menü öğelerine isim verirken & işareti kullanılarak, kullanıcın klavyenin ALT tuşuyla bu öğeyi çalıştırmasını sağlanabilir. & işareti hangi karakter ile kullanılırsa (hangi

RightToLeftAutoMirro	False
ShortcutKeyDisplayStr	
ShortcutKeys	None
ShowShortcutKeys	True
Size	51; 20
Tag	
Text	&Dosya
TextAlign	MiddleCenter
TextDirection	Horizontal
TextImageRelation	ImageBeforeText

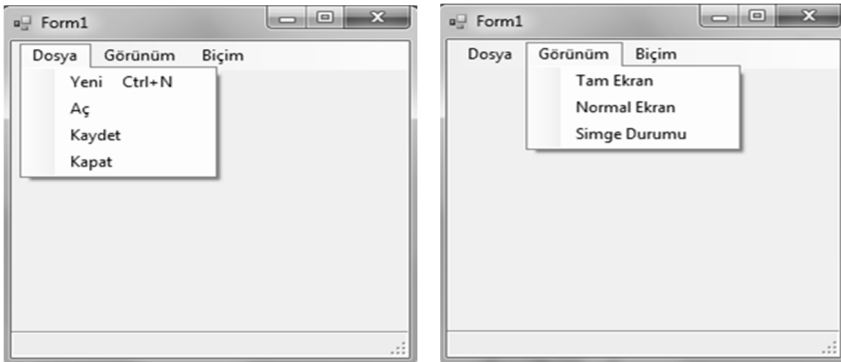
karacterin solundaysa) kısa yol olarak o karakter kullanılır.

Yandaki şekilde **&Dosya** ile ALT+D tuş kombinasyonuyla Dosya menüsü açılır.

MenuItem Özellikleri

<u>Özellik</u>	<u>Değer Tipi</u>	<u>Açıklama</u>
Checked	Booleand	Menü öğesinin yanında seçili olduğuna dair bir işaretin gözükmelerini sağlar
Enabled	Boolean	Menü öğesinin aktif durumda olup olmadığını belirler
RadioCheck	Boolean	Öğenin seçilme stiline RadioButton düğmesi olarak gözükmelerini sağlar.
ShortcutKeys	Shortcut	Menüye ulaşım için bir kısa yol tanımlar .
ShowShortcutKeys	Boolean	Menünün kısa yolunun, isminin yanında gözükmelerini belirler
MenuItems	MenuItemCollection	Alt menülerin tutulduğu koleksiyondur .

Örnek Uygulama: Menülü not defteri uygulaması.





// Dosya Menüsü → Yeni

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

```
{  
    if (richTextBox1.Visible == false)  
        richTextBox1.Visible = true;  
    else  
    {  
        DialogResult s;  
        s = MessageBox.Show("Dosya Kaydedilsinmi?",  
        "Kayıt ???", MessageBoxButtons.YesNo);  
        if (s == DialogResult.Yes)  
        {  
            dosyaKaydetToolStripMenuItem_Click(sender, e);  
            richTextBox1.Text = null;  
        }  
        else  
            richTextBox1.Text = null;  
        }  
    }  
}
```

// Dosya Menüsü → Aç

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

```
{
```

```

string dosya;
if (richTextBox1.Visible == false)
{
    openFileDialog1.ShowDialog();
    dosya = openFileDialog1.FileName;
    richTextBox1.Visible = true;
    richTextBox1.LoadFile(dosya,
    RichTextBoxStreamType.PlainText);
}

else
{
    DialogResult s;
    s = MessageBox.Show("Açık Dosya Kaydedilsinmi?",
    "Kayıt ???", MessageBoxButtons.YesNo);
    if (s == DialogResult.Yes)
    {
        dosyaKaydetToolStripMenuItem_Click(sender, e);
        richTextBox1.Visible = false;
        dosyaAçToolStripMenuItem_Click(sender, e);
    }
    else
    {
        richTextBox1.Visible = false;
        dosyaAçToolStripMenuItem_Click(sender, e);
    }
}
}

```

// Dosya Menüsü → Kaydet

```

private void dosyaKaydetToolStripMenuItem_Click(object sender,
EventArgs e)
{
    string dosya;
    saveFileDialog1.ShowDialog();
}

```

```

        dosya = saveFileDialog1.FileName;
        richTextBox1.SaveFile(dosya);
    }

    // Dosya Menüsü → Kapat
    private void kapatToolStripMenuItem_Click(object sender,
    EventArgs e)
    {
        if (richTextBox1.Visible == false)
            this.Close();
        else
        {
            DialogResult s;
            s = MessageBox.Show("Dosya Kaydedilsinmi?",
            "Kayıt ???", MessageBoxButtons.YesNo);
            if (s == DialogResult.Yes)
            {
                dosyaKaydetToolStripMenuItem_Click(sender, e);
                this.Close();
            }
            else
                this.Close();
        }
    }

    // Görünüm Menüsü → Tam Ekran
    private void tamEkranToolStripMenuItem_Click(object sender,
    EventArgs e)
    {
        this.WindowState = FormWindowState.Maximized;
    }

    // Görünüm Menüsü → Normal Ekran
    private void normalEkranToolStripMenuItem_Click(object sender,
    EventArgs e)

```

```

{
    this.WindowState = FormWindowState.Normal;
}

// Görünüm Menüsü → Simge Durumu
private void simgeDurumuToolStripMenuItem_Click(object sender,
EventArgs e)
{
    this.WindowState = FormWindowState.Minimized;
}

// Biçim Menüsü → Yazı Tipi
private void yazıTipiToolStripMenuItem_Click(object sender,
EventArgs e)
{
    fontDialog1.ShowDialog();
    richTextBox1.SelectionFont = fontDialog1.Font;
}

// Biçim Menüsü → Yazı Rengi
private void yazıRengiToolStripMenuItem_Click(object sender,
EventArgs e)
{
    colorDialog1.ShowDialog();
    richTextBox1.SelectionColor = colorDialog1.Color;
}

```

ContextMenu(ContextMenuStrip) Kontrolü

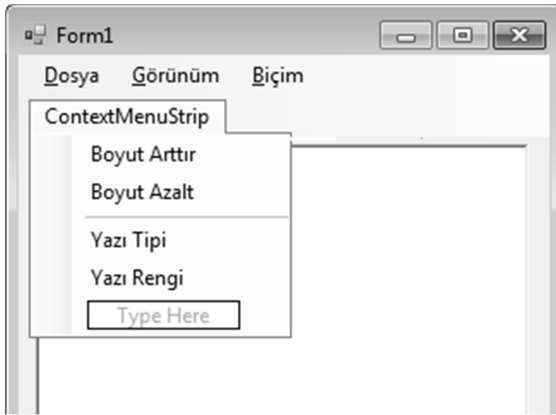
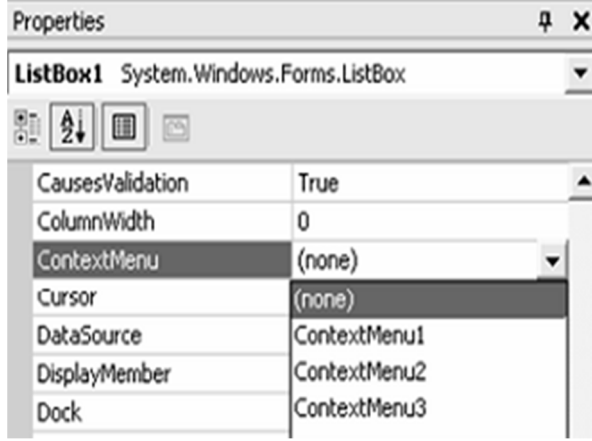
Kontrollerin ContextMenu özelliğine atanır. Kontrollere sağ tıklandığı zaman çıkan menüdür.

Toolboxtan ContextMenuStrip kontrolü forma sürüklenip bırakılır ve menü oluşturulur.

Hangi kontrolün sağ tuş menüsü olacak ise o kontrolün ContextMenu özelliğinden oluşturulan ContextMenuStrip ögesi seçilir.

ContextMenu, bir kontrolün üstüne sağ tıklandığı zaman çıkan menüdür. Bu menü uygulamaya eklendiği zaman Properties panelinde, kontrollerin ContextMenu özelliği olarak bu menü atanabilir.

Uygulamanın Devamı : Bir önceki uygulamada RichTextBox için bir Context Menü ekleyip kodlarını yazalım.



ContextMenu oluşturduktan sonra RichTextBox'ın ContextMenu özelliğinden oluşturduğumuz menüyü seçmemiz gerekir.

```
// Context Menü → Boyut Arttır
private void boyutArttırToolStripMenuItem_Click(object sender,
EventArgs e)
{
    richTextBox1.ZoomFactor += 0.1f;
}
```

```
// Context Menü → Boyut Azalt
private void boyutAzaltToolStripMenuItem_Click(object sender,
EventArgs e)
{
    richTextBox1.ZoomFactor -= 0.1f;
}
```

```
// Context Menü → Yazı Tipi
private void yazıTipiToolStripMenuItem1_Click(object sender,
EventArgs e)
{
    // Biçim Menüsünde aynı işlem yapıldığı için
    // Biçim Menüsünün Yazı Tipi Metodunu çağırıyoruz
    yazıTipiToolStripMenuItem_Click(sender, e);
}
```

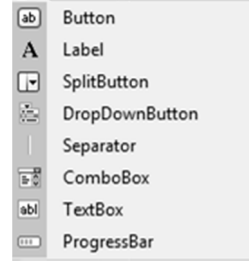
```
// Context Menü → Yazı Rengi
private void yazıRengiToolStripMenuItem1_Click(object sender,
EventArgs e)
{
    // Biçim Menüsünde aynı işlem yapıldığı için
    // Biçim Menüsünün Yazı Rengi Metodunu çağırıyoruz
    yazıRengiToolStripMenuItem_Click(sender, e);
}
```

ToolStrip Kontrolü

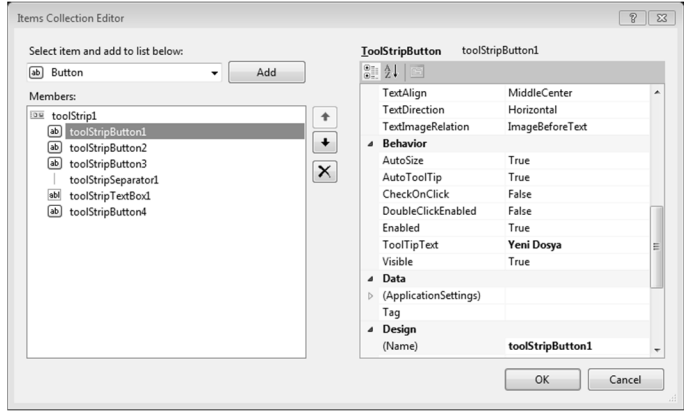
Menülerin işlevlerine görsel kısa yollar sunar.

ToolStrip çeşitli nesnelere oluşturur. →

Hangi düğmeye basıldığı ButtonClick olayı ile anlaşılır.



ToolStrip kontrolü menülerin altında kullanıcıya kısa yollar, kullanım kolaylığı sunan bir kontroldür. Kontroldeki öğeler çoğu zaman Image



özelliğinin sağladığı resimler ile gösterilir. Resim yerine yazı da gösterilebilir ancak yazı ile işlem listelemek menüler ile sağlanır.

ToolStrip kontrolüne nesnelere eklemek için kontrolün Items Collection özelliğinde faydalanılır. Tasarım anında Properties panelinden Items Collection özelliğine basıldığı zaman çıkan pencerede, kontrole yeni nesne eklenir.

ToolTipText Özelliği

Kontrollerin üzerine gelindiğinde bilgi mesajı verir. Mesaj, kontrollerin "ToolTipText" özelliğine yazılır.



Uygulamanın Devamı : Bir önceki uygulamaya ToolStrip ile araç çubuğu ekleyip kodlarını yazalım.



```
// ToolStrip → Yeni Dosya Butonu
private void toolStripButton1_Click(object sender, EventArgs e)
{
    // Dosya Menüsünde aynı işlem yapıldığı için
    // Dosya Menüsünün Yeni Dosya Metodunu çağırıyoruz
    yeniDosyaToolStripMenuItem_Click(sender, e);
}
```

```
// ToolStrip → Dosya Aç Butonu
private void toolStripButton2_Click(object sender, EventArgs e)
{
    // Dosya Menüsünde aynı işlem yapıldığı için
    // Dosya Menüsünün Dosya Aç Metodunu çağırıyoruz
    dosyaAçToolStripMenuItem_Click(sender, e);
}
```

```
// ToolStrip → Dosya Kaydet Butonu
private void toolStripButton3_Click(object sender, EventArgs e)
{
    // Dosya Menüsünde aynı işlem yapıldığı için
    // Dosya Menüsünün Dosya Kaydet Metodunu çağırıyoruz
    dosyaKaydetToolStripMenuItem_Click(sender, e);
}
```

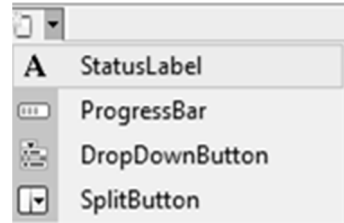


```
// ToolStrip → Arama Butonu
private void toolStripButton4_Click(object sender, EventArgs e)
{
    string aranan;
    aranan = toolStripTextBox1.Text;
    int sonuc = richTextBox1.Text.IndexOf(aranan, 0);
    if (sonuc == -1)
        MessageBox.Show("Aranan Kelime Yok");
    else
    {
        richTextBox1.Focus();
        richTextBox1.Select(sonuc, aranan.Length);
    }
}
}
```

Status Bar (ToolStrip Kontrolü)

Windows formlarının durum çubuğudur. Windows uygulamalarında formların altında bulunan durum çubuğunu temsil eder.

StatusStrip kontrolündeki etiket ile durum çubuğunda yazı gösterilebileceği gibi farklı nesnelere eklenebilir. →



Uygulamanın Devamı : Bir önceki uygulamaya ToolStrip ile StatusLabel ekleyip Dosya → Yeni menüsüne Mouse konulduğunda durum çubuğunda bilgi verelim.



```

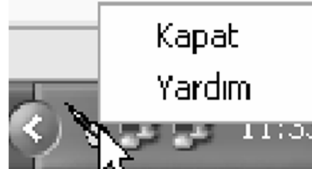
private void yeniDosyaToolStripMenuItem_MouseEnter(object
sender, EventArgs e)
{
    // Mouse Dosya Menüsünde Yeni üstüne gelince
    // Durum Çubuğunda "Yeni Dosya Oluştur" görüntülenir.
    toolStripStatusLabel1.Text = "Yeni Dosya Oluştur";
}

private void yeniDosyaToolStripMenuItem_MouseLeave(object
sender, EventArgs e)
{
    // Mouse Dosya Menüsünde Yeni kısmından ayrılınca
    // Durum Çubuğunu boşaltmamız gerekir.
    toolStripStatusLabel1.Text = null;
}

```

NotifyIcon Kontrolü

Windows görev çubuğunda görüntülenen simgedir.



NotifyIcon Özellikleri

<u>Özellik</u>	<u>Değer Tipi</u>	<u>Acıklama</u>
Icon	Icon	Görev çubuğunda gözükecek simgeyi belirler
ContextMenu	Menu	Simgeye sağ tıklandığı zaman açılacak menüyü belirler
Text	String	Simge üzerine gelindiğinde görüntülenecek yazıyı belirler.

Dosyalama

C# ta dosya işlemleri temel olarak akımlar (streamler) üzerine kuruludur. Akım (Stream), byte düzeyinde bir girdi bilgisi oluşturan veya çıktı bilgisi elde eden mantıksal bir birimdir. Bu birimler I/O sistemi aracılığıyla dosya, ekran gibi fiziksel aygıta bağlanır. Bu konuda en çok kullanılan dosya tipi disk dosyalarıdır. Disk üzerinde bilgi okuma, yazma, klavyeden bilgi alma ve ekrana bilgi yazdırma gibi işlemler için akım yöntemi kullanılır. Akım (Stream) yöntemi okuma veya yazma bakımından bellek kadar hızlı ve kullanışlı olmadığı için ve işletim sistemi düzeyinde tüm dosyaların byte olması nedeniyle okuma, yazma işlemleri bir kere de değil parça parça byte olarak işlenir.

.NET Framework'te hem byte hem de karakter akımı sınıflar mevcuttur. Karakter akımı sınıflar temelde byte akımı sınıfları kullanır. Akım (Stream) sınıflarının temelini soyut bir sınıf olan "System.IO.Stream" sınıfı oluşturur. MemoryStream ve FileStream byte Stream sınıfından türemiş sınıflardır. Karakter akımı için kullanılan StreamWriter, StreamReader sınıfları ise TextWriter ve TextReader sınıflarından türemiş sınıflardır. Bu akımlara ek olarak kullanılan binary(ikili) akım yönteminde ise BinaryWriter ve BinaryReader sınıfları kullanılır.

Temelde metin tabanlı dosyaları okuma, yazma işlemleri için TextWriter, TextReader ve bu sınıflardan türemiş diğer sınıflar kullanılır. İkili (Binary) dosyalar için BinaryReader ve BinaryWriter sınıfları kullanılır. Her iki dosya tipinde çalışmak için ise FileStream ve bu sınıftan türeyen sınıflar kullanılır.

StreamWriter ve StreamReader Sınıfları

StreamWriter Sınıfı ve Elemanları

Belirli kodlamalardaki karakterleri bir stream(akım)a yazarken

"TextWriter" uygulamasını sağlar. Yazma işlemi için StreamWriter sınıfı kullanılır. Bunun için önce bu sınıftan bir nesne oluşturulur. StreamWriter nesnesi tanımlarken string tipinde bir parametre girilmesi gerekir. Bu parametre üzerinde işlem yapacağımız dosyanın adresinden ve isminden oluşan bir path'tir. Verilen adreste path içinde adı geçen dosya olmayabilir. Bu durumda program önce dosyayı oluşturacak sonra üzerinde işlem yapacaktır.

```
private void button1_Click(object sender, EventArgs e)
{
    StreamWriter SW = new StreamWriter(@"c:\deneme.txt");
    // TextBox taki bilgi dosyaya yazdırılıyor.
    SW.WriteLine(textBox1.Text);
    // Dosya değişkeni kapatılıyor.
    SW.Close();
}
```

StreamWriter Metodları

- **Close:** Mevcut "StreamWriter" ı kapatır.
- **Flush:** Mevcut "StreamWriter" için tüm ara belleği (buffer) siler ve herhangi bir ara bellek (buffer) bilgisinin akıma (stream) yazılmasını sağlar.
- **Equals:** İki örnek nesnenin eşit olduğunu tanımlar.
- **GetType:** Mevcut olayın tipini alır.
- **ToString:** Örnek nesneyi katar (string) çevirir.
- **Write:** Dosyaya akım olarak yazar.
- **WriteLine:** Dosyaya akım olarak yazar ve dosya işaretçisini yeni satıra yönlendirir.

StreamWriter Özellikleri

- **BaseStream:** Temel akımı arayüz ile yedeklemeyi sağlar.
- **Encoding:** Yazılan çıktıların kodlamalarını alır.

- **NewLine:** Akım (Stream) kullanılarak bitirilmiş katarlar satırlarını alır ve düzenler.

"StreamWriter" sınıfına örnek kısa bir uygulama olarak aşağıdaki kodlar verilebilir. Bu kodlarda verilen adreste bir "**TestDosyasi.txt**" oluşturularak içerisine istenilen bilgilerin yazılması sağlanır.

```
StreamWriter sw = new StreamWriter(@"C:\TestDosyasi.txt");
sw.Write("Baslik ");
sw.WriteLine("İçerik.");
sw.WriteLine("-----");
sw.Write("Tarih: ");
sw.WriteLine(DateTime.Now);
```

StreamReader Sınıfı ve Elemanları

Okuma işlemi için StreamReader sınıfı kullanılır. Bunun için önce bu sınıftan bir nesne oluşturulur. StreamReader nesnesi tanımlarken string tipinde bir parametre girilmesi gerekir. Bu parametre üzerinde işlem yapacağımız dosyanın adresinden ve isminden oluşan bir path'tir. Verilen adreste path içinde adı geçen dosya olmak zorundadır.

```
private void button2_Click(object sender, EventArgs e)
{
    StreamReader SR = new StreamReader(@"c:\deneme.txt");
    // Dosyadaki bir satırlık bilgi RichTextBox'a yazdırılıyor.
    richTextBox1.Text = SR.ReadLine();
    // Dosya değişkeni kapatılıyor.
    SR.Close();
}
```

StreamReader Metodları

- **Close:** "StreamReader"i kapatır.
- **GetType:** Geçerli örneğin tipini alır.

- **Read:** Dosyadan ilgili satırı okur.
- **ReadLine:** Dosyadan ilgili satırı okuyarak kataca (string) çevirir ve dosya işaretçisini yeni satıra yönlendirir.
- **ReadToEnd:** Dosyanın tamamını sonuna kadar okur.
- **ToString:** Mevcut nesneyi katar (string) olarak değiştirir.

StreamReader Özellikleri

- **BaseStream:** Temel akımı (stream) geri çevirir.

"StreamReader" sınıfına örnek olarak aşağıdaki gibi bir basit bir uygulama verilebilir. Bu örnekte "**TestDosyasi**" isimli bir dosyanın içerisindeki bilgileri okuyarak konsol ekranına yazılmasını sağlar.

```

StreamReader sr = new StreamReader(@"C:\TestDosyasi.txt");
string line;
while ((line = sr.ReadLine()) != null)
{
    Console.WriteLine(line);
}

```

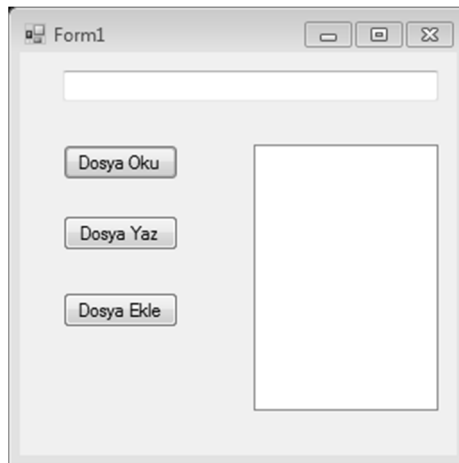
Ekleme işlemi, yazma işlemine benzemektedir. Bu işlem için de StreamWriter sınıfını kullanılır. Aralarındaki fark tanımlamada ve uygulamada ortaya çıkar. Ekleme işlemi tanımlarken File.AppendText kullanılır. AppendText string tipinde bir parametre ister. Bu parametre ekleme işlemi yapılacak dosyanın adresinden ve isminden oluşan bir path' tir. Uygulama olarak yazma işleminden farkı, dosya içinde daha önceden var olan verileri koruyarak dosyaya yeni veriler eklemesidir. Bu işlemde de verilen adreste path içinde adı geçen dosya olmayabilir. Bu durumda program önce dosyayı oluşturacak sonra üzerinde işlem yapacaktır.

```

private void button3_Click(object sender, EventArgs e)
{
    StreamWriter SW = File.AppendText(@"c:\deneme.txt");
    // TextBox taki bilgi dosyaya yazdırılıyor.
    SW.WriteLine(textBox1.Text);
    // Dosya değişkeni kapatılıyor.
    SW.Close();
}

```

Uygulama: Dosya Yazma, Okuma, Ekleme uygulaması.



```

// Dosya Oku Butonu
private void button1_Click(object sender, EventArgs e)
{
    listBox1.Items.Clear();
    openFileDialog1.ShowDialog();
    StreamReader sr = new
    StreamReader(openFileDialog1.FileName);
    string satir;
    while ((satir = sr.ReadLine()) != null)
    {
        listBox1.Items.Add(satir);
    }
}

```

```

    }
    sr.Close();
}

// Dosya Yaz Butonu
private void button2_Click(object sender, EventArgs e)
{
    StreamWriter sw;
    saveFileDialog1.ShowDialog();
    sw = new StreamWriter(saveFileDialog1.FileName);
    sw.WriteLine(textBox1.Text);
    sw.Close();
    listBox1.Items.Clear();
    StreamReader sr = new
    StreamReader(saveFileDialog1.FileName);
    string satir;
    while ((satir = sr.ReadLine()) != null)
    {
        listBox1.Items.Add(satir);
    }
    sr.Close();
}

// Dosya Ekle Butonu
private void button3_Click(object sender, EventArgs e)
{
    StreamWriter sa =
    File.AppendText(saveFileDialog1.FileName);
    sa.WriteLine(textBox1.Text);
    sa.Close();

    listBox1.Items.Clear();
    StreamReader sr = new
    StreamReader(saveFileDialog1.FileName);
    string satir;

```



```
        while ((satir = sr.ReadLine()) != null)
        {
            listBox1.Items.Add(satir);
        }
        sr.Close();
    }
```

FileStream Sınıfı

Her tipteki dosya için işlem yapabilme yeteneğine sahip metod ve özellikleri barındıran bir sınıftır. Bu sınıf dosyaları, dosya sisteminde yazmak, okumak ya da açıp kapamak için kullanılır. Bunun yanında dosya ilişkili işletim sistemlerinde veri geçişi, standart girdi ve çıktılarda tanıtıcı değer olarak kullanılabilir. Dosya okuma ve yazma işlemleri isteğe bağlı olarak senkronize çalışabilirler. Ayrıca "FileStream" daha iyi bir performans için girdi ve çıktılarda ara bellek (buffer) olarak kullanılabilmesini sağlar.

Kullanım Şekli;

```
FileStream Nesne = new FileStream(Dosya_Yolu, Dosya_Modu, Erişim_tipi, Paylaşım Tipi);
```

İlk parametre hangi dosyadan işlem yapacaksanız o dosya yolunu, ikinci parametre ise **FileMode** türünden bir enum. Bu enum türünden dosyayı ne modda üreteceğinizi, üçüncü parametre dosyanın hangi amaçla açılacağını(okuma, yazma) belirtmek için **FileAccess** türünden bir enum, sonuncu parametre ise dosyanın farklı prosesler tarafından kullanımı ayarlamak için **FileShare** türünden bir enum değerini alır.

FileStream aşırı yüklemeli olduğu için bazı parametreler kullanılmayabilir.

FileMode Seçenekleri;

- **FileMode.Append** : Açılan dosyanın sonuna ekleme yapmak için kullanılır. Eğer dosya yoksa oluşturulur.
- **FileMode.Create** : Yeni dosya oluşturmak için kullanılır. Zaten dosya varsa üzerine yazılır.
- **FileMode.CreateNew** : Yeni dosya oluşturmak için kullanılır, belirtilen dosya mevcutsa çalışma zamanı hatası verir.
- **FileMode.Open** : Dosyayı açmak için kullanılır.
- **FileMode.OpenOrCreate** : Belirtilen dosya varsa açılır, yoksa yenisi oluşturulur.
- **FileMode.Truncate** : Belirtilen dosya açılır ve içeriği tamamen silinir.

FileAccess Seçenekleri

- **FileAccess.Read** : Dosya okumak için kullanılır.
- **FileAccess.ReadWrite** : Dosya okunmak ve yazılmak üzere açılır.
- **FileAccess.Write** : Dosya sadece yazılmak için açılır.

FileShare Seçenekleri

- **FileShare.Inheritable** : Dosyanın farklı prosesler tarafından türetilmesini sağlar.
- **FileShare.None** : Dosyanın aynı anda başka prosesler tarafından açılmasını engeller.
- **FileShare.Read** : Dosyanın aynı anda başka proseslerce de açılabilmesini sağlar.
- **FileShare.ReadWrite** : Dosyanın aynı anda başka proseslerce de açılıp, okunup, yazılabilmesini sağlar.
- **FileShare.Write** : Dosyaya aynı anda başka proseslerce yazılabilmesini sağlar.

Örnek Kullanım:

```
FileStream fs = new FileStream(@"c:\Deneme\personel.txt",  
    FileMode.Truncate, FileAccess.Write, FileShare.None );
```

Örnek Kullanım:

```
public static void Main()  
{  
    FileStream fs = new FileStream( "c:\\Deneme.txt",  
        FileMode.Append, FileAccess.Write, FileShare.Write);  
    fs.Close();  
    StreamWriter sw = new  
        StreamWriter("c:\\Deneme.txt",true, Encoding.ASCII);  
    string satir = ("Eklenen satir.");  
    sw.Write(satir);  
    sw.Close();  
}
```

FileStream Sınıfı Metodları

- **BeginRead:** Eş zamanlı olmayan okuma işlemlerini başlatır.
- **BeginWrite:** Eş zamanlı olmayan yazma işlemlerini başlatır.
- **Close:** "FileStream"i kapatır.
- **GetType:** Geçerli örneğin tipini alır.
- **Read:** Girilen veri topluluğunu okur.
- **ReadByte:** Girilen veri topluluğunu byte cinsinden okur.
- **ToString:** Mevcut nesneyi katar (string) türüne dönüştürür.
- **Write:** Akımda (Stream) tuttuğu veri gruplarını alarak yazar.
- **WriteByte:** "FileStream" in mevcut durumunu byte olarak yazar.

FileStream Özellikleri

- **CanRead:** Geçersiz bir sınıf türetildiğinde mevcut akımda (stream) bir değer olup olmadığını okur.
- **CanWrite:** Geçersiz bir sınıf türetildiğinde mevcut akımda (stream) bir değer olup olmadığını yazarak belirtir.
- **Lenght:** Akımın (Stream) uzunluğu byte cinsinden alınır.
- **Name:** Mevcut örnekteki "FileStream" in ismini alır.

Uygulama: Dosya Okuma Yazma.

Kayıt No	Adı	Soyadı
1001	Ahmet	Soytürk

// Form yüklendiğinde dosyadaki bilgiler ListBox'lara aktarılıyor.

```
private void Form1_Load(object sender, EventArgs e)
{
    FileStream fs;
    fs = new FileStream(@"c:\Deneme\personel.txt",
        FileMode.OpenOrCreate, FileAccess.Read,
        FileShare.None);
}
```

```

StreamReader sr;
sr = new StreamReader(fs);
string str;
while ((str = sr.ReadLine()) != null)
{
    lbxKayitNo.Items.Add(str);
    str = sr.ReadLine();
    lbxAdi.Items.Add(str);
    str = sr.ReadLine();
    lbxSoyadi.Items.Add(str);
}
sr.Close();
fs.Close();
}
// Kayıt Ekle Butonu
private void btnKayitEkle_Click(object sender, EventArgs e)
{
    if (txtKayitNo.Text == "" || txtAdi.Text == "" ||
        txtSoyadi.Text == "")
    {
        MessageBox.Show("Tüm Alanları Doldurunuz");
    }
    else
    {
        lbxKayitNo.Items.Add(txtKayitNo.Text);
        lbxAdi.Items.Add(txtAdi.Text);
        lbxSoyadi.Items.Add(txtSoyadi.Text);
    }
}

// Kayıt No ListBox'ında seçili değerin değişmesi olayı
private void lbxKayitNo_SelectedIndexChanged(object sender,
EventArgs e)
{
    lbxAdi.SelectedIndex = lbxKayitNo.SelectedIndex;
}

```

```

        lbxSoyadi.SelectedIndex = lbxKayitNo.SelectedIndex;
    }

// Kayıt Sil Butonu
private void btnKayitSil_Click(object sender, EventArgs e)
{
    int indexno = lbxKayitNo.SelectedIndex;
    if (indexno >= 0)
    {
        lbxKayitNo.Items.RemoveAt(indexno);
        lbxAdi.Items.RemoveAt(indexno);
        lbxSoyadi.Items.RemoveAt(indexno);
    }
    else
    {
        MessageBox.Show("Silinecek Kaydı Seçiniz!!!");
    }
}

// Form Kapanırken ListBox'lardaki bilgiler dosyaya yazılıyor.
private void Form1_FormClosing(object sender,
FormClosingEventArgs e)
{
    FileStream fs;
    fs = new FileStream(@"c:\Deneme\personel.txt",
        FileMode.Truncate, FileAccess.Write,
        FileShare.None);
    StreamWriter sw;
    sw = new StreamWriter(fs);
    int ks,i;
    ks = lbxKayitNo.Items.Count;
    if (ks > 0)
    {
        for (i = 0; i < ks; i++)
        {

```

```

        sw.WriteLine(lbxKayitNo.Items[i].ToString());
        sw.WriteLine(lbxAdi.Items[i].ToString());
        sw.WriteLine(lbxSoyadi.Items[i].ToString());
    }
}
else
{
    MessageBox.Show("Yazılacak Kayıt Yok");
}
sw.Close();
fs.Close();
}

```

BinaryReader ve BinaryWriter Sınıfları

İlkel veri tiplerini **ikili (binary)** değerler olarak okur ve yazar.

BinaryReader Sınıfı Metodları

- **Close:** Akımda (Stream) bulunan mevcut okumayı kapatır.
- **Equals:** İki örnek nesnenin eşit olduğunu söyler.
- **GetType:** Mevcut örneğin tipini alır.
- **Read:** Alt düzeydeki ve gelişmiş durumdaki akım (stream) karakterlerini okur.
- **ReadByte:** Mevcut akımın (stream) bir sonraki byte veriyi okur ya da tek byte kapasitesindeki akımın mevcut durumunu okur.
- **ReadInt32:** Mevcut işaretlenmiş akımda (stream) 4byte'lık bir bölümü okur.
- **ReadInt64:** Mevcut işaretlenmiş akım (stream) 8 byte'lık bir bölümü okur.
- **ReadString:** Mevcut sistemde bir katar (string) okur.
- **ReadUInt16:** 2 byte'lık işaretlenmemiş tamsayıyı mevcut akımdan okur.

- **ReadUInt32:** 4 byte'lik işaretlenmemiş tamsayıyı mevcut akımdan okur.
- **ReadUInt64:** 8 byte'lik işaretlenmemiş tamsayıyı mevcut akımdan okur.
- **ToString:** Nesneyi katar (string) yapıya dönüştürür.

BinaryReader Sınıfı Özellikleri

- **BaseStream:** "BinaryReader" in temel akıma (stream) girişini sağlar.

Örnek Kullanım:

```
int i;
decimal d;
char c;
FileStream fs = new FileStream(@"C:\Deneme\bilgi.dat",
                             FileMode.Open,
                             FileAccess.Write,
                             FileShare.None);
BinaryReader br = new BinaryReader(fs);
i = br.ReadInt32();
d = br.ReadDecimal();
c = br.ReadChar();
```

BinaryWriter Sınıfı Metodları

- **Close:** Akımda (Stream) bulunan "BinaryWriter" i kapatır.
- **Equals:** İki örnek nesnenin birbirine eşit olduğunu belirtir.
- **GetType:** Mevcut örneğin tipini almak için kullanılır.
- **ToString:** Mevcut nesnenin türünü katar.
- **Write:** Arabelleğe alınmış veri bloklarının yazılmasını sağlar (Mevcut akımda bir değer yazar).

BinaryWriter Sınıfı Özellikleri

- **BaseStream:** "BinaryWriter" in alt akımını alır.

Örnek Kullanım:

```
int i;
decimal d;
char c;
i = Convert.ToInt32(textBox1.Text);
d = Convert.ToDecimal(textBox2.Text);
c = Convert.ToChar(textBox3.Text);
FileStream fs = new FileStream(@"C:\Deneme\bilgi.dat",
                             FileMode.OpenOrCreate,
                             FileAccess.Write,
                             FileShare.None);
BinaryWriter bw = new BinaryWriter(fs);
bw.Write(i);
bw.Write(d);
bw.Write(c);
```

File Sınıfı

File sınıfının tüm metotları statiktir. Bu nedenle dosya işlemlerinde tek bir eylem gerçekleştirilecekse File sınıfı uygundur. Ancak daha kapsamlı bir işlem söz konusu ise yine Sytem I/O Namespace'i (İsim uzayı) içinde bulunan FileInfo sınıfını kullanmak daha uygun olur.

File.Create Metodu

```
File.Create("C:\\deneme.txt");
```

Yeni bir dosya oluşturmak için File.Create metodu kullanılır. Bu metod ile C# ta dilediğiniz dizine dosya oluşturabilirsiniz. Yapmanız gereken dosyanın hangi isimle nerede oluşturulacağını yazmanız.

File.AppendAllText Metodu

Bu metod ile var olan bir dosyaya eklemek istediğiniz bir satırı yazabilirsiniz. Bu metodun şöyle bir özelliği var: Eğer yolunu belirttiğiniz dosya varsa dosyayı açar, gönderdiğiniz değeri içerisine ekler. Dosya içerisinde herhangi bir veri varsa bunlar silinmez, sadece gönderilen değeri dosyaya ekler. Diğer bir özelliği ise, eğer belirttiğiniz yolda böyle bir dosya yoksa exception vermeyecektir. Çünkü eğer böyle bir dosya yoksa kendisi oluşturup, içerisine gönderdiğiniz veriyi yazacaktır.

```
string dosyaYolu = @"c:\MyTest.txt";  
string eklenecekYazi = "Bu satır dosyaya yazılacak";  
eklenecekYazi += Environment.NewLine;  
File.AppendAllText(dosyaYolu, eklenecekYazi);
```

Aşağıdaki satırı kullanmamızın amacı yeni bir satır oluşturmaktır.

```
eklenecekYazi += Environment.NewLine;
```

İkinci aşırı yüklenmiş haliyle de Encoding değerini kendiniz belirleyebilirsiniz. Şöyleki türkçe karakter sorunu yaşarsanız ikinci aşırı yüklenmiş halini kullanabilirsiniz.

```
File.AppendAllText(dosyaYolu, eklenecekYazi,Encoding.UTF8);
```

File.Copy Metodu

```
File.Copy("c:\\deneme.txt", "e:\\deneme.txt");
```

C# ta bir dizinden başka bir dizine dosya kopyalamak için File.Copy metodu kullanılır. Eğer belirttiğiniz kaynak dosya yoksa dosyanın olmadığına dair exception alırsınız. Aynı şekilde hedef dizinde aynı isimle bir dosya varsa uygulamanız yine hataya düşecektir. Bunun için bu işlemi yapmadan önce File.Exists metodu ile dosyanın olup olmadığını kontrol edebilirsiniz.

File.CreateText Metodu

Bu metod ile içerisine yazılmak için bir dosya açılır. Geriye StreamWriter nesnesi döner. Bu metodu kullanacaksanız içerisine veri girmek için StreamWriter nesnesini kullanabilirsiniz.

```
using (StreamWriter sw = File.CreateText("C:\\deneme.txt"))
{
    sw.WriteLine("İlk Satır");
    sw.WriteLine("ve");
    sw.WriteLine("Üçüncü Satır");
}
```

File.Delete Metodu

```
File.Delete("C:\\deneme.txt");
```

Bu metodu kullanarak parametre olarak verdiğiniz dosyayı silebilirsiniz. Ancak dosya kullanılıyorsa veya belirttiğiniz dosya yoksa exception alırsınız.

File.Exists Metodu

```
File.Exists("C:\\deneme.txt");
```

Bu metodu kullanarak parametre olarak verdiğiniz dosyanın var olup olmadığını öğrenebilirsiniz. Eğer belirttiğiniz isimde bir dosya varsa True döner, yoksa false döner.

File.GetAttributes Metodu

```
FileAttributes attr = File.GetAttributes("c:\\deneme.txt");
```

Bu metodu kullanarak bir dosyaya ait belli başlı özellikleri alabilirsiniz. Mesela dosyanın gizli dosya olup olmadığını, salt okunurluğu gibi özellikleri alabilirsiniz.

File.GetCreationTime ve File.GetCreationTimeUtc Metodu

```
string dosya = "C:\\deneme.txt";  
DateTime olusturmaZamani= File.GetCreationTime(dosya);
```

Bu metodu kullanarak parametre olarak verilen dosyanın oluşturulma zamanını alabilirsiniz. İkinci metotta ise yani File.GetCreationTimeUtc metodu ile evrensel saate göre oluşturulma zamanını getirir.

Directory sınıfı

Directory sınıfının hiçbir özelliği yoktur, System.IO altında bulunur, sadece static metotlar içerir.

Directory.CreateDirectory(string adres)

Adres ile belirtilen adreste bir klasör oluşturur ve bu klasör bilgilerini bir DirectoryInfo nesnesi olarak tutar. Programımızın çalıştığı klasörde bir klasör oluşturmak için sadece klasörün adını yazmak yeterlidir.

Örnekler:

```
Directory.CreateDirectory(@"C:\WINDOWS\deneme");
```

Bu kod C:\WINDOWS altında deneme isimli bir klasör oluşturur.

```
Directory.CreateDirectory("deneme");
```

Bu kod programın çalıştığı klasörde deneme isimli bir klasör oluşturur.

```
Directory.CreateDirectory(@"..\deneme");
```

Bu kod programın çalıştığı klasörün bir üst klasöründe deneme isimli bir klasör oluşturur.

```
Directory.CreateDirectory(@"..\..\deneme");
```

Bu kod programın çalıştığı klasörün iki üst klasöründe deneme isimli bir klasör oluşturur. ..\ sayıları bu şekilde artırılabilir. Bu tür bir adres belirtme şekli bütün diğer metotlarda da geçerlidir. Ayrıca bu ve diğer bütün metotlarda da adres diye tarif ettiğimiz veriye dosya / klasörün adı da dâhildir.

void Delete(string adres)

Belirtilen adresteki boş klasörü silmek için kullanılır. Başka bir kullanımı daha vardır.

void Delete(string adres,bool a)

Bu metot ile eğer a true ise belirtilen adresteki klasör, içindeki bütün dosya ve klasörlerle birlikte silinir.

bool Exists(string adres)

Belirtilen adresteki klasörün olup olmadığını bool cinsinden tutar. Klasör varsa true, yoksa false döndürür.

string GetCurrentDirectory()

Çalışan programın hangi klasörde olduğunu verir. Örneğin Windows'taysak C:\WINDOWS'u tutar.

string[] GetDirectories(string adres)

Belirtilen adresteki bütün klasörleri adresleriyle birlikte bir string dizisi olarak tutar.

string GetDirectoryRoot(string adres)

Belirtilen adresteki klasörün kök dizin bilgisini verir. Örneğin adres C:\Program Files\CONEXANT ise C:\ değerini döndürür.

string[] GetFiles(string adres)

Belirtilen adresteki dosyaları adresleriyle birlikte string dizisi olarak tutar. Bu ve benzer metotlarda liste İngilizce alfabetik sırasına göredir. GetFiles() metodunun bir prototipi daha vardır:

string [] GetFiles(string adres, string dosya)

Adresteki dosya(lar) adresleriyle birlikte string dizisi olarak tutulur. Dosya isminde joker karakterleri (*, ?) kullanılabilir.

string [] GetFileSystemEntries(string adres)

Belirtilen adresteki bütün dosya ve klasörleri adresleriyle birlikte bir string dizisi olarak tutar.

DateTime GetLastAccessTime(string adres)

Belirtilen adresteki dosya ya da klasöre en son ne zaman erişildiğini DateTime türünden tutar.

DateTime GetLastWriteTime(string adres)

Belirtilen adresteki dosya ya da klasörün en son ne zaman değiştirildiğini DateTime türünden tutar.

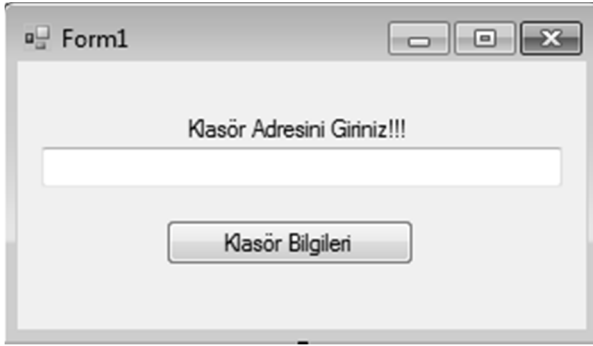
DateTime GetCreationTime(string adres)

Belirtilen adresteki dosya ya da klasörün ne zaman oluşturulduğunu DateTime türünden tutar.

DirectoryInfo sınıfı

DirectoryInfo sınıfı Directory sınıfının aksine static olmayan metot ve özellikleri içerir. Önce özellikleri bir örnek üzerinde görelim:

Uygulama: Girilen Klasör bilgilerini veren program.



```
private void button1_Click(object sender, EventArgs e)
{
    string adres = @textBox1.Text;
    DirectoryInfo d = new DirectoryInfo(adres);
    if (d.Exists) // Dosyanın var olup olmadığı kontrol ediyor
    {
        MessageBox.Show("Özellikler: " + d.Attributes +
            "\n" + "Oluşturulma tarihi: " + d.CreationTime +
            "\n" + "Uzanti: " + d.Extension +
            "\n" + "Tam adres: " + d.FullName +
            "\n" + "Son erişim zamanı: " + d.LastAccessTime +
            "\n" + "Son değişiklik zamanı: " + d.LastWriteTime +
            "\n" + "Klasör adı: " + d.Name +
            "\n" + "Bir üst klasör: " + d.Parent +
            "\n" + "Kök dizin: " + d.Root );
    }
    else
        MessageBox.Show("Belirtilen Klasör
        Bulunamıyor!!!");
}
```

Şimdi de DirectoryInfo sınıfının metotlarına geçelim. Bu metotların tamamı static değildir. Bu metotların çalışması için gereken adres

bilgisi, kendisine ulaşılması için kullanılan DirectoryInfo nesnesindedir.

void Create()

Klasör oluşturur.

DirectoryInfo CreateSubdirectory(string adres)

Belirtilen adreste bir alt dizin oluşturur. Örneğin C:\deneme altında \deneme2\deneme3 dizini oluşturmak için şu kodları yazarız.

```
string adres=@"C:\deneme";  
DirectoryInfo d=new DirectoryInfo(adres);  
d.Create();  
DirectoryInfo alt=d.CreateSubdirectory("deneme2");  
alt.CreateSubdirectory("deneme3");
```

Gördüğümüz gibi CreateSubdirectory metodu kendisine ulaşılan nesne içinde parametredeki klasörü oluşturuyor ve oluşturduğu klasörü de DirectoryInfo nesnesi olarak döndürüyor.

Delete

İki farklı aşırı yüklenmiş versiyonu vardır.

void Delete()

void Delete(bool a)

Birincisinde klasör boşsa silinir, ikincisinde a true ise klasör, içindeki her şeyle silinir.

DirectoryInfo[] GetDirectories()

İlgili klasörde bulunan bütün dizinleri bir DirectoryInfo dizisinde tutar

FileInfo Sınıfı

Bu sınıfa ait benzer metod ve özellikler vardır. İşlemleri dosyalar üzerinde gerçekleştirir.

Uygulama: Seçilen dosya bilgilerini gösteren program.

```
private void button1_Click(object sender, EventArgs e)
{
    openFileDialog1.ShowDialog();
    string adres = openFileDialog1.FileName;
    FileInfo d = new FileInfo(adres);
    MessageBox.Show( "Öznitelikler: " + d.Attributes +
        "\n" + "Oluşturulma tarihi: " + d.CreationTime +
        "\n" + "Var mı? " + d.Exists +
        "\n" + "Uzanti: " + d.Extension +
        "\n" + "Tam adres: " + d.FullName +
        "\n" + "Son erişim zamanı: " + d.LastAccessTime +
        "\n" + "Son değişiklik zamanı: " + d.LastWriteTime
        + "\n" + "Boyut: " + d.Length +
        "\n" + "Dosya adı: " + d.Name +
        "\n" + "Bulunduğu klasör: " + d.DirectoryName);
}
```

Path Sınıfı

Path sınıfı çeşitli işlemler yapan static üye elemanlara sahiptir.

Uygulama: Klasör bilgilerini konsol ekranında gösteren program

```
string adres=@"C:\dizin\deneme.txt";
Console.WriteLine("Uzanti: "+Path.GetExtension(adres));
string yeniAdres=Path.ChangeExtension(adres,"jpg");
Console.WriteLine("Klasör: "+Path.GetDirectoryName(adres));
Console.WriteLine("Dosya adı: "+Path.GetFileName(adres));
Console.WriteLine("Tam adres: "+Path.GetFullPath(adres));
```

```
Console.WriteLine("Kök dizin: "+Path.GetPathRoot(adres));
Console.WriteLine("Geçici dosya adı: "+Path.GetTempFileName());
Console.WriteLine("Geçici dosya dizini: "+Path.GetTempPath());
Console.WriteLine("Dizin ayırıcı: "+Path.DirectorySeparatorChar);
Console.Write("Geçersiz dosya adı karakterleri: ");
char[] dizi=Path.GetInvalidFileNameChars();
foreach(char b in dizi)
Console.Write(b+" ");
Console.Write("\nGeçersiz adres karakterleri: ");
char[] dizi2=Path.GetInvalidPathChars();
foreach(char b in dizi)
Console.Write(b+" ");
Console.WriteLine("\nAdres ayırıcı karakter:
                    "+Path.PathSeparator);
Console.WriteLine("Kök dizin ayırıcı:
                    "+Path.VolumeSeparatorChar);
```

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.

Uygulama gelişim ihtiyacı arttıkça, yeni uygulamalarda Web uygulama modeline olan bağıllık gittikçe azalmaktadır. Şimdilerde ise ağ bağlantıları üzerinden veriyi rahatça aktarabilmek için XML kullanımına olan yönelim artmaktadır. İşte ADO.NET, XML ve ADO.NET'in .NET Framework içinde en uygun şekilde programlama ortamı oluşturmamızı sağlar.

ADO.NET nesne modeli iki ana bölümden oluşmaktadır.

- DataSet Sınıfları
- .NET Veri Sağlayıcı Sınıfları

DataSet sınıfları, çevrimdışı ortamlar için veri depolama ve yönetme işlemlerini sağlar. DataSet sınıfları veri kaynağından bağımsız her tür uygulama ve veri tabanı için kullanılabilir. Özellikle İlişkisel Veri tabanı, XML ve XML Web servisleri üzerinden veri çekmek için kullanılır.

.NET veri sağlayıcı sınıfları, farklı türdeki veri tabanlarına 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

Hangi veri kaynağı kullanılacaksa, sadece ona uygun veri sağlayıcı sınıfı kullanılmalıdır.

ADO.NET Veri Sağlayıcıları:

.NET veri sağlayıcıları, ADO.NET mimarisinin veri tabanı ile uygulama (Windows, Web) veya XML Web Servis arasında bağlantı kurmak için her tür alt yapıyı barındıran çekirdek bileşendir. Tüm veri sağlayıcıları, **System.Data** isim alanı içinde tanımlanmıştır.

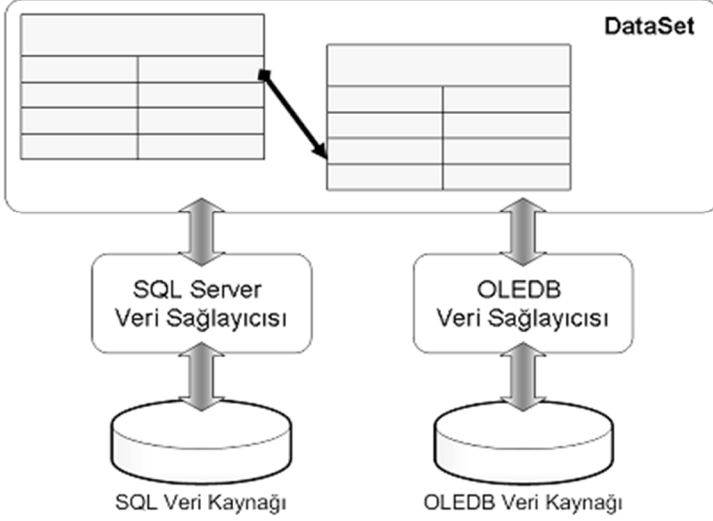
.NET Framework 1.0 sürümü ile birlikte SQL Server .NET ve OLE DB .NET veri sağlayıcı sınıfları gelmiştir.

- **SQL Server .NET:** SQL Server 7.0 ve SQL Server 2000 ve sonraki versiyonlara ait veri tabanlarına hızlı bağlantı sağlar. SQL Server bağlantı nesnelere **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 nesnelere **System.Data.OleDb** isim alanında bulunur.
NET Framework 1.1 sürümü ile birlikte SQL Server .NET ve OLE DB .NET veri sağlayıcılarına Oracle .NET ve ODBC .NET veri sağlayıcıları da eklenmiştir.
- **ORACLE .NET:** Oracle veri tabanlarına bağlantı için tasarlanmış veri sağlayıcısıdır. Oracle bağlantı nesnelere **System.Data.OracleClient** isim alanında bulunur.
System.Data.OracleClient isim alanını kullanmak için, projeye “System.Data.OracleClient.dll” referansı eklenmelidir.
- **ODBC .NET:** Diğer veri tabanlarını destekleyen genel bir veri sağlayıcıdır. ODBC bağlantı nesnelere **System.Data.Odbc** isim alanında bulunur.

Öğrenim ve kullanım kolaylığı olması amacıyla ADO.NET veri sağlayıcıların isimlendirilmesinde genelleştirmeye gidilmiştir. SQL

Server .NET veri sağlayıcılarının sınıf isimleri “Sql” ön eki ile, OLE DB .NET veri sağlayıcılarının sınıf isimleri ise “OleDb” ön eki ile başlar. Bu genellemeye “SqlConnection” ve “OleDbConnection” örnekleri verilebilir.

ADO.NET Veri Sağlayıcıları



Her bir veri sağlayıcısı içerisinde, birçok bağlantı nesnesi bulunur.

- Connection
- Command
- DataReader
- DataAdapter

XxxConnection: Veri kaynağına bağlantı için kullanılan sınıftır.

XxxCommand: Veri kaynağı üzerinde sorgu çalıştırmak için kullanılır. Veri kaynağından dönen kayıtlar XxxDataReader veya DataSet kullanılarak veri bağlantılı kontrollere aktarılır.

XxxDataReader: Çevrimiçi bağlantılarda sadece veri okumak için kullanılan sınıftır.

XxxDataAdapter: Çevrimdışı bağlantılarda kullanılan veri işleme nesnesidir.

Xxx yerine seçilen veri sağlayıcısına göre SQL, OLEDB, Oracle ve ODBC örneklerinden biri kullanılır.

Veri Kaynaklarına Bağlanmak

Veriyi yöneten uygulamalar, bu verilerin bulunduğu kaynağa bağlanma ihtiyacı duyar. Visual C# .NET ile veri kaynağına bağlanmak için, kaynağın tipine, yapısına göre farklı nesnelere ve farklı veri sağlayıcıları kullanılır.

Veri Sağlayıcı Seçmek

Veri Sağlayıcı: ADO.NET mimarisi, uygulama ile veri tabanı arasında bağlantı kurmak ve kurulan bağlantı üzerinden kayıtları almak, değiştirmek ve silmek için veri sağlayıcılarını kullanır. Farklı veri tabanları için farklı veri sağlayıcıları kullanılır.

Uygun veri sağlayıcı seçiminde en önemli kriter “Hangi sağlayıcı en iyi performansı verir?” sorusunun cevabıdır. Çünkü Sql Server, Oracle, Access gibi veri tabanlarına farklı veri sağlayıcıları ile erişilebilir.

Microsoft .NET Framework, veri tabanları ile bağlantı kurmak için farklı veri sağlayıcılarını destekler.

- SQL Server .NET
- OLEDB .NET
- ODBC .NET

Veri Sağlayıcı Sınıfları

.NET Framework içindeki veri sağlayıcıları, System.Data.dll içerisinde ki System.Data isim alanında yer alır. Yandaki resimde hangi sağlayıcı isim alanı ile hangi veri tabanına bağlanılabileceği gösterilmektedir.

Veri Tabanı	Veri Sağlayıcısı İsim Alanı
Sql Server 7.0 ve sonrası sürümler	System.Data.SqlClient
Sql Server 6.5 ve öncesi sürümler	System.Data.OleDb
Microsoft Access veri tabanı	System.Data.OleDb
Oracle Server	System.Data.OracleClient
Diğer veri tabanları	System.Data.OleDb

Class	Veri Kaynağı
System.Data.SqlClient.SqlConnection	SQL Server
System.Data.OleDb.OleDbConnection	OLE DB veri sağlayıcısı
System.Data.Odbc.OdbcConnection	ODBC veri sağlayıcısı
System.Data.OracleClient.OracleConnection	Oracle

ODBC .NET veri sağlayıcıları, diğer veri sağlayıcılarından farklı olarak, veri kaynağına bağlanırken hiçbir ara katman kullanmaz. Bunun yerine, bağlantı için ODBC API'leri kullanır.

.NET Framework veri sağlayıcıları aşağıdaki resimde belirtilen sınıfları kullanmaktadır. Sınıf isimlerinin önündeki XXX ön eki kullanılan veri sağlayıcı ismini simgeler. Eğer veri tabanına OLEDB veri sağlayıcısı ile bağlanılırsa OLEDB ön ekini, eğer SQL Server veri sağlayıcısı ile bağlanıyorsa SQL ön ekini alır.

Sınıf	Açıklama
XXXConnection	Bağlantı açmak ve kapatmak için kullanılan sınıftır.
XXXCommand	Veritabanı üzerinde Stored Procedure (Saklı Yordamlar) veya SQL Cümleleri çalıştırmak için kullanılan sınıftır.
XXXDataReader	Veritabanından sadece okunur ve ileri hareketli kayıtlar çekmek için kullanılan sınıftır.
XXXDataAdapter	Veritabanından çekilen verileri DataSet içerisine veya DataSet'e çevrimdışı eklenmiş verileri veritabanına aktarmak için kullanılan sınıftır

VERİ ORTAMLARI

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 ortamıdır.
- Veri kaynağına yapılan eş zamanlı erişimlerde, veri kaynağının kontrolünü kolaylaştırır.

Dezavantajları:

- Uygulama ile veri kaynağı arasında gerçekleşen bağlantıyı koruyabilmek için sabit bir ağ bağlantısının olması gerekir.
- Uygulama ile veri kaynağı arasındaki bağlantı ağ üzerinden gerçekleştiği için, ağ trafiğinin yoğunluğunu artırır.

Bağlantısız (Disconnected) Veri Ortamları

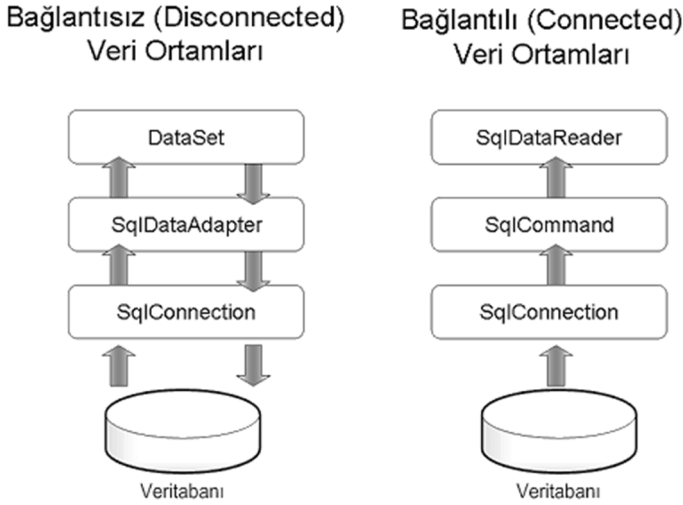
Bağlantısız veri ortamı, uygulamanın veri kaynağına sürekli bağlı kalmadığı veri ortamıdır. Uygulama ile veri kaynağı arasında bağlantı, veri alış verişi yapılırken açılır ve işlem bittikten sonra kapatılır. Bu veri ortamları çevrimdışı çalışmak için kullanılır.

Avantajları:

- Laptop, Notebook ve Pocket PC gibi araçlarla girilen veriler, istenilen zamanda veri ortamlarına aktarılabilir.
- Çevrimdışı ortamlar sayesinde, verilerin depolandığı uygulama üzerindeki yük hafifletilir. Bu durum performans artışını sağlar.

Dezavantajları:

- Bağlantısız veri ortamlarında, verilerin güncel kalmasına dikkat edilmelidir. Bu ortamlarda veri güncelleme işlemleri farklı zamanlarda gerçekleştirilebilir. Veri üzerinde yapılan bu değişimlerin, diğer kullanıcılara gösterilebilmesi için çeşitli çözümler geliştirilmelidir.
- Bağlantısız veri ortamları içerisinde farklı kullanıcılar eşzamanlı güncelleme işlemleri gerçekleştirebilir. Bu durumda oluşacak veri çakışmalarının engellenmesi gerekir.



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 sorgularının sonuçlarını geri döndürmek için kullanılan sınıftır.

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.

NOT: DataSet, System.Data isim alanı içerisinde yer alır.

System.Data.OleDb isim alanı içerisinde **çevrimiçi bağlantılar** geliştirmek için OleDbConnection, OleDbCommand, OleDbDataReader sınıfları kullanılır.

- **OleDbConnection**; Access veya diğer veri tabanları üzerinde bağlantı açmak ve kapatmak için kullanılan sınıftır.
- **OleDbCommand**; Access veya diğer veri tabanları üzerinde Stored Procedure (Saklı Yordamlar) veya SQL Cümleleri çalıştırmak için kullanılan sınıftır.
- **OleDbDataReader**; Access veya diğer veri tabanları üzerinde OleDbCommand ile çalıştırılan SELECT sorgularının sonuçlarını geri döndürmek için kullanılan sınıftır.

System.Data.OleDb isim alanı içerisinde **çevrimdışı bağlantılar** geliştirmek için OleDbConnection, OleDbDataAdapter sınıfları kullanılır.

- **OleDbConnection**; Access veya diğer veri tabanları üzerinde bağlantı açmak ve kapatmak için kullanılan sınıftır.
- **OleDbDataAdapter**; Access veya diğer veri tabanlarından çekilen verileri DataSet içerisine ve DataSet'e çevrimdışı eklenmiş verileri ilgili veri tabanına aktarmak için kullanılan sınıftır.

Bağlantı Cümlesi (Connection String) Oluşturmak

Bağlantı cümlesi, veri kaynağına bağlanmak için gerekli bilgileri tutar. Bu cümle, veri kaynağına bağlantı kurmak için gerekli bağlantı parametrelerin birleşiminden oluşur. Bu parametrelerin listesi aşağıda gösterilmiştir.

Parametre	Tanımı
Provider	Sadece OleDbConnection nesnelerinde kullanılır. Bağlantı sağlayıcısının ismini tutar. Sağlayıcı isimleri Tablo 2.2 de belirtilmiştir.
ConnectionTimeout veya Connect Timeout	Veritabanı bağlantı için beklenmesi gereken maksimum saniye sayısıdır. Varsayılan değer 15 saniye dir.
Initial Catalog	Veri tabanı adı
Data Source	SQL Server adı, veya MS Access veri tabanı için dosya adı
Password (pwd)	SQL Server login(giriş) parolası
User Id (uid)	SQL Server login(giriş) adı
Integrated Security veya Trusted Connection	SQL sunucusuna Windows hesabı ile bağlantı yapılacağını belirtir. True , False veya SSPI girilebilir. SSPI, True ile eş anlamlıdır ve bu durumda Windows hesabı kullanılır.

Persist Security Info	Varsayılan değeri False olur. Bu durumda güvenlik için hassas bilgileri geri döndürmez. True olduğunda ise güvenlik risk taşımaya başlar.
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.

Aşağıdaki örnekte SQL Server veri tabanı için bağlantı cümlesi oluşturulmuştur. HSKPB isimli sunucuda bulunan OgrenciDB veri tabanına, hsk kullanıcı ismi ve 1234 parolası ile bağlanılıyor. Eğer veri tabanı sunucusundan 60 saniye içinde cevap alamazsa bağlantı iptal ediliyor.

```
System.Data.SqlClient.SqlConnection bag;  
bag = new System.Data.SqlClient.SqlConnection( );  
bag.ConnectionString = "Data Source=HSKPB; Initial  
    Catalog=OgrenciDB; User ID=hsk;  
    Password=1234; Connection Timeout=60";
```

Aşağıdaki örnekte Microsoft Access 2003 veri tabanı için bağlantı cümlesi oluşturulmuştur. OleDb bağlantısı yapıldığı için Provider özelliğinin Microsoft.Jet.OleDB.4.0 olarak belirtilmesi gerekir. Bağlantının yapılacağı Personel veri tabanınının local makinede C:\Deneme dizini altında bulunduğu belirtiliyor.

```
System.Data.OleDb.OleDbConnection baglanti;  
baglanti = new System.Data.OleDb.OleDbConnection();  
baglanti.ConnectionString=@"Provider=Microsoft.Jet.OLEDB.4.0;  
    Data Source=C:\Deneme\Personel.mdb";
```

Aşağıdaki örnekte Sql Server 6.5 veri tabanı için bağlantı cümlesi oluşturulmuştur. SQL Server 7.0 sürümünden eski bir veri

tabanı sunucuna bağlantı yapıldığı için Provider özelliği SQLOLEDB olarak belirtiliyor. ProdServ isimli sunucudaki Pubs veritabanına, Windows hesabı (SSPI) ile bağlantılıyor.

```
System.Data.OleDb.OleDbConnection bag2;  
bag2 = new System.Data.OleDb.OleDbConnection();  
bag2.ConnectionString = "Provider=SQLOLEDB;  
                          Data Source=ProdServ; Initial Catalog=Pubs;  
                          Integrated security=SSPI";
```

Dikkat: Microsoft Access veri kaynağı, tek veri tabanından oluşur. SQL Server veri kaynağı ise birden fazla veri tabanından oluşur. Bu yüzden SQL Server veri tabanı bağlantı cümlesinde Initial Catalog parametresi kullanılır.

Örnek Ms Access ile OLEDB Bağlantı Cümleleri

(2003 ve önceki Versiyon)

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(Uzak Server) 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"

Örnek SQL Server ile SQL Server Bağlantı Cümleleri

SQL Server'a SQL Authentication ile bağlanmak	"Data Source=_Server_Name;Initial Catalog=Db_Name; User Id= Username;Password=sifreniz;"
SQL Server'a SQL Authentication ile bağlanmak	"Server= Server_Name;Database=Db_Name; UserID=Username; Password=sifreniz; Trusted_Connection=False"
SQL Server'a Windows Authentication ile bağlanmak	"Data Source= Server_Name;Initial Catalog=Db_Name;Integrated Security=SSPI;"
SQL Server'a SQL Authentication ile bağlanmak	"Server=Server_Name;Database=Db_Name; Trusted_Connection=True;"

Bağlantıyı Açmak ve Kapatmak

Bağlantı cümlesini oluşturduktan sonra, bağlantıyı açmak ve kapamak için Connection sınıfının iki önemli metodu kullanılır.

- Open
- Close

Open metodu, bağlantı cümlesinde belirtilen veri kaynağını açmak için kullanılır. Close metodu, açılan bağlantıyı kapatmak için kullanılır. Close metodu ile kullanılmayan bağlantıları kapatmak, kaynak tüketimini azaltır. Open metodu; uygulama ile veri kaynağı arasındaki bağlantıyı, bağlantı cümlesinin Timeout parametresinde belirtilen süre içerisinde kurmaya çalışır. Eğer belirtilen süre içerisinde bağlantı gerçekleşmiyorsa, uygulama hata üretir. Bu süre için herhangi bir değer belirtilmemişse, varsayılan değer 15 saniyedir.

```
bag.ConnectionString = @"Provider=Microsoft.Jet.OLEDB.4.0;  
Data Source=C:\Deneme\Personel.mdb";  
  
//Bağlantıyı açmak  
bag.Open( );  
  
//Veri tabanı işlemleri bu arada gerçekleştirilir.
```

```
//Bağlantıyı kapatmak  
bag.Close( );
```

Bağlantı Durumlarını Kontrol Etmek

Bağlantı sınıfının durumu hakkında bilgi almak için, bağlantı sınıfının State özelliği kullanılır.

State özelliğinin alabileceği değerler

İsim	Açıklama	Değeri
Broken	Yalnızca, açık bir bağlantının kopup tekrar bağlanıldığı durum	16
Closed	Bağlantı kapalı	0
Connecting	Veri kaynağına bağlanma aşamasında	2
Executing	Bağlantı üzerinden bir komutu çalıştırılıyor	4
Fetching	Bağlantı üzerinden veri çekiliyor	8
Open	Bağlantı açık	1

```
private void ConnectionAc(OleDb.OleDbConnection con)  
{  
    //Bağlantı, sadece kapalı ise açılacak  
    If (bag.State == ConnectionState.Closed)  
    {  
        bag.Open();  
    }  
}
```

Bağlantı nesnelerinin durumu değiştiği zaman StateChange olayı tetiklenir. Bu olay ile bağlantının hangi durumlarda açılıp kapandığı öğrenilebilir.

Property	Açıklama
CurrentState	Bağlantının yeni durumu hakkında bilgi verir.
OriginalState	Bağlantının değişmeden önceki durumu hakkında bilgi verir.

Command ile Çalışmak

XxxCommand, veritabanı üzerinde Stored Procedure (Saklı Yordam) ve Sorgu çalıştırmak için kullanılır. Command Nesneleri ile veri tabanı tablolarında; sorgu, ekleme, silme ve güncelleme işlemleri yapılabilir. Aşağıda hangi veri sağlayıcı için hangi Command Nesnesinin kullanıldığı gösterilmektedir.

Nesne	Veri Sağlayıcıları
System.Data.SqlClient.SqlCommand	SQL Server .NET Veri Sağlayıcısı
System.Data.OleDb.OleDbCommand	OLE DB .NET Veri Sağlayıcısı
System.Data.OleDb.OdbcCommand	ODBC .NET Veri Sağlayıcısı

Veritabanı üzerinde Stored Procedure ve Sorgu çalıştırmak için Command Nesnelerinin belirli özelliklerini kullanmak gerekir. Command Nesnelerinin bu özellikleri aşağıda belirtilmiştir.

- Name: Command nesnesinin kod içerisindeki ismidir. Bu isim Command nesnesine başvurmak için kullanılır.
- 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: Command içerisinde çalıştırılacak Stored Procedure veya Sorgu cümlesine, dışardan değer almak ve dışarıya değer göndermek için kullanılır.

Command özelliklerine değer girildikten sonra, Command'ı çalıştırmak için aşağıdaki metotlardan uygun olan seçilir.

Command Sınıfının Metodu	Açıklama
ExecuteScalar	Tek bir değer döndürecek sorguları çalıştırır.
ExecuteReader	Birkaç satır bilgiyi döndürecek sorguları çalıştırır.
ExecuteNonQuery	Veri güncelleme yapılacağında çalıştırılır ve bu güncellemeden etkilenen satırların sayısını geri döndürür.
ExecuteXmlReader	Sorgu sonucunu XML olarak verir. Sadece SqlCommand nesnesinde bulunur.

Uygulama: Access 2010 Veri Tabanı (Personel Veri Tabanı)

Veri Tabanı Yapısı

Görünümler		Araçlar	Göster/Gizle	Alan, Kayıt ve T
Tüm Access Nesneleri		Tablo1		
Ara...				
Tablolar				
Tablo1				
	Alan Adı	Veri Türü		
	PerID	Otomatik Sayı		
	ADI	Metin		
	SOYADI	Metin		
	MAAS	Sayı		

Form Görünümü

	PerID	ADI	SOYADI	MAAS
*				

```
using System;  
using System.Collections.Generic;  
using System.ComponentModel;  
using System.Data;  
using System.Drawing;  
using System.Linq;  
using System.Text;  
using System.Windows.Forms;  
using System.Data.OleDb;
```

```
namespace _1213BDDers07_01  
{  
    public partial class Form1 : Form  
    {  
        public Form1()  
        {
```

```

        InitializeComponent();
    }

// Bağlantı Cümlesi
OleDbConnection bag= new OleDbConnection
    (@"Provider=Microsoft.ACE.OLEDB.12.0;
    Data Source=C:\Users\HSK\Documents\personel.accdb");

// Formun Yüklenmesi
private void Form1_Load(object sender, EventArgs e)
{
    if (bag.State == ConnectionState.Closed)
        bag.Open();

    DataTable dt = new DataTable();
    OleDbDataAdapter da = new OleDbDataAdapter
        ("SELECT * FROM Tablo1", bag);
    da.Fill(dt);
    dataGridView1.DataSource = dt;
}

// Listele Butonu
private void button1_Click(object sender, EventArgs e)
{
    if (bag.State == ConnectionState.Closed)
        bag.Open();

    DataTable dt = new DataTable();
    OleDbDataAdapter da = new OleDbDataAdapter
        ("SELECT * FROM Tablo1", bag);
    da.Fill(dt);
    dataGridView1.DataSource = dt;
}

```

```

// Ekle Butonu
private void button2_Click(object sender, EventArgs e)
{
    if (bag.State == ConnectionState.Closed)
        bag.Open();

    OleDbCommand komut = new OleDbCommand
        ("INSERT INTO Tablo1(ADI,SOYADI,MAAS)
        VALUES('" + textBox1.Text + "','" + textBox2.Text +
        "','" + float.Parse(textBox3.Text) + "')", bag);
    komut.ExecuteNonQuery();

    DataTable dt = new DataTable();
    OleDbDataAdapter da = new OleDbDataAdapter
        ("SELECT * FROM Tablo1", bag);
    da.Fill(dt);
    dataGridView1.DataSource = dt;
}

```

```

// Sil Butonu
private void button3_Click(object sender, EventArgs e)
{
    if (bag.State == ConnectionState.Closed)
        bag.Open();

    OleDbCommand komut = new OleDbCommand
        ("DELETE FROM Tablo1
        WHERE ADI='" + textBox1.Text + "' AND
        SOYADI='" + textBox2.Text + "'", bag);
    komut.ExecuteNonQuery();

    DataTable dt = new DataTable();
    OleDbDataAdapter da = new OleDbDataAdapter
        ("SELECT * FROM Tablo1", bag);
    da.Fill(dt);
}

```

```

        dataGridView1.DataSource = dt;
    }

    // Güncelle Butonu
    private void button4_Click(object sender, EventArgs e)
    {
        if (bag.State == ConnectionState.Closed)
            bag.Open();

        OleDbCommand komut = new OleDbCommand
            ("UPDATE Tablo1 SET MAAS=" +
            float.Parse(textBox3.Text) + "
            WHERE ADI='" + textBox1.Text + "' AND
            SOYADI='" + textBox2.Text + "'", bag);
        komut.ExecuteNonQuery();

        DataTable dt = new DataTable();
        OleDbDataAdapter da = new OleDbDataAdapter
            ("SELECT * FROM Tablo1", bag);
        da.Fill(dt);
        dataGridView1.DataSource = dt;
    }

    // Ara Butonu (Adı'na göre)
    private void button5_Click(object sender, EventArgs e)
    {
        if (bag.State == ConnectionState.Closed)
            bag.Open();

        DataTable dt = new DataTable();
        OleDbDataAdapter da = new OleDbDataAdapter
            ("SELECT * FROM Tablo1
            WHERE ADI='" + textBox1.Text + "'", bag);
        da.Fill(dt);
    }

```

```

        dataGridView1.DataSource = dt;
    }

    // Formun Kapanması
    private void Form1_FormClosing(object sender,
    FormClosingEventArgs e)
    {
        bag.Close();
    }

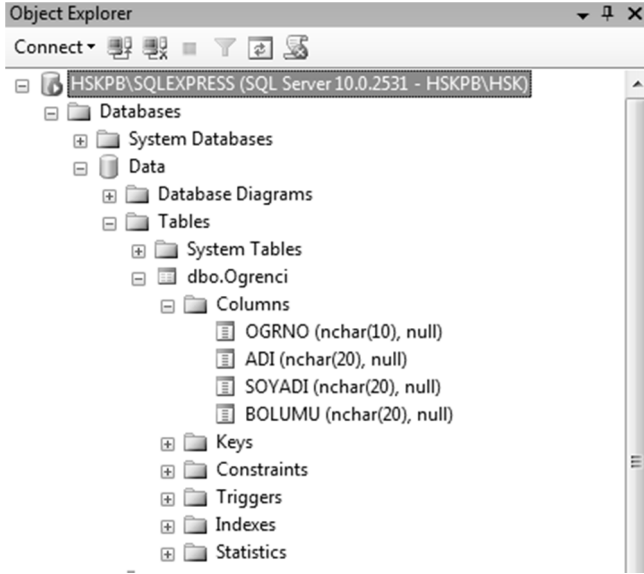
    // Datagridde seçili değerin değışmesi
    private void dataGridView1_SelectionChanged(object sender,
    EventArgs e)
    {
        textBox1.Text = dataGridView1.CurrentRow.Cells[1].Value.ToString();
        textBox2.Text = dataGridView1.CurrentRow.Cells[2].Value.ToString();
        textBox3.Text = dataGridView1.CurrentRow.Cells[3].Value.ToString();
    }

    // Kayıt Ekleme için başka bir yöntem!
    string sorgu = "INSERT INTO Personel
        (Ad, Soyad, PNO, TCNO, Unvan, Bolum, Cinsiyet, Dogum)
    VALUES (@ad,@soyad,@pno,@tcno,@unvan,@bolum,@cinsiyet,
        @dogumtarihi)";
    OleDbCommand cmd= new OleDbCommand(sorgu,baglanti);
    cmd.AddWithValue.Add("@ad",txtad.Text);
    cmd.AddWithValue.Add("@soyad",txtsoyad.Text);
    cmd.AddWithValue.Add("@pno",txtpno.Text);
    .....
    baglanti.open();
    cmd.ExecuteNonQuery();
    baglanti.close();
    }
}

```

Uygulama: SQL Server Veri Tabanı (Data Veri Tabanı)

Veri tabanı yapısı



Form Görünümü

Form1

Öğrenci No

Adı

Soyadı

Bölümü

Listele Ekle Güncelle

Sil Ara

	OGRNO	ADI	SOYADI	BOLUMU
*				

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace _1213BDDers08_04
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        // Bağlantı Cümlesi
        SqlConnection bag = new SqlConnection(
            @"Data Source=.\SQLEXPRESS; Initial Catalog=Data;
            Integrated Security=True");

        // Bağlan Metodu
        void baglan()
        {
            if (bag.State == ConnectionState.Closed)
                bag.Open();
        }

        // Listele Metodu
        void listele()
        {
            DataTable dt = new DataTable();

```



```

SqlDataAdapter da = new SqlDataAdapter
    ("SELECT * FROM Ogrenci", bag);

da.Fill(dt);

dataGridView1.Columns[0].HeaderText = "Öğrenci No";
dataGridView1.Columns[1].HeaderText = "Adı";
dataGridView1.Columns[2].HeaderText = "Soyadı";
dataGridView1.Columns[3].HeaderText = "Bölümü";

dataGridView1.DataSource = dt;

}

// Listele Butonu
private void button1_Click(object sender, EventArgs e)
{
    baglan();
    listele();
}

// Formun Yüklenmesi
private void Form1_Load(object sender, EventArgs e)
{
    baglan();
    listele();
    dataGridView1.SelectionMode =
        DataGridViewSelectionMode.FullRowSelect;
}

// Ekle Butonu
private void button2_Click(object sender, EventArgs e)
{
    baglan();

```

```

SqlCommand komut = new SqlCommand();
komut.Connection = bag;
komut.CommandText = "INSERT INTO
    Ogrenci(OGRNO, ADI, SOYADI, BOLUMU)
    VALUES ('" + textBox1.Text + "','" + textBox2.Text +
        "','" + textBox3.Text + "','" + textBox4.Text + "')";
komut.ExecuteNonQuery();

listele();

bag.Close();
}

// Güncelle Butonu
private void button3_Click(object sender, EventArgs e)
{
    baglan();

    SqlCommand komut = new SqlCommand();
    komut.Connection = bag;
    komut.CommandText = "UPDATE Ogrenci
        SET ADI='" + textBox2.Text + "', SOYADI='" +
        textBox3.Text + "', BOLUMU='" + textBox4.Text + "'
        WHERE OGRNO='" + textBox1.Text + "'";
    komut.ExecuteNonQuery();

    listele();
    bag.Close();
}

// Sil Butonu
private void button4_Click(object sender, EventArgs e)
{
    baglan();

```

```

SqlCommand komut = new SqlCommand();
komut.Connection = bag;
komut.CommandText = "DELETE FROM Ogrenci
    WHERE OGRNO='" + textBox1.Text + "'";
DialogResult s;
s = MessageBox.Show(textBox1.Text + " Numaralı Öğrenci
    Silinsin mi", "Silme İşlemi",
    MessageBoxButtons.YesNo);
if (s == DialogResult.Yes)
{
    if (komut.ExecuteNonQuery() >= 1)
    {
        listele();
        MessageBox.Show("Kayıt Silindi");
    }
    else
        MessageBox.Show(textBox1.Text + "
        Numaralı Öğrenci Sistemde Kayıtlı Değil!!!");
}
else
    MessageBox.Show("Kayıt Silinmedi");

bag.Close();
}

// Ara Butonu (Öğrenci No'ya göre)
private void button5_Click(object sender, EventArgs e)
{
    baglan();

    DataTable dt = new DataTable();
    SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM
        Ogrenci WHERE OGRNO='" + textBox1.Text + "'",
        bag);
    da.Fill(dt);
}

```

```

        dataGridView1.DataSource = dt;
    }

    // Data Gridde Seçili Değerin Değişmesi
    private void dataGridView1_SelectionChanged(object sender,
        EventArgs e)
    {
        textBox1.Text = dataGridView1.CurrentRow.Cells[0].Value.ToString();
        textBox2.Text = dataGridView1.CurrentRow.Cells[1].Value.ToString();
        textBox3.Text = dataGridView1.CurrentRow.Cells[2].Value.ToString();
        textBox4.Text = dataGridView1.CurrentRow.Cells[3].Value.ToString();
    }
}

```

Örnek Uygulama:

Veri Tabanı Yapısı:



Tablolar		
Ara...		
	Musteriler	
	Siparisler	
	Urunler	

Musteriler		
	Alan Adı	Veri Türü
	MusID	Otomatik Sayı
	MUS_ADI	Metin
	ADRES	Metin
	TELNO	Metin

Musteriler Siparisler Urunler		
	Alan Adı	Veri Türü
🔑	SipID	Otomatik Sayı
	URUN_ADI	Metin
	MUS_ADI	Metin
	ADET	Sayı
	FIYAT	Sayı

Musteriler Siparisler Urunler		
	Alan Adı	Veri Türü
🔑	UrunID	Otomatik Sayı
	URUN_ADI	Metin
	BIRIM	Metin
	FIYAT	Sayı

Form Görünümleri:

Giriş Ekranı

Kullanıcı Adı

Parola

Ana Ekran

Ürünler Formu

Ekle Düzenle Sil Ara

Ürün Adı

Birimi

Fiyatı

	UrunID	URUN_ADI	BIRIM	FIYAT
*				

Müşteri Formu

Ekle Düzenle Sil Ara

Müşteri Adı

Adres

Tel No

	MusID	MUS_ADI	ADRES	TELNO
*				

Siparişler

Sipariş Ekle Ürün Ekle Müşteri Ekle

Ürün

Müşteri

Adet

	SipID	URUN_ADI	MUS_ADI	ADET	FIYAT
*					

RAPORLAMA

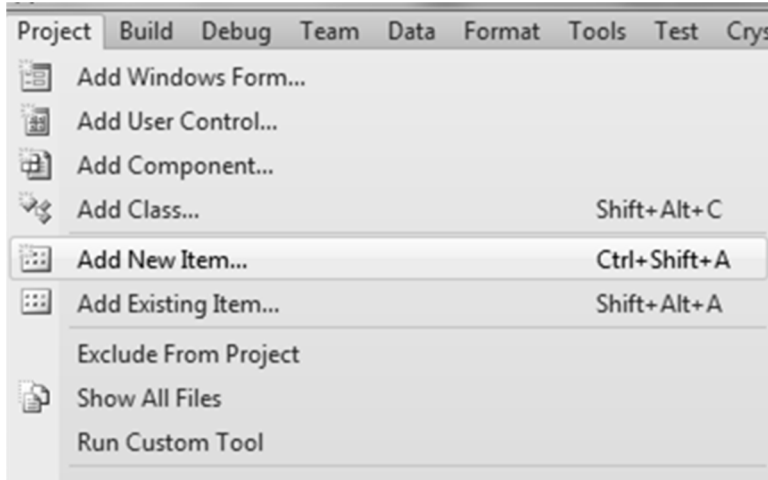
Uygulamalarda programcıya sağladığı bir çok özellik nedeniyle en çok tercih edilen raporlama sistemi Crystal Report nesnesidir.

Temel rapor işlemleri sırasıyla;

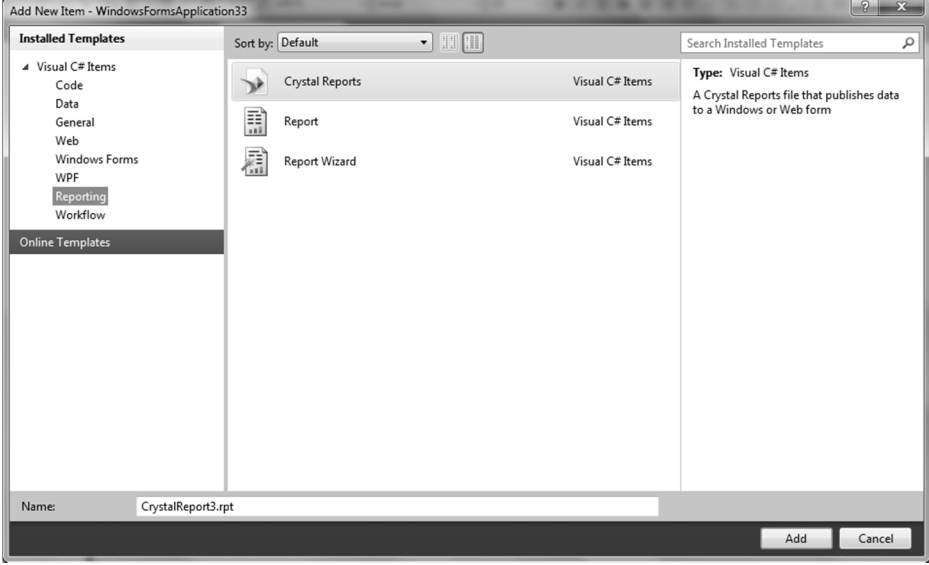
- Rapor tasarımı yapılır
- Raporun görüntüleneceği forma CrystalReportViewer kontrolü eklenir.
- CrystalReportViewer kontrolünün ReportSource özelliğinde tasarımı yapılan rapor nesnesi seçilir.

Raporlama İşlemleri

Rapor oluşturabilmek için öncelikle projeye yeni bir nesne, rapor nesnesi eklenmelidir.

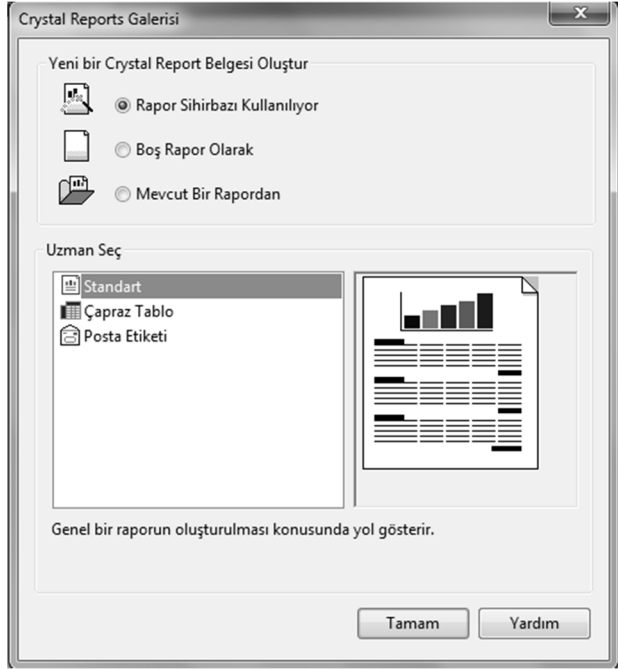


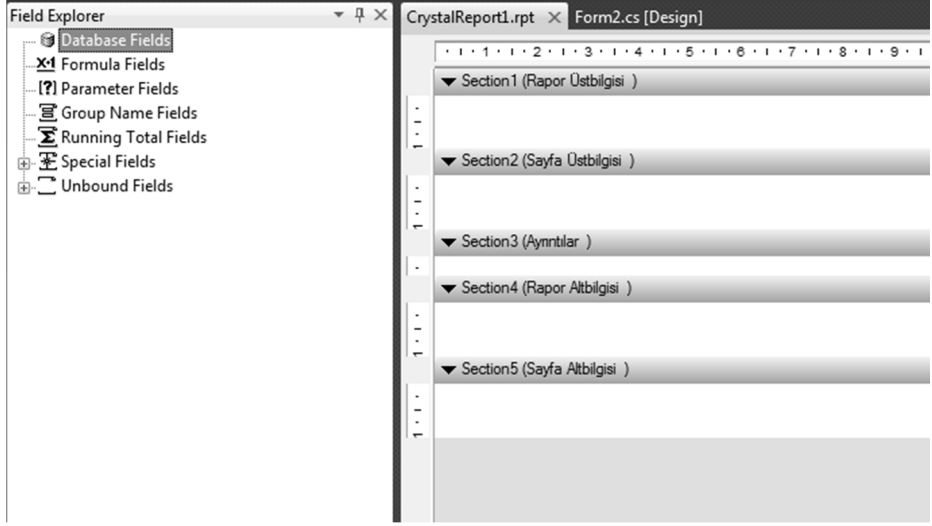
Project menüsünden Add New Item Seçilir. Karşımıza gelen ekranda Templates kısmından Reporting seçilir ve Crystal Reports seçilip Add tıklanır.



Karşımıza gelen
Crystal Report
Başlangıç ekranından
uygun seçeneği
seçmeliyiz.

Boş Rapor
seçildikten sonra
karşımıza tasarımı
yapmamız gereken
boş bir rapor ekranı
gelecektir.





Bu sayfada;

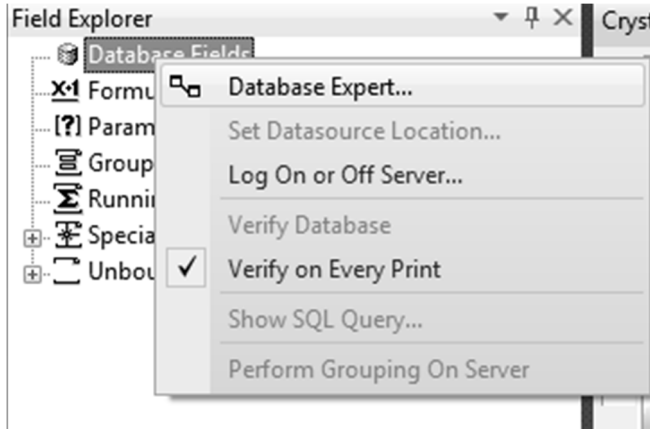
Rapor Bölümü

- **Report Header(Rapor Üstbilgisi)** bölümü raporunuzun başlangıç sayfasında gözükmelerini istediğiniz yazı ve bilgiler için kullanılır .
- **Page Header(Sayfa Üstbilgisi)** bölümü raporunuzda her sayfanın üstünde gözükmelerini istediğiniz bilgiler için kullanılır.
- **Details(Ayrıntılar)** bölümü raporunuzdaki bilgilerin bulunduğu bölümdür.
- **Report Footer(Rapor Altbilgisi)** bölümü raporunuzun son sayfasında gözükmelerini istediğiniz bilgiler için kullanılır.
- **Page Footer(Sayfa AltBilgisi)** bölümü raporunuzun her sayfasının sonunda gözükmelerini istediğiniz bilgiler için kullanılır.

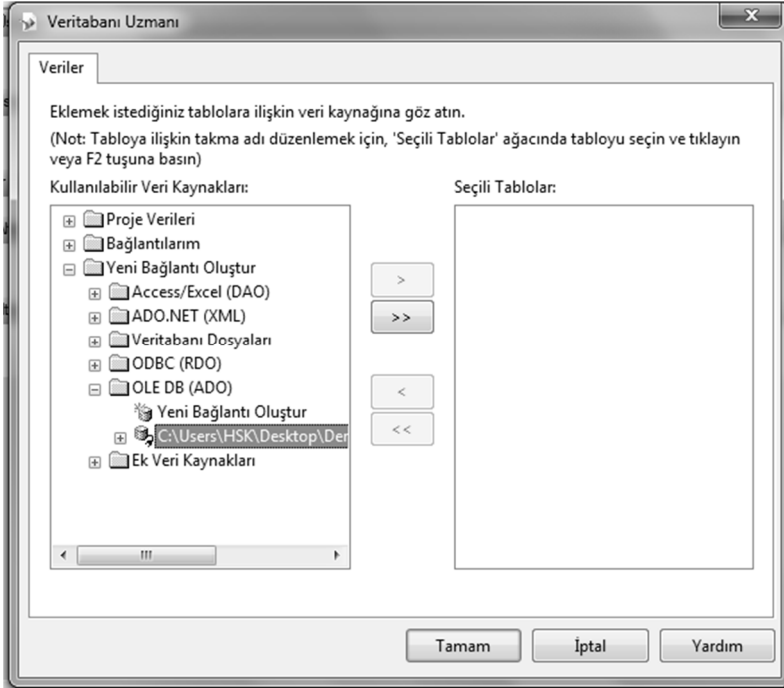
Fields Explorer Menüünde:

- **Database Fields** seçeneği veri tabanı bağlantılarının oluşturulduğu bölümdür.

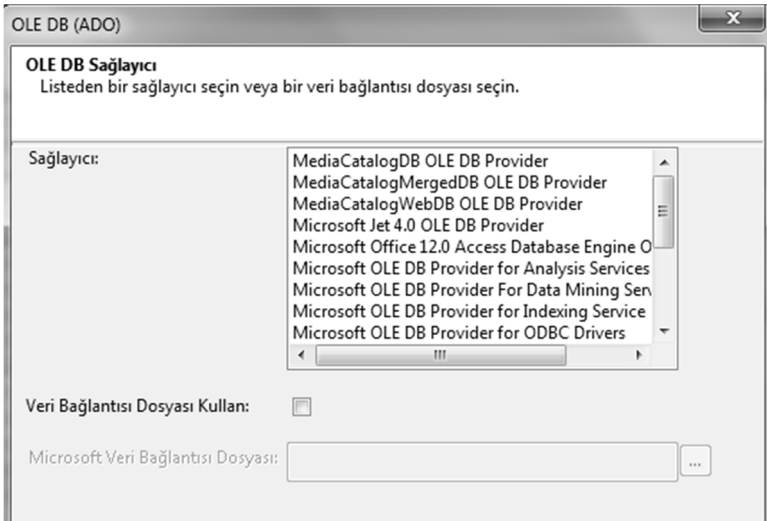
- **Formul fields** seçeneği duruma özel formüllerin oluşturulduğu bölümdür.(alt toplamlar,ilçe il birleştirme gibi)
- **Parameter Fields** seçeneği bazı durumlarda parametreyle çalışmak zorunda kalabilirsiniz bu durumlarda bu bölüm kullanılır (örn: stored procedure kullanılan raporlarda)
- **Running Fields** seçeneği hesaplanacak toplam alanların belirleneceği bölümdür.
- **Grup Name Fields** seçeneği verilerimizi belli kriterlere göre gruplamayı sağlar.
- **Special Fields** seçeneğinde Crystal Reportun hazır functionları bulunmaktadır.
- **Unbound Fields** seçeneği yeni eklenecek olan değer tiplerinin bulunduğu alandır.(Date,Datetime,Boolean vb.)



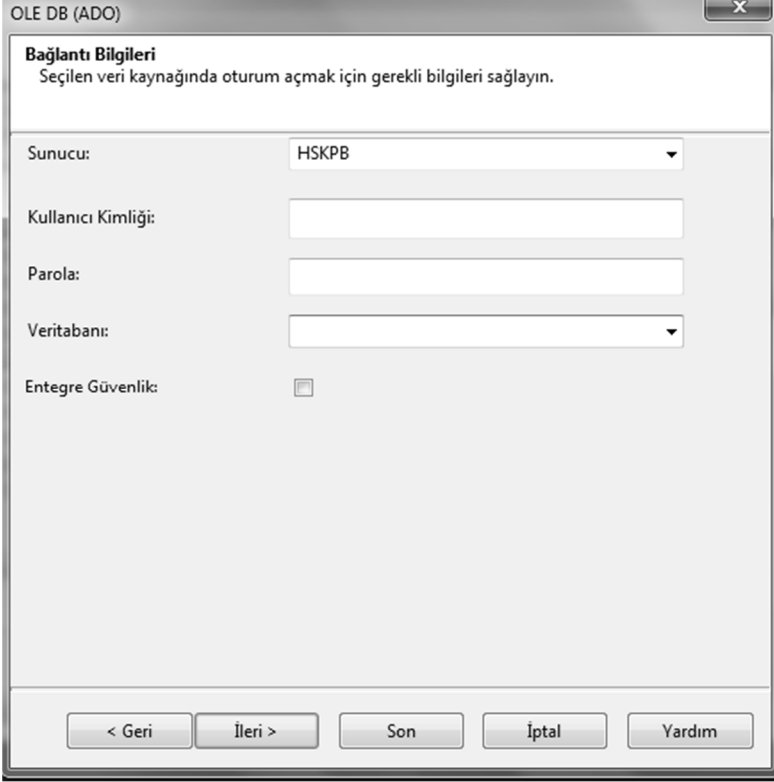
Field Explorer menüsünde Database Fields seçeneğinin üzerinde sağ tıklayalım ve açılan menüden " Database Expert " seçeneğini seçelim .



Karşımıza gelen bu pencerede yeni bir bağlantı yapacağımız için Create New Connection(Yeni Bağlantı Oluştur) seçeneğinin artısını tıklayalım oradan da " OLEDB (ADO) " seçeneğini seçelim,



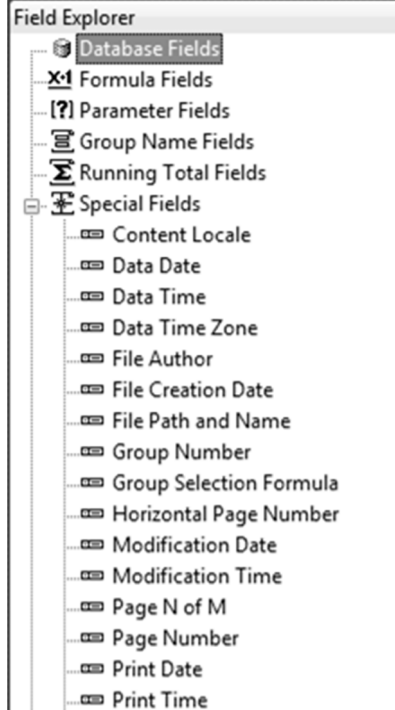
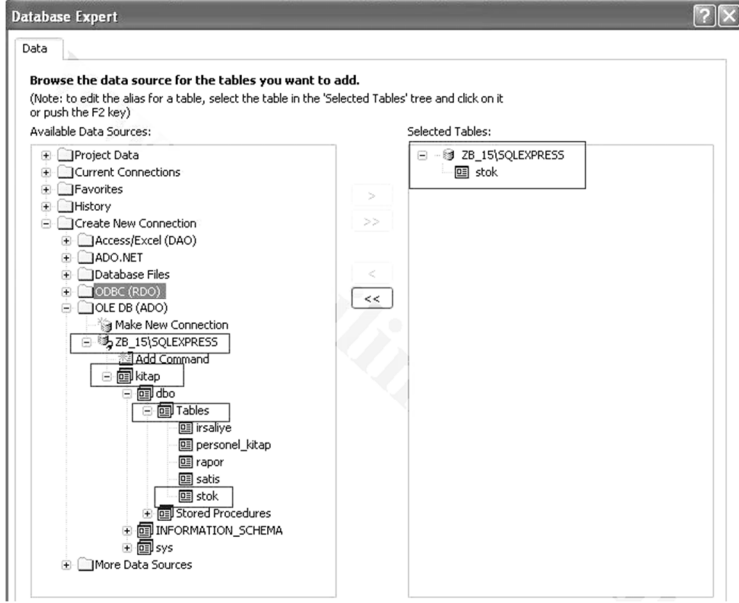
Açılan pencerede hangi veri tabanına bağlanılacaksa o bağlantı türünü seçelim (biz SQLSERVER bağlanacağımız için “Microsoft OLE DB Provider for SQL Server”) ve NEXT butonuna basalım,



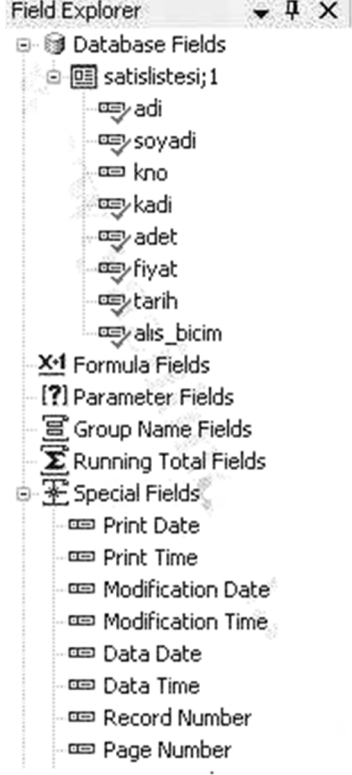
Bu bölümün Server Name kısmına server ismimizi yazalım. SQLSERVER ' a Windows Authentication olarak bağlanıyorsanız integrated security seçeneğini “true “, SQL Authentication olarak bağlanıyorsanız integrated security seçeneğini “ false” yapın ve User ID,Password kısımlarını doldurun. Database kısmından da bağlanmak istediğiniz database seçelim ve Finish butonuna basalım.

Açılan pencerede artık bağlantımız gelmiş olacak. Bundan sonra yapmamız gereken istediğimiz tabloları sağ tarafa yollamak ve tamam butonuna tıklamak. Artık veri tabanındaki tablomuz Crystal Reporta

bağlanmış oldu. Tablomuz Field Explorer penceresinde sütunları ile birlikte görünecektir.



Report Header kısmına başlık eklemekle başlayalım. Örneğin sağ üst köşeye tarih. Special Fields sekmesinden " Print Date " i sürükleyip Report Header kısmına bırakalım

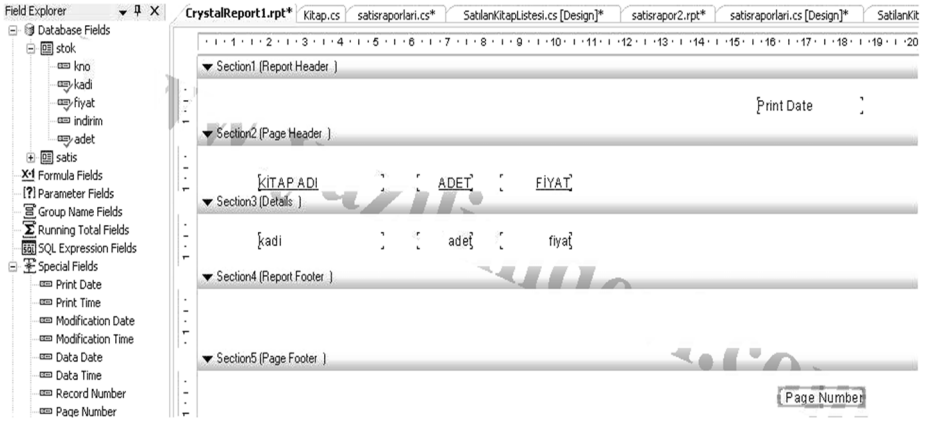


Page Header kısmına göstermek istenilen sütunların başlıklarını ekleyelim. Örneğin Kitap Adı, Adet, Fiyat,

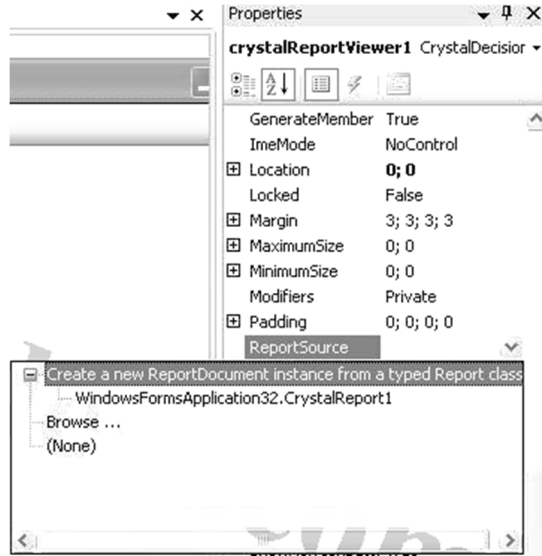
Details kısmına göstermek istenilen sütunları ekleyelim. Örneğin kadi,adet,fiyat...

Page Footer kısmına ise sayfa sonunda sayfa numarası gösterelim.

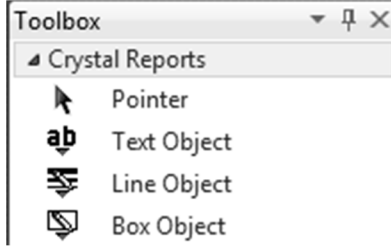
Raporu tasarımının son hali



Rapor tasarımını tamamlayıp kaydettikten sonra formumuza Crystal Report Viewer Kontrolünü ekleyip bu kontrolün properties panelindeki Report Source özelliğinden tasarımını yapıp kaydettiğimiz Crystal Raporumuzu seçelim.



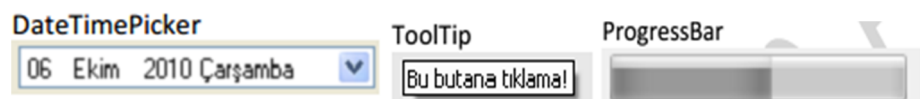
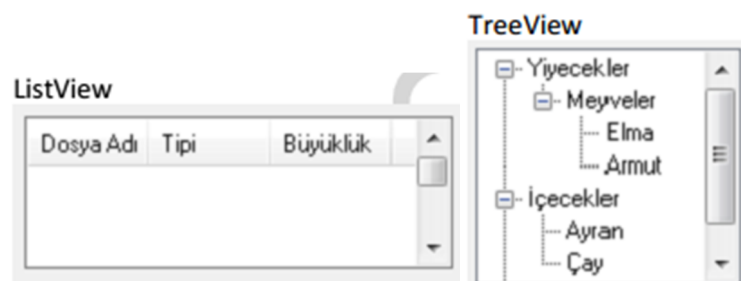
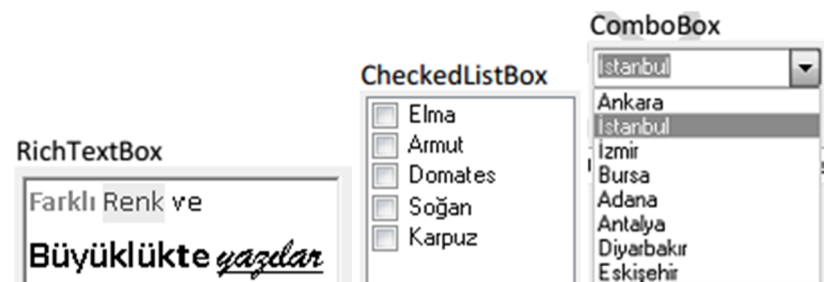
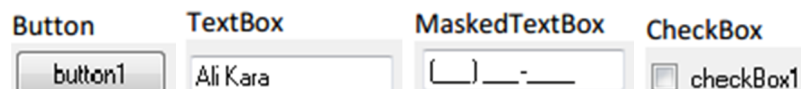
Crystal Report Toolbox Araçları



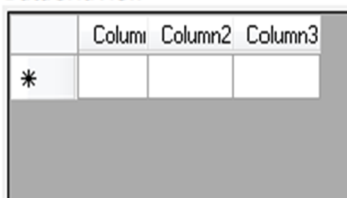
Rapor tasarım ekranında bu toolbox kontrolleri rapora eklenip kullanılabilir.

Referanslar 1: Genel Kontroller

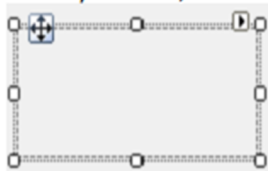
No	Adı	Araç Çubuğu Görünümü	Açıklama
1	Button	 Button	Tıklama butonu
2	TextBox	 TextBox	Metin kutusu
3	MaskedTextBox	 MaskedTextBox	Girişi maskelenebilir metin kutusu
4	RichTextBox	 RichTextBox	Zengin metin kutusu
5	CheckBox	 CheckBox	İsteğe bağlı Seçim kutucuğu
6	CheckedListBox	 CheckedListBox	İsteğe bağlı seçim kutucukları listesi
7	RadioButton	 RadioButton	Birden fazla seçenekten sadece birini seçme kutusu
8	ComboBox	 ComboBox	Açılır kutu, içlerinden bir tanesi seçilir
9	Label	 Label	Etiket
10	LinkLabel	 LinkLabel	Link etiket
11	NumericUpDown	 NumericUpDown	Artırılıp azaltılabilen sayı kutusu
12	PictureBox	 PictureBox	Resim kutusu
13	ListBox	 ListBox	Liste kutusu
14	ListView	 ListView	Liste görünümü
15	TreeView	 TreeView	Ağaç görünümlü liste
16	DateTimePicker	 DateTimePicker	Tarih Seçici
17	MonthCalendar	 MonthCalendar	Aylık takvim
18	NotifyIcon	 NotifyIcon	Uyarı iconu
19	ProgressBar	 ProgressBar	İlerleme kutusu
20	ToolTip	 ToolTip	Açıklama kutusu
21	DataGridView	 DataGridView	Veri tabanı uygulamalarında kullanılan veri tablosu
22	FlowLayoutPanel	 FlowLayoutPanel	Kontrolleri zorunlu sıraya koyan panel
23	GroupBox	 GroupBox	Kontrolleri Gruplama kutusu
24	Panel	 Panel	Panel
25	SplitContainer	 SplitContainer	Genişletilebilir/daraltılabilir panel
26	TabControl	 TabControl	Sekme kutusu
27	TableLayoutPanel	 TableLayoutPanel	Tablo şeklinde panel
28	ContextMenuStrip	 ContextMenuStrip	Şağ tıklanınca açılan menü
29	MenuStrip	 MenuStrip	Normal Menü
30	StatusStrip	 StatusStrip	Durum Çubuğu
31	ToolStrip	 ToolStrip	Araç çubuğu
32	ToolStripContainer	 ToolStripContainer	Araç çubuğu sıralayıcı
33	ColorDialog	 ColorDialog	Renk seçme kutusu
34	FolderBrowserDialog	 FolderBrowserDialog	Klasör Seçme kutusu
35	FontDialog	 FontDialog	Font Seçme kutusu
36	OpenFileDialog	 OpenFileDialog	Açmak için Dosya seçme kutusu
37	SaveFileDialog	 SaveFileDialog	Kaydetmek için dosya isim seçme kutusu



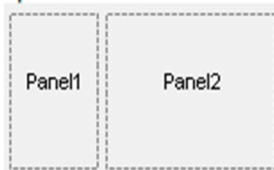
DataGridView



FlowLayoutPanel, Panel



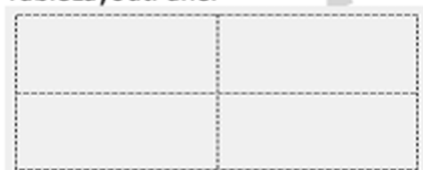
SplitContainer



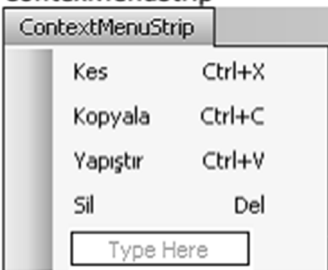
TabControl



TableLayoutPanel



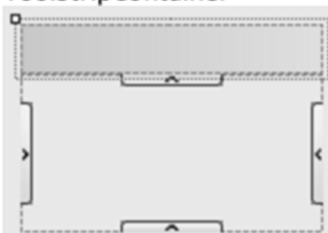
ContextMenuStrip



MenuStrip



ToolStripContainer



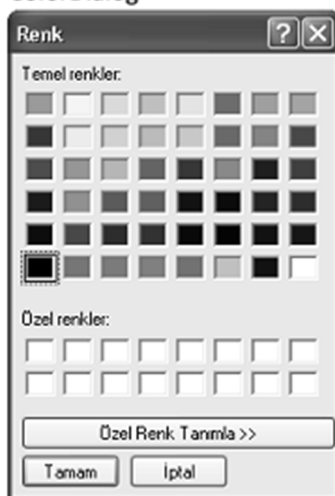
ToolStrip



StatusStrip



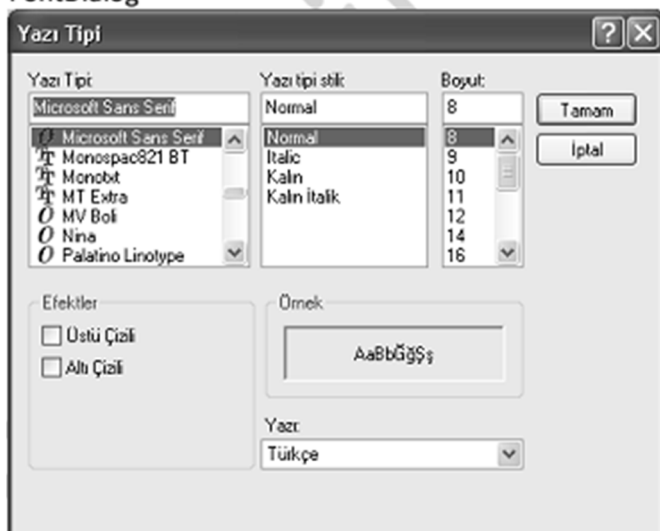
ColorDialog



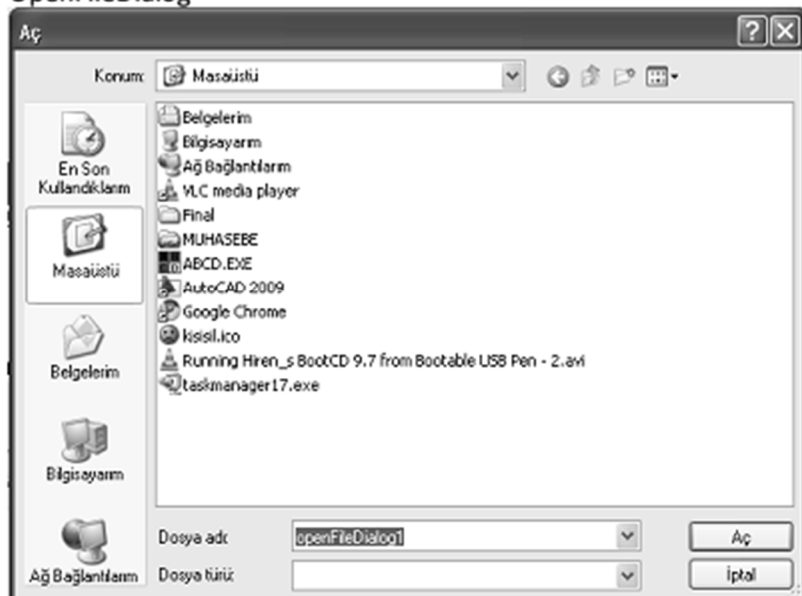
FolderBrowserDialog



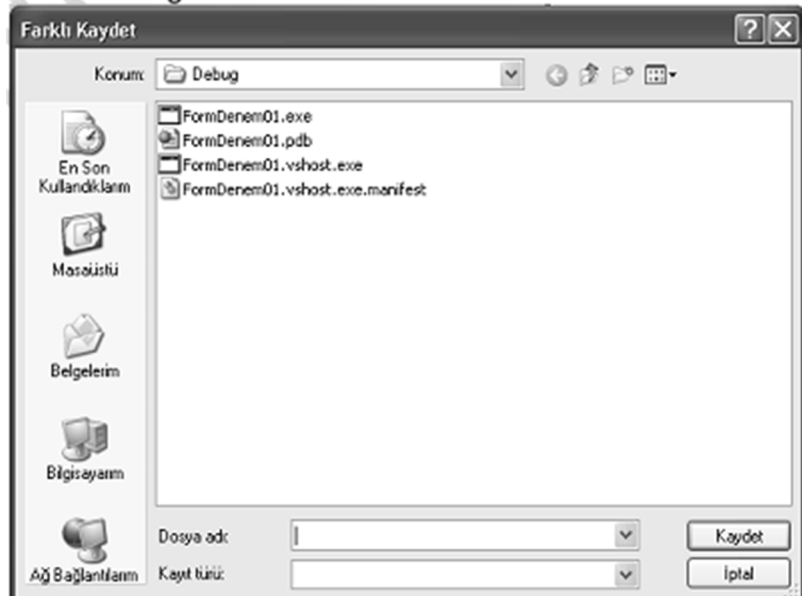
FontDialog



OpenFileDialog



SaveFileDialog



Referanslar 2: Çok Kullanılan Form Özellikleri

BackColor

Formun arka plan rengini değiştirir. Alabileceği değerler SystemColors, WebColors veya Color sınıfından oluşturulur.

BackgroundImage

Formun arka planına resim atmaya yarar. Alabileceği değerler Image sınıfından oluşturulur.

BackgroundImageLayout

Formun arka planına atılmış resmin özelliklerini değiştirir.

Alabileceği değerler :

{ None | Center | Stretch | Tile | Zoom }

Cursor

Formun üzerinde Mouse un şeklini değiştirir. Alabileceği değerler sistemin varsayılan Cursor nesnelere veya Cursor sınıfından oluşturulan nesnelere.

Font

Formun varsayılan yazı tipini değiştirir. Alabileceği değerler Font sınıfından oluşturulur.

ForeColor

Yazı rengini değiştirir. Alabileceği değerler Color sınıfından oluşturulur.

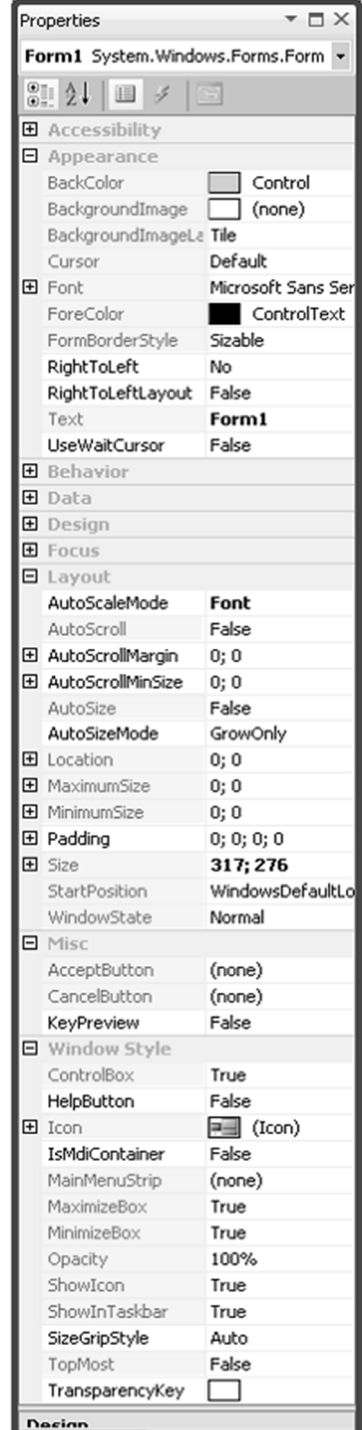
FormBorderStyle

Formun kenarlıklarını ve davranışlarını değiştirir.

Alabileceği değerler :

{None| Fixed3D | FixedDialog | FixedSingle | FixedToolWindow | Sizable | SizableToolWindow}

None olunca formun kenarlık ve başlığı oluşmaz. Fixed li bir değer olunca form çalışma anında büyütülemez veya



küçültülemez. Sizable li değerler çalışma anında ebatlar değiştirilebilir.

Text

Formun başlığındaki yazıyı değiştirir. Veri tipi string dir.

AcceptButton

Entere basınca tıklandığı varsayılan butondur. Alabileceği değerler Button sınıfından oluşturulur.

CancelButton

Esc ye basınca tıklandığı varsayılan butondur. Alabileceği değerler Button sınıfından oluşturulur.

AutoScroll

Kontroller formun üzerine sığmayınca kaydırma çubuğunun görünürlüğünü kontrol eder. Alabileceği değer true veya false olur.

AutoSize

Kontrollerin yerleşimine göre formun büyüklüğünü kontrol eder. Alabileceği değer true veya false olur.

Size

Formun anlık büyüklüğünü ifade eder. Alabileceği değerler Size sınıfından oluşturulur. Size sınıfının Width (genişlik) ve Height (yükseklik) olmak üzere iki bileşeni vardır.

MaximumSize

Formun maksimum alabileceği büyüklüğü ifade eder. Alabileceği değerler Size sınıfından oluşturulur.

MinimumSize

Formun minimum alabileceği büyüklüğü ifade eder. Alabileceği değerler Size sınıfından oluşturulur.

StartPosition

Formun başlangıç yerini verir.

Alabileceği değerler :

{ CenterParent | CenterScreen | Manual | WindowsDefaultBounds | WindowsDefaultLocation }

CenterParent: Form kendisini başlatan ana formun merkezinde gözüktür.

CenterScreen: Form ekranın ortasında gözüktür.

Manual: Form Location ile belirtilen noktada gözüktür.

WindowsDefaultBounds ve WindowsDefaultLocation değerlerinde işlemler sisteme bırakılır.

Location

Form başlatılınca sol üst köşesinin konumlandığı noktadır.

Alabileceği değer Point sınıfından oluşturulur. Point sınıfının X ve Y olmak üzere iki bileşeni vardır.

WindowState

Formun büyütme, simge durumuna getirme durumunu belirler.

Alabileceği değerler:{ Maximized | Minimized | Normal}

Maximized durumunda form ekranı kaplamış bir şekilde başlar.

Minimized durumunda form simge durumunda başlar.

Normal durumda varsayılan büyüklük ile başlar.

ControlBox

Formun büyütme ve simge durumuna getirme kontrollerini gösterir veya gizler. Alabileceği değerler true veya false olur.

Icon

Formun iconunun değiştirilmesini sağlar. Alabileceği değerler Icon sınıfından oluşturulur.

MaximizeBox

Formun maksimum edilme kutucuğunu aktif/deaktif eder. Alabileceği değerler true veya false olur.

MinimizeBox

Formun minimum edilme kutucuğunu aktif/deaktif eder. Alabileceği değerler true veya false olur.

Opacity

Formun saydamlığını kontrol eder. Alabileceği değerler 0 ile 100 arasında olmalıdır. Değer 100 olunca form normal görünür. Değer 0'a yaklaştıkça formun saydamlığı artar.

ShowIcon

Formun iconunun görünürlüğünü kontrol eder. Alabileceği değerler true veya false olur.

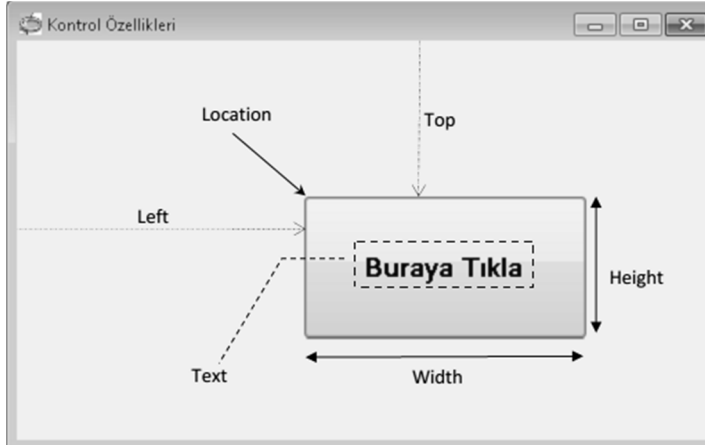
ShowInTaskbar

Görev çubuğunda görünürlüğü kontrol eder. Alabileceği değerler true veya false olur.

TopMost

Sürekli en üste olup olmamayı kontrol eder. Alabileceği değerler true veya false olur.

Referanslar 3: En çok kullanılan Kontrol Özellikleri



Location:

Bir kontrolün form veya başka bir kontrol üzerindeki konumunu belirtir. Location özelliğinin aslında bir nokta (Point) nesnesidir. Bizimde bildiğimiz gibi noktanın X ve Y olmak üzere iki tane elemanı vardır. Bir nesnenin, örneğin bir butonun konumunu ayarlamak için özellikler penceresinden Location özelliğini bulup değiştirebiliriz. Fakat kod ile bunu yapmak istersek aşağıdaki gibi bir kod yazmamız gerekecektir.

```
button1.Location = new Point(250, 350);
```

Böyle bir kod tanımladığımız zaman button1 nesnesinin X koordinatı 250, Y koordinatı 350 olacaktır. Bir başka deyişle button1 nesnesi formun solundan 250 piksel ötede olacak, formun üst tarafından ise 350 piksel aşağıda olacaktır. Bu koordinatları böyle ayarlayabildiğimiz gibi Left ve Top ile de ayarlayabiliriz. Bunun için aşağıdaki gibi bir kod yazmalıyız.

```
button1.Left = 250; // formun sol kenarına olan uzaklık
```

```
button1.Top = 350; // formun üst kenarına olan uzaklık
```

Size:

Bir kontrolün büyüklüğünü ifade eder. Bir nesnenin Size özelliği, Size sınıfının bir nesnesidir ve Width ve Height olmak üzere iki elemanı vardır. Bir nesnenin, örneğin bir butonun büyüklüğünü ayarlamak için özellikler penceresinden Size özelliğini bulup değiştirebiliriz. Fakat kod ile bunu yapmak istersek aşağıdaki gibi bir kod yazmamız gerekecektir.

```
button1.Size = new Size(150, 80);
```

Böyle bir kod tanımladığımız zaman button1 nesnesinin genişliği 150 piksel, yüksekliği ise 80 piksel olacaktır.

Kontrolün büyüklüğünü; genişliğini ve yüksekliğini ayrı ayrı belirterek de belirleyebiliriz. Bunun için aşağıdaki gibi bir kod yazmalıyız.

```
button1.Width = 150; // genişlik
```

```
button1.Height = 80; // yükseklik
```

MinimumSize & MaximumSize :

MinimumSize özelliği ile kontrolün izin vereceği en küçük yükseklik ve genişlik ayarlanır. MaximumSize ile de kontrolün izin vereceği en büyük genişlik ve yükseklikler ayarlanır. Bir nesne için bu özellikleri ayarlamak istersek yine bu nesnelere seçip özellikler penceresinden MinimumSize ve MaximumSize özelliklerini bulup ayarlayabiliriz. Kod ile yapmak istersek, örneğin bir buton için aşağıdaki gibi bir kod yazmak gerekecektir.

```
button1.MinimumSize = new Size(20, 8);
```

```
// en küçük genişlik = 20, yükseklik = 8
```

```
button1.MaximumSize = new Size(200, 150);
```

```
// en BÜYÜK genişlik = 200, yükseklik = 150
```

Text:

Nesnelerin üzerindeki yazıyı kontrol eder. Üzerinde yazı olan tüm kontrollerde geçerlidir. Bir kontrolün üzerindeki yazıyı ayarlamak için o kontrol seçilip özellikler penceresinin Text özelliğinden değiştirilebileceği gibi kod ile yapmak için aşağıdaki gibi bir kod yazmak gerekecektir.

```
button1.Text = "Buraya Tıkla";
```

Yukarıdaki kod ile button1 kontrolündeki yazı "Buraya Tıkla" olacaktır.

TextAlign:

Nesnelerin üzerindeki yazının hizalamasını kontrol eder. Örnek ile beraber alabileceği değerler aşağıda verilmiştir. Alabileceği değerler ContentAlignment. İle beraber yazılır.

```
button1.TextAlign = ContentAlignment.MiddleCenter;
```

```
// Yazı hizalamayı orta merkeze alır.
```

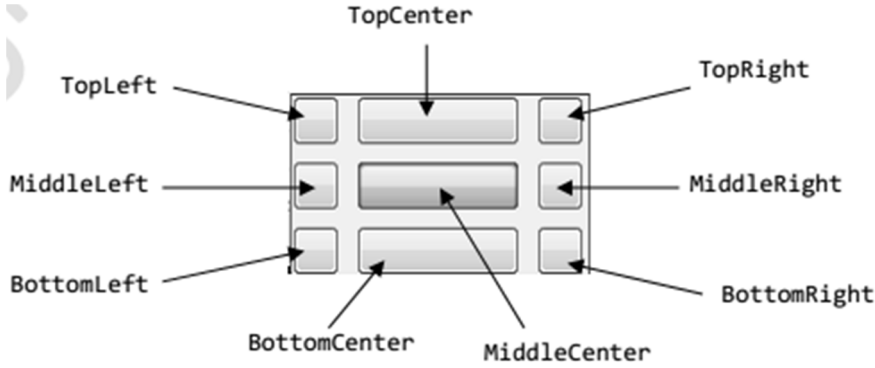
Bu özellik ofis dosyalarında tablolardaki yazı hizalaması ile aynı mantıktadır. Alabileceği değerler aşağıda listede verilmiştir.

```
ContentAlignment.BottomCenter // Yazıyı altta ortaya alır
```

```
ContentAlignment.BottomLeft // Yazıyı altta sola hizalar
```

ContentAlignment.BottomRight // Yazıyı altta sağa hizalar
ContentAlignment.MiddleCenter // Yazıyı ortada merkeze hizalar
ContentAlignment.MiddleLeft // Yazıyı ortada sola hizalar
ContentAlignment.MiddleRight // Yazıyı ortada sağa hizalar
ContentAlignment.TopCenter // Yazıyı üstte ortaya hizalar
ContentAlignment.TopLeft // Yazıyı üstte sola hizalar
ContentAlignment.TopRight // Yazıyı üstte sağa hizalar

TextAlign özelliği özellikler penceresinden ayarlandığı zaman aşağıdaki gibi bir şekil açılır. Bu şekilden yazının kontrolün hangi bölgesinde olması gerekeceği seçilerek ayarlanır. Hangi bölgenin hangi anlama geldiği oklarla belirtilmiştir.



Font:

Kontrollerin üzerindeki yazı tipini, büyüklüğünü ve stilini belirler. Kontrol seçilerek o kontrolün üzerindeki yazı tipini, büyüklüğünü ve stilini değiştirmek için özellikler penceresinden Font özelliği ayarlanarak yapılmak istenen değişiklik sağlanabilir. Bir nesnenin, örneğin bir butonun üzerindeki yazı tipi, büyüklüğü ve stilini değiştirmek için aşağıdaki kod gibi bir kod kullanılır.

```
button1.Font = new Font("Arial", 12, FontStyle.Bold);
```

Dikkat edilirse Font tanımlarken üç tane parametre belirtilmiş. Bunlardan ilki fontun aile ismini, ikincisi fontun büyüklüğünü, üçüncüsü ise fontun stilini belirtir. Fontun büyüklüğünü belirtmek için kesirli sayılar da kullanılabilir (11.5 gibi).

Yazı Stilinin olarak alabildiği değerler aşağıdaki gibidir.

Bold | **Italic** | **Regular** | **Strikeout** | **Underline**

Bold : Yazı stilinin koyu (kalın) olmasını sağlar.

Italic : Yazı stilinin eğik olmasını sağlar.

Regular : Yazı stilinin normal olmasını sağlar.

Strikeout : Yazının üstü çizili olmasını sağlar.

Underline : Yazının altının çizili olmasını sağlar.

Bu stiller beraber kullanılabilir. Bu durumda stiller arasına dik-çizgi | karakteri konulur. Örneğin yazı stilinin **koyu, altı çizili ve italik** olmasını istersek aşağıdaki gibi bir kod yazarız.

```
button1.Font = new Font("Arial",12,FontStyle.Bold | FontStyle.Italic  
| FontStyle.Underline);
```

ForeColor:

Kontrolün yazı rengini belirler. Kontrol seçildikten sonra özellikler penceresinden kontrolün üzerindeki yazının rengi ayarlanabilir. Kod ile ayarlamak gerekirse, örneğin bir butonun üzerindeki yazı rengini ayarlamak için aşağıdaki gibi bir kod yazmak gerekecektir.

```
button1.ForeColor = Color.Red;
```

ForeColor renk özelliği olduğu için alabildiği değerler BackColor ile aynıdır.

BackColor:

Kontrollerin arka plan rengini belirler. Kontrol seçildikten sonra özellikler penceresinden kontrolün arka plan rengi ayarlanabilir. Kod ile ayarlamak gerekirse, örneğin bir butonun arka plan rengini ayarlamak için aşağıdaki gibi bir kod yazmak gerekecektir.

```
button1.BackColor = Color.Yellow; // buton rengi sarı olacak
```

Renk oluşturmak için Color. ile çıkan ve önceden tanımlanmış renkler kullanabildiğimiz gibi kendimiz de RGB renk kodlarını belirterek renk oluşturabiliriz. Bunun için **Color.FromArgb** fonksiyonu kullanılır. Bu fonksiyon üç parametre alır. Bu parametreler RGB (red, green, blue) (Kırmızı, Yeşil, Mavi) renklerinin tonlarından oluşur.

```
button1.BackColor = Color.FromArgb(150, 120, 180);
```



Yandaki şekilde

Color.FromArgb(150, 120, 180) kodu ile oluşacak rengin tonu verilmiştir.

Burada dikkat edilmesi gereken renk tonları 0 ile 255 arasında olmalıdır.

Bunların yanında birde sistemde Windows stilleri için tanımlanmış SystemColors sınıfı vardır. BackColor veya ForeColor için bunlarda kullanılabilir.

```
button1.BackColor = SystemColors.ButtonFace;
```

```
// buton rengi varsayılan windows stilindeki buton rengi olacaktır.
```

```
button1.ForeColor = SystemColors.WindowText;
```

```
// buton yazı rengi varsayılan windows stilindeki yazı rengi olacaktır.
```

TabStop:

TAB tuşuna basılınca kontrolün odaklanıp odaklanmamasına karar verir. TAB tuşu ile kontroller arasında gezerken durmasını istediğimiz kontroller için true, istemediklerimiz için false yaparız. Kod ile yazacak olursak örneğin bir buton için aşağıdaki kodu yazmamız gerekecektir.

```
button1.TabStop = true;
```

```
//TAB tuşu ile kontroller arasında gezerken button1 nesnesi odaklanacaktır.
```

TabIndex:

TAB tuşuna basılınca kontroller arasında odaklama yapılma sırasını belirler. Örneğin kullanıcı adı için bir alan, şifre için bir alan, tamam için de bir buton



olduğunu varsayalım. Bunun için özellikler penceresinden sırasıyla kontrolleri ayrı ayrı seçip kullanıcı adı için ayrılmış metin kutusunun tab sırasını 1, şifre için olan metin kutusunun tab sırasını 2, tamam için olan butonun da tab sırasını 3 yaparak tab sıralarını ayarlayabiliriz. Bu işlemleri kod ile yapacak olursak aşağıdaki gibi bir kod yazmamız gerekecektir.

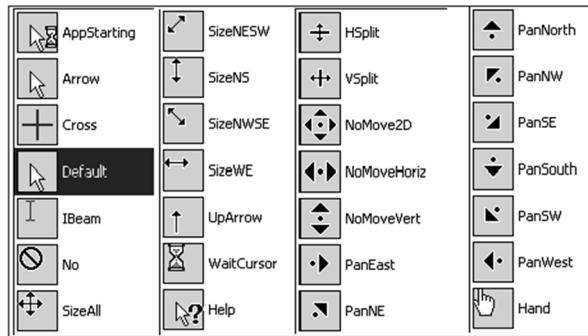
```
txtUser.TabIndex = 1; // önce kullanıcı adına odaklanacak
```

```
txtPass.TabIndex = 2; // sonra şifre alanına odaklanacak
```

```
btnTamam.TabIndex = 3; // sonrada tamam butonuna odaklanacak.
```

Cursor:

Kontrollerin üzerinde Mouse ile gelindiğinde kursörün değişmesini kontrol eder. Sistemde varsayılan olarak tanımlanan Kursörlerden kullanabileceğimiz gibi kendimiz de dosyadan



kursör oluşturabiliriz. Sistemde varsayılan kursörler Cursors. ile çıkıyor. Cursor. ile çıkabilecek Kursörler yanda şekilleri ile beraber verilmiştir. Bir kontrol için örneğin bir buton için korsörü kod ile ayarlamak istersek aşağıdaki gibi bir kod yazmamız gerekecektir. Kursörü hazır kursörlerden kullanabileceğimiz gibi bir kursör dosyasından da oluşturabiliriz. Aşağıdaki ikinci satırdaki kod bir kursör dosyasından kursör yapmayı gösteriyor.

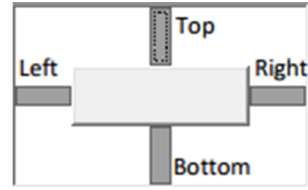
```
button1.Cursor = Cursors.Hand;  
// butonun üzerine geldiğimizde kursör el şekline girecektir.  
button1.Cursor = new Cursor(@"C:\Kursorler\cursor1.cur");  
// kursörü dosyadan oluşturuyoruz.
```

Anchor:

Nesneyi içinde bulunduğu kontrolün belirli yerlerine kilitlemeye yarar. Örneğin formun sağ alt tarafına koyduğumuz bir butonun, formun büyütülüp küçültülmesi durumunda da sürekli olarak altta sağda durmasını isteriz. Bunu sağlamak için aşağıdaki kodu yazmamız gerekiyor.

```
button1.Anchor = AnchorStyles.Bottom | AnchorStyles.Right;  
// butonun sağ alta kilitlenecek.
```

Hangi tarafa veya taraflara kilitleyeceğimizi AnchorStyles. ile beraber kullanıyoruz. Eğer bir kontrolün Anchor özelliğini özellikler penceresinden ayarlamak istersek özellikler penceresinden Anchor un alabileceği değerler için yabdaki şekil açılacaktır. Fakat kod ile yazacak olursak Anchor özelliğinin alabileceği değerler aşağıdaki listedeki gibi olabilecektir.



```
AnchorStyles.None  
AnchorStyles.Bottom  
AnchorStyles.Left  
AnchorStyles.Right  
AnchorStyles.Top  
AnchorStyles.Bottom | AnchorStyles.Right  
AnchorStyles.Left | AnchorStyles.Bottom  
AnchorStyles.Top | AnchorStyles.Right  
AnchorStyles.Top | AnchorStyles.Bottom | AnchorStyles.Left  
AnchorStyles.Top | AnchorStyles.Left | AnchorStyles.Right |  
AnchorStyles.Bottom
```

Yukarıdaki örnekte görüldüğü gibi herhangi iki, üç veya dördünü beraber kullanılabiliyoruz. Bu durumda alabilecek değerler arasına dik-çizgi | koyuyoruz.

Dock:

Nesneyi içinde bulunduğu kontrolün belirli bir kenarını veya tamamını dolduracak şekilde kaplamasını sağlar. Örneğin butonumuz formun sağ tarafını doldursun istiyorsak aşağıdaki gibi bir kod yazarız.

button1.Dock = DockStyle.Right;

Dock özelliğinin alabileceği değerler DockStyle. ile beraber kullanılır. Aşağıda alabileceği değerler listelenmiştir. Eğer özellikler penceresinden bu özellik ayarlanacaksa yandaki gibi bir şekil açılacaktır. Alabildiği değerler aşağıda gösterilmiştir.



None : Kontrolün herhangi bir kenara yapışmasını önler.

Fill : Nesneyi içinde bulunduğu kontrolün tamamını doldurmasını sağlar.

Bottom : Nesneyi içinde bulunduğu kontrolün altına yapışmasını sağlar.

Left : Nesneyi içinde bulunduğu kontrolün soluna yapışmasını sağlar.

Right : Nesneyi içinde bulunduğu kontrolün sağına yapışmasını sağlar.

Top : Nesneyi içinde bulunduğu kontrolün üstüne yapışmasını sağlar.

Referanslar 4: Kontroller için genel olaylar (Event)

<u>Adı</u>	<u>Tanımlama</u>
BackColorChanged	Kontrolün BackColor özelliği değiştiği zaman devreye girer.
BackgroundImageChanged	Kontrolün BackgroundImage özelliği değiştiğinde devreye girer.
ControlAdded	Kontrolün üzerine yeni bir kontrol eklenince devreye girer.
ControlRemoved	Kontrolün üzerindeki kontrollerden biri silinince devreye girer.
CursorChanged	Kontrolün Cursor özelliği değişince devreye girer.
Click	Kontrole tıklanınca devreye girer
DoubleClick	Kontrol çift tıklandığında devreye girer.
DragDrop	Kontrolün üzerine bir şey sürüklenip bırakılınca devreye girer.
DragEnter	Kontrolün üzerine bir şey sürüklerken kontrolün sınırı geçilip içeriye girerken devreye girer.
DragLeave	Kontrolünden dışarıya bir şey sürüklerken kontrolün sınırı geçilip dışarıya çıkarken devreye girer.
DragOver	Kontrolün üzerine bir şey sürüklenip henüz bırakılmamışken devrededir.
GotFocus	Kontrole focus edilince (aktif edilince) devreye girer.
KeyDown	Kontrol aktifken bir tuşa basılınca devreye girer.
KeyPress	Kontrol aktifken bir tuşa basılıp bırakılınca devreye girer.
KeyUp	Kontrol aktifken bir tuşa basılıp bırakılınca bırakılırken devreye girer.
Layout	Kontrol ekranda gözükrken ve üzerindeki nesnelere yerleştirirken devreye girer.

LocationChanged	Kontrolün yeri deęişince devreye girer.
MouseClicked	Kontrolle Mouse ile tıklanınca devreye girer.
MouseDoubleClick	Kontrolle Mouse ile çift tıklanınca devreye girer.
MouseDown	Kontrolle Mouse ile tıklanınca Mouse butonu henüz inerken devreye girer.
MouseEnter	Kontrolün üzerine Mouse ile girince sınırını geçip içeri girince devreye girer.
MouseLeave	Kontrolün üzerine Mouse ile girilmişken sınırını geçip dışarı çıkınca devreye girer.
MouseMove	Kontrolün üzerinde her Mouse hareketinde devreye girer.
MouseUp	Kontrolle Mouse ile tıklanınca Mouse butonu henüz geri bırakılınca devreye girer.
MouseWheel	Kontrol aktifken ün üzerinde Mouse tekerlekleri yuvarlatılınca devreye girer.
Move	Kontrol yerini deęiştirince (hareket ederken) devreye girer.
Paint	Kontrol ekranda çizdirilirken devreye girer.
Resize	Kontrolün büyüklüğü deęiştirilince devreye girer.
SizeChanged	Kontrolün Size özellięi deęişince devreye girer.
TextChanged	Kontrolün Text özellięi deęişince devreye girer.
VisibleChanged	Kontrolün Visible özellięi (Ekranda görünürlülük) deęişince devreye girer.
Disposed	Kontrol Dispose metodu çağırılarak yok edilirken devreye girer.

Geçmiş Dönem Soruları:

Adı Soyadı :

A Grubu

Aldığı Not

Numarası :

NOT: Sınav Süresi 50 Dakikadır – Test Bölümü 40, Klasik Bölüm 60 Puandır.

BAŞARILAR DİLERİM

Öğr. Gör. Hasan SANCAK

1213BDBİLNTPIIVA

1. richTextBox1.Text.IndexOf("Bilgisayar", 0) işlemi neyi gerçekleştirir?
A) richTextBox içindeki Bilgisayar kelimesine konumlanır.
B) Indexin bulunduğu yerden itibaren Bilgisayar kelimesini arar.
C) richTextBox in sonundan itibaren Bilgisayar kelimesini arar.
D) richTextBox in başından itibaren Bilgisayar kelimesini arar.
E) richTextBox in başından itibaren Bilgisayar kelimesine göre indexler.
2. Form kontrolünün varsayılan olayı (Default Event) nedir?
A) Load B) Click C) Enter D) Leave E) KeyDown
3. Aşağıdakilerden hangisi ToolStrip nesnelere birisi değildir?
A) Button B) Separator C) TextBox
D) RadioButton E) ComboBox
4. Kontrollerin üzerine geldiğinde küçük bilgi mesajı verilmesi için kullanılan özellik hangisidir?
A) Text B) ToolTipText C) Checked
D) Font E) TextAlign
5. Çoklu form uygulamalarında uygulamanın başlangıç formunu belirlemek için hangisi kullanılır?
A) Application.Run B) Form.ShowDialog C) Form.Show
D) this.Hide E) this.Show
6. Aşağıdakilerden hangisi richTextBox Box 'ın tasarımında ulaşılabilecek özelliklerinden biri değildir?
A) ZoomFactor B) WordWrap C) SelectedText
D) DetectUrls E) BulletIntend
7. Form üzerindeki Kapat, Ekranı Kapat, Simge Durumu düğmelerini aktif hale getirmek ya da tam tersi için formun hangi özelliği kullanılır?
A) Opacity B) AcceptButton C) CancelButton
D) ControlBox E) TopMost
8. .Net Framework çatısında uygulamaları başlatmak yönetmek ve sonlandırmak için hangi sınıf kullanılır?
A) Form B) Application C) FileStream
D) File E) String
9. Form sınıfının Hide metodu formun hangi özelliğini değiştirerek işlem gerçekleştirir?
A) Opacity B) Text C) Font D) Enabled E) Visible
10. richTextBox içine yazılan uzun yazıların bir sonraki satıra geçerek görüntülenmesini sağlamak için aşağıdakilerden hangisi kullanılmalıdır?
A) richTextBox1.ZoomFactor=0;
B) richTextBox1.DetectUrls=True;
C) richTextBox1.WordWrap=True;
D) richTextBox1.AcceptsTab=True;
E) richTextBox1.Capture=False;
11. I. Enabled=True özelliği ile Start metodu aynı işlemi yapar.
II. Enabled=False özelliği ile Stop metodu aynı işlemi yapar.
III. Interval özelliği sayacın çalışma için zaman aralığını belirler.
Timer kontrolü için yukarıdaki ifadelerden hangileri doğrudur?
A) III B) I-II C) I-II-III D) II-III E) I-III
12. Dialog kutularında ya da Mesaj pencerelerinde kullanıcının hangi butona bastığını kontrol etmek için hangisi kullanılır?
A) LoadFile B) SaveFile C) Interval
D) DialogResult E) Enabled
13. string adres = @"C:\Windows\system.ini";
FileInfo f = new FileInfo(adres);
MessageBox.Show(f.Extension);

Mesaj Penceresi ekranında ne gösterilir?
A) C: B) Windows C) system.ini
D) ini E) C:\
14. string adres = @"C:\Windows\system.ini";
FileInfo f = new FileInfo(adres);
MessageBox.Show(f.DirectoryName);

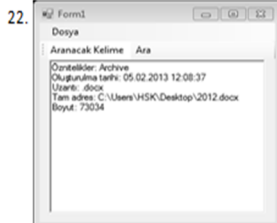
Mesaj Penceresi ekranında ne gösterilir?
A) C:\Windows B) Windows\system.ini C) system.ini
D) Windows E) C:\
15. string adres = @"C:\Personel";
DirectoryInfo d = new DirectoryInfo(adres);
Personel klasörünü oluşturmak için aşağıdakilerden hangisi kullanılır?
A) d.Create(); B) d.Directory();
C) CreateDirectory(); D) d.CreationTime;
E) d.FullName;
16. string adres = @"C:\Personel";
DirectoryInfo d = new DirectoryInfo(adres);
Belirtilen klasörün var olup olmadığını bulmak için hangisi kullanılır?
A) d.Extension B) d.Exists C) d.Root
D) d.Name E) d.Parent
17. FileStream sınıfı hangi tip dosyalar için kullanılır?
A) Sayı Tipi Veri Dosyaları B) İkili Tipteki Dosyalar
C) Her Tip Dosya için D) String Tipi Veri Dosyaları
E) Tarih Tipi Veri Dosyaları
18. Hangisi genellikle ikili tipteki dosyalarda işlem yapmak için kullanılan sınıflardandır?
A) StreamReader B) TextReader C) StreamWriter
D) BinaryReader E) File
19. Aşağıdakilerden hangisi FileStream sınıfının FileMode seçeneklerinden biri değildir?
A) CreateNew B) Truncate C) Open
D) ReadWrite E) OpenOrCreate
20. Aşağıdakilerden hangisi VS 2010 Toolbox Panelinin Menus&Toolbars grubunda yer almayan bir kontroldür?
A) MenuStrip B) ContextMenuStrip C) ToolStrip
D) StatusStrip E) RichTextBox



Diğer Sorular için sayfayı çeviriniz !!!



- I. Başla butonuna(btnBasla) basılınca her üç saniyede bir(Timer kontrolü ile) sistem tarih ve saatini form başlığında gösteren, bu tarih ve saati C:\Saat.txt dosyasına(FileStream ile) yazdıran.
- II. Dur Butonuna(btnDur) basılınca dosyaya yazmayı durduran
- III. Dosya Oku butonuna(btnDosyaOku) basılınca C:\Saat.txt dosyasındaki bilgileri satır satır ListBox'a aktaran programı yazınız?



Dosya Menüsü

- Dosya Aç
- Dosya Kaydet
- openFileDialog
- saveFileDialog
- (TextBox + Button)
- richTextBox

- I. Dosya Aç tıklanınca aç iletişim kutusundan(openFileDialog ile) istenilen dosya seçilince bu dosyanın (FileInfo ile) Özelliklerini, Oluşturulma Tarihini, Uzantısını, Tam Adresini ve Boyutunu richTextBox'a aktaran,
- II. Dosya Kaydet tıklanınca richTextBox'taki bilgileri kaydet iletişim kutusundan (saveFileDialog ile) istenilen yere istenilen isimle dosyaya yazdıran (StreamWriter ile)
- III. Aranacak kelime girilip Ara butonuna basılınca richTextBox içinde kelimeyi arayan varsa o kelimeye odaklanan programı yazınız?

Adı Soyadı :
Numarası :

A Grubu

Aldığı Not

NOT: Sınav Süresi 50 Dakikadır – Test Bölümü 60, Klasik Bölüm 40 Puandır.

BAŞARILAR DİLERİM

Öğr. Gör. Hasan SANCAR

1213BDBİLNTPIIFA

- Crystal Report Toolbox aracında aşağıdaki kontrollerden hangisi yer almaz?
A) Label Object B) Text Object C) Pointer
D) Line Object E) Box Object
- Data Grid te seçili satırın 3. Alanına ait değeri textBox1'e aktarmak için hangi komut kullanılmalıdır?
A) textBox1.Text = dataGridView1.CurrentRow.Cells[0].Value.ToString();
B) textBox1.Text = dataGridView1.CurrentRow.Cells[2].Value.ToString();
C) textBox1.Text = dataGridView1.Columns[2].HeaderText
D) textBox1.Text = dataGridView1.CurrentRow.Cells[3].Value.ToString();
E) textBox1.Text = dataGridView1.Columns[0].HeaderText
- Access ve diğer veri tabanlarında bağlantı açmak ya da kapatmak için hangi sınıf kullanılır?
A) SqlConnection B) SqlDataAdapter
C) OleDbConnection D) OleDbCommand
E) OleDbDataAdapter
- Kontrollerin üzerine gelindiğinde küçük bilgi mesajı verilmesi için kullanılan özellik hangisidir?
A) Text B) ToolTipText C) Checked
D) Font E) TextAlign
- XxxConnection sınıfının State özelliğinin hangi değeri veri tabanı bağlantısının kapalı olduğu anlaşılabilir?
A) Broken B) Fetching C) Executing
D) Connecting E) Closed
- Crystal Report Fields menüsünde hazır özel fonksiyonlar barındıran seçenek aşağıdakilerden hangisidir?
A) Database Fields B) Special Fields C) Form Fields
D) Running Fields E) Unbound Fields
- Crystal Report tasarımı yapılan raporun formda görüntülenmesi için hangi kontrole ihtiyaç duyulur?
A) ReportViewer B) Report Source C) DataGridView
D) CrystalReportViewer E) GroupBox
- Data Grid te herhangi bir hücre seçildiğinde o hücrenin bulunduğu tüm sütunun seçilmesi için SelectionMode özelliğine hangi değer aktarılır?
A) DataGridViewSelectionMode.FullColumnSelect
B) DataGridViewSelectionMode.CellSelect
C) DataGridViewSelectionMode.FullRowSelect
D) DataGridViewSelectionMode.ColumnHeaderSelect
E) DataGridViewSelectionMode.RowHeaderSelect
- XxxCommand sınıfı metodlarından hangisi komutu çalıştırıp etkilenen kayıt sayısını döndürür?
A) ExecuteReader B) ExecuteScalar
C) ExecuteNonQuery D) ExecuteXmlReader
E) Execute
- SQL Server veri tabanının kullanılabilecek bir uygulamada SQL Sunucusuna Windows hesabı ile bağlanılacağını belirtmek için bağlantı cümlesinde hangi parametre kullanılır?
A) Initial Catalog B) ConnectionTimeout
C) Integrated Security D) Data Source
E) Provider

- I. Enabled=True özelliği ile Start metodu aynı işlemi yapar.
II. Enabled=False özelliği ile Stop metodu aynı işlemi yapar.
III. Interval özelliği sayacın çalışması için zaman aralığını belirler.
Timer kontrolü için yukarıdaki ifadelerden hangileri doğrudur?
A) III B) I-II C) I-II-III D) II-III E) I-III
- Crystal Report rapor nesnesinde sayfa numaraları genellikle raporun hangi bölümüne eklenir?
A) Report Header B) Page Header C) Details
D) Report Footer E) Page Footer
- dt isimli sanal tablo nesnesine (Data Table) aktarılan bilgileri Data Grid te görüntülemek için hangi komut kullanılmalıdır?
A) DataTable dt = new DataTable();
B) SqlDataAdapter da = new SqlDataAdapter();
C) da.Fill(dt);
D) OleDbDataAdapter da = new OleDbDataAdapter();
E) dataGridView1.DataSource = dt;
- string adres = @"C:\Windows\system.ini";
FileInfo d = new FileInfo(adres);
MessageBox.Show(d.DirectoryName);
Mesaj Penceresi ekranında ne gösterilir?
A) C:\Windows B) Windows\system.ini C) sytem.ini
D) Windows E) C:\
- CrystalReportViewer kontrolünün hangi özelliği ile formda görüntülenecek rapor nesnesi belirlenir?
A) Name B) Report Source C) Text
D) Size E) Fore Color
- Tablo kayıtları üzerinde değişiklik (Güncelleme, Silme vb..) gerçekleştirebilmek için hangi sınıf kullanılır?
A) XxxCommand B) XxxConnection
C) XxxDataReader D) XxxDataAdapter
E) DataSet
- Data Grid te görüntülenen tablonun alanlarını farklı isimle görüntülemek için hangisi kullanılır?
A) dataGridView1.Columns[0].HeaderText
B) DataGridViewSelectionMode
C) dataGridView1.DataSource
D) dataGridView1.CurrentRow.Cells[0].Value
E) dataGridView1.CurrentRow.Cells[3].Value.ToString();
- Çoklu form uygulamalarında uygulamanın başlangıç formunu belirlemek için hangisi kullanılır?
A) Application.Run B) Form.ShowDialog C) Form.Show
D) this.Hide E) this.Show
- Aşağıdakilerden hangisi FileStream sınıfının FileMode seçeneklerinden biri değildir?
A) CreateNew B) Truncate C) Open
D) ReadWrite E) OpenOrCreate
- Crystal Report Fields menüsünde hangisi yer almaz?
A) Database B) Formula C) Special
D) ReportViewer E) Group Name



Diğer Sorular için sayfayı çeviriniz !!!

#/ Kitap Takip

ISBN No

Kitap Ad

Yazar Ad

Konu

Listele Ekle

Güncelleme Sil

ISBNNO	KITAPADI	YAZARADI	KONUSU
*			

SQL Sunucu Adı : HSKPB\MSSQLServer
 Veri Tabanı Adı : Kutuphane
 Tablo Adı : KITAP
 Tablo Alanları : ISBNNO → NCHAR(10)
 KITAPADI → NCHAR(20)
 YAZARADI → NCHAR(20)
 KONUSU → NCHAR(100)

21. a. Bağlantı cümlesini ve bağlantı kontrolünü yapan Bağlan() metodunu yazınız.

b. Tüm kayıtların dataGridView1'de listelenmesi için gerekli Listele() Metodunu yazınız?

22. textBox'lardaki değerleri veri tablosuna kaydeden Ekle butonuna ait kodları yazınız?

23. ISBNNO alanına göre textBox'lardaki değerlerle kaydı güncelleyen Güncelle butonuna ait kodları yazınız?

24. ISBNNO alanına göre belirtilen kaydı silen Sil butonuna ait kodları yazınız?

KAYNAKLAR

1. Öğr. Gör. Hasan SANCAK Ders Notları
2. Her Yönüyle C# 5.0 Volkan AKTAŞ
3. Her Yönüyle C# 4.0 Sefer ALGAN
4. Visula Studio 2008 ile Microsoft C# 3.0 Memik YANIK
5. Bilge Adam C# Kurs Notları
6. Megep Dökümanları

Ders Notları

