

Windows Forms kullanarak Graphical User Interface

Yrd. Doç. Dr. Özgü Can

GUI

- GUI'ler kullanıcının program ile **görsel olarak etkileşim** kurmasını sağlar.
 - Mouse
 - Keyboard
 - Diğer input'lar (ÖR: Ses komutları)
- GUI kontrolleri kullanılarak oluşturulur.
 - Components
 - Widgets
 - Window Gadgets

GUI Kontrolleri

- Bazı temel GUI kontrolleri:

Control	Description
Label	Displays images or uneditable text.
TextBox	Enables the user to enter data via the keyboard. It can also be used to display editable or uneditable text.
Button	Triggers an event when clicked with the mouse.
CheckBox	Specifies an option that can be selected (checked) or unselected (not checked).
ComboBox	Provides a drop-down list of items from which the user can make a selection either by clicking an item in the list or by typing in a box.
ListBox	Provides a list of items from which the user can make a selection by clicking one or more items.
Panel	A container in which controls can be placed and organized.
NumericUpDown	Enables the user to select from a range of numeric input values.

Windows Forms

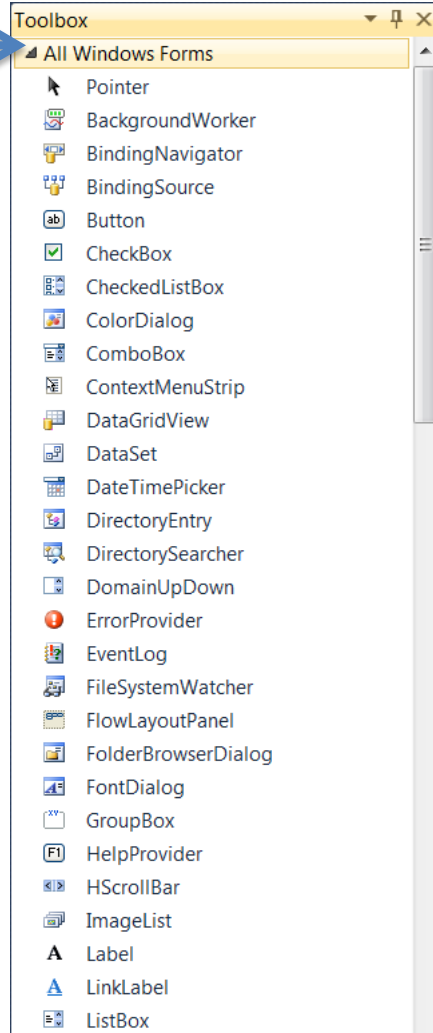
- GUI'lerin yaratılmasında kullanılır.
- Form
 - Grafiksel bir element
 - Kontrol ve bileşenler için bir kapsayıcı (container)
 - Masaüstünde görüntülenen bir
 - Diyalog
 - Pencere
 - MDI (**M**ultiple **D**ocument **I**nterface) penceresi olabilir.

Windows Forms

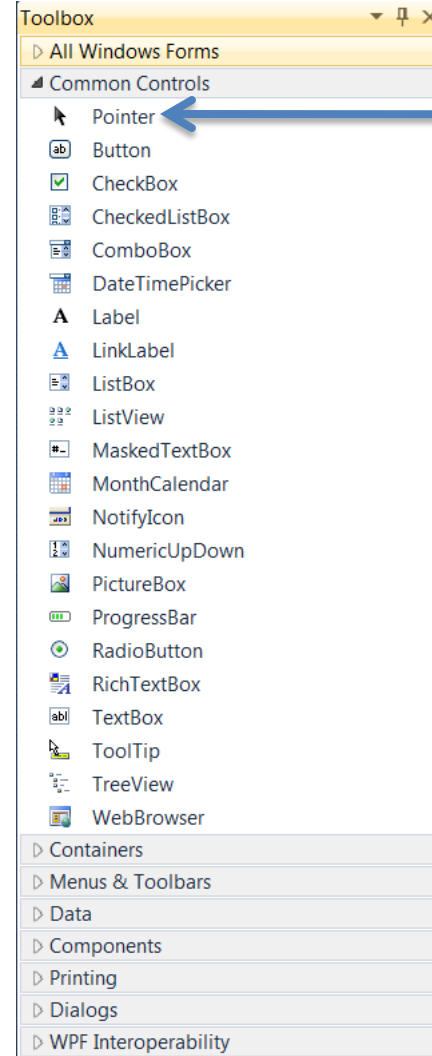
- Kontroller, yürütüm süresinde (run-time) **grafiksel gösterime** sahiptir.
 - Button
 - Label
- Bazı bileşenler grafiksel gösterime sahip değildir.
 - Timer
 - Yürütüm süresinde görünür değildir.

Toolbox

Bütün
kontrollerin
listelenmesi



Seçimi Kaldırma



Fonksiyonelliğe
bağlı
Kategoriler

Windows Forms

- Kontrol ve bileşenler
 - Namespace → **System.Windows.Forms**
- Kontroller ve olay-işleyiciler (event-handler) yaratıldığında, GUI ile ilişkili kodlar **Visual Studio** tarafından üretilir.

Temel Form Özellikleri, Metotlar ve Olay

Form properties, methods and an event	Description
<i>Common Properties</i>	
AcceptButton	Button that is clicked when <i>Enter</i> is pressed.
AutoScroll	bool value that allows or disallows scrollbars when needed.
CancelButton	Button that is clicked when the <i>Escape</i> key is pressed.
FormBorderStyle	Border style for the Form (e.g., none, single, three-dimensional).
Font	Font of text displayed on the Form, and the default font for controls added to the Form.
Text	Text in the Form's title bar.
<i>Common Methods</i>	
Close	Closes a Form and releases all resources, such as the memory used for the Form's contents. A closed Form cannot be reopened.
Hide	Hides a Form, but does not destroy the Form or release its resources.
Show	Displays a hidden Form.
<i>Common Event</i>	
Load	Occurs before a Form is displayed to the user. The handler for this event is displayed in the Visual Studio editor when you double click the Form in the Visual Studio designer.

Olay-İşleme (Event-Handling)

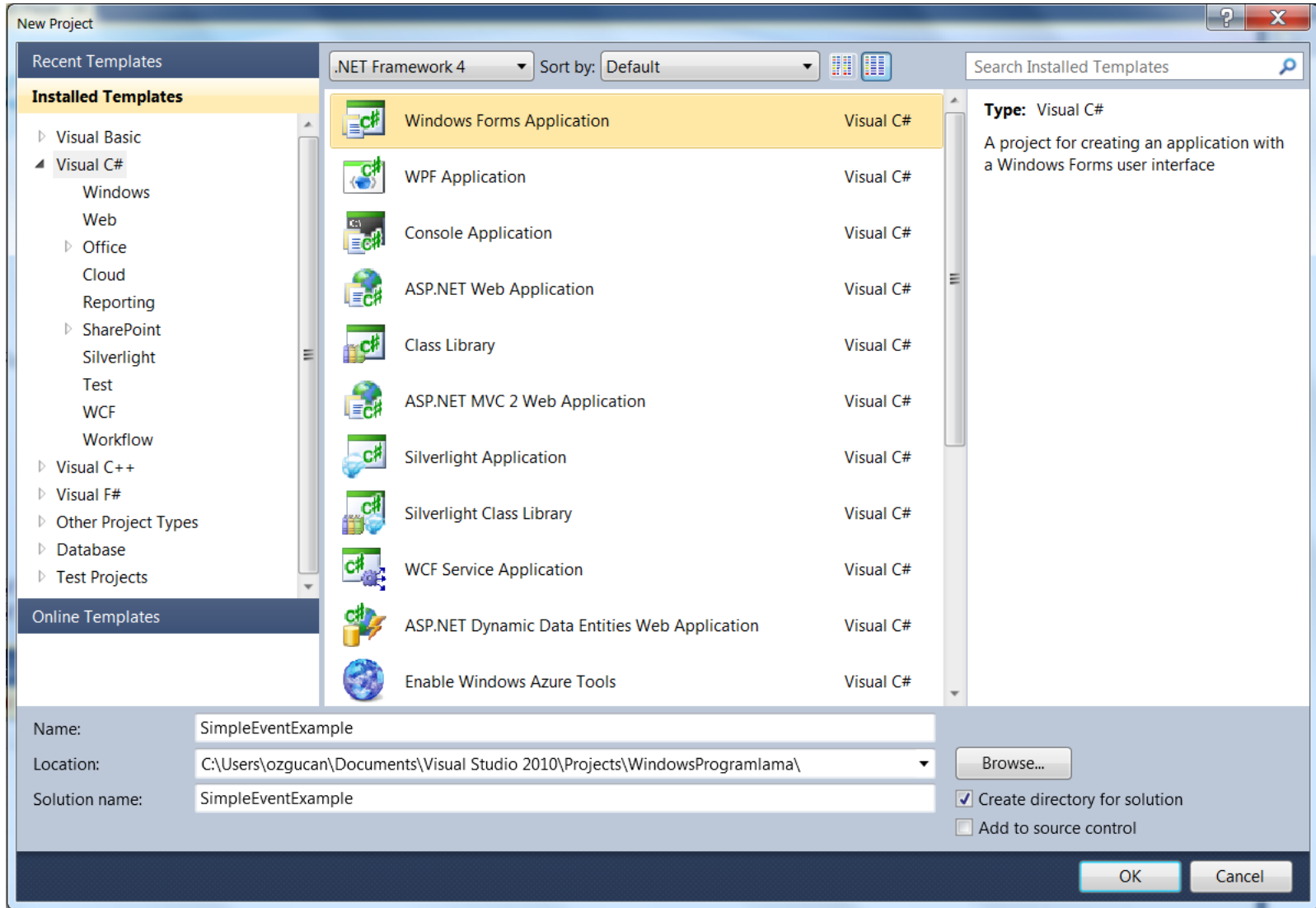
- GUI'ler olay güdümlüdür (*event-driven*).
- Kullanıcının GUI ile etkileşimi → **Olay (Event)**
- Olay → Uygulamanın bir görev (task) gerçekleştirmesine yol açar.
 - **Button**'a tıklanması
 - Mouse'un hareket ettirilmesi
 - Pencerenin kapatılması
 - **Textbox**'a metin girilmesi

Olay-İşleme

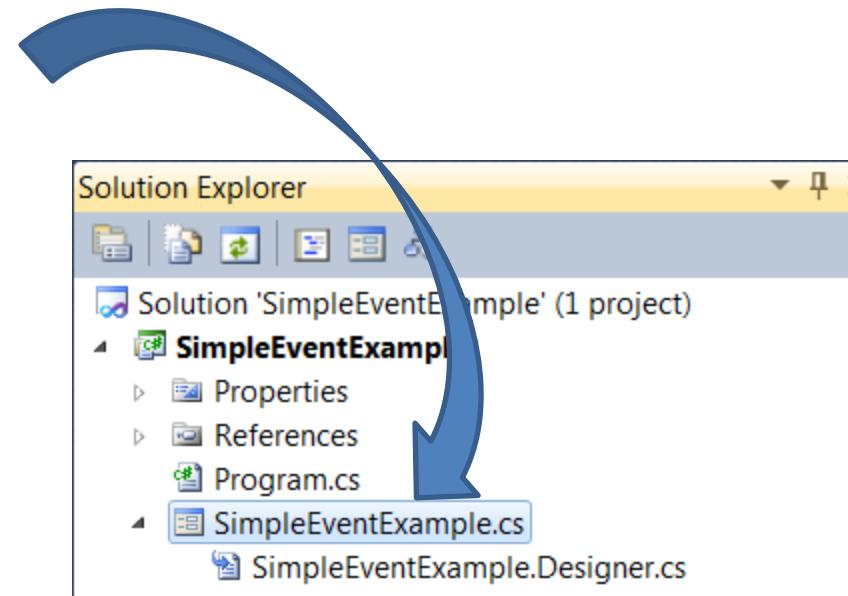
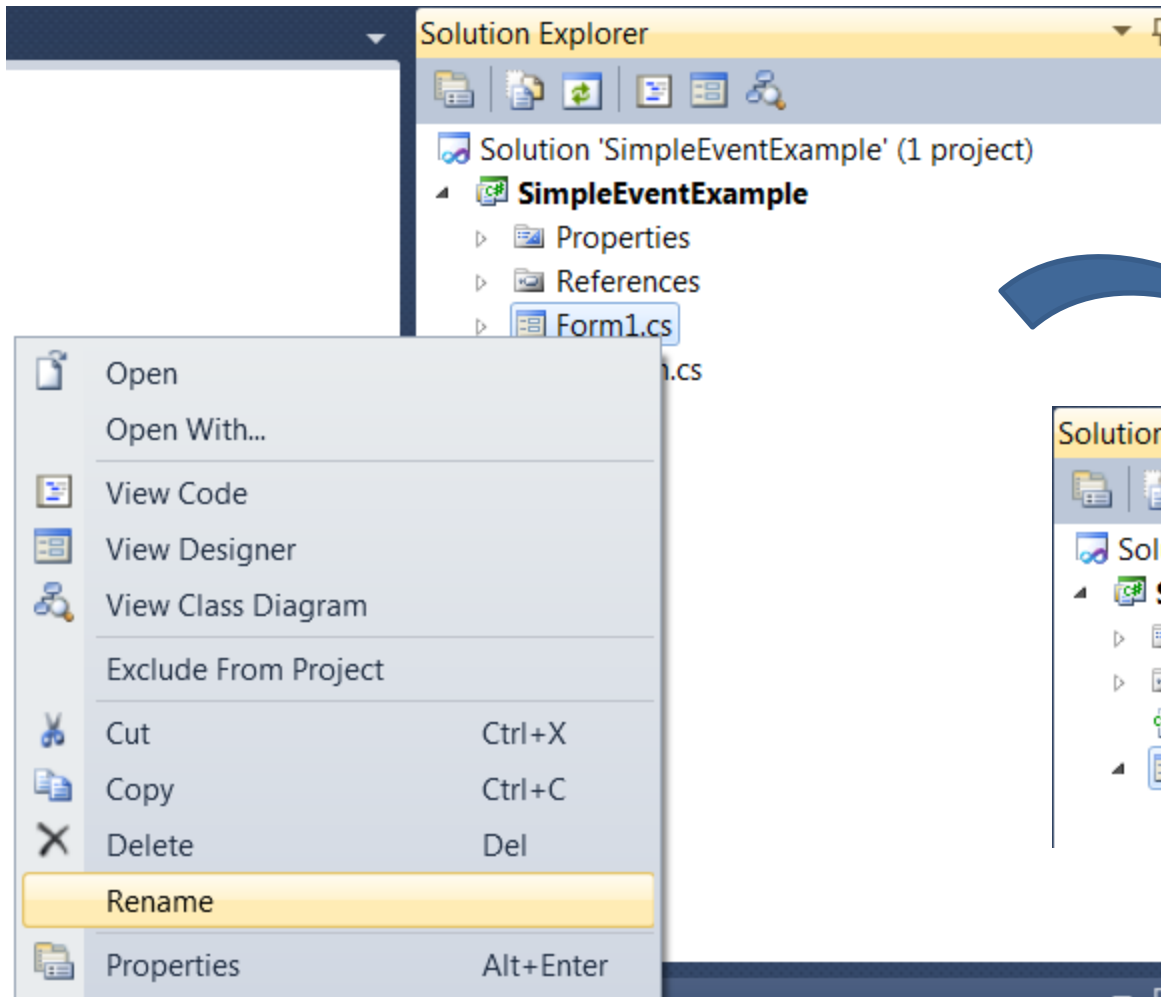
(Event-Handling)

- Bütün GUI kontrolleri ile ilişkili olaylar bulunmaktadır.
- Bir olay karşısında bir görev gerçekleştiren metot → **Olay-İşleyici (*Event-Handler*)**
- Olayı oluşturan kontrol → **Olay-Gönderici (*Event Sender*)**
 - Olay meydana geldiği zaman, görevi yerine getirecek olay-işleyiciyi çağırır.
- Olaylara cevap veren sürecin tamamına → **Olay İşleme (*Event Handling*)**
- .NET olay işleme düzeneği, olay-işleme metotlarının isimlerini seçmeye izin verir.

Örnek Olay Uygulaması



Örnek Olay Uygulaması



Örnek Olay Uygulaması

Properties

SimpleEventExample System.Windows.Forms.Form

AccessibleName	
AccessibleRole	Default
Appearance	
BackColor	<input type="checkbox"/> Control
BackgroundImage	<input type="checkbox"/> (none)
BackgroundImageLayout	Tile
Cursor	Default
Font	Microsoft Sans Serif; 7,8pt
ForeColor	<input checked="" type="checkbox"/> ControlText
FormBorderStyle	Sizable
RightToLeft	No
RightToLeftLayout	False
Text	Form1

Text
The text associated with the control.

Properties

SimpleEventExample System.Windows.Forms.Form

AccessibleName	
AccessibleRole	Default
Appearance	
BackColor	<input type="checkbox"/> Control
BackgroundImage	<input type="checkbox"/> (none)
BackgroundImageLayout	Tile
Cursor	Default
Font	Microsoft Sans Serif; 7,8pt
ForeColor	<input checked="" type="checkbox"/> ControlText
FormBorderStyle	Sizable
RightToLeft	No
RightToLeftLayout	False
Text	Simple Event Example

Text
The text associated with the control.

Örnek Olay Uygulaması

Properties

SimpleEventExample System.Windows.Forms.Form

Accessibility

AccessibleDescription	
AccessibleName	
AccessibleRole	Default

Appearance

BackColor	<input type="checkbox"/> Control
BackgroundImage	<input type="checkbox"/> (none)
BackgroundImageLayout	Tile
Cursor	Default
Font	Microsoft Sans Serif; 7,8p...
ForeColor	<input checked="" type="checkbox"/> ControlText
FormBorderStyle	Sizable
RightToLeft	No

Font

The font used to display text in the control.



Properties

SimpleEventExample System.Windows.Forms.Form

Accessibility

AccessibleDescription	
AccessibleName	
AccessibleRole	Default

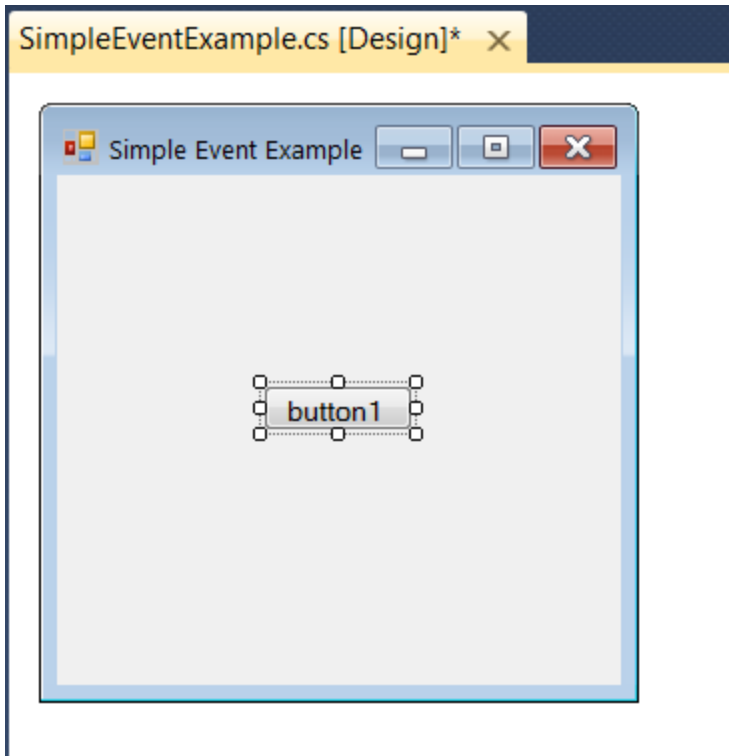
Appearance

BackColor	<input type="checkbox"/> Control
BackgroundImage	<input type="checkbox"/> (none)
BackgroundImageLayout	Tile
Cursor	Default
Font	Bradley Hand ITC; 12pt; ...
ForeColor	<input checked="" type="checkbox"/> ControlText
FormBorderStyle	Sizable
RightToLeft	No

Font

The font used to display text in the control.

Örnek Olay Uygulaması



Properties

button1 System.Windows.Forms.Button

Tag

Design

(Name)	clickButton
GenerateMember	True
Locked	False
Modifiers	Private

Focus

CausesValidation	True
------------------	------

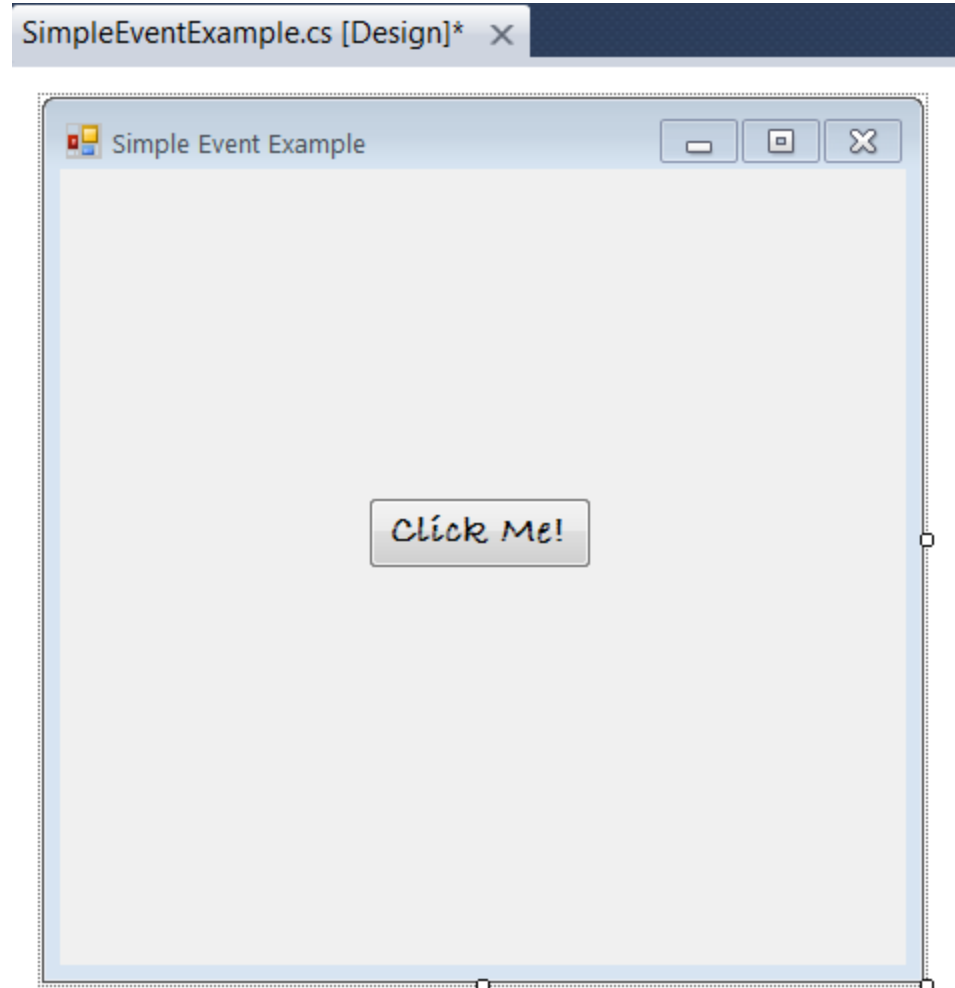
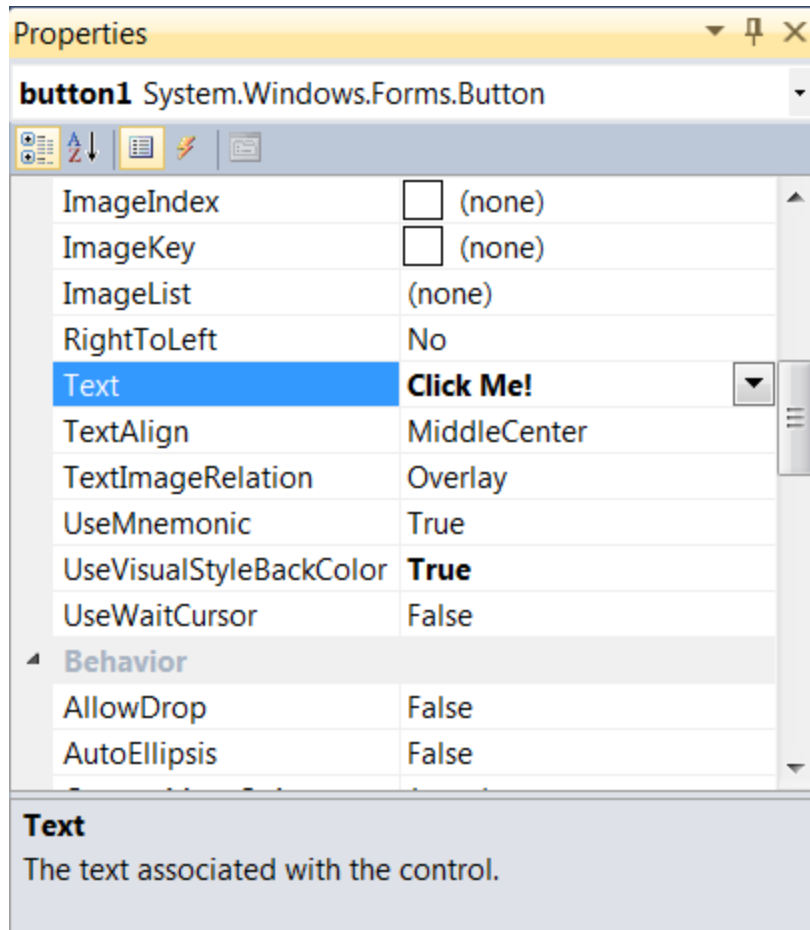
Layout

Anchor	Top, Left
AutoSize	False
AutoSizeMode	GrowOnly
Dock	None

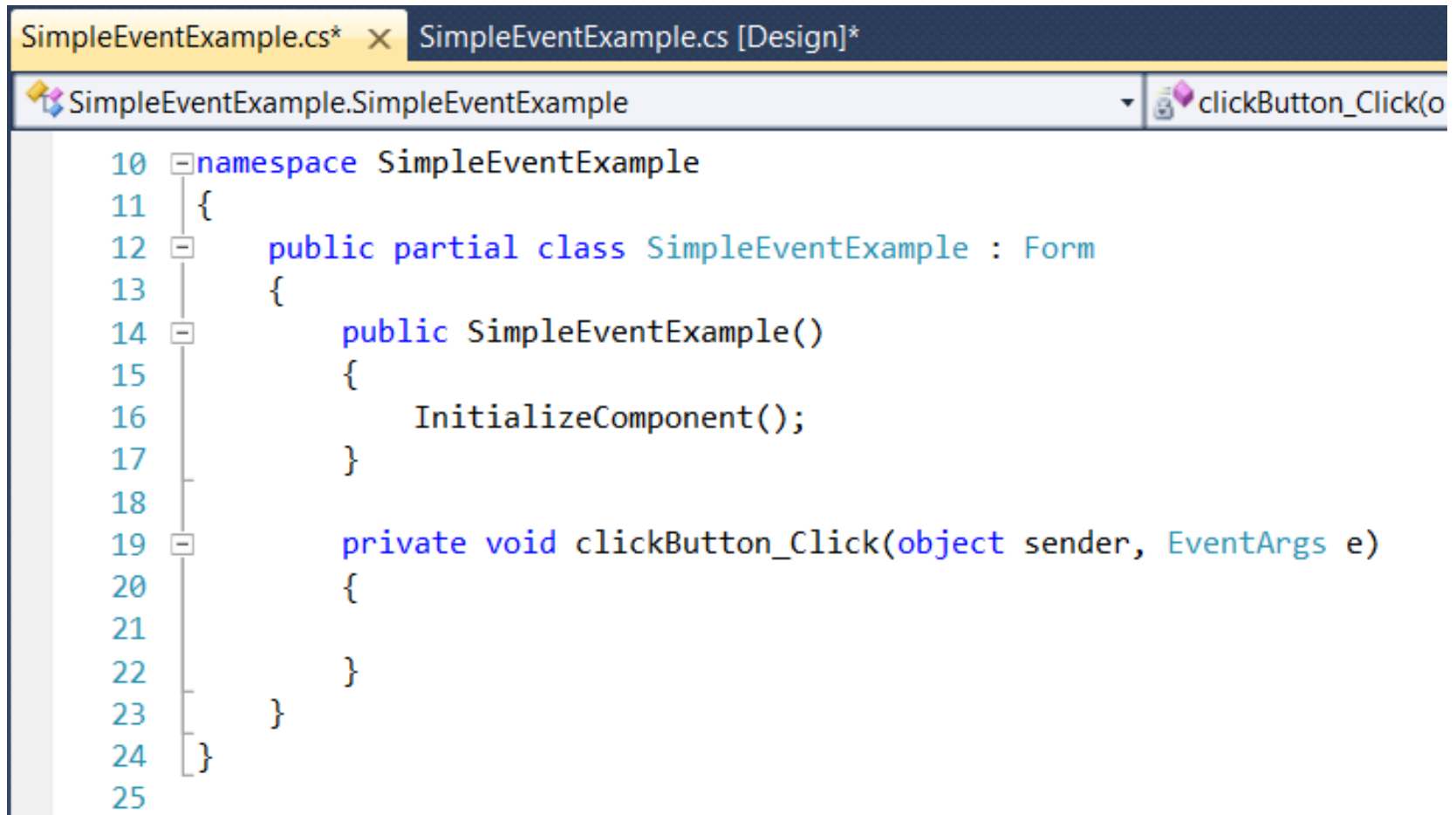
(Name)

Indicates the name used in code to identify the object.

Örnek Olay Uygulaması



Örnek Olay Uygulaması



```
10 namespace SimpleEventExample
11 {
12     public partial class SimpleEventExample : Form
13     {
14         public SimpleEventExample()
15         {
16             InitializeComponent();
17         }
18
19         private void clickButton_Click(object sender, EventArgs e)
20         {
21         }
22     }
23 }
24
25
```

Ad Uzayı (Namespace)

```
namespace SimpleEventExample
```

```
{  
    public partial class SimpleEventExample : Form  
    {  
        public SimpleEventExample()  
        {  
            InitializeComponent();  
        }  
  
        private void clickButton_Click(object sender, EventArgs e)  
        {  
        }  
    }  
}
```

- İlişkili sınıfların gruplanmasıdır.
- Tamamen nitelendirilmiş sınıf ismi (*Fully qualified class name*) → Her bir sınıf ismi, araya **nokta (.)** konularak, ad uzayı ismi ile bir bileşim oluşturur.

Ad Uzayı (Namespace)

```
namespace SimpleEventExample
```

- Nitelendirilmemiş sınıf ismi (*Unqualified class name*) → **SimpleEventExample**
 - Başka bir uygulamada bu sınıf kullanılmak istendiğinde:
 - **nitelendirilmiş sınıf ismi** `System.Console.WriteLine("Hello World!");`
 - ya da
 - **using ifadesi**
`using System;`
`Console.WriteLine("Hello World!");`
- kullanılır.

Aynı sınıf ismini kullanan başka bir ad uzayı var ise, nitelendirilmiş sınıf ismi **ad çakışmalarını (name collision/name conflict)** önler.

Örnek Olay Uygulaması

Kullanıcının
clickButton'a tıklaması



Event-handler metot

```
private void clickButton_Click(object sender, EventArgs e)
{
}
}
```

- İki parametre:

- **sender**

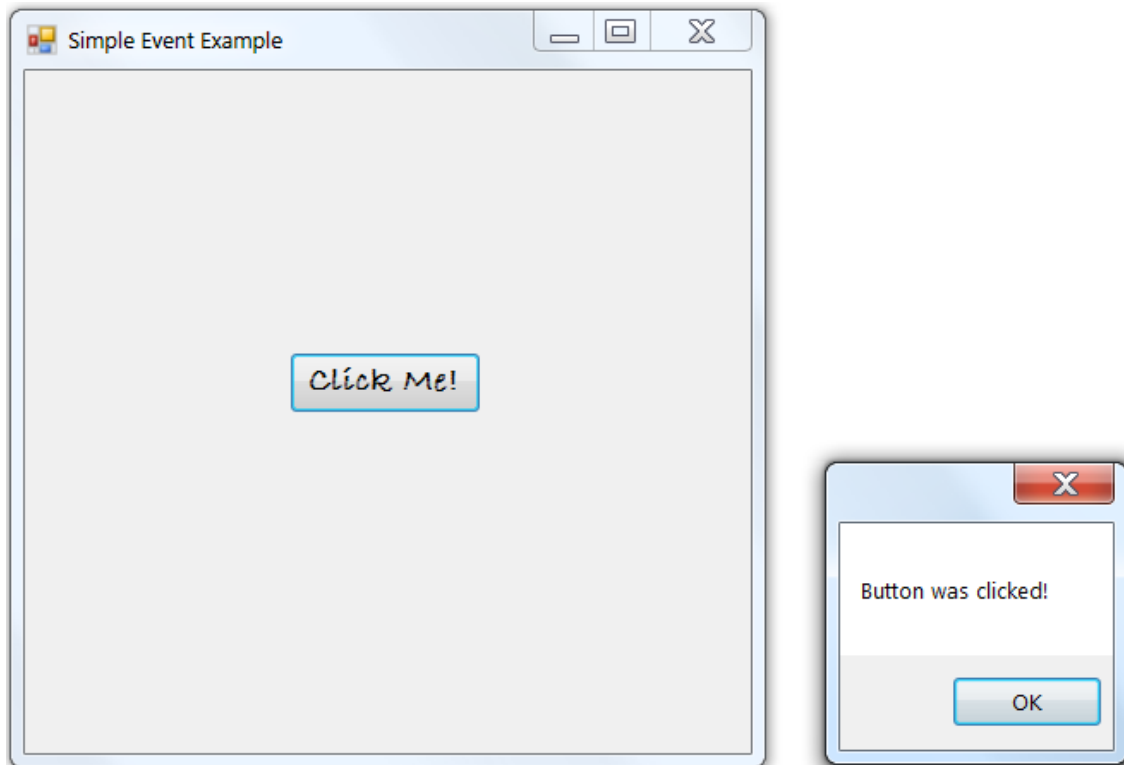
- **object** referansı
 - Olayı oluşturan nesneye referanstır.

- **e**

- Meydana gelen olay ile ilgili ek bilgiyi içerir.
 - **EventArgs** türündeki olay değişken nesnesine bir referanstır.
 - **EventArgs** → Olay bilgisini temsil eden bütün sınıfların ana sınıfıdır.

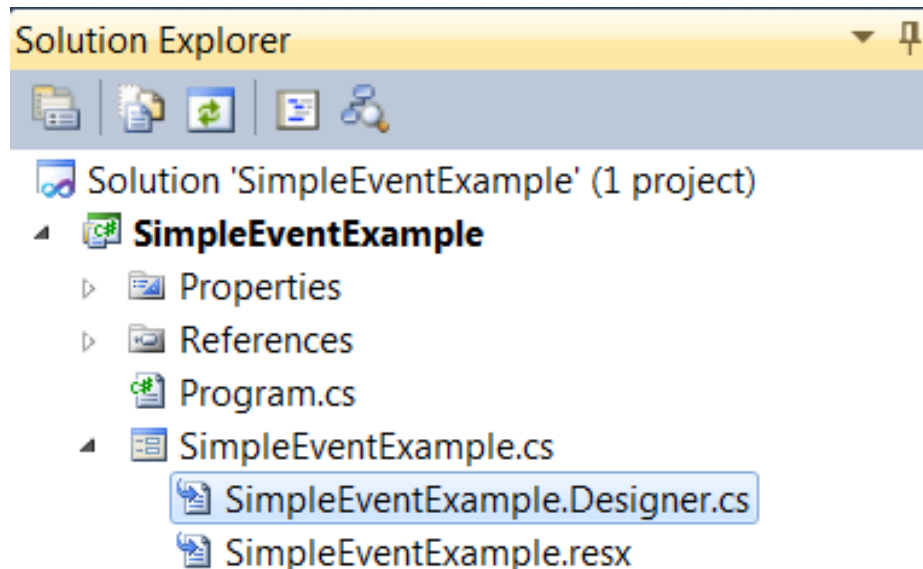
Örnek Olay Uygulaması

```
private void clickButton_Click(object sender, EventArgs e)
{
    MessageBox.Show("Button was clicked!");
}
```



Visual Studio GUI Kodu

- **Designer.cs** → Visual Studio tarafından otomatik olarak üretilmiş GUI kodu.
 - **SimpleEventExample.Designer.cs**



Visual Studio GUI Kodu

```
namespace SimpleEventExample
{
    partial class SimpleEventExample
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        Windows Form Designer generated code

        private System.Windows.Forms.Button clickButton;
    }
}
```

Sınıfın parçalara bölünmesi

Kaynakların serbest bırakılması

Visual Studio GUI Kodu

```
private void InitializeComponent()
{
    this.clickButton = new System.Windows.Forms.Button();
    this.SuspendLayout();
    //
    // clickButton
    //
    this.clickButton.Location = new System.Drawing.Point(154, 164);
    this.clickButton.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);
    this.clickButton.Name = "clickButton";
    this.clickButton.Size = new System.Drawing.Size(112, 36);
    this.clickButton.TabIndex = 0;
    this.clickButton.Text = "Click Me!";
    this.clickButton.UseVisualStyleBackColor = true;
    this.clickButton.Click += new System.EventHandler(this.clickButton_Click);
    //
    // SimpleEventExample
    //
    this.AutoScaleDimensions = new System.Drawing.SizeF(12F, 25F);
    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
    this.ClientSize = new System.Drawing.Size(423, 398);
    this.Controls.Add(this.clickButton);
    this.Font = new System.Drawing.Font("Bradley Hand ITC", 12F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, (byte)0);
    this.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);
    this.Name = "SimpleEventExample";
    this.Text = "Simple Event Example";
    this.ResumeLayout(false);
}
```

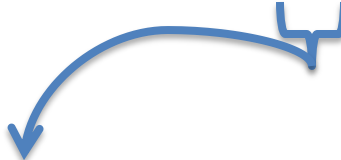
- Form yaratıldığında çağrılır.
- **Button**'un yaratılması
- **Button** ve **Form** özelliklerinin ayarlanması

Olay-İşleme (Event-Handling)

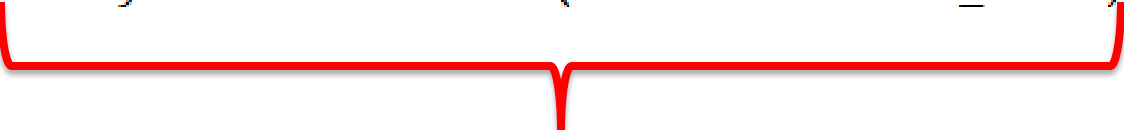
- **Delegate**

- Olay işleyicinin, kontrolün olayına bağlanmasıdır.
- İlgili metodu çağırır.
- **Multicast** → Bir olay için bir çok farklı metot çağrılabilir. (*Event Multicasting*)

```
this.clickButton.Click += new System.EventHandler(this.clickButton_Click);
```



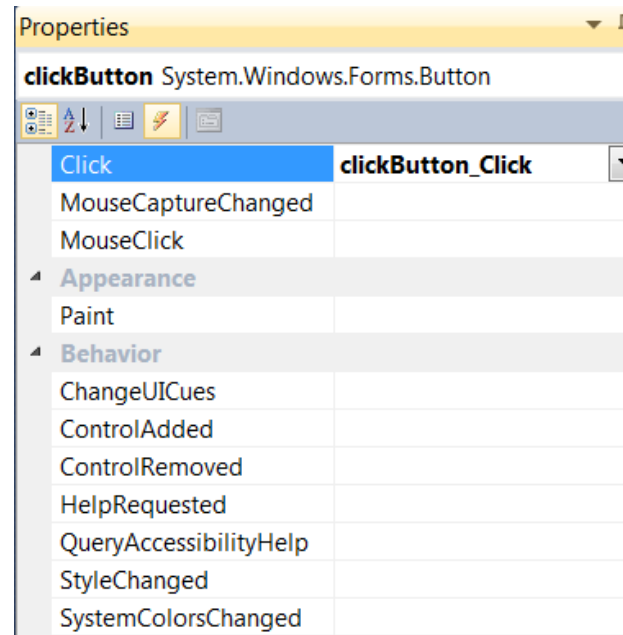
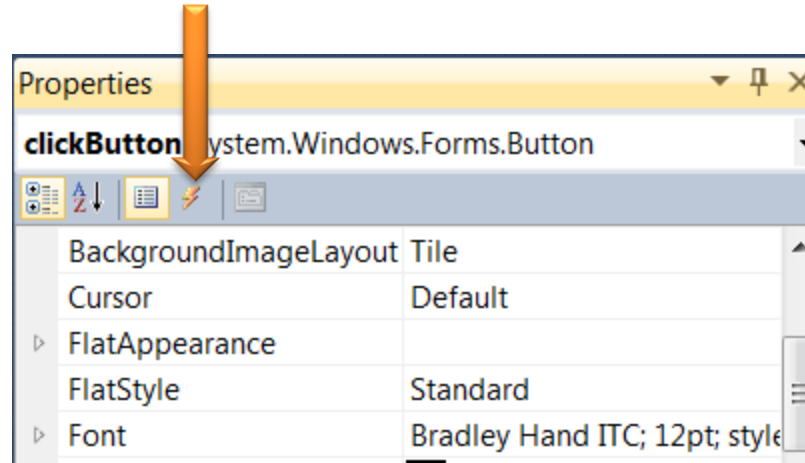
Delegate nesnesi
Button.Click olayına
eklenir.



Bir **EventHandler** nesnesi yaratılır ve bu
nesne **clickButton_Click** metodu ile
başlatılır (initialize).

Kullanıcı Button'a tıkladığında → `clickButton.Click` cevap verecektir

Olay-İşleyici Yaratma - 2

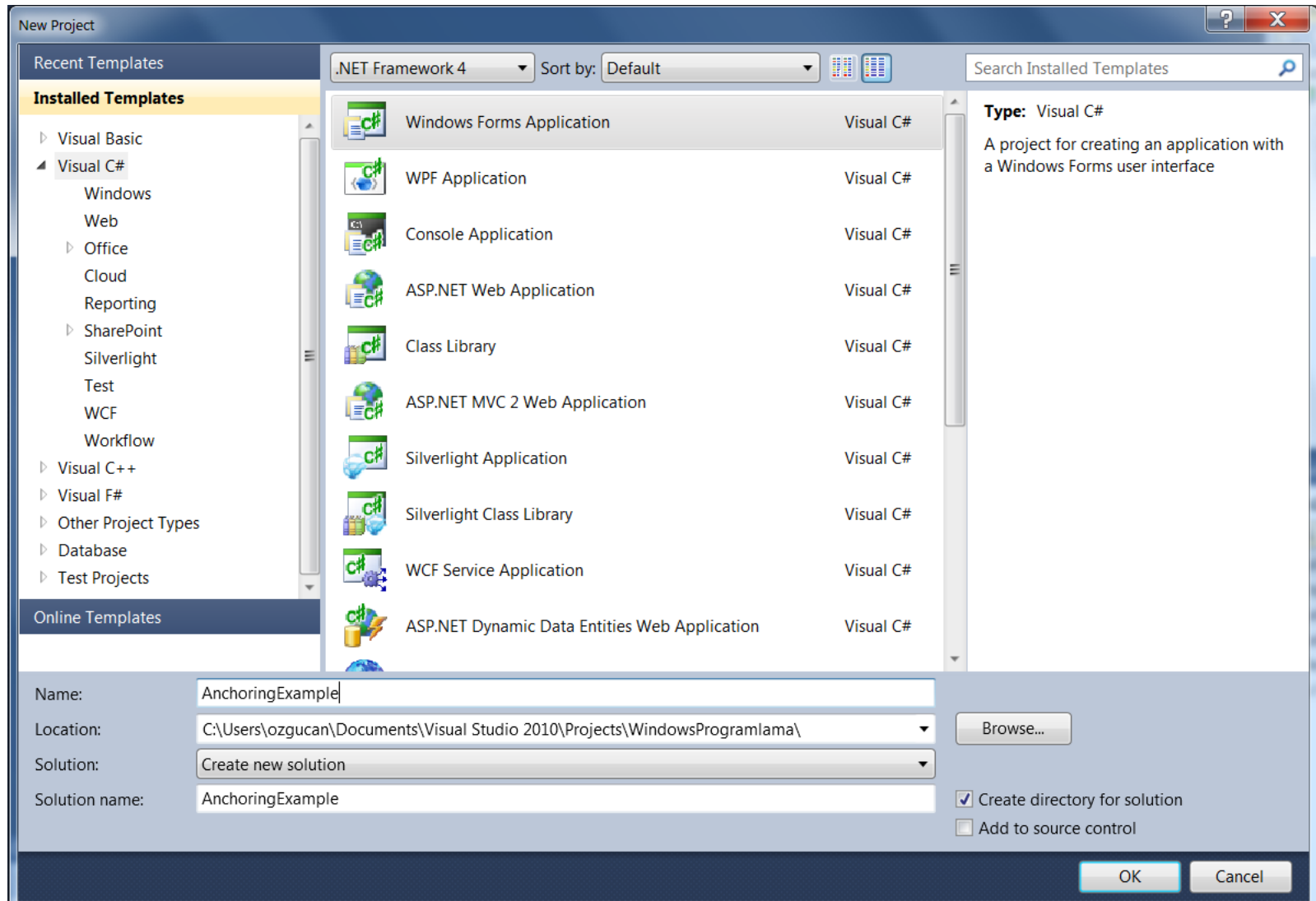


**Birden fazla
olay için tek bir
olay işleyici
tanımlanabilir.**

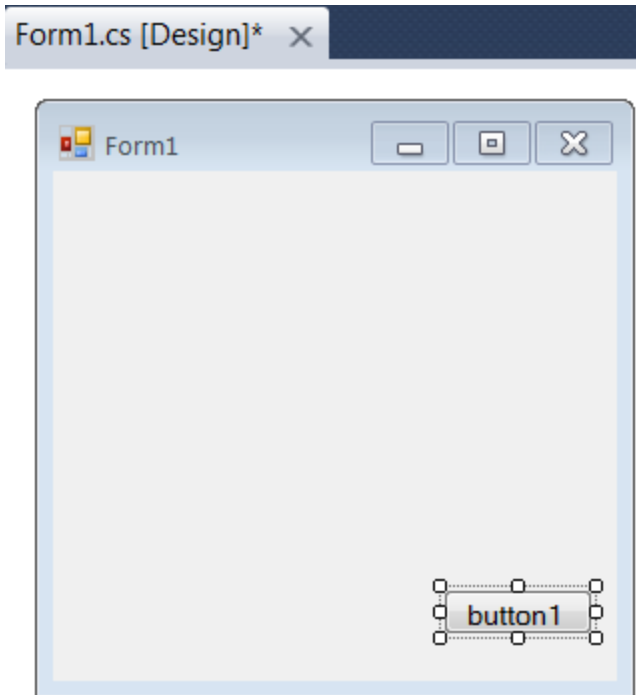
Kontrol Özellikleri

Class Control properties and methods	Description
<i>Common Properties</i>	
BackColor	The control's background color.
BackgroundImage	The control's background image.
Enabled	Specifies whether the control is enabled (i.e., if the user can interact with it). Typically, portions of a disabled control appear “grayed out” as a visual indication to the user that the control is disabled.
Focused	Indicates whether the control has the focus.
Font	The Font used to display the control's text.
ForeColor	The control's foreground color. This usually determines the color of the text in the Text property.
TabIndex	The tab order of the control. When the <i>Tab</i> key is pressed, the focus transfers between controls based on the tab order. You can set this order.
TabStop	If true, then a user can give focus to this control via the <i>Tab</i> key.
Text	The text associated with the control. The location and appearance of the text vary depending on the type of control.
Visible	Indicates whether the control is visible.
<i>Common Methods</i>	
Hide	Hides the control (sets the Visible property to false).
Select	Acquires the focus.
Show	Shows the control (sets the Visible property to true).

Uygulama (Anchoring)



Uygulama (Anchoring)



Properties

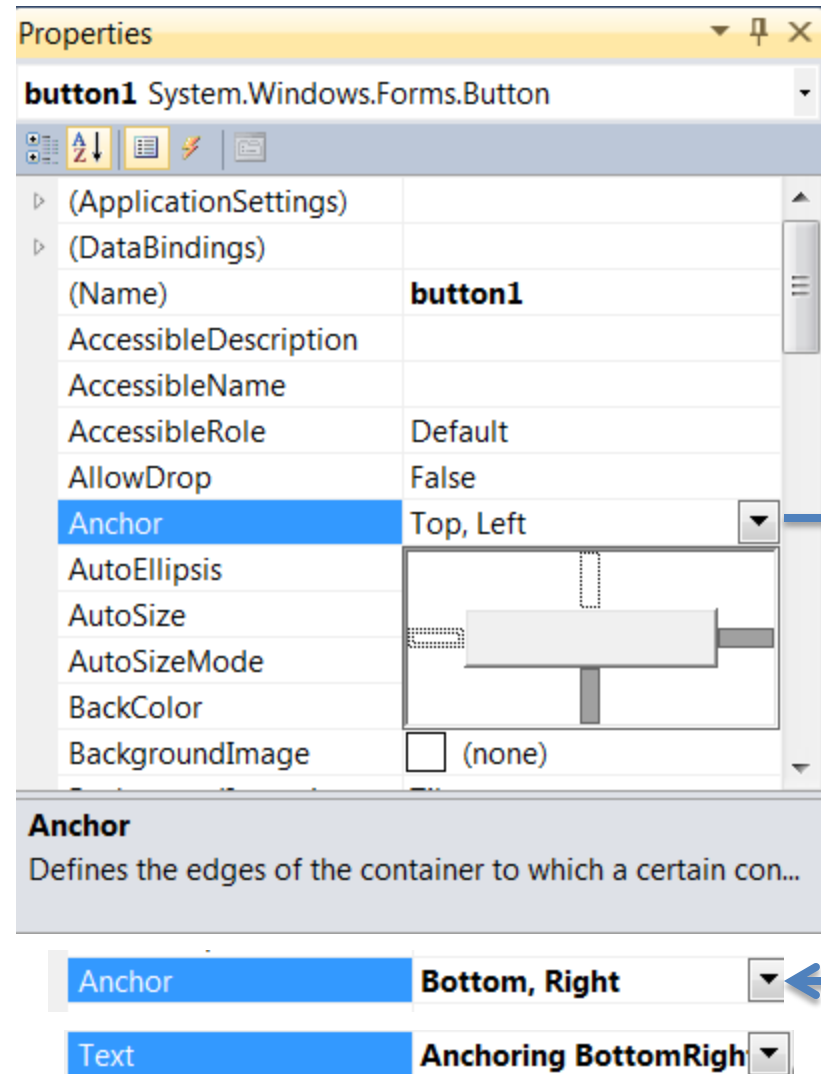
button1 System.Windows.Forms.Button

(ApplicationSettings)	
(DataBindings)	
(Name)	button1
AccessibleDescription	
AccessibleName	
AccessibleRole	Default
AllowDrop	False
Anchor	Top, Left
AutoEllipsis	
AutoSize	
AutoSizeMode	
BackColor	
BackgroundImage	(none)

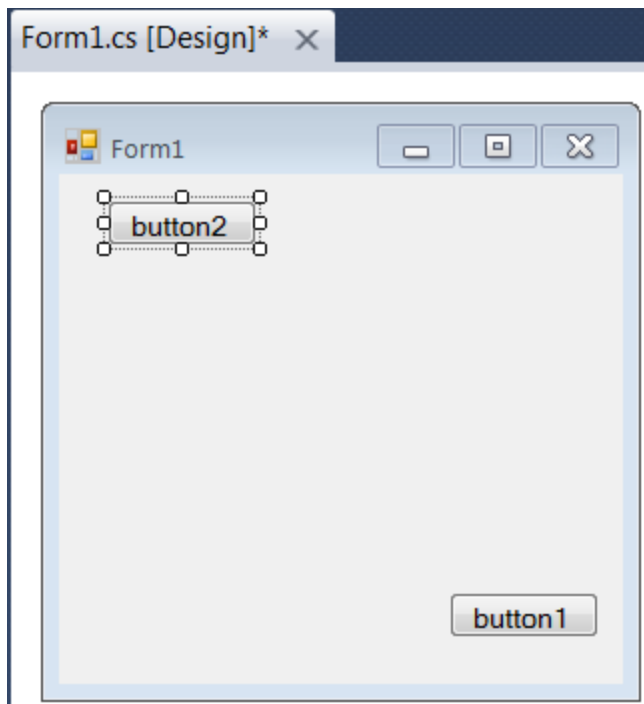
Anchor
Defines the edges of the container to which a certain con...

Anchor Bottom, Right

Text Anchoring BottomRight



Uygulama (Anchoring)



Properties

button2 System.Windows.Forms.Button

(ApplicationSettings)

(DataBindings)

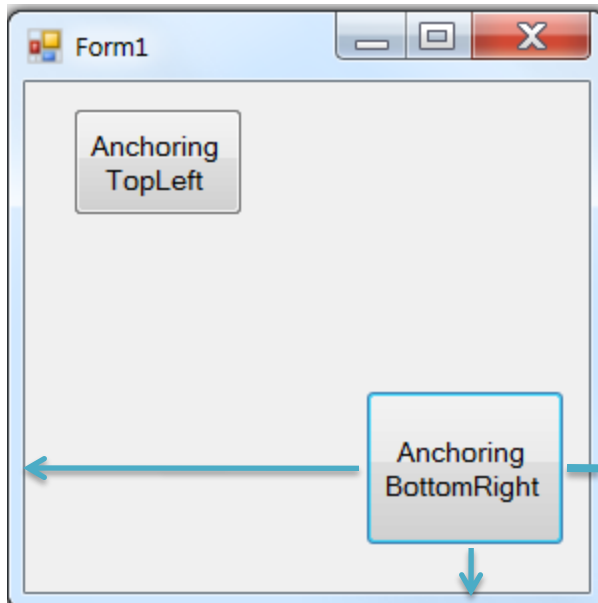
(Name)	button2
AccessibleDescription	
AccessibleName	
AccessibleRole	Default
AllowDrop	False
Anchor	Top, Left
AutoEllipsis	False
AutoSize	False
AutoSizeMode	GrowOnly
BackColor	<input type="checkbox"/> Control
BackgroundImage	<input type="checkbox"/> (none)

Anchor
Defines the edges of the container to which a certain con...

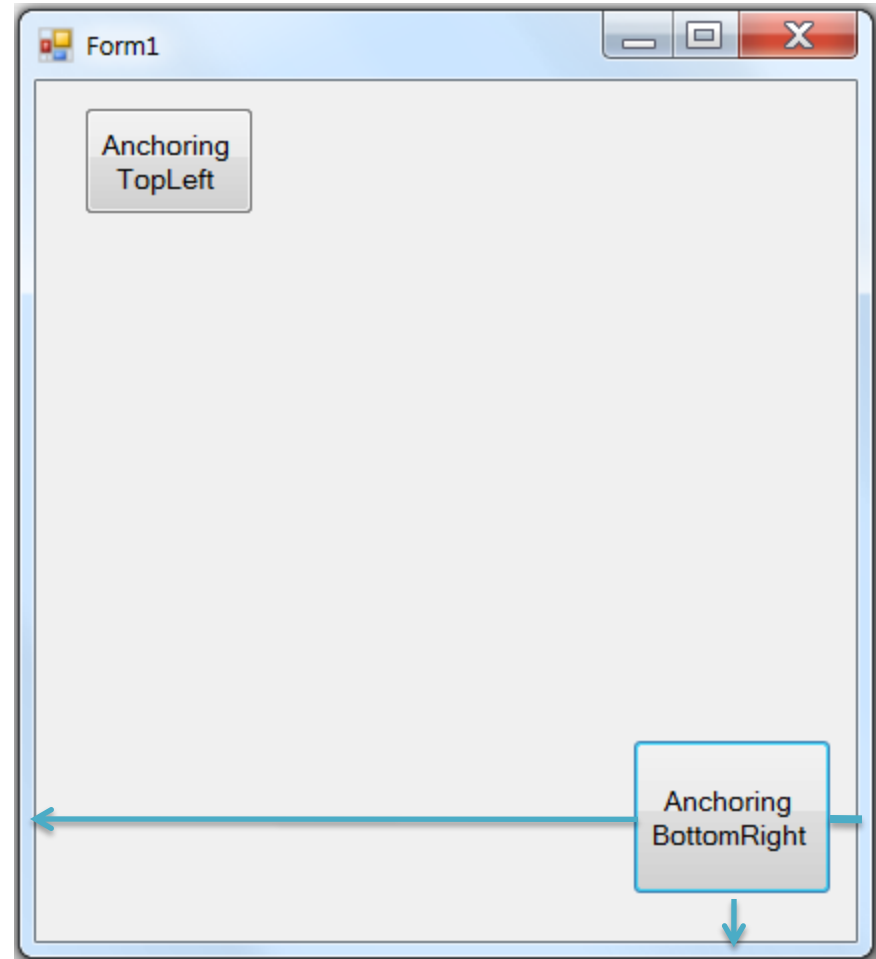
Text

Anchoring TopLeft

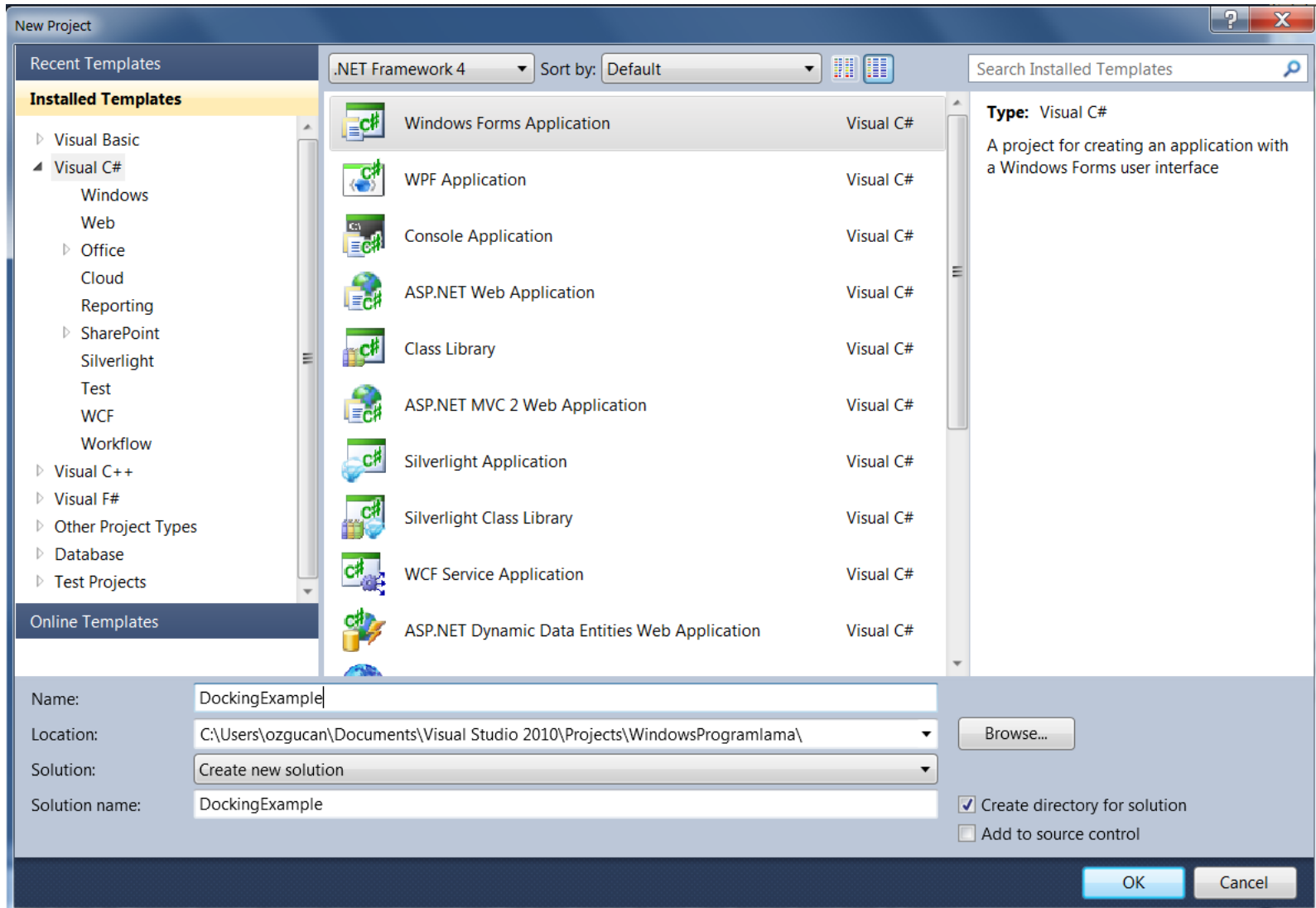
Uygulama (Anchoring)



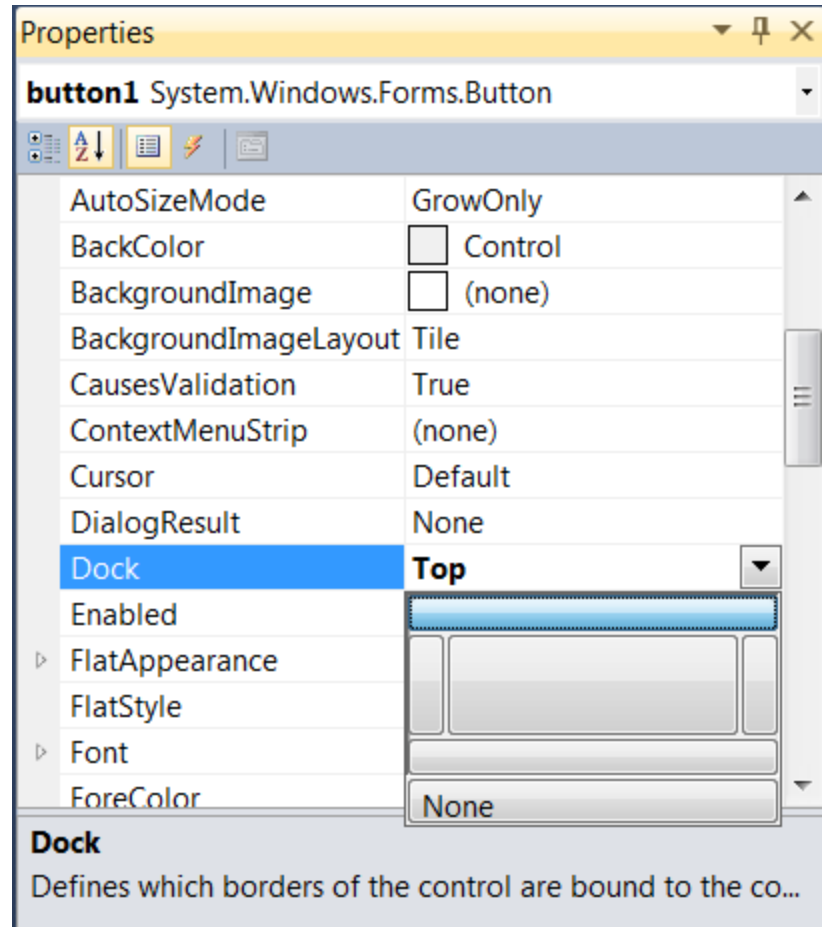
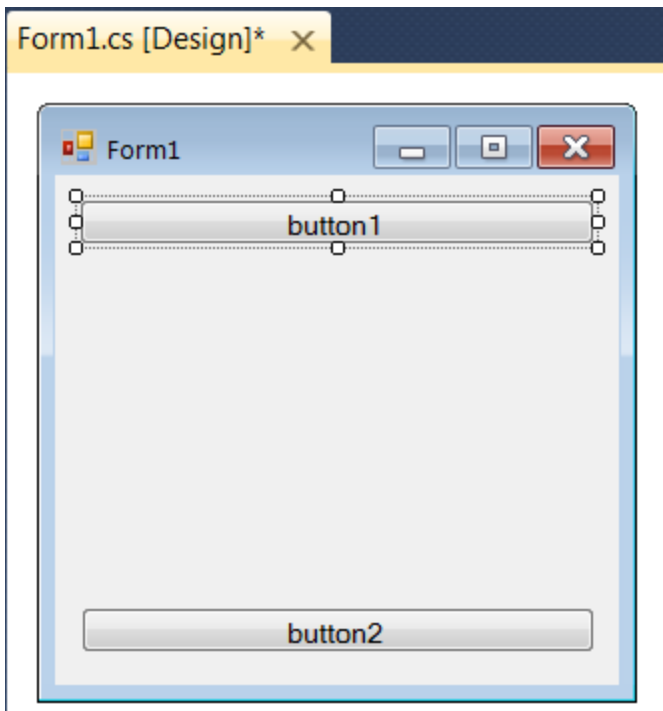
Kenarlara uzaklık
sabit



Uygulama (Docking)

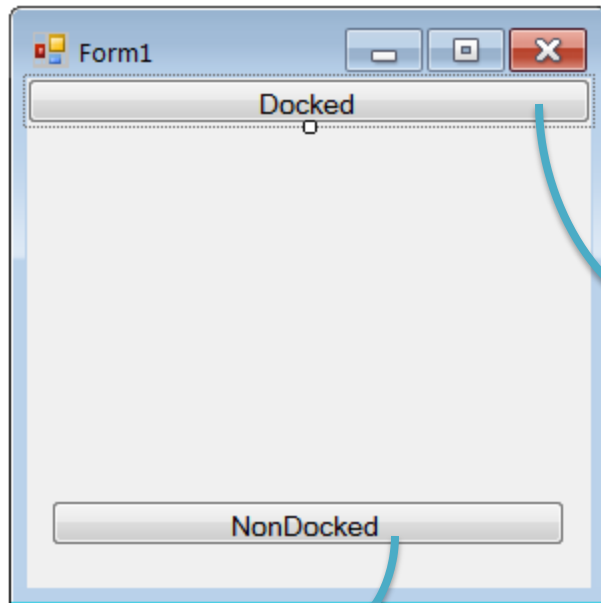


Uygulama (Docking)



Uygulama (Docking)

Form1.cs [Design] x



Properties

button1 System.Windows.Forms.Button

Location	0; 0
Locked	False
Margin	3; 3; 3; 3
MaximumSize	0; 0
MinimumSize	0; 0
Modifiers	Private
Padding	0; 0; 0; 0
RightToLeft	No
Size	282; 23
TabIndex	0
TabStop	True
Tag	
Text	Docked
TextAlign	MiddleCenter

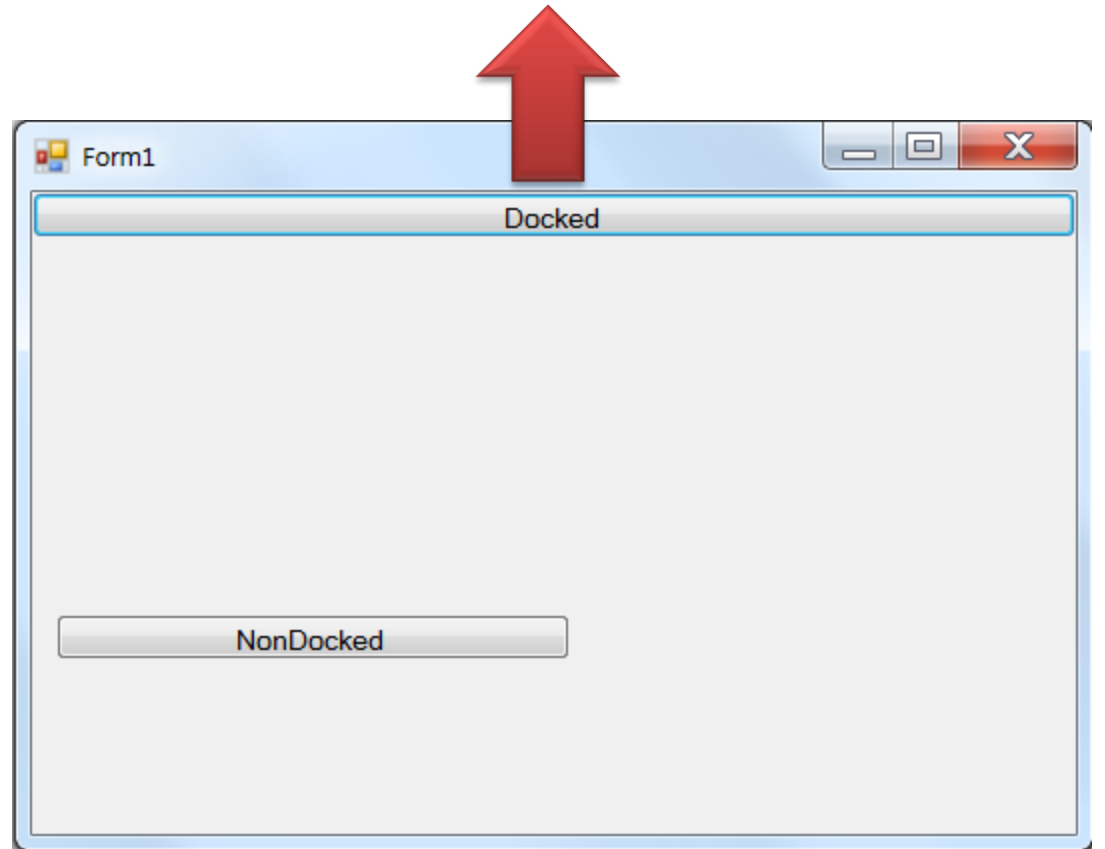
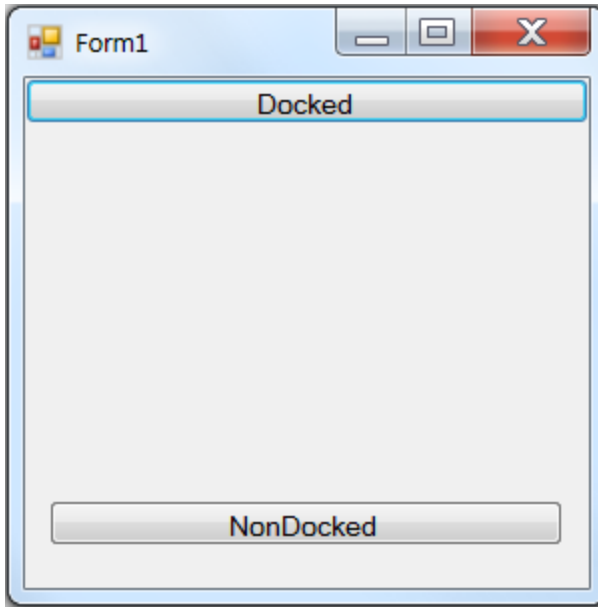
Text
The text associated with the control.

Text

NonDocked

Uygulama (Docking)

Sayfa boyunca uzar.



Kontrol Yerleşim Özellikleri

Control layout properties	Description
Anchor	Causes a control to remain at a fixed distance from the side(s) of the container even when the container is resized.
Dock	Allows a control to span one side of its container or to fill the remaining space in the container.
Padding	Sets the space between a container's edges and docked controls. The default is 0, causing the control to appear flush with the container's sides.
Location	Specifies the location (as a set of coordinates) of the upper-left corner of the control, in relation to its container's upper-left corner.
Size	Specifies the size of the control in pixels as a Size object, which has properties Width and Height.
MinimumSize, MaximumSize	Indicates the minimum and maximum size of a Control, respectively.

Label

- Kullanıcının direkt olarak değiştiremeyeceği metni görüntüler.

Common Label properties	Description
Font	The font of the text on the Label.
Text	The text on the Label.
TextAlign	The alignment of the Label's text on the control—horizontally (left, center or right) and vertically (top, middle or bottom). The default is top, left.

TextBox

- Program tarafından görüntülenen metnin ya da kullanıcının klavye aracılığı ile gireceği metnin bulunduğu alandır.
- Şifre için **TextBox** kullanımı:
 - Kullanıcının girdiği şifre karakterlerini maskeler.
 - **UseSystemPasswordChar → True**

TextBox

TextBox properties and an event	Description
<i>Common Properties</i>	
AcceptsReturn	If true in a multiline TextBox, pressing <i>Enter</i> in the TextBox creates a new line. If false (the default), pressing <i>Enter</i> is the same as pressing the default Button on the Form. The default Button is the one assigned to a Form's AcceptButton property.
Multiline	If true, the TextBox can span multiple lines. The default value is false.
ReadOnly	If true, the TextBox has a gray background, and its text cannot be edited. The default value is false.
ScrollBars	For multiline textboxes, this property indicates which scrollbars appear (None—the default, Horizontal, Vertical or Both).
Text	The TextBox's text content.
UseSystemPasswordChar	When true, the TextBox becomes a password TextBox, and the system-specified character masks each character the user types.
<i>Common Event</i>	
TextChanged	Generated when the text changes in a TextBox (i.e., when the user adds or deletes characters). When you double click the TextBox control in Design mode, an empty event handler for this event is generated.

Button

- Kullanıcının uygulamada komut vermesi
 - **CheckBox, RadioButton, Button, vs..**

Button properties and an event

Description

Common Properties

Text

Specifies the text displayed on the Button face.

FlatStyle

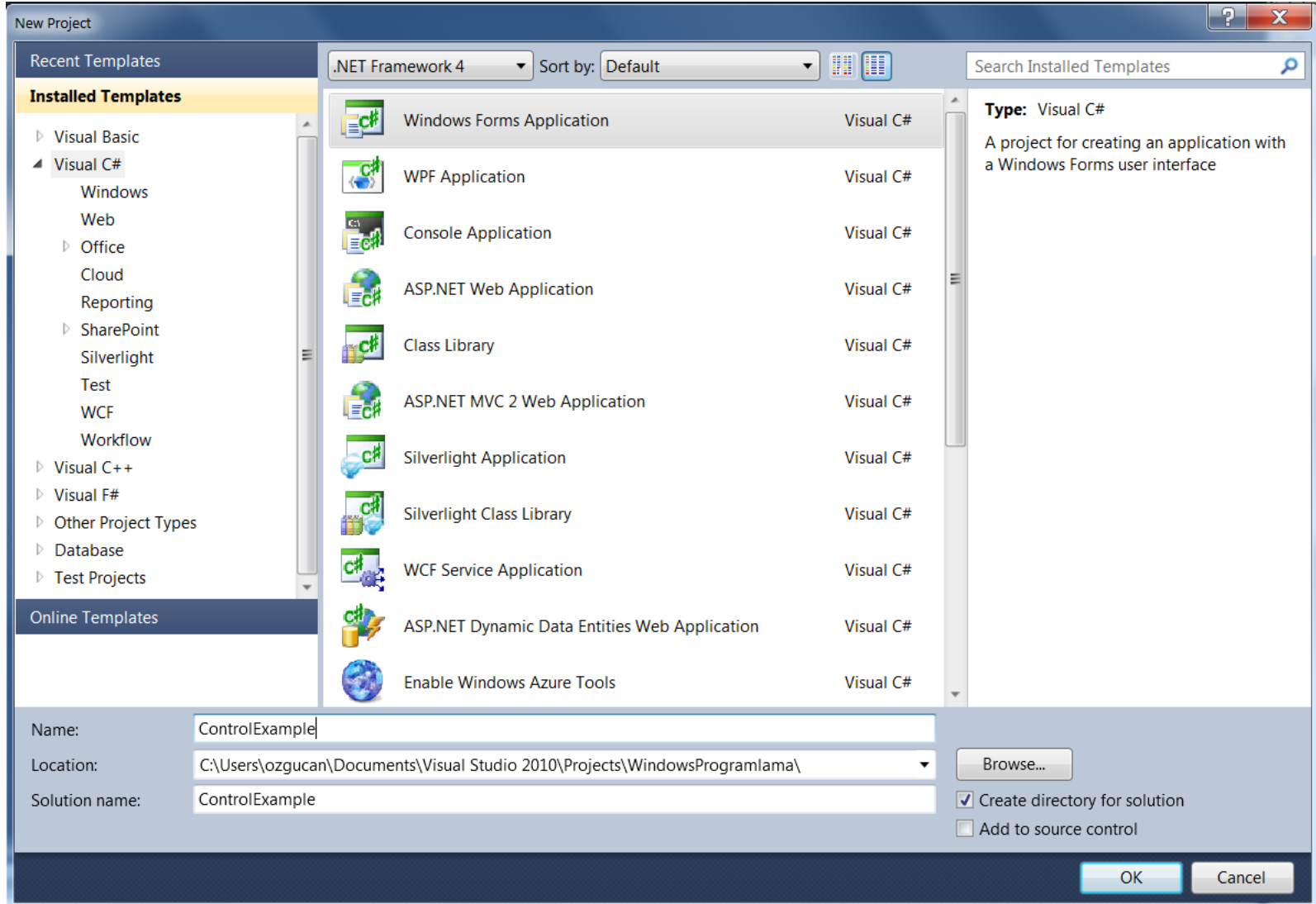
Modifies a Button's appearance—attribute Flat (for the Button to display without a three-dimensional appearance), Popup (for the Button to appear flat until the user moves the mouse pointer over the Button), Standard (three-dimensional) and System, where the Button's appearance is controlled by the operating system. The default value is Standard.

Common Event

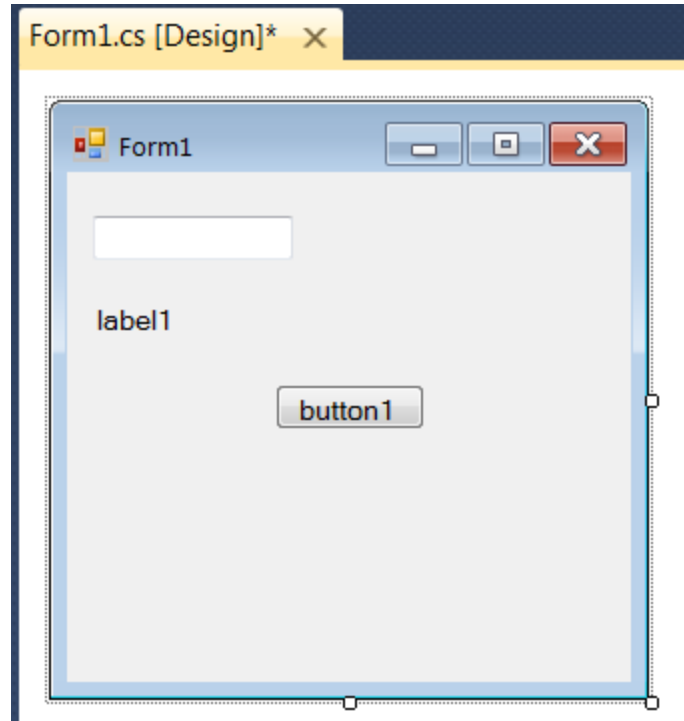
Click

Generated when the user clicks the Button. When you double click a Button in design view, an empty event handler for this event is created.

Örnek Uygulama (Kontrollerin Kullanımı)



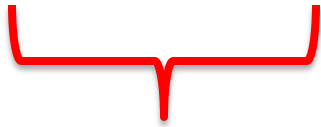
Örnek Uygulama (Kontrollerin Kullanımı)



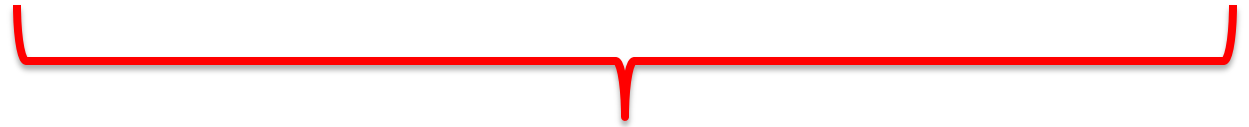
Örnek Uygulama

(Kontrollerin Kullanımı)

- **TextBox** → `inputPasswordTextBox`
- **Label** → `displayPasswordLabel`
- **Button** → `displayPasswordButton`



Kontroller

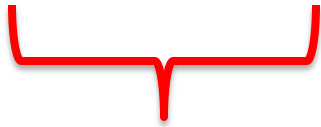


Kontrollerin name özelliği

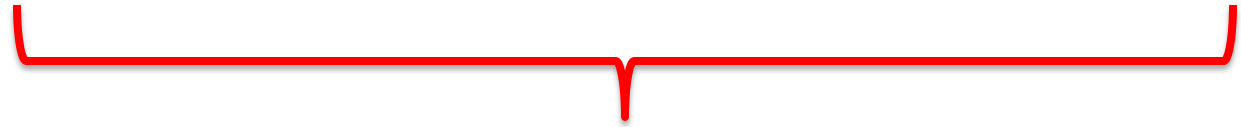
Örnek Uygulama

(Kontrollerin Kullanımı)

- **Form** → Display Password
- **Label** → - (Clear)
- **Button** → Display



Kontroller



Kontrollerin Text özelliği

Örnek Uygulama

(Kontrollerin Kullanımı)

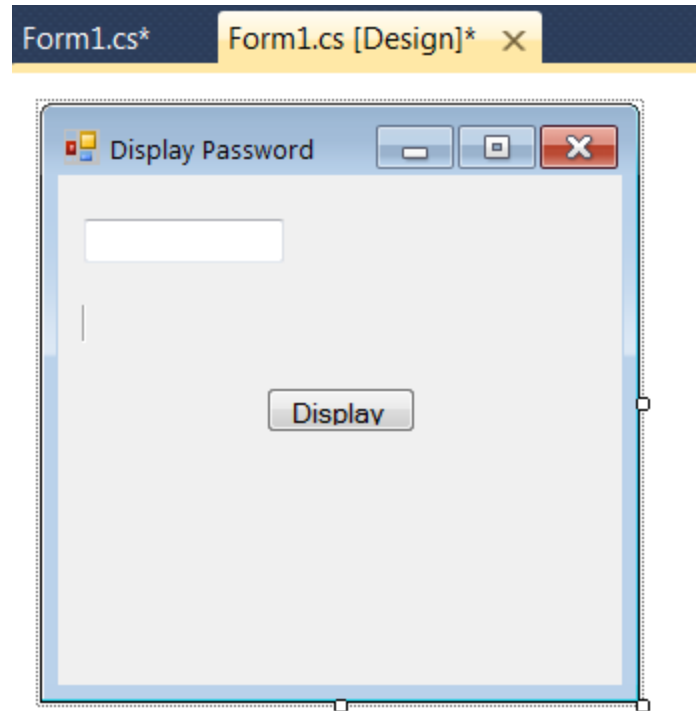
- **Label :**

- `BorderStyle = Fixed3D`
- `TextAlign = MiddleLeft`

- **TextBox :**

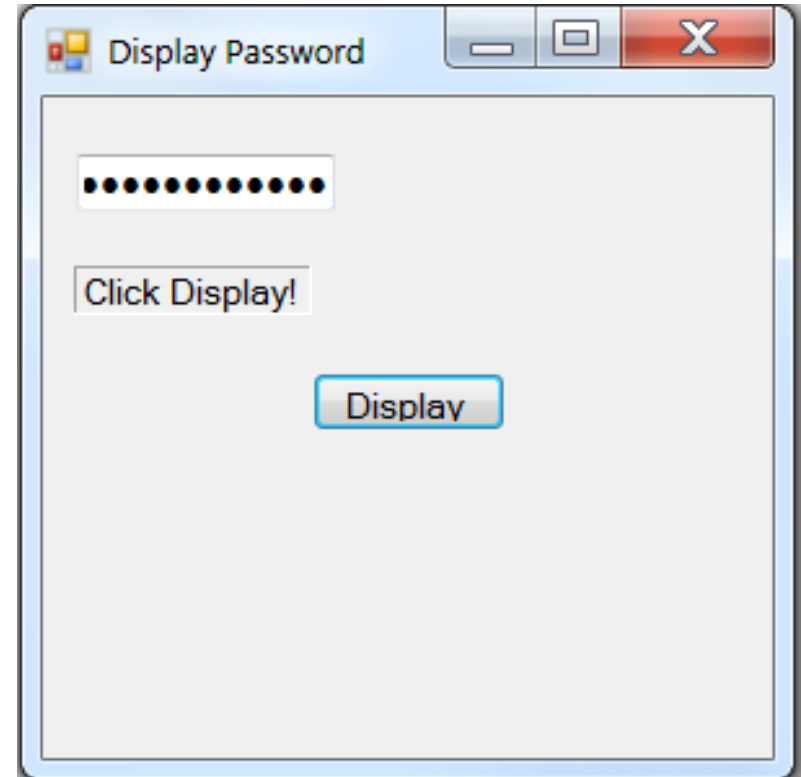
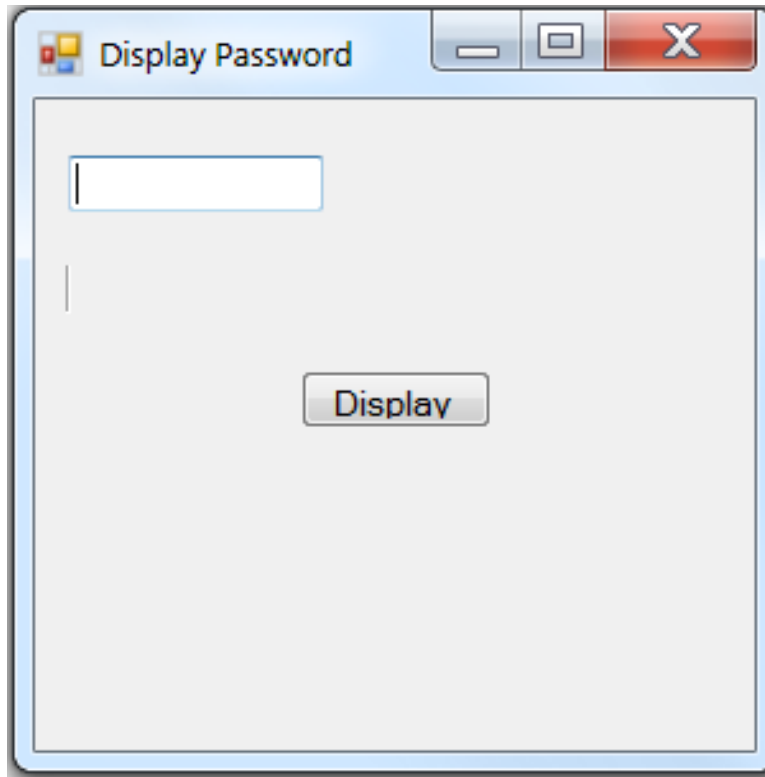
- `UseSystemPassswordChar = True`

Örnek Uygulama (Kontrollerin Kullanımı)



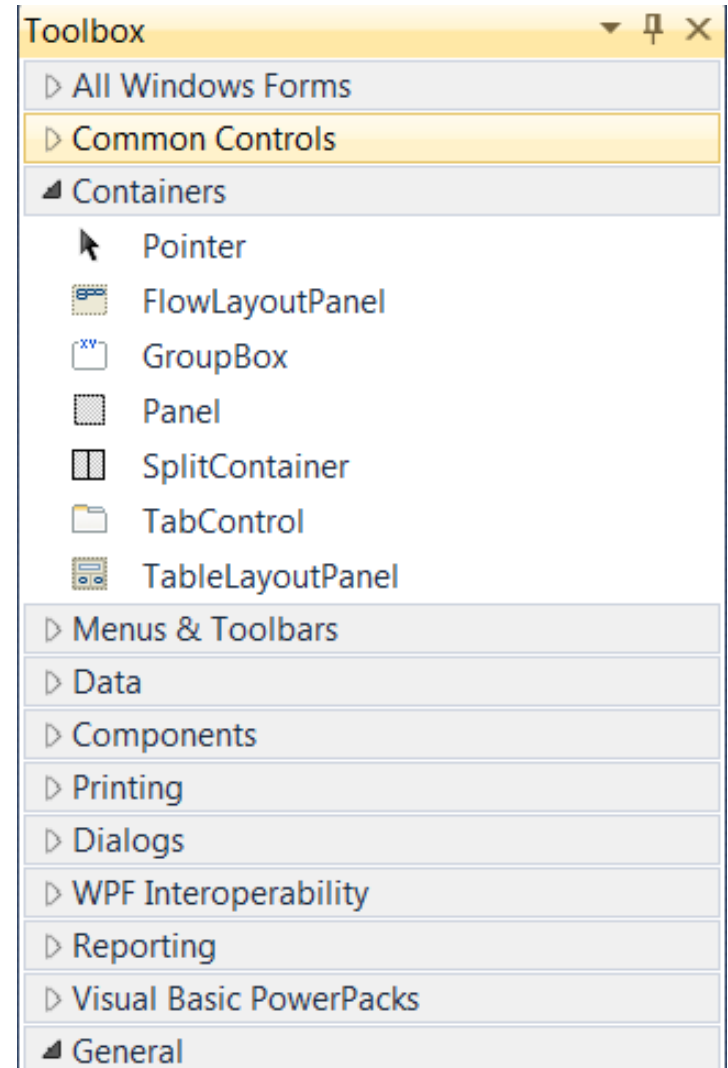
```
private void displayPasswordButton_Click(object sender, EventArgs e)
{
    displayPasswordLabel.Text = inputPasswordTextBox.Text;
}
```

Örnek Uygulama (Kontrollerin Kullanımı)



GroupBox & Panel

- GUI kontrollerini düzenler.
- Kontrolleri gruplar.
- GroupBox ve Panel taşınırken, kontroller de onlarla birlikte taşınır.
- Kontrollerin **aynı anda** *görüntülenmesi* ya da *gizlenmesi* için de kullanılır.
 - **Visible** özelliği



GroupBox & Panel

GroupBox

- Başlık **görüntüler**.
- Kaydırma çubuğu (scrollbar) **yoktur**.
- Kenarlar (borders) **varsayılan (default)** değer olarak **incedir**.

Panel

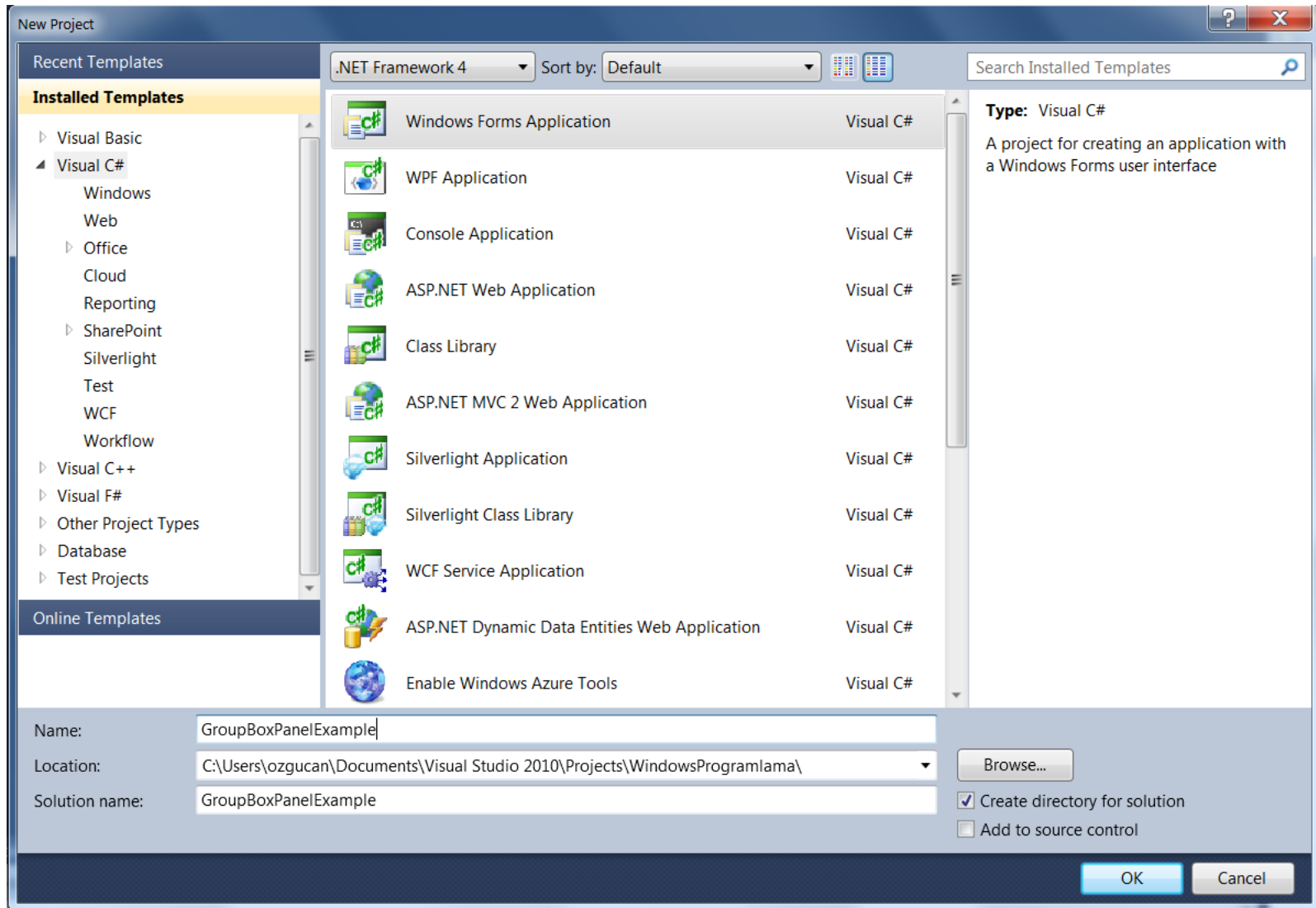
- Başlık **görüntülemez**.
- Kaydırma çubuğu (scrollbar) **vardır**.
- Kenar ayarlamaları:
 - **BorderStyle** özelliği

GroupBox & Panel

GroupBox properties	Description
Controls	The set of controls that the GroupBox contains.
Text	Specifies the caption text displayed at the top of the GroupBox.

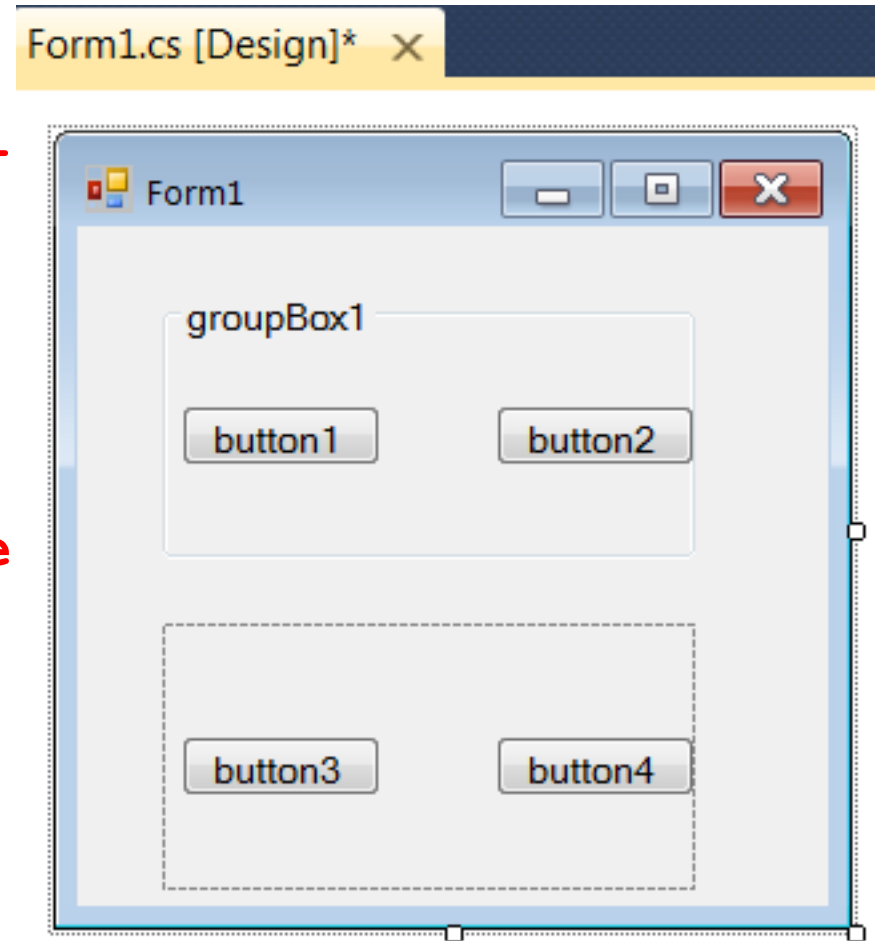
Panel properties	Description
AutoScroll	Indicates whether scrollbars appear when the Panel is too small to display all of its controls. The default value is false.
BorderStyle	Sets the border of the Panel. The default value is None; other options are Fixed3D and FixedSingle.
Controls	The set of controls that the Panel contains.

Örnek Uygulama (GroupBox & Panel)



Örnek Uygulama (GroupBox & Panel)

- **Form**
 - Text = **GroupBox & Panel Example**
- **GroupBox**
 - 2 Button
 - Text = **GroupBox Example**
- **Panel**
 - 2 Button
- **Label**



Örnek Uygulama

(GroupBox & Panel)

- **Button1 & Button3**
 - Text = **OK**
 - Name = **groupBoxButton1**
 - Name = **panelButton1**
- **Button2 & Button4**
 - Text = **Cancel**
 - Name = **groupBoxButton2**
 - Name = **panelButton2**
- **Panel**
 - AutoScroll = **True**
- **Label**
 - Text → **Blank**
 - Name = **messageLabel**

Örnek Uygulama

(GroupBox & Panel)

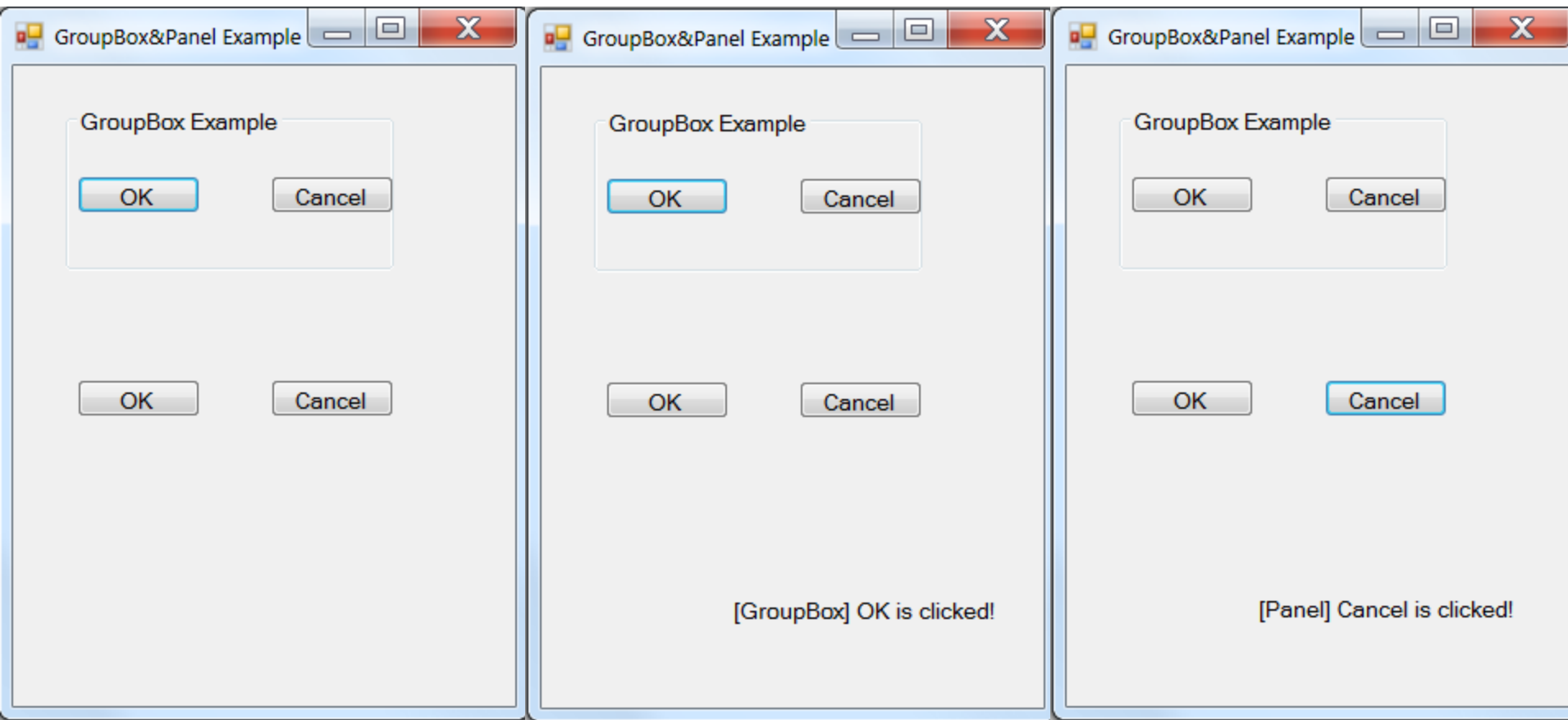
```
private void groupBoxButton1_Click(object sender, EventArgs e)
{
    messageLabel.Text = "[GroupBox] OK is clicked!";
}
```

```
private void groupBoxButton2_Click(object sender, EventArgs e)
{
    messageLabel.Text = "[GroupBox] Cancel is clicked!";
}
```

```
private void panelButton1_Click(object sender, EventArgs e)
{
    messageLabel.Text = "[Panel] OK is clicked!";
}
```

```
private void panelButton2_Click(object sender, EventArgs e)
{
    messageLabel.Text = "[Panel] Cancel is clicked!";
}
```

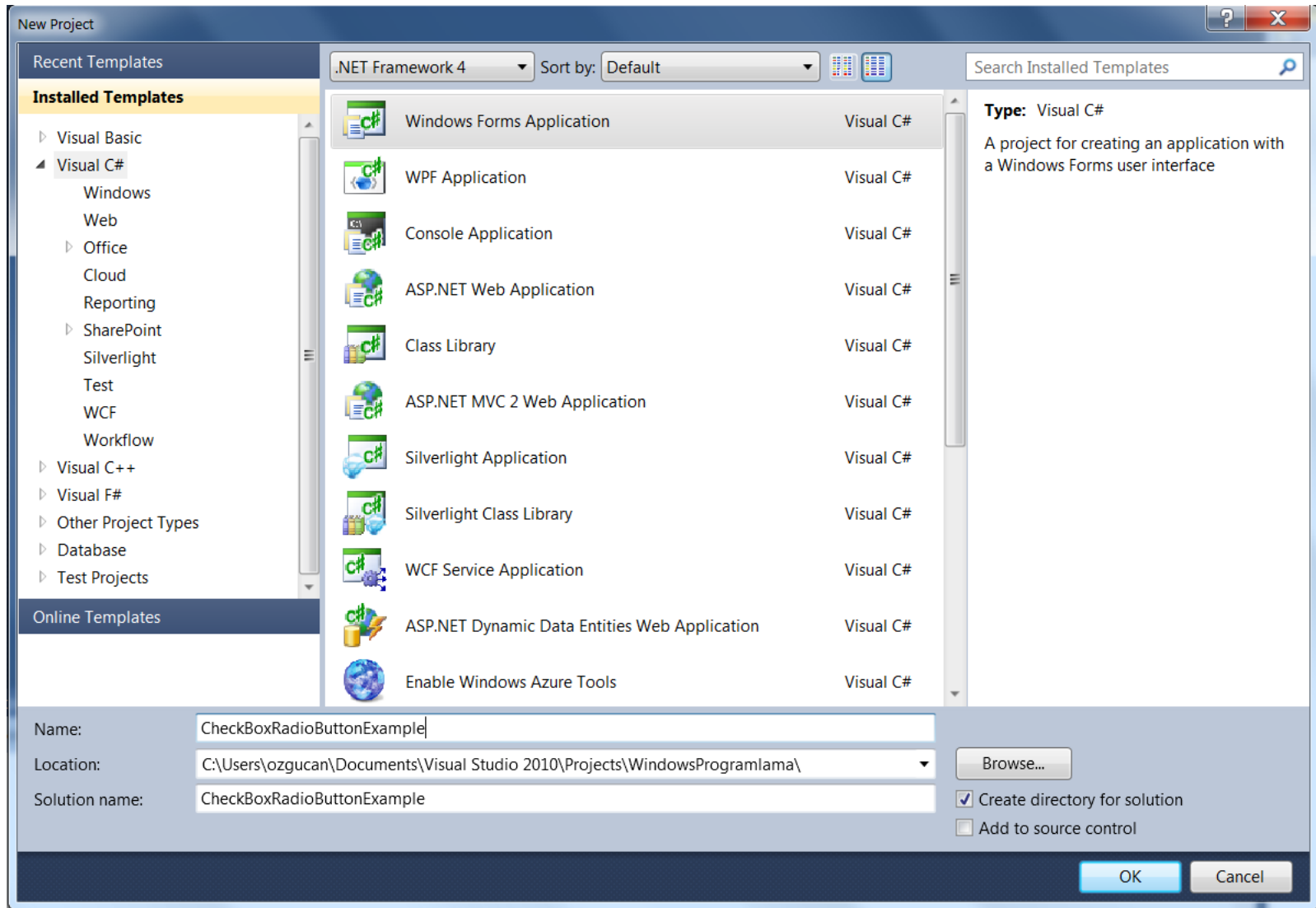
Örnek Uygulama (GroupBox & Panel)



CheckBox & RadioButton

- Durum button'u
 - On/Off
 - True/False

CheckBox & RadioButton



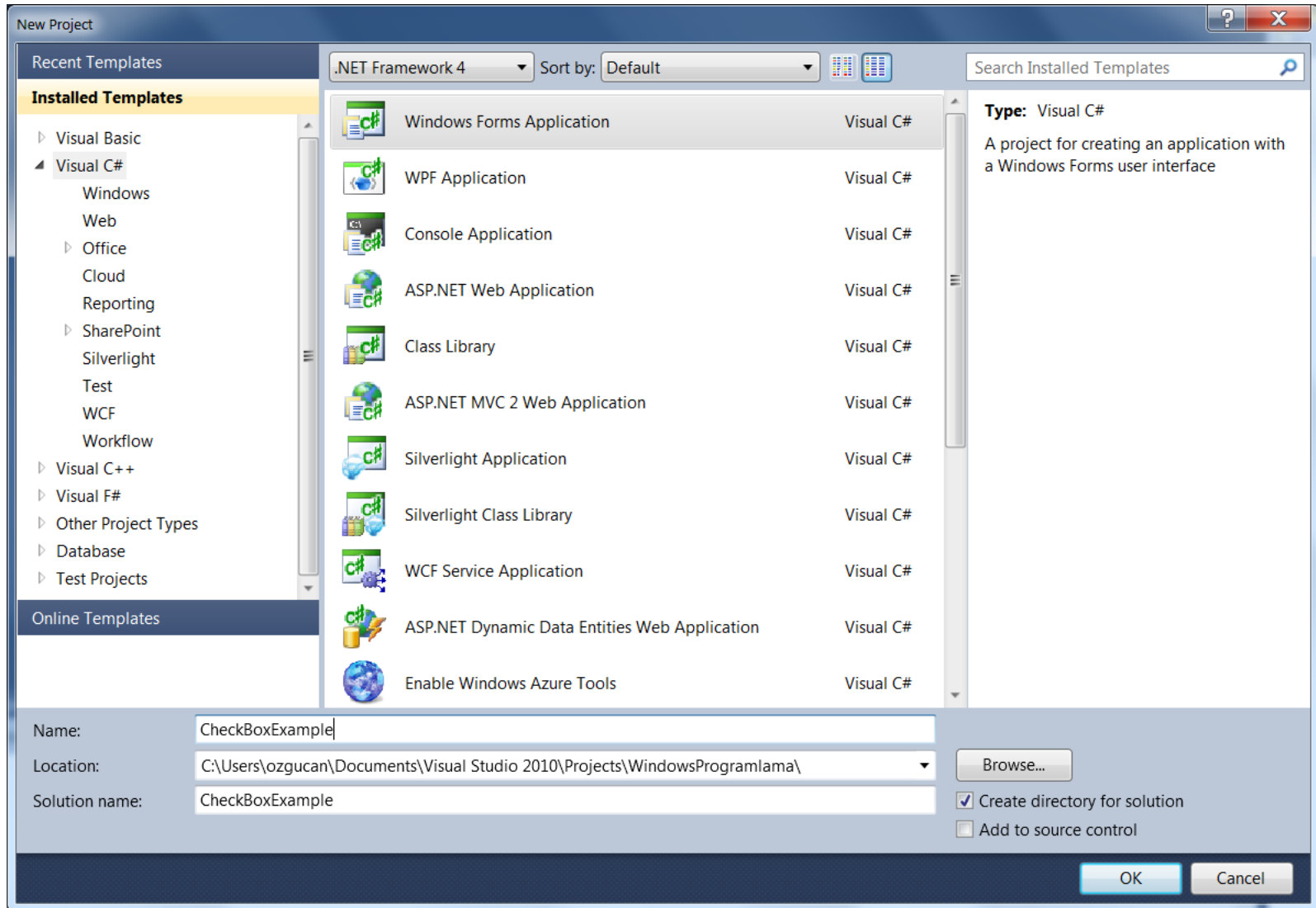
CheckBox

- Kullanıcı **CheckBox** | seçtiğinde:
 - onay imi belirir
- seçimi kaldırdığında:
 - onay imi kalkmaktadır
- İstenilen sayıda **CheckBox** seçilebilir.
- **ThreeState = True**
 - Checked
 - Unchecked
 - Indeterminate
- **ThreeState = False**

CheckBox

CheckBox properties and events	Description
<i>Common Properties</i>	
Appearance	By default, this property is set to <code>Normal</code> , and the <code>CheckBox</code> displays as a traditional checkbox. If it's set to <code>Button</code> , the <code>CheckBox</code> displays as a <code>Button</code> that looks pressed when the <code>CheckBox</code> is checked.
Checked	Indicates whether the <code>CheckBox</code> is checked (contains a check mark) or unchecked (blank). This property returns a <code>bool</code> value. The default is <code>false</code> (unchecked).
CheckState	Indicates whether the <code>CheckBox</code> is checked or unchecked with a value from the <code>CheckState</code> enumeration (<code>Checked</code> , <code>Unchecked</code> or <code>Indeterminate</code>). <code>Indeterminate</code> is used when it's unclear whether the state should be <code>Checked</code> or <code>Unchecked</code> . When <code>CheckState</code> is set to <code>Indeterminate</code> , the <code>CheckBox</code> is usually shaded.
Text	Specifies the text displayed to the right of the <code>CheckBox</code> .
ThreeState	When this property is <code>true</code> , the <code>CheckBox</code> has three states—checked, unchecked and indeterminate. By default, this property is <code>false</code> and the <code>CheckBox</code> has only two states—checked and unchecked.
<i>Common Events</i>	
CheckedChanged	Generated when the <code>Checked</code> property changes. This is a <code>CheckBox</code> 's default event. When a user double clicks the <code>CheckBox</code> control in design view, an empty event handler for this event is generated.
CheckStateChanged	Generated when the <code>CheckState</code> property changes.

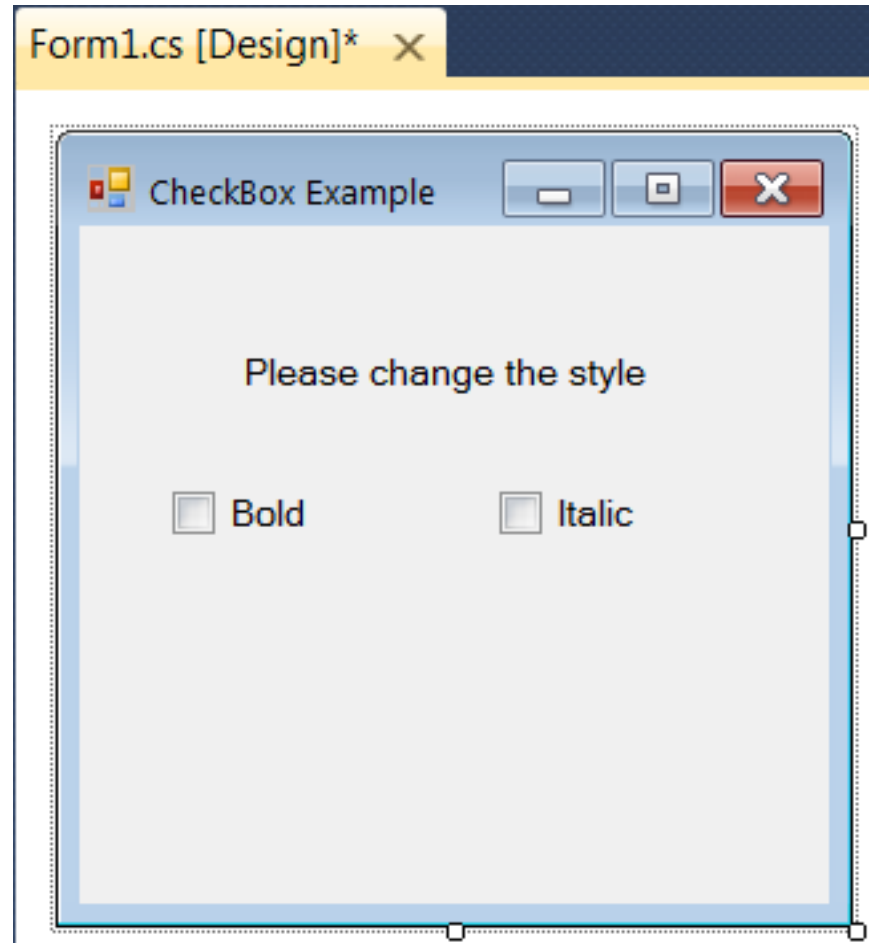
Örnek Uygulama (CheckBox)



Örnek Uygulama (CheckBox)

- **Form**
 - Text = **CheckBox Example**
- **Label**
 - Text = **Please change the style**
 - Name = **outputLabel**
- **2 Checkbox**
 - **CheckBox1**
 - Text = **Bold**
 - Name = **checkBoxBold**
 - **CheckBox2**
 - Text = **Italic**
 - Name = **checkBoxItalic**

Örnek Uygulama (CheckBox)



Örnek Uygulama (CheckBox)

```
private void checkBoxBold_CheckedChanged(object sender, EventArgs e)
{
    outputLabel.Font = new Font(outputLabel.Font, outputLabel.Font.Style ^ FontStyle.Bold);
}

private void checkBoxItalic_CheckedChanged(object sender, EventArgs e)
{
    outputLabel.Font = new Font(outputLabel.Font, outputLabel.Font.Style ^ FontStyle.Italic);
}
```

Font sınıfı:

- **System.Drawing** ad uzayında yer alır.
- Mevcut font'u ve yeni stili değişken olarak alır
 - **outputLabel.Font** → **outputLabel**'ın font adını ve boyutunu kullanır.

Font Stillerinde Bitwise İşlemler

- Farklı stilleri uygulamak ya da stilleri silmek için kullanılır.
 - OR |
 - XOR ^
- ÖR: **OR** işlemi
 - 01 = **Bold**
 - 10 = *Italic*
 - — —
 - 11 = ***Bold ve Italic***

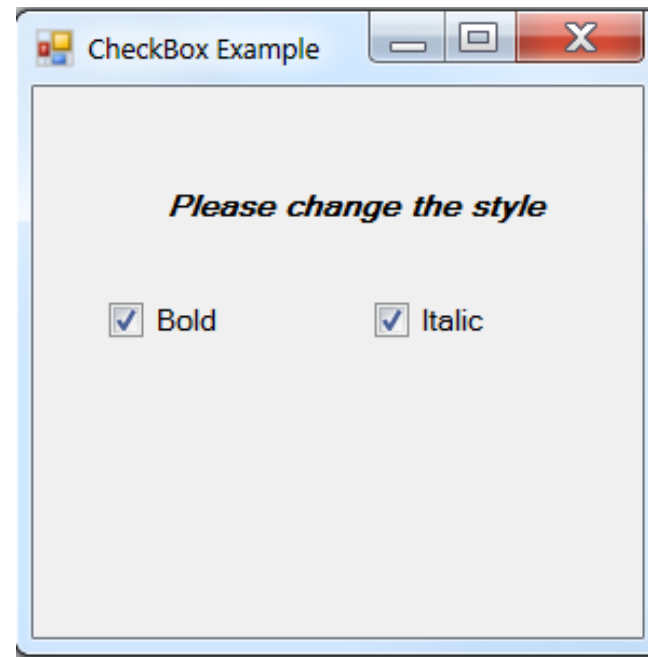
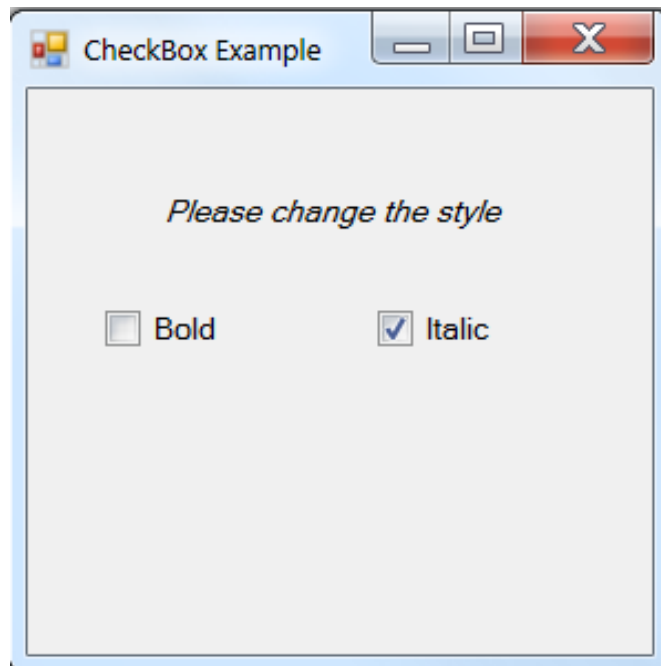
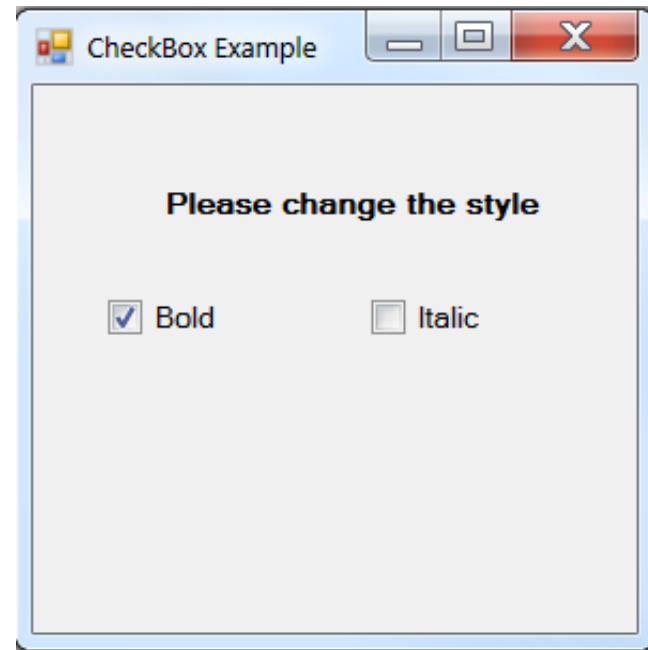
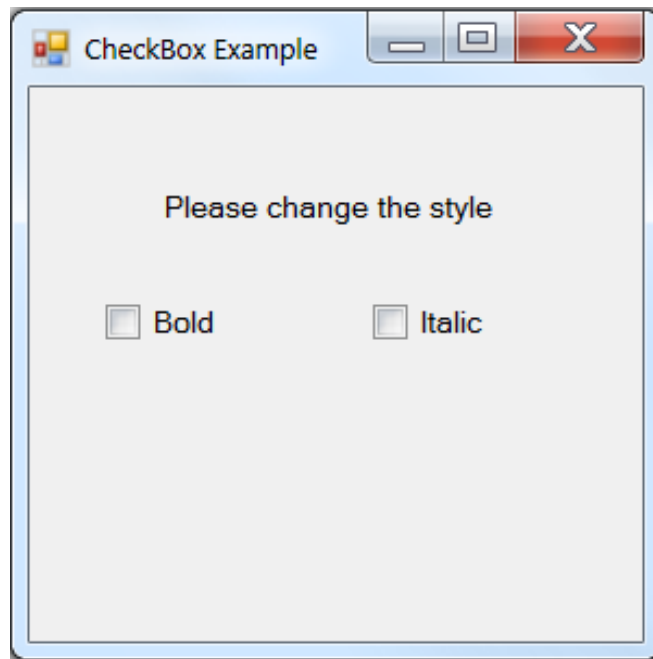
Font Stillerinde Bitwise İşlemler

- ÖR: Bold stili silme → **XOR** işlemi

11 = ***Bold ve Italic***

01 = **Bold**

10 = *Italic*



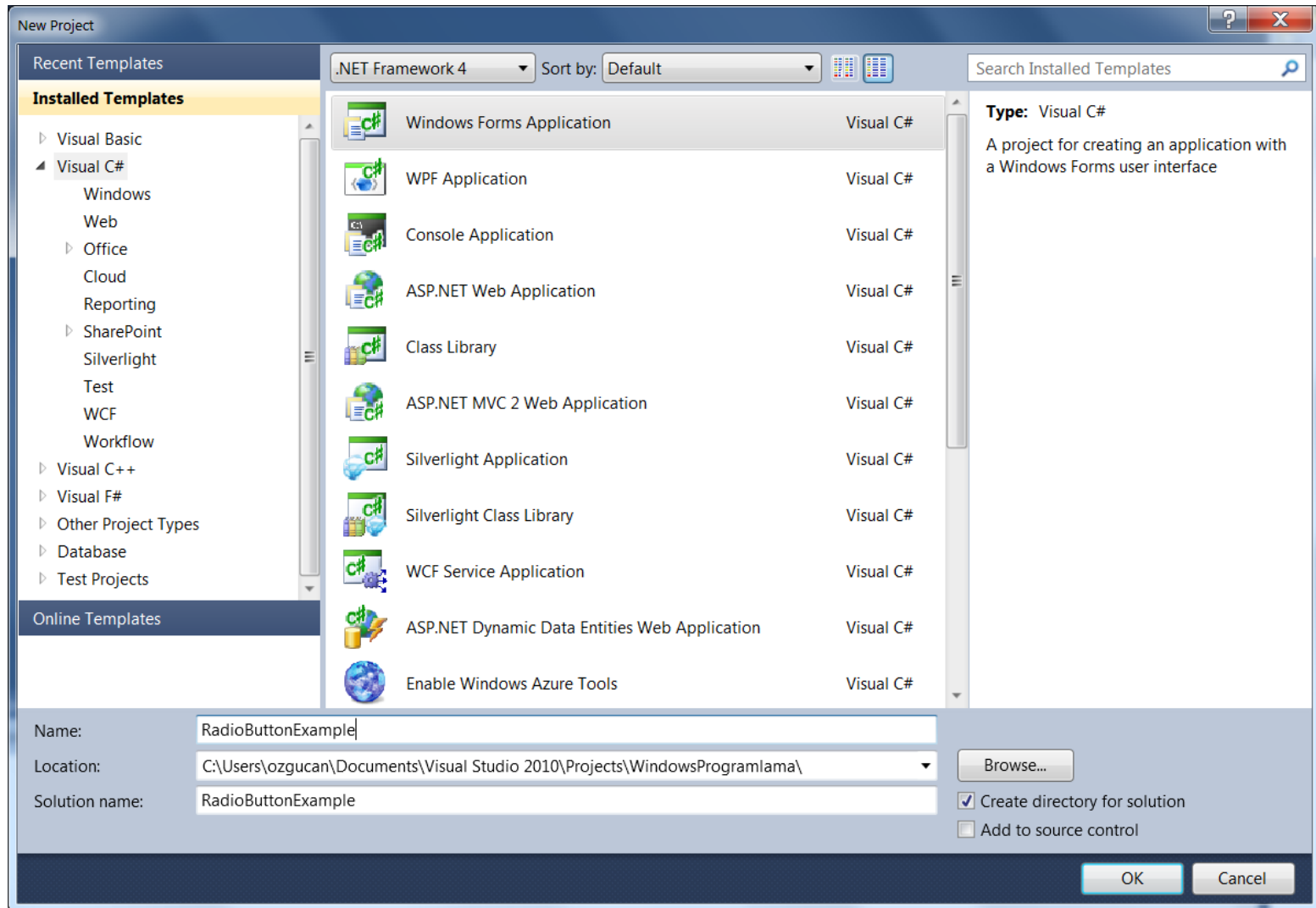
RadioButton

- İki durumu vardır:
 - Seçili (Selected)
 - Seçili değil (Not Selected/Deselected)
- Grup olarak görüntülenir → Aynı anda sadece bir RadioButton seçili diğerleri seçili değil durumda olur.
- Bir kapsayıcı (container) içine eklenen her bir **RadioButton** aynı grubun bir parçası olur.
 - Farklı gruplarda olması isteniyorsa, ayrı kapsayıcılara eklenmelidir.

RadioButton

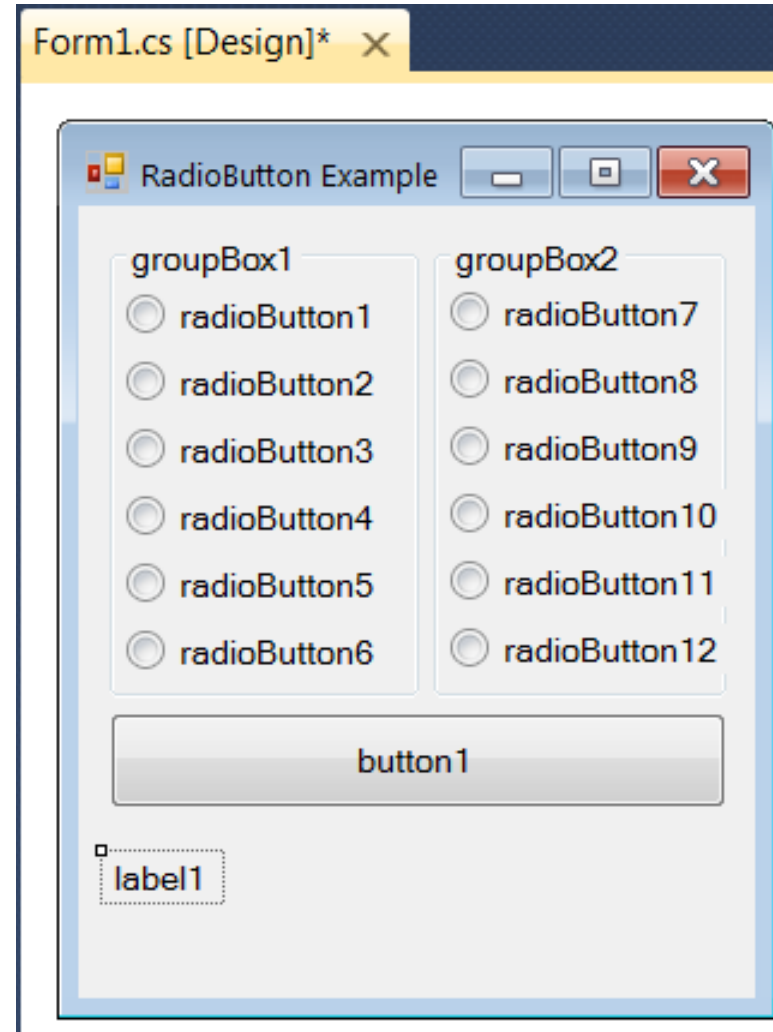
RadioButton properties and an event	Description
<i>Common Properties</i>	
Checked	Indicates whether the RadioButton is checked.
Text	Specifies the RadioButton's text.
<i>Common Event</i>	
CheckedChanged	Generated every time the RadioButton is checked or unchecked. When you double click a RadioButton control in design view, an empty event handler for this event is generated.

Örnek Uygulama (RadioButton)



Örnek Uygulama (RadioButton)

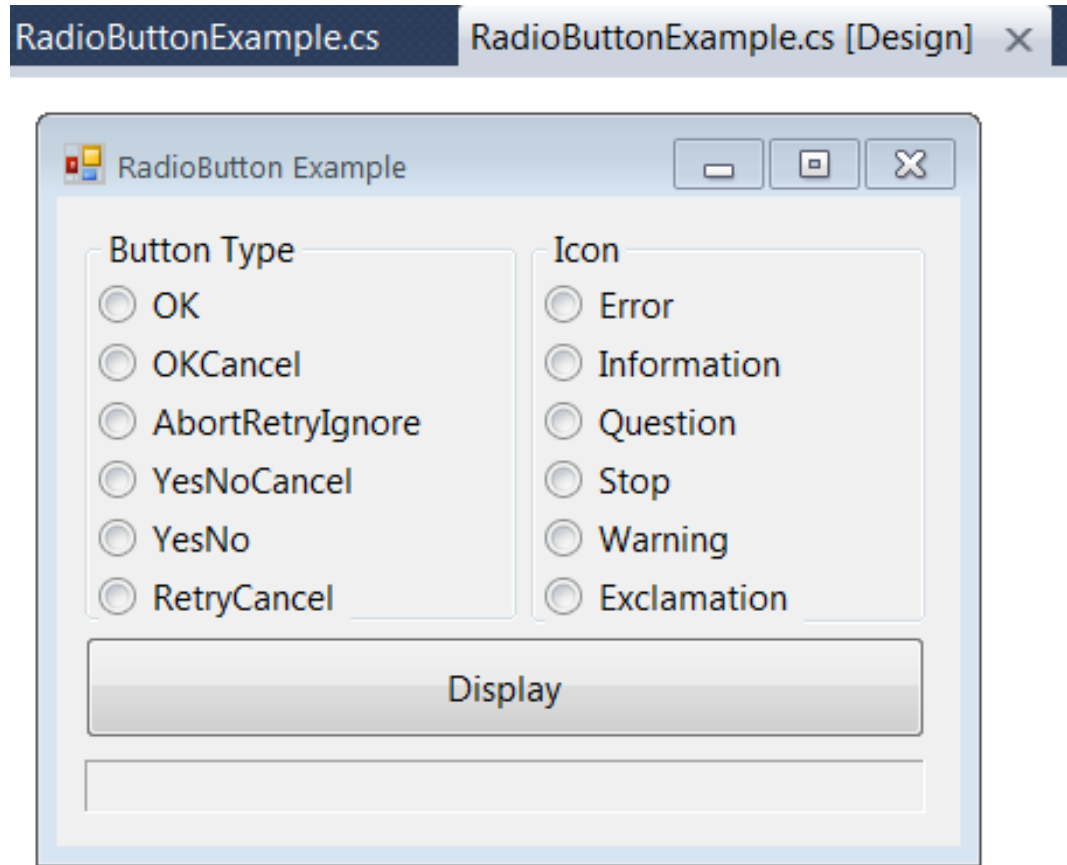
- 2 GroupBox
 - 2 * 6
RadioButton
- Button
- Label



Örnek Uygulama (RadioButton)

- **Form**
 - Text =
**RadioButton
Example**
- **groupBox1**
 - Text = **Button Type**
 - Name =
buttonTypeGroupBox
- **groupBox2**
 - Text = **Icon**
 - Name = **iconGroupBox**
- **Button**
 - Text = **Display**
 - Name = **displayButton**
- **Label**
 - Text = **Blank**
 - Name = **displayLabel**
- **RadioButton**
 - Text = **OK**
 - Name = **okRadioButton**

Örnek Uygulama (RadioButton)



Örnek Uygulama (RadioButton)

```
public partial class RadioButtonExample : Form
{
    // create variables that store the user's choice of options
    private MessageBoxIcon iconType;
    private MessageBoxButtons buttonType;

    public RadioButtonExample()
    {
        InitializeComponent();
    }
}
```

Örnek Uygulama (RadioButton)

```
private void buttonType_CheckedChanged(  
    object sender, EventArgs e )  
{  
    if ( sender == okRadioButton )  
        buttonType = MessageBoxButtons.OK;  
  
    else if ( sender == okCancelRadioButton )  
        buttonType = MessageBoxButtons.OKCancel;  
  
    else if ( sender == abortRetryIgnoreRadioButton )  
        buttonType = MessageBoxButtons.AbortRetryIgnore;  
  
    else if ( sender == yesNoCancelRadioButton )  
        buttonType = MessageBoxButtons.YesNoCancel;  
  
    else if ( sender == yesNoRadioButton )  
        buttonType = MessageBoxButtons.YesNo;  
  
    else  
        buttonType = MessageBoxButtons.RetryCancel;  
}
```

Örnek Uygulama (RadioButton)

```
private void iconType_CheckedChanged( object sender, EventArgs e )
{
    if ( sender == errorRadioButton )
        iconType = MessageBoxIcon.Error;

    else if ( sender == exclamationRadioButton )
        iconType = MessageBoxIcon.Exclamation;

    else if ( sender == informationRadioButton )
        iconType = MessageBoxIcon.Information;

    else if ( sender == questionRadioButton )
        iconType = MessageBoxIcon.Question;

    else if ( sender == stopRadioButton )
        iconType = MessageBoxIcon.Stop;

    else
        iconType = MessageBoxIcon.Warning;
}
```

Örnek Uygulama (RadioButton)

```
private void displayButton_Click( object sender, EventArgs e )
{
    DialogResult result = MessageBox.Show("This is your MessageBox.", "Custom Message", buttonType, iconType);

    switch ( result )
    {
        case DialogResult.OK:
            displayLabel.Text = "OK was pressed.";
            break;
        case DialogResult.Cancel:
            displayLabel.Text = "Cancel was pressed.";
            break;
        case DialogResult.Abort:
            displayLabel.Text = "Abort was pressed.";
            break;
        case DialogResult.Retry:
            displayLabel.Text = "Retry was pressed.";
            break;
        case DialogResult.Ignore:
            displayLabel.Text = "Ignore was pressed.";
            break;
        case DialogResult.Yes:
            displayLabel.Text = "Yes was pressed.";
            break;
        case DialogResult.No:
            displayLabel.Text = "No was pressed.";
            break;
    }
}
```


Örnek Uygulama (RadioButton)

RadioButton Example

Button Type	Icon
<input type="radio"/> OK	<input type="radio"/> Error
<input type="radio"/> OKCancel	<input checked="" type="radio"/> Information
<input checked="" type="radio"/> AbortRetryIgnore	<input type="radio"/> Question
<input type="radio"/> YesNoCancel	<input type="radio"/> Stop
<input type="radio"/> YesNo	<input type="radio"/> Warning
<input type="radio"/> RetryCancel	<input type="radio"/> Exclamation

Display

Custom Message

 This is your MessageBox.

Abort Retry Ignore

RadioButton Example

Button Type	Icon
<input type="radio"/> OK	<input type="radio"/> Error
<input type="radio"/> OKCancel	<input checked="" type="radio"/> Information
<input checked="" type="radio"/> AbortRetryIgnore	<input type="radio"/> Question
<input type="radio"/> YesNoCancel	<input type="radio"/> Stop
<input type="radio"/> YesNo	<input type="radio"/> Warning
<input type="radio"/> RetryCancel	<input type="radio"/> Exclamation

Display

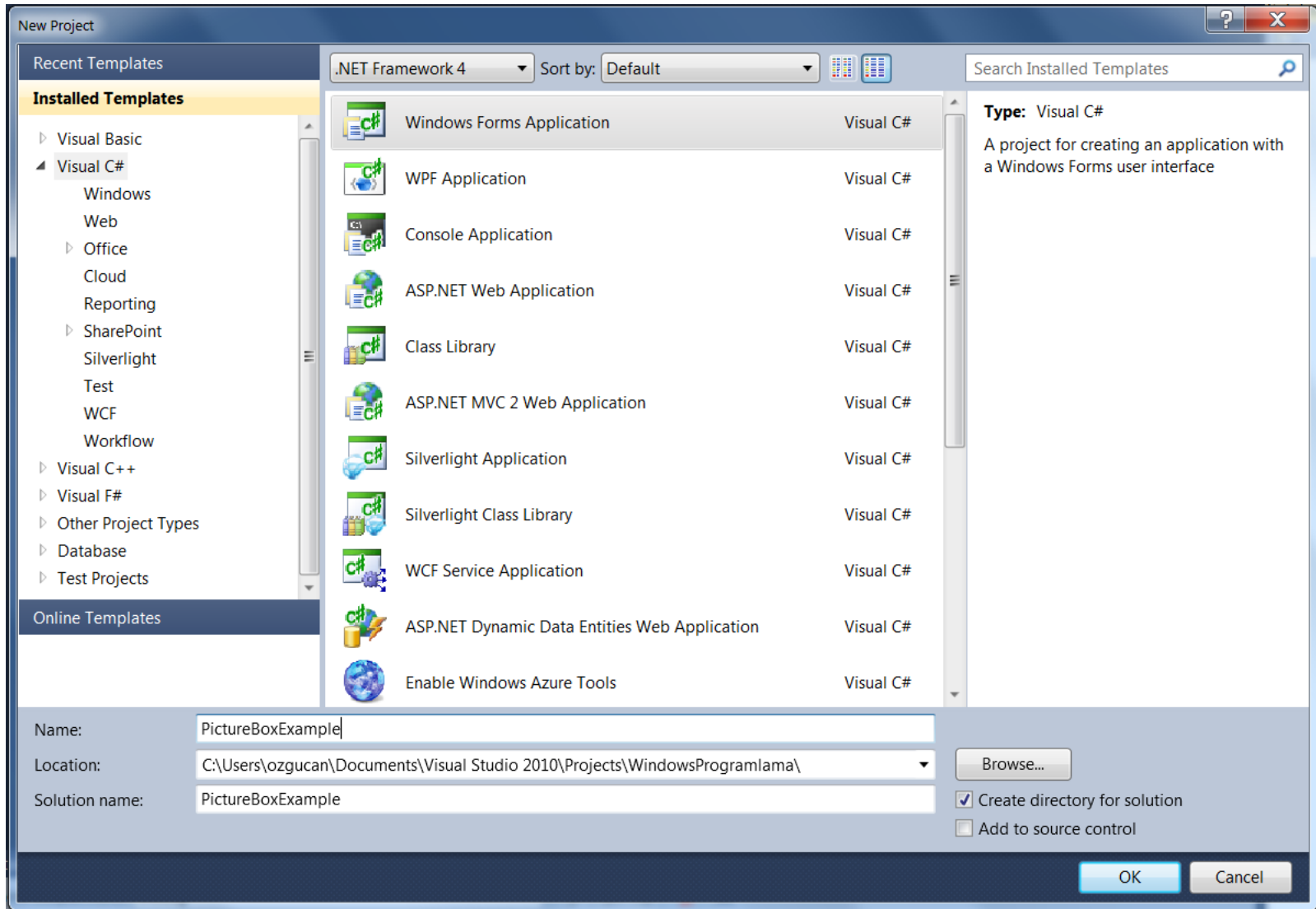
Abort was pressed.

PictureBox

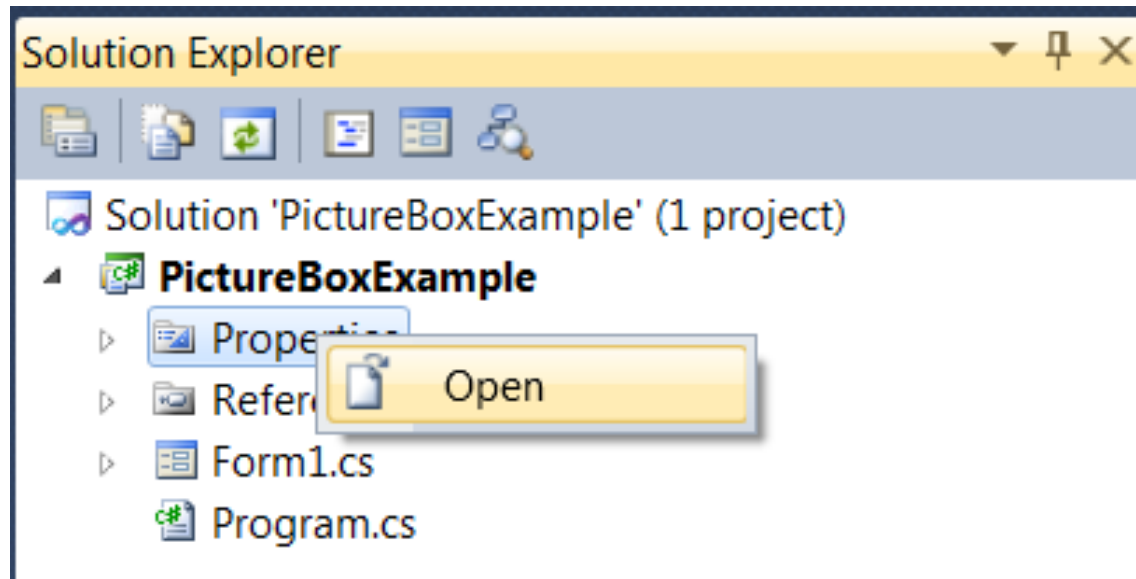
- Resim dosyasını görüntüler. (GIF, JPEG, etc.)

PictureBox properties and an event	Description
<i>Common Properties</i>	
Image	Sets the image to display in the PictureBox.
SizeMode	Enumeration that controls image sizing and positioning. Values are Normal (default), StretchImage, AutoSize, CenterImage, and Zoom. Normal places the image in the PictureBox's top-left corner, and CenterImage puts the image in the middle. These two options truncate the image if it's too large. StretchImage resizes the image to fit in the PictureBox. AutoSize resizes the PictureBox to hold the image. Zoom resizes the image to fit the PictureBox but maintains the original aspect ratio.
<i>Common Event</i>	
Click	Occurs when the user clicks a control. When you double click this control in the designer, an event handler is generated for this event.

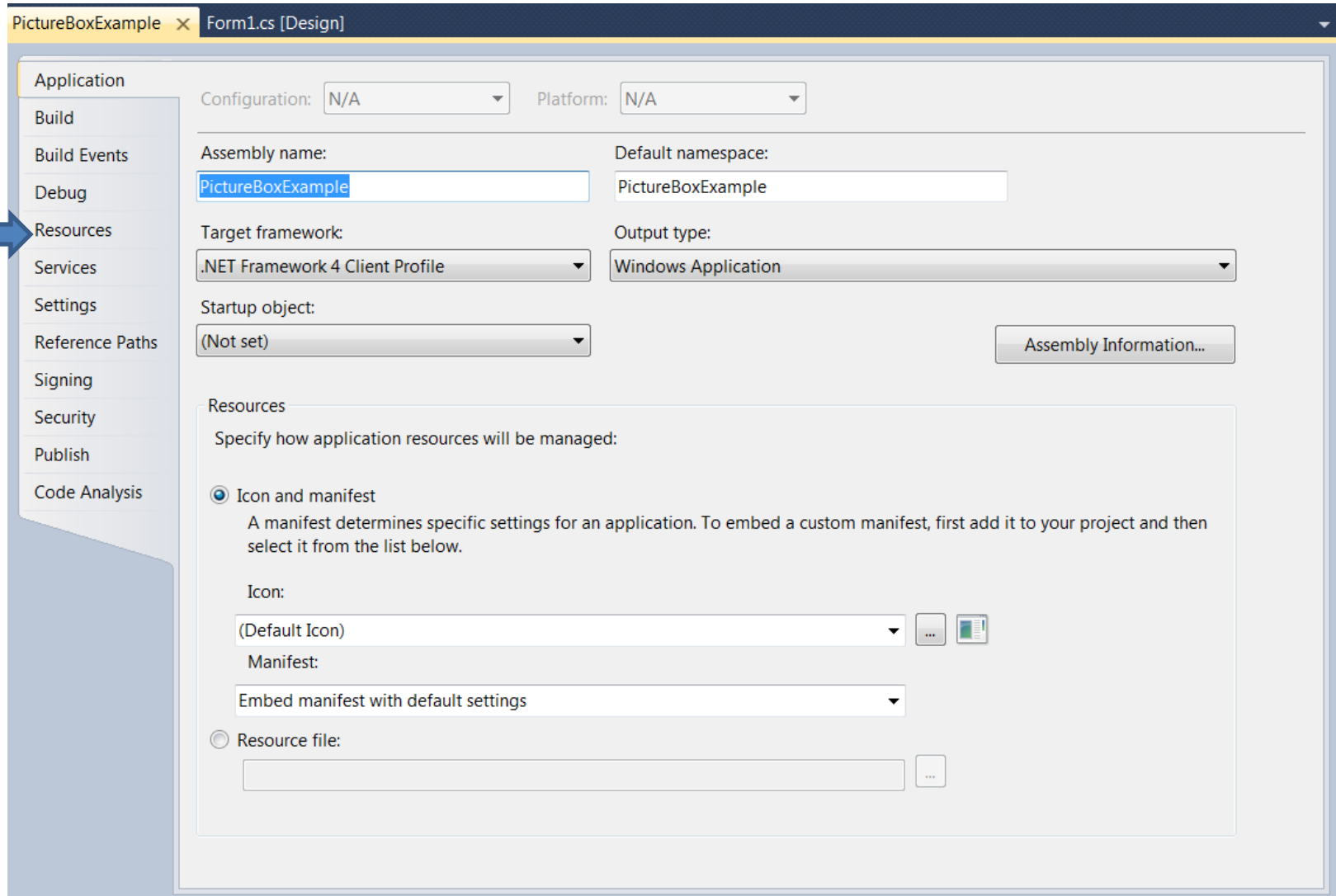
Örnek Uygulama (PictureBox)



Örnek Uygulama (PictureBox)



Örnek Uygulama (PictureBox)



PictureBoxExample x Form1.cs [Design]

Configuration: N/A Platform: N/A

Assembly name: PictureBoxExample Default namespace: PictureBoxExample

Target framework: .NET Framework 4 Client Profile Output type: Windows Application

Startup object: (Not set) Assembly Information...

Resources

Specify how application resources will be managed:

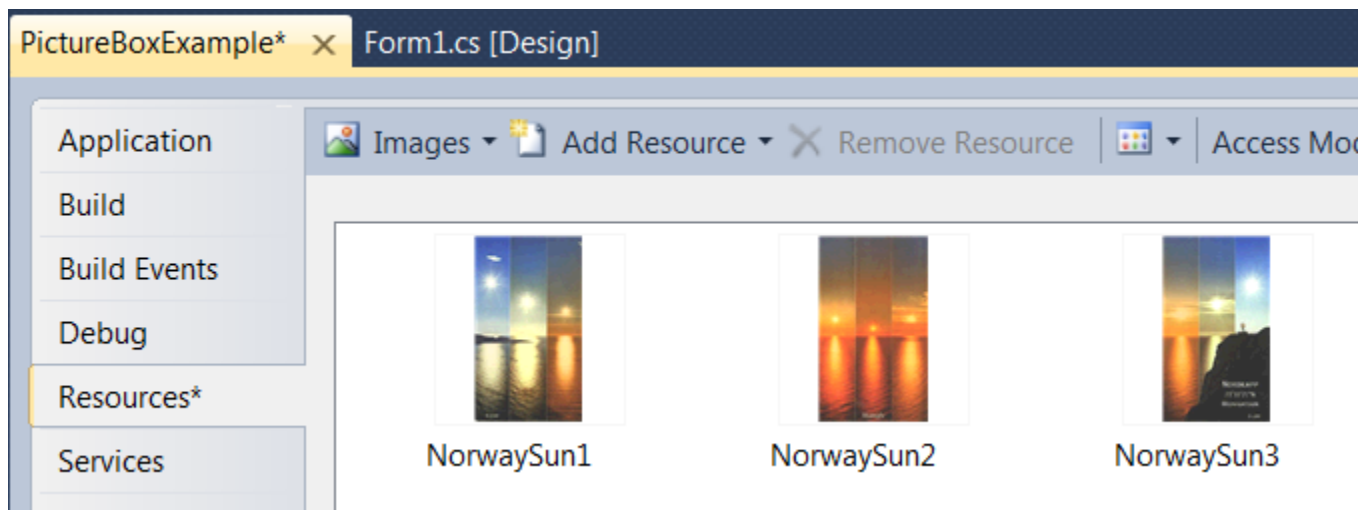
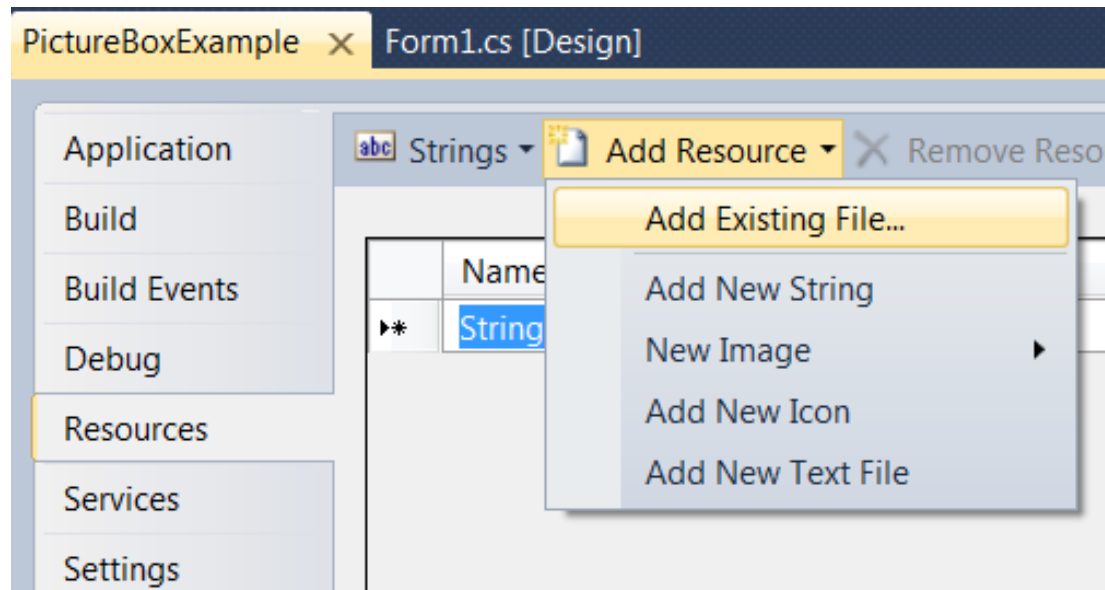
☒ Icon and manifest
A manifest determines specific settings for an application. To embed a custom manifest, first add it to your project and then select it from the list below.

Icon: (Default Icon) ...

Manifest: Embed manifest with default settings

☐ Resource file: ...

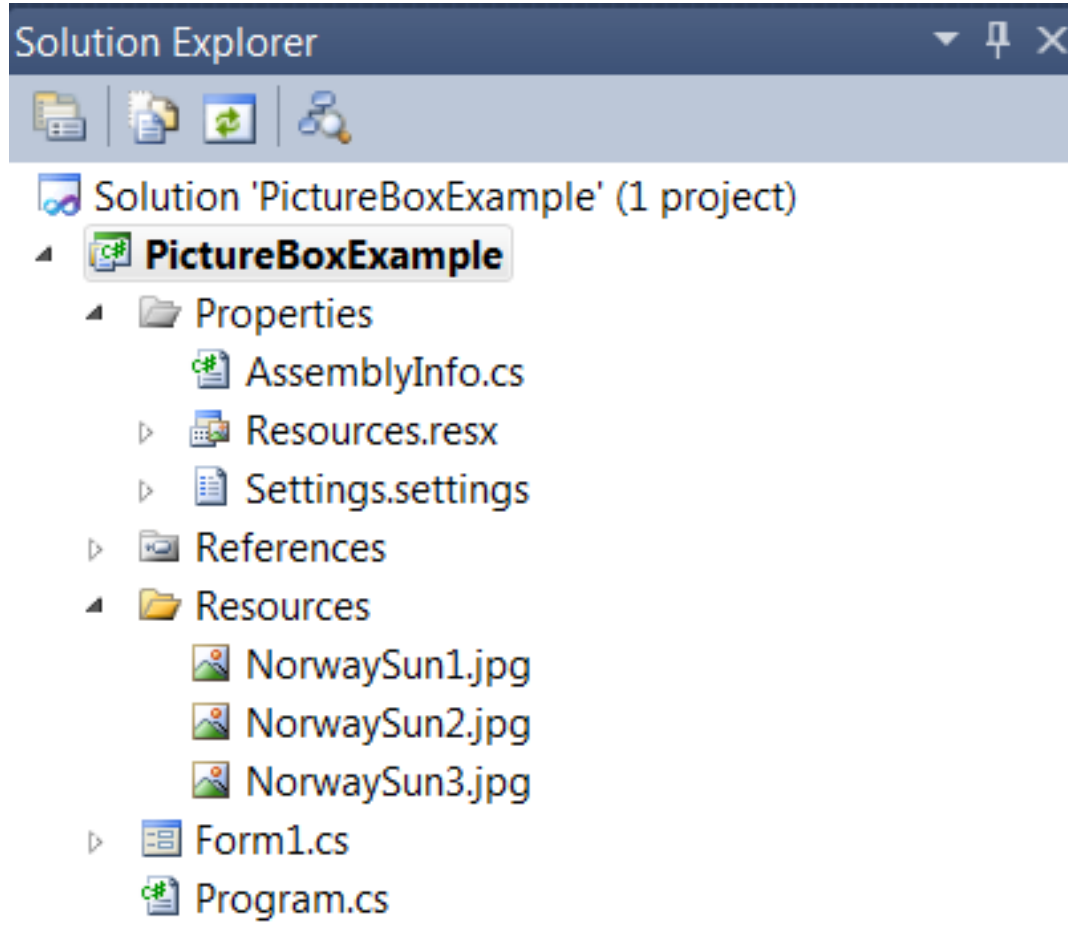
Örnek Uygulama (PictureBox)



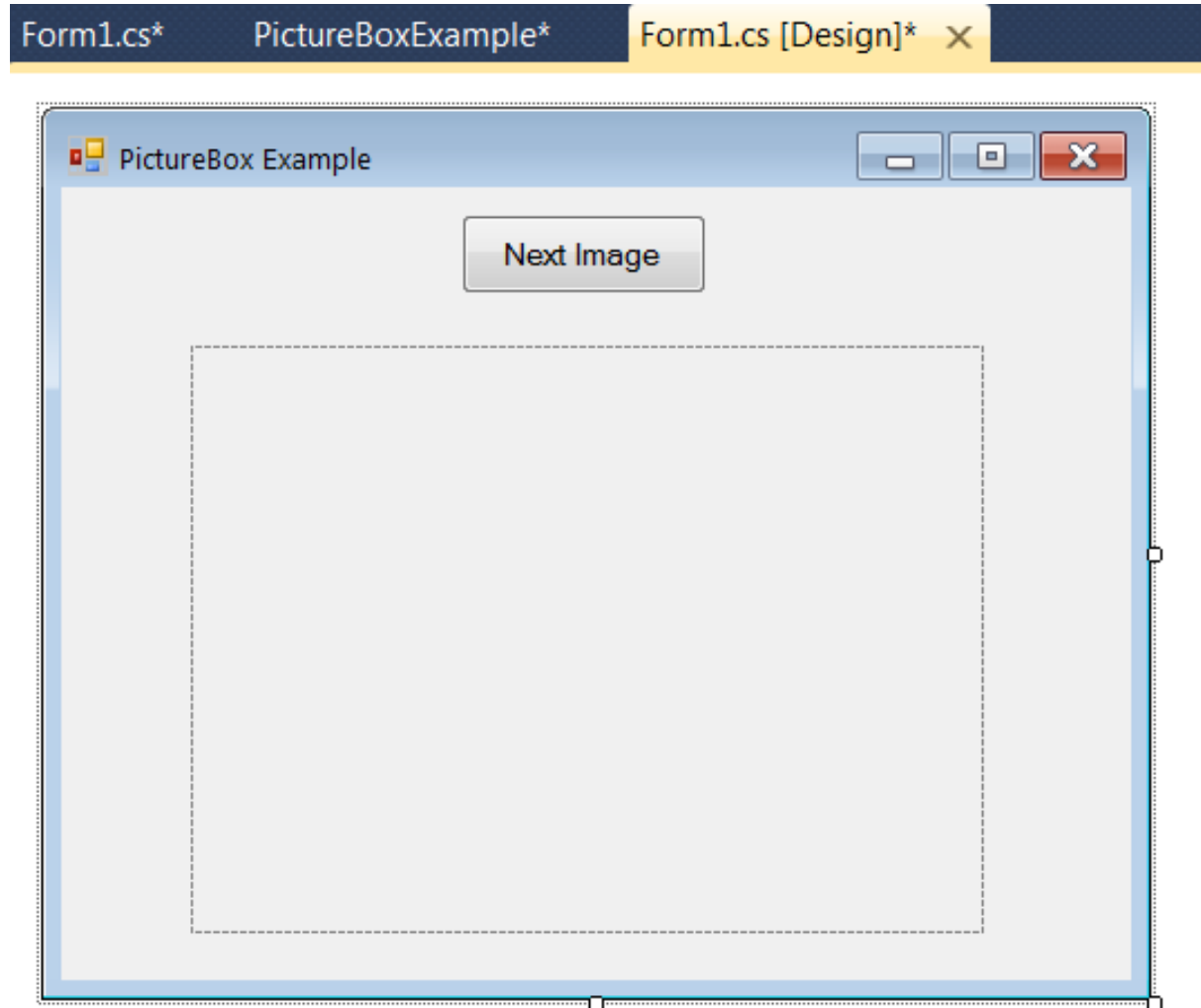
Örnek Uygulama (PictureBox)

- Button
 - Text = **Next Image**
 - Name = **nextButton**
- PictureBox
 - Name = **imagePictureBox**

Örnek Uygulama (PictureBox)



Örnek Uygulama (PictureBox)



Örnek Uygulama (PictureBox)

```
namespace PictureBoxExample
{
    public partial class Form1 : Form
    {
        private int imageNum = 0; // determines which image is displayed

        public Form1()
        {
            InitializeComponent();
        }

        private void nextButton_Click(object sender, EventArgs e)
        {
            imageNum = (imageNum + 1) % 4; // imageNum cycles from 1 to 3

            imagePictureBox.Image = (Image)(Properties.Resources.ResourceManager.GetObject(string.Format("NorwaySun{0}", imageNum)));
        }
    }
}
```

Örnek Uygulama (PictureBox)

