Visual Basic .NET

Buttons, Checkboxes, Radio Buttons, Panels e Group Boxes

Professor: Danilo Giacobo

Página pessoal: www.danilogiacobo.eti.br
E-mail: danilogiacobo@gmail.com

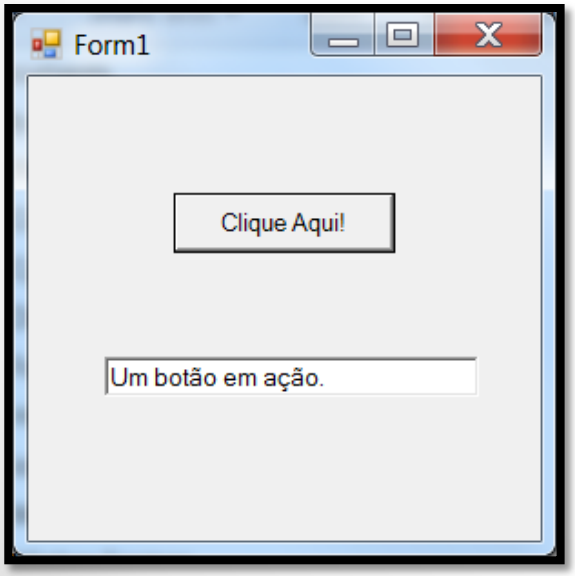
Objetivos da aula

- ✓ Trabalhar com o controle **Button**
- ✓ Trabalhar com o controle **CheckBox**
- ✓ Trabalhar com o controle **RadioButton**
- ✓ Trabalhar com o controle **Panel**
- ✓ Trabalhar com o controle **GroupBox**

Buttons

- O controle mais popular do Visual Basic é o Button (botão de comando).
- Ele provê o meio mais popular de criação e manipulação de eventos.
- Eles podem ser acessados por meio do clique do mouse ou pela tecla ENTER (se o botão contiver o foco).
- Eles também são bastante usados em Caixas de Diálogo.

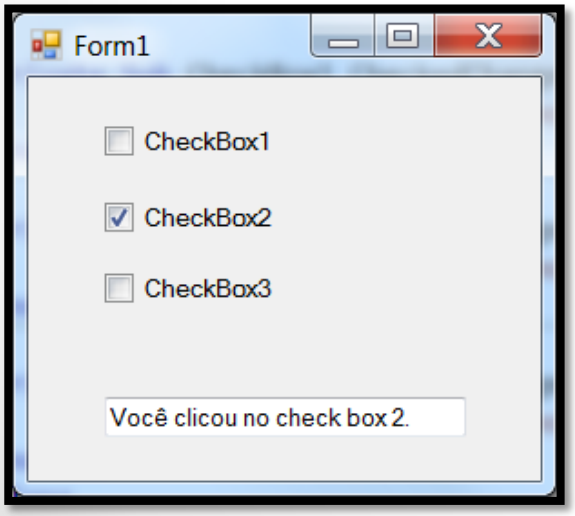
Exemplo de form usando um controle Button e TextBox:



CheckBoxes

- São controles para marcar/desmarcar uma opção.
- Ele pode mostrar um texto, uma imagem ou ambos.
- Você pode modificar a aparência de um **CheckBox** usando as propriedades **Appearance** e **FlatStyle**.
- Você usa a propriedade **Checked** para controlar as seleções realizadas pelo usuário.

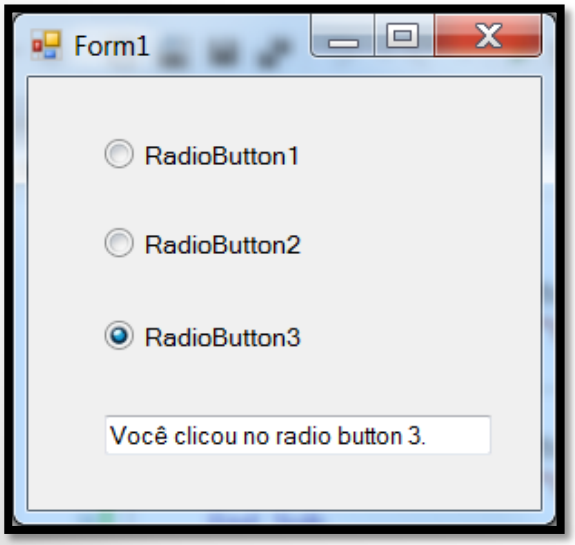
Exemplo:



RadioButtons

- São chamados também de Option Buttons e são similares aos CheckBoxes.
- Eles são redondos e usualmente criados de forma agrupada.
- Eles funcionam de forma dependente, isto é, quando uma das opções é marcada as outras são automaticamente desmarcadas.
- As mesmas propriedades de aparência e controle de seleção do CheckBox são aplicadas a este componente.

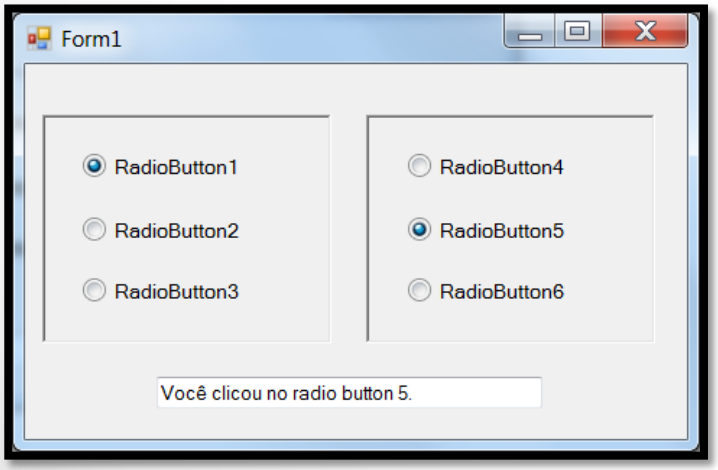
Exemplo:



Panels

- São usados para agrupar outros controles.
- Eles dividem o form em áreas distintas.
- Fica fácil de visualizar quais controles estão relacionados entre si.
- Se você mover um painel os controles dentro dele são movidos junto.
- Use a propriedade **BorderStyle** para mudar a aparência da borda dele.
- Ele pode conter barras de rolagem (propriedade **AutoScroll**).

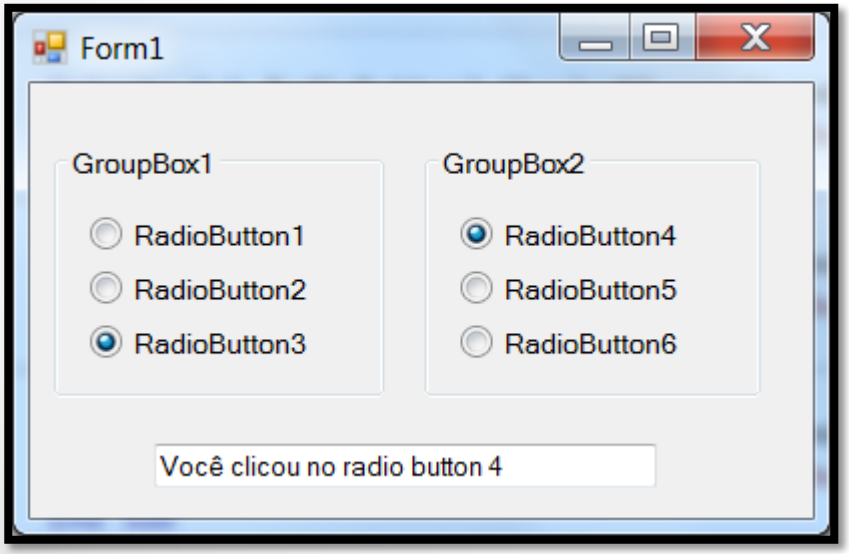
Exemplo:



Group Boxes

- Similar a um controle Panel.
- Ele mostra uma legenda no topo do componente (propriedade **Text**).
- Não possui barras de rolagens.
- A forma mais usada dele é para agrupar controles do tipo **Radio Buttons**.

Exemplo:



Tudo sobre Botões

- Todo mundo que já usou o Windows conhece sobre botões.
- O único evento digno de nota para este controle é o **Click**.
- Você pode configurar a aparência do botão alterando o alinhamento do texto, adicionando uma imagem e ajustando o estilo da borda.
- A hierarquia de classes do controle **Button** é a seguinte:

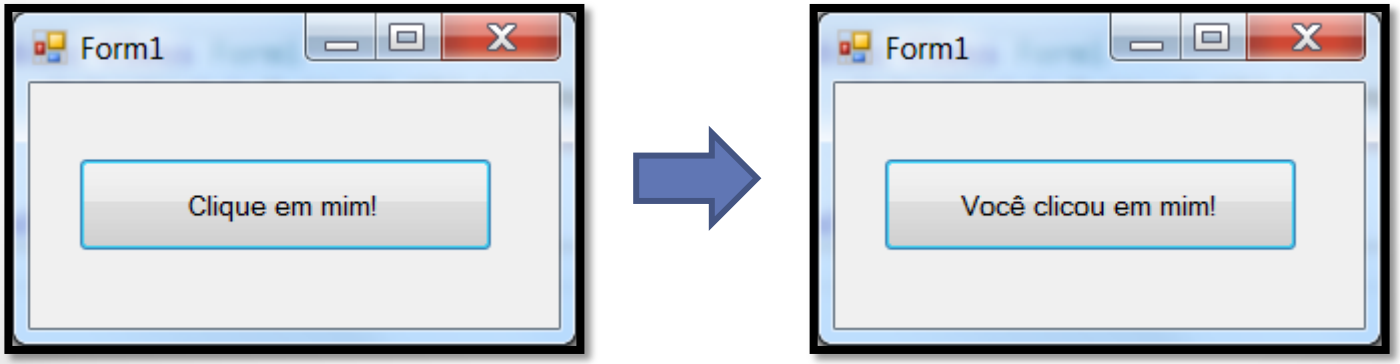
Object
    MarshalByRefObject
        Component
            Control
                ButtonBase
                    Button

Configurando o texto do botão

- Para alterar o texto exibido pelo botão (caption) é necessário usar a propriedade **Text** e então fornecer um texto para o mesmo.

Exemplo:

```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          Button1.Text = "Você clicou em mim!"
4      End Sub
5  End Class
```

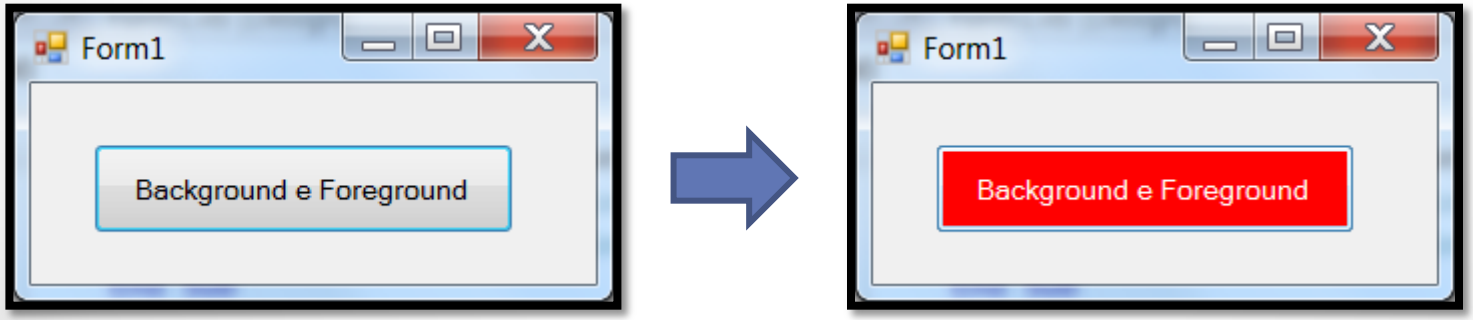


Configurando as cores do botão

- Use a propriedade **BackColor** para alterar a cor de fundo de um botão.
- Use a propriedade **ForeColor** para alterar a cor do primeiro plano.
- As cores disponíveis estão na enumeração **Color**.

Exemplo:

```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          Button1.BackColor = Color.Red
4          Button1.ForeColor = Color.White
5      End Sub
6  End Class
```

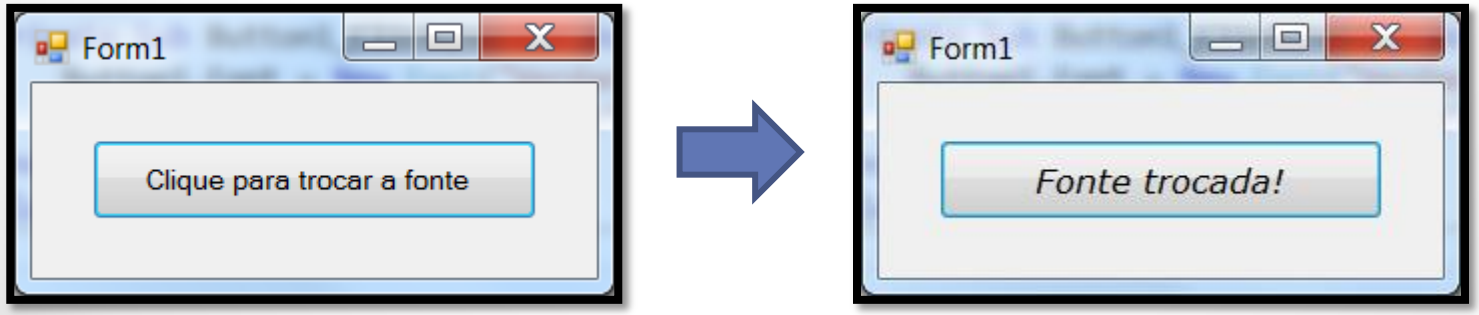


Configurando a fonte de um botão

- Para alterar o tipo, tamanho e estilo do texto de um botão é preciso usar a propriedade **Font**.
- Ela pode ser configurada tanto em modo design quanto em tempo de execução.

Exemplo:

```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          Button1.Font = New Font("Verdana", 10, FontStyle.Italic)
4          Button1.Text = "Fonte trocada!"
5      End Sub
6  End Class
```

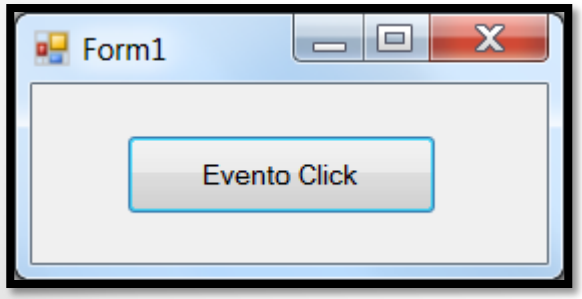


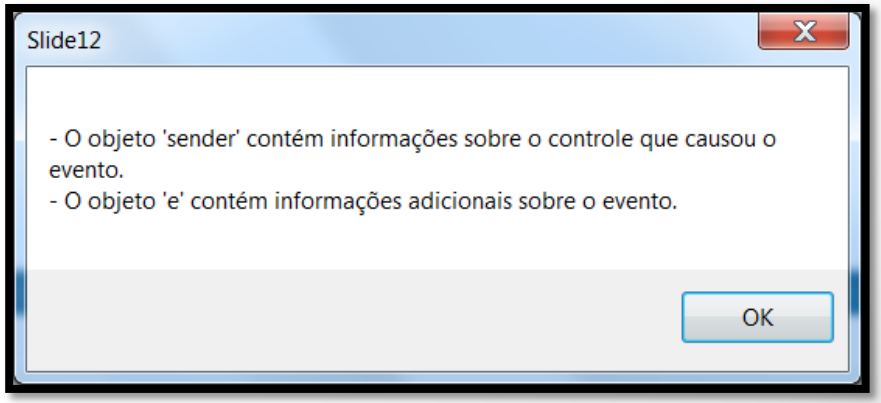
Manipulando eventos com um botão

- Para adicionar um evento ao botão (o padrão é o **Click**) basta clicar duas vezes sobre o mesmo no modo design. Será adicionado ao código um procedimento do tipo **Sub** para armazenar o código para tratar este evento.

Exemplo:

```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          Dim strMensagem As String
4          strMensagem = "- O objeto 'sender' contém informações sobre o controle que causou o evento."
5          strMensagem = strMensagem & vbCrLf & "- O objeto 'e' contém informações adicionais sobre o evento."
6          MsgBox(strMensagem)
7      End Sub
8  End Class
```





Alterando o foco depois do clique do botão

- Quando você clica em um botão o foco é transferido para o mesmo. Em alguns casos você pode querer manter o foco em outro campo.
- O controle **Button** possui dois eventos deste tipo: **GotFocus** e **LostFocus**.

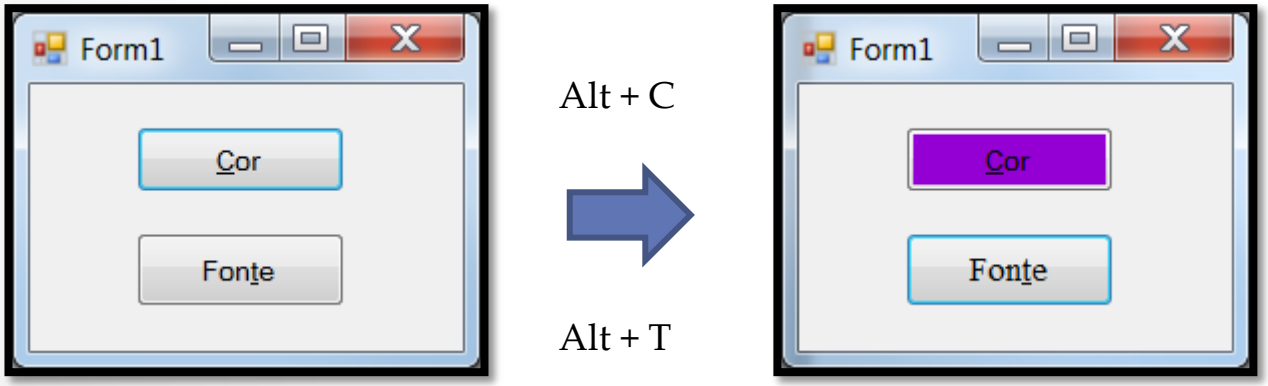
Exemplo:

```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          RichTextBox1.Find("Visual Basic")
4          RichTextBox1.Focus()
5      End Sub
6
7      Private Sub Button1_GotFocus(sender As Object, e As EventArgs) Handles Button1.GotFocus
8          RichTextBox1.Text = "O foco está no botão."
9      End Sub
10
11     Private Sub Button1_LostFocus(sender As Object, e As EventArgs) Handles Button1.LostFocus
12         RichTextBox1.Text = "O botão perdeu o foco."
13     End Sub
14 End Class
```

Inserindo os Caracteres de Acesso

Para colocar caracteres de acesso em controles do tipo **Button** e demais elementos gráficos basta colocar o caractere **"&"** na frente do caractere do texto do botão. Só tenha o cuidado de não usar o mesmo caractere de acesso para vários controles disponíveis em um form.

Veja um exemplo:




```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          Button1.BackColor = Color.DarkViolet
4      End Sub
5
6      Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
7          Button2.Font = New Font(System.Drawing.FontFamily.GenericSerif, 10)
8      End Sub
9  End Class
```

Configurando a ordem de tabulação

Para tornar seus controles do tipo Button mais acessíveis por meio do teclado (especialmente se você possuir vários deles) você pode usar as propriedades **TabStop** e **TabIndex**.

- **TabStop:** indica se o botão pode aceitar o foco quando a pessoa usa a tecla TAB.
- **TabIndex:** é o índice do botão corrente na ordem de tabulação (começa em 0).

Uma outra dica importante de controle de tabulação é quando você possui muitos controles do tipo text box em seu form. Você pode mudar de campo apenas usando a tecla TAB e isto pode ser altamente produtivo.

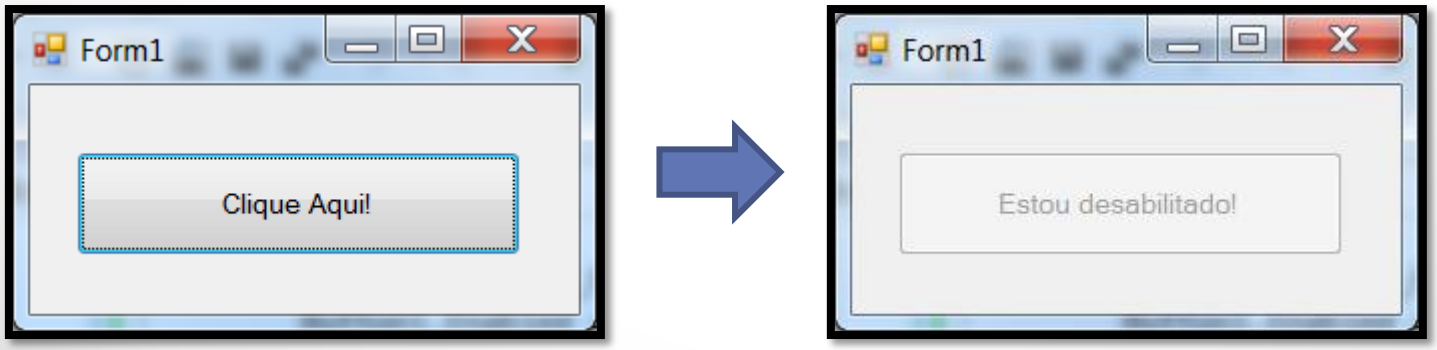
```
1  Public Class Form1
2      Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
3          Button1.TabIndex = 0
4          Button2.TabIndex = 1
5          Button3.TabIndex = 2
6          Button2.TabStop = False
7      End Sub
8  End Class
```

Desabilitando Botões

Você pode desabilitar um botão configurando a propriedade **Enabled** para **False**. Ela é bastante útil quando você não quer que o usuário fique clicando várias vezes no mesmo botão após ter clicado uma vez e ter de esperar o processo terminar. Você pode fazer isso em modo design ou em tempo de execução.

Exemplo:

```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          Button1.Enabled = False
4          Button1.Text = "Estou desabilitado!"
5      End Sub
6  End Class
```



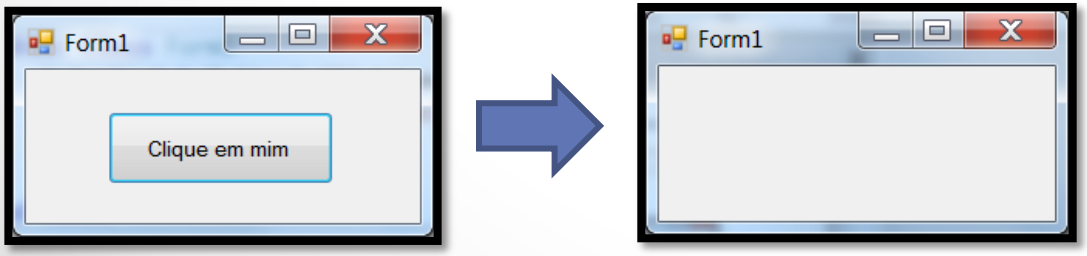
Mostrando e escondendo Botões

- Para fazer um botão desaparecer você configura a propriedade **Visible** para **False**.
- Para fazer um botão reaparecer você configura a propriedade **Visible** para **True**.

Dica: Você pode usar também os métodos **Show** e **Hide** da classe **Control**.

Exemplo:

```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          MsgBox("Adeus mundo cruel")
4          Button1.Visible = False
5      End Sub
6  End Class
```



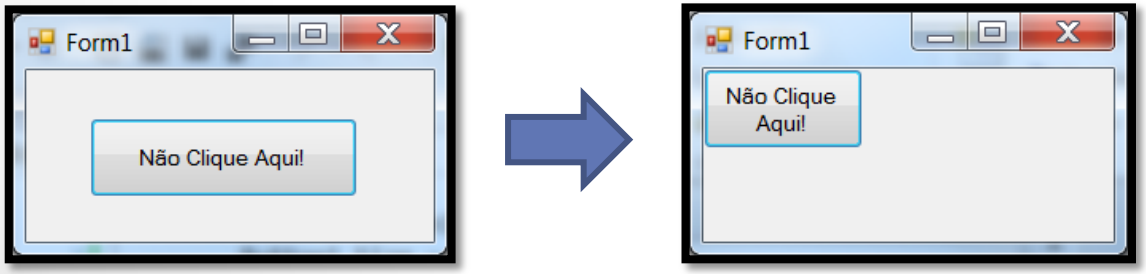
Redimensionando e Movendo Botões

Para redimensionar e mover botões em tempo de execução é necessário usar as propriedades **Size** e **Location**. Os valores atribuídos para essas propriedades estão em pixel. A sintaxe dessas propriedades é:

Size(dimensão_X, dimensão_Y) e **Point**(posição_X, posição_Y)

Exemplo:

```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          Button1.Size = New Size(100, 50)
4          Button1.Location = New Point(0, 0)
5      End Sub
6  End Class
```



Dica: Tente usar o método SetBounds para fazer o mesmo processo.

Adicionando uma Imagem ao Botão

Você pode adicionar imagens em botões tanto em modo design quanto em tempo de execução. Para isso basta usar a propriedade **Image** e o método **FromFile**.

Exemplo:

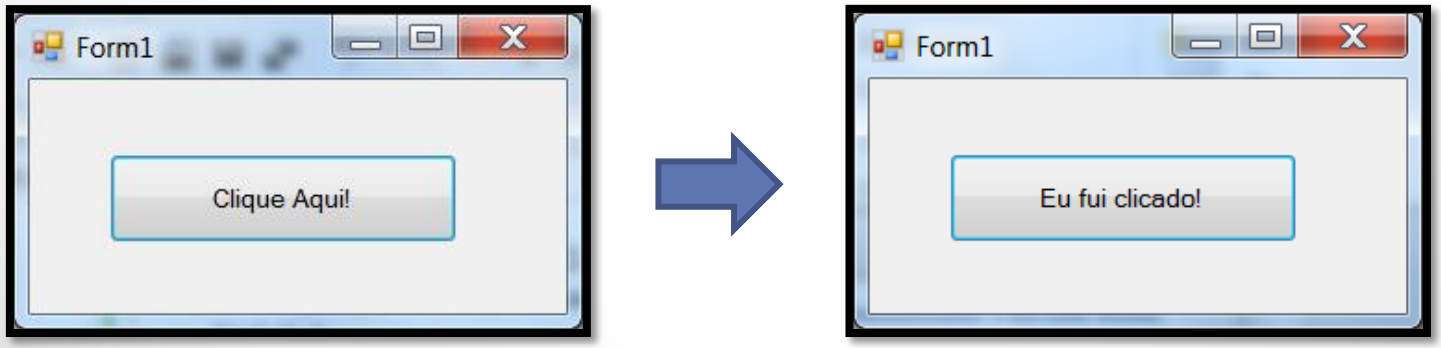
```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          Button1.Image = Image.FromFile("C:\300px-Button_Icon_Green.svg.png")
4          Button1.ImageAlign = ContentAlignment.MiddleCenter
5          Button1.Text = ""
6          Button1.FlatStyle = FlatStyle.Flat
7          TextBox1.Text = "Você clicou no botão"
8      End Sub
9  End Class
```

Passando um botão para um procedimento

- Você pode passar o objeto Button via parâmetro para um procedimento.
- Você deve passar ele por *referência*.

Exemplo:

```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          AjustarTexto(Button1)
4      End Sub
5
6      Private Sub AjustarTexto(ByRef btnBotao As Button)
7          btnBotao.Text = "Eu fui clicado!"
8      End Sub
9  End Class
```



Evento de liberação do botão

Para saber quando um botão foi clicado você usa o evento **Click** mas para poder capturar o evento de liberação do clique do mouse você usa o evento **MouseUp**.

Exemplo:

```
1  Public Class Form1
2      Private Sub Button1_MouseUp(sender As Object, e As MouseEventArgs) Handles Button1.MouseUp
3          TextBox1.Text = "O evento MouseUp ocorreu."
4      End Sub
5  End Class
```

Dica: Utilizar os eventos **MouseDown** e **MouseUp** juntos pode ser útil para ações que possuam duas partes.

Usando a classe Checkbox

- Controle usado para armazenar uma opção ou uma lista de opções marcadas pela pessoa.
- Ele possui dois estados: **marcado** e **desmarcado**.
- A hierarquia de classes do controle **Checkbox** é a seguinte:

Object
MarshalByRefObject
Component
Control
ButtonBase
CheckBox

Criando Checkboxes

Quando o valor da propriedade **Checked** muda, o evento **CheckChanged** ocorre. Neste evento você pode adicionar o código para realizar algum tipo de tratamento.

Exemplo:

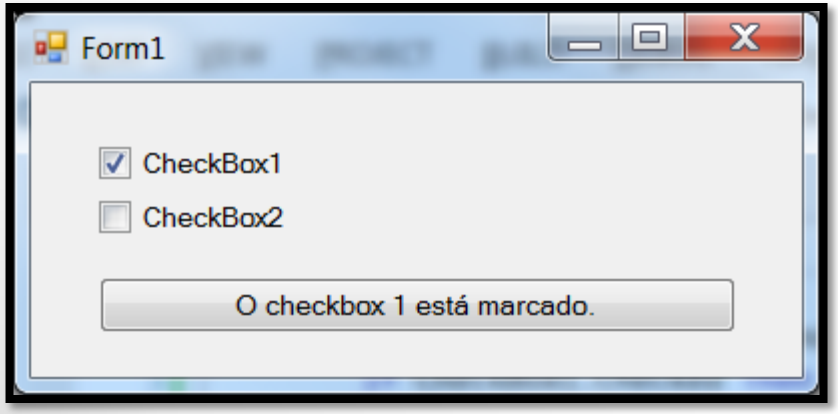
```
1  Public Class Form1
2      Private Sub CheckBox1_CheckedChanged(sender As Object, e As EventArgs) _
3          Handles CheckBox1.CheckedChanged
4          TextBox1.Text = "Você clicou no checkbox 1"
5      End Sub
6
7      Private Sub CheckBox2_CheckedChanged(sender As Object, e As EventArgs) _
8          Handles CheckBox2.CheckedChanged
9          TextBox1.Text = "Você clicou no checkbox 2"
10     End Sub
11
12     Private Sub CheckBox3_CheckedChanged(sender As Object, e As EventArgs) _
13         Handles CheckBox3.CheckedChanged
14         TextBox1.Text = "Você clicou no checkbox 3"
15     End Sub
16 End Class
```

Obtendo o estado do Checkbox

Para saber se um determinado checkbox foi marcado ou desmarcado basta acessar a propriedade **Checked**. Se for **True** ele está marcado; caso contrário (**False**) está desmarcado.

Exemplo:

```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          If CheckBox1.Checked Then
4              Button1.Text = "O checkbox 1 está marcado."
5          End If
6      End Sub
7  End Class
```



Configurando o estado de um Checkbox

Você pode configurar o estado (marcado ou desmarcado) de um checkbox usando a propriedade **Checked**.

- Checked = True: Marcado
- Checked = False: Desmarcado

Exemplo:

```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          CheckBox1.Checked = True
4      End Sub
5  End Class
```

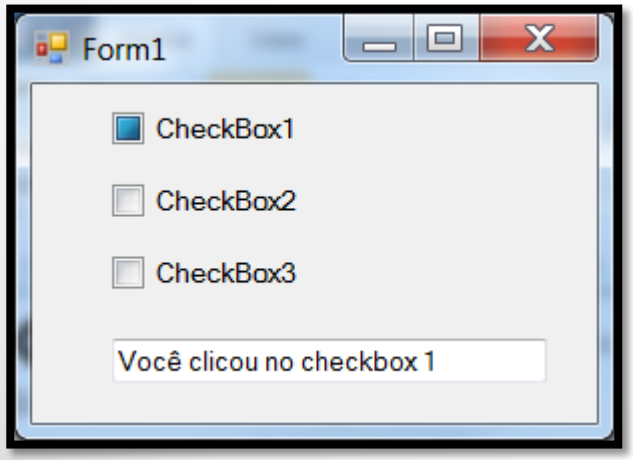
Dica: Para um checkbox aparecer marcado quando o programa é iniciado é preciso marcar a opção **Checked** no modo de design.

Criando Checkboxes com 3 estados

- Se você definir a propriedade **ThreeState** para **True** o seu controle checkbox ganhará um estado a mais, chamado de indeterminado.
- Para configurar ou obter o estado do checkbox nesse caso é necessário então usar a propriedade **CheckState**.

Exemplo:

```
1  Public Class Form1
2      Private Sub CheckBox1_CheckedChanged(sender As Object, e As EventArgs) Handles CheckBox1.CheckedChanged
3          TextBox1.Text = "Você clicou no checkbox 1"
4          CheckBox1.CheckState = CheckState.Indeterminate
5      End Sub
6  End Class
```



Usando a classe RadioButton

- Controle usado para armazenar uma única opção a partir de uma lista de opções disponíveis.
- Ele possui dois estados: **marcado** e **desmarcado**.
- Quando um membro do grupo é marcado os outros são automaticamente desmarcados.
- A hierarquia de classes do controle **RadioButton** é a seguinte:

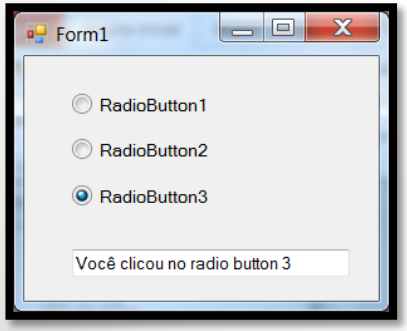
Object
MarshalByRefObject
Component
Control
ButtonBase
RadioButton

Criando Radio Buttons

Quando o valor da propriedade **Checked** muda, o evento **CheckChanged** ocorre. Neste evento você pode adicionar o código para realizar algum tipo de tratamento.

Exemplo:

```
1  Public Class Form1
2      Private Sub RadioButton1_CheckedChanged(sender As Object, e As EventArgs) Handles RadioButton1.CheckedChanged
3          TextBox1.Text = "Você clicou no radio button 1"
4      End Sub
5
6      Private Sub RadioButton2_CheckedChanged(sender As Object, e As EventArgs) Handles RadioButton2.CheckedChanged
7          TextBox1.Text = "Você clicou no radio button 2"
8      End Sub
9
10     Private Sub RadioButton3_CheckedChanged(sender As Object, e As EventArgs) Handles RadioButton3.CheckedChanged
11         TextBox1.Text = "Você clicou no radio button 3"
12     End Sub
13 End Class
```



Obtendo o estado do Radio Button

Para saber se um determinado radio button foi marcado ou desmarcado basta acessar a propriedade **Checked**. Se for **True** ele está marcado; caso contrário (**False**) está desmarcado.

Exemplo:

```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          If RadioButton2.Checked Then
4              MsgBox("O Radio Button 2 está selecionado.")
5          Else
6              MsgBox("O Radio Button 2 não está selecionado.")
7          End If
8      End Sub
9  End Class
```

Configurando o estado de um Radio Button

Você pode configurar o estado (marcado ou desmarcado) de um radio button usando a propriedade **Checked**.

- Checked = True: Marcado
- Checked = False: Desmarcado

Exemplo:

```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          RadioButton1.Checked = True
4      End Sub
5  End Class
```

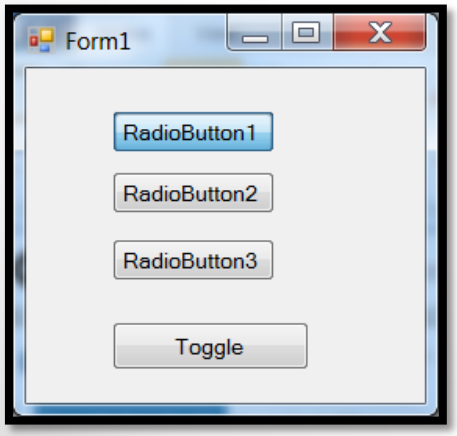
Dica: Para um radio button aparecer marcado quando o programa é iniciado é preciso marcar a opção **Checked** no modo de design.

Criando Toggle Buttons

Os botões de "alternância" podem ser criados a partir de controles do tipo checkbox ou radio button. Para isto funcionar basta alterar a propriedade **Appearance** para **Button**.

Exemplo:

```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          RadioButton1.Appearance = Appearance.Button
4          RadioButton2.Appearance = Appearance.Button
5          RadioButton3.Appearance = Appearance.Button
6      End Sub
7  End Class
```



Usando a classe Panel

- Controle usado para agrupar controles em um Windows form.
- A hierarquia de classes do controle **Panel** é a seguinte:

Object
    MarshalByRefObject
        Component
            Control
                ScrollableControl
                    Panel

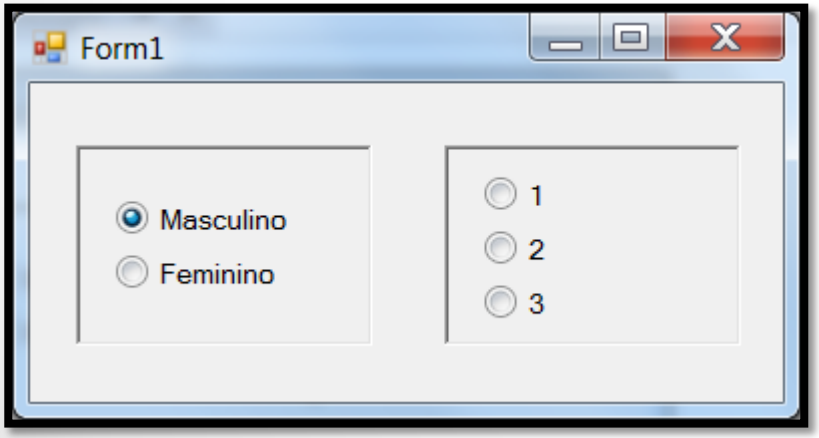
As propriedades dignas de nota de um objeto **Panel** são as seguintes:

- AutoScroll
- AutoScrollMargin
- AutoScrollMinSize
- AutoScrollPosition
- DockPadding

Usando a classe Panel

- Você pode criar painéis em modo de design ou em tempo de execução.
- Eles são úteis em formulários que possuem vários radio buttons que precisam se comportar de forma independente.
- Alterar a propriedade **BorderStyle** para **Fixed3D** deixa seu painel com uma aparência melhor.
- Quando você move um painel ele "leva junto" seus componentes.

Exemplo:



Adicionando controles via código a um Panel

- Você pode criar painéis em tempo de execução e adicionar componentes nele usando a coleção **Controls**.

Exemplo:

```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          Dim Panel1 As New Panel()
4          Dim TextBox1 As New TextBox()
5          Dim Label1 As New Label()
6
7          Panel1.Location = New Point(60, 20)
8          Panel1.Size = New Size(100, 150)
9          Panel1.BorderStyle = BorderStyle.Fixed3D
10
11         Label1.Location = New Point(16, 16)
12         Label1.Text = "Digite um texto: "
13         Label1.Size = New Size(60, 16)
14
15         TextBox1.Location = New Point(16, 32)
16         TextBox1.Text = ""
17         TextBox1.Size = New Size(60, 20)
18
19         Me.Controls.Add(Panel1)
20         Panel1.Controls.Add(Label1)
21         Panel1.Controls.Add(TextBox1)
22     End Sub
23 End Class
```

Usando a classe GroupBox

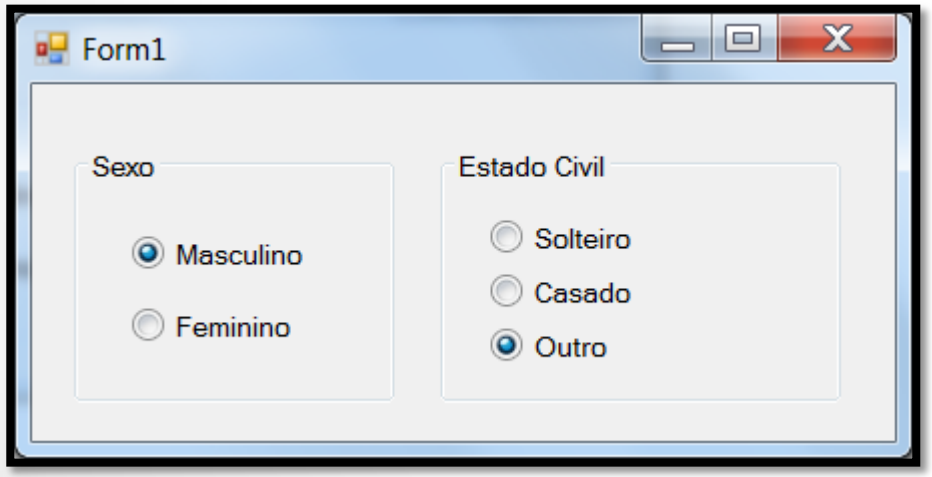
- Controle usado para agrupar controles em um Windows form.
- A diferença deste controle para o Panel é que ele não possui barras de rolagem mas sim um título que você pode colocar para identificar o mesmo.
- A hierarquia de classes do controle **GroupBox** é a seguinte:

Object
    MarshalByRefObject
        Component
            Control
                GroupBox

Criando Group Boxes

- Você pode criar group boxes em modo de design ou em tempo de execução.
- Para mudar o título use a propriedade **Text**.
- Quando você o move ele "leva junto" seus componentes.

Exemplo:



Criando Group Boxes em tempo de execução

Este exemplo é muito grande para ser colado aqui.
Por gentileza verificar o projeto de nome **Slide37**.

Referências Bibliográficas

- HOLZNER, Steven. **Visual basic.NET: black book**. Arizona: Coriolis Group Books, 2002. xxxviii, 1144 p ISBN 1-57610-835-X.