Visual Basic .NET

Windows Forms

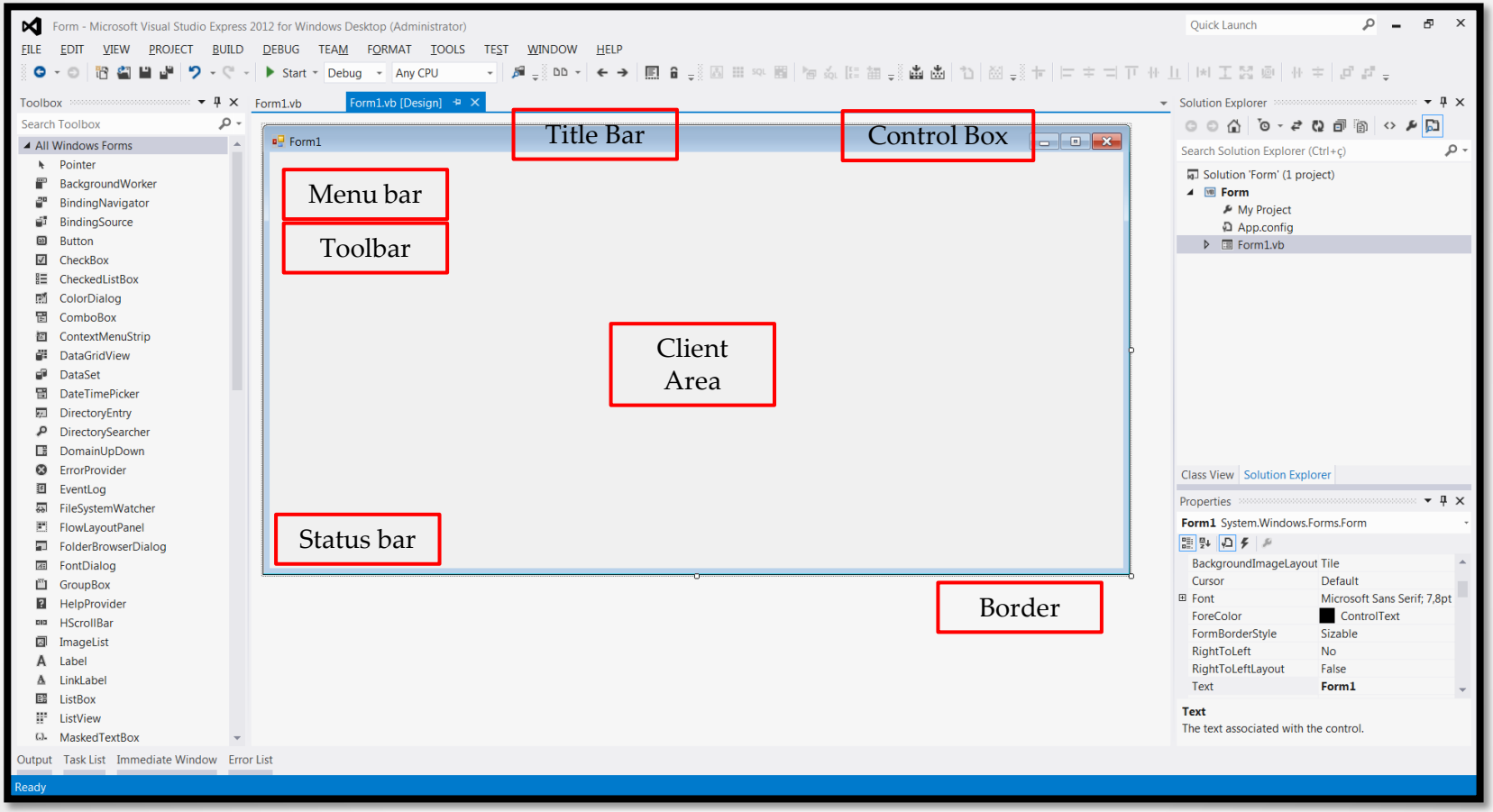
Professor: Danilo Giacobo

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

Objetivos da aula

- ✓ Windows Forms - Propriedades, Métodos e Eventos
- ✓ Windows MDI Forms
- ✓ Criando aplicações Windows
- ✓ Adicionando controles ao formulário
- ✓ Trabalhando com eventos
- ✓ Ajustando o estilo da borda de um form
- ✓ Ajustando a ordem de tabulação
- ✓ Ajustando a posição inicial de um form
- ✓ Movendo e dimensionando Forms
- ✓ Mostrando e escondendo controles e forms
- ✓ Usando as funções MsgBox e InputBox
- ✓ Trabalhando com múltiplos forms
- ✓ Criando Dialog Boxes
- ✓ Criando Forms Proprietários
- ✓ Adicionando/Removendo controles em tempo de execução
- ✓ Usando as funções SendKeys e Beep

Introdução



* Todo **form** colocado no projeto pertence a classe **Form** do namespace **System.Windows.Forms**.

Windows Forms - Propriedades

Quando um form é criado e inserido em um projeto gráfico a ferramenta Visual Studio permite que você ajuste várias propriedades deste.

A lista completa das propriedades disponíveis para a classe Form está no seguinte documento: Visual Basic .NET - Windows Forms Public Object Properties

A primeira coisa a se fazer quando um form é incluído no projeto é renomear o mesmo para que fique mais fácil de manipular ele no código fonte.

Para mudar o nome de um form clique na propriedade **Name** que está no grupo **Design** e coloque um nome para o mesmo. Eu recomendo usar o seguinte padrão:

frm + <Nome do Form> (com a primeira letra de cada nome em maiúscula)

Exemplos: frmCadastroClientes, frmPesquisaProdutos, etc...

Windows Forms - Propriedades

As propriedades mais utilizadas de um Form são:

- **Name**
- BackColor
- BackgroundImage
- ControlBox
- Cursor
- Enabled
- FormBorderStyle
- Icon
- MaximizeBox
- MinimizeBox
- ShowIcon
- ShowInTaskbar
- Size
- StartPosition
- Text
- WindowState

Importante:

Algumas propriedades só podem ser usadas em tempo de execução. Exemplo: **Visible** (torna um form visível ou invisível).

Windows Forms - Propriedades

Atividade Prática 1:

- Crie um novo projeto gráfico chamado **Windows_Forms_Exemplo1** e usando o form já disponível no projeto realize os seguintes ajustes:

1. Modifique o nome do form para **frmExemplo1**.
2. Troque a cor de fundo para a cor "**Info**".
3. Desabilite a **caixa de controle** do form em questão.
4. Mude o cursor do form para a opção **Hand**.
5. Troque a borda do form para **Fixed3D**.
6. Altere o tamanho do form para **300 x 300**.
7. Mude a propriedade **Text** para Exemplo 1.

Compile e execute o projeto para ver o resultado.

Windows Forms - Propriedades

Atividade Prática 2:

- Crie um novo projeto gráfico chamado **Windows_Forms_Exemplo2** e usando o form já disponível no projeto realize os seguintes ajustes nas propriedades:

1. Modifique o nome do form para **frmExemplo2**.
2. Coloque uma **imagem de fundo** qualquer no form.
3. Associe um **ícone** ao form em questão.
4. Mude a propriedade **Text** para Exemplo 2.
5. Não mostre o form na **barra de tarefas** do Windows.
6. Faça o form aparecer **centralizado** na tela.
7. Mostre o form de modo **maximizado**.
8. Desabilite o form.

Compile e execute o projeto para ver o resultado.

Windows Forms - Métodos

Um form possui vários métodos interessantes que podem ser usados em tempo de execução para alterar o comportamento do mesmo e de outros elementos.

A lista dos métodos mais importantes da classe Form está no seguinte documento: Visual Basic .NET - Windows Forms Public Object Methods

Para a lista completa de métodos consulte o endereço:
http://msdn.microsoft.com/en-us/library/system.windows.forms.form_methods(v=vs.110).aspx

Na prática os métodos da classe **Form** só funcionam em tempo de execução e eles precisam de outros componentes para poderem ser usados.

Windows Forms - Eventos

Windows forms também suportam **eventos**. Eventos permitem que você saiba o que está acontecendo em um determinado formulário.

Exemplo: Quando você clica em um formulário, o evento **Click** ocorre e quando um formulário é fechado, o evento **Close** ocorre.

A lista dos eventos mais interessantes para um form está no seguinte documento: Visual Basic .NET - Windows Forms Events

Experimente criar um projeto gráfico e adicione os eventos Click e Close no form para capturar esses eventos e exibir uma mensagem quando eles ocorrerem.

Windows Forms - Eventos

Os principais eventos disparados por meio da interação com o teclado são:

- KeyDown

O evento KeyDown ocorre quando uma tecla qualquer é pressionada e é mantida pressionada.

- KeyPress

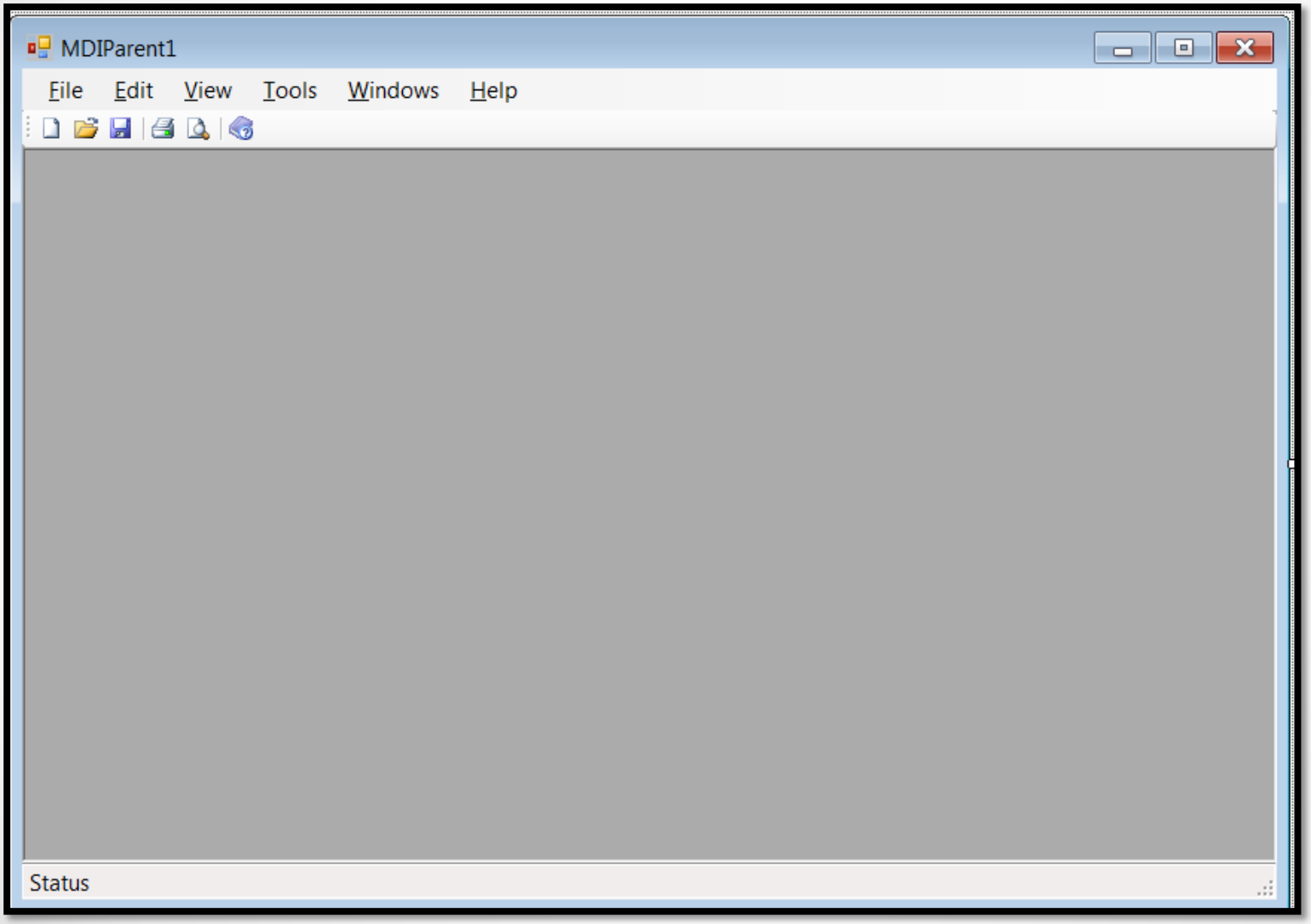
O evento KeyPress ocorre quando uma tecla qualquer é pressionada. Ela não consegue capturar teclas complementares (Shift, Alt, Control, etc...)

- KeyUp

O evento KeyUp ocorre quando uma tecla qualquer é pressionada e é solta logo em seguida.

Experimente criar um projeto gráfico e adicione os eventos acima no form para capturar esses eventos e exibir uma mensagem quando eles ocorrerem.

Windows MDI Forms



Criando aplicações Windows

Quando você cria um no projeto e solução Windows, a ferramenta gera os seguintes diretórios e arquivos:

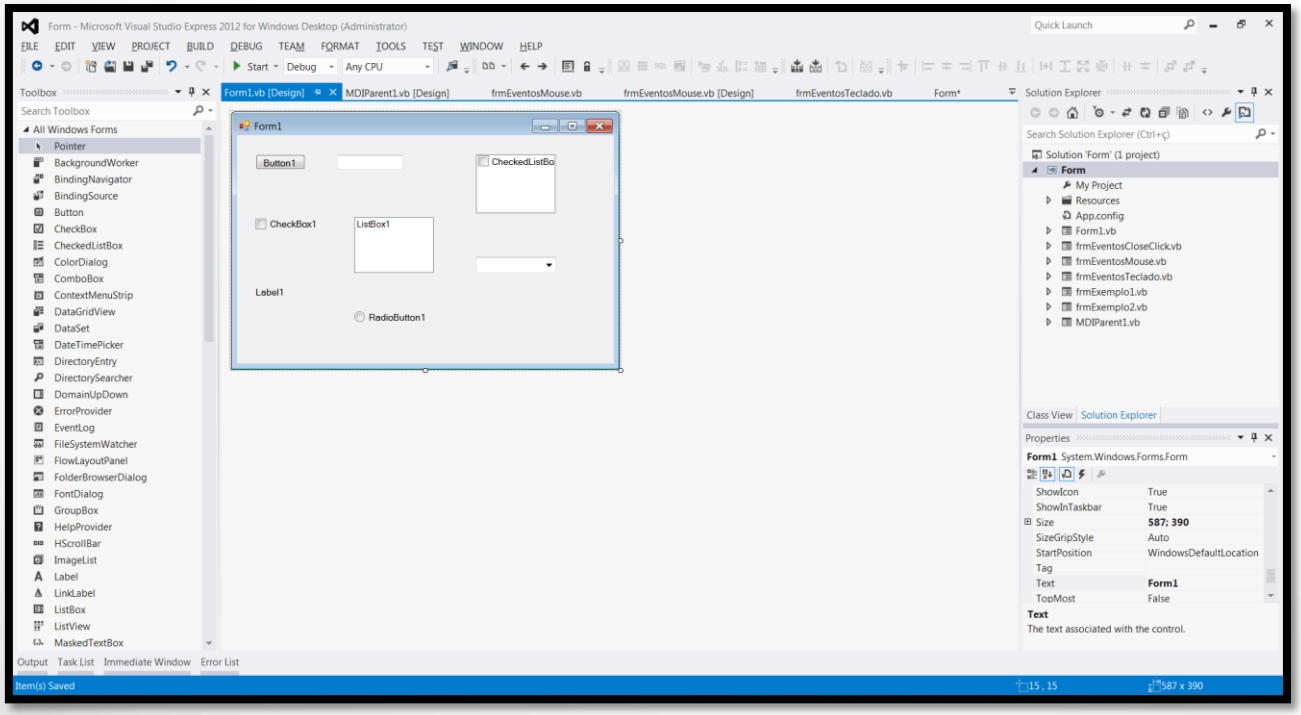
- **Form1.sln**
- **Form1.vb**
- **Form1.resx**
- **Form1.vbproj**
- **App.config**
- **bin**
- **obj**

Experimente criar um projeto gráfico e depois abra a pasta onde o mesmo foi criado para verificar se os itens acima são facilmente localizados. Tente também abrir em um editor de texto os mesmos para ver seu conteúdo.

Adicionando controles ao formulário

No sistema operacional Windows, usuários interagem com ele usando controles gráficos, tais como: barras de rolagem, botões, caixa de texto, menus e muito mais.

No Visual Studio usaremos a caixa de ferramentas (Toolbox) para incluir componentes gráficos de interação com o usuário em nossos programas.



Trabalhando com eventos

Para cada evento a ser tratado a linguagem VB .NET, um procedimento com escopo púbico é criado.

A maioria dos procedimento que tratam eventos possui dois argumentos:

- O objeto que causou o evento (**sender**) e
- Um objeto do tipo **EventArgs** com mais informações sobre o evento.

```
1  Public Class Form1
2      Private Sub Form1_Click(sender As Object, e As EventArgs) Handles Me.Click
3
4      End Sub
5  End Class
```

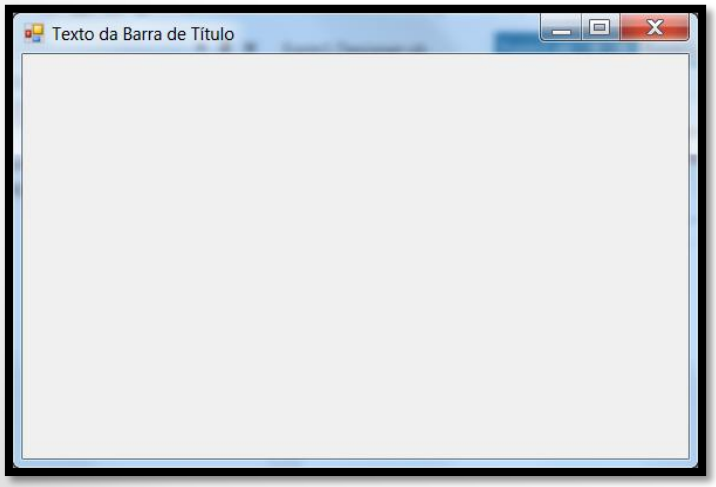
A parte **Handles Me.Click** significa que o procedimento **Form1_Click** trata do evento **Click** do form **Form1**.

Ajustando o texto da barra de título

Para alterar o texto da barra de título de um form você usa a propriedade **Text** atribuindo a ela um título para o seu formulário. Você pode fazer isso no modo design ou em tempo de execução:

Exemplo:

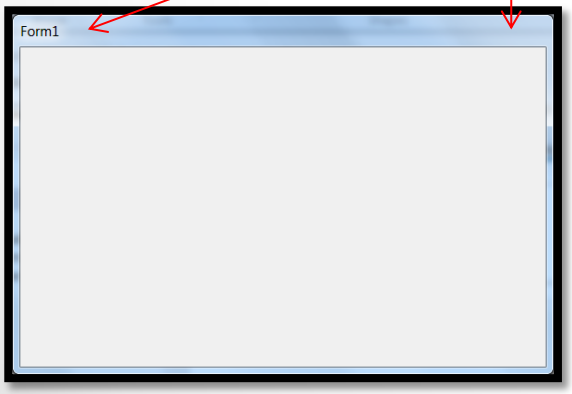
```
1  Public Class Form1
2      Private Sub Form1_Load(sender As Object, e As EventArgs) Handles Me.Load
3          Me.Text = "Texto da Barra de Título"
4      End Sub
5  End Class
```



Adicionando/Removendo Botões

Todo form que é criado em um projeto contém 3 botões ao lado direito da barra de título. Um serve para minimizar a janela, outro para maximizar ela e o último para fechar esta. Você pode removê-los alterando o valor da propriedade **ControlBox** para **False**. Essa alteração pode ser feita de forma independente para os botões de maximizar e minimizar usando as propriedades **MaximizeBox** e **MinimizeBox**. Exemplo:

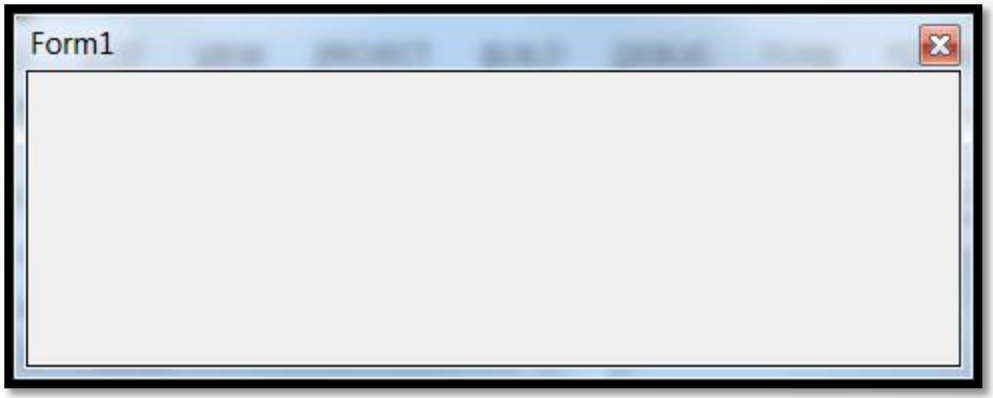
```
1  Public Class Form1
2      Private Sub Form1_Load(sender As Object, e As EventArgs) Handles Me.Load
3          Me.ControlBox = False
4      End Sub
5  End Class
```



Ajustando o estilo da borda de um Form

Para ajustar o estilo de borda de um form Windows você usa a propriedade **FormBorderStyle**. Os possíveis valores para esta propriedade são:

- **Fixed3D**
- **FixedDialog**
- **FixedSingle**
- **FixedToolWindow**
- **None**
- **Sizable**
- **SizableToolWindow**



Ajustando a ordem de tabulação

De acordo com a Microsoft: "*Usuários poderiam executar todos os programas baseados na interface Windows apenas pelo teclado*".

Cada componente que você insere em um form possui um número inteiro único (começando em 0) que representa a ordem de tabulação deste no form. Para ajustar a ordem de tabulação em seu programa siga esses passos:

1. Selecione o controle cuja ordem você quer ajustar.
2. Verifique se o valor da propriedade **TabStop** é **True**. Se esse valor for **False** o componente não recebe o foco da tecla **Tab**.
3. Ajuste o valor da propriedade **TabIndex** com o número desejado (não podem haver componentes com valores iguais para esta propriedade) .
4. Quando você executa o programa o primeiro controle (com **TabIndex = 0**) recebe o foco e o cursor fica piscando ou ele é destacado (depende do tipo de componente).

Ajustando a posição inicial de um form

Você pode usar a propriedade **StartPosition** para definir onde na tela do computador o seu formulário irá aparecer quando o programa for iniciado. Os possíveis valores da propriedade **StartPosition** (que é uma enumeração de valores) são:

- **CenterParent**
- **CenterScreen**
- **Manual**
- **WindowsDefaultBounds**
- **WindowsDefaultLocation**

Para ajustar essa propriedade em tempo de execução use o seguinte modelo:

```
Form.StartPosition = FormStartPosition.CenterScreen
```

Movendo e Dimensionando Forms

A linguagem VB .NET permite que você mova e altere as dimensões de um formulário em tempo de execução.

Exemplo:

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

Dica:

Você pode usar o método **SetBounds** para obter o mesmo resultado:

SetBounds(0, 0, 100, 100)

Button1.SetBounds(0, 0, 100, 100)

Mostrando e Escondendo Controles e Forms

Mostrar e Esconder controles e formulários em um programa é fácil: basta usar a propriedade **Visible**. Se o valor for **True** o controle/form é **mostrado**; se o valor for **False** ele é **escondido** do usuário.

Exemplo:

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

Dica:

Você pode usar os métodos **Show** e **Hide** para mostrar ou esconder um controle/form.

```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          Button1.Hide()
4          TextBox1.Show()
5      End Sub
6  End Class
```

Usando a função MsgBox

Uma das funções mais importante e mais usada em um projeto visual é a conhecida **MsgBox**.

Ela permite que você exiba mensagens com tom de informação, aviso, erro, entre tantas outras possibilidade além de poder funcionar como uma forma de interação com o usuário.

A sua forma de utilização é a seguinte:

Public Function MsgBox(*Prompt* As Object [, *Buttons* As MsgBoxStyle = MsgBoxStyle.OKOnly [, *Title* As Object = Nothing]]) As MsgBoxResultArguments

* O que está entre [] é opcional.

O argumento *Buttons* é uma combinação de possíveis valores de constantes definidas na linguagem VB .NET. Cada uma delas representa o tipo de mensagem a ser utilizada. A lista de constantes e sua respectiva descrição pode ser vista no documento Visual Basic .NET - MsgBox constants

Usando a função MsgBox

Exemplos:

```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          Dim intResult As Integer
4          intResult = MsgBox("Esta é uma Caixa de Mensagem!", MsgBoxStyle.OkCancel +
5                             MsgBoxStyle.Information + MsgBoxStyle.SystemModal, "Caixa de Mensagem")
6
7          If intResult = MsgBoxResult.Ok Then
8              TextBox1.Text = "Você clicou no botão OK."
9          End If
10     End Sub
11 End Class
```

```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          MsgBox("Erro Fatal!", MsgBoxStyle.Critical + MsgBoxStyle.AbortRetryIgnore, "ERRO")
4      End Sub
5  End Class
```

Usando o método MessageBox.Show

Além da função **MsgBox** vista no slide anterior você pode usar o método **Show** da classe **MessageBox** para mostrar caixas de mensagens. O resultado é idêntico.

Exemplo:

```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          Dim intResult As Integer
4          intResult = MessageBox.Show("Esta é uma Caixa de Mensagem!", "Caixa de Mensagem", MessageBoxButtons.OKCancel, _
5                                      MessageBoxIcon.Information, MessageBoxDefaultButton.Button1, _
6                                      MessageBoxOptions.DefaultDesktopOnly)
7
8          If intResult = DialogResult.OK Then
9              TextBox1.Text = "Você clicou no botão OK."
10         End If
11     End Sub
12 End Class
```

Usando a função InputBox

Você pode usar a função **InputBox** para obter uma cadeia de caracteres digitada pelo usuário.

Exemplo:

```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          Dim strTexto As String
4          strTexto = InputBox("Digite um texto!", "Input Box", "Você não digitou nada.")
5          TextBox1.Text = strTexto
6      End Sub
7  End Class
```

Trabalhando com múltiplos forms

- Para adicionar um novo form ao seu projeto basta clicar com o botão direito do mouse em cima do nome do projeto e escolher a opção **Add -> Windows Form...** Um novo form será alocado no projeto.

Exemplo:

```
1  Public Class Form1
2      Dim frmOutraJanela As New Form2()
3
4      Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
5          frmOutraJanela.Show()
6      End Sub
7
8      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
9          TextBox1.Text = frmOutraJanela.TextBox1.Text
10         frmOutraJanela.Hide()
11     End Sub
12 End Class
```

Usando propriedades para comunicação entre forms

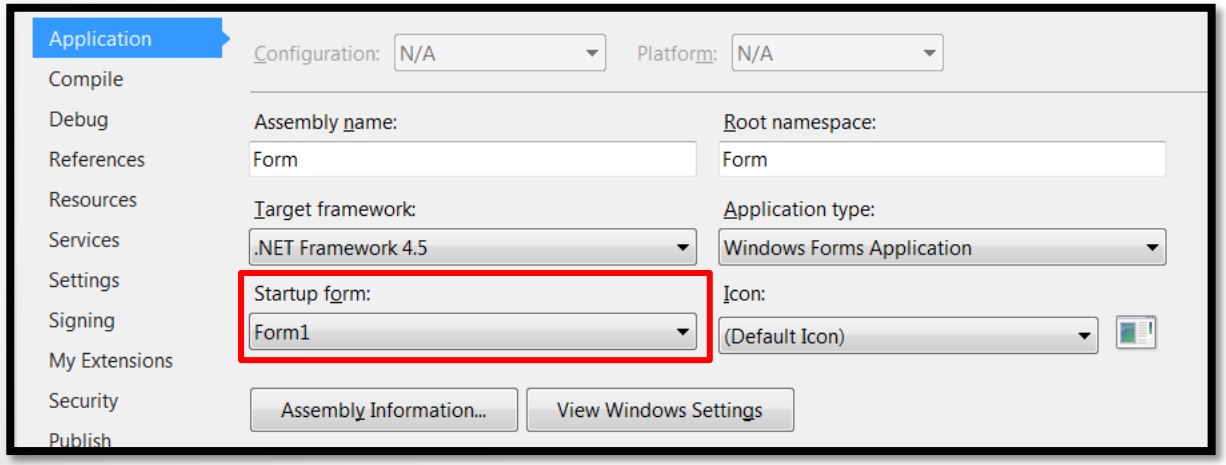
```
1  Public Class Form2
2      Property TextData() As String
3          Get
4              Return TextBox1.Text
5          End Get
6          Set(value As String)
7              TextBox1.Text = value
8          End Set
9      End Property
10 End Class
```

```
1  Public Class Form1
2      Dim frmOutraJanela As New Form2()
3
4      Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
5          frmOutraJanela.Show()
6      End Sub
7
8      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
9          'TextBox1.Text = frmOutraJanela.TextBox1.Text
10         TextBox1.Text = frmOutraJanela.TextData
11         frmOutraJanela.Hide()
12     End Sub
13 End Class
```

Configurando o form de inicialização

Imagine a situação: seu projeto visual de 1 milhão de dólares com 240 forms criados e funcionando perfeitamente. No seu primeiro uso em vez de aparecer o form de login aparece o form de importação de arquivos texto ou a tela de impressão de relatórios...

Para corrigir esse problema, isto é, definir qual é o form inicial do seu projeto, basta clicar com o botão direito do mouse em cima do projeto e na lista de opções chamada **Startup form** selecionar o form desejado.

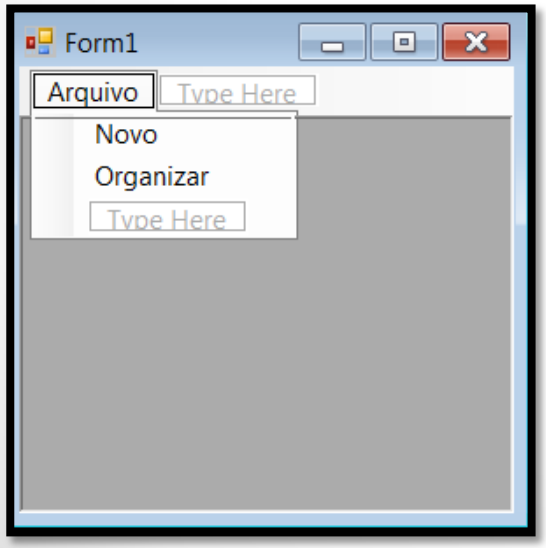


Criando Aplicações MDI (Multiple Document Interface)

- Você usa aplicações MDI para poder mostrar múltiplas janelas filhas dentro de um mesmo form. A própria IDE do Visual Basic funciona assim. Editores de texto fazem imenso uso deste tipo de aplicação também.

- Crie um novo projeto visual e no primeiro form do projeto ajuste a propriedade **IsMdiContainer** para **True** (a aparência do form ficará diferente).

- Arraste para o form um componente **MenuStrip** e crie os itens de menu abaixo:



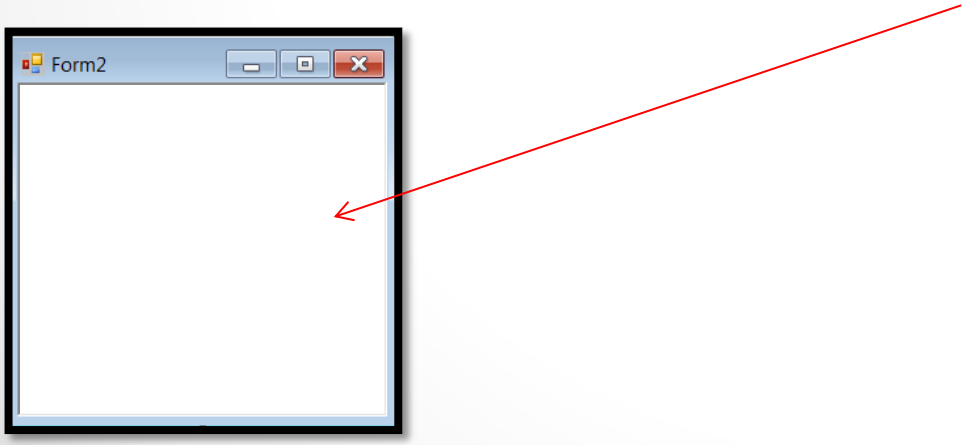
Criando Aplicações MDI (Multiple Document Interface)

- Dê um duplo clique no item de menu **Novo**. Um código para tratamento do evento **Click** será criado.

```
1  Public Class Form1
2      Private Sub NovoToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles NovoToolStripMenuItem.Click
3
4      End Sub
5  End Class
```

- Adicione um novo form (para servir de form filho) ao projeto e coloque no mesmo um componente do tipo **RichTextBox**. Ajuste a propriedade **Multiline** para **True**.

- Para garantir que um componente **RichTextBox** use toda a área disponível do form é necessário alterar a propriedade **Dock** para o valor **Fill**.



Criando Aplicações MDI (Multiple Document Interface)

- Cada vez que a pessoa clicar na opção **Novo** no menu um novo form (filho) será criado. O código abaixo mostra este processo:

```
1  Public Class Form1
2      Dim intNumeroForms As Integer = 0
3      Dim arrForms(10) As Form2
4
5      Private Sub NovoToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles NovoToolStripMenuItem.Click
6          intNumeroForms += 1
7          arrForms(intNumeroForms) = New Form2
8          arrForms(intNumeroForms).Text = "Documento " & Str(intNumeroForms)
9          arrForms(intNumeroForms).MdiParent = Me
10         arrForms(intNumeroForms).Show()
11     End Sub
12 End Class
```

Dica: Você pode usar a propriedade **MdiChildren** de um form MDI pai para obter um array de forms MDI filhos.

Criando Aplicações MDI (Multiple Document Interface)

- Por fim para organizar as janelas filhas de um form MDI é utilizado o método **LayoutMdi**. Ele possui apenas um argumento que é um valor de uma enumeração chamada **MdiLayout**. Os possíveis valores são:

- **ArrangeIcons**
- **Cascade**
- **TileHorizontal**
- **TileVertical**

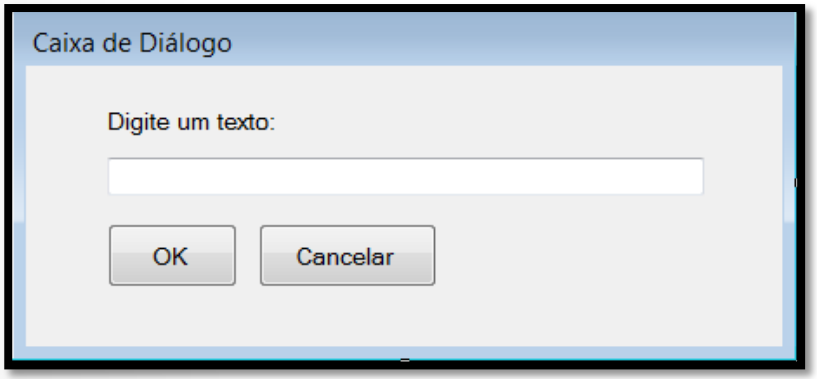
Exemplo:

```
Private Sub OrganizarToolStripMenuItem_Click(sender As Object, e As EventArgs) _
    Handles OrganizarToolStripMenuItem.Click
    Me.LayoutMdi(MdiLayout.TileHorizontal)
End Sub
```

Criando Dialog Boxes

Você pode criar suas próprias caixas de diálogo em VB .NET em vez de ter que sempre usar **Message Boxes** e **Input Boxes** que são restritas. Siga os seguintes passos para criar uma nova caixa de diálogo customizável:

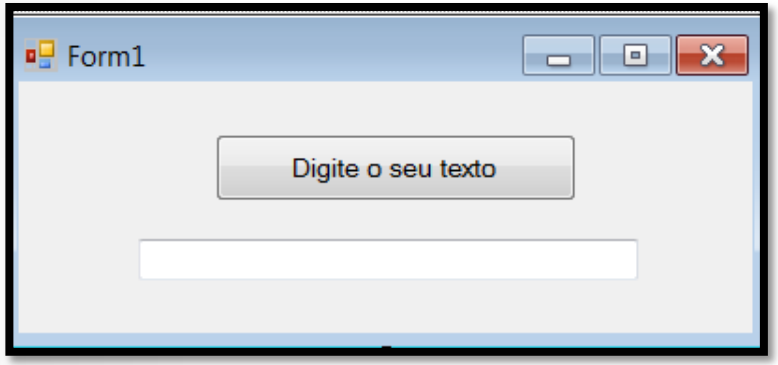
1. Adicione um **novo form** ao projeto chamado **Form2** e adicione **dois botões**, **um label** e **uma caixa de texto** conforme mostra a imagem abaixo:



2. Altere o valor da propriedade **FormBorderStyle** para **FixedDialog** e o valor da propriedade **ControlBox** para **False**. Ajuste também a propriedade **ShowInTaskbar** para **False**. Por fim ajuste a propriedade **DialogResult** do botão OK para **OK** e faça o mesmo para o botão Cancelar (**Cancel**).

Criando Dialog Boxes

3. No **formulário principal** adicione **um botão** e uma **caixa de texto** para testar a caixa de diálogo criada. Para chamar a caixa de diálogo a partir de um form é necessário usar o método **ShowDialog**.




```
1  Public Class Form1
2
3      Dim DialogBox As New Form2()
4
5      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
6          If DialogBox.ShowDialog = DialogResult.OK Then
7              TextBox1.Text = DialogBox.TextBox1.Text
8          End If
9      End Sub
10 End Class
```

Criando Dialog Boxes

4. Por último é necessário incluir o código que fecha o diálogo quando a pessoa clica em um botão e também alterar as propriedades AcceptButton e CancelButton para que a pessoa possa pressionar a tecla Enter e Esc para interagir com os botões.

```
1  Public Class Form2
2      Private Sub Form2_Load(sender As Object, e As EventArgs) Handles MyBase.Load
3          Me.AcceptButton = Button1
4          Me.CancelButton = Button2
5      End Sub
6
7      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
8          Me.Close()
9      End Sub
10
11     Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
12         Me.Close()
13     End Sub
14 End Class
```

Dica: Uma boa prática ao trabalhar com caixas de diálogo é sempre colocar um botão para fechar a mesma (CancelButton).

Criando Forms Proprietários

Você pode criar também em Visual Basic .NET os chamados "owned forms". Ele é atrelado a um formulário que "manda" nele. Isto significa que uma mudança nele reflete em outro (por exemplo, minimizar uma janela).

Exemplo:

```
1  Public Class Form1
2      Dim OwnedForm As New Form2()
3
4      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
5          Me.AddOwnedForm(OwnedForm)
6          OwnedForm.Show()
7      End Sub
8  End Class
```

Dicas:

- Para **adicionar** um "owned form" use o método **AddOwnedForm**.
- Para **remover** um "owned form" use o método **RemoveOwnedForm**.

Passando Forms para Procedimentos

Você pode passar um objeto do tipo **Form** para um procedimento. No exemplo abaixo o procedimento **PaintItBlack** muda a cor de fundo de um form para preto.

Exemplo:

```
1  Public Class Form1
2      Sub PaintItBlack(ByVal frmForm As Form)
3          frmForm.BackColor = Color.Black
4      End Sub
5
6      Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
7          PaintItBlack(Me)
8      End Sub
9  End Class
```

Minimizando/Maximizando Forms

Para garantir um pouco mais de controle sobre as janelas em seus programas, você pode modificar a propriedade **WindowState** para minimizar ou maximizar as mesmas. As possíveis configurações essa propriedade são:

- **FormWindowState.Maximized**
- **FormWindowState.Minimized**
- **FormWindowState.Normal**

```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          Me.WindowState = FormWindowState.Minimized
4      End Sub
5  End Class
```

Dica: Você pode usar a propriedade **Enabled** para habilitar (True) ou desabilitar (False) uma janela. Quando ela está desabilitada a pessoa recebe apenas um beep se tentar colocar o foco na mesma.

Adicionando/Removendo controles em tempo de execução

Você pode adicionar ou remover controles de sua aplicação em tempo de execução. Tudo que você precisa fazer é usar a coleção **Controls** e os métodos **Add** e **Remove**.

Exemplo:

```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          Dim txtNovoTextBox As New TextBox()
4          txtNovoTextBox.Size = New Size(100, 20)
5          txtNovoTextBox.Location = New Point(100, 100)
6          txtNovoTextBox.Text = "Desenvolvimento Visual"
7          Me.Controls.Add(txtNovoTextBox)
8      End Sub
9  End Class
```

Outras configurações interessantes

- Propriedade **Anchor**
- Propriedade **Dock**
- Propriedade **TopMost**
- Métodos **BringToFront** e **SendToBack**

Manipulando eventos do Mouse

Os possíveis eventos para tratamento/manipulação por meio da interação com o mouse são os seguintes:

- **MouseDown**
  - Ocorre quando o ponteiro do mouse está sobre um controle e o botão é pressionado.
- **MouseEnter**
  - Ocorre quando o ponteiro do mouse entra em um controle.
- **MouseHover**
  - Ocorre quando o ponteiro do mouse paira sobre um controle.
- **MouseLeave**
  - Ocorre quando o ponteiro do mouse deixa um controle.
- **MouseMove**
  - Ocorre quando o ponteiro do mouse é movido sobre um controle.
- **MouseUp**
  - Ocorre quando o ponteiro do mouse está sobre um controle e o botão é liberado.
- **MouseWheel**
  - Ocorre quando a roda do mouse se move enquanto o controle tem o foco.

Manipulando eventos do Mouse

As propriedades do objeto **MouseEventArgs** passadas para o tratador de eventos são:

- **Button**
- **Clicks**
- **Delta**
- **X**
- **Y**

A propriedade **Button** é uma enumeração e contém os seguintes membros:

- *Left*
- *Middle*
- *None*
- *Right*
- *XButton1*
- *XButton2*

Função SendKeys

A função **SendKeys** permite que você envie comandos de dentro de um programa escrito em Visual Basic para um outro aplicativo que está em foco.

Exemplo:

```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          AppActivate("Documento1 - Microsoft Word")
4          System.Windows.Forms.SendKeys.Send("Boa tarde Word!")
5          System.Windows.Forms.SendKeys.Send("~~Estou invadindo o seu documento!")
6      End Sub
7  End Class
```

O caractere "~" significa que eu enviei um ENTER para o aplicativo.

Para ver o código de outras teclas consulte o seguinte arquivo:

Visual Basic .NET – SendKeys Key Codes

A Função Beep

A função **Beep** não se encaixa em nenhuma categoria mas pode vir a calhar dependendo da situação. Quando chamada, essa função faz o computador emitir um beep (um alarme sonoro).

Exemplo (emite o som das notas musicais):

```
1   Imports System.Console
2
3   Module Module1
4       Sub Main()
5           Beep()
6           Beep(264, 1000)
7           Beep(297, 1000)
8           Beep(330, 1000)
9           Beep(352, 1000)
10          Beep(396, 1000)
11          Beep(440, 1000)
12          Beep(495, 1000)
13          Beep(528, 1000)
14      End Sub
15  End Module
```

Referências Bibliográficas

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