



# **PBP - Capacitação em Programação .NET (WFA – *Windows Forms Application*)**

## **Semana 3**

Estudos dos componentes: ComboBox, ListBox, Timer, DateTimePicker e seus métodos, eventos e propriedades.

**Prof. Fabrício Braoios Azevedo**

**Prof. Tiago Jesus de Souza**

# ComboBox

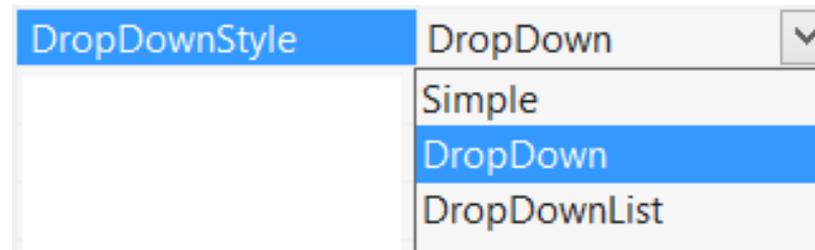
## ComboBox

A função do **combobox**, é armazenar e exibir uma lista de itens (opções) dentro do formulário. Para este componente estão disponíveis 3 (três) modelos, são eles:



Este modelo não mostra a lista de itens. Para selecionar um item devemos usar as **setas para cima e para baixo** do teclado.

Para selecionar o estilo do **combobox** utilizaremos a propriedade **DropDownStyle**.



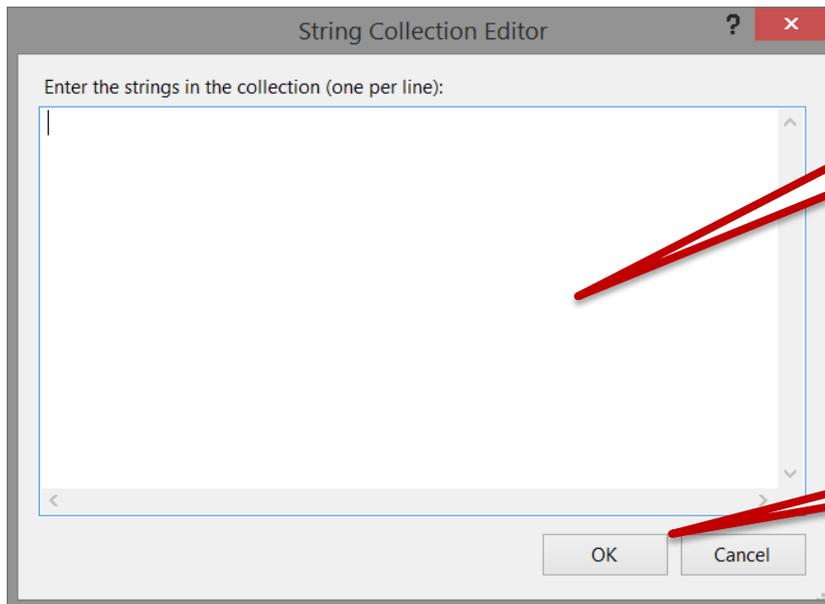
# ComboBox

Podemos adicionar itens (opções) de 2 (duas) formas, através da propriedade **Items** ou pela programação.

## Adicionando itens através da propriedade Items



Clicando no botão  aparecerá a seguinte janela:



Neste espaço devem ser digitados os itens que serão listados no componente.

Ao terminar clicar no botão **OK**.

# ComboBox

Podemos adicionar itens (opções) de 2 (duas) formas, através da propriedade **Items** ou pela programação.

## Adicionando itens através da programação

```
ComboBox1.Items.Add(<item>);
```

<item> → Texto no qual será adicionado ao componente.

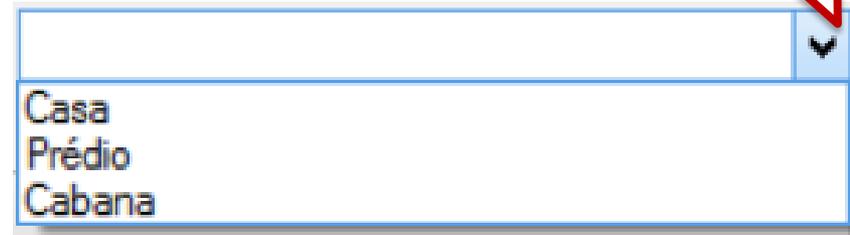
Nome (**Propriedade Name**) do componente.

Método para adicionar um item dentro do componente.

Para listar todos os itens, clicar neste botão.

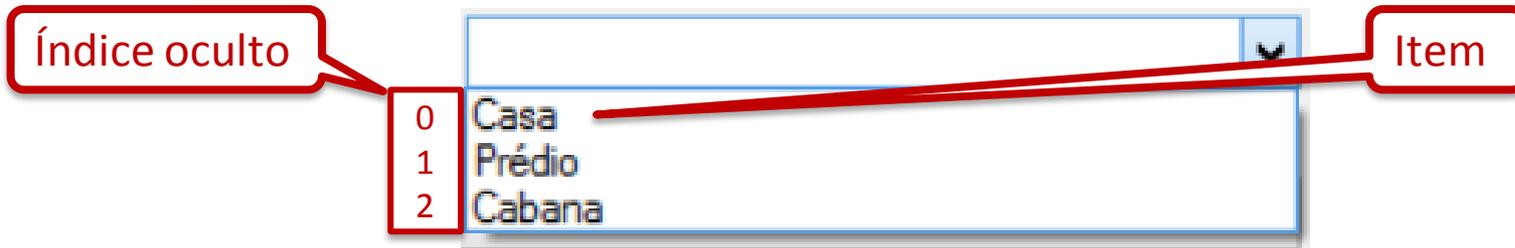
## Exemplo:

```
comboBox1.Items.Add("Casa");  
comboBox1.Items.Add("Prédio");  
comboBox1.Items.Add("Cabana");
```



# ComboBox

Cada item adicionado possui um índice (oculto) associado a ele. O índice sempre começará com **0 (Zero)**.



O índice **-1** serve para desmarcar qualquer item selecionado, ou validar se nenhum item foi selecionado.

## Removendo item através da programação

```
ComboBox1.Items.RemoveAt(<índice>);
```

Método para remover um item do componente.

**<índice>** → Índice do item que será eliminado.

# ComboBox

Podemos manipular os itens (opções) através das seguintes propriedades:

- ✓ **comboBox1.SelectedIndex** → Retorna o índice do item que está selecionado.
- ✓ **comboBox1.SelectedItem** → Retorna a descrição do item que está selecionado.
- ✓ **comboBox1.Items.Count** → Retorna a quantidade total de itens do componente.

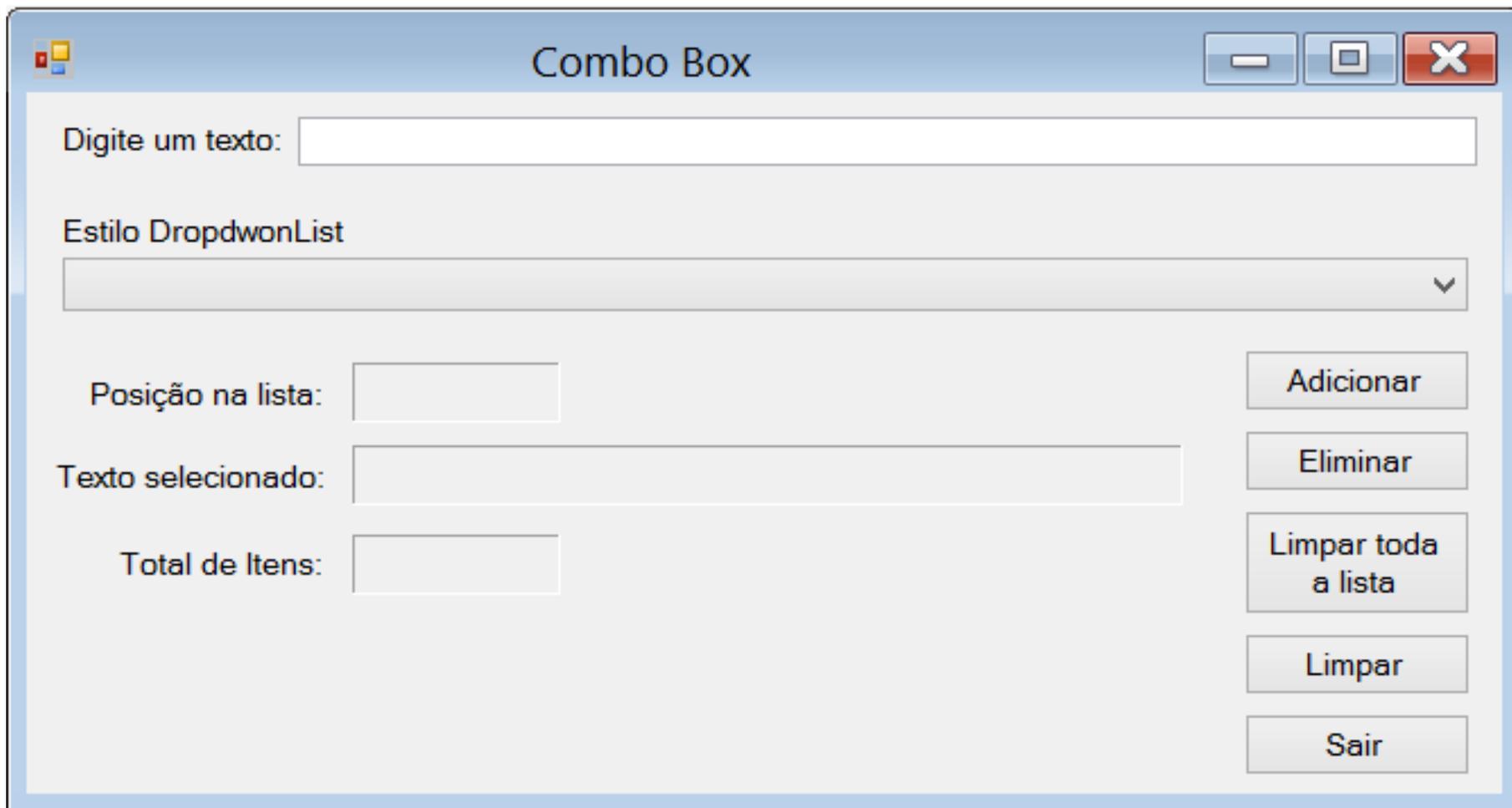
Caso precise ordenar os itens em ordem alfabética , utilize a propriedade **Sorted**. Por default (padrão) o seu valor é **false (falso)**, caso alterar para **true (verdadeiro)**, todos os itens ficarão em ordem.

O evento default para o ComboBox é o **SelectedIndexChanged**, ou seja, quando selecionamos um item, será executado o código de programação que estiver dentro.

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{

}
```

# Exemplo – ComboBox



The image shows a Windows-style application window titled "Combo Box". The window contains the following elements:

- A text input field with the label "Digite um texto:".
- A dropdown menu labeled "Estilo DropdwonList" (note the typo) with a downward arrow.
- A text input field labeled "Posição na lista:".
- A text input field labeled "Texto selecionado:".
- A text input field labeled "Total de Itens:".
- A vertical stack of five buttons on the right side: "Adicionar", "Eliminar", "Limpar toda a lista", "Limpar", and "Sair".

# Exemplo – ComboBox

The image shows a software application window titled "o Box" with several input fields and buttons. Red callout boxes provide explanations for each element:

- Input field:** "Digite um texto:" - Digitar um texto para adicionar no ComboBox.
- Dropdown menu:** "Estilo DropdwonList" - Serve para adicionar o texto digitado no ComboBox.
- Dropdown arrow:** - Serve para eliminar o item selecionado do ComboBox.
- Buttons:** "Adicionar", "Eliminar", "Limpar toda a lista", "Limpar", "Sair".
- Input field:** "Posição na lista:" - Exibir o índice do item selecionado.
- Input field:** "Texto selecionado:" - Exibir a descrição do item selecionado.
- Input field:** "Total de Itens:" - Exibir a quantidade total de itens do ComboBox.
- Buttons:** "Limpar toda a lista" - Limpar todos os itens do ComboBox.
- Buttons:** "Limpar" - Limpar a caixa de texto e o item selecionado no ComboBox.

# Exemplo – Código Fonte

```
private void btnAdicionar_Click(object sender, EventArgs e)
{
    cboListaDropDownList.Items.Add(txtTexto.Text);
    txtTexto.Clear();
    txtTexto.Focus();
}
```

Evento que executa quando pressionar um botão.

Através do método **Add**, iremos adicionar o texto digitado na caixa de texto (txtTexto), para a ComboBox (cboListaDropDownList).

Este evento serve para colocar o cursor (focar) dentro da caixa de texto (txtTexto).

Este método faz limpar o conteúdo de uma caixa de texto. (**O Clear() não funciona para Label**)

# Exemplo – Código Fonte

```
private void btnEliminar_Click(object sender, EventArgs e)
{
    if (cboListaDropDownList.SelectedIndex == -1)
    {
        MessageBox.Show("Nenhum item foi selecionado!!!", "ComboBox", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    else
    {
        cboListaDropDownList.Items.RemoveAt(cboListaDropDownList.SelectedIndex);
    }
}
```

Verificar se não foi selecionado nenhum item na ComboBox, caso não foi selecionado, exibir uma mensagem.

Através do método **RemoveAt** iremos remover o item selecionado da ComboBox (cboListaDropDownList), através do índice do item selecionado.

```
private void btnLimparLista_Click(object sender, EventArgs e)
{
    cboListaDropDownList.Items.Clear();
}
```

Remover (Limpar) todos os itens do ComboBox.

# Exemplo – Código Fonte

```
private void btnLimpar_Click(object sender, EventArgs e)
{
    1 txtTexto.Clear();
    2 cboListaDropDownList.SelectedIndex = -1;
    3 lblPosLista.Text = "";
    4 lblTextoSel.Text = "";
    5 lblTotal.Text = "";
    6 txtTexto.Focus();
}
```

1. Limpar o conteúdo da caixa de texto
2. Desmarcar o item selecionado na ComboBox (cboListaDropDownList)
3. Limpar o conteúdo da label (lblPosLista)
4. Limpar o conteúdo da label (lblTextoSel)
5. Limpar o conteúdo da label (lblTotal)
6. Colocar o cursor (foco) na caixa de texto (txtTexto)

# Exemplo – Código Fonte

Verifica se foi selecionado um item na ComboBox.

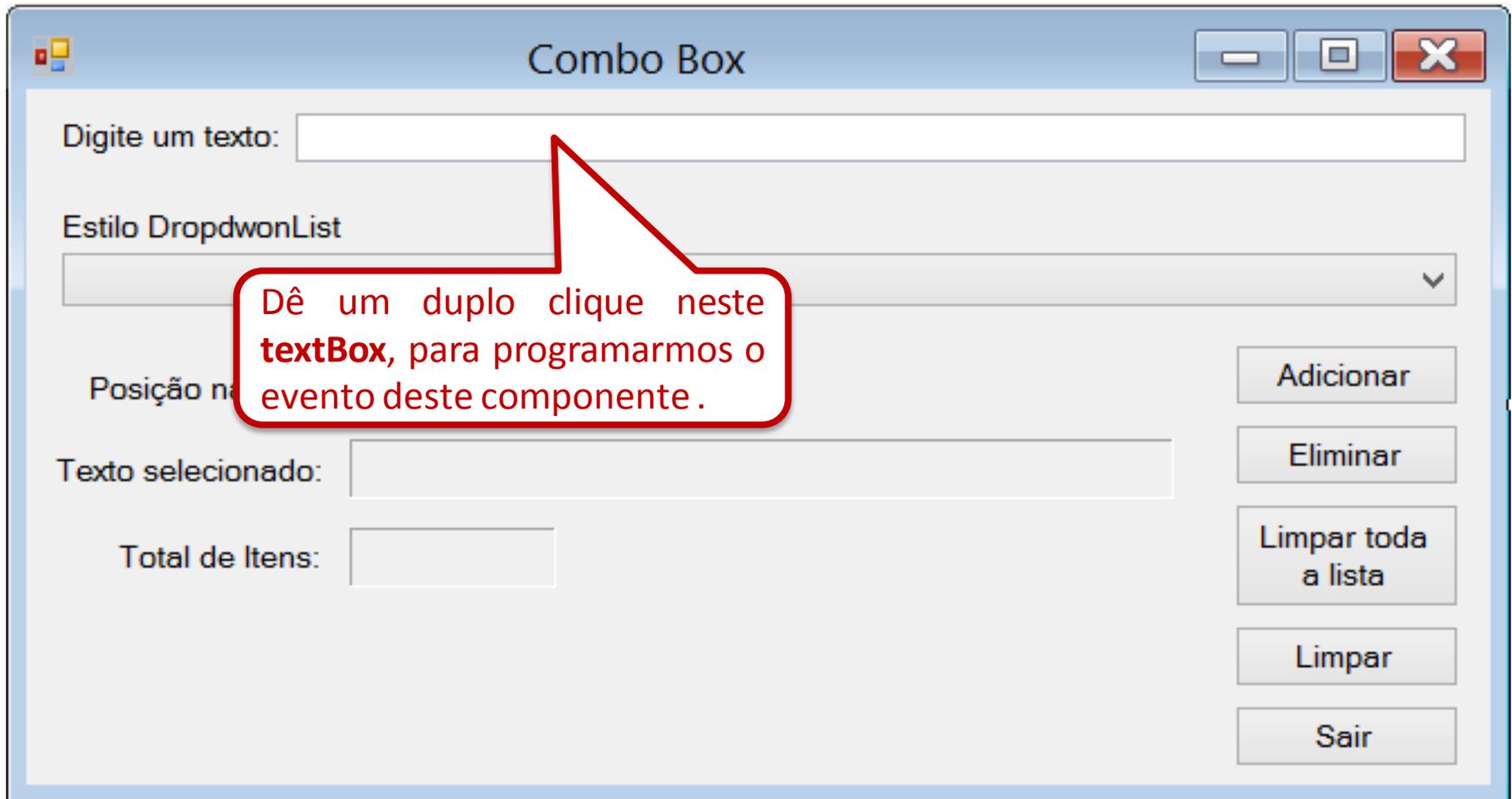
```
private void cboListaDropDownList_SelectedIndexChanged(object sender, EventArgs e)
{
    if (cboListaDropDownList.SelectedIndex != -1)
    {
        lblPosLista.Text = cboListaDropDownList.SelectedIndex.ToString();
        lblTextoSel.Text = cboListaDropDownList.SelectedItem.ToString();
        lblTotal.Text = cboListaDropDownList.Items.Count.ToString();
    }
}
```

Este método (**ToString()**) serve para converter um número em texto (string).

```
private void btnSair_Click(object sender, EventArgs e)
{
    Application.Exit();
}
```

Sair da aplicação.

# Exemplo – ComboBox



The screenshot shows a Windows application window titled "Combo Box". The window contains several UI elements: a text input field labeled "Digite um texto:", a dropdown menu labeled "Estilo DropdwonList", a text input field labeled "Texto selecionado:", and another text input field labeled "Total de Itens:". On the right side, there are five buttons: "Adicionar", "Eliminar", "Limpar toda a lista", "Limpar", and "Sair". A red callout box with a white background and a red border points to the "Digite um texto:" text input field. The text inside the callout box is: "Dê um duplo clique neste **textBox**, para programarmos o evento deste componente."

# Exemplo – Código Fonte

Propriedade do evento **KeyPress** que armazena a tecla pressionada.

Evento que permite verificar a tecla pressionada.

```
private void txtTexto_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == 13)
    {
        btnAdicionar_Click(sender, e);
    }
}
```

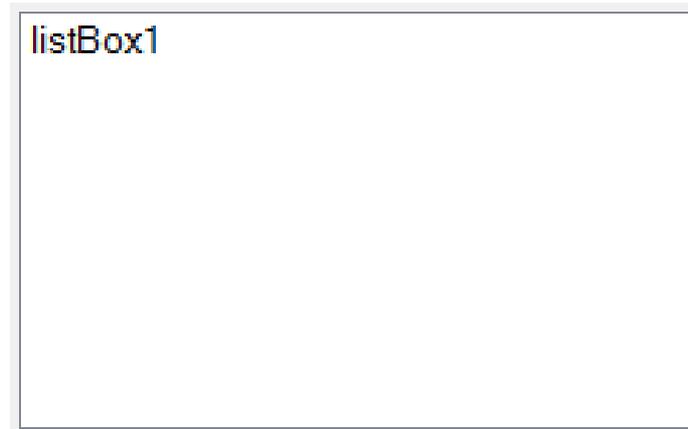
O valor 13 equivale a tecla **ENTER** do teclado.

Quando a tecla **ENTER** for pressionada, será executado o evento **Click** do botão **Adicionar**, passando como parâmetro os mesmos do componente atual. (Neste caso estamos **redirecionando ao mesmo código do botão**)

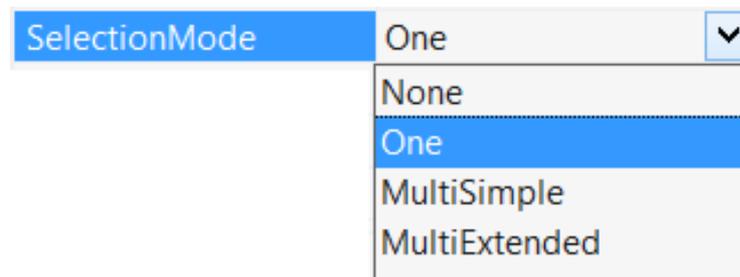
# ListBox

## ListBox

A função do **listbox**, é armazenar e exibir uma lista de itens (opções) dentro do formulário.



Para este componente estão disponíveis 3 (três) modos para selecionar os itens através da propriedade **SelectMode**, são eles:



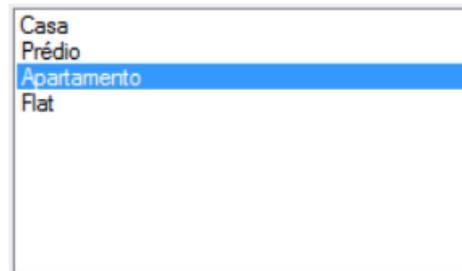
# ListBox

## Propriedade SelectedMode

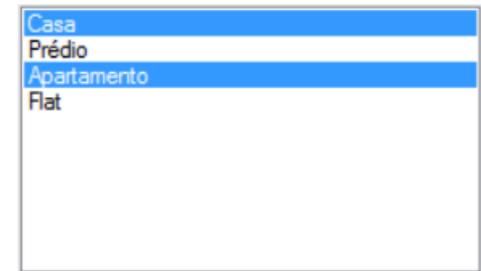
### None



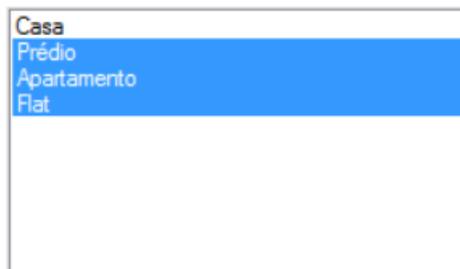
### One



### MultiSimple



### MultiExtended



**None** → Não permite selecionar nenhum item.

**One** → Permite selecionar **somente** um item por vez.

**MultiSimple** → Permite selecionar ou desmarcar um ou mais itens pressionando a tecla **Ctrl + Botão esquerdo do mouse**.

**MultiExtended** → Permite selecionar um ou mais itens através de um intervalo. Temos que selecionar o primeiro item, em seguida segurar pressionada tecla **Shift (↑) + Botão esquerdo do mouse** ou a seta para baixo do teclado.

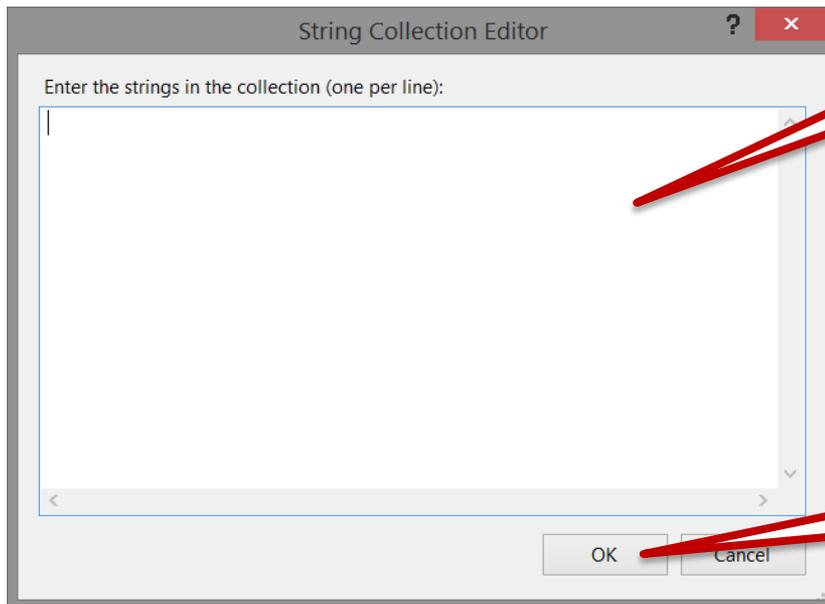
# ListBox

Podemos adicionar itens (opções) de 2 (duas) formas, através da propriedade **Items** ou pela programação.

## Adicionando itens através da propriedade Items



Clicando no botão  aparecerá a seguinte janela:



Neste espaço devemos digitar o itens que serão listados no componente.

Ao terminar clicar no botão **OK**.

# ListBox

Podemos adicionar itens (opções) de 2 (duas) formas, através da propriedade **Items** ou pela programação.

## Adicionando itens através da programação

```
ListBox1.Items.Add(<item>);
```

<item> → Texto no qual será adicionado ao componente.

Nome (**Propriedade Name**) do componente.

Método para adicionar um item dentro do componente.

## Exemplo:

```
listBox1.Items.Add("Casa");  
listBox1.Items.Add("Prédio");  
listBox1.Items.Add("Apartamento");  
listBox1.Items.Add("Flat");
```

# ListBox

Cada item adicionado possui um índice (oculto) associado a ele. O índice sempre começará com **0 (Zero)**.



O índice **-1** serve para desmarcar qualquer item selecionado, ou validar se nenhum item foi selecionado.

## Removendo item através da programação

```
listBox1.Items.RemoveAt(<índice>);
```

Método para remover um item do componente.

**<índice>** → Índice do item que será eliminado.

# ListBox

Podemos manipular os itens (opções) através das seguintes propriedades:

- ✓ **listBox1.SelectedIndex** → Retorna o índice do item que está selecionado.
- ✓ **listBox1.SelectedItem** → Retorna a descrição do item que está selecionado.
- ✓ **listBox1.Items.Count** → Retorna a quantidade total de itens do componente.

Caso precise ordenar os itens em ordem alfabética, utilize a propriedade **Sorted**. Por default o seu valor é **false (falso)**, caso alterar para **true (verdadeiro)**, todos os itens ficarão em ordem.

O evento default para o ListBox é o **SelectedIndexChanged**, ou seja, quando selecionamos um item, será executado o código de programação que estiver dentro.

```
private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
{

}
```

# Exemplo – ListBox

The image shows a screenshot of a Windows application window titled "ListBox". The window has a standard Windows title bar with minimize, maximize, and close buttons. Inside the window, there is a text input field labeled "Digite um texto:". Below this is a large, empty list box. At the bottom left, there are three text input fields labeled "Posição na lista:", "Texto selecionado:", and "Total de Itens:". On the right side, there are five buttons: "Adicionar", "Eliminar", "Limpar todos os itens", "Limpar", and "Sair".

# Exemplo – ListBox

The screenshot shows a window titled "ListBox" with the following elements:

- A text input field labeled "Digite um texto:".
- A list box containing several items.
- A text input field labeled "Índice na lista:".
- A text input field labeled "Texto selecionado:".
- A text input field labeled "Total de Itens:".
- A vertical stack of buttons: "Adicionar", "Eliminar", "Limpar todos os itens", "Limpar", and "Sair".

Digitar um texto para adicionar no ListBox.

Exibir o índice do item selecionado.

Exibir a quantidade total de itens do ListBox.

Índice na lista:

Texto selecionado:

Total de Itens:

Serve para eliminar o item selecionado do ListBox.

Limpar todos os itens do ListBox.

Exibir a descrição do item selecionado.

Limpar a caixa de texto e o item selecionado no ListBox.

Serve para adicionar o texto digitado no ListBox.

Adicionar

Eliminar

Limpar todos os itens

Limpar

Sair

# Exemplo – Código Fonte

Evento que executa quando pressionar um botão.

```
private void btnAdicionar_Click(object sender, EventArgs e)
{
    lstLista.Items.Add(txtTexto.Text);
    txtTexto.Clear();
    txtTexto.Focus();
}
```

Através do método **Add**, iremos adicionar o texto digitado na caixa de texto (txtTexto), para o ListBox (lstLista).

Este evento serve para colocar o cursor (focar) dentro da caixa de texto (txtTexto).

Este método faz limpar o conteúdo de uma caixa de texto. (**O Clear() não funciona para Label**)

# Exemplo – Código Fonte

```
private void btnEliminar_Click(object sender, EventArgs e)
{
    int posAnterior;
    if(lstLista.SelectedIndex == -1)
    {
        MessageBox.Show("Nenhuma opção foi selecionada!!!", "ListBox", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    else
    {
        posAnterior = lstLista.SelectedIndex - 1;
        lstLista.Items.RemoveAt(lstLista.SelectedIndex);
        lstLista.SelectedIndex = posAnterior;
    }
}
```

Verificar se não foi selecionado nenhum item no ComboBox, caso não foi selecionado, exibir uma mensagem.

Através do método **RemoveAt** iremos remover o item selecionado do ComboBox (cboListaDropDownList), através do índice do item selecionado.

```
private void btnLimparItens_Click(object sender, EventArgs e)
{
    lstLista.Items.Clear();
}
```

Remover (Limpar) todos os itens do ListBox.

# Exemplo – Código Fonte

```
private void btnLimpar_Click(object sender, EventArgs e)
{
    1 txtTexto.Clear();
    2 lstLista.Items.Clear();
    3 lblPosLista.Text = "";
    4 lblTextoSel.Text = "";
    5 lblTotal.Text = "";
    6 txtTexto.Focus();
}
```

1. Limpar o conteúdo da caixa de texto
2. Desmarcar o item selecionado no ListBox (lstLista)
3. Limpar o conteúdo do label (lblPosLista)
4. Limpar o conteúdo do label (lblTextoSel)
5. Limpar o conteúdo do label (lblTotal)
6. Colocar o cursor (foco) na caixa de texto (txtTexto)

# Exemplo – Código Fonte

```
private void lstLista_SelectedIndexChanged(object sender, EventArgs e)
{
    if (lstLista.SelectedIndex != -1)
    {
        lblPosLista.Text = lstLista.SelectedIndex.ToString();
        lblTextoSel.Text = lstLista.SelectedItem.ToString();
        lblTotal.Text = lstLista.Items.Count.ToString();
    }
}
```

Verifica se foi selecionado um item na ListBox.

Este método (**ToString()**) serve para converter um número em texto (string).

```
private void btnSair_Click(object sender, EventArgs e)
{
    Application.Exit();
}
```

Sair da aplicação.

# Exemplo – ListBox

The image shows a screenshot of a Windows application window titled "ListBox". The window contains a text input field with the label "Digite um texto:". A red callout box with a pointer pointing to the text input field contains the text: "Dê um duplo clique neste **textBox**, para programarmos o evento deste componente." Below the text input field is a large empty rectangular area, likely a list box. At the bottom left of the window, there are three text input fields with labels: "Posição na lista:", "Texto selecionado:", and "Total de Itens:". On the right side of the window, there are five buttons stacked vertically: "Adicionar", "Eliminar", "Limpar todos os itens", "Limpar", and "Sair".

# Exemplo – Código Fonte

Propriedade do evento **KeyPress** que armazena a tecla pressionada.

Evento que permite verificar a tecla pressionada.

```
private void txtTexto_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == 13)
    {
        btnAdicionar_Click(sender, e);
    }
}
```

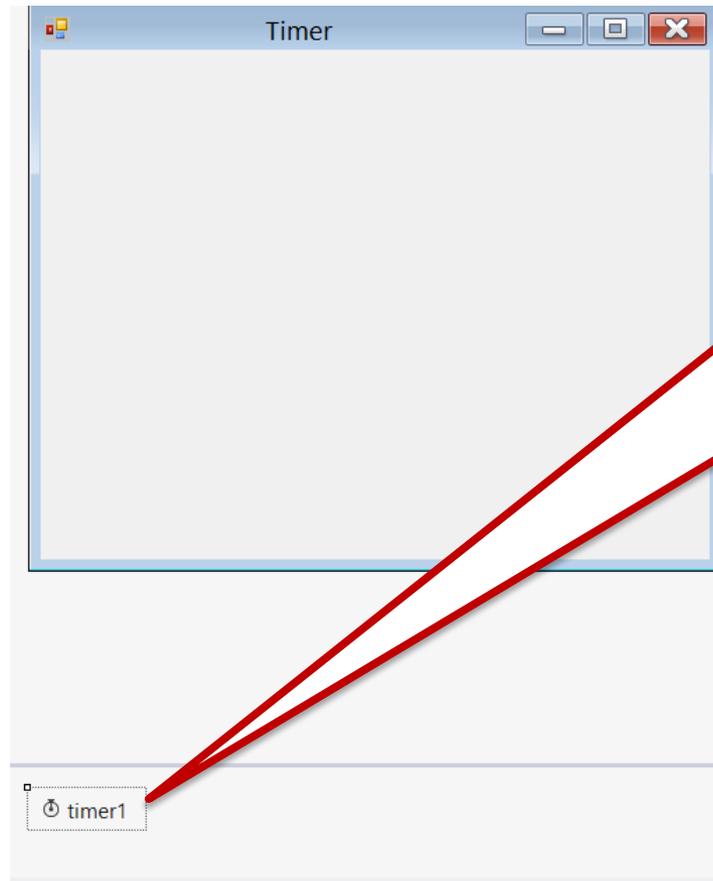
O valor 13 equivale a tecla **ENTER** do teclado.

Quando a tecla **ENTER** for pressionada, será executado o evento **Click** do botão **Adicionar**, passando como parâmetro os mesmos do componente atual. (Neste caso estamos **redirecionando ao mesmo código do botão**)

# Timer

## Timer

A função do **timer**, é esperar um determinado tempo para realizar alguma instrução. Funciona igual a um relógio.



**Este componente não é visual, ou seja, faz parte do projeto mas não tem aparência gráfica. Esse tipo de componente sempre aparecerá na parte inferior do projeto.**

# Timer

O tempo é medido em **ms (milissegundos)** através da propriedade **interval**. Cada 1000ms equivale a 1s (segundo).

Interval	1000
----------	------

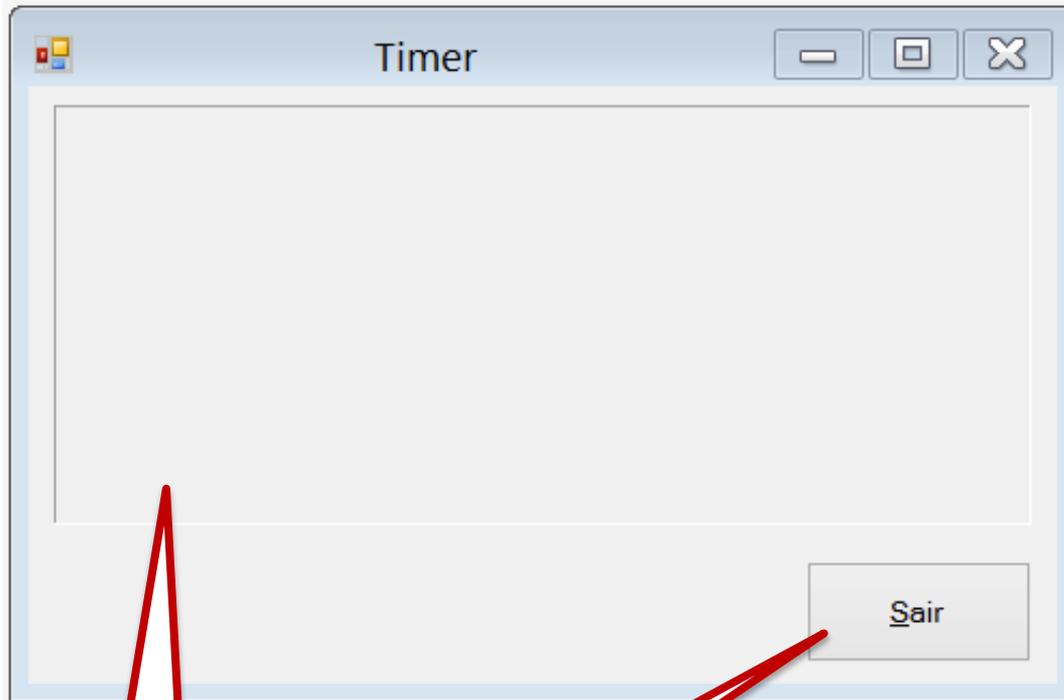
Para ativar (**ligar**) ou desativar (**desligar**) o Timer utilizaremos a propriedade **Enabled**, onde o valor **true** indica que o timer está **ligado**, e o valor **false** indica que o timer está **desligado**.

O evento default (Padrão) para o Timer é o **Tick**, ou seja, enquanto o timer estiver **true ligado**, será executado o código de programação que estiver dentro.

```
private void timer1_Tick(object sender, EventArgs e)
{

}
```

# Exemplo - Timer



Properties

tmrTempo System.Windows.Forms.Timer

⊞ (ApplicationSettings)	
(Name)	tmrTempo
Enabled	True
GenerateMember	True
Interval	1
Modifiers	Private
Tag	

Para este exemplo foram alteradas as seguintes propriedades: **Name, Enabled e Interval**

# Exemplo – Código Fonte

Dê um clique duplo no componente **Timer (tmrTempo)**, em seguida digite o código como mostra a imagem abaixo:

```
private void tmrTempo_Tick(object sender, EventArgs e)
{
    lblHora.Text = DateTime.Now.ToString("HH:mm:ss");
}
```

Como definimos o **interval** do timer em **1 ms**, então enquanto o Timer estiver **ligado (true)**, o código acima sempre será executado a cada **1ms**.

Abaixo o código do botão Sair:

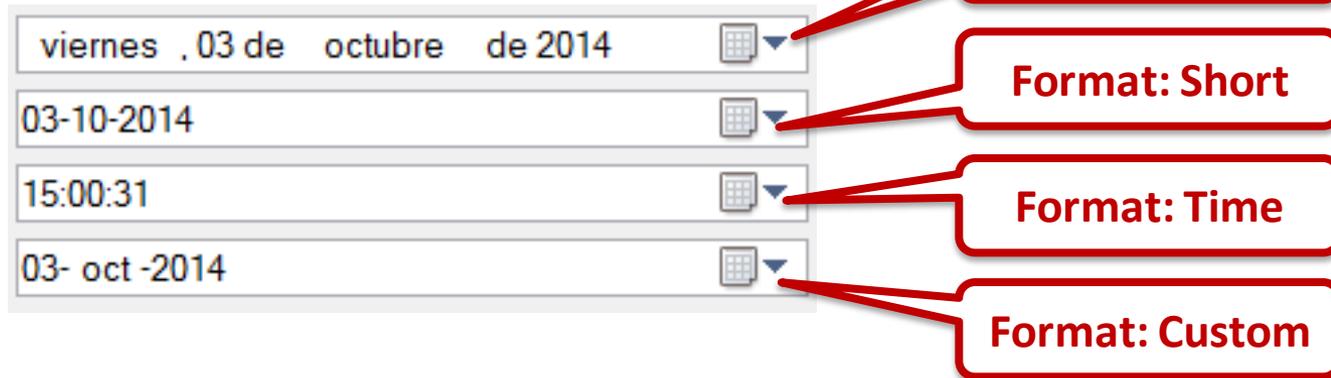
```
private void btnSair_Click(object sender, EventArgs e)
{
    Application.Exit();
}
```

# DateTimePicker



## DateTimePicker

A função do **datetimepicker**, é disponibilizar um calendário dentro do formulário. Podemos optar por 4 tipos de formatos através da propriedade **Format** do componente, como mostra as figuras abaixo:



The image shows four examples of the DateTimePicker component with different formats. Red callout boxes point to each format:

- Format: Long**: viernes .03 de octubre de 2014
- Format: Short**: 03-10-2014
- Format: Time**: 15:00:31
- Format: Custom**: 03- oct -2014

Para o formato do tipo **Custom**, é necessário digitar o formato desejado na propriedade **CustomFormat**.

CustomFormat

dd-MMM-yyyy

# DateTimePicker

Abaixo segue os tipos de dados para Data:

<b>d</b>	→	O dia de um ou dois dígitos	→	"3"
<b>dd</b>	→	O dia de dois dígitos. Valores de dígito único dia são precedidos por 0	→	"03"
<b>ddd</b>	→	Abreviação do dia da semana de três caractere	→	"Ter"
<b>dddd</b>	→	Nome completo do dia da semana	→	"Terça"
<b>M</b>	→	O número do mês de um ou dois dígitos	→	"6"
<b>MM</b>	→	O número do mês de dois dígitos. Valores de dígito único são precedidos por 0	→	"06"
<b>MMM</b>	→	Abreviação de três caractere mês	→	"Jun"
<b>MMMM</b>	→	Nome completo do mês	→	"Junho"
<b>y</b>	→	O ano de um dígitos (2001 é exibido sistema autônomo "1").	→	"1"
<b>yy</b>	→	Dois últimos dígitos do ano (2001 é exibido sistema autônomo "01").	→	"01"
<b>yyy</b>	→	O ano inteiro (2001 é exibido sistema autônomo "2001").	→	"2001"

# DateTimePicker

Abaixo segue os tipos de dados para Hora:

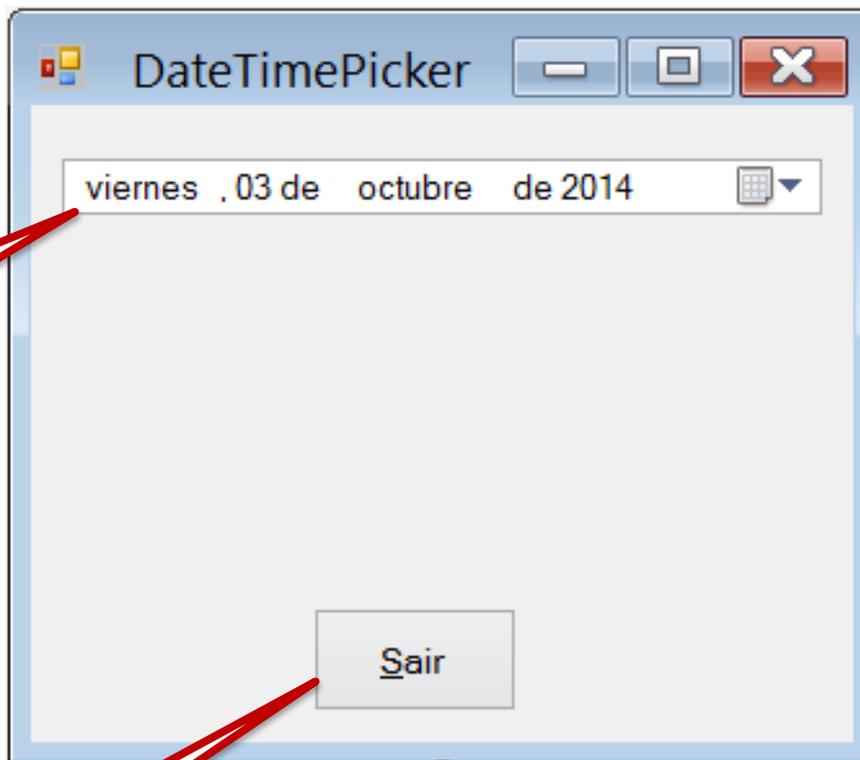
h	→	A hora de um ou dois dígitos no formato de 12 horas	→	"1"
hh	→	A hora de dois dígitos no formato de 12 horas. Dígitos únicos são precedidos por 0.	→	"01"
H	→	A hora de um ou dois dígitos no formato de 24 horas.	→	"3"
HH	→	A hora de dois dígitos no formato de 24 horas. Dígitos únicos são precedidos por 0.	→	"03"
m	→	O minuto de um ou dois dígitos	→	"6"
mm	→	O minuto de dois dígitos. Valores de dígitos únicos são precedidos por 0	→	"06"
s	→	Os segundos de um ou dois dígitos.	→	"1"
SS	→	Os segundos de dois dígitos. Valores de dígitos únicos são precedidos por 0	→	"01"
t	→	AM/PM a uma letra.abreviação (A.M.é exibido sistema autônomo "A")	→	"A"
T	→	AM/PM de duas letras.abreviação (A.M.é exibida sistema autônomo "AM").	→	"AM"

# DateTimePicker

O evento default (Padrão) para o DateTimePicker é o **ValueChanged**, ou seja, quando selecionamos uma data no componente, será executado o código de programação que estiver dentro.

```
private void dateTimePicker1_ValueChanged(object sender, EventArgs e)
{
}
}
```

# Exemplo - DateTimePicker



DateTimePicker

Button

# Exemplo – Código Fonte

Dê um clique duplo no componente **DateTimePicker (dtpCalendario)**, em seguida digite o código como mostra a imagem abaixo:

```
private void dtpCalendario_ValueChanged(object sender, EventArgs e)
{
    MessageBox.Show(dtpCalendario.Value.ToString("dd/MM/yyyy"));
}
```

**Ao selecionar a Data no componente  
exibirá uma janela com a data  
selecionada no formato "dd/MM/yyyy"**

Abaixo o código do botão Sair:

```
private void btnSair_Click(object sender, EventArgs e)
{
    Application.Exit();
}
```

# Referências

[http://msdn.microsoft.com/pt-br/library/system.windows.forms.listbox\(v=vs.110\).aspx](http://msdn.microsoft.com/pt-br/library/system.windows.forms.listbox(v=vs.110).aspx)

[http://msdn.microsoft.com/pt-br/library/system.windows.forms.combobox\(v=vs.110\).aspx](http://msdn.microsoft.com/pt-br/library/system.windows.forms.combobox(v=vs.110).aspx)

[http://msdn.microsoft.com/pt-br/library/System.Windows.Forms.Timer\(v=vs.110\).aspx](http://msdn.microsoft.com/pt-br/library/System.Windows.Forms.Timer(v=vs.110).aspx)

[http://msdn.microsoft.com/pt-br/library/system.windows.forms.timer\\_methods\(v=vs.110\).aspx](http://msdn.microsoft.com/pt-br/library/system.windows.forms.timer_methods(v=vs.110).aspx)

[http://msdn.microsoft.com/pt-br/library/System.Windows.Forms.DateTimePicker\(v=vs.90\).aspx](http://msdn.microsoft.com/pt-br/library/System.Windows.Forms.DateTimePicker(v=vs.90).aspx)

[http://msdn.microsoft.com/pt-br/library/system.windows.forms.datetimepicker.customformat\(v=vs.90\).aspx](http://msdn.microsoft.com/pt-br/library/system.windows.forms.datetimepicker.customformat(v=vs.90).aspx)