(Parte VII)

Esta parte do documento C# Essencial, dá início ao estudo da biblioteca *Windows Forms* a qual permite desenvolver aplicações Windows. A biblioteca está definida no *namespace System.Windows.Forms.*

Sumário:

Figuras

Manipulação de Forms

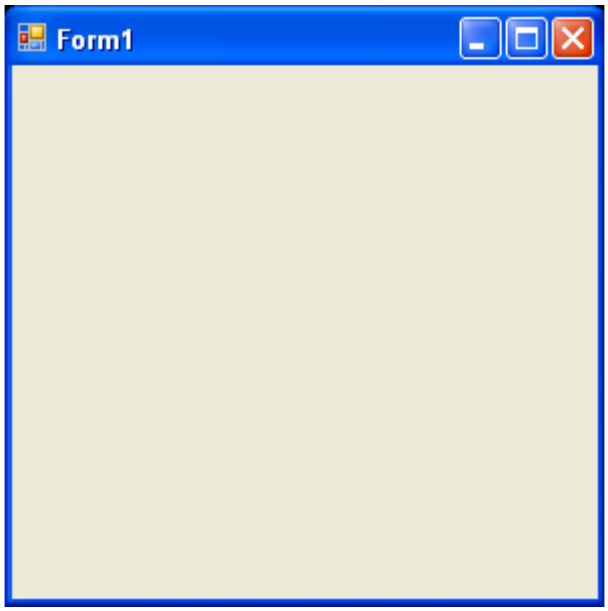
Através da biblioteca *Windows Forms* podemos desenvolver vários tipos de interfaces Windows, por exemplo:

1. MDI (Multiple Document Interface): Aplicação que suporta múltiplos documentos abertos simultaneamente (ex: Word)
2. SDI (Single Document Interface): Aplicação que permite a abertura de apenas um documento de cada vez (ex: NotePad)
3. Janelas modais: Janelas informativas conhecidas como diálogos.

Interface MDI

Definição e inserção de um menu de opções

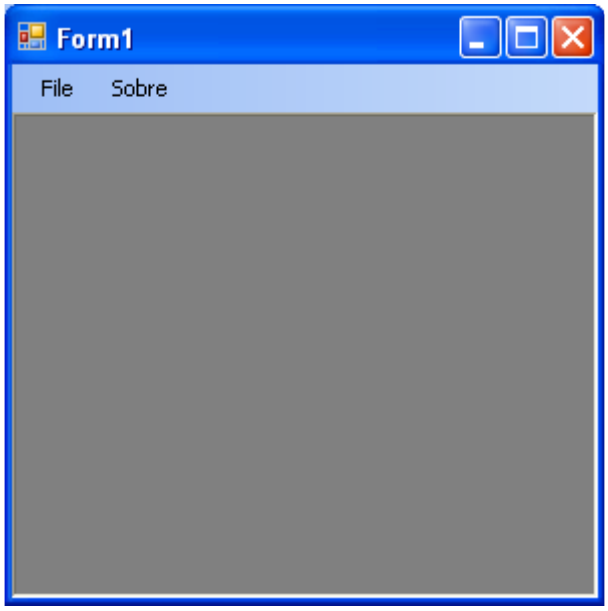
De modo a criar uma aplicação Windows com a plataforma Microsoft Visual Studio .NET, é necessário criar um novo projecto com as opções *File → New Project,* seleccionar o tipo de projecto *Visual C# Projects*, e finalmente o template *Windows Application*. Por defeito é gerado automaticamente um formulário do tipo SDI com nome *Form1*.



Drawing 1: Formulário SDI

Para proceder a definição de uma interface MDI devemos alterar a propriedade do formulário principal *IsMdiContainer* para *true*.

A inserção de um menu na janela principal passa por activar a *toolbox* e de seguida arrastar para a janela principal o componente *MenuStrip.* Finalmente procede-se a definição e inserção dos respectivos itens de menu.



Drawing 2: Formulário MDI

Para cada item de menu é possível associar código ao evento click do MenuItem, por exemplo:

```
private void menuItem1_Click(object sender, System.EventArgs e)
{
  MessageBox.Show("Olá Mundo!");
}
```

Este código referencia o método estático *Show* da classe *MessageBox* com o objectivo de mostrar uma mensagem ao utilizador da aplicação. A instrução permite abrir uma nova janela com o texto passado por parâmetro e um botão *Ok* para fechar.

```
private void menuItem4_Click(object sender,  System.EventArgs e)
{
  Application.Exit( );
}
```

Neste caso, faz-se uso da classe *Application* e o seu método estático *Exit( )* para fechar a aplicação.

Existem ainda muitas propriedades que podem ser configuradas para cada formulário. Como por exemplo a propriedades *ShowInTaskBar* e *WindowState*. A primeira faz com que a aplicação seja mostrada, ou não, na barra de tarefas. A segunda permite-nos definir o estado inicial do formulário: *Minimized*, *Normal* ou *Maximized*.

Para adicionar um novo formulário (do tipo SDI) à aplicação devemos executar as seguintes opções do menu *Project → Add New Item,* seleccionar *Windows Forms* e adicionar logo a seguir o respectivo menu principal (componente *MenuStrip*) e os seus itens de menu. Por defeito o formulário tem nome *Form2.*

Neste momento o projecto é composto por um formulário (*Form1*) do tipo MDI e outro (*Form2*) do tipo SDI.

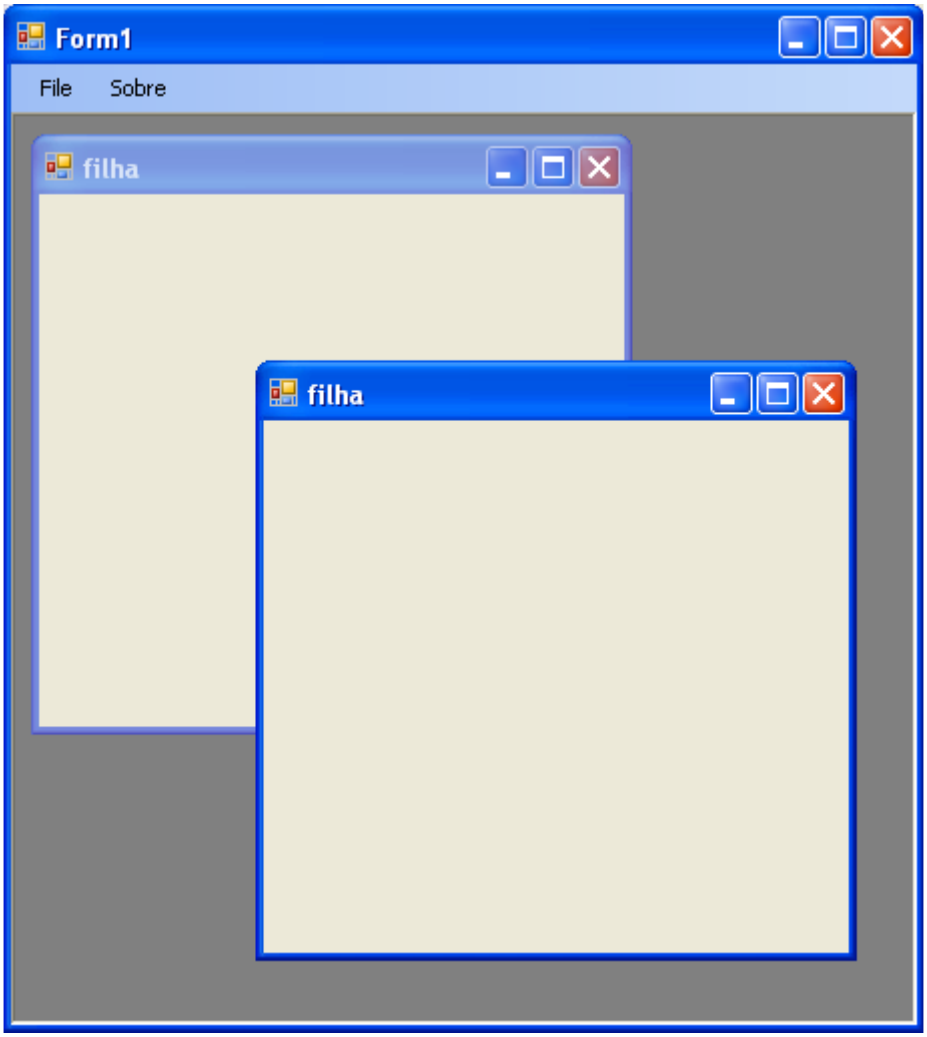
Adicionar uma nova janela filha

Vamos agora implementar uma opção no menu da janela pai (formulário *Form1*). A execução da opção deverá inserir uma nova janela filha. O código deverá ser o seguinte:

```
Form2  form = new Form2( );
form.MdiParent = this;

form.Text = "Janela Filha";
form.Show( );
```

O primeiro passo consiste em criar uma nova instância do formulário que constitui a nossa janela filha (neste caso *Form2* do tipo SDI).



Drawing 3: Formulário MDI com dois formulários SDI

A seguir, definimos o pai da janela filha alterando a propriedade *MdiParent* do formulário como sendo o formulário principal. A atribuição do valor *this* a essa propriedade permite indicar que a janela pai é o objecto correspondente ao formulário principal. Para terminar devemos ainda mostrar a janela filha invocando o método *Show( )*.

Fechar a janela filha activa

Para fechar a janela filha activa é necessário definir uma nova opção no menu da janela pai, por exemplo a opção *"Fechar janela activa"*. A opção deverá estar associada ao código seguinte:

```
this.ActiveMdiChild.Close( );
```

A propriedade *ActiveMdiChild* é um objecto do tipo *Form*, mais precisamente o objecto da janela filha activa nesse instante. A execução do método *Close()* permite encerrar de imediato a janela activa.

Organização das janelas filhas

As janelas filhas podem ser organizadas em cascata, lado a lado na horizontal e na vertical. Para o efeito basta executar na janela pai respectivamente o código seguinte:

```
this.LayoutMdi(System.Windows.Forms.MdiLayout.Cascade); // em cascata

this.LayoutMdi(System.Windows.Forms.MdiLayout.TileHorizontal); // na horizontal

this.LayoutMdi(System.Windows.Forms.MdiLayout.TileVertical);  // na vertical
```

Utilização do componente TabControl

O componente *TabControl* permite agrupar vários formulários num só. A sua inserção é realizada através da *ToolBox* arrastando para a janela o componente *TabControl*. A seguir clicando no botão direito do rato temos acesso as opções *Add Tab* e *Remove Tab* as quais permitem adicionar ou remover páginas de um TabControl. Para ter acesso às diversas páginas de um *TabControl*, basta clicar na propriedade *TabPages* de modo a visualizar uma janela com as páginas criadas.

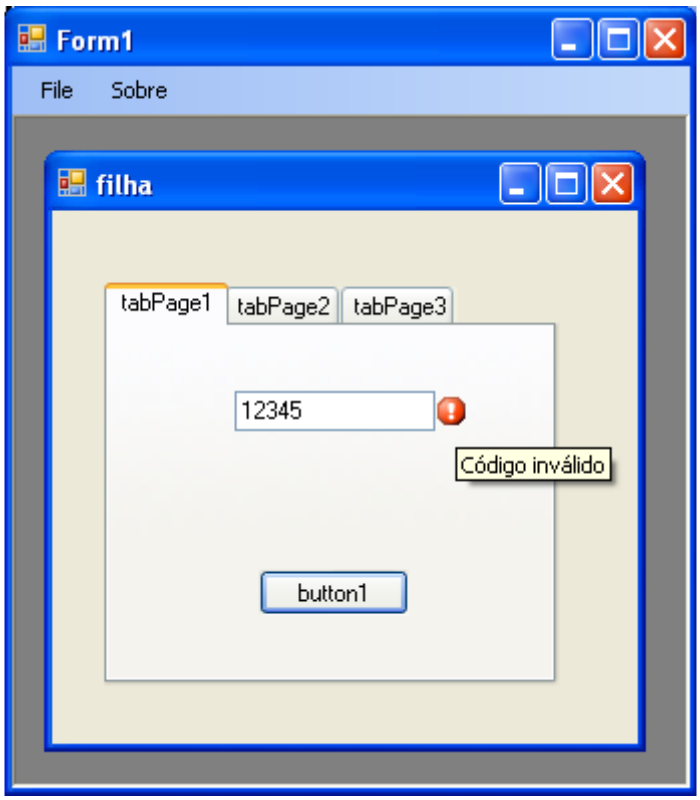
Drawing 4: Componente TabControl

O componente *ErrorProvider*

Para validar os dados inseridos nos componentes pelo utilizador podemos utilizar o componente *ErrorProvider*.
Por exemplo para validar o contéudo de uma *TextBox* (TextBox tb = new TextBox();) utilizando o componente *ErrorProvider* devemos primeiro alterar a propriedade *CausesValidation* da TextBox para *true*.

```
if (tb.Text.Length < 6)   errorProvider.SetError(tb, "Código inválido");
```

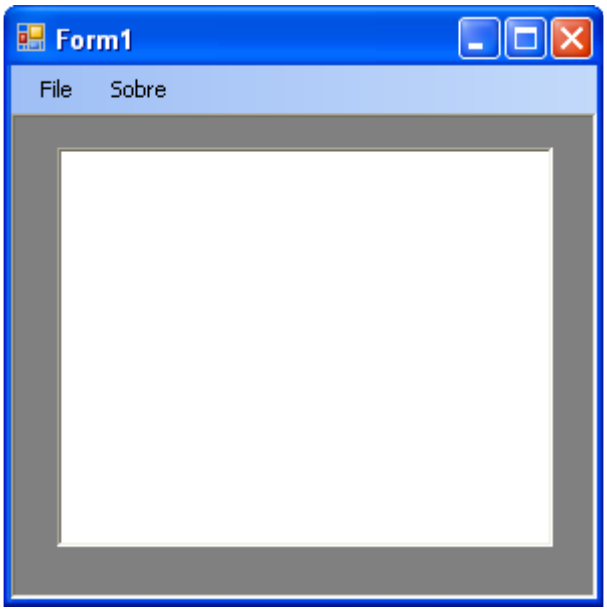
Este código permite lançar a mensagem *"Código inválido"* sempre que o conteúdo inserido na *TextBox* tenha menos de 6 caracteres.



Drawing 5: Componente ErrorProvider

Desenvolvimento de um editor de texto

No desenvolvimento de um editor de texto podemos aplicar o componente *RichTextBox*. Este componente permite editar, copiar, colar, localizar e formatar texto.



Drawing 6: Componente RichTextBox

```
RichTextBox richEditor = new RichTextBox();
```

O método para copiar o texto seleccionado
:

```
private void copiar_Click(object sender, System.EventArgs e)

{   if (richEditor.SelectedText != "")    richEditor.Copy( );

}
```

O método para colar o texto seleccionado:

```
private void colar_Click(object sender, System.EventArgs e)
{     richEditor.Paste( );
}
```

Para ler e guardar um ficheiro no sistema podemos executar os componentes OpenFileDialog e SaveFileDialog:

```
private void abrir_Click(object sender, System.EventArgs e)
{ of = new OpenFileDialog( );
  of.ShowDialog( );
  ...
  if (of.ShowDialog( ) == DialogResult.OK)
          this.richEditor.LoadFile(of.FileName, RichTextBoxStreamType.RichText);

}
```

Para determinar qual foi o botão seleccionado pelo utilizador no diálogo devemos utilizar a lista *DialogResult:*

Ok: O botão clicado no diálogo foi *OK*

Cancel: O botão clicado no diálogo foi *Cancel*

Yes: O botão clicado no diálogo foi *Yes*

No: O botão clicado no diálogo foi *No*

Ignore: O botão clicado no diálogo foi *Ignore*

Retry: O botão clicado no diálogo foi *Retry*

None: Nenhum botão foi clicado

Abort: O botão clicado no diálogo foi Abort

Para abrir o ficheiro seleccionado:

```
this.richEditor.LoadFile(of.FileName,
System.Windows.Forms.RichTextBoxStreamType.RichText);
```

Para guardar um ficheiro:

```
sf = new SaveFileDialog( );
if (sf.ShowDialog( ) == DialogResult.OK)
      this.richEditor.SaveFile(sf.FileName, RichTextBoxStreamType.RichText);
```

Para imprimir o ficheiro devemos utilizar os componentes *PrintPreviewDialog, PrintDialog* e *PrintDocument.*

O componente PrintPreviewDialog deve ser invocado para pre-visualizar um documento:

```
private void printPreview_Click(object sender,
 System.EventArgs e)

{    stringR = new StringReader(this.richEditor.Text);
     PrintPreviewDialog printPreviewDialog1 = new PrintPreviewDialog( );
     printPreviewDialog1.Document = this.printDocument1 ;
     printPreviewDialog1.ShowDialog( );

}
```

O componente PrintDialog deve ser invocado para mostrar o diálogo para o envio de um documento para a impressora:

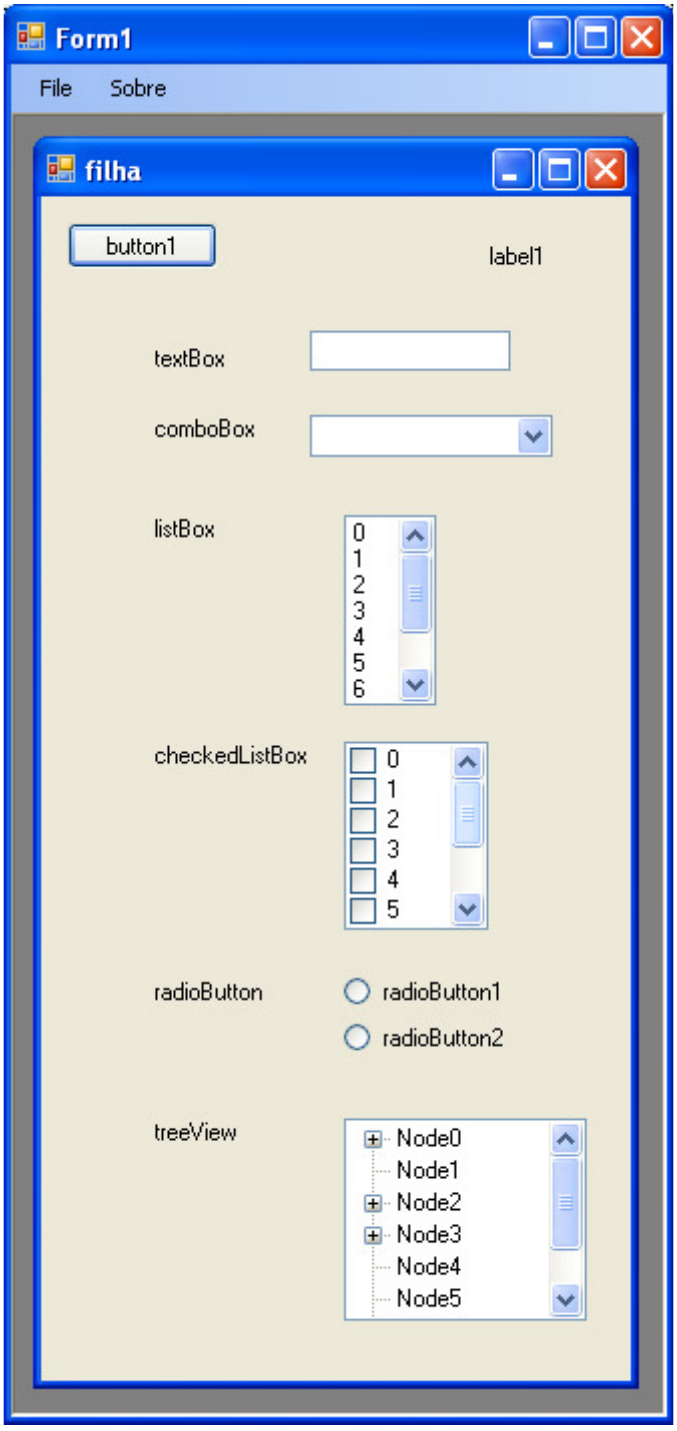
```
private void imprimir_Click(object sender,
 System.EventArgs e)

{   stringR = new StringReader(this.richEditor.Text);
     printDialog1.Document = this.printDocument1;
     if (printDialog1.ShowDialog( ) == DialogResult.OK)
          this.printDocument1.Print( );
}
```

Os componentes indispensáveis

Crie um novo projecto no visual studio com template Windows Application, e manipule os componentes seguintes. Para cada componente consulte todas as propriedades disponíveis:

- Button
- Label
- TextBox
- ComboBox
- ListBox
- Checked ListBox
- Radio Button
- TreeView
- ListView
- PictureBox
- DateTimePicker
- MonthCalendar



Drawing 7: Componentes Button, LabelTextBox, ComboBoxListBox, Checked ListBox, Radio Button e TreeView

Drawing 8: Componentes PictureBoxDate, TimePicker e MonthCalendar

Exercícios

1. Pretende-se implementar uma aplicação Windows através da biblioteca *Windows Forms*. A aplicação deverá implementar uma agenda de contactos telefónicos.

   Cada contacto é definido com o *nome* (os nomes são únicos) e o número de *telemóvel*;

   As funcionalidades a disponibilizar na interface gráfica são:

   - número total de contactos
   - adicionar um novo contacto na agenda
   - remover um contacto dado o nome
   - determinar a existência de um contacto dado o nome
   - obter o número de telemóvel dado o nome do contacto
   - listar os nomes de todos os contactos por ordem crescente
   - guardar em ficheiro
   - ler o ficheiro de dados
   - Pre-visualizar e imprimir

   Define uma interface MDI com múltiplas janelas filha. Organize as janelas lado a lado e define um ícone na barra *Tray*. Deverá utilizar os componentes *TabControl*, *ComboBox*, *Check ListBox, Radio Button, TreeView* e *ListView.*

2. Define uma aplicação Windows para gestão de um stand automóvel. Para cada automóvel o sistema deverá armazenar os dados seguinte:

   - matrícula
   - marca
   - modelo
   - cilindrada
   - quilómetros
   - preço
   - estado (a venda ou vendido)

   Cada cliente do stand deverá ficar armazenado no sistema com a informação seguinte:

   - número de bilhete de identidade
   - nome
   - morada

- lista das matrículas dos veículos comprados no stand

Define uma interface MDI com múltiplas janelas filha. Organize as janelas lado a lado e define um ícone na barra *Tray.* Deverá utilizar os componentes *TabControl*, *ComboBox*, *Check ListBox, Radio Button, TreeView* e *ListView.*

Referências

- http://www.softsteel.co.uk/tutorials/cSharp/
- http://www.csharp-station.com/Tutorial.aspx
- http://csharpcomputing.com/Tutorials/TOC.htm
- http://msdn2.microsoft.com/en-us/library/9b9dty7d(VS.80).aspx
- http://msdn2.microsoft.com/en-us/library/2s05feca(VS.80).aspx
- C# School – Programmers Heaven (cf. Site da Disciplina)

continua

lufer, jcsilva, ajtavares, marco