

C# Avançado

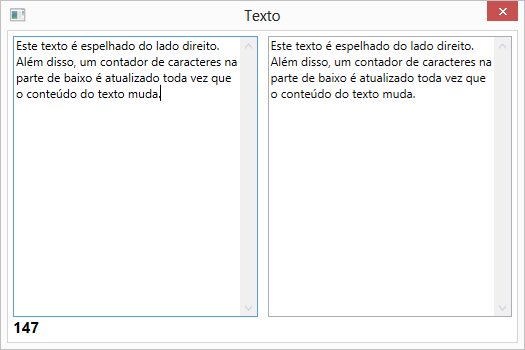
Exercícios Propostos

WPF Parte 2: Manipulando Dados

www.softblue.com.br

# Exercício

Crie uma aplicação WPF com uma janela semelhante à da figura abaixo:

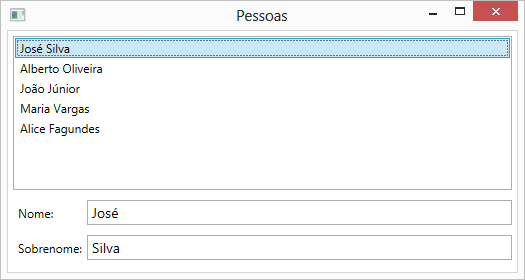


A janela é composta por duas caixas de texto e um contador na parte inferior. Todo texto digitado na caixa de texto da esquerda é replicado para o lado direito automaticamente (a caixa de texto do lado direito não pode ser editada). Além disso, o contador na parte inferior da janela é atualizado toda vez que o texto muda, e mostra a quantidade de caracteres que compõem o texto digitado.

A atualização da caixa de texto da direita e também do contador devem ser feitas através do uso de bindings.

# Exercício

Crie uma aplicação WPF com uma janela semelhante à mostrada na figura abaixo:



A janela deve exibir uma lista (ListBox) de nomes de pessoas (você pode criar uma lista manualmente no código, com pessoas quaisquer). A lista exibe o nome completo de cada pessoa. Na parte inferior da janela, dois controles TextBox exibem e permitem a edição do nome e do sobrenome da pessoa. Quando o nome ou sobrenome da pessoa é editado, o nome completo é atualizado na lista imediatamente, sem a necessidade de tirar o foco do controle.

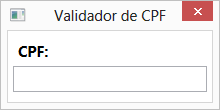
A classe que representa uma pessoa deverá se chamar Pessoa e deverá ter três properties:

* Nome: nome da pessoa.
* Sobrenome: sobrenome da pessoa.
* NomeCompleto: concatenação das properties Nome e Sobrenome, com um espaço em branco separando o valor de ambas.

**Dica**: Para que a lista saiba que o nome completo da pessoa foi alterado, será necessário fazer com que a classe Pessoa implemente a interface INotifyPropertyChanged. Desta forma o objeto pode avisar o WPF a respeito da alteração, e o WPF pode atualizar o conteúdo exibido na lista.

# Exercício

Crie uma aplicação onde o usuário preenche um número de CPF e o WPF verifica se o CPF é válido, já durante a digitação. A tela pode ter o seguinte layout:



Para validar o CPF, utilize um validador customizado. O algoritmo utilizado para validação do CPF pode ser feito desta forma (o algoritmo original foi extraído de <http://www.devmedia.com.br/validacao-de-cpf-e-cnpj/3950>):

public static bool ValidarCpf(string cpf)

{

string valor = cpf.Replace(".", "");

valor = valor.Replace("-", "");

if (valor.Length != 11)

{

return false;

}

bool igual = true;

for (int i = 1; i < 11 && igual; i++)

{

if (valor[i] != valor[0])

{

igual = false;

}

}

if (igual || valor == "12345678909")

{

return false;

}

int[] numeros = new int[11];

for (int i = 0; i < 11; i++)

{

try

{

numeros[i] = int.Parse(valor[i].ToString());

}

catch (FormatException)

{

return false;

}

}

int soma = 0;

for (int i = 0; i < 9; i++)

{

soma += (10 - i) \* numeros[i];

}

int resultado = soma % 11;

if (resultado == 1 || resultado == 0)

{

if (numeros[9] != 0)

{

return false;

}

}

else if (numeros[9] != 11 - resultado)

{

return false;

}

soma = 0;

for (int i = 0; i < 10; i++)

{

soma += (11 - i) \* numeros[i];

}

resultado = soma % 11;

if (resultado == 1 || resultado == 0)

{

if (numeros[10] != 0)

{

return false;

}

}

else if (numeros[10] != 11 - resultado)

{

return false;

}

return true;

}