Trabalhando com menus e caixas de diálogos

Neste laboratório, você vai aprender:

- Criar menus para aplicativos Microsoft Windows Presentation Fundation (WPF) utilizando as classes Menu e MenuItem.
- Realizar o processamento em resposta a eventos de menu quando um usuário clica em um controle de menu.
- Manipular menus por meio de código e criar menus dinâmicos.
- Utilizar caixas de diálogos comuns do Windows em um aplicativo para solicitar ao usuário o nome de um arquivo.

Menus e eventos de menu

O WPF fornece o controle Menu como um contêiner pata itens de menu. Esse controle fornece um Shell básico para definir um menu e, como ocorre com a maioria dos aplicativos WPF, ele é muito flexível, e você pode definir uma estrutura de menus consistindo em praticamente qualquer tipo de controle WPF. Os menus WPF podem conter botões, caixa de texto, caixas de combinação etc.

Criando um menu

Vamos criar um menu apenas para servir de exemplo.

1. Crie uma aplicação WPF e forneça um nome para ela.
2. Na Toolbox, arraste um controle DockPanel a partir da seção Controls para qualquer lugar no formulário. Na janela Properties, configure a propriedade Width do DockPanel para **Auto**, configure a propriedade HorizontalAlighment com **Strech**, configure a propriedade VerticalAlignment como **Top** e configure a propriedade Margin como **0**.
3. Na Toolbox, arraste um controle Menu a partir da seção Controls até o controle DockPanel. Na janela Properties, configure a propriedade DockPanel.Dock como **Top**, configure a propriedade Width como **Auto**, configure a propriedade HorizontalAlignment como **Strech** e configure a propriedade VerticalAlignment como **Top**.

O controle Menu aparece como uma barra cinza ao longo da parte superior do DockPanel. Se examinar o código, ele deve se parecer como isto:

```
<DockPanel VerticalAlignment="Top">
   <Menu DockPanel.Dock="Top" VerticalAlignment="Top" Name="menu1">
   </Menu>
 </DockPanel>
```

A propriedade HorizontalAlignment não aparece no código XAML porque o valor “Stretch” é o valor padrão para essa propriedade.

4. No painel XAML, modifique a definição do controle Menu e adicione o elemento MenuItems como mostra em negrito o código a seguir. Observe que os elementos MenuItems aparecem como filhos do controle Menu.

```
   <DockPanel VerticalAlignment="Top" Height="100" Name="dockPanel1">
      <Menu DockPanel.Dock="Top" VerticalAlignment="Top" Name="menu1">
          <MenuItem Header="_File">
             <MenuItem Header="_New Member" Name="newMember"/>
             <MenuItem Header="_Save Member" Name="saveMember"/>
             <Separator />
             <MenuItem Header="E_xit" Name="exit"/>
          </MenuItem
      </Menu>
 </DockPanel>
```

O atributo Header do elemento MenuItem especifica o texto que aparece para o item de menu. O sublinhado (_) na frente de uma letra fornece acesso rápido a esse item de menu quando o usuário pressiona a tecla Alt e a letra depois do sublinhado (exe.: Alt + F para File ou Alt + H para Help).

5. Compile a aplicação e teste estas funcionalidades.
6. No XAML, modifique a definição do item do Menu _File e adicione os itens do menu secundário, como mostra o código a seguir:

```
<MenuItem Header="_File">
   <MenuItem Header="_New Member" Name="newMember"/>
   <MenuItem Header="_Save Member" Name="saveMember"/>
   <Separator />
   <MenuItem Header="E_xit" Name="exit"/>
</MenuItem>
```

7. Modifique a definição do item de menu _Help e adicione esse item de menu secundário como mostrado a seguir:

```
<MenuItem Header="_Help">
   <MenuItem Header="_About Middleshire Bell Ringers"
   Name="about"/>
</MenuItem>
```

8. Adicione imagens no formato BMP para o seu projeto. De preferência do tipo ícones. Modifique cada item de menu secundário para exibir um ícone, além do texto. Veja o código a seguir:

```
<MenuItem Header="_New Member" Name="newMember">
    <MenuItem.Icon>
        <Image Source="Filenew.bmp"/>
    </MenuItem.Icon>
</MenuItem>
```

9. Execute o projeto e veja o resultado.

Vimos que é possível especificar os itens de menu de primeiro nível como elementos MenuItems e então adicionar elementos MenuItems aninhados para definir a estrutura do seu menu. Os próprios elementos MenuItems aninhados podem conter outros elementos MenuItems aninhados se você quiser criar menus em cascata. Teoricamente, você pode prosseguir nesse processo até um nível muito além, mas na prática você provavelmente não deve passar de dois níveis de aninhamento.

Tratamento de eventos de menu

Você acabou de construir a aparência do menu. Para torná-lo funcional, você precisa escrever código que trate os vários eventos de menu. Diversos eventos podem ocorrer quando um usuário seleciona um item de menu, sendo alguns mais úteis que outros. O Click é o evento utilizado com mais freqüência , que ocorre quando o usuário clica no item de menu. Normalmente, você captura esse evento para executar as tarefas associadas ao item de menu.

No nosso exemplo, quando o usuário clicar no item de menu NewMember, uma nova janela para cadastro de membros será exibida. Portanto, adicione o código no evento clique desse item para abrir esta nova janela.

10. No Design XAML, adicione o evento Click, como mostra o código a seguir:

```
<MenuItem Header="_New Member" Name="newMember"
Click="newMember_Click">
     <MenuItem.Icon>
         <Image Source="Filenew.bmp"/>
     </MenuItem.Icon>
</MenuItem>
```

11. Navegue até o editor de código e inclua o código para abrir a janela CadMember.xaml. Antes, disso adicione uma nova janela Window.xaml em seu projeto.

```
private void newMember_Click(object sender, RoutedEventArgs e)
{
    CadMember cadMember = new CadMember();
    cadMember.Show();
}
```

12. Faça o mesmo com as outras opções do menu **File**, com exceção do item **Exit**.
13. Adicione o evento Click ao item Exit e em seguite, navegue até o código e insira o seguite comando:

```
private void exit_Click(object sender, RoutedEventArgs e)
{
    this.Close();
}
```

14. No meu Project, clique em Add Window.
15. Na caixa de diálogo que surge, selecione o item Window (WPF) e então clique em Add.
16. Na janela Properties de AboutBox, altere a propriedade Title para “Sobre o nosso exemplo”, configure a propriedade Width como 300 e Height como 156.
17. Adicione dois controles de rótulo e um controle de botão ao formulário. No painel XAML, modifique suas propriedades como mostrado a seguir:

```
<Grid>
   <Label Margin="80,20,0,0" Name="version" Height="30"
          VerticalAlignment="Top" HorizontalAlignment="Left"
          Width="75">Version 1.0</Label>
   <Label Margin="80,50,0,0" Name="buildDate" Height="30"
          VerticalAlignment="Top" HorizontalAlignment="Left"
          Width="160">Build date: setembro de 2011</Label>
   <Button Margin="100,85,0,0" Name="ok" HorizontalAlignment="Left"
           Width="78" Height="23" VerticalAlignment="Top">OK</Button>
</Grid>
```

18. Na janela Design View, dê um clique duplo no botão OK.
19. No manipulador do botão, digite o seguinte código:

```
private void ok_Click(object sender, RoutedEventArgs e)
{
     this.Close();
}
```

20. Retorne á janela Design para exibir o arquivo Window1.xaml. no painel XAML, localize a definição do item de menu About e utilize o comando <New Event Handler> para especificar um método de evento Click chamado about_Click. (Esse é o padrão gerado pelo comando <New Event Handler>).
21. Na janela Code and Text Editor que exibe o arquivo Window1.xaml.cs, adicione as seguintes instruções mostradas a seguir:

```
private void about_Click(object sender, RoutedEventArgs e)
{
     About aboutWindow = new About();
     aboutWindow.ShowDialog();
}
```

Caixas de diálogo

O nosso aplicativo permite que você salve as informações sobre os membros, mas ele sempre salva os dados no mesmo arquivo, sobrescrevendo tudo que já estava lá. Então vamos tratar esta questão.

Algumas tarefas cotidianas exigem que o usuário especifique algum tipo de informação. Por exemplo, se o usuário quiser abrir ou salvar um arquivo, normalmente se solicita ao usuário qual arquivo ou onde salvar. Para isso a Microsoft disponibilizou um componente fornecido pelo sistema operacional Microsoft Windows que você pode utilizar em seus próprios aplicativos. A biblioteca de classes do Microsoft .NET Framework fornece as classes **OpenFileDialog**, **SaveFileDialogs** que agem como empacotamento para essas caixas de diálogo comuns.

Utilizando a classe SaveFileDialog

No nosso aplicativo de exemplo, quando o usuário salvar os detalhes em arquivo, você vai solicitar o nome e o local do arquivo exibindo a caixa de diálogo comum Save File.

22. Na janela Code Text Editor que exibe Window.xam.cs, adicione a seguinte diretiva using à lista na parte superior do arquivo:

a classe SaveFileDialog está no namespace Microsoft.win32.

23. Localize o método saveMember_Click e adicione o código mostrado a seguir:

```
private void saveMember_Click(object sender, RoutedEventArgs e)
{
    SaveFileDialog saveDialog = new SaveFileDialog();
    saveDialog.DefaultExt = "txt";
    saveDialog.AddExtension = true;
    saveDialog.FileName = "Members";
    saveDialog.InitialDirectory = @"C:\Users\Cunha\Documents";
    saveDialog.OverwritePrompt = true;
    saveDialog.Title = "Meu sistema exemplo";
    saveDialog.ValidateNames = true;
}
```

Esse código cria uma instância da classe SaveFileDialog e configura suas propriedades. A tabela a seguir descreve o propósito dessas propriedades:

Propriedade	Descrição
DefaultExt	A extensão de nome de arquivo padrão a ser usada se o usuário não especificar a extensão ao fornecer o nome de arquivo.
AddExtension	Faz com que a caixa de diálogo adicione a extensão do nome de arquivo indicada pela propriedade DefaultExt ao nome do arquivo especificado pelo usuário, se o usuário omitir a extensão.
FileName	O nome do arquivo selecionado no momento. Você pode preencher essa propriedade para especificar um nome de arquivo padrão ou desmarcá-la se você não quiser um nome de arquivo padrão.
InitialDirectory	O diretório padrão a ser usado pela caixa de diálogo.
OverwritePrompt	Faz a caixa de diálogo alertar o usuário quando é feita uma tentativa de sobrescrever um arquivo existente com o mesmo nome. Para funcionar a propriedade ValidateNames

	também deve estar configurada como True.
Title	Uma string que é exibida na barra de título da caixa de diálogo.
ValidateNames	Indica se os nomes de arquivos são válidos. Se a propriedade ValidateNames estiver configurada como true, a caixa de diálogo também verificar se o nome de arquivo digitado pelo usuário contém apenas caracteres válidos.

24. Modifique o método saveMember_Click para ficar como mostrado a seguir:

```
private void saveMember_Click(object sender, RoutedEventArgs e)
{
    SaveFileDialog saveDialog = new SaveFileDialog();
    if (saveDialog.ShowDialog().Value)
    {
       using (StreamWriter write = new
StreamWriter(saveDialog.FileName))
       {
           saveDialog.DefaultExt = "txt";
           saveDialog.AddExtension = true;
           saveDialog.FileName = "Members";
           saveDialog.InitialDirectory =
@"C:\Users\Cunha\Documents";
           saveDialog.OverwritePrompt = true;
           saveDialog.Title = "Meu sistema exemplo";
           saveDialog.ValidateNames = true;
        }
    }
 }
```