Melhorando código de MessageBox()

Mário Leite

Com relação à linguagem **C#**, cheguei à seguinte conclusão: é, realmente, uma excelente linguagem de programação, com recursos extraordinários; e além de ser uma linguagem orientada a objetos também é orientada a eventos na plataforma Windows. E dentro do ambiente do Visual Studio esta linguagem se torna uma ferramenta de desenvolvimento de sistemas de alto nível profissional e de fácil aprendizado para os que já possuem uma base da linguagem **C**.

Entretanto, mesmo sendo uma excelente opção de ferramenta de desenvolvimento em ambiente gráfico, **C#** traz alguns inconvenientes, paradoxalmente, devido às facilidades oferecidas, com um preço a pagar: às vezes, o programador tem que escrever instruções muito longas: é a minha opinião! Neste contexto, tenho observado que as classes e estruturas oferecidas pela linguagem, não raramente, exige linhas de código de tamanho descomunal, o que torna o processo de codificação um pouco tedioso, fazendo com que o programador se perca em interináveis "acessórios" pendurados nos métodos. Observe esta linha de código para capturar o retorno de uma janela de diálogo com **MessageBox()** com dois botões de opções oferecidos: **[OK]** e **[Cancelar]**:

```
if(MessageBox.Show("Deseja mesmo sair?", "Saída", MessageBoxButtons.OKCancel, MessageBoxIcon.Question)==DialogResult.OK)
```

Realmente, é quase uma tortura digitar uma linha de código desta, devido ao tamanho dos nomes dos métodos oferecidos pela linguagem; e o pior: induz o programador a erros de digitação (mesmo com auxílio do *IntelliSense*) pois, a linha de código ocupa quase toda a tela do monitor de vídeo, obrigando-o a usar a barra de rolagem, o que é extremamente desconfortável para qualquer programador.

Por outro lado, sendo **C#** uma linguagem orientada a objetos, o programador pode (e deve) se aproveitar dos recursos oferecidos por essa tecnologia e criar instâncias das classes envolvidas para encurtar as linhas de código e oferecer um conforto na codificação, além de ser uma ótima oportunidade para melhorar e expandir seus conhecimentos sobre esse paradigma de programação.

A **figura 1** mostra como normalmente é feito para criar um código de janela de diálogo com o método **MessageBox**(), aumentado o tamanho da página do editor para conter toda a linha de código. A **figura 2** introduz um recurso muito utilizado por alguns programadores: a "partição" da linha de código em pedaços. A **figura 3**, na qual o programador pode se valer dos recursos da OOP, cria instâncias para serem colocadas como parâmetros do método **MessageBox()**, indicando um maior profissionalismo no tratamento da linguagem; e a **figura 4** exibe a janela de diálogo produzida por estas soluções.

Em princípio, o leitor pode achar que a primeira solução (da **figura 1**) é a melhor, pois exige menos linhas de código; entretanto, é muito importante frisar (principalmente para os iniciantes) que **C#** é uma linguagem orientada a objetos, e assim deve ser tratada. Deste modo, mesmo consumindo mais linhas de código, a solução da **figura 3** apresenta uma codificação mais elegante e totalmente integrada ao paradigma da OOP, permitindo uma melhor customização dos parâmetros de **MessageBox()**. Além do mais, é fundamental que o programador possa se valer do potencial oferecido pelo ambiente e usar, efetivamente, os recursos da OOP para se firmar neste paradigma, caso contrário de nada adiantará codificar nessa linguagem; seria como usar um fuzil para matar uma barata!

```
private void mnuSaida_Click(object sender, EventArgs e)
{
   if(MessageBox.Show("Deseja mesmo sair?","Saída",MessageBoxButtons.OKCancel,MessageBoxIcon.Question)==DialogResult.OK)
   {
       Application.Exit();
   }
   else
   {
       return;
   }

}
```

Figura 1 - Código tradicional com decisão em uma única linha

```
private void mnuSaida_Click(object sender, EventArgs e)
{
   if(MessageBox.Show("Deseja mesmo sair?",
       "Saida",
        MessageBoxButtons.OKCancel,
        MessageBoxIcon.Question)==DialogResult.OK)
    {
        Application.Exit();
    }
 }
```

Figura 2 - Código tradicional com decisão partida em várias linhas

```
private void mnuPrincCad_Click(object sender, EventArgs e)
{
    //Cria as instâncias
    MessageBoxButtons objBots = new MessageBoxButtons();
    MessageBoxIcon objIcon = new MessageBoxIcon();
    DialogResult objOK = new DialogResult();
    objBots = MessageBoxButtons.OKCancel;
    objIcon = MessageBoxIcon.Question;
    objOK = DialogResult.OK;
    //Cria as strings da mensagem
    string msg = "Deseja mesmo sair?";
    string tit = "Saída";
    //Executa a estrutura de decisão
    if (MessageBox.Show(msg, tit, objBots, objIcon) == objOK)
    {
        Application.Exit();
    }
    else
    {
        return;
    }
}
```

Figura 3 - Código completo personalizado com os recursos de OOP

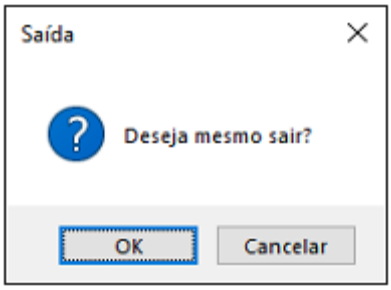

Figura 4 - Janela de diálogo