**Validando Entrada de Dados Numa Interface**

**Mário Leite**

...

Uma das preocupações que o programador deve ter ao implementar um módulo de entrada de dados em qualquer sistema CRUD, é com a validação instantânea desses dados digitados pelo usuário do pois, não raramente, ocorrem erros na entrada. Problemas esses que não são, evidentemente, de ordem ortográfica (por exemplo, o nome do cliente digitado como *Sousa* ao invés de *Souza*), mas sim quando um caractere estranho (por exemplo, um *número* em um nome de pessoa).Por outro lado, imaginem que no momento de digitar o valor da duplicata de um cliente o usuário digite **R** no lugar do número **4**, por exemplo; nos dois casos tem-se "situações de risco" que poderiam ser evitadas já no ato da digitação, e não precisando validar só no momento de confirmar a entrada de todos os dados com um clique num botão de pressão.

Outra situação bastante usual na entrada de dados numa interface de cadastramento é quando o usuário deseja "passar" para uma outra caixa de texto pressionando a tecla **Enter** ou as setas de navegação “**Para Cima**” e/ou “**Para Baixo**”. Estas são situações reais herdadas de sistemas desenvolvidos no ambiente DOS e que o programador-usuário de ferramentas do tipo RAD (Java, C#, Delphi, VB.net, etc) tem que conviver, apesar de ser a tecla **Tab** o padrão para passar de uma caixa de texto para outra, no ambiente Windows**.** Todavia, convencer o usuário disto é muito difícil, e a "bomba" acaba sempre estourando nas mãos do desenvolvedor; é aquela velha estória que se repete: "*facilidade para o usuário*, *dificuldade para o programador*".Mas, como o usuário (leia-se cliente) sempre tem razão, o programador deve atendê-lo nas suas exigências; aliás, exigências essas que forçam o desenvolvedor a produzir *softwares* mais profissionais e mais eficazes, além de eficientes**.**

Para satisfazer as exigências citadas, são mostradas três *procedures-eventos* em Delphi e em C#: na primeira é validada uma caixas de texto de modo que só aceite **letras**, **BackSape** e o **espaço**; a segunda rotina valida uma caixa de edição que só aceita **ponto decimal**, **números**, e **BackSape**, e a terceira rotina permite que o usuário navegue pelas caixas pressionando a tecla **Enter** oucom as teclas de “**Seta para Baixo**” e “**Seta para Baixo**”. Neste último caso a propriedade *TabOrder* das caixas de edição deve estar na ordem pré-estabelecida para “receber o foco”; isto quer dizer que a primeira caixa deve ter *TabOrder* igual a 0, a segunda 1, a terceira 2, e assim sucessivamente para que os "saltos" sejam dados na sequência correta, conforme desejado pelo usuário.Para validar o tipo de caractere digitado é utilizada a *procedure* ligada ao evento *KeyPress*; e aproveitando o fato do parâmetro **Key** (código da tecla pressionada) ser recebido “por referência”, foi configurado o valor para **0** (nenhuma tecla pressionada) quando esta não for uma tecla permitida.

No caso de navegação pelas caixas de texto foi usada a *procedure* gerada pelo evento *KeyDown*, empregando o método **Perform()** com os parâmetros adequados para isto, em função do valor de **Key**. Neste caso é possível "saltar" para a próxima caixa de edição pressionando a tecla **Enter** ou com a tecla “**Seta para Baixo**”; também é possível "voltar" para a caixa anterior pressionando a tecla “**Seta Para Cima**”. E nesta codificação um detalhe muito importante deve ser observado pelo programador**:** é preciso digitar o código apenas uma vez (na primeira caixa de edição); depois é só fazer a associação desse evento para todas as outras**.** Mas, ATENÇÃO: não pode existir nenhum botão de pressão na interface com propriedade *Default* igual a *True*; se isto ocorrer não se consegue passar de uma caixa para outra com **Enter** pois, neste caso, a preferência de execução é do botão configurado.

Nos códigos em **C#**, para que as teclas **Enter**, “**Seta para Cima**” e “**Seta para Baixo**” funcionem como **Tab** passando de uma caixa de texto para a outra seguinte, a propriedade *KeyPreview* do *formulário* deve estar configurada para *True*.

------------------------------------------------------------------------------------------------------------------------------------

**procedure** TForm1**.**TxtNome**KeyPress**(Sender: TObject; **var** Key: **car**);

**begin**

**If**(**Not**(Key **IN** [**'a'..'z'**, **'A'..'Z'**, #8, #32])) **then begin**

**MessageBeep**(0); //emite um alerta sonoro

Key := #0; //reinicializa o parâmetro Key

**end**;

**end**;

**Rotina 1.1 (Delphi) - Só aceita Letras,** <**BackSpace**> **e** <**Espaço**>

**procedure** TForm1**.**TxtValor**KeyPress**(Sender: TObject; **var** Key: **char**);

**begin**

**If(Not**(Key **IN** [#46, #48..#57, #8])) **then** **begin**

**MessageBeep**(0); //emite um alerta sonoro

Key := #0; //reinicializa o parâmetro Key

**end**;

**end**;

**Rotina 1.2 (Delphi) - Só aceita Ponto decimal, Números e** <**BackSpace**>

**procedure** TForm1**.**TxtCodigo**KeyDown**(Sender: TObject; **var** Key: **word**;

Shift: TShiftState);

//As duas instruções da estrutura "Case" devem ser mapeadas para os eventos

//"***OnKeyDown***" de todas as outras caixas de edição.

**begin**

**Case** Key **of**

VK\_RETURN,VK\_DOWN**: Perform**(WM\_NEXTDLGCTL, 0, 0);

VK\_UP**:** **Perform**(WM\_NEXTDLGCTL, 1, 0);

**end**;

**end**;

**Rotina 1.3 (Delphi) - Permite navegação com setas ou** <**Enter**>

**private void** txtNome\_KeyUp(**object** sender, KeyEventArgs e)

{

int tecla = e.KeyValue;

if (**!**( ((tecla >= 65) && (tecla <= 90)) || ((tecla >= 97) && (tecla <= 122))

|| ((tecla == 8) || (tecla == 32)))) {

txtNome.Clear();

}

}

**Rotina 2.1 (C#) - Só aceita Letras,** <**BackSpace**> **e <Espaço**>

**private** **void** txtSalario\_KeyUp(**object** sender, KeyEventArgs e)

{

int tecla = e.KeyValue;

if (**!**(((tecla >= 48) && (tecla <= 57)) || ((tecla == 46) && (tecla == 8))))

{

txtSalario.Clear();

}

}

**Rotina 2.2 (C#) - Só aceita Ponto decimal, Números e** <**BackSpace**>

**private** **void** Form1\_KeyDown(**object** sender, KeyEventArgs e)

{

if((e.KeyCode == Keys.Enter) || (e.KeyCode == Keys.Up) || (e.KeyCode ==

Keys.Down))

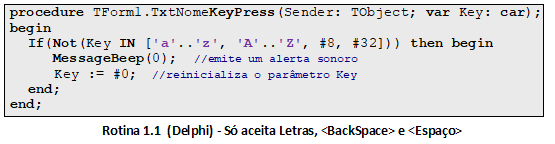
{

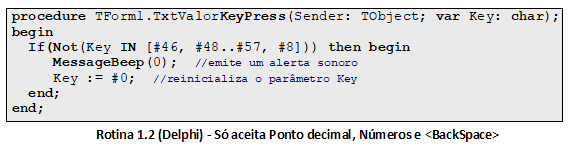
this.SelectNextControl(this.ActiveControl, !e.Shift, true, true, true);

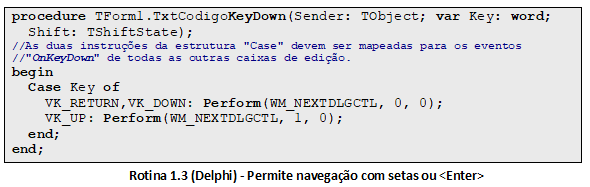
}

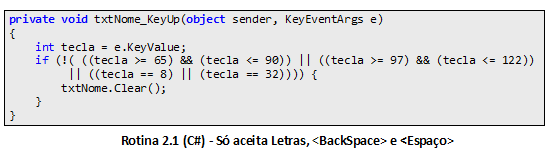
}

**Rotina 2.3 (C#) - Permite navegação com setas ou** <**Enter**>







****

