MÓDULO III – PARTE FINAL

No módulo II, vimos que devemos ter um cuidado todo especial quando armazenamos configurações que não devem sofrer alterações (re-gravações). A função Existe_ArquiINI que não é uma função que acompanha o Delphi, resolve o nosso principal problema. Nessa última parte de nossa apostila, vamos trabalhar com algumas funções e procedimentos que ainda não vimos, são elas: DeleteKey, EraseSection, ReadSection, ReadSections, ReadSectionValues, SectionExists, UpdateFile e ValueExists. Você pode estar se perguntando, porque será que o "cara" deixou essas funções e procedimentos para o final? Será que são as mais difíceis? Por contrário, as deixei mais para o final justamente porque algumas dessas funções e procedimentos você jamais irá usar; outras funções e procedimentos você irá usar muito pouco. É evidente que tudo depende também do gosto e como cada um desenvolve os seus sistemas.

Função UpdateFile

A função UpdateFile não possui qualquer importância se os sistemas operacionais em uso forem de versões superiores ao Windows 98. Se você ainda desenvolve aplicações para as versões do Windows 95 e 98, **deve usar sempre essa função**. Ela libera o buffer e grava os dados no arquivo. Se você ainda trabalha com essas versões, sempre chame essa função após executar o(s) comando(s) **Write**. Vamos usar um fragmento da Lição III e mostrar onde o comando deve ser "encaixado".

Vejamos um pequeno exemplo:

```
Var
  mMeuIni : TInifile;
Begin
      mMeuIni := TIniFile.Create('Configuracoes.Ini');
      ...
      ...
      mMeuIni.WriteString ('APLICACAO','NOME',Edit1.Text);
      mMeuIni.WriteBool   ('APLICACAO','INI_BACANA',CheckBox1.Checked);
      mMeuIni.WriteBool   ('APLICACAO','ARMAZENAR',CheckBox2.Checked);
      mMeuIni.WriteInteger('APLICACAO','VALORES',ComboBox1.ItemIndex);
      mMeuIni.UpdateFile;
      mMeuIni.Free;
end;
```

EXEMPLO 7 – USANDO ReadSection, ReadSections, ReadSectionValues, EraseSection e SectionExists

Muitas vezes você gostaria de recuperar um conjunto de valores que estão armazenados dentro de uma certa seção do seu arquivo ini. Um bom exemplo para isso seria recuperar os dados cadastrais da empresa ao invés de fazer uma leitura chave a

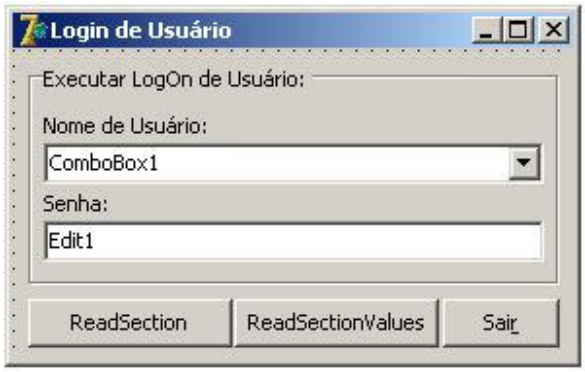
Contatos: Exio – E-Mail: bitzero_000@yahoo.com.br

chave. Um outro exemplo que poderia ser utilizado é carregar valores em um ComboBox de usuários que efetuaram login no sistema. Para deixar a coisa mais interessante, vamos usar essa segunda idéia como exemplo de desenvolvimento e o uso das procedures acima.

OBSERVAÇÃO: ESSA ABORDAGEM É MERAMENTE ILUSTRATIVA. O COMPONENTE COMBOBOX POSSUI AS FUNÇÕES: **SaveToFile** e **LoadFromFile** QUE SERIAM MUITO MAIS FÁCEIS DE SEREM UTILIZADAS, DO QUE CRIAR UMA SEÇÃO NO ARQUIVO INI PARA ESSA TAREFA.

1. Crie uma nova aplicação;
2. Coloque um componente ComboBox;
3. Coloque três componentes Buttom;

Veja a figura abaixo (o meu form possui alguns componentes a mais, mas são somente para enfeite. O que me interessa nesse form é somente o combobox e o buttom):



4. No evento **onExit** do ComboBox1, entre com o código abaixo:

```
begin
if (ComboBox1.Items.IndexOf(ComboBox1.Text) = -1) then
   ComboBox1.Items.Add(ComboBox1.Text);

{A função IndexOf retorna o valor -1 se a String não estiver armazenada
 no ComboBox1. Qualquer valor superior a -1 é sinal de que a String já
 está armazenada e o valor de retorno é a posição da String dentro do
 ComboBox.

 A leitura seria mais ou menos algo assim: Procure no ComboBox o texto
 digitado no próprio ComboBox, se não achar o texto, insere ele
 na lista; se achar o texto não faz nada.}

 end;
```

5. Execute a aplicação e veja se tudo está dentro do esperado. Digite alguns nomes e não esqueça de pressionar a tecla **TAB** para sair do ComboBox e testar o processo de inserção de nomes.
6. Vamos criar uma procedure que armazenará todos os nomes que foram inseridos no ComboBox, para isso, digitamos o seguinte código:

Contatos: Exio – E-Mail: bitzero_000@yahoo.com.br

```
procedure TForm1.GravarLista(vCombo: TComboBox);
var
  vMeuIni : TIniFile;
  I : SHORT; //só por questão de economia
begin
   vMeuIni := TIniFile.Create('ListasDeString.ini');
  try

   if vMeuIni.SectionExists('USUARIOS') then
      vMeuIni.EraseSection('USUARIOS');

   For I := 0 to Pred(vCombo.Items.Count) do
   begin
       vMeuIni.WriteString('USUARIOS','NOME_' +
                        IntToStr(I),vCombo.Items.Strings[I]);
   end;
  finally
   vMeuIni.Free;
  end;
end;
```

EXPLICANDO A PROCEDURE GravarLista: Declaramos a variável I como SHORT justificando-se o baixo número de usuários do sistema que poderiam se logar em um único computador. Isso aqui é só uma criação que pode ser até levada em consideração, porém não recomendo. O laço **FOR** é usado para pegar cada nome armazenado no ComboBox e ir gravando com o seu respectivo índice numa chave que é a combinação da string '**NOME_**' mais a variável **I**. Para demonstrar o uso das função **SectionExists**, eu fiz com que fosse realizada uma verificação antes de executar o processo de escrita no arquivo ini. Se a seção **USUARIOS** EXISTIR, toda a seção será excluída (juntamente com as chaves e seus respectivos valores). A procedure **EraceSection** é a responsável por executar o processo de exclusão da seção.

7. Vamos adicionar a procedure **GravarLista** no evento onClick do botão **Sair** dessa forma:

```
procedure TForm1.BuSairClick(Sender: TObject);
begin
  if MessageDlg('Você deseja salvar os nomes inseridos no ComboBox?',
     mtConfirmation,[mbYes,mbNo], 0) = mrYes then
     GravarLista(ComboBox1);

  Close;
end;
```

8. Agora vamos criar a procedure que fará a leitura dos nomes no momento que o form de logon for apresentado. Adicione o código abaixo no evento **onShow** do Form de logon:

Contatos: Exio – E-Mail: bitzero_000@yahoo.com.br

```
procedure TForm1.CarregarComboBox(vCombo:TComboBox);
var
  vMeuIni : TIniFile;
  vLista  : TStringList;
  I       : SHORT;
  Excluir : String;
begin
   vMeuIni := TIniFile.Create('ListasDeString.ini');
   vLista  := TStringList.Create;
   ComboBox1.Clear; //Exclui todos os itens - se houverem é claro
  try
    if vMeuIni.SectionExists('USUARIOS') then
    begin
     vMeuIni.ReadSectionValues('USUARIOS',vLista);

     For I := 0 to Pred(vLista.Count) do
     begin
       if (Pos('=', vLista.Strings[I]) > 0) then
       begin
         Excluir := vLista.Strings[I];
         Delete(Excluir,1,Pos('=', Excluir));
         ComboBox1.Items.Add(Excluir);
       end;
     end;
     ComboBox1.ItemIndex := 0;
    end;
  finally
   vMeuIni.Free;
   vLista.Free;
  end;
end;
```

EXPLICAÇÕES DA PROCEDURE CarregarComboBox: Como vimos nas primeiras páginas desse tutorial, a procedure **ReadSectionValues** retorna para o usuário uma lista contendo todas as chaves e seus respectivos valores. Foi isso que nos "obrigou" a criar uma variável do tipo TStringList chamada **vLista**. A procedure **ReadSection** retorna somente os nomes da chaves de uma seção e não os seus valores. Após obter a lista de nomes na variável vLista, um laço **For** se encarrega de executar a "limpeza", excluindo de cada string da lista o início "**NOME_**" até o sinal de igual. Executado esse processo, o item é inserido agora no **ComboBox** e o índice do ComboBox passa a ser o primeiro elemento da tabela.

9. Compile e execute a aplicação. Após a primeira rodada de nomes que você inseriu, pressione o botão Sair e confirme a pergunta pressionando no botão Ok.

No exercício que acompanha esse tutorial, pode ver que existem 3 botões, rode a **lição VII** e veja o que cada botão faz e principalmente os seus resultados.

BONUS PACK – JUNTO COM ESSAS TRÊS APOSTILAS, VOCÊ RECEBEU TAMBÉM OS RESPECTIVOS ARQUIVOS PARA ESTUDAR, PRATICAR E USAR COMO REFERÊNCIA. VOCÊ

Contatos: Exio – E-Mail: bitzero_000@yahoo.com.br

PODERÁ ALTERAR, COPIAR, DISTRIBUIR E OBVIAMENTE, A ÚNICA COISA PROIBIDA SERÁ VENDER.

Contatos: Exio – E-Mail: bitzero_000@yahoo.com.br