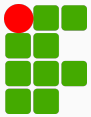
Lazarus

02 - Formulários e Caixas de Diálogo

Marcos Roberto Ribeiro



Instituto Federal Minas Gerais - Campus Bambuí

2018

Introdução

- Formulários são interfaces visuais de um programa com o usuário;
- Dentro de um formulários podem ser inseridos diversos componentes que podem facilitar a interação entre o usuário e o programa;
- Os formulários possuem diversas propriedades e eventos que podem caracterizar seu comportamento;
- O Lazarus fornece um construtor de formulários visual que agiliza bastante o processo de construção de interfaces;

Formulários - Principais Propriedades

ActiveControl: Componente focado quando o formulário é criado;

AutoScroll: Barras de rolagens automáticas se os componentes não couberem na área visível do formulário;

BorderIcons: Ícones visíveis na borda do formulário[1];

BorderStyle: Estilo da borda do formulário. Principais valores:

- **bsDialog:** Estilo de caixa de diálogo;
- **bsNone:** Sem bordas;
- **bsSingle:** Simples;
- **bsSizeable:** Redimensionável;

BorderWidth: Largura da borda do formulário;

Caption: Título do formulário;

Color: Cor do formulário;

Cursor: Cursor do mouse;

[1] Seu funcionamento efetivo poder ser afetado pelas propriedades *BorderStyle* e *WindowState* dependendo do ambiente gráfico utilizado.

Formulários - Principais Propriedades

Enabled: Habilita ou desabilita o formulários (O usuário não consegue interagir com um formulário desabilitado);

Height: Altura do formulário;

Hint: Dica a ser exibida quando o cursor do mouse é posicionado sobre o formulário;

Icon: Ícone do formulário;

Left: Posição esquerda;

Name: Nome do formulário (será usado na classe do formulário e em sua instância);

Position: Posição do formulário na tela. Principais valores:

- **poDesigned:** Posição de edição;
- **poScreenCenter:** No centro da tela;

ShowHint: Indica se a dica (*Hint*) será exibida;

Top: Posição superior;

Width: Largura do formulário;

WindowState: Estado do formulário. Principais valores:

- **wsNormal:** A propriedade *BorderStyle* não é afetada;
- **wsMaximized:** A propriedade *BorderStyle* é afetada;

Formulários - Principais Eventos

Activate: Ativação (exibição);
Click: Clique;
Close: Fechamento;
CloseQuery: Antes do fechamento;
Create: Criação;
DblClick: Clique duplo;
Deactivate: Desativação;
Hide: Ocultação;
KeyDown: Pressionamento de tecla (na descida);
KeyPress: Pressionamento de tecla (pressionar e soltar);
KeyUp: Soltar uma tecla pressionada;
MouseDown: Pressionamento do botão do mouse (na descida);
MouseMove: Movimentação do mouse;
MouseUp: Soltar o botão do mouse pressionado;
Resize: Redimensionamento;
Show: Exibição (quando o formulário estava oculto);

Exemplos de Manipulação de Eventos de Formulário

```
 procedure TFormPrincipal.FormCloseQuery(Sender: TObject;
↪  var CanClose: boolean);
 begin
   If MessageDlg('Deseja encerrar o programa?',
↪  mtConfirmation, [mbYes, mbNo], 0) = mrNo then
     CanClose:=false;
 end;

 procedure TFormPrincipal.FormMouseMove(Sender: TObject;
↪  Shift: TShiftState; X, Y: Integer);
 begin
   Caption:='X = ' + IntToStr(X) + ',Y = ' + IntToStr(Y);
 end;
```

Múltiplos Formulários

- Em muitos programas são necessários vários formulários;
- Para controlar a exibição de vários formulários, podemos utilizar os métodos:

  Show(): Exibe um formulário;

  Hide(): Oculta um formulário;

  ShowModal(): Exibe um formulário de forma "modal" (não é possível alternar com outro formulário do programa);
- Para definir o formulário inicial basta acessar as "Opções do Projeto" no menu: *Project* > *Project Options*
  - Na aba *Forms* podemos mudar a ordem de criação dos formulários;
  - O primeiro formulário criado será o formulário inicial.

Dicas

- Antes de chamar o método *ShowModal()* chame o método *Hide*, pois se o formulário está visível e chamamos *ShowModal* ocorrerá um erro;
- Na codificação adote "()" após os métodos para diferenciá-los de propriedades;

Formulários - Exercícios com Propriedades

1) Crie um projeto com um formulário, aumente o seu tamanho, insira diversos componentes *Edits*, reduza o tamanho do formulário e configure-o para apresentar barras de rolagem de forma que todos os componentes possam ser acessados;
2) Alterne entre os *Edits* a serem focados na inicialização do formulário;
3) Altere o título do formulário para "Teste";
4) Altere o cursor do mouse e a cor do formulário;
5) Configure o formulário para que seja exibida a dica "Testando";
6) Faça com que o formulário seja exibido no centro da tela em sua inicialização;

Formulários - Algumas Dicas

Dica

É importante adotarmos padrões nos nomes de formulários e componentes pois assim fica mais fácil de referenciá-los no código. No projeto anterior poderíamos *renomear* o formulário *FormTestePropriedades*.

Dica

É importante também padronizarmos os nomes de arquivos aos salvá-los. No caso de formulários é interessante começar o nome com *f* ou *fm*. No caso do formulário anterior devemos salvá-lo como *ftestepropriedades* ou *fmtestepropriedades*. Além disso os nomes do arquivos devem ser todos em *minúsculas* para não termos problemas com sistemas de arquivos que diferenciam minúsculas de maiúsculas.

Formulários - Exercícios com Eventos

1) Crie um projeto com um formulário altere seu título de acordo com a manipulação de seus eventos. Por exemplo, suponha que o evento *KeyPress* deva ser manipulado para que o título do formulário mude para *Tecla pressionada*. O seguinte código resolve este problema:

```
procedure TFormPrincipal.FormKeyPress(Sender: TObject; var Key: char);
begin
  Caption:='Tecla pressionada';
end;
```

Os seguintes eventos devem ser considerados (use *ShowMessage* para os métodos marcados com *):

- Activate;
- Click;
- Create*;
- DblClick*;
- Deactivate;
- KeyDown;
- KeyUp;
- MouseDown;
- MouseMove;
- MouseUp;
- Resize;
- Show*.

Mais Exercícios

- O que é necessário para disparar o evento *Activate* mais de uma vez?
- Os eventos marcados com * no exercícios anterior foram codificados usando a função *ShowMessage* para não sobrepor o *Caption* que é gravado em outros eventos. Explique o que isto quer dizer. Dê um exemplo de eventos que se sobrepõem.

Caixas de Diálogo

As principais caixas de diálogos presentes do Lazarus são:

ShowMessage(): Exibição de mensagens simples;

InputBox(): Obtenção de respostas digitadas usuário;

MessageDlg(): Comunicação com o usuário de acordo com os botões pressionados.

ShowMessage()

- A **ShowMessage()** recebe um único parâmetro (**String**) e exibe para o usuário.

```
...
procedure TForm1.FormCreate(Sender: TObject);
begin
  ShowMessage('Olá Mundo!');
end;
...
```

InputBox()

- A **InputBox()** além de exibir uma mensagem, permite que o usuário informe uma resposta;
- Os parâmetros da rotina é **function InputBox(const ACaption, APrompt, ADefault: String): String** são:

  ACaption: título da janela;
  APrompt: mensagem a ser exibida;
  ADefault: valor padrão para a resposta.
- Como a rotina é uma função, seu resultado será a resposta informada pelo usuário.

```
...
procedure TForm1.FormCreate(Sender: TObject);
var
  Texto: String;
begin
  Texto:= InputBox('Saudações', 'Informe o seu nome', 'Digite aqui');
  ShowMessage('Bem vindo! ' + Texto);
end;
...
```

MessageDlg()

- A **MessageDlg()** também exibe uma mensagem e permite obter uma resposta do usuário. Porém a resposta não é digitada e sim um informação sobre um botão pressionado;
- Os parâmetros da rotina é **function MessageDlg(const aMsg: String; DlgType: TMsgDlgType; Buttons: TMsgDlgButtons; HelpCtx[2]: Longint): Integer;** são:

  aMsg: mensagem a ser exibida;

  DlgType: tipo de caixa de diálogo;

  Buttons: botões que serão exibidos.
- Os tipos de mensagens são:

  mtWarning: Mensagem de aviso de atenção;

  mtError: Mensagem de erro;

  mtInformation: Mensagem informativa;

  mtConfirmation: Mensagem de confirmação;

  mtCustom: Mensagem Customizada;

[2] O parâmetro **HelpCtx** está relacionado com o sistema de ajuda que não é abordado nesta aula. Portanto vamos informar sempre **0** para este parâmetro

MessageDlg()

- Os botões são conjuntos contendo:
  - **mbYes:** Sim;
  - **mbNo:** Não;
  - **mbOK:** Ok;
  - **mbCancel:** Cancelar;
  - **mbAbort:** Abortar;
  - **mbRetry:** Repetir;
  - **mbIgnore:** Ignorar;
  - **mbNoToAll:** Não para todos;
  - **mbYesToAll:** Sim para todos.
- Por se tratar de um conjunto, devemos informar os botões desejados separados por vírgula e entre colchetes. Por exemplo, para termos os botões **Sim** e **Não** devemos informar **[mbYes, mbNo]**.

MessageDlg()

- A **MessageDlg()** é uma função que retorna um inteiro. A identificação do botão pressionado pelo usuário é feita pelas contantes:

  mrYes: Sim;
  mrNo: Não;
  mrOK: Ok;
  mrCancel: Cancelar;
  mrAbort: Abortar;
  mrRetry: Repetir;
  mrIgnore: Ignorar;
  mrNoToAll: Não para todos;
  mrYesToAll: Sim para todos;

- Para utilizar as constantes é preciso declarar a **unit controls** na cláusula **uses**.

Exemplo com MessageDlg()

```
...
procedure TForm1.FormCreate(Sender: TObject);
begin
  case MessageDlg('Pressione um botão', mtWarning, [mbYes, mbNo,
  ↪  mbOK, mbCancel, mbAbort, mbRetry, mbIgnore, mbNoToAll,
  ↪  mbYesToAll], 0) of
    mrYes: ShowMessage('Você pressionou SIM');
    mrNo: ShowMessage('Você pressionou NÃO');
    mrOk: ShowMessage('Você pressionou OK');
    mrCancel: ShowMessage('Você pressionou CANCELAR');
    mrAbort: ShowMessage('Você pressionou ABORTAR');
    mrRetry: ShowMessage('Você pressionou REPETIR');
    mrIgnore: ShowMessage('Você pressionou IGNORAR');
    mrNoToAll: ShowMessage('Você pressionou NÃO PARA TODOS');
    mrYesToAll: ShowMessage('Você pressionou SIM PARA TODOS');
  end;
end;
...
```

Conversão de Tipos

- A **InputBox()** obtém sempre **Strings** do usuário. Porém, certas vezes precisamos trabalhar com outros tipos de dados;
- Para converter **Strings** para tipos numéricos podemos usar a rotina **Val(S: String; var N: Integer ou Real; Erro: Integer)**[3];
- Já o processo inverso pode ser feito com as funções:
  IntToStr(n: Integer): String Converte de inteiro para **String**;
  FloatToStr(n: Real): String Converte de real para **String**.
- A conversão para **String** é importante pois muitas vezes devemos mostrar mensagens para o usuário usando apenas **Strings**. Como por exemplo no **ShowMessage()**;
- Estas e outras rotinas de conversão estão disponíveis na **unit SysUtils**.

[3] A rotina **Val()** tenta converter o **String S** para valor numérico. Se a conversão for possível, **Erro** recebe **0**.

Referências

- LCL Documentation. Unit Dialogs. Disponível em http://lazarus-ccr.sourceforge.net/docs/lcl/dialogs
- LCL Documentation. Unit SysUtils. Disponível em http://lazarus-ccr.sourceforge.net/docs/rtl/sysutils
- LCL Documentation. Unit Controls. Disponível em http://lazarus-ccr.sourceforge.net/docs/lcl/controls
- LCL Documentation. Forms. Disponível em http://lazarus-ccr.sourceforge.net/docs/lcl/forms/index.html;
- Lazarus Wiki, Dialog Examples. Disponível em http://wiki.lazarus.freepascal.org/Dialog_Examples.