Nesta seção você encontra artigos intermediários sobre Delphi Win32 e Delphi for .NET

Criando um componente de pesquisa

Reutilize as telas de pesquisa em sua aplicação

Resumo DevMan

O reaproveitamento de código é a preocupação de praticamente todo desenvolvedor. Evitar redigitar o mesmo código diversas vezes no mesmo sistema ou pior, replicar para outros produtos da empresa é uma perda de tempo para toda e qualquer equipe. Portanto, o melhor a se fazer é tentar sanar esse problema utilizando-se de artifícios bem comuns de toda linguagem de desenvolvimento.

A criação de funções, dlls e componentes é sem dúvida a melhor opção. Nesse artigo, veremos como criar um componente de pesquisa para ser utilizado em todo o sistema. Dessa forma, evitamos redigitação desnecessária de código em partes distintas do sistema.

Nesse artigo veremos

• Criação de componentes;

• Criação de pacote para conter nossos componentes.

Qual a finalidade

• Criar um componente de pesquisa totalmente funcional que poderá ser utilizado em qualquer parte do sistema.

Quais situações utilizam esses recursos?

• Em qualquer aplicação que utiliza pesquisas, praticamente em toda aplicação pode ser usado esse tipo de recurso.

Luciano Pimenta

lucianoalmeidapimenta@gmail.com

é Técnico em Processamento de Dados, desenvolvedor Delphi/C#. Palestrante da 4ª edição da Borland Conference (BorCon). Autor de mais de 50 artigos e de mais de 250 vídeo-aulas publicadas em revistas e sites especializados. Atualmente é programador Web da FullSoft - Mobile Solutions em Caxias do Sul-RS. Blog: lucianopimenta.blogspot.com. Site: www.lucianopimenta.net. É consultor da FP2 Tecnologia (www.fp2.com.br).

Em uma aplicação, a quantidade de consultas auxiliares é bastante grande, ou seja, pesquisa de clientes, fornecedores, produtos, etc., além de serem utilizadas em vários locais da aplicação. Criar um formulário de consulta para cada uma das pesquisas é trabalhoso, então por que não criar um componente para essas consultas? Certamente, posso citar uma infinidade de motivos para se criar um componente para efetuar pesquisas, alguns dos principais motivos são:

• *Encapsulamento:* encapsulamento é uma das práticas, em minha opinião, mais interessantes em programação OO, já que o código fica todo isolado em uma classe, componente, biblioteca etc., fazendo com que tenhamos mais controle sobre o que ocorre na aplicação;

• *Reaproveitamento de Código:* utilizar o mesmo código em diversos lugares da aplica-

ção com certeza é dar um *tiro no pé*. Estando o código em um único repositório, temos a certeza de que após uma possível manutenção nele, toda a aplicação sofrerá a alteração;

O que vou mostrar neste artigo, é um componente simples, para localizar dados, que pode ser modificado de acordo com a necessidade do desenvolvedor.

Iniciando a construção

A primeira providência a tomarmos é criar um pacote para alocar os arquivos de nosso componente. É importante criarmos um pacote, pois a partir da versão 2006 do Delphi, não há como instalarmos componentes utilizando apenas o *.pas* do mesmo. Além disso, nosso componente conterá um formulário para efetuar as pesquisas e devolver o resultado. Nesse artigo utilizaremos o Delphi 7, porém essa mesma prática pode ser realizada em qualquer outra versão de sua escolha.

Para criar um pacote clique em *File>New>Other>New>Package* conforme podemos visualizar na **Figura 1**. Após a criação do pacote salve o mesmo com o nome *pkLocalizar.dpk* utilizando o menu *File>Save*.

Criando o formulário

Teremos de criar o formulário de pesquisa, que deve conter os componentes para digitar o texto, botão de pesquisa e um *grid* onde serão mostrados os dados, além de um *DataSource* que será vinculado a um *DataSet*. Você pode incluir componentes de sua necessidade. A idéia da criação do formulário é bem simples. Ele se tornará padrão em nosso sistema, ou seja, toda tela que necessitar de uma pesquisa fará uso do componente e conseqüentemente da janela padrão de pesquisa, dando um ar profissional à aplicação.

Entre no menu *File>New>Form* e salve-o como *frmPesquisa.pas* e em sua propriedade *Name* digite *frLocalizar*. Insira alguns componentes de maneira que o layout da tela se pareça com a **Figura 2**.

No código do formulário devemos criar alguns métodos e propriedades, conforme o código a seguir:

```
procedure Buscar;
function GetCodigo: integer;
property DataSet: TClientDataSet;
```

No código anterior, criamos um método que será responsável por realizar a consulta (*Buscar*) e uma função. Além disso, temos que criar uma propriedade do tipo *TClientDataSet*, responsável por ter os dados da consulta.

Caso deseje, mude para o controle que utiliza para trabalhar com dados locais (*IBQuery*, *Query* etc.). O código do *Buscar*, simplesmente, repassa para o *ClientDataSet* o valor do *Edit1*, onde o usuário digitará o valor de busca:

```
FDataSet.Close;
FDataSet.Params[0].AsString := '%' + Edit1.
Text + '%';
FDataSet.Open;
```

O código para retornar o campo chave da tabela é bastante simples, onde retornamos o primeiro campo (*Field*) do *ClientDataSet*:

```
Result := DataSet.Fields[0].AsInteger;
```

Claro, precisamos indicar o campo chave na consulta SQL do *ClientDataSet* para que seja o primeiro registro. Ao apertar *Crtl+ Shift + C*, o Delphi cria o manipulador para a propriedade *DataSet*. O código do *SetDataSet* fica conforme o código a seguir:

```
if FDataSet <> Value then
begin
  FDataSet := Value;
  DataSource1.DataSet := FDataSet;
end;
```

No código, atribuímos a propriedade *DataSet* do *DataSource*, o valor da propriedade. Assim, ao configurarmos a propriedade no *Object Inspector*, o *DataSet* é vinculado ao *DataSource*.

Codificando o formulário

Para os botões, que vão indicar se devemos confirmar o item selecionado no *DBGrid* ou cancelar o formulário (fechando-o), devemos chamar o *Close*, e a seguir, configurar o *ModalResult*, que para o botão OK, será *mrOK* e *Cancelar* como *mrCancel*.

Veremos adiante, o motivo de usarmos o *ModalResult*. No evento *OnDblClick* do *DBGrid*, devemos usar o mesmo código do botão OK, apenas verificando antes se existem registros na consulta, verificando a propriedade *RecordCount* do *FDataSet*. Assim, o usuário pode dar um duplo clique no *grid* e o mesmo será fechado.

No *OnDataChange* do *DataSource*, vamos habilitar o OK, apenas se existirem registros no *ClientDataSet*, ou seja, mais uma validação que fizemos para não termos problemas com o componente:

```
btnOk.Enabled := FDataSet.RecordCount > 0;
```

Nota do DevMan

Até a versão 7.0 do Delphi era possível fazer a instalação de componentes apenas utilizando o arquivo .pas ou .dcu do componente. Já no Delphi 8.0 em diante esse recurso (*Component>Install component*) não está mais acessível. Agora, é necessário criar um novo pacote utilizando o menu File>New>Package – Delphi for Win32. Em seguida salve o pacote e adicione o arquivo .pas ou .dcu do componente que deseja instalar. Salve novamente o pacote, compile e instale.

A partir de agora, sempre que quiser instalar esse mesmo componente, basta abrir o pacote, compilar e instalar.

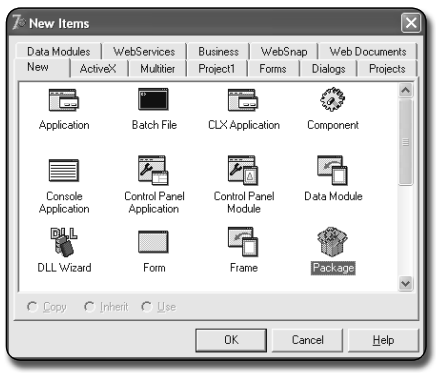

Figura 1. Criando o pacote do componente

Figura 2. Formulário de pesquisa

Para finalizar o formulário, precisamos tratar o evento *OnKeyPress* do *Edit*, usando o código da **Listagem 1**.

No código anterior, primeiramente, verificamos se foi apertada a tecla *Enter*. Após, verificamos se o *Edit* não está vazio, isso é bom para que uma pesquisa muito grande não seja executada, trazendo lentidão ao sistema.

Poderíamos ainda, colocar uma quantidade mínima de caracteres a serem digitados para que o usuário chegue o mais perto possível do que está procurando. Após verificarmos se a propriedade *DataSet* não está vazia, chamamos o método *Buscar*, configuramos o foco para o *DbGrid* e adicionamos uma mensagem na *StatusBar*, com a quantidade de registros encontramos com o valor digitado no *Edit*.

O *key := #0* não emite o bip característico do sistema operacional. Com essa configuração, o formulário de consulta está pronto. Alguns retoques a mais poderiam ser feitos, como configurar as colunas do *Grid* para que fiquem centralizadas pelo código e alinhadas à esquerda com o texto.

Listagem 1. Código do OnKeyPress do Edit

```
{ se apertou ENTER, realiza a busca }
if key = #13 then
  if Edit1.Text <> '' then
  begin
    if Assigned(FDataSet)then
    begin
      Buscar;
      DBGrid1.SetFocus;
      StatusBar1.SimpleText := Format(
        '%d registro(s) encontrado(s) com "%s"',
        [FDataSet.RecordCount, Edit1.Text]);
      Key := #0;
    end;
  end;
```

Listagem 2. Unit do componente

```
unit uLocalizar;

interface

uses Classes, Forms, DB, Dialogs, ufrLocalizar, Graphics, Controls,DBClient;

  type
   TLocalizar = class(TComponent)
   private
    FDataSet: TClientDataSet;
    FFormLocalizar: TfrLocalizar;
    procedure SetDataSet(const Value: TClientDataSet);
   protected
    procedure Notification(AComponent: TComponent;
      Operation: TOperation); override;
   public
     function Execute: Boolean;
     function ReturnValue: integer;
   published
    property DataSet: TClientDataSet read FDataSet write SetDataSet;
  end;

procedure Register;

implementation

procedure Register;
begin
  RegisterComponents('Localizar Component',
    [TLocalizar]);
end;
```

Listagem 3. Método Execute

```
if not FDataSet.Active then
  FDataSet.Active := True;
FFormLocalizar := TfrLocalizar.Create(nil);
try
  FFormLocalizar.DataSet := FDataSet;
  FFormLocalizar.ShowModal;
  Result := FFormLocalizar.ModalResult = mrOk;
finally
  FFormLocalizar.Free;
end;
```

Quando o usuário apertar ESC, poderíamos fechar o formulário, ou ainda, quando o usuário apertar *Enter* no *grid* o mesmo fechar como se fosse clicado o botão OK. Isso são funcionalidades simples, que deixarei a cargo do leitor, mas a versão para download possui essas funcionalidades.

Criando o componente

Agora, precisamos criar uma *unit*, que receberá mais propriedades que serão repassadas ao formulário no momento de sua criação em memória, assim como também fazer o registro do componente no Delphi. Crie uma nova *unit* e adicione o código da **Listagem 2**.

No código, criamos uma classe *TLocalizar*, descendente de *TComponent*. Na seção *private* temos duas variáveis, uma referente à propriedade publicada *DataSet* (seção *published*) que teremos e outra referente ao formulário criado neste artigo.

O *SetDataSet* configura a propriedade e sua implementação é bem simples, onde apenas precisamos configurar a propriedade *DataSet* e verificar se o formulário está criado, através do *FFormLocalizar*.

Na seção *protected*, precisamos implementar o *Notification*, que é responsável, por atribuir *nil* à propriedade *DataSet*, quando por exemplo o *ClientDataSet* associado a mesma, for excluído. Isso evita erros no IDE do Delphi. A sua implementação, usa o seguinte código:

```
inherited;
if Operation = opRemove then
  if AComponent = FDataSet then
    FDataSet := nil;
```

Continuando o código da **Listagem 2**, na seção *public* temos duas funções. A primeira, *Execute*, indicará se houve algum registro escolhido no formulário, assim podemos repassar para a consulta principal, o valor do registro. A segunda retorna o valor da chave primária do registro escolhido no *grid* do formulário, onde simplesmente, chama o *GetCodigo* da variável *FFormLocalizar*. Por fim, o *Register* (abaixo de *implemetation*), registra o componente na *Component Pallete* do Delphi, onde

passamos como parâmetro o nome a paleta e a classe do componente.

Chamando o formulário de pesquisa

O *Execute*, é o principal método do nosso componente. Ele se comporta semelhante ao *Execute* dos controles *Dialogs* do Delphi, ou seja, é retornado *true* se foi escolhido um registro na consulta. Na **Listagem 3**, temos a implementação do método.

O código é bastante simples, primeiro verificamos se o *ClientDataSet* vinculado ao componente está ativo, se não abrimos o mesmo. A seguir, instanciamos o formulário de pesquisa, repassamos a propriedade *DataSet* e chamamos o *ShowModal*.

O resultado do *Execute*, será de acordo com o *ModalResult* do formulário, ou seja, caso seja *mrOK* foi selecionado algum registro na consulta, senão, nenhum registro foi selecionado. Por isso, configuramos nos botões do formulário o *ModalResult*.

Por fim, no *finally*, liberamos o formulário da memória. Para finalizar, toda a criação do componente, no pacote, clique em *Compile* e depois em *Install*. Caso tudo tenha dado certo, você terá um componente instalado em uma nova paleta no Delphi (**Figura 3**).

Colocando um ícone no componente

Para adicionarmos um ícone ao componente, precisamos editar um arquivo DCR e adicionar no pacote. Vamos usar o *Image Editor* do Delphi, acessando o menu *Tools>Image Editor*. No editor, clique em *File>New>Component resource file (dcr)*.

Clique com o botão direito em *Contents* e escolha *New>Bitmap*. Escolha uma imagem de tamanho 24x24 (use *SuperVGA* nas opções de cores). Após criar, dê um duplo clique *Bitmap1* e crie ou cole uma imagem de sua preferência. Após adicionar a imagem, renomeie o arquivo *Bitmap1* para *TLOCALIZAR*. Feche o editor, salvando o arquivo na mesma pasta do pacote criado no Delphi. Agora no *source* do pacote (DPK), adicione o código que irá adicionar o arquivo:

```
{$R *.dcr}
```

Agora, basta recompilar a aplicação e seu componente terá um novo ícone (**Figura 4**).

Testando o componente

Agora, precisamos testar o componente. Crie uma conexão com um banco de dados, usando *dbExpress* ou qualquer outra suíte de componentes para acesso a dados. Nosso principal componente é o *ClientDataSet*.

Adicione um *SQLDataSet* fazendo a ligação ao *SQLConnection* que conecta ao banco. Adicione também um *DataSetProvider* e modifique para *True* a propriedade *Options>AllowCommandText*, assim indicamos que o responsável por receber as consultas (instruções SQL) é o próprio *ClientDataSet*.

Adicione dois *ClientDataSets* ao formulário (*cdsEmployee* e *cdsBuscaEmployee*). No *cdsEmployee* configure o *CommandText* para:

```
select * from EMPLOYEE where EMP_NO = :EMP_NO
```

Estamos selecionando todos os dados da tabela, mas parametrizando a mesma pelo código, assim nossa consulta retornará todos os registros, apenas quando indicarmos o valor do parâmetro.

No *cdsBuscaEmployee* digite na mesma propriedade:

```
select EMP_NO, FIRST_NAME from EMPLOYEE
where FIRST_NAME like :FIRST_NAME
```

Nesse controle, estamos retornando apenas o código e nome da tabela *Employee*, parametrizando pelo nome, que será configurado no nosso componente. Não esqueça de configurar os parâmetros nos dois *ClientDataSets*. Adicione um *Localizar* no formulário e configure *DataSet* para *cdsBuscaEmployee*. Adicione os campos do *cdsEmployee* no formulário para que o mesmo fique semelhante à

ClubeDelphi PLUS www.devmedia.com.br/clubedelphi/portal.asp

Acesse agora o mesmo o portal do assinante ClubeDelphi e assista a uma vídeo aula de Guinther Pauli que mostra como criar componentes.

*www.devmedia.com.br/articles/viewcomp.asp?comp=7506&hl=*cria%E7%E3o*%20and%20*de*%20and%20*componentes**

Figura 3. Componente instalado no Delphi

Figura 4. Novo ícone do componente

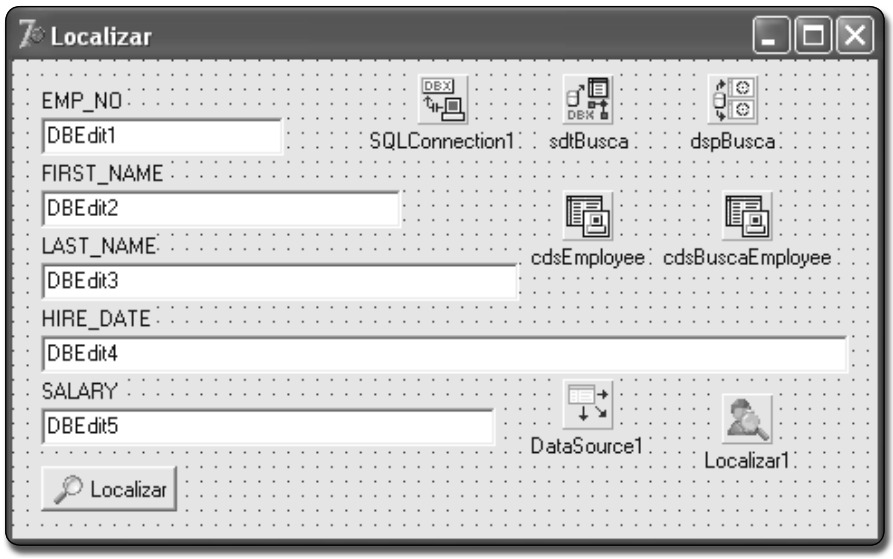

Figura 5. Formulário de cadastro para teste

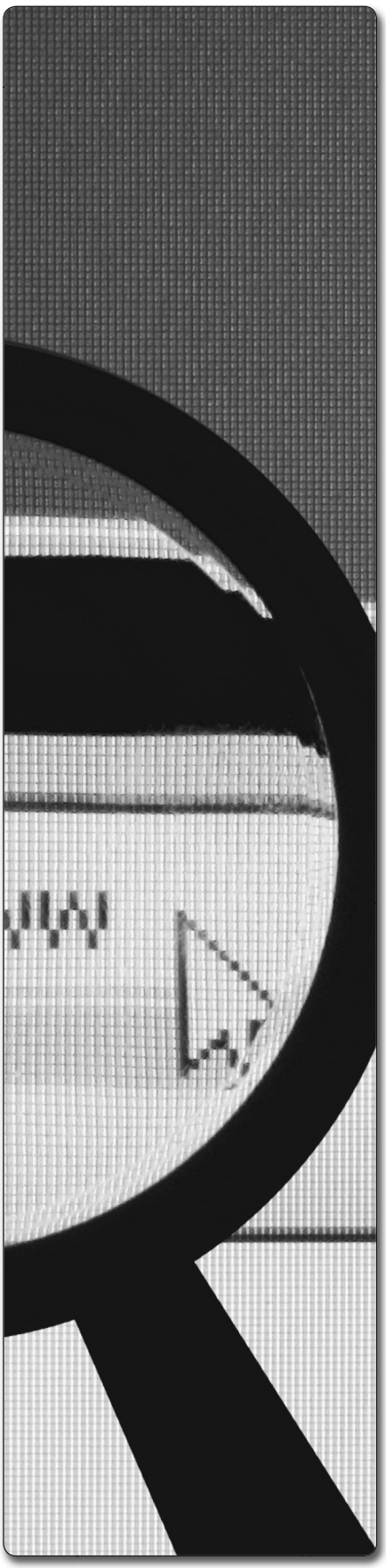

Figura 5. Adicione também um botão para chamar o *Localizar*.

Agora, simplesmente, precisamos usar o seguinte código no botão:

```
if Localizar1.Execute then
begin
  cdsEmployee.Close;
  cdsEmployee.Params[0].AsInteger :=
    Localizar1.ReturnValue;
  cdsEmployee.Open;
end;
```

Veja que o código é bem simples, onde verificamos se foi escolhido algum registro no formulário de consulta, usando o *Execute*. Após, simplesmente, configuramos o parâmetro da consulta com o valor retornado pelo componente (*ReturnValue*).

Com isso, temos um componente de pesquisa que pode ser reaproveitado em vários formulários usando o mesmo componente de dados. Veja o mesmo em execução na **Figura 6**.

Sugestões de melhorias

Imagine criar uma propriedade que seria uma coleção de parâmetros com *Caption*, campo da consulta etc. Ao adicionar um item na coleção, o mesmo seria vinculado a um item de um *RadioButton* no formulário, assim, para cada parâmetro teríamos um item em um componente para escolher e o campo onde faríamos a busca.

Teríamos assim, um componente de busca bastante customizável, criando consultas complexas com vários parâmetros. Também poderíamos melhorar o visual do formulário de consulta, trazendo facilidades ao usuário.

Caso implemente uma funcionalidade que queira divulgar para outros programadores, mande suas alterações que publicarei em meu site (*www.lucianopimenta.net*) e no da ClubeDelphi.

Conclusão

Vimos neste artigo como podemos criar componentes que nos auxiliem em aplicações client/Server para ser utilizado em consultas auxiliares, poupando tempo de programação e diminuição de código.

O que mostrei aqui foi apenas o pontapé inicial, use sua criatividade para agilizar seu dia a dia de programador. Um grande abraço a todos! ●

Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

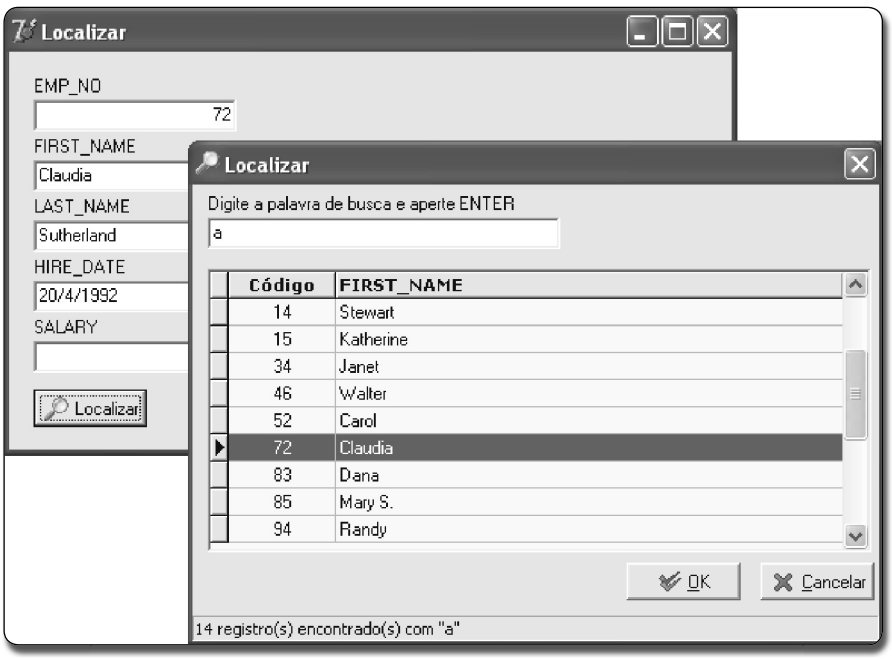

Figura 6. Testando o componente