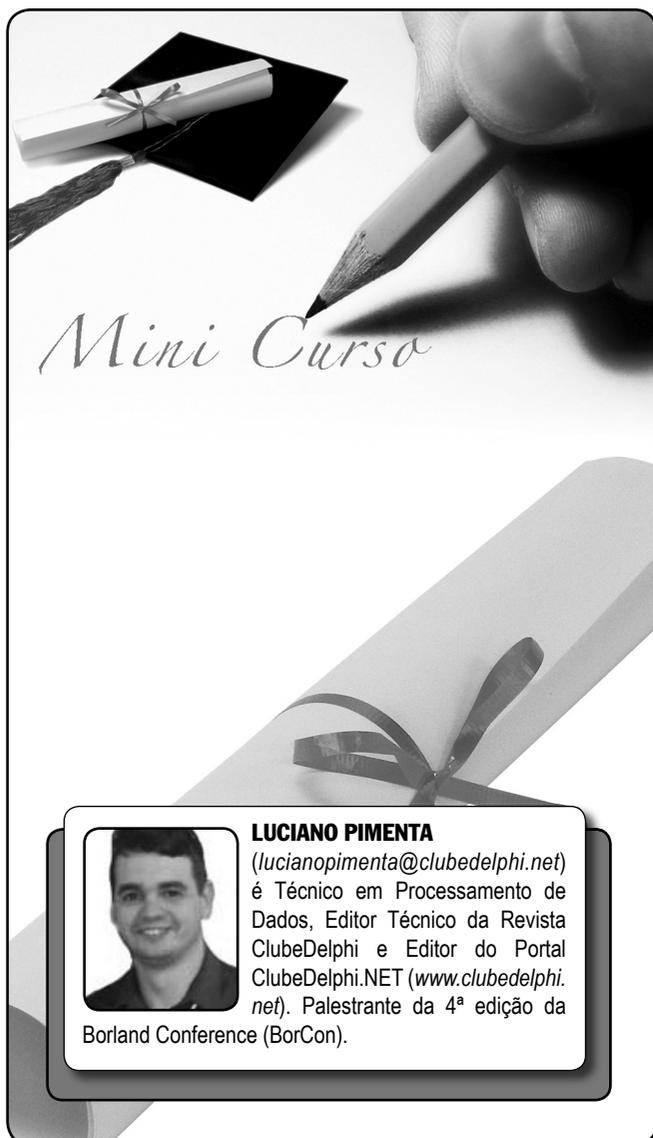


Aplicação Web Completa

Cadastros, consultas e customização de controles - Parte 2



LUCIANO PIMENTA

(lucianopimenta@clubedelphi.net)
é Técnico em Processamento de Dados, Editor Técnico da Revista ClubeDelphi e Editor do Portal ClubeDelphi.NET (www.clubedelphi.net). Palestrante da 4ª edição da Borland Conference (BorCon).

Começamos na edição anterior nosso minicurso de uma aplicação Web completa. Veremos neste artigo como criar os cadastros de produtos. Faremos uma pesquisa para o usuário localizar os produtos que deseja, além de várias configurações de componentes e boas práticas de programação Web.

Cadastro de produtos

Como estamos trabalhando em uma aplicação Web, como comentado no artigo anterior, temos sempre que primar pela performance da aplicação, retornando do banco de dados apenas a quantidade mínima de registros. Assim como em aplicações Desktop, no cadastro de produtos não teremos todos os registros da tabela e uma barra de navegação, mas sim apenas o registro escolhido pelo usuário na consulta que criaremos mais adiante.

Crie um novo *WebForm* (*File>New>Other>New ASPNET Files>ASPNET Pages*), renomeie-o para "cadastroprodutos.aspx" e adicione os dois *User Controls* criados na edição anterior. Na **Tabela 1** temos a nomenclatura, tipo e valores de propriedades alterados para os componentes que adicionaremos no formulário (com exceção dos *Labels*). Nosso formulário ficará semelhante ao da **Figura 1**.

Clique com o botão direito no *File Upload* e escolha a opção *Run As Server Control*, assim podemos referenciar o mesmo no código. No *Click* do botão *Cancelar* vamos apenas redirecionar para a página principal, com o seguinte código:

```
Response.Redirect ("Home.aspx");
```

Stored Procedure de inserção e atualização

Vamos agora criar uma *Stored Procedure* para a inserção dos dados no banco. Execute o código da **Listagem 1** no IBExpert ou outra ferramenta que esteja usando para manutenção do banco de dados da aplicação.

Componente	Nome	Propriedade/Valor
TextBox	txtProduto	
DropDownList	dpCategoria	
TextBox	txtPreco	
TextBox	txtDescricao	TextMode=MultiLine
HTML File Upload	File1	
Image	imgFigura	
Button	btnSalvar	
Button	btnCancelar	

Tabela 1. Componentes adicionados no formulário

Figura 1. Tela de cadastro de produtos

Listagem 1. Stored Procedure de inserção/atualização de produtos

```

CREATE PROCEDURE PRODUTOS_INS_UPD (
    ID_PRODUTO INTEGER,
    ID_CATEGORIA INTEGER,
    NOME_PRODUTO VARCHAR(50),
    PRECO NUMERIC(9,2),
    DESCRICAO VARCHAR(50),
    URL VARCHAR(50))
AS
BEGIN
    IF (EXISTS(SELECT ID_PRODUTO FROM PRODUTOS WHERE (
        ID_PRODUTO = :ID_PRODUTO))) THEN
        UPDATE PRODUTOS
        SET ID_CATEGORIA = :ID_CATEGORIA,
            NOME_PRODUTO = :NOME_PRODUTO,
            PRECO = :PRECO,
            DESCRICAO = :DESCRICAO,
            URL = :URL
        WHERE (ID_PRODUTO = :ID_PRODUTO);
    ELSE
        INSERT INTO PRODUTOS (
            ID_PRODUTO,
            ID_CATEGORIA,
            NOME_PRODUTO,
            PRECO,
            DESCRICAO,
            URL)
        VALUES (
            GEN_ID (GEN_CATEGORIA_ID, 1),
            :ID_CATEGORIA,
            :NOME_PRODUTO,
            :PRECO,
            :DESCRICAO,
            :URL);
    END

```

Não estranhe o código da listagem anterior, pois nossa SP possui duas funcionalidades: inserção ou atualização. A SP verifica se o valor do parâmetro ID_PRODUTO existe (*if*), e se a condição for verdadeira, então quer dizer que estamos atualizando os dados da tabela e executamos o comando *Update*. Caso a condição seja falsa, indica que estamos inserindo dados na tabela, assim executamos o comando necessário para inserção dos dados. Note o *Generator* nos parâmetros de inserção. Com isso, usamos apenas uma SP para realizar inserção ou atualização dos produtos, otimizando nosso código.

Nota: É altamente recomendado o uso de Stored Procedures quando se usa o ADO.NET e ASP.NET. Esse é um fator crítico em ambiente Web, para garantir a performance.

Voltando ao formulário de cadastro de produtos, adicione um *FbConnection* e um *FbCommand*. Faça acesso ao banco de dados da aplicação e configure a ligação dos componentes. Na propriedade *CommandText* adicione o seguinte código:

```
execute procedure PRODUTOS_INS_UPD (?, ?, ?, ?, ?, ?)
```

PENSE...

QUANTO TEMPO
VOCÊ GASTARIA
PARA DESENVOLVER
COBRANÇA COM BOLETOS
BANCÁRIOS PARA
APENAS UM BANCO
NO SEU SOFTWARE

COBREBEMX

-  56 BANCOS E MAIS DE 430 CARTEIRAS DE COBRANÇA PARA IMPRESSÃO E/OU ENVIO DE BOLETO BANCÁRIO POR EMAIL;
-  GERAÇÃO DE BOLETOS ON LINE;
-  GERAÇÃO E LEITURA DE ARQUIVOS (REMESSA/RETORNO) NOS PADRÕES FEBRABAN E CNAB;
-  MAIS DE 40 EXEMPLOS EM DIVERSAS LINGUAGENS DE PROGRAMAÇÃO

**cobre
bem**
Tecnologia

DOWNLOADS E INFORMAÇÕES EM WWW.COBREBEM.COM

Acesse a propriedade *Parameters* do componente e adicione seis parâmetros, referente a cada campo da tabela PRODUTOS. Para fins de padronização, dê o nome ao parâmetro com o mesmo nome da coluna (propriedades *ParameterName* e *SourceColumn*) e o tipo (propriedade *FbDbType*) seja o mesmo tipo da coluna cadastrada na tabela do banco.

No editor de códigos vamos criar dois métodos: um para salvar o registro e outro para salvar a figura do produto em uma pasta no servidor. Adicione os seguintes métodos na seção *public* da unit:

```
public
{ Public Declarations }
procedure SalvarRegistro;
function SalvarFiguraDisco: string;
```

Adicione o código para a implementação, conforme a **Listagem 2**.

Listagem 2. Métodos para salvar a figura em disco e o registro no banco

```
function TWebForm1.SalvarFiguraDisco: string;
var
  aFile, aDiretorio, aPath: string;
begin
  { Declare em uses System.IO }
  aFile := Path.GetFileName(
    file1.PostedFile.FileName);
  aDiretorio := 'http://localhost/CursoASPNET/images';
  aPath := aDiretorio + '/' + aFile;
  file1.PostedFile.SaveAs(Server.MapPath(
    '\images\' + aFile));
  imgFigura.ImageUrl := aPath;
  Result := aPath;
end;

procedure TWebForm1.SalvarRegistro;
begin
  FbConnection1.Open;
  try
    if file1.PostedFile.FileName <> '' then
      begin
        with FbCommand1 do
          begin
            if Request.QueryString[
              'CodProduto'] <> nil then
              Parameters[0].Value :=
                Request.QueryString['CodProduto']
            else
              Parameters[0].Value := 0Object(0);
              Parameters[1].Value :=
                dpCategoria.SelectedValue;
              Parameters[2].Value := txtProduto.Text;
              Parameters[3].Value := txtPreco.Text;
              Parameters[4].Value := txtDescricao.Text;
              Parameters[5].Value := SalvarFiguraDisco;
              ExecuteNonQuery;
              Response.Write('<script>Javascript:alert('+
                '''Cadastro efetuado com sucesso!''');'+
                '</script>');
            end;
          end
        else
          Response.Write('<script>Javascript:alert('+
            '''Escolha um arquivo para salvar o '+
            'registro!''');</script>');
        finally
          FbConnection1.Close;
        end;
      end;
    end;
```

A função *SalvarFiguraDisco* retorna o caminho da figura que estamos salvando em disco. Utilizamos essa técnica pois será mais fácil referenciar a figura nos demais módulos do site.

Nota: A pasta *images* foi criada na edição anterior.

O método *SalvarRegistros* repassa para os parâmetros do *FbCommand* os valores dos componentes. Para o campo chave da tabela, verificamos se a *QueryString CodProduto* está vazia, isso é necessário para saber se estamos trabalhando com edição ou inserção.

Vamos agora carregar os dados da tabela de categorias no *DropDownList*. Adicione um *FbCommand*, faça a ligação do *FbConnection* e na propriedade *CommandText* digite:

```
select ID_CATEGORIA, NOME_CATEGORIA
from CATEGORIA
```

Crie um novo método, chamado “CarregaCategoria” e implemente-o conforme o código da **Listagem 3**.

Listagem 3. Código para carregar o DropDownList de categorias

```
procedure TWebForm1.CarregaCategoria;
begin
  FbConnection1.Open;
  try
    dpCategoria.DataSource :=
      FbCommand2.ExecutesReader;
    dpCategoria.DataTextField := 'NOME_CATEGORIA';
    dpCategoria.DataValueField := 'ID_CATEGORIA';
    dpCategoria.DataBind;
  finally
    FbConnection1.Close;
  end;
end;
```

No evento *Load* da página adicione o seguinte código:

```
if not IsPostBack then
  CarregaCategoria;
```

Por fim, no *Click* do botão *Salvar*, basta chamar o método *SalvarRegistro*. Antes de testar, adicione diretamente no banco de dados alguns valores para a tabela *Categorias*. Caso deseje, faça uma página semelhante para o cadastro das categorias.

No design da página, clique com o botão direito do mouse e escolha a opção *View In Browser*. Após a compilação clique em OK e cadastre os produtos normalmente. Uma dica é adicionar as validações para o cadastro, semelhante ao que fizemos com o cadastro de usuários, na edição anterior. Veja na **Figura 2** a aplicação em execução.

Uma dica para o leitor implementar é a atualização do produto. Para isso, basta passar o código do produto como parâmetro na URL, selecionar os dados e exibir os mesmos nos controles, como por exemplo:

```
{ Evento Load da página }
if Request.QueryString['CodProduto'] <> nil then
  begin
    SeleccionaDados;
    ExibeControles;
  end;
```

E não precisamos alterar mais nada, pois no código que salva os dados, verificamos se o valor do *QueryString* está vazio, assim passamos o valor zero (na SP será feita a inclusão). Como preenchemos o *QueryString* com o código

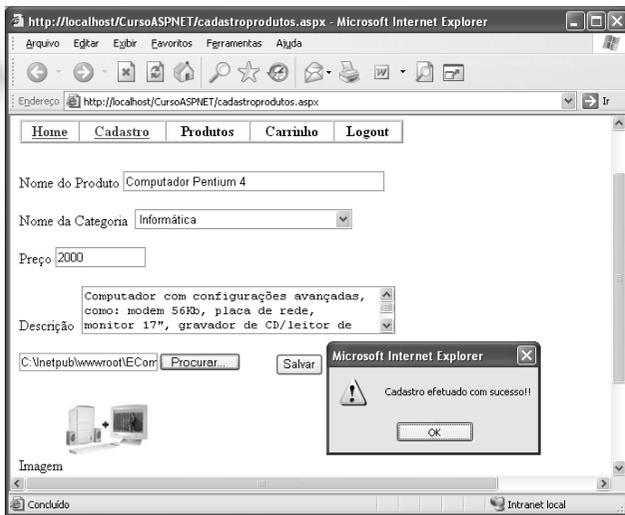


Figura 2. Cadastro de Produtos

do produto para atualização, o mesmo será usado no parâmetro do *FbCommand* (executando o comando *Update* da Stored Procedure no banco).

Consulta de produtos

Para visualizar os produtos do site, temos que ter uma consulta para o usuário escolher aquele que deseja obter mais informações e/ou comprar. Essa funcionalidade é bastante usada em sites de e-commerce. Crie um novo *WebForm* (*File > New > Other > New ASP.NET Files > ASP.NET Pages*), renomeie-o para "consultaprodutos.aspx" e adicione os dois *User Controls* criados na edição anterior.

Adicione um *TextBox* ("txtBusca"), um *Button* ("btnBusca") e um *DataGrid*. Adicione também os componentes de acesso a dados (*FbConnection* e *FbCommand*). Faça a ligação entre os componentes, formate o *DataGrid* através das opções de formatação (*Auto Format*). Seu formulário de busca deve estar semelhante a **Figura 3**.

Na propriedade *CommandText* do *FbCommand* digite o seguinte código:

```
select ID_PRODUTO, NOME_PRODUTO, URL
from PRODUTOS
where UPPER(NOME_PRODUTO) like ?
```

Adicione um parâmetro no *FbCommand*, semelhante ao que já fizemos no cadastro anterior e digite o código da **Listagem 4**, no *Click* do botão.

Listagem 4. Consulta de produtos no banco

```
FbConnection1.Open;
try
    FbCommand1.Parameters[0].Value := txtBusca.Text.ToUpper + '%';
    DataGrid1.DataSource := FbCommand1.ExecuteReader;
    DataGrid1.DataBind;
finally
    FbConnection1.Close;
end;
```

Note que chamamos *ToUpper* da propriedade *Text* do *TextBox* para que o valor seja repassado todo em maiúsculo.

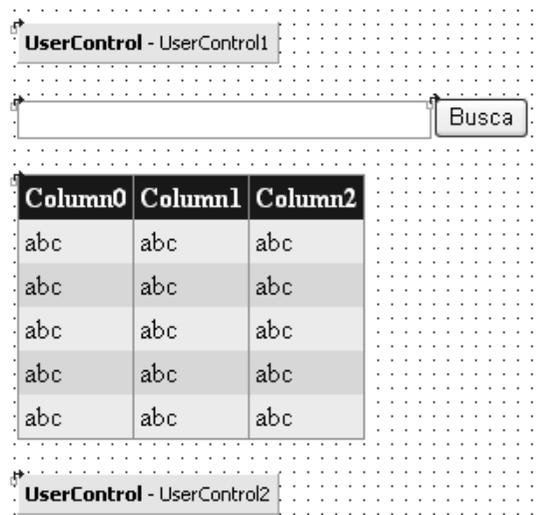


Figura 3. Formulário de busca da aplicação

Passamos o parâmetro para o *FbCommand* e executamos a consulta. Veja que colocamos o caractere coringa (%) apenas no final do parâmetro, assim se o usuário digitar "a", serão mostrados os produtos que começam com a referida letra.

Alguns podem querer que a pesquisa seja realizada por qualquer parte do nome do produto, adicionando assim o coringa também, antes do parâmetro. Esse tipo de pesquisa não traz boa performance para a aplicação, pois a quantidade de registros retornados pode ser muito grande.

Outra dica é solicitar ao usuário que digite uma quantidade mínima de caracteres para realizar a busca. Por exemplo, com o seguinte código, você pode forçar o usuário a digitar no mínimo três letras para realizar a busca:

```
if txtBusca.Text.Length > 3 then
    { Executa a consulta }
else
    { Aviso de quantidade mínima }
```

Para testar, abra a página usando a opção *View In Browser*.

Personalizando o DataGrid

Sem dúvida nenhuma o *DataGrid* é um dos componentes mais usados em aplicações Web, pois mostra os dados de maneira tabular, de forma rápida e simples. Podemos ter várias configurações do *DataGrid* e uma que faremos agora é mostrar uma figura.

Abra o editor do *DataGrid* através da propriedade *Columns* e desmarque a opção de gerar automaticamente as colunas (*Create columns automatically at run time*). Adicione respectivamente, um *Hyperlink Column* e dois *Bound Column*.

No *Hyperlink* configure as seguintes propriedades: em *Header Text* digite "Código" e em *Text Field* digite "ID_PRODUTO". Para as outras duas colunas configure *Header Text* como "Nome do Produto" e "Figura" e *Data Field* com os campos *NOME_PRODUTO* e *URL*.

Na propriedade *Data Formatting string* da coluna *URL*, adicione o seguinte valor: "". Assim, no momento da renderização do controle, será adicionado no

HTML final, a tag *img src* e o valor do campo URL, que é o local onde esta salvo as imagens.

Mostraremos assim, no *DataGrid*, imagens que armazenamos no disco (**Figura 4**), poupando espaço no banco e codificação para a exibição da figura, sem falar na performance que é muito maior.

Vale ressaltar que o tamanho da imagem, conta muito, pois imagens grandes podem distorcer e deixar o site com um layout estranho. Por isso, vale a dica de padronizar o tamanho das imagens, claro.

Quando o usuário realizar a busca, temos que dar a opção do mesmo abrir as informações do produto que ele deseja comprar. Assim na coluna *ID_PRODUTO*, vamos colocar um valor para redirecionarmos o usuário para uma página com opção de compra do produto.

Na propriedade *URL field* digite "ID_PRODUTO" e em *URL format string*, digite: "visualisaproduto.aspx?CodProduto={0}". A página *visualisaproduto.aspx*, ainda não está criada, faremos isso agora.

Comprando o produto

Crie um novo *WebForm*, conforme a técnica anterior, dando o nome de "visualisaproduto.aspx". Mostraremos as informações do produto, dando a possibilidade de compra do mesmo. Nessa página é onde podemos colocar todas as informações/detalhes do produto, para que o usuário saiba suas características.

Uma dica é adicionar comentários nessa página, sobre o produto de quem já realizou a compra do mesmo. Adicione os *User Controls* no formulário.

Nota: Como já realizamos o cadastro do produto, essa página destina-se apenas a exibir as informações, então não precisamos usar *TextBox* para todos os campos, por exemplo, e tão pouco um *DropDownList*, pois não alteraremos a categoria do produto.

Veja na **Figura 5**, como ficaram os componentes no formulário (perceba o nome dos mesmos).

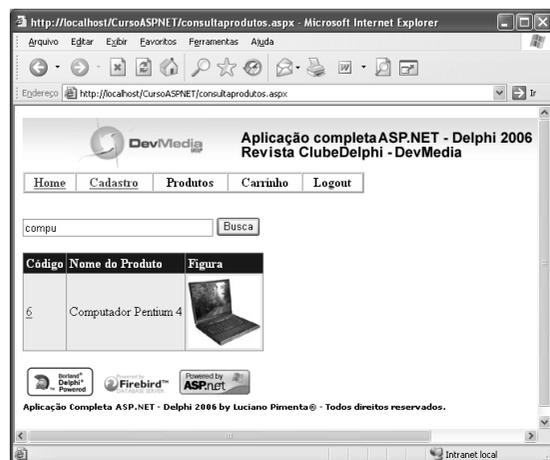


Figura 4. Mostrando imagens no DataGrid

Temos um *ImageButton* que possui a figura, mostrando ao usuário onde clicar para que o mesmo possa adicionar ao carrinho de compras o produto. No *TextBox* altere a propriedade *ReadOnly* para *True* e *TextMode* para *MultiLine*.

Adicione os componentes de acesso a dados e vamos usar a consulta que está na **Listagem 5**, que deve ser adicionada na propriedade *CommandText* do *FbCommand*:

Listagem 5. Consulta para mostrar os dados do produto

```
select PRODUTOS.ID_PRODUTO, PRODUTOS.NOME_PRODUTO,
       CATEGORIA.NOME_CATEGORIA, PRODUTOS.PRECÓ,
       PRODUTOS.DESCRICAO, PRODUTOS.URL
from PRODUTOS
inner join CATEGORIA on (PRODUTOS.ID_CATEGORIA =
                        CATEGORIA.ID_CATEGORIA)
where PRODUTOS.ID_PRODUTO=?
```

Veja que na consulta estamos trazendo a categoria do produto e parametrizando a mesma pelo código do produto. Adicione o parâmetro no *FbCommand* (com o mesmo nome do campo). Adicione um método chamado "Select" no código e implemente-o conforme a **Listagem 6**.

Listagem 6. Código que mostra os dados da consulta na tela

```
procedure TWebForm1.Select;
var
  dr: FbDataReader;
begin
  if Request.QueryString['CodProduto'] <> nil then
  begin
    FbConnection1.Open;
    try
      FbCommand1.Parameters[0].Value :=
        Request.QueryString['CodProduto'];
      dr := FbCommand1.ExecuteReader;
      if dr.HasRows then
      begin
        dr.Read;
        lblProduto.Text := dr['NOME_PRODUTO'].ToString;
        lblCategoria.Text := dr['NOME_CATEGORIA'].ToString;
        lblPreco.Text := System.String.Format('{0:c}', dr['PRECÓ']);
        txtDescricao.Text := dr['DESCRICAO'].ToString;
        Image1.ImageUrl := dr['URL'].ToString;
      end;
    finally
      FbConnection1.Close;
    end;
  end;
end;
```

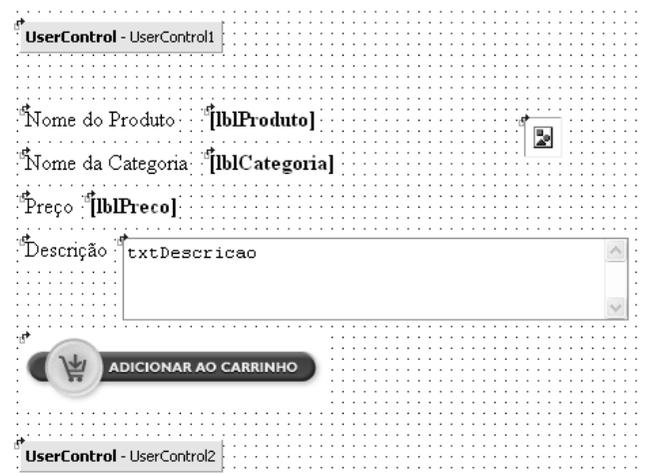


Figura 5. Componentes da página para visualizar os dados do produto e opção de compra

No código anterior, temos algo diferente: *System.&String.Format("{0:c}', dr[PRECO']*). Esse código formata o valor do campo PRECO para o formato moeda. Com esse código podemos fazer várias formatações com data/hora, casas decimais etc. Para finalizar, no *Load*, digite:

```
if not IsPostBack then
    Select;
```

Abra o *header.ascx* e no componente *Hyperlink* chamado *Produtos*, adicione na propriedade *NavigateURL*, o valor: "consultaprodutos.aspx". Rode a aplicação, abra a consulta de produtos. Escolha o produto no link do *DataGrid* e visualize os dados do produto na página recém criada.

“Data Modules” no ASP.NET

Vamos implementar na página principal da aplicação

uma filtragem de produtos, para que o usuário possa navegar no site através das categorias do mesmo. Assim, precisamos abrir o *Home.aspx* e adicionar um *ListBox* (“1stCategoria”) na página.

Temos que preencher o *ListBox* com o nome das categorias cadastradas no banco, semelhante ao que fizemos com o *DropDownList* do cadastro de produtos. Mas será que não podemos aproveitar essa consulta? Sim, basta organizarmos o código em *units*.

Estamos acostumados a trabalhar com *DataModules* em aplicações Win32, onde podemos concentrar os componentes de acesso a dados e regras de negócios. Mas posso fazer isso em ASP.NET? Sim, podemos “simular” um *DataModule* no ASP.NET. Crie uma nova classe (*File>New>Other>New Files>Class*), salve e dê o nome de “uDM.pas”. O código da

Impressão Rápida em Matriciais...

RDprint 4.0

O mais completo componente para impressão em MATRICIAIS !
LIDERANÇA absoluta na sua categoria !

Ideal para Notas Fiscais, Duplicatas, Boletos Bancários, etiquetas e relatórios em geral.

- Opção para impressão colorida
- Ajustes de margens para impressão gráfica
- Opção para ocultar a barra de progresso
- Variáveis PAGINAS, DATA, HORA e TÍTULO

Novo form de SETUP com :

- Mapeamento das impressoras e Modelos
- Seleção de páginas igual ao word (1-5,7,8)
- Opção para Inverter e Agrupar cópias na impressão

Novo form de PREVIEW com:

- Função para Procura de TEXTO no relatório
- ROLAGEM com salto automático de página
- ARRASTO da imagem do preview
- StatusBar com informações da impressão
- Novos ícones personalizados

* Disponível para Delphi 5, 6, 7, 2005 e 2006 (VCL)
* Compatível com todas as versões do Windows
* Imprime em portas LPT / COM e USB (modo gráfico)

RDprint Setup

Configuração da Impressão

Impressora: HP DeskJet 870Cxi Propriedades...

Modelo: Gráfico - Compatível com Windows Visualizar

Intervalo de Páginas:
 Todas
 Página Atual
 Páginas: 1-5,7
 Imprimir: Todas as páginas do intervalo

Cópias:
 Número de Cópias: 3
 Agrupar
 Ordem inversa

Digite a páginas e/ou intervalos separados por vírgula. Por exemplo: 1,3,5-12

Ok Cancel

Nova
Versão!

Deltress

Fone/Fax (14) 3454-7880
www.deltress.com.br

Página: 2 de 19
87%
Impressora: HP DeskJet 870Cxi
Gráfico
* O RDprint 4.0 não imprime gráficos !

classe deve ficar, semelhante a **Listagem 7**.

Listagem 7. Simulando um DataModule em ASP.NET

```
unit uDM;

interface

uses System.ComponentModel;

type
  TDM = class (System.ComponentModel.Component)
  private
    { Private Declarations }
  public
    constructor Create;
  end;

implementation

constructor TDM.Create;
begin
  inherited Create;
  // TODO: Add any constructor code here
end;
end.
```

Feche a classe e abra-a novamente. Note que agora temos um container onde podemos colocar nossos componentes de acesso e dados e na classe trabalhar normalmente com código para as regras de negócio.

Adicione um *FbConnection* no "DataModule" e faça a configuração com o banco. Veja que o Delphi cria o *InitializeComponent* automaticamente, assim precisamos apenas chamar o *InitializeComponent* dentro do *Create* da classe.

Na seção *public*, crie uma função que retornará um *FbDataReader*, chamada "GetCategorias" e implemente-a conforme a **Listagem 8**.

Listagem 8. Função para retornar as Categorias cadastradas no banco

```
function TDM.GetCategorias: FbDataReader;
var
  dr: FbDataReader;
  cmd: FbCommand;
begin
  { Declare no uses System.Data }
  FbConnection1.Open;
  cmd := FbCommand.Create('select * from CATEGORIA', FbConnection1);
  dr := cmd.ExecuteReader(CommandBehavior.CloseConnection);
  Result := dr;
end;
```

O código da listagem anterior cria uma instância de *FbCommand* e passa como parâmetro a instrução SQL que usaremos, juntamente com a conexão do banco (*FbConnection1*). Após chamamos o *ExecuteReader* passando como parâmetro o *CommandBehavior.CloseConnection* que ficará encarregado de liberar a conexão com o banco. Assim não precisamos fechar a conexão nesse código, ela será fechada quando chamarmos o método *Close* do *DataReader*.

Nota: Para saber mais sobre as opções do *CommandBehavior*, dê uma olhada na documentação do .NET Framework.

Para testar, abra o cadastro de produtos e altere o código do *CarregaCategoria* para o código da **Listagem 9**.

Listagem 9. Alterando o código que retorna as categorias

```
var
  DM: TDM;
  dr: FbDataReader;
begin
  { Declare em uses uDM e FirebirdSql.Data.Firebird }
  DM := TDM.Create;
  dr := DM.GetCategorias;
  dpCategoria.DataSource := dr;
  ...
  dr.Close;
end;
```

No *Home.aspx*, vamos implementar um código igual ao *CarregaCategoria* do cadastro de produtos, apenas alterando o nome do componente (que nesse caso é *IstCategoria*). Não esqueça de fazer a chamada ao método no *Page_Load* do formulário. Para finalizar, adicione um *DataGrid* e implemente o código da **Listagem 10** no "DataModule".

Listagem 10. Filtra os produtos de acordo com a categoria

```
function TDM.GetProdutos(
  aIdCategoria: string): FbDataReader;
var
  dr: FbDataReader;
  cmd: FbCommand;
begin
  FbConnection1.Open;
  cmd := FbCommand.Create(
    'select ID_PRODUTO, NOME_PRODUTO, URL '+
    'from PRODUTOS where ID_CATEGORIA=?',
    FbConnection1);
  cmd.Parameters.Add('ID_CATEGORIA',
    FbDbType.Integer, 0,
    'ID_CATEGORIA').Value := aIdCategoria;
  dr := cmd.ExecuteReader(
    CommandBehavior.CloseConnection);
  Result := dr;
end;
```

A diferença desse código fica por conta que estamos criando um parâmetro em tempo de execução. Volte a página *Home.aspx* e altere a propriedade *AutoPostBack* do *IstCategoria* para *True*.

Isso indica que ao clicarmos no item do *IstCategoria*, o evento *SelectedIndexChanged* será disparado. No referido evento adicione o código da **Listagem 11**.

Listagem 11. Código para filtrar os dados no DataGrid ao clicar no IstCategoria

```
var
  DM: TDM;
  dr: FbDataReader;
begin
  DM := TDM.Create;
  dr := DM.GetProdutos(IstCategoria.SelectedValue);
  DataGrid1.DataSource := dr;
  DataGrid1.DataBind;
  dr.Close;
end;
```

Para finalizar, faça a mesma técnica de formatação do *DataGrid*, utilizada no formulário de pesquisa, para mostrar a figura e também redirecionar o usuário para a página de compra do produto, ao clicar no *DataGrid*.

Uma dica é copiar o *DataGrid* da página de consulta e colar na *Home.aspx*, apenas alterando sua formatação (*AutoFormat*), caso seja necessário. Veja na **Figura 6** a aplicação em execução.



Figura 6. Filtrando os produtos por categoria

Mensagens no ASP.NET

Estamos acostumados a usar mensagens de alerta e confirmação em aplicações Win32. Para trabalhar com esse tipo de mensagens no ASP.NET devemos usar JavaScript, como fizemos anteriormente. Mas e se tivéssemos um componente que nos tirasse esse trabalho?

Pois bem, Rodrigo Glauser, criou um componente *free*, onde podemos usar em nossa aplicação, que encapsula *script client side*, interagindo com a aplicação ASP.NET. Para saber como instalar o componente e baixar o mesmo, veja uma vídeo aula que criei no seguinte link: www.devmedia.com.br/visualizacomponente.aspx?comp=1405&site=3.

Nota: Junto ao download dos arquivos do artigo, encontra-se o componente.

A instalação é rápida e simples. Mostrarei aqui, como adaptar os exemplos mostrados até agora, utilizando o componente. Por exemplo, nos cadastros, podemos adicionar o componente no formulário e alterar o código (de cada cadastro).

Como também no formulário de consulta, onde mostramos uma mensagem ao usuário que digitar menos de três caracteres na busca, conforme a **Listagem 12**.

Listagem 12. Alterando o código para utilizar o componente de mensagens

```
Cadastro de Usuários e Produtos
...
ExecuteNonQuery;
MessageBox1.ShowMessage (
  'Cadastro efetuado com sucesso!');
end;

finally
  FbConnection1.Close;
end;

except
  MessageBox1.ShowMessage ('Erro!');
end;

Consulta de Produtos
...
DataGrid1.DataBind;
end

else
  MessageBox1.ShowMessage ('Digite mais de 3 digitos!');
```

Usamos apenas mensagens de alerta, mas podemos usar mensagens de confirmação, onde o usuário escolherá opções (Sim ou Não, por exemplo), que veremos como funciona nos próximos artigos e exemplos.

Conclusão

Nessa segunda parte do curso, realizamos o cadastro e consulta de produtos. Configuramos o *DataGrid* para mostrar figuras e que usasse um link para passar como parâmetro o código do produto e redirecionar para a página de compra. Criamos na página principal, uma opção de filtragem dos produtos por categoria.

Vimos como simular um *DataModule* em aplicações ASP.NET, seguindo o mesmo padrão que estamos acostumados a trabalhar na plataforma Win32. Por fim, mostramos um ótimo componente para exibir caixas de mensagens e confirmação, sem a utilização direta de JavaScript (o mesmo está encapsulado no componente).

No próximo artigo, faremos uma das partes mais importantes do nosso mini-curso, o carrinho de compras. Um grande abraço a todos e até lá! ■

+ de 80.000 membros cadastrados
+ de 15.000 exemplos com fontes
+ de 900 apostilas
+ de 4.000 dicas
Fórum Delphi
Artigos

TOTALMENTE GRÁTIS

www.delphi.eti.br

Um dos maiores sites de apoio a desenvolvedores Delphi do Brasil!!!