

Aplicação Web Completa

Carrinho de Compras - Parte 3



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).

Continuaremos neste artigo a criação da nossa aplicação Web completa, onde mostraremos como criar um carrinho de compras para o site. Para este artigo, veremos algumas funcionalidades de um dos componentes mais importante da arquitetura ADO.NET, o *DataSet*.

O *DataSet* trabalha totalmente desconectado do banco, com os dados em memória (cache), sendo de grande utilidade para criarmos o carrinho de compras, pois vamos adicionar os produtos comprados pelo cliente e no final incluiremos os mesmos no banco, sem a necessidade de a cada produto inserido, ser feita uma conexão com o banco.

Criando o carrinho de compras

Abra o projeto que estamos trabalhando, crie um novo *Web Form* (*File>New>Other>New ASP.NET Files>ASP.NET Page*) e dê o nome de "carrinho.aspx" para o mesmo. Adicione um *DataGrid* e configure-o para o estilo que mais lhe convier.

Adicione no formulário um *DataSet* (da categoria *Data Components*) e dê o nome de "dsCarrinho". Abra a propriedade *Tables*, no editor clique em *Add*, para adicionar uma "tabela" e em *TableName* digite "CARRINHO". Clique na propriedade *Columns* para adicionar as colunas da tabela, conforme a **Tabela 1**.

Nome da Coluna	Tipo (DataType)
ID_PRODUTO	System.Int32
NOME_PRODUTO	System.String
PRECO	System.Single
QUANTIDADE	System.Int32
SUBTOTAL	System.Single
TOTAL	System.Single

Tabela 1. Colunas da tabela CARRINHO do DataSet

Depois de criar as colunas, volte ao editor de *DataTables* do *DataSet* e acesse a propriedade *PrimaryKey*, marcando o campo *ID_PRODUTO* conforme visto na **Figura 1**.

Feche o editor. A configuração do *DataSet* está pronta, precisamos agora criar métodos e funções do nosso carrinho de compras.

Codificando o carrinho

Abra o editor e adicione o seguinte código na seção *private*:

```
procedure ConsultaProduto;
procedure AddProdutoCarrinho (IdProduto, NomeProduto,
  Preco, Quant: string);
procedure MostraGrid;
function GetDataSet: DataSet;
```

Aperte **CRTL+SHIFT+C** para que o Delphi crie o cabeçalho dos mesmos. A função *GetDataSet* simplesmente colocará nosso *DataSet* em sessão, com o seguinte código:

```
if (Session['CARRINHO'] = nil) then
  Session['CARRINHO'] := dsCarrinho;
Result := Session['CARRINHO'] as DataSet;
```

Se não colocássemos o *DataSet* em sessão, a cada nova requisição ele teria seus valores perdidos, ficando apenas o último registro adicionado. Este é um erro bastante comum nos desenvolvedores que estão iniciando a criação de aplicações ASP.NET com *DataSets*, lembre-se sempre de colocá-lo em sessão.

Repassaremos as informações do produto e quantidade através da URL da aplicação, com *QueryString*, como fizemos na edição anterior. O método responsável por selecionar as informações (dados) do produto é o *ConsultaProduto*.

Adicione um *FbConnection* no formulário e configure a conexão com o banco de dados. Adicione também um *FbCommand* ("cmdSelect"), faça a ligação ao *FbConnection* e adicione na propriedade *CommandText*, o seguinte código:

```
select ID_PRODUTO, NOME_PRODUTO, PRECO
from PRODUTOS
where ID_PRODUTO=?
```

Configure o parâmetro da consulta (dê o nome de "ID_PRODUTO") na propriedade *Parameters* do *cmdSelect*. No método *ConsultaProduto* adicione o código da **Listagem 1**.

Listagem 1. Método para consulta dos dados do Produto

```
var
  dr: FbDataReader;
begin
  FbConnection1.Open;
  try
    cmdSelect.Parameters[0].Value := Request.QueryString['CodProduto'];
    dr := cmdSelect.ExecuteReader;
    dr.Read;
    AddProdutoCarrinho(dr['ID_PRODUTO'].ToString,
      dr['NOME_PRODUTO'].ToString,
      dr['PRECO'].ToString,
      Request.QueryString['Quant'].ToString);
  finally
    FbConnection1.Close;
  end;
end;
```

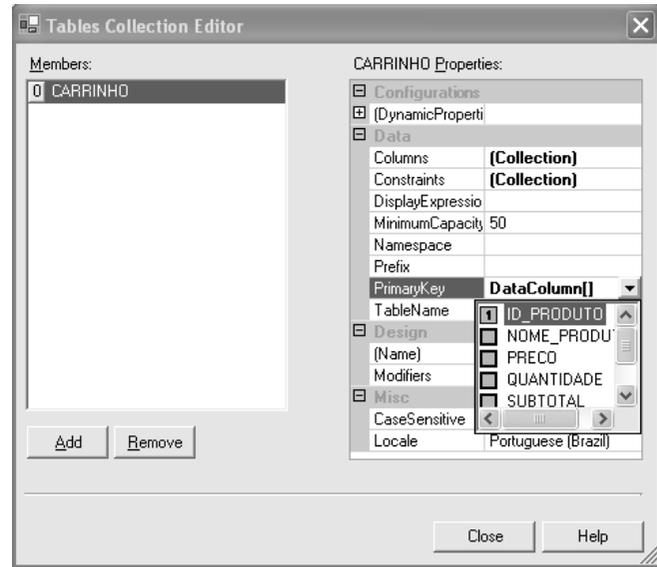


Figura 1. Escolhendo a chave primária da tabela

O código da listagem anterior faz a conexão com o banco, passa para o parâmetro da consulta o valor contido na *QueryString* e chama o *ExecuteReader*. Após, chamamos o método *AddCarrinho* passando os dados da consulta do produto. Esse método adiciona os valores dos parâmetros no *dsCarrinho*, conforme o código da **Listagem 2**.

Listagem 2. Método AddCarrinho

```
var
  dt: DataTable;
  row: DataRow;
begin
  dt := GetDataSet.Tables[0];
  row := dt.NewRow;
  row['ID_PRODUTO'] := IdProduto;
  row['NOME_PRODUTO'] := NomeProduto;
  row['PRECO'] := Preco;
  row['QUANTIDADE'] := Quant;
  dt.Rows.Add(row);
end;
```

O código adiciona uma nova linha no *dsCarrinho*. Note que não fizemos referência ao nome do *DataSet*, mas sim à função *GetDataSet* para que pegue o *DataSet* que está em sessão. No *MostraGrid* adicione o seguinte código:

```
DataGrid1.DataSource := GetDataSet;
DataGrid1.DataKeyField := 'ID_PRODUTO';
DataGrid1.DataBind;
```

Nesse código, apenas indicamos o *DataSource* do *DataGrid* e chamamos o *DataBind*. Vá até a página *visualizaproduto.aspx* e adicione um *TextBox* ("txtQuantidade") ao lado do botão *Adicionar ao Carrinho*. Coloque um *Label* para indicar que deve ser digitada a quantidade de produtos que deseja comprar.

Na propriedade *Text* do *txtQuantidade* configure "1", assim não precisamos validar se o usuário digitou um valor maior que zero (veja o box *Validações com JavaScript*).

Validações com JavaScript

Para validar que somente números sejam digitados em um TextBox, podemos usar os controles de validação vistos anteriormente neste curso. Aproveitarei para mostrar uma forma alternativa, a fim de demonstrar como usar JavaScript em aplicações ASP.NET. No *Load* da página *visualizaproduto.aspx* adicione o seguinte código:

```
txtQuantidade.Attributes.Add('onkeypress',
    'if (event.keyCode < 48 || event.keyCode > 57) '+
    '{event.keyCode = 0;}');
```

Com esse código o *txtQuantidade* aceitará somente números.

No evento *Click* do *ImageButton* altere o código já adicionado, para o seguinte valor:

```
Response.Redirect('carrinho.aspx?CodProduto='+
    Request.QueryString['CodProduto'] +
    '&Quant='+txtQuantidade.Text);
```

Por fim, precisamos codificar o *Load* da página do carrinho para fazer a consulta do produto, adicionar o mesmo no *DataSet* e mostrar no *DataGrid*. Para isso, adicione o código da **Listagem 3** no *Page_Load*:

Listagem 3. Page_Load do carrinho.aspx

```
if not IsPostBack then
begin
    if Request.QueryString['CodProduto'] <> nil then
        ConsultaProduto;
    MostraGrid;
end;
```

Rode a aplicação e teste.

Melhorias na aplicação

Precisamos adicionar mais algumas validações/melhorias no nosso carrinho de compras, por exemplo, verificar antes de adicionar, se o produto não existe no *DataSet*, se existir, apenas precisamos alterar a quantidade do mesmo. Volte ao carrinho e vamos modificar o código do *AddCarrinho*, conforme a **Listagem 4**.

Listagem 4. Alterando o método AddCarrinho

```
var
...
    qtd: System.Int32;
begin
    dt := GetDataSet.Tables[0];
    dt.Rows.Find(IdProduto);
    if row = nil then
    begin
        row := dt.NewRow;
        row['ID_PRODUTO'] := IdProduto;
        row['NOME_PRODUTO'] := NomeProduto;
        row['PRECO'] := Preco;
        row['QUANTIDADE'] := Quant;
        dt.Rows.Add(row);
    end
    else
    begin
        qtd := row['QUANTIDADE'] as Int32;
        qtd := qtd + Convert.ToInt32(Quant);
        row['QUANTIDADE'] := System.&Object(qtd);
    end;
    dt.Columns['SUBTOTAL'].Expression := 'QUANTIDADE * PRECO';
    dt.Columns['TOTAL'].Expression := 'SUM(SUBTOTAL)';
end;
```

Veja que estamos verificando (*Find*) se o código do produto já existe. Caso seja verdadeiro, a variável (*row*) será diferente de *nil*, portanto, apenas alteramos o campo QUANTIDADE. Senão, adicionamos o registro no *DataSet*. No final, através da propriedade *Expression*, passamos o código para calcular os campos SUBTOTAL e TOTAL. Veja na **Figura 2** o carrinho de compras em execução.

Vamos adicionar o valor do campo TOTAL no rodapé do *DataGrid*, para isso, acesse o evento *ItemDataBound* do *DataGrid* e adicione o seguinte código:

```
if e.Item.ItemType = ListItemType.Footer then
e.Item.Cells[3].Text := 'Total: ' +
    GetDataSet.Tables[0].Rows[0]['TOTAL'].ToString;
```

Altere a propriedade *ShowFooter* do *DataGrid* para *True*. Adicione manualmente os campos no *DataGrid*, removendo os campos ID_PRODUTO e TOTAL. Faça também a configuração do campo PRECO para que mostre o valor no formato moeda (na propriedade *Data formatting expression*, digite "{0:c}"). Veja o resultado na **Figura 3**.

Para adicionar o formato de moeda no total do rodapé, basta alterar o *ItemDataBound* com o seguinte código:

```
e.Item.Cells[3].Text := 'Total: ' +
    System.&String.Format('{0:c}',
    GetDataSet.Tables[0].Rows[0]['TOTAL']);
```

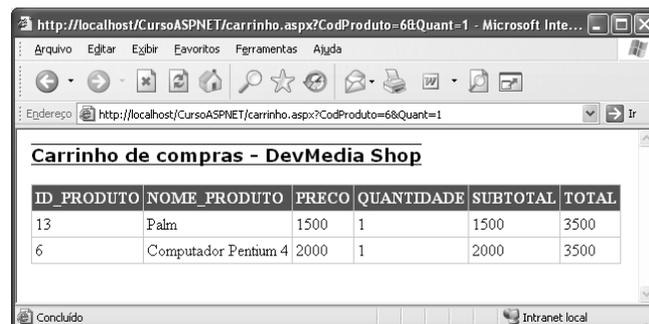


Figura 2. Carrinho de compras em uso

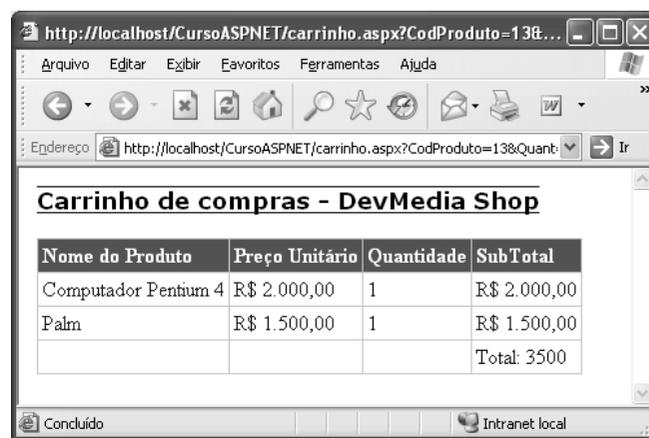


Figura 3. DataGrid mostrando o total do carrinho de compras no rodapé

Removendo produtos do carrinho

Vamos incluir a capacidade de remover um produto do carrinho. Abra o editor do *DataGrid* e adicione um botão do tipo *Button Column>Delete*. No evento *DeleteCommand* do *DataGrid*, adicione o código da **Listagem 5**.

Listagem 5. Removendo um produto do carrinho

```
var
  dt: DataTable;
  row: DataRow;
begin
  dt := GetDataSet.Tables[0];
  row := dt.Rows.Find(DataGrid1.DataKeys[
    e.Item.ItemIndex]);
  if row <> nil then
    dt.Rows.Remove(row);
  MostraGrid;
end;
```

A propriedade *DataKeys* do *DataGrid* armazena o valor do campo indicado na propriedade *DataKeyField*, que foi configurado no método *MostraGrid*. Assim, pesquisamos o valor do *ID_PRODUTO* e se for encontrado, removemos o mesmo do *DataSet*. Por fim, basta chamar novamente o *MostraGrid* para atualizar o carrinho.

Segurança

Nosso site de e-commerce não possui ainda qualquer validação para indicar qual usuário efetuou a compra. Vamos criar agora a tela de login da aplicação. Adicione um novo *Web Form* e dê o nome de "login.aspx". Adicione dois *TextBoxes* ("txtUser" e "txtSenha"), dois *Labels* e um botão. No *TextBox* de senha, altere a propriedade *TextMode* para *Password*. Veja na **Figura 4** como deve ficar o formulário.

Adicione também um *MessageBox*, um *FbConnection* e um *FbCommand*. Faça a ligação dos componentes de acesso a dados e utilize a seguinte instrução SQL no *FbCommand*:

```
select ID_USUARIO, SENHA
from USUARIOS
where UPPER(NOME_USUARIO) = ?
```

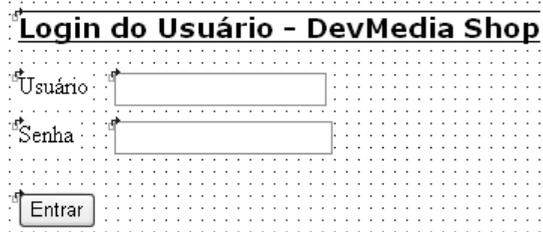


Figura 4. Criando a tela de login

Configure o parâmetro no *FbCommand* com o nome de "USUARIO". No botão de login, adicione o código da **Listagem 6**.

Listagem 6. Código para login da aplicação

```
var
  dr: FbDataReader;
begin
  FbConnection1.Open;
  try
    FbCommand1.Parameters[0].Value := txtUsuario.Text.ToUpper;
    dr := FbCommand1.ExecuteReader;
    dr.Read;
    if (dr['SENHA'].ToString.Trim = txtSenha.Text) then
      begin
        Session['ID_USUARIO'] := dr['ID_USUARIO'];
        Response.Redirect(Session['URL'].ToString);
      end
    else
      MessageBox1.ShowMessage(
        'Usuário ou senha inválidos!');
  finally
    FbConnection1.Close;
  end;
end;
```

Note que se a senha conferir, salvamos em uma variável de sessão o código do usuário e redirecionamos para a página do carrinho, usando outra variável de sessão, que vai armazenar (veremos mais adiante na página *visualizaproduto.aspx*) a URL correta para inserção do produto no carrinho.

Normalmente, o usuário de um site e-commerce realiza o login ou faz um cadastro somente quando for finalizar a compra, ou no nosso exemplo, quando adicionar o primeiro item no carrinho. Dessa forma o usuário não precisa realizar o login para visualizar os produtos.

Mas, e se o usuário digitar diretamente no browser o caminho da página *carrinho.aspx*? Simplesmente, vamos verificar se o mesmo está logado, sabendo o valor da variável de sessão *ID_USUARIO*. No *Page_Load* do formulário do carrinho adicione o seguinte código:

```
if Session['ID_USUARIO'] = nil then
  Response.Redirect('login.aspx')
else
  ... <- código existente
```

Se ele não estiver logado, redirecionamos para a página de login. Essa técnica pode ser usada para qualquer página que não possa ser acessada por usuários anônimos.

Finalizando a compra

Para completar o nosso carrinho de compras, precisamos realizar algumas configurações. No User Control *header.ascx*, adicione no componente *Hyperlink* do carrinho, a página do carrinho de compras (*carrinho.aspx*), na proprie-

dade *NavigateURL*.

Se você tentar acessar a página por esse link, terá um erro, pois ainda não estamos verificando se queremos inserir um item no carrinho ou somente visualizar o mesmo. Para isso, altere o código do *Load* do *carrinho.aspx* com o código da **Listagem 7**.

Listagem 7. Alterando o código do Load da página carrinho.aspx

```
if Session['ID_USUARIO'] = nil then
    Response.Redirect('login.aspx')
else
begin
    if not IsPostBack then
    begin
        if Request.QueryString['Add'] <> nil then
            begin
                if Request.QueryString['CodProduto'] <> nil then
                    ConsultaProduto;
                end;
                MostraGrid
            end;
        end;
    end;
end;
```

Note que estamos verificando mais uma vez a *QueryString*. Na página *visualizaproduto.aspx*, também temos que modificar a URL, para que tenha a *QueryString* que indica que estamos adicionando um produto no carrinho, ficando com o seguinte código:

```
Session['URL'] := 'carrinho.aspx?Add=True&CodProduto='+
Request.QueryString['CodProduto']+
'%Quant='+txtQuantidade.Text;
Response.Redirect(Session['URL'].ToString);
```

Veja que nesse momento estamos salvando a URL para o carrinho. Quando for o primeiro item a ser adicionado, o usu-

ário terá que logar, então a página *login.aspx* não sabe qual o produto escolhido pelo cliente, por isso estamos salvando o caminho (URL). Como já fizemos na página de login, após o usuário digitar a senha, repassamos esse caminho para que o item seja adicionado no carrinho de compras.

Adicione dois botões na página *carrinho.aspx*, um para voltar à página principal (para o usuário continuar comprando) e outro para finalizar a compra (crie um novo *Web Form* e chame-o de "finaliza.aspx"). Adicione dois *FbCommands* na página, dando o nome de "cmdVendas" e "cmdItens". Configure-os para apontar para o *FbConnection*.

Vamos criar duas *Stored Procedures* para adicionar os dados da venda e itens no banco. Abra o *IBExpert* e adicione o código da **Listagem 8** para criar a *Stored Procedure* de inserção de Vendas e de Itens.

Listagem 8. SPs de inserção de Vendas e Itens

```
CREATE PROCEDURE VENDAS_INS (
    ID_USUARIO INTEGER,
    DATA_VENDA DATE,
    TOTAL_VENDA NUMERIC(9,2))
```

```
AS
BEGIN
    INSERT INTO VENDAS (
        ID_VENDA,
        ID_USUARIO,
        DATA_VENDA,
        TOTAL_VENDA)
    VALUES (
        GEN_ID(GEN_VENDAS_ID, 1),
        :ID_USUARIO,
        :DATA_VENDA,
        :TOTAL_VENDA);
END
```

```
CREATE PROCEDURE ITENS_VENDAS_INS (
    ID_VENDA INTEGER,
    ID_PRODUTO INTEGER,
    QUANTIDADE INTEGER,
    VALOR_UNITARIO NUMERIC(9,2))
```

```
AS
BEGIN
    INSERT INTO ITENS_VENDAS (
        ID_VENDA,
        ID_PRODUTO,
        QUANTIDADE,
        VALOR_UNITARIO)
    VALUES (
        :ID_VENDA,
        :ID_PRODUTO,
        :QUANTIDADE,
        :VALOR_UNITARIO);
END
```

Voltando para o Delphi, configure a propriedade *CommandText* do *cmdVendas* para:

```
execute procedure VENDAS_INS (?, ?,?)
```

Crie os parâmetros com o mesmo nome dos campos (conforme fizemos em exemplos anteriores). Para o *cmdItens*, adicione o seguinte código na propriedade *CommandText*:

```
execute procedure ITENS_VENDAS_INS(?, ?, ?, ?)
```

Configure os parâmetros do componente. Adicione mais um *FbCommand* (chamado "cmdGenerator") para que possamos saber o valor do *ID_VENDA*, criado através de um *Generator* (configuração realizada no banco). Adicione o seguinte código no *CommandText* do *cmdGenerator*:

```
select GEN_ID(GEN_VENDAS_ID, 0) from RDBSDATABASE
```



Esse código pega o valor atual do *Generator*, passado como parâmetro. No botão de finalizar, vamos adicionar o código da **Listagem 9**.

Listagem 9. Finalizando as compras

```
var
row: DataRow;
IdCompra: System.&Object;
begin
FbConnection1.Open;
try
cmdVenda.Parameters[0].Value := Session['ID_USUARIO'];
cmdVenda.Parameters[1].Value := DateTime.Now;
cmdVenda.Parameters[2].Value :=
  GetDataSet.Tables[0].Rows[0]['TOTAL'];
cmdVenda.ExecuteNonQuery;
IdCompra := cmdGenerator.ExecuteScalar;
for row in GetDataSet.Tables[0].Rows do
begin
cmdItens.Parameters[0].Value := IdCompra;
cmdItens.Parameters[1].Value := row['ID_PRODUTO'];
cmdItens.Parameters[2].Value := row['QUANTIDADE'];
cmdItens.Parameters[3].Value := row['PRECO'];
cmdItens.ExecuteNonQuery;
end;
finally
FbConnection1.Close;
end;
Session['CARRINHO'] := nil;
Response.Redirect('finaliza.aspx');
end;
```

Na listagem anterior executamos a SP de Vendas e pegamos o valor do *Generator* (note que usamos o *ExecuteScalar*, pois teremos o retorno de somente uma linha e uma coluna, ficando assim mais otimizado).

Usamos o novo comando *for in* para varrer todas as colunas do *DataSet* e adicionar os valores dos itens da venda no respectivo *Command*. Por fim, fechamos a conexão com o banco, limpamos a sessão com o carrinho de compras (pois não será mais necessária) e redirecionamos para a página de finalização de compras. Para finalizar, você pode adicionar nas páginas os User Controls, criados anteriormente. Rode a aplicação e teste (**Figura 5**).

Conclusão

Vimos mais um capítulo do nosso mini-curso de ASP.NET e Delphi 2006. Aprendemos a criar o carrinho de compras, usando *DataSet* em memória, onde inserimos e excluimos os itens do carrinho. Adicionamos segurança ao site, pois o usuário pode visualizar os produtos do carrinho e conseqüentemente finalizar a compra, somente se estiver logado. Na próxima edição, vamos adicionar as informações referente a forma de pagamento, onde usaremos o CobreBem para gerar boletos na aplicação. Um grande abraço a todos e até a próxima!

Links

Curso de E-Commerce com Delphi 2005 e ASP.NET

www.devmedia.com.br/curso/ecommerce2005

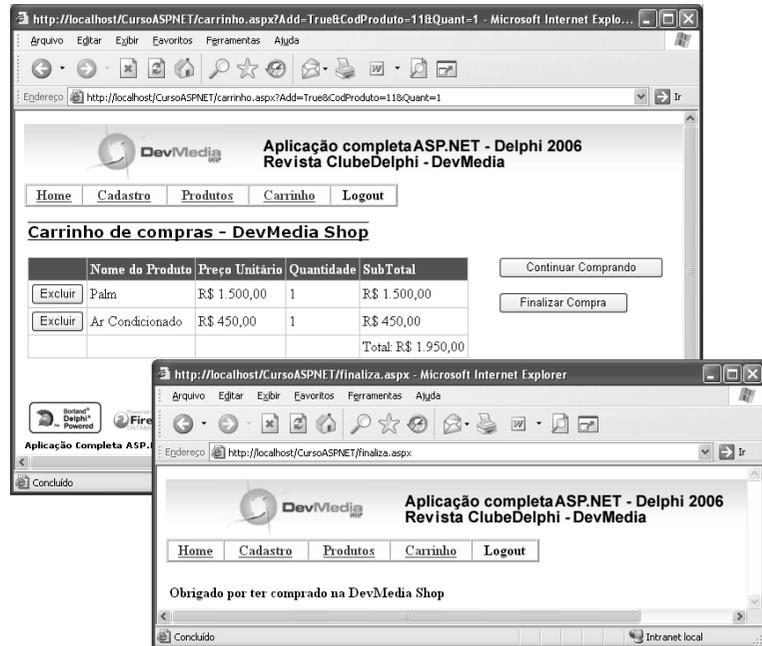


Figura 5. Finalizando a compra do site

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

DOWNLOADS E INFORMAÇÕES EM WWW.COBREBEM.COM

Tecnologia