

Nesta seção você encontra artigos para iniciantes na linguagem Delphi

Treinamento em ASP.NET – Parte 2

Primeiros passos com ASP.NET

Vamos à segunda parte do treinamento em ASP.NET, na qual você aprenderá a manipular alguns dos controles utilizados com mais frequência durante o acesso a banco de dados. Atualmente é normal que os dados fiquem armazenados em alguma fonte de dados, seja um banco de dados no SQL Server, Oracle, MS-Access, DB2, Firebird ou então, em arquivos XML e textos. No .NET, isso é completamente transparente, ou seja, não importa onde estejam os dados.

Escolhendo o provider para ADO.NET

Temos somente uma opção para acessar o Firebird a partir de aplicações construídas para o .NET Framework: o ADO.NET. Sabendo disso, o próximo passo é escolher um *Provider* para ADO.NET que permita o melhor acesso ao banco de dados.

O .NET Framework 1.1 já é distribuído com quatro *Providers* nativos para ADO.NET:

SQL Provider - para acesso ao SQL Server;
Oracle Provider - para acesso ao Oracle;
OleDb Provider - para acesso a fontes de dados que possuam um driver OleDb;
ODBC Provider - para acesso a fontes de dados que possuam um driver ODBC;

Para acessar o Firebird, poderíamos usar qualquer um dos dois últimos *Providers* (ODBC ou OLEDB). No entanto, seria necessário instalar uma camada extra (um driver OLEDB ou ODBC), o que comprometeria o tempo de resposta, que é crítico em soluções Web.

Download e instalação do Firebird Data Provider

Acesse o endereço www.firebirdsql.com, clique no link *Download* e a seguir em *Firebird .NET Data Provider*. Baixe e instale a última versão do *Data Provider for .NET Framework 1.1*, bastando seguir os passos no assistente que será iniciado.



Renato Haddad

é Microsoft Most Valuable Professional (MVP).
Autor de diversos treinamentos multimídia .NET e SQL Reporting Services para Microsoft Brasil e América Latina.

Nota: Neste artigo usaremos versão 1.7 do *Provider*, versão mais recente disponível até o fechamento desta edição.

Após a instalação, no Delphi 2006 clique no menu *Component\Installed .NET Components*. Digite “Firebird Data Provider” na opção *Category*, clique no botão *Select an Assembly* (Figura 1) e escolha o arquivo *FirebirdSql.Data.Firebird.dll*, localizado no diretório de instalação do *Provider*, por padrão em *C:\Arquivos de programas\FirebirdNETProvider(versão)*. Clique em *Ok* e observe que os novos componentes para acesso ao Firebird estão agora disponíveis no IDE (Figura 2).

Desenvolvendo o exemplo

Na parte 1, você aprendeu a criar uma aplicação .NET no Developer Studio 2006. Então, abra a aplicação *DelphiMag*, criada na parte 1, e adicione um novo formulário usando a opção *File\New\Other>Delphi for .NET Projects>New ASP.NET Files>ASP.NET Page*. Salve o arquivo como “BancoDados.aspx” usando a opção *File>Save as*. Vamos montar um formulário semelhante ao da Figura 3. Para isso a primeira providência é digitar o título principal da página. Clique em qualquer lugar do formulário e digite “Banco de Dados – ASP.NET”. Para formatar basta selecionar o texto e usar os botões na barra de ferramentas superior. Em seguida pressione *Enter* e clique no botão *Insert table* também na barra de ferramentas.

Na caixa de diálogo que se abre (Figura 4) digite “3” para *Rows* (linhas) e “2”



Figura 2. Componentes do provider .NET Firebird já instalados

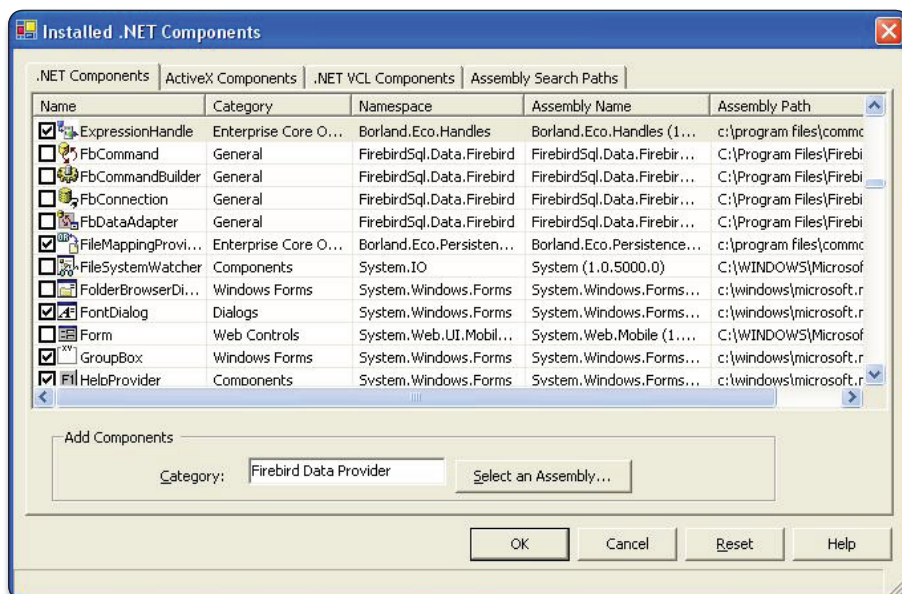


Figura 1. Instalação do provider .NET no Delphi

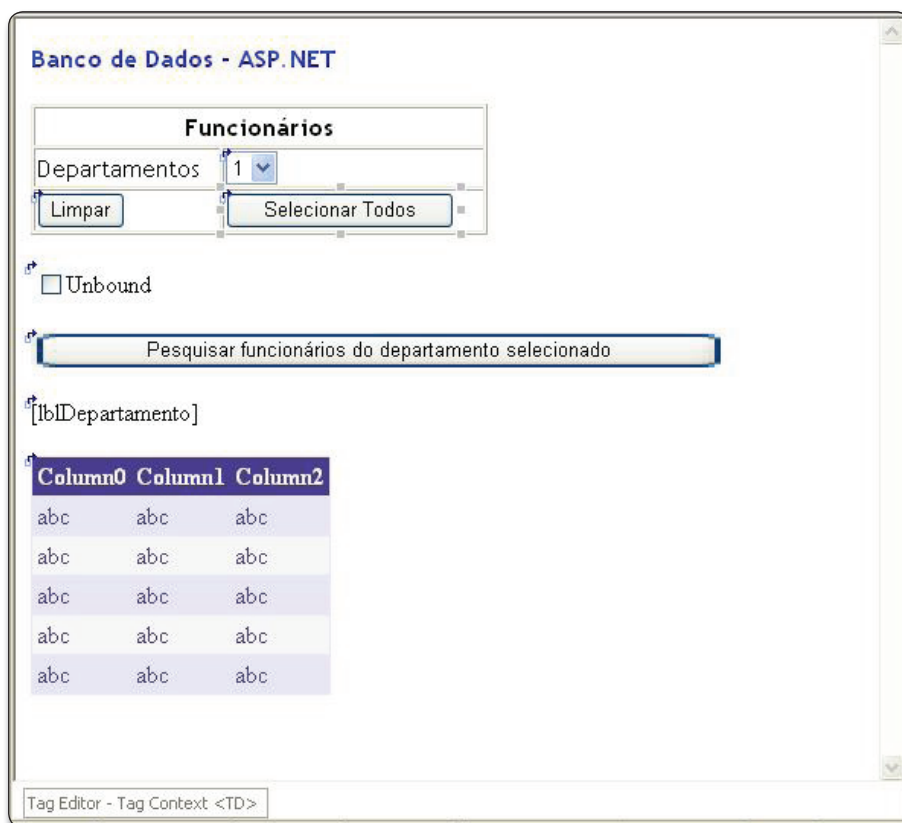


Figura 3. Exemplo de página em nosso exemplo

para *Columns*(colunas), ou seja, estamos criando uma tabela com três linhas e duas colunas.

Nota: Na Web as tabelas são normalmente utilizadas para alinhar elementos nas páginas. Elas são representadas pela tag *Table* e possuem outras tags tais como: *TD* = célula e *TR* = linha. Cada uma dessas tags possui seus próprios atributos/propriedades.

Após inserida a tabela, note que no *Tag Editor* na parte inferior do Delphi (**Figura 5**) são criadas as tags *TR* e *TD* e assim por diante. À medida que vamos adicionando novos elementos à página, este editor é alterado simultaneamente, isso porque ele reflete todas as alterações e converte-as em código HTML, isso significa que se você tem algum conhecimento de HTML ficará bastante fácil de entender.

Perceba as linhas a seguir e veja que

foram criadas as tags *TD* e *TR*. Com elas podemos mexer em alinhamentos, cores etc. Nesse momento faremos um pequeno ajuste. Mesclaremos as duas primeiras células para que se tornem uma só.

```
<tr>
  <td><font face="Arial">Funcionários
</font></td>
  <td><font face="Arial"></font></td>
</tr>
```

Para tanto faça uma pequena alteração: altere a primeira tag *TD* para

```
<td colspan="2">
```

Por fim exclua a segunda linha *TD* que encontrar. Seu código final ficará como a seguir:

```
<td colspan="2">
<font face="Arial">Funcionários</font></td>
```

O que fizemos foi introduzir o comando HTML *colspan* informando a ele quantas células serão usadas para a mesclagem, nesse caso duas. Esse efeito também pode ser notado diretamente na página.

Nota: Caso não esteja visualizando as tags *TD* e *TR* no *Tag Editor*, experimente clicar nas setas para esquerda e/ou direita na barra de ferramentas do *Tag Editor*. Elas mostram todos os elementos HTML do objeto selecionado no Delphi.

Logo em seguida digite os textos em seus devidos lugares dentro da tabela de acordo com a **Figura 4**. Para os componentes usaremos a paleta *Web Controls*, portanto, arraste para a célula correspondente um componente *DropDownList* ("DropDepartamentos"). Marque sua propriedade *AutoPostBack* como *True* e adicione a sua propriedade *Items* apenas um elemento onde, tanto no *Text* quanto o *Value*, deverão conter o valor 1. Faremos uma instrução SQL na tabela *Employee* do BD de mesmo nome, que acompanha os exemplos do Firebird, para selecionar os funcionários do departamento escolhido.

Inclua, logo abaixo, um componente do

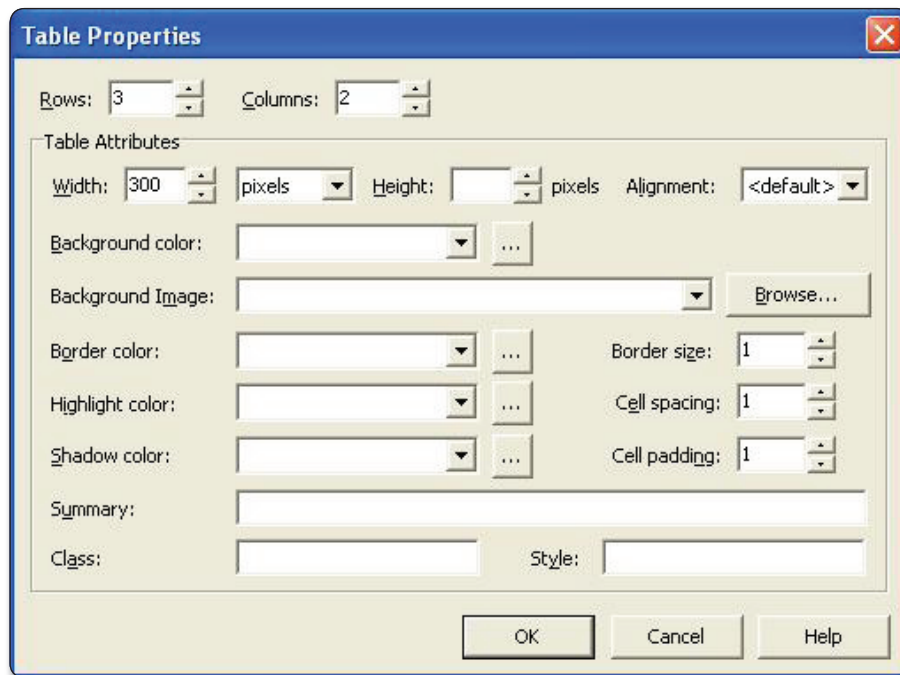


Figura 4. Inclusão de tabela na página

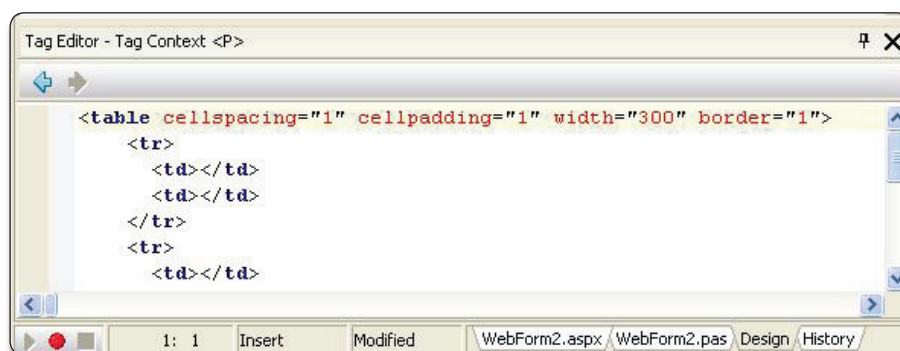


Figura 5. Tag Editor – Editor de tags do Delphi 2006



tipo *CheckBoxList*("cblDepartamentos"), modifique a propriedade *RepeaterColumns* para "3", assim teremos nossos departamentos mostrados em 3 colunas. Em seguida inclua um *Button* ("btnPesquisar") fora da tabela e modifique seu *Text* para "Pesquisar funcionários dos departamentos selecionados". Adicione logo abaixo um *Label*("lblDepartamento"). Nas duas últimas células da tabela coloque dois *Buttons*("btnLimpa") e ("btnSelecionarTodos"), respectivamente. Mude o *Text* de cada *Button* para "Limpar" e "Selecionar todos" também respectivamente. Por último inclua um componente *DataGrid* ("gridEmployee").

Nota: Para que o *CheckBoxList* fique com uma aparência agradável recomendando que altere as propriedades de fonte e aumente também a largura do componente de forma que ocupe quase toda largura da página.

Nota: Sempre que você quiser mudar de linha pressione ENTER ou SHIFT + ENTER. Existe uma diferença sutil no espaçamento, onde ENTER provoca um novo parágrafo e SHIFT + ENTER provoca uma quebra de linha.

Com relação ao *DataGrid*, você pode formatá-lo de acordo com as opções de formato pré-definidas. Para isso, pressione o botão direito sobre o controle e selecione *Auto Format* ou então a partir do *Object Inspector*. Escolha um formato de acordo com a sua necessidade e observe que as propriedades do estilo serão configuradas automaticamente.

O banco de dados usado como exemplo é o *Employee*, que está instalado no Firebird. O funcionamento se dará da seguinte forma: na primeira vez que a página for carregada, o controle *cblDepartamentos* será preenchido com todos os departamentos existentes na tabela

Listagem 1. Código do evento Page_Load

```
procedure TWebForm2.Page_Load(sender: System.Object;
e: System.EventArgs);
var
  Conn : FbConnection;
  DataAdapter : FbDataAdapter;
  Ds : DataSet;
  Command : FbCommand;
begin
  { Verifica se foi dado algum Post na página }
  if not Page.IsPostBack Then
    begin
      { Criação dos objetos de conexão }
      Conn := FbConnection.Create;
      DataAdapter := FbDataAdapter.Create;
      Command := FbCommand.Create;

      { Atribuição da string de conexão e abertura do BD }
      Conn.ConnectionString := Conexao;
      Conn.Open;

      { Atribuição dos atributos de seleção dos dados }
      DataAdapter.SelectCommand := Command;
      DataAdapter.SelectCommand.Connection := Conn;
      DataAdapter.SelectCommand.CommandText := 'Select * from Department order by Department';

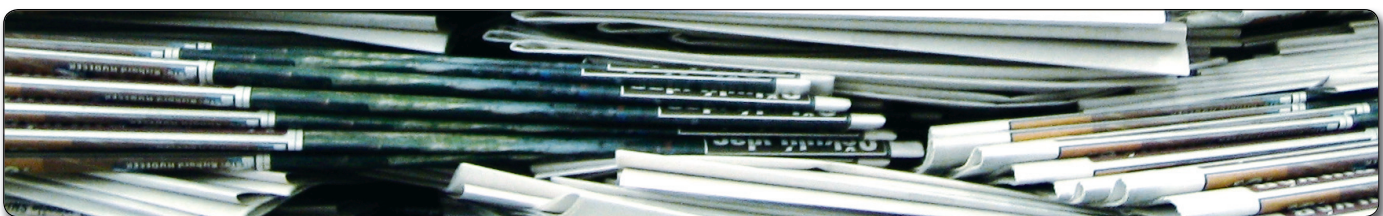
      { Criação em memória do DataSet auxiliar }
      Ds := DataSet.Create;
      DataAdapter.Fill(Ds, 'Departamentos');
      try
        { Configura o objeto CheckBoxList }
        with cblDepartamentos do
          begin
            { Define que campo será exibido }
            DataTextField := 'Department';
            { Define que campo será armazenado }
            DataValueField := 'Dept_No';
            { Define a origem do controle que é a tabela }
            { Coloca em memória os departamentos }
            DataSource := Ds;
            { Preenche o controle }
            cblDepartamentos.DataBind;
          end;
        finally
          Conn.Close;
        end;
      end;
    end;
  end;
```

Listagem 2. Código para desmarcar ou selecionar todos os departamentos

```
procedure TWebForm2.ChecarDepartamentos(AChecar: Boolean);
var
  I : Integer;
begin
  for I := 0 to cblDepartamentos.Items.Count-1 do
    cblDepartamentos.Items[I].Selected := AChecar;
  if not AChecar then
    begin
      gridEmployee.Visible := False;
      lblDepartamento.Visible := False;
    end;
  end;
end;
```

Listagem 3. Código para desmarcar ou selecionar todos os departamentos

```
procedure TWebForm2.btnSelecionarTodos_Click(sender: System.Object; e: System.EventArgs);
begin
  ChecarDepartamentos(True);
end;
procedure TWebForm2.btnLimpa_Click(sender: System.Object; e: System.EventArgs);
begin
  ChecarDepartamentos(False);
end;
```



Listagem 4. Código do botão de Pesquisa

```

procedure TWebForm2.btnPesquisar_Click(sender: System.Object; e: System.EventArgs);
var
  Conn : FbConnection;
  DataAdapter : FbDataAdapter;
  Ds : DataSet;
  Command : FbCommand;
  DR : FbDataReader;
  Condicao : StringBuilder;
  Selecao : System.Text.StringBuilder;
  I : Integer;
begin
  { Criação dos objetos de conexão }
  Conn := FbConnection.Create;
  DataAdapter := FbDataAdapter.Create;
  Command := FbCommand.Create;
  { Atribuição da string de conexão e abertura do BD }
  Conn.ConnectionString := Conexao;
  Conn.Open;

  { Select para acessar os dados }
  { Atribuição dos atributos de seleção dos dados }
  DataAdapter.SelectCommand := Command;
  DataAdapter.SelectCommand.Connection := Conn;
  Selecao := System.Text.StringBuilder.Create;
  Selecao.Append('Select Emp_No, First_Name, Last_Name from Employee ');
  Condicao := System.Text.StringBuilder.Create;
  Condicao.Append(' where Dept_no in (');
  for I := 0 to cblDepartamentos.Items.Count-1 do
    if cblDepartamentos.Items[I].Selected then
      { Para usar o QuotedStr no ASP.NET declare o Namespace Borland.Vcl.SysUtils }
      Condicao.Append(QuotedStr( cblDepartamentos.Items[I].Value ) + ',' );
  end;
  Condicao.Append(')');
  Condicao := Condicao.Replace(',', ' ');
  Selecao.Append(Condicao);
  Command.CommandText := Selecao.ToString;
  DR := Command.ExecuteReader;

  //Exibe o DataGrid de produtos e o Label
  gridEmployee.Visible := True;
  lblDepartamento.Visible := True;

  //Define a origem do DataGrid
  gridEmployee.DataSource := DR;

  //Preenche o DataGrid
  gridEmployee.DataBind;

  //Exibe um texto contendo o número de //produtos do DataGrid
  lblDepartamento.Text := 'Existem ' + gridEmployee.Items.Count.ToString() + ' funcionários.';

  //Fecha o DataReader e a conexão
  DR.Close;
  Conn.Close;
end;

```

Department do banco de dados. Com isso, o internauta poderá selecionar qualquer departamento existente neste controle.

Quando o botão *btnPesquisar* for clicado serão exibidos no *gridEmployee* todos os funcionários referentes aos departamentos selecionados. Os demais controles existem para que possamos explorar a parte de design interativo com os internautas, ou seja, você pode criar uma página para que o internauta configure de acordo com as opções selecionadas. Esse processo é possível porque os controles permitem atribuir propriedades em tempo de execução, dispensando completamente a edição manual do design. Esse é um dos pontos fortes do ASP.NET.

Agora que você já conhece o design da página e como deverá funcionar, o próximo passo será digitar os códigos.

Codificando o exemplo

Vamos entrar no editor de códigos do Delphi usando a tecla *F12* ou *View>Show Code*. Como vamos acessar um banco de dados Firebird, declare na seção *Uses* do formulário a referência da classe do Firebird como segue:

```
FirebirdSql.Data.Firebird
```

Como iremos abrir o banco de dados em dois lugares, para facilitar a declaração da conexão, declare a variável conexão antes do evento *Page_Load*. Essa variável é do tipo *string* e contém o nome do banco de dados (Database), o nome do servidor (Server) e o usuário (User ID) de acordo com suas especificações da instalação do Firebird na sua máquina. Caso necessite, declare a *Password* e os demais parâmetros da conexão. É importante ressaltar que todos os parâmetros declarados na conexão dependem de como você fez a sua instalação do Firebird, os usuários existentes e as senhas. Caso você execute a página e ocorra um erro de *login* ou conexão, verifique estes itens para que o código possa acessar o banco de dados *Employee*.

A constante que criaremos pode ser adicionada logo abaixo a seção *Implementation* como segue:

```

implementation
const
  Conexao = 'User=SYSDBA;
  Password=masterkey;
  Database=C:\Program Files\Firebird\
  Firebird_2_0\examples\empbuild\EMPLOYEE.FDB;';

```

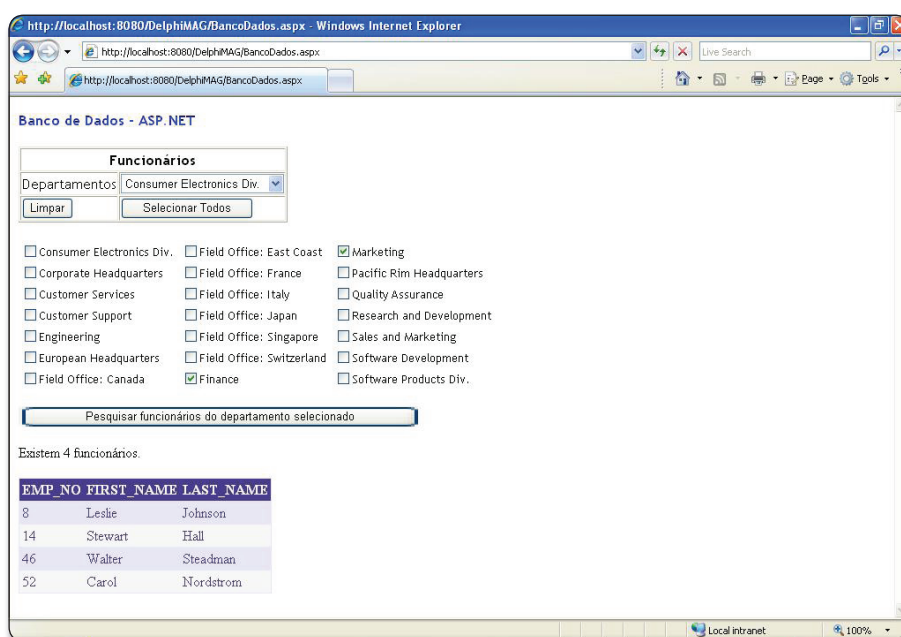


Figura 6. Execução da página

Digite o código da **Listagem 1** no evento *Page_Load* para preencher o controle *cblDepartamentos* com todos os departamentos existentes no banco de dados. Vale dizer que esse evento será executado todas as vezes que a página for carregada. No entanto, os departamentos não mudam, e a pergunta é: por que então carregar todas as vezes que a página for executada? Realmente, não há a menor razão para isso e, desse modo, é possível verificar este fato através do *Page.IsPostBack*, ou seja, tudo o que estiver dentro deste bloco será executado apenas na primeira vez que a página for carregada.

A explicação para o código da **Listagem 1** parece bastante complicado, mas não é. Vamos ver passo a passo o que estamos codificando. Para iniciar estamos criando todos os componentes de conexão em tempo de execução, por isso declaramos as variáveis *Conn*, *DataAdapter*, *Ds* e *Command*, como podemos ver a seguir:

```
var
  Conn : FbConnection;
  DataAdapter : FbDataAdapter;
  Ds : DataSet;
  Command : FbCommand;
```

Em seguida atribuímos a *string* de conexão ao componente *FbConnection*, conectamos ao *BD* e atualizamos os demais componentes de conexão com os atributos necessários para seleção dos dados. Note que criamos um *DataAdapter* e *FbCommand*. Logo após atribuímos a propriedade *SelectCommand* do *DataAdapter* do *FbCommand* recém criado. O *DataAdapter* é o componente que usamos para selecionar (*SelectCommand*), atualizar (*UpdateCommand*), inserir (*InsertCommand*) e apagar (*DeleteCommand*) os registros na tabela do banco de dados. Como pode ver podemos ter as quatro operações básicas de banco de dados em um único componente, mas para que cada operação seja executada precisamos ter um *FbCommand*, por isso o criamos e o atribuímos à propriedade *SelectCommand*.

Depois da conexão estabelecida, criamos um *DataSet* em memória, o preenchemos e por último atualizamos o controle *CheckBoxList*. Mas o que é um *DataSet*? *DataSet* é uma representação de uma ou mais tabelas na memória (contendo os métodos *Select*, *Insert*, *Delete* e *Update*) e é possível inclusive estabelecer relacionamentos entre elas. Isso significa dizer

Listagem 5. Trecho de código de preenchimento do *DropDepartamentos* no evento *Page_Load*

```
procedure TWebForm2.Page_Load(sender: System.Object; e: System.EventArgs);
begin
  ...
  with DropDepartamento do
  begin
    Items.Clear;
    DataTextField := 'Department';
    DataValueField := 'Dept_No';
    DataSource := Ds;
    DropDepartamento.DataBind;
  end;
  ...
```

que podemos ter um *DataSet* contendo diversas tabelas.

Pelo fato dessa representação ser feita na memória, tudo é mais rápido. A grande característica do ADO.NET é trabalhar com dados desconectados, ou seja, todas as alterações são feitas na memória até que você as efetive no banco de dados.

Voltando ao exemplo: como todos os departamentos estão listados e dispõem de um *CheckBox* para cada departamento, vamos facilitar a vida do internauta. Caso o internauta queira selecionar todos os departamentos existentes, para que ele não precise clicar em cada departamento, podemos montar um código que faça isso automaticamente. A mesma idéia serve para desmarcar todos os departamentos selecionados.

Na **Listagem 2** faremos a criação de um procedimento para selecionar ou limpar a seleção de todos os *CheckBoxes* do *cblDepartamentos*. Note que estamos recebendo um parâmetro (*ACheckar*) do tipo *Boolean*, dessa forma podemos fazer a chamada ao procedimento tanto do botão *btnLimpa(False)* quanto para *btnSelecionarTodos(True)*. Localize então a seção *Public* no Delphi e declare o procedimento como a seguir:

```
procedure ChecarDepartamentos(ACheckar: Boolean);
```

Para criar o cabeçalho da implementação do procedimento pressione *Ctrl + Shift + C*. Digite o código da **Listagem 2**. A providência que tomamos na **Listagem 2** é bem simples. Apenas envolvemos em um laço *For* os itens do componente *cblDepartamentos* e atribuímos o valor do parâmetro *ACheckar* marcando ou desmarcando os itens de acordo com o valor atribuído ao parâmetro. Em seguida ocultamos o *lblFuncionarios* e o *gridFuncionarios* de acordo com o valor do parâmetro.

Agora basta chamarmos a *prodecure*

ChecarDepartamentos passando para ela *True* ou *False* dependendo da situação. Para o botão *btnLimpa* digite a chamada do procedimento passando *False* no parâmetro e para *btnSelecionarTodos* repita a chamada passando *True*. Veja na **Listagem 3** o código completo de ambos botões.

A grande facilidade do ASP.NET é demonstrada aqui. Após selecionar os departamentos, você precisa montar o *DataGrid* com todos os funcionários, de acordo com os respectivos departamentos selecionados. A construção desse código é relativamente simples porque basta você aplicar um filtro à cláusula *Where* do SQL. No entanto, como o internauta pode selecionar uma ou todos os departamentos, o filtro precisará ser montado de forma que o *looping* percorra todos os itens do *cblDepartamentos* e verifique quais estão selecionados. Para cada item selecionado, é concatenada uma variável que contém o respectivo *ID* do departamento para executar o *DataReader* com a *string SQL*.

Quando usar um *DataReader*? Use o *DataReader* sempre que precisar ler uma fonte de dados; o *DataReader* é do tipo *Forward Only*, ou seja, você não consegue navegar em registros anteriores. Em termos de performance, um *DataReader* é muito mais rápido que um *DataAdapter*. A desvantagem é que você não conseguirá aplicar paginação em um *DataGrid* se este tiver sido preenchido com um *DataReader*, como é o caso deste exemplo.

Você irá notar que, na concatenação de *strings*, estou usando a classe *System.Text.StringBuilder*. Sempre que você precisar concatenar *strings*, opte pelo *StringBuilder* ele é infinitamente mais rápido que outras formas de concatenação, por exemplo, *variável := variável + conteúdo*. Na **Listagem 4** podemos conferir o código completo do botão de pesquisa.

Vamos entender a **Listagem 4**. Note que

no início o código se parece muito com o código do evento *Page_Load*. E realmente é bem parecido. No início declaramos as variáveis necessárias para conexão e seleção dos dados do BD. As novidades são que declaramos 3 novas variáveis:

```
var
  DR : FbDataReader;
  Condição : StringBuilder;
  Seleção : System.Text.StringBuilder;
```

Declaramos um *FbDataReader*. O *FbDataReader* executa a pesquisa no banco e traz os dados em memória. Com seus dados carregados atualizamos nosso *DataGrid*. Duas variáveis do tipo *StringBuilder* que recebem as *strings* de filtro, ou seja, a instrução *SQL* propriamente dita. Criamos então os

componentes em memória, atualizamos as devidas conexões e filtros. Filtrados os dados, atribuímos o *FbDataReader* ao *DataGrid* e mostramos os dados em tela.

Após digitar todo o código, clique com o botão direito em *BancoDados.aspx* e selecione *Set As Start Page*. Compile o projeto e pressione *F9* para que o Delphi execute o exemplo e abra o *browser*.

A última alteração que fiz foi incluir os departamentos da tabela *Department* também no *DropDepartamentos*. O procedimento é bem simples: Basta repetir o *with..do* que fizemos no evento *Page_Load* mudando somente as referências, ou seja, mudando de *cb1Departamentos* para *DropDepartamento*. Veja na **Listagem 5**

como fica a instrução completa. Insira-a logo após o bloco *with..do* do *cb1Departamentos*. O código completo do evento *Page_Load* depois da alteração você encontra na **Listagem 6**. Na **Figura 6** podemos ver a página finalizada e em execução.

Conclusão

O ASP.NET nos permite interagir com o internauta de forma simples e objetiva com apenas algumas linhas de programação, personalizando a exibição do conteúdo da página para o internauta. Quanto ao acesso a fontes de dados, use e abuse das classes e controles de que o ASP.NET dispõe; afinal, programar para *Web* está cada vez mais simples, fácil e produtivo. ●

Listagem 6. Evento completo *Page_Load*

```
procedure TWebForm2.Page_Load(sender: System.Object; e: System.EventArgs);
var
  Conn : FbConnection;
  DataAdapter : FbDataAdapter;
  Ds : DataSet;
  Command : FbCommand;
begin
  { Verifica se foi dado algum Post na página }
  if not Page.IsPostBack Then
  begin
    { Criação dos objetos de conexão }
    Conn := FbConnection.Create;
    DataAdapter := FbDataAdapter.Create;
    Command := FbCommand.Create;

    { Atribuição da string de conexão e abertura do BD }
    Conn.ConnectionString := Conexão;
    Conn.Open;

    { Select para acessar os dados }
    { Atribuição dos atributos de seleção dos dados }
    DataAdapter.SelectCommand := Command;
    DataAdapter.SelectCommand.Connection := Conn;
    DataAdapter.SelectCommand.CommandText := 'Select * from Department order by Department';

    { Criação em memória do DataSet auxiliar }
    Ds := DataSet.Create;
    DataAdapter.Fill(Ds, 'Departamentos');
  try
    { Configura o objeto CheckBoxList }
    with cb1Departamentos do
    begin
      { Define que campo será exibido }
      DataTextField := 'Department';
      { Define que campo será armazenado }
      DataValueField := 'Dept_No';
      { Define a origem do controle que é a tabela }
      { Coloca em memória os departamentos }
      DataSource := Ds;
      { Preenche o controle }
      cb1Departamentos.DataBind;
    end;

    with DropDepartamento do
    begin
      Items.Clear;
      DataTextField := 'Department';
      DataValueField := 'Dept_No';
      DataSource := Ds;
      DropDepartamento.DataBind;
    end;
  finally
    Conn.Close;
  end;
end;
```

