

## Artigos intermediários sobre Delphi Win32 e Delphi .NET


 SEÇÃO

# Crie um fórum de discussão

## Usando Delphi 2006, ASP.NET e Firebird



**Maikel Marcelo Scheid**

([maikelscheid@gmail.com](mailto:maikelscheid@gmail.com))

é técnico em Informática com ênfase em Análise e Programação de Sistemas. Atua na área de Desenvolvimento de Softwares em Delphi para plataforma win32 e .NET com banco de dados Firebird e MS SQL há 3 anos.

Neste artigo veremos uma forma simples de criar um site de fórum. Criaremos um espaço para discussão de assuntos e temas propostos pelos usuários do fórum. Começaremos fazendo a análise da estrutura de dados, após passaremos a criação do banco de dados. Utilizaremos o Firebird 1.5 e para desenvolvimento da aplicação o Delphi 2006 for ASP.NET.

Criaremos um fórum simples, com uma tela de listagem de áreas do fórum, e partindo-se da área selecionada, a listagem de todos os tópicos relacionados à mesma, e por fim o conteúdo de um tópico selecionado e a possibilidade dos usuários em responder ao tópico ou criar um novo.

Adotaremos neste exemplo uma regra de que a resposta e cadastro de tópicos, somente será realizada por usuários logados. Para fazer o controle de permissão de resposta e usuário logado, utilizaremos *User Controls* para rotinas de ocultar áreas de acesso restrito.

---

**Nota:** Devido ao pouco espaço disponível, alguns formulários não serão apresentados completamente. Faça o download dos fontes no endereço do artigo na página da DevMedia e acompanhe todas as funcionalidades do site de fórum.

---

### Criando o banco de dados de exemplo

Para iniciarmos o exemplo, criaremos uma estrutura simplificada de tabelas e relacionamentos (**Figura 1**).

Após definida a estrutura, proceda com a criação do arquivo de dados. Crie sua base com o nome de "database.fdb". Com o banco de dados criado, passaremos agora a criação da aplicação no Delphi 2006, fique a vontade para utilizar também a versão do Delphi 2005 (o *script* do banco encontra-se para download).

### Criando a aplicação

Crie uma nova *ASP.NET Web Application*

for Delphi .NET através do menu *File* do Delphi 2006, e nomeia a mesma como "forum". Utilizaremos neste artigo como servidor Web o *Internet Information Services* (IIS), se preferir poderá utilizar outros servidores compatíveis.

Pressione OK e aguarde para que o Delphi realize as configurações necessárias e crie a estrutura de páginas do projeto, onde você será direcionado à página *WebForm1.aspx*. Utilize o *Project Manager* do Delphi e renomeie a página para "index.aspx". Passaremos agora a criar a conexão da aplicação ao banco de dados criado anteriormente.

### Configurando o componente de acesso a dados

Para realizar o acesso ao banco de dados Firebird faremos uso do provider, que deverá ser instalado no Delphi. Faça o

download do componente no endereço [www.firebirdsql.org](http://www.firebirdsql.org). Localize no *Project Manager* a *Global.asax* e com um duplo clique ative a mesma, onde passaremos a configurar o componente de acesso.

Adicione na *Global.asax* um *FbConnection*, alterando o nome do mesmo para "conexao" e a propriedade *Modifiers* para *Public*. Ao selecionar a propriedade *ConnectionString* uma caixa de diálogo será exibida, onde configuraremos as informações de acesso ao banco de dados (Figura 2). Informe o *Data Source* ou servidor do banco de dados, como "localhost" se a base estiver na mesma máquina.

Informe também o *Database* (caminho do banco). Se a senha do Firebird não foi alterada, teste sua conexão (*Test*) e finalize a configuração clicando em *Accept*. Acesse agora o código do arquivo utilizando a guia *Global.pas* na parte inferior da tela.

Localize entre o final da declaração de métodos e o início da declaração de *implementation*, e crie a variável "dados" do tipo *TGlobal*. Os demais componentes de consultas e parâmetros SQL serão criados em *runtime*, o que permite um maior controle das ações a serem executadas no decorrer da programação.

### Customizando as páginas do fórum

Antes de retornarmos à *index.aspx*, criaremos uma unit para configurar códigos com consultas da aplicação. No menu *File>New>Other>Delphi for .NET Projects* na lista *New Files*, selecione *Unit* e confirme. Salve o projeto e atribua o nome de "UntGeral.pas" para a unit criada.

Criaremos na *UntGeral* uma *procedure*, que chamaremos no *Page\_Load* de todas as páginas, chamada *VerificaConexao*, responsável por abrir o componente

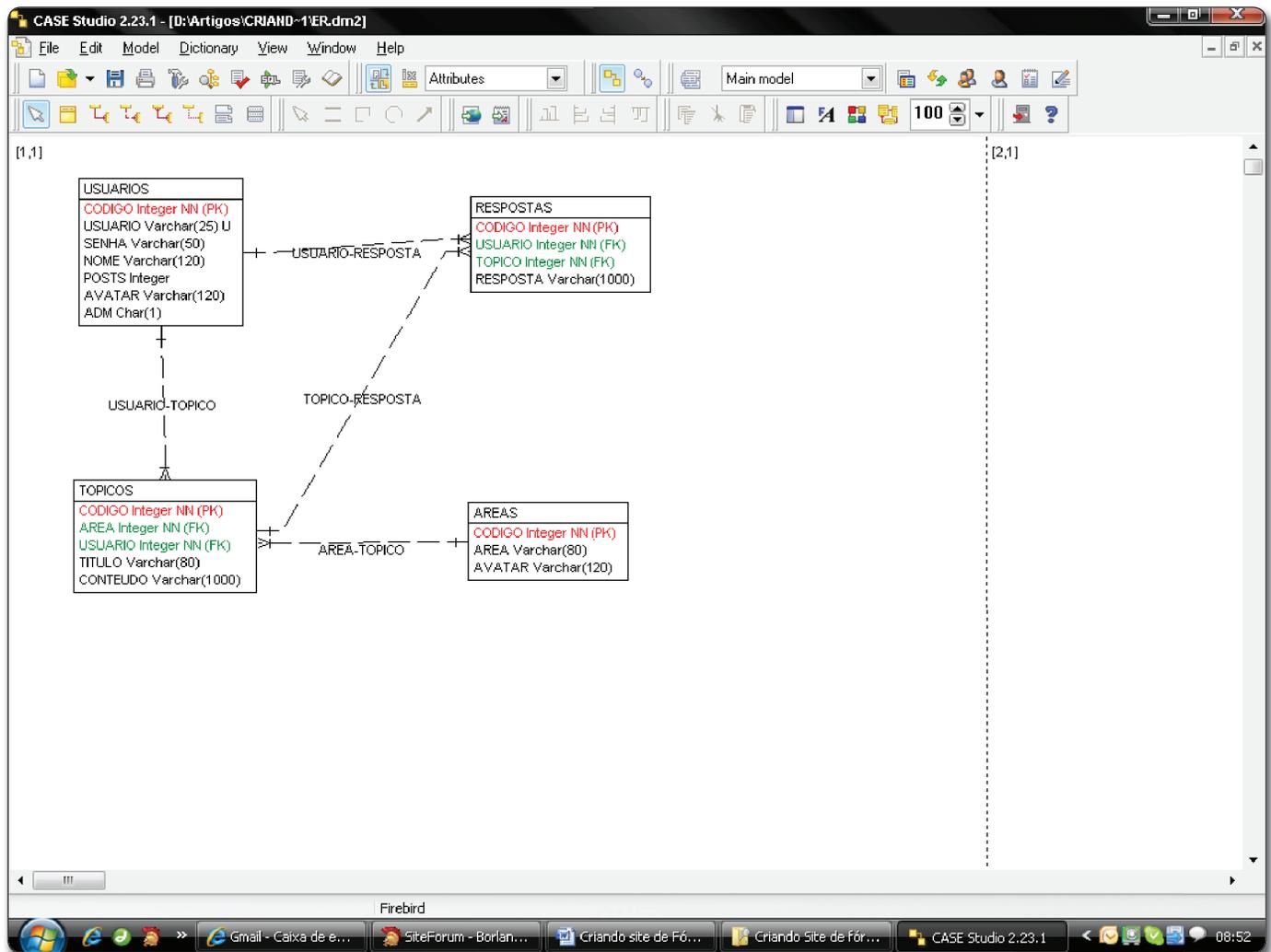


Figura 1. Diagrama ER da aplicação

de conexão a base de dados. Antes de codificarmos a mesma precisamos criar uma referência à *Global.asax*, onde colocamos o componente de acesso. Através do menu *File>Use unit* localize *Global* e confirme. Atribua o código da **Listagem 1** a *UntGeral*.

Na página principal (*index*), no *Page\_Load* chamaremos a *procedure* recém criada (não esqueça de adicionar a unit *UntGeral*).

## Controle de usuários na página

Criaremos um *User Control*, para adicionar o cabeçalho e apresentação da página, além de realizarmos o controle de permissões e usuários logados no site. No menu *File>New>Other* em *New ASP.NET Files* selecione *ASP.NET User Control* e confirme.

Altere o nome do *User Control* para “*header.ascx*”. Adicionaremos na tela uma tabela para melhor organizar a disposição dos componentes. Adicionaremos também ao *User Control* os componentes que utilizaremos para controle dos usuários. Adicione dois *TextBox* (“*TxtUsuario*” e “*TxtSenha*”), e altere a propriedade *TextMode* do controle de senha para *Password*.

Adicione também um *Button*, altere o nome para “*BtnLogin*” e o *Text* para “*Login*”. Adicione três *Labels* alterando o nome do primeiro para “*LblUsuario*”, deixando a propriedade *Text* em branco e *Visible* para *False*.

No segundo, dê o nome de “*LblUser*” com o *Text* de “*Usuário:*” e no último, dê o nome de “*LblSenha*” com *Text* de “*Senha:*”. Organize os componentes

adicionados ao *User Control* conforme a **Figura 3**.

No *BtnLogin*, adicione o código da **Listagem 2**, que será responsável pela autenticação do usuário e senha no banco de dados. Adicione no *uses* os namespaces *FirebirdSQL.Data.Firebird* e *System.Web.Security*. Adicione também através do menu *File>Use Unit* a referência à *Global.asax*.

O código da listagem anterior é responsável por autenticar o usuário e senha na base de dados. Através de dois objetos criados (*FbCon* e *FbRead*), onde *FbCon* vinculado ao componente de conexão da *Global.asax* recebe um código SQL, que faz uma pesquisa de informações a partir dos parâmetros usuário e senha criados.

Perceba que o parâmetro *senha* recebe o resultado da criptografia do valor digitado pelo usuário (através do *FormsAuthentication.HashPasswordForStoringInConfigFile*). Em caso de sucesso, a SQL retorna informações que serão armazenadas em variáveis de sessão e será habilitada na página a descrição do usuário logado.

Pensando em utilizar esse *User Control* em todas as páginas do fórum, precisamos também no *Page\_Load* realizar uma verificação para saber se o usuário já tenha passado pela autenticação, habilitando para os casos os campos de login e/ou a identificação do usuário logado. Adicione o código da **Listagem 3** ao *Page\_Load* do *User Control*.

Nesse código, a presença do bloco *try..except* é de extrema necessidade, pois em caso do usuário ainda não ter passado pela autenticação, as variáveis de sessão também não existiriam, o que provocaria um erro em tempo de execução.

## Exibição das áreas do fórum – página inicial

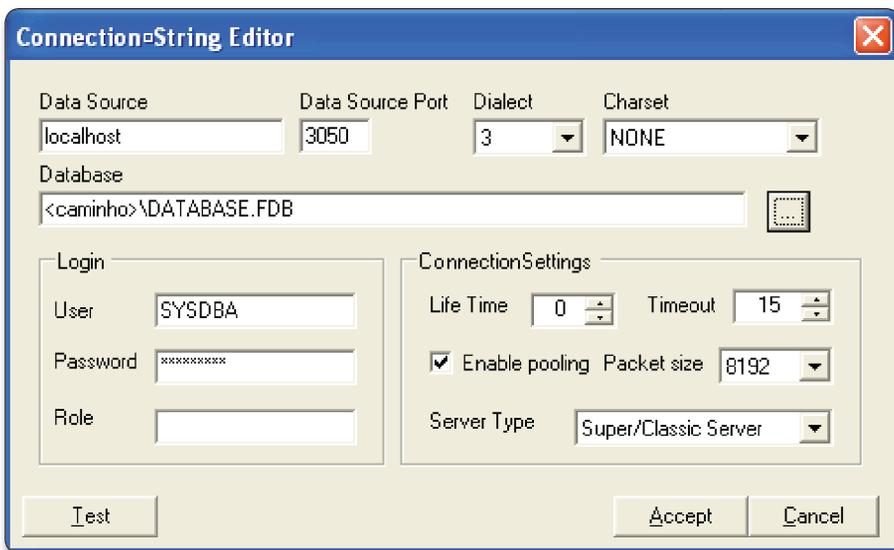
De forma semelhante ao *User Control*, usaremos também uma tabela para organização dos componentes na página.

```

Listagem 1. Criando a VerificaConexao na UntGeral

procedure VerificaConexao;
begin
  if dados = nil then
  begin
    dados := TGlobal.Create;
    dados.conexao.Open;
  end;
end;

```



**Figura 2.** Configurando o componente de acesso a dados



**Figura 3.** Organização dos componentes no User Control

**ClubeDelphi PLUS** [www.devmedia.com.br/dubedelphi/portal.asp](http://www.devmedia.com.br/dubedelphi/portal.asp)

Accesse agora o mesmo o portal do assinante ClubeDelphi e assista a uma vídeo aula de Luciano Pimenta, que mostra como salvar senhas de usuários com criptografia em aplicações ASP.NET.

[www.devmedia.com.br/articles/viewcomp.asp?comp=1281](http://www.devmedia.com.br/articles/viewcomp.asp?comp=1281)

Nessa página, faremos uso apenas de um *DataGrid* que deverá ser renomeado para "GridAreas".

Na estrutura da tabela, teremos três linhas e uma coluna, onde na primeira arrastaremos o *User Control* do *Project Manager*. Na segunda adicionaremos o *GridAreas* e na última, os direitos do site (**Figura 4**). Fique a vontade para utilizar outros recursos de personalização e *layout* da página.

Para configurar o *GridAreas*, acesse o item *Property Builder* no rodapé do *Object Inspector*, onde uma caixa de diálogo será exibida. Clique na guia *Columns*, desmarque a opção *Create columns automatically at run time* e adicione quatro colunas conforme mostra a **Tabela 1**. Na guia *Format* configure a largura posição do texto das colunas.

Ainda na configuração da coluna onde será exibida a área, adicione na opção *URL field* o valor "CODIGO", que será carregado da base de dados. Em *URL format string* adicione "topicos.aspx?area={0}", que será o link a ser chamado quando o usuário clicar na linha do *DataGrid*, onde "topicos.aspx" é a página a ser exibida (criada mais adiante) e "area={0}" é o código da área onde serão listados os tópicos na página.

Voltando para as rotinas de código, criaremos agora na unit uma função que trará como resultado um *DataTable* contendo todas as áreas cadastradas no fórum. Através do *Project Manager* de um clique duplo sobre a unit e registre o seguinte código:

```
function ListaAreasForum: DataTable;
```

**Nota:** Adicione após a declaração *interface*, uma nova seção *uses* e declare os namespaces *System.Data*, *FirebirdSQL.Data.Firebird*.

Implementando o código a nossa função, crie ela na seção *implementation* da unit digitando o código da **Listagem 4**.

No código anterior, observe que logo no início, criamos quatro objetos, que são utilizados para realizar a consulta no banco de dados (*FbCom* e *FbRead*) e também para montagem de uma lista dos resultados obtidos criando um *DataSource*. Após o *while*, onde preenchemos nossa

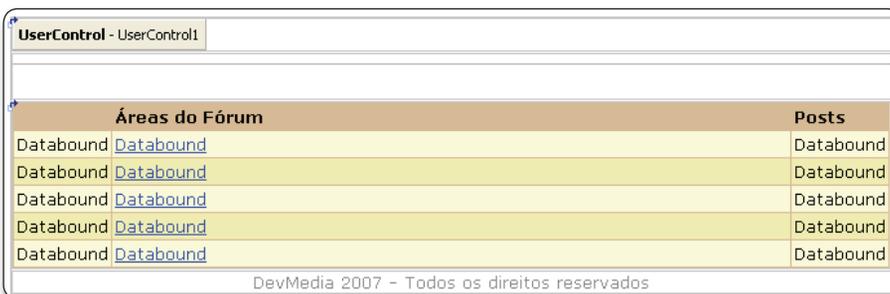


**Listagem 2.** Validando usuário e senha

```
procedure TWebUserControl1.BtnLogin_Click(sender: System.Object; e: System.EventArgs);
var
  FbCon: FbCommand;
  FbRead: FbDataReader;
begin
  try
    FbCon := FbCommand.Create('select usuarios.nome, usuarios.posts, '+
      'usuarios.avatar, usuarios.usuario, usuarios.codigo from usuarios where '+
      '((usuarios.usuario = ?) and (usuarios.senha = ?))', dados.CONEXAO);
    FbCon.Parameters.Clear;
    FbCon.Parameters.Add('USUARIO', DbType.&String);
    FbCon.Parameters.Add('SENHA', DbType.&String);
    FbCon.Parameters['USUARIO'].Value := TxtUsuario.Text.ToUpper;
    FbCon.Parameters['SENHA'].Value := FormsAuthentication.
      HashPasswordForStoringInConfigFile(TxtSenha.Text, 'SHA1');
    FbRead := FbCon.ExecuteReader;
    if FbRead.Read then
      begin
        Session['USUARIO'] := FbRead['usuario'].ToString;
        Session['CODIGO'] := FbRead['codigo'].ToString;
        LblUsuario.Text := '<strong>' + FbRead['nome'].ToString +
          '</strong>, seja bem vindo ao site;';
        LblUsuario.Visible := true;
        TxtUsuario.Visible := false;
        TxtSenha.Visible := false;
        BtnLogin.Visible := false;
        LblUser.Visible := false;
        LblSenha.Visible := false;
      end;
    except
      LblUsuario.Text := 'Não foi possível efetuar login.';
      LblUsuario.Visible := true;
    end;
  end;
end;
```

**Listagem 3.** Page\_Load do User Control

```
try
  if Session['NOME'].ToString <> '' then
    begin
      LblUsuario.Text := '<strong>' + Session[
        'NOME'].ToString +
        '</strong>, seja bem vindo ao site;';
      LblUsuario.Visible := true;
      TxtUsuario.Visible := false;
      TxtSenha.Visible := false;
      BtnLogin.Visible := false;
      LblUser.Visible := false;
      LblSenha.Visible := false;
    end;
  except
  end;
end;
```



**Figura 4.** Organização dos objetos na página inicial

Tipo de Coluna	Texto de Cabeçalho	Campo de Dados	Visível
Bound Column	"Código"	CODIGO	Desmarque a opção
Bound Column		AVATAR	Deixe marcada
HyperLink Column	"Área"	AREA	Deixe marcada
Bound Column	"Tópicos"	TOPICOS	Deixe Marcada

**Tabela 1.** Configurando as colunas do DataGrid

**Listagem 4.** Função para listagem das áreas do fórum

```
function ListaAreasForum: DataTable;
var
  FbCom : FbCommand;
  FbRead: FbDataReader;
  Tabela: DataTable;
  Linha: DataRow;
begin
  Tabela := DataTable.Create;
  Tabela.Columns.Add('CODIGO', TypeOf(&String));
  Tabela.Columns.Add('AVATAR', TypeOf(&String));
  Tabela.Columns.Add('AREA', TypeOf(&String));
  Tabela.Columns.Add('TOPICOS', TypeOf(&String));
  FbCom := FbCommand.Create(
    'select areas.codigo, areas.area, areas.avatar, (select count(topicos.codigo) '+
    'from topicos where topicos.area = areas.codigo) as topicos from areas', dados.CONEXAO);
  FbRead := FbCom.ExecuteReader;
  while FbRead.Read do
  begin
    Linha := Tabela.NewRow;
    Linha[0] := FbRead['codigo'].ToString;
    Linha[1] := '<img hspace="0"src="" + FbRead['avatar'].ToString +
    "" align="absMiddle" border="0">';
    Linha[2] := FbRead['area'].ToString;
    Linha[3] := FbRead['topicos'].ToString;
    Tabela.Rows.Add(Linha);
  end;
  Result := Tabela;
end;
```

lista, jogamos o valor do objeto *Tabela* para o *Result*.

Para verificar o resultado da função, volte na página *index.aspx* e no *Page\_Load* adicione o seguinte código, logo abaixo da *VerificaConexao*:

```
GridAreas.DataSource := ListaAreasForum;
GridAreas.DataBind;
```

Antes de executar sua aplicação, acesse o banco de dados e faça o cadastro de forma manual de uma área do fórum e após cadastre também um tópico qualquer relacionado à área cadastrada. Realizado o cadastro, compile e rode a aplicação para visualizar o resultado.

Veja agora que na estrutura da página foi carregada a área do fórum cadastrada com avatar e quantidade de *posts* a ela relacionados (Figura 5).

---

**Nota:** Quando acessar o banco através do gerenciador de banco de dados para fazer os cadastros, crie antes os auto-incrementos das chaves primárias em todas as tabelas do banco. Como ainda não criamos nenhuma página de cadastro de usuários e por usar criptografia na validação da senha, cadastre um usuário através do gerenciador e informe como senha o valor criptografado *8AEFB06C426E07A0A671A1E2488B4858-D694A730*, que terá o valor *000* ao ser digitada na área de autenticação. Atribua seu *username* e grave o registro.

---

### Definindo estrutura de exibição de tópicos

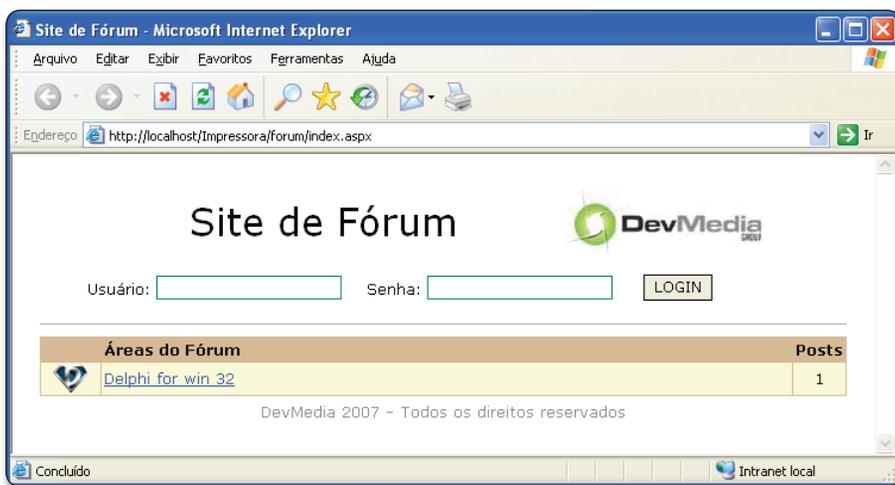
Nosso próximo passo é criar uma nova página que chamaremos de "topicos.aspx", configurando os componentes conforme mostra a Figura 6, para que nela sejam exibidas os tópicos referentes à área selecionada.

Configure o *DataGrid* através da criador de propriedades, conforme mostra a Tabela 2 e altere seu nome para "GridTopicos". Para o *Voltar*, utilizamos um *HTML Label*, alterando o texto para "<< Voltar" e na propriedade *onclick* atribua o *javascript: "history.back();"*

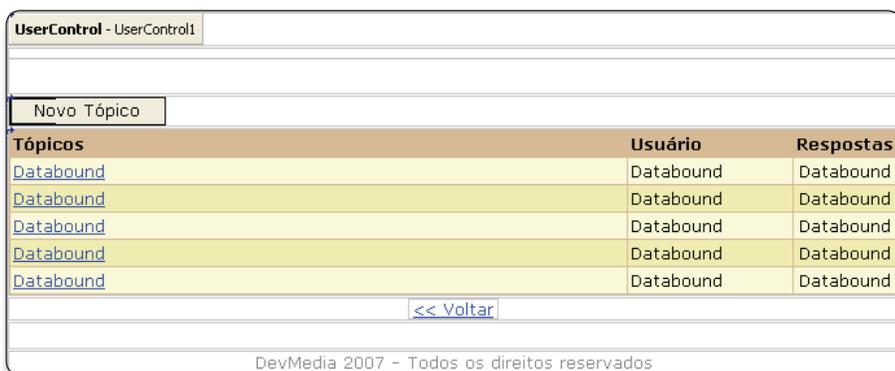
Voltaremos agora para a unit de funções, onde criaremos uma consulta que retornará um *DataTable* carregado com

Tipo de Coluna	Texto de Cabeçalho	Campo de Dados	Visível
Bound Column	Código	CODIGO	Desmarque a opção
HyperLink Column	Tópicos	TOPICO	Deixe marcada
Bound Column	Usuário	USUARIO	Deixe marcada
Bound Column	Resposta	RESPOSTA	Deixe Marcada

**Tabela 2.** Configurando as colunas do DataGrid de tópicos



**Figura 5.** Aplicação em execução



**Figura 6.** Organização dos objetos na página de exibição de tópicos

os tópicos referentes a área selecionada (**Listagem 5**).

Volte a *topicos.aspx* e no *Page\_Load* da mesma, adicione o código a seguir:

```
VerificaConexao;
GridTopicos.DataSource :=
  ListaTopicosArea(Request.
    QueryString['area']);
GridTopicos.DataBind;
Session['CODAREA'] := Request.
  QueryString['area'];
```

Explicando o código anterior, na primeira linha estamos usando o código que verifica o status da conexão. Na segunda, estamos passando para a propriedade *DataSource* do *DataGrid* o resultado da consulta responsável pela listagem dos tópicos.

Como parâmetro para a função, estamos buscando o valor que mandamos junto ao endereço da página. Utilizamos o método *Request* de uma *QueryString*. Após, basta chamar o *DataBind* para que as informações do *DataGrid* sejam escritas na página.

Execute sua aplicação novamente e selecione a área do fórum, visualizando na próxima tela os tópicos referentes à área. Ao passar o mouse sobre um tópico, observe na barra que status que temos um link para exibição do conteúdo do tópico, que será a página que criaremos agora.

## Exibindo tópicos e respectivas respostas

Adicione uma nova página ao projeto, salve-a como "leitura.aspx". Utilizaremos uma estrutura de componentes, um pouco diferente. Para mostrar as informações do tópico usaremos *Labels*, sendo um para o título ("*LblTitulo*"), um para exibir o nome do usuário que realizou o *post* ("*LblUsuario*") e também um *Label* para exibição do conteúdo do tópico ("*LblConteudo*").

Adicione também um *DataGrid* que usaremos para exibir as respostas que o tópico já obteve ("*GridRespostas*"), e organize os componentes da estrutura em tabelas, conforme mostra **Figura 7**.

Para exibir as informações do tópico na tela, criaremos novamente na unit um método que chamaremos de *ListarInformacoesTopico*, que receberá como parâmetro o código do tópico e resultará em três variáveis do tipo *string*, que serão os resultados a serem exibidos na página. Adicione o código da **Listagem 6** na implementação da *procedure*.

### Listagem 5. Função para listagem das tópicos de uma área do fórum

```
function ListaTopicosArea(Area: string): DataTable;
var
  FbCom: FbCommand;
  FbRead: FbDataReader;
  Tabela: DataTable;
  Linha: DataRow;
begin
  Tabela := DataTable.Create;
  Tabela.Columns.Add('CODIGO', TypeOf(&String));
  Tabela.Columns.Add('TOPICO', TypeOf(&String));
  Tabela.Columns.Add('USUARIO', TypeOf(&String));
  Tabela.Columns.Add('RESPOSTA', TypeOf(&String));
  FbCom := FbCommand.Create(
    'select topicos.codigo, topicos.titulo, usuarios.usuario, (select '+
    'count(respostas.codigo) from respostas where respostas.topico = topicos.codigo) '+
    'as respostas from topicos inner join usuarios on (topicos.usuario = '+
    'usuarios.codigo) where ((topicos.area = ?))', dados.CONEXAO);
  FbCom.Parameters.Clear;
  FbCom.Parameters.Add('area', DbType.&String);
  FbCom.Parameters['area'].Value := Area;
  FbRead := FbCom.ExecuteReader;
  while FbRead.Read do
  begin
    Linha := Tabela.NewRow;
    Linha[0] := FbRead['codigo'].ToString;
    Linha[1] := FbRead['titulo'].ToString;
    Linha[2] := FbRead['usuario'].ToString;
    Linha[3] := FbRead['respostas'].ToString;
    Tabela.Rows.Add(Linha);
  end;
  Result := Tabela;
end;
```

### Listagem 6. Procedure para listagem do conteúdo do tópico

```
procedure ListarInformacoesTopico(Codigo: string;
  var Titulo, Usuario, Conteudo: string);
var
  FbCom: FbCommand;
  FbRead: FbDataReader;
begin
  FbCom := FbCommand.Create(
    'select topicos.titulo, topicos.conteudo, usuarios.usuario, '+
    'usuarios.posts, usuarios.avatar from topicos inner join usuarios on (topicos.usuario = '+
    'usuarios.codigo) where ((topicos.codigo = ?))', dados.CONEXAO);
  FbCom.Parameters.Clear;
  FbCom.Parameters.Add('topico', DbType.&String);
  FbCom.Parameters['topico'].Value := Codigo;
  FbRead := FbCom.ExecuteReader;
  if FbRead.Read then
  begin
    Titulo := 'Assunto: <b>'+ FbRead['titulo'].ToString+'</b>';
    Usuario := ''+
    '<br><font face="Verdana" '+style="font-size:11px">'+
    FbRead['usuario'].ToString+'</font><br>'+<font face="Verdana" '+
    'style="font-size:10px">Posts: '+ FbRead['posts'].ToString+'</font>';
    Conteudo := FbRead['conteudo'].ToString;
  end;
end;
```

The screenshot shows a web page layout for displaying a forum topic. At the top, there's a header "UserControl - UserControl1". Below it, there are several labels: "[LblTitulo]", "[LblUsuario]", and "[LblConteudo]". To the right of "[LblConteudo]" is a button labeled "RESPONDER". Below these labels is a table with the following structure:

Respostas	
Databound	Databound

At the bottom of the page, there is a button labeled "<< Voltar" and a footer that reads "DevMedia 2007 - Todos os direitos reservados".

**Figura 7.** Organização dos objetos na página de leitura do tópico

### Listagem 7. Page\_Load da página de leitura do tópico

```
procedure TWebForm1.Page_Load(sender: System.Object;
e: System.EventArgs);
var
Codigo, Titulo, Usuario, Conteudo: string;
begin
VerificaConexao;
Codigo := Request.QueryString['topico'];
ListaInformacoesTopico(Codigo, Titulo, Usuario, Conteudo);
LblTopico.Text := Titulo;
LblUsuario.Text := Usuario;
LblConteudo.Text := Conteudo;
Session['CODTOPICO'] := Codigo;
end;
```

### Listagem 8. Implementação da ListaRespostasTopicos

```
function ListaRespostasTopicos(
Codigo: string): DataTable;
var
FbCom: FbCommand;
FbRead: FbDataReader;
Tabela: DataTable;
Linha: DataRow;
begin
Tabela := DataTable.Create;
Tabela.Columns.Add('USUARIO', TypeOf(&String));
Tabela.Columns.Add('RESPOSTA', TypeOf(&String));
FbCom := FbCommand.Create(
'SELECT RESPOSTAS.RESPOSTA, '+
'USUARIOS.USUARIO, USUARIOS.AVATAR, '+
'USUARIOS.POSTS FROM RESPOSTAS '+
'INNER JOIN USUARIOS ON (RESPOSTAS.USUARIO = '+
'USUARIOS.CODIGO) WHERE ((RESPOSTAS.TOPICO = ?))',
dados.CONEXAO);
FbCom.Parameters.Clear;
FbCom.Parameters.Add('TOPICO', DbType.&String);
FbCom.Parameters['TOPICO'].Value := Codigo;
FbRead := FbCom.ExecuteReader;
while FbRead.Read do
begin
Linha := Tabela.NewRow;
Linha[0] := ''+
'<br><font face="Verdana" '+
'style="font-size:11px">'+
FbRead['USUARIO'].ToString+
'</font><br><font face="Verdana" '+
'style="font-size:10px">'+
'Posts: '+ FbRead['POSTS'].ToString+'</font>';
Linha[1] := FbRead['RESPOSTA'].ToString;
Tabela.Rows.Add(Linha);
end;
Result := Tabela;
end;
```

No código da listagem anterior, além de realizar uma consulta parametrizada e obter seus resultados, temos a formatação do conteúdo utilizando códigos *JavaScript* deixando os valores prontos para serem mostrados na página, no caso, onde as três variáveis da *procedure* estão recebendo valor.

Voltando para a *leitura.aspx*, onde mostraremos os resultados do procedimento na tela, atribua o código da **Listagem 7** no *Page\_Load*, lembrando que precisa também criar as três variáveis que serão passadas à *procedure* e uma variável que utilizaremos para armazenar o *Request* do código do tópico.

Observe que primeiramente passamos as variáveis a *procedure* e depois quando estão carregadas com as informações, são vinculadas em seus respectivos *Labels*. Se executar o sistema, poderá observar que o conteúdo do tópico será exibido na página, porém se houver alguma resposta ainda não será exibida.

Para exibir as respostas, criaremos uma consulta, que receberá como parâmetro o código do tópico e retornará um *DataTable* carregado com os registros de resposta ao tópico. Crie o método com o nome de *ListaRespostasTopicos* e atribua o código da **Listagem 8**. Crie também no *GridRespostas* duas colunas, alterando as informações para "USUARIO" e "RESPOSTA".

No código anterior, muito semelhante aos já apresentados, estamos criando as colunas do *DataTable*, realizando a pesquisa no banco de dados e adicionando os resultados ao *DataTable*. Para exibir os resultados na tela, no *Page\_Load* da página de leitura adicione o seguinte código:

```
GridRespostas.DataSource :=
ListaRespostasTopicos(Codigo);
GridRespostas.DataBind;
```

Se executar a aplicação já será possível visualizar as respostas ao tópico sendo que as mesmas foram anteriormente cadastradas diretamente na base de dados (**Figura 8**).

### Respondendo tópicos

Para possibilitar que o tópico seja respondido ou que novas respostas sejam adicionadas ao mesmo, criaremos uma nova página. Adicione um *Button* na tela

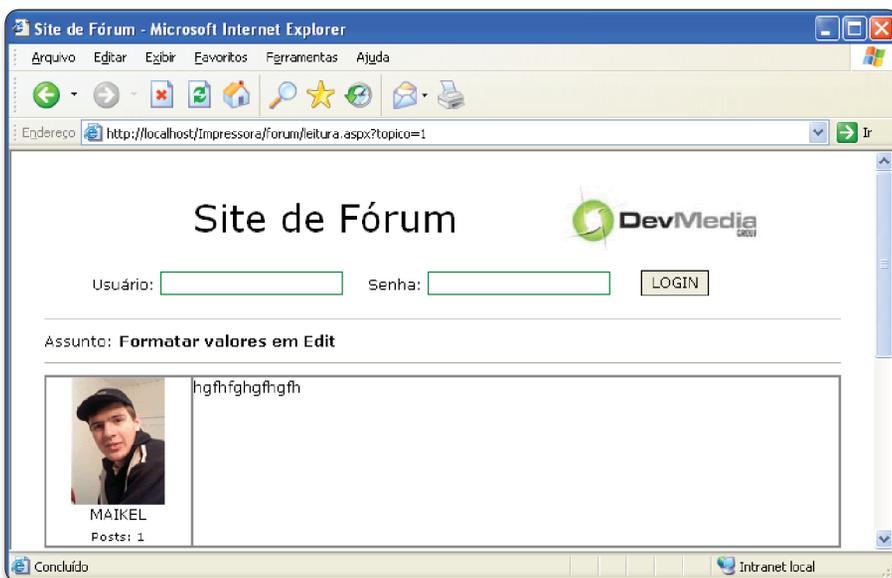


Figura 8. Visualizando tópicos e respostas

de leituras, alterando o nome para “Btn-Resposta” e a propriedade *Caption* para “Responder”. Com um duplo clique sobre o mesmo (evento *Click*) adicione o código da **Listagem 9**, que verifica se o usuário está ou não logado.

Na nova página que acabamos de criar, altere o nome para “responder.aspx” e organize os componentes conforme a **Figura 9**.

No *Page\_Load* adicione o código da **Listagem 10** que será responsável por verificar se o usuário está logado, para o caso do mesmo digitar a URL diretamente na barra de endereços no navegador e também fazer a verificação se o usuário possui algum tópico selecionado, além de verificar o status da conexão com a base de dados.

Perceba que não estamos fazendo nenhuma grande verificação nessa página, estamos apenas chamando as variáveis de sessão e testando se possuem algum valor. Caso alguma delas esteja em branco ou *nil*, quer dizer que ainda não foi criada, criando uma exceção, que será redirecionada para a página *index* da aplicação.

No botão de resposta do tópico, adicione ao evento *Click* o código da **Listagem 11** que será o responsável por cadastrar a resposta ao tópico selecionado, redirecionado após novamente para a listagem do tópico.

No código anterior, estamos criando novamente a variável *FbCom* (não se esqueça de declarar no *uses* o namespace *FirebirdSQL.Data.Firebird* e da página *Global.asax*), atribuímos o SQL de inserção do registro, criamos os parâmetros e após executamos o comando.

Agora que já temos um fórum montado, e que já estamos respondendo aos tópicos do mesmo, voltaremos até a *topicos.aspx*

#### Listagem 9. Verifica se o usuário está ou não logado

```
try
    if Session['USUARIO'].ToString <> '' then
        Response.Redirect('responder.aspx');
    except
        RegisterStartupScript('alerta',
            '<script>javascript:alert' +
            '(' + 'Efetue Login')';</script>');
    end;
```

#### Listagem 10. Verificando se o usuário esta logado

```
try
    VerificaConexao;
    if Session['USUARIO'].ToString <> '' then
        begin
            if Session['CODTOPICO'].ToString <> '' then
                end;
        except
            Response.Redirect('index.aspx');
        end;
```

#### Listagem 11. Código no botão de resposta de um tópico

```
procedure TWebForm1.Button1_Click(
    sender: System.Object; e: System.EventArgs);
var
    FbCom: FbCommand;
begin
    try
        FbCom := FbCommand.Create(
            'INSERT INTO RESPOSTAS (USUARIO, TOPICO, RESPOSTA) VALUES (?, ?, ?)', dados.CONEXAO);
        FbCom.Parameters.Clear;
        FbCom.Parameters.Add('USUARIO', DbType.&String);
        FbCom.Parameters.Add('TOPICO', DbType.&String);
        FbCom.Parameters.Add('RESPOSTA', DbType.&String);
        FbCom.Parameters['USUARIO'].Value := Session['CODIGO'].ToString;
        FbCom.Parameters['TOPICO'].Value := Session['CODTOPICO'].ToString;
        FbCom.Parameters['RESPOSTA'].Value := TxtResposta.Text;
        if FbCom.ExecuteNonQuery > 0 then
            begin
                Response.Redirect('leitura.aspx?topico=Session['CODTOPICO'].ToString);
            end;
        except
            on e:exception do
                RegisterStartupScript('erro', '<script>javascript:alert' +
                    '(' + 'Ocorreu o seguinte erro:\n' + e.Message + ')';</script>');
            end;
        end;
```

#### Listagem 12. Adicionando um novo tópico

```
try
    if Session['USUARIO'].ToString <> '' then
        Response.Redirect('new_topico.aspx');
    except
        RegisterStartupScript('alerta', '<script>'+
            'javascript:alert(' + 'Efetue Login')';</script>');
    end;
```

onde veremos como adicionar um novo tópico na área selecionada. No *BtnNovo-Topico* que já adicionamos anteriormente, atribua o código da **Listagem 12**.

### Cadastrando novos tópicos

Crie uma nova página e salve-a como “new\_topico.aspx”, que será utilizada para o cadastro de novos tópicos por usu-



**Listagem 13.** Page\_Load da new\_topico.aspx

```
try
  VerificaConexao;
  if Session['USUARIO'].ToString <> '' then
  begin
    if Session['CODAREA'].ToString <> '' then
    end;
  except
    Response.Redirect('index.aspx');
  end;
```

**Listagem 14.** Codificando o botão de cadastro de um novo tópico

```
procedure TWebForm1.BtnAddTopico_Click(
  sender: System.Object; e: System.EventArgs);
var
  FbCom: FbCommand;
begin
  try
    FbCom := FbCommand.Create(
      'INSERT INTO TOPICOS ( '+
      'AREA, USUARIO, TITULO, CONTEUDO) '+
      'VALUES (?, ?, ?, ?)', dados.CONEXAO);
    FbCom.Parameters.Clear;
    FbCom.Parameters.Add('AREA', DbType.&String);
    FbCom.Parameters.Add('USUARIO', DbType.&String);
    FbCom.Parameters.Add('TITULO', DbType.&String);
    FbCom.Parameters.Add('CONTEUDO', DbType.&String);
    FbCom.Parameters['AREA'].Value := Session['CODAREA'].ToString;
    FbCom.Parameters['USUARIO'].Value := Session['CODIGO'].ToString;
    FbCom.Parameters['TITULO'].Value := TxtAssunto.Text;
    FbCom.Parameters['CONTEUDO'].Value := TxtDescricao.Text;
    if FbCom.ExecuteNonQuery > 0 then
    begin
      Response.Redirect('topicos.aspx?area=Session['CODAREA'].ToString);
    end;
  except
    on e: Exception do
      RegisterStartupScript('erro', '<script>javascript:alert' +
        ' (''Ocorreu o seguinte erro:\n' + e.Message+'');</script>');
    end;
  end;
```

ários. Atribua no *Page\_Load* da mesma o código da **Listagem 13**, após configure os componentes na tela, conforme demonstrado na **Figura 10**.

Após definida toda a estrutura da página de cadastro de novos tópicos, faremos a codificação do botão responsável por cadastrar as informações do tópico, e para isso atribua ao evento *Click* do *BtnAddTopico* o código da **Listagem 14**.

Da mesma forma que nas outras páginas, declare o namespace *FirebirdSQL.Data.Firebird* e adicione referência à unit da *Global.asax* onde está configurada a conexão do projeto. Declaramos a variável *FbCom*, que será usada para a criação do comando SQL de inserção do registro.

Se o comando for executado corretamente, o usuário será redirecionado para a listagem do tópico cadastrado. Em caso de algum problema durante o cadastro, usaremos o tratamento de exceções para exibir a falha.

É interessante que você faça a verificação e validação dos campos, não permitindo que o usuário salve registros com campos sem informação. Para isso, você poderá usar os componentes de validação (*RequireFieldValidator*, *RegularExpressionValidator* etc), entre outras formas possíveis, ou também por controle manual.

**Nota:** Neste exemplo não foi abordado o cadastro de usuários, bem como a manutenção dos mesmos, além da manutenção de áreas e tópicos do fórum, sendo muito simples de ser feito. Você poderá fazer o download completo deste exemplo no site da DevMedia.

## Conclusão

Neste artigo vimos como desenvolver de forma simples um fórum de discussão, contemplando desde os controles de usuários, listagem de áreas, tópicos e respostas. Até a próxima. ●

**ClubeDelphi PLUS** [www.devmedia.com.br/dubedelphi/portal.asp](http://www.devmedia.com.br/dubedelphi/portal.asp)

Acesse agora o mesmo o portal do assinante ClubeDelphi e assista a uma vídeo aula de Luciano Pimenta, que mostra como trabalhar com Validators no ASP.NET.

[www.devmedia.com.br/artides/viewcomp.asp?comp=2341](http://www.devmedia.com.br/artides/viewcomp.asp?comp=2341)



**Figura 9.** Organização dos objetos na página de respostas



**Figura 10.** Organização dos objetos na página de novo tópico