Delphi - Arquivo INI para conexão com o Firedac

Caro amigo leitor,

Neste artigo irei tratar de um assunto utilizado na distribuição de praticamente todo sistema, o uso de arquivos de configuração do tipo .INI. Usarei como ferramenta o Delphi XE 5 e como conexão de dados o Firedac. O Firedac é uma biblioteca de acesso universal a dados a diferentes tipos de Bancos de Dados. (Este artigo será com base no SQL Server e Firebird). Podemos dizer que o FireDAC possui uma engine otimizada para acesso a dados e disponível a partir do **Delphi XE 3**. Para maiores informações recomendo a leitura do artigo **do mês 06/2013 chamado "Firedac" de nosso colunista Luciano Pimenta**. Iremos criar uma classe para a leitura do arquivo .INI para atribuirmos os parâmetros de conexão para nosso componente "TFDConnection" dentro de um "DataModule".

Criando o Arquivo .INI

O **formato de arquivo INI** é um padrão informal para **arquivos de configuração** para algumas **plataformas** ou software. Arquivos INI são **arquivos de texto** simples com uma estrutura básica composta de "seções" e "**propriedades**". A partir destas configurações conseguimos utilizá-la dentro do delphi com uma enorme facilidade.

Crie uma pasta para organizar o exemplo e dentro da mesma, crie um arquivo vazio com estensão INI. Ver Imagem 01.

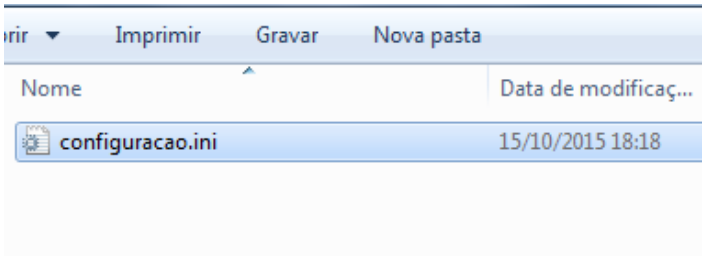

Figura 01: Criando o Arquivo INI.

Iremos criar duas seções, sendo uma para o Banco FireBird e outro para o SQL Server com algumas propriedades. Ver Imagem 02.

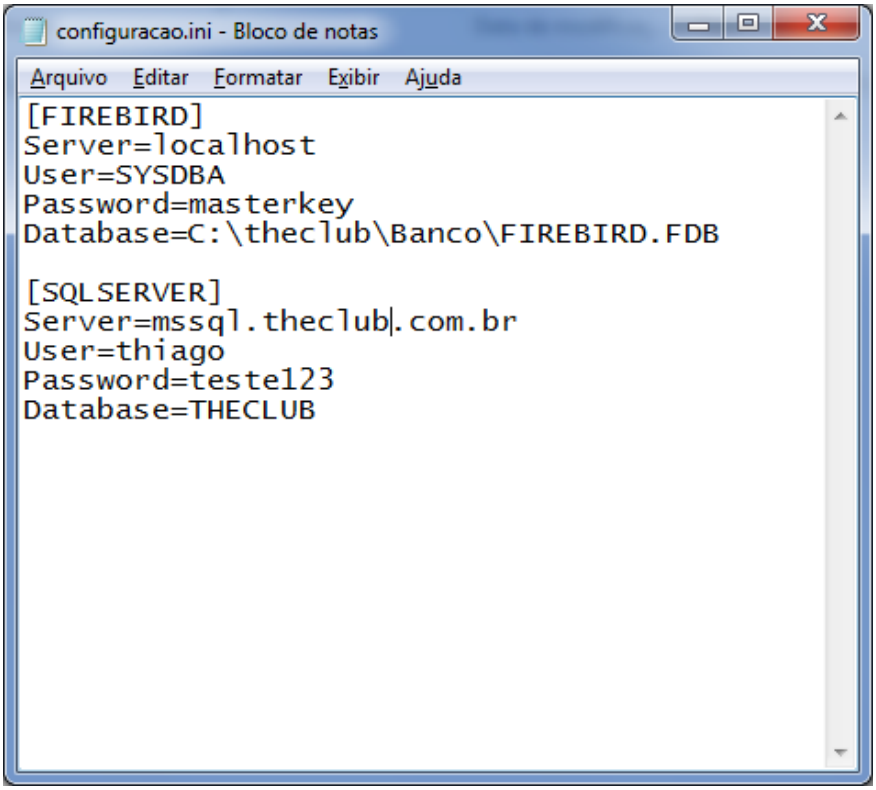

Figura 02: Propriedades.

Configurações do Firebird

Podemos conferir as configurações detalhadas na Tabela 01.

[FIREBIRD]	Nome da seção, indicando o Banco de dados FireBird.
Server	Servidor onde está armazenado o Banco de Dados. (No meu caso defini Localhost, que significa que está no mesmo computador onde se encontra a aplicação)
User	SYSDBA (Usuário Padrão)
Password	Masterkey (Senha Padrão)
DataBase	Caminho onde o Banco está localizado.

Tabela 01.

Configurações do SQL Server

Podemos conferir as configurações detalhadas na Tabela 02.

[SQLSERVER]	Nome da seção, indicando o Banco de dados SQL Server.
Server	Servidor onde está armazenado o Banco de Dados. (No meu caso defini um servidor ficticio)
User	Thiago (usuário ficticio)
Password	Teste123 (senha ficticia)
DataBase	Apenas o nome da Base de Dados

Tabela 02.

Dentro da mesma pasta criada anteriormente iremos implementar nosso exemplo desenvolvido em Delphi, ou seja, teremos o executável exatamente na mesma pasta onde está localizado o arquivo de configuração.

Criando a classe para leitura do arquivo INI

Para quem estiver utilizando a versão do Delphi XE 5, crie um projeto do início adicionado ao projeto uma Unit vazia.

```
unit UnFuncoes;
interface
```

Declaramos algumas “units” necessárias na cláusula “Uses”.

```
uses IniFiles, IWSystem, SysUtils;
```

Na declaração da Classe teremos apenas o método estático “LerIni”, contendo três parâmetros, sendo os dois primeiros indicando a seção/propriedade e o último um valor padrão vazio que utilizaremos no método “ReadString” da classe “TiniFile”.

```
type
  TFuncoes = class
  Public
     class function LerIni(Chave1, Chave2: String; ValorPadrao: String = ''): String;
static;
  end;

implementation

class function TFuncoes.LerIni(Chave1, Chave2: String; ValorPadrao: String = ''):
String;
var
  Arquivo: String;
  FileIni: TIniFile;
begin
  Arquivo := gsAppPath + gsAppName + '.ini';
  result := ValorPadrao;
  try
    FileIni := TIniFile.Create(Arquivo);
```

```
      if FileExists(Arquivo) then
        result := FileIni.ReadString(Chave1, Chave2, ValorPadrao);
    finally
      FreeAndNil(FileIni)
    end;


end;


end.
```

Este método é bem simples, a variável "Arquivo" receberá o Path absoluto da aplicação, com o auxílio das funções gsAppPath (para recuperar o Path) concatenando com o nome do arquivo "configuracao.ini". Já a variável "FileIni" criará uma instância de um objeto encapsulando no construtor o arquivo INI criado anteriormente e por final usaremos o método "ReadString" para a seção/propriedade indicada retornando o valor.

Carregando os Parâmetros de Conexão

Usaremos um DataModule para organização dos componentes utilizados. Na palheta "FireDAC" Adicione dois TFDConnection (FDConnectionFirebird e FDConnectionSQLServer). Na "FIreDAC Links" adicionaremos mais dois componentes, sendo o "TFDPhisMSSQLDriverLink e o TFDPhisFBDriverLink" respectivamente para os bancos SQLServer e Firebird. Por último adicione um "FDGUIxWaitCursor" da "FireDAC UI".

Importante: Todos estes componentes são necessários para o funcionamento de uma aplicação FIREDAC. Para maiores detalhes sobre eles recomendo a leitura do artigo supracitado.

Podemos conferir maiores detalhes na Imagem 03.

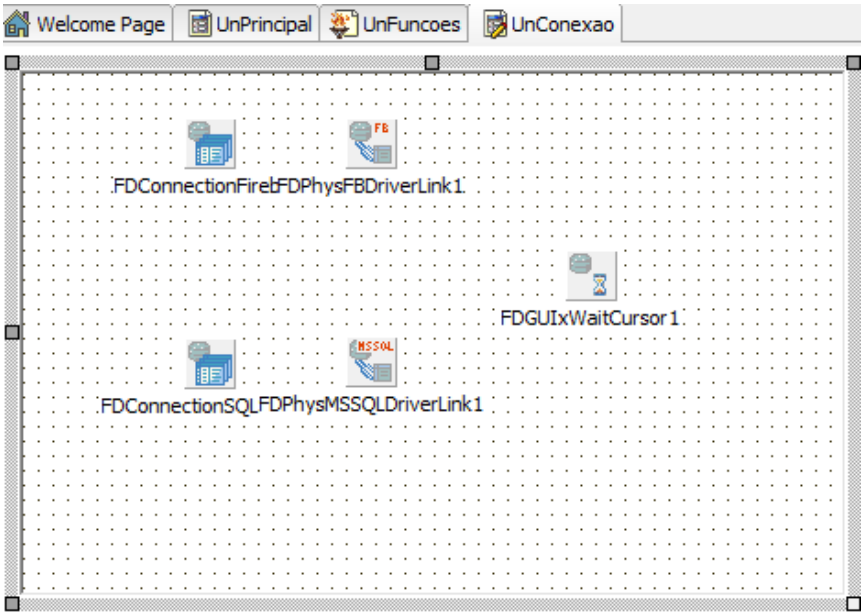

Figura 03. Configurando os componentes necessários para Conexão.

```
unit UnConexao;
...
```

Usaremos as units abaixo, sendo a primeira para acesso a classe criada e a segunda para termos acesso ao método ShowMessage.

```
uses unFuncoes, Dialogs;

procedure TDataModule1.DataModuleCreate(Sender: TObject);
begin
 try
    with FDConnectionFirebird do
      begin
        Params.Clear;
        Params.Values['DriverID']  := 'FB';
        Params.Values['Server'] := TFuncoes.LerIni('FIREBIRD','Server');
        Params.Values['Database'] := TFuncoes.LerIni('FIREBIRD','Database');
        Params.Values['User_name'] := TFuncoes.LerIni('FIREBIRD','User');
```

```
          Params.Values['Password'] := TFuncoes.LerIni('FIREBIRD','Password');
          Connected := True;
        end;
    except
        ShowMessage('Ocorreu uma Falha na configuração no Banco Firebird!');
    end;


    try
      with FDConnectionSQLServer do
        begin
          Params.Clear;
          Params.Values['DriverID']  := 'MSSQL';
          Params.Values['Server'] := TFuncoes.LerIni('SQLSERVER','Server');
          Params.Values['Database'] := TFuncoes.LerIni('SQLSERVER','Database');
          Params.Values['User_name'] := TFuncoes.LerIni('SQLSERVER','User');
          Params.Values['Password'] := TFuncoes.LerIni('SQLSERVER','Password');
          Connected := True;
        end;
    except
        ShowMessage('Ocorreu uma Falha na configuração no Banco SQL Server!');
    end;
  end;
  end.
```

No evento OnCreate do DataModule poderemos carregar todas as configurações necessárias. Usaremos um bloco try/Except junto com o Método “Params.values” para alimentarmos nossos componentes.

Tanto para o Firebird quanto para o SQl Server utilizaremos os parâmetros “Server”, “DataBase”, “User_name” e “Password”. Divergindo apenas o valor “DriverID”.

Prontinho, caso não ocorra nenhum erro de configuração poderemos partir para a utilização deste “Datamodule” nos nossos formulários, mas aí fica para o próximo artigo.

Conclusões

Pudemos aprender neste artigo o uso correto de arquivos de configuração em parceria com a Biblioteca de componentes FireDAC. Fica aí a dica para quem deseja implementar este recursos em seus sistemas.

Um forte abraço e até o mês que vem!