

Nesta seção você encontra artigos intermediários sobre Delphi Win32 e Delphi .NET

Diretivas de Compilação

Aprenda na prática este poderoso recurso da linguagem Delphi



Ricardo C. Boaro

(rboaro@aquasoft.com.br)

trabalha com desenvolvimento de sistemas em Delphi há mais de 10 anos e PocketStudio há 3 anos. Atualmente é gerente de informática na Di Hellen Indústria de Cosméticos, e atua como instrutor certificado Borland na Aquasoft Tecnologia da Informação parceira da Borland, em Porto Alegre – RS. Borland Instrutor, Delphi 7, 2007 e Certified.

Nesse artigo desvendaremos os “mistérios” das diretivas de compilação, para muitos considerada uma maneira complicada de desenvolver e manter os fontes do sistema. Antes de entrar no mérito da questão e apresentar-lhe a minha visão sobre diretivas, conceitos, utilização e por que utilizá-las, vamos entendê-las.

O que são diretivas de compilação?

As *Diretivas de Compilação* são baseadas na existência de condições para o compilador. São símbolos condicionais que trabalham como variáveis *Booleanas*, ou seja, *True* ou *False*. Qualquer condição válida é tratada como *False* até que seja definido o inverso. Podemos usar *\$DEFINE* ou *\$UNDEF* para especificar um conjunto de símbolos verdadeiros.

Podemos definir os blocos de códigos válidos identificando as condições válidas em *Project>Options*, na aba *Directories/Conditionals* no item *Conditional defines*.

As diretivas *\$IFDEF*, *\$IFNDEF*, *\$IF*, *\$ELSEIF*, *\$ELSE*, *\$ENDIF*, e *\$IFEND*, permitem ao compilador omitir blocos de código, baseado no status da condição *\$IF* e *\$ELSEIF*. Algumas características das diretivas são listadas a seguir:

- Podem ser aninhadas, em até 32 níveis;
- Seu identificador deve iniciar com letras seguidas por uma combinação de letras, números e *underscores*;
- Podem ter qualquer tamanho, mas apenas os primeiros 255 caracteres serão considerados;

Logicamente que não teremos um identificador maior que 255 caracteres. O Delphi utiliza diretivas de compilação em seu fonte para identificar muitas variáveis, na **Listagem 1** segue alguns exemplos da utilização no próprio Delphi.

A **Listagem 1** é o início da declaração da *Unit classes* do Delphi. Perceba que após a seção *Interface* existem duas diretivas

identificando quais *Units* serão utilizadas na unidade *Classes* de acordo com o sistema operacional.

Nesse exemplo estão sendo utilizadas as diretivas `{IFDEF Condição}` e `{ENDIF}`, elas são as mais comuns e fáceis de utilizarmos por serem análogas ao `IF..Then`. O Delphi utiliza diretivas para identificar a sua versão da mesma forma que as utiliza para identificar se estamos desenvolvendo uma aplicação WIN32 ou .NET.

Existem muitos tipos de diretivas e precisaria muito mais que um artigo para estudarmos cada uma delas, e qual a sua funcionalidade. Ao invés disso criaremos um exemplo para abstrair da nossa realidade a aplicabilidade.

Onde podemos utilizá-las?

Imagine a seguinte situação: Possuímos um ERP completíssimo, com um cadastro de clientes excepcional, controle de estoque perfeito, financeiro totalmente integrado à contabilidade, enfim para nós ele é o ERP perfeito. Mas, sempre haverá um cliente querendo alterá-lo e adaptá-lo a sua empresa, e mais, cada cliente terá a sua particularidade. Por mais que tenhamos um sistema “completo”, as empresas nos mais variados ramos, trabalham da forma mais variada possível, jamais existirá uma empresa com a mesma maneira de trabalhar que outra.

Cada uma vive a sua realidade, sendo assim, tem sua necessidade específica. Como resolver essa situação? Manter vários sistemas? Fazer a famosa POG, Programação Orientada a Gambiarra? Claro que não, vários sistemas seriam difíceis de manter e elevariam muito o custo de manutenção. POG está descartada, pois o sinônimo de POG com certeza são problemas futuros. Os famosos remendos podem resolver uma determinada situação momentaneamente, mas com o passar do tempo irão causar novos problemas e isso acabará sendo uma bola de neve até que o sistema se transforme em uma fonte de estresse ilimitada.

Sendo assim meus amigos, vamos nos valer das muitas facilidades oferecidas pelo Delphi para facilitarmos nosso trabalho e melhorarmos a qualidade dos nossos sistemas. Nessa situação

Diretivas de Compilação serão a nossa salvação, pois são fáceis de implementar e fáceis de manter.

Vamos criar um exemplo para fixarmos a utilização das mesmas. As diretivas estão disponíveis em todas as versões do Delphi em nosso exemplo trabalharemos com o Delphi 7 criando uma aplicação para Win32.

Inicie uma nova aplicação em *File|New|Application*. A idéia inicial é criarmos um exemplo simples, mas que será muito útil para entendermos o que podemos alterar trabalhando com diretivas.

Digamos que temos o *Cliente A* que possui uma base de dados Paradox, pois é uma empresa pequena e trabalha apenas com base de dados local. Desenvolvemos para ele um cadastro de clientes simples apenas com os dados necessários para

que ele possa consultar suas informações quando precisar entrar em contato com o mesmo, sendo assim utilizamos os componentes da paleta BDE.

Por outro lado temos o *Cliente B* que possui uma empresa considerável que usa os dados dos clientes para emitir notas fiscais e controlar seu limite de compra. Para complicar um pouco mais, ele tem vários terminais acessando o cadastro de clientes ao mesmo tempo, consultando, utilizando os dados para emitir notas fiscais etc. Nesse cenário precisamos trabalhar com uma base de dados mais robusta, e melhor preparada para o uso em rede. Para isso utilizaremos o SQL Server Express, e acessaremos a base de dados utilizando os componentes da paleta *dGGo*.

Listagem 1. Exemplo de diretivas de compilação

```
unit Classes;
{$R-,T-,X+,H+,B-}
{$IFDEF MSWINDOWS}
{ ACTIVEEX.HPP is not required by CLASSES.HPP }
(*$NOINCLUDE ActiveX*)
{$ENDIF}
{$IFDEF LINUX}
{$DEFINE _WIN32}
{$ENDIF}
{$IFDEF MSWINDOWS}
{$DEFINE _WIN32}
{$ENDIF}

interface
{$IFDEF MSWINDOWS}
uses Windows, Messages, SysUtils, Variants, TypInfo, ActiveX;
{$ENDIF}
{$IFDEF LINUX}
uses Libc, SysUtils, Variants, TypInfo, Types;
{$ENDIF}
```

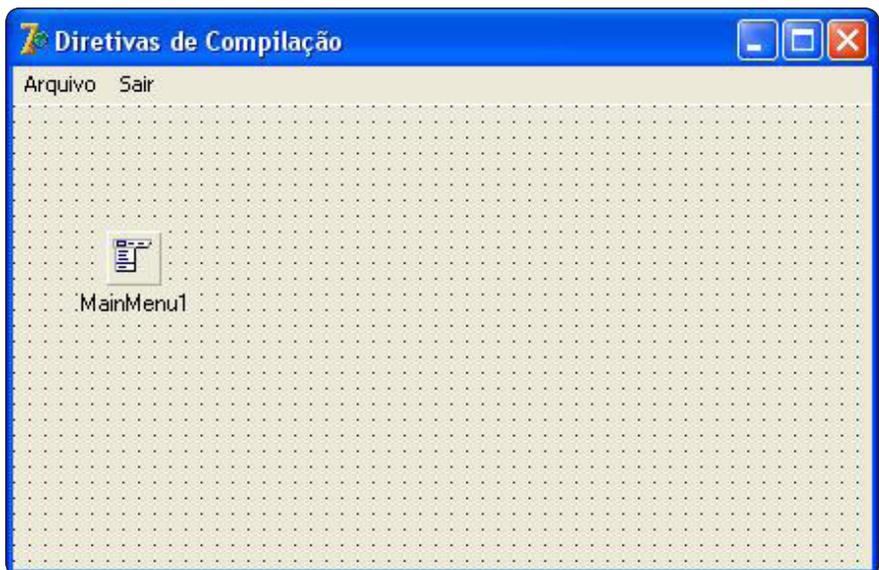


Figura 1. Exemplo de tela principal

Construindo o Exemplo

Vamos nos basear no exemplo exposto na explicação acima, temos o *ClienteA*, pequeno com um sistema de cadastro de clientes mono-usuário, que irá funcionar apenas no servidor. Já o *ClienteB* possui uma estrutura robusta e tem vários acessos a base de dados simultaneamente. Partindo desse princípio vamos criar uma tela principal com um componente *MainMenu* e configurá-lo como mostrado na **Figura 1**.

Após a configuração do componente *MainMenu*, vamos adicionar um novo formulário em nosso projeto que será o cadastro de clientes. Devemos salvá-lo como "UClientes.pas", e na propriedade *Name* chamá-lo de "FrmClientes".

Seu *Caption* deve ser configurado com "Cadastro de Clientes". Estou partindo do princípio que já temos a base de dados configurada, não vou entrar em detalhes aqui sobre a criação do BD no SQL Server e no Paradox.

Para manipularmos as tabelas que possuem os mesmos campos, montei essa estrutura propositalmente para que possamos simplificar o exemplo e facilitar o entendimento do exercício proposto. A tela de clientes deve ser configurada com *DBEdit's* e *Button's* como mostrado na **Figura 2**. O código completo dos botões superiores você encontra **Listagem 2**.

Feito isso vamos adicionar um *Data Module* ao nosso projeto e salvá-lo com

o nome de "uDmBase.pas" e nomeá-lo como "DmBase". Nele iremos adicionar um componente *Table*("TblClientes") da paleta *BDE*, que será ligado a nossa tabela clientes do *Paradox* através de um *alias*. Um componente *ADOConnection*, que deve ser conectado ao servidor SQL Server, e um *ADOTable*("AdoTblClientes") ligado ao *AdoConnection* e na tabela de Clientes do SQL Server. Na **Figura 3** temos o *DmBase* configurado.

Feito isso temos a estrutura do nosso programa montada. Lembrem-se que o *ClienteA* deve trabalhar com a tabela de clientes ligada ao *Paradox* e o *ClienteB* com a tabela de clientes ligada ao SQL Server. A idéia é utilizarmos a mesma tela de cadastro de clientes para os dois clientes, mudando apenas a base que será utilizada conforme a diretiva informada.

Vamos colocar em prática nossa idéia. Percebam na **Listagem 1** que estou chamando o *Insert*, *Edit* etc., diretamente de um *DataSource*. Esse *DataSource*("dtClientes") deve ser adicionado ao formulário de clientes.

Estamos chamando os métodos de inclusão, alteração etc. diretamente da propriedade *DataSet*, assim não precisamos saber se o *DataSource* está conectado ao *Paradox* ou ao SQL Server. A única coisa que precisamos fazer é mudar seu *DataSet* para apontar para uma base ou outra conforme a compilação que fizermos, utilizando a diretiva do *ClienteA* ou do *ClienteB*.

Para informarmos ao compilador que utilizaremos uma diretiva ou outra vamos em *Project>Options*, na aba *Directories/Conditionals* e logo após em *Conditional defines*. Como explicado no

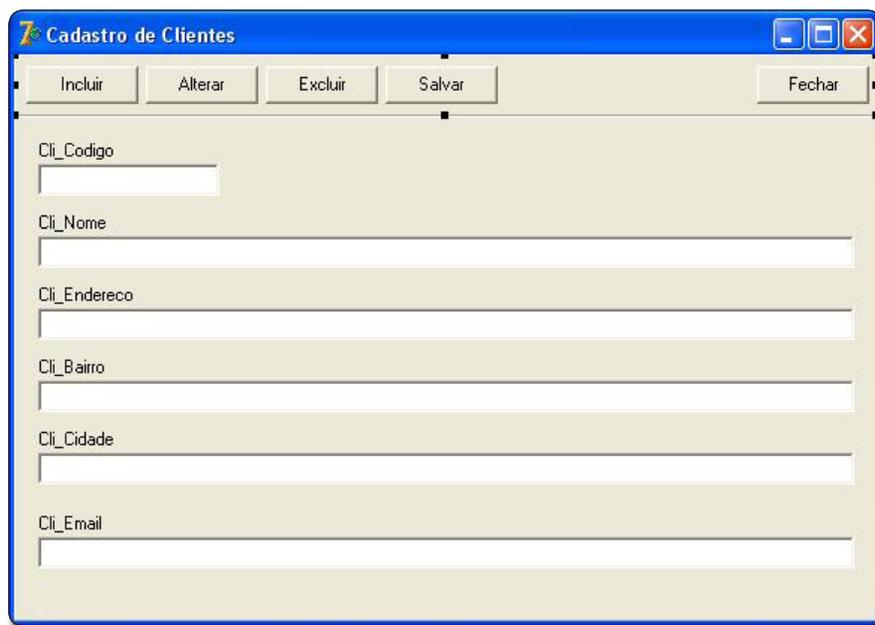


Figura 2. Exemplo de tela de clientes

Listagem 2. Código dos botões

```
procedure TFrmClientes.BtnInserirClick(Sender: TObject);
begin
  DtClientes.DataSet.Insert;
end;
procedure TFrmClientes.BtnAlterarClick(Sender: TObject);
begin
  DtClientes.DataSet.Edit;
end;
procedure TFrmClientes.BtnExcluirClick(Sender: TObject);
begin
  DtClientes.DataSet.Delete;
end;
procedure TFrmClientes.BtnSalvarClick(Sender: TObject);
begin
  DtClientes.DataSet.Post;
end;
procedure TFrmClientes.BtnFecharClick(Sender: TObject);
begin
  Close;
end;
```

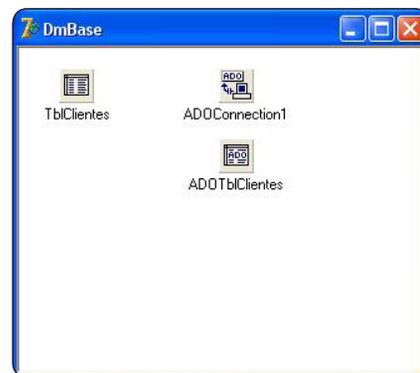


Figura 3. Data Module já configurado

início do artigo para iniciarmos vamos definir a diretiva *ClienteA* para ficar bem intuitivo, como mostra a **Figura 4**.

O próximo passo é informarmos no código do projeto como ele irá tratar as condições, por isso codifique o evento *OnCreate* do formulário de clientes conforme a **Listagem 3**.

Note que estou utilizando as duas diretivas que vamos trabalhar em nosso exemplo e vejam que é simples para trocarmos de uma base para outra. Para incrementar um pouco mais nosso exemplo no evento *OnCreate* do formulário principal, vamos digitar o código da **Listagem 4**.

Vamos rodar a aplicação com a diretiva do *ClienteA* ativa e ver o resultado. Logo no formulário principal podemos ver a mudança. Como mostrado na **Figura 5** o fundo do formulário foi modificado para a cor azul. Feche a aplicação, modifique a diretiva para *ClienteB* e execute novamente o sistema. Note que novamente a cor é modificada na **Figura 6**.

Logicamente aqui percebemos apenas as alterações na cor e no *Caption* do formulário, mas acessando o cadastro de clientes podemos incluir registros tanto para o *ClienteA* acessando o Paradox como para o *ClienteB* acessando o SQL Server. O importante é que você entenda que podemos alterar qualquer configuração no projeto trabalhando com diretivas.

Considerações Importantes

As diretivas de compilação só serão atualizadas quando trocarmos da diretiva *ClienteA* para *ClienteB* fazendo um *Buid* da aplicação. Se apenas compilarmos, a diretiva não é aceita. Um dos segredos da compilação é sempre

Listagem 3. Código do evento OnCreate do formulário

```
procedure TfrmClientes.FormCreate(Sender: TObject);
begin
  {$IFDEF ClienteA}
    DtClientes.DataSet := DmBase.Tb1Clientes;
  {$ENDIF}
  {$IFDEF ClienteB}
    DtClientes.DataSet := DmBase.ADOtb1Clientes;
  {$ENDIF}
end;
```

Listagem 4. Evento OnCreate do formulário principal

```
procedure TfrmDiretivas.FormCreate(Sender: TObject);
begin
  {$IFDEF ClienteA}
    Application.Title:= 'Cliente A';
    FrmDiretivas.Caption:= 'Sistema de controle de clientes do Cliente A';
    FrmDiretivas.Color:= C1Blue;
  {$ENDIF}
  {$IFDEF ClienteB}
    Application.Title:= 'Cliente B';
    FrmDiretivas.Caption:= 'Sistema de controle de clientes do Cliente B';
    FrmDiretivas.Color:= C1Red;
  {$ENDIF}
end;
```

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

obrebem
Tecnologia

ClubeDelphi PLUS www.devmedia.com.br/clubedelphi/portal.asp

Acesse agora o mesmo o portal do assinante ClubeDelphi e assista a uma video aula de Ricardo Boaro que mostra como trabalhar com diretivas de compilação.

<http://www.devmedia.com.br/articles/viewcomp.asp?comp=7248&hl=diretivas>

ClubeDelphi PLUS www.devmedia.com.br/clubedelphi/portal.asp

Acesse agora o mesmo o portal do assinante ClubeDelphi e assista a uma video aula de Guinther Pauli que mostra como trabalhar com diretivas de compilação.

<http://www.devmedia.com.br/articles/viewcomp.asp?comp=530&hl=diretivas>

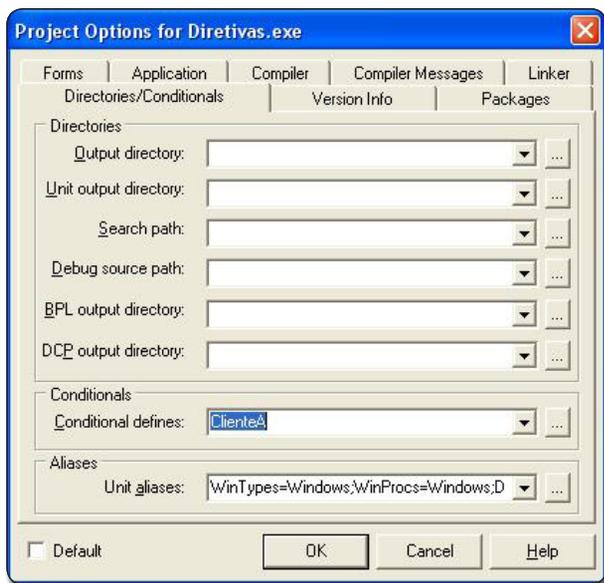


Figura 4. Definindo a diretiva de compilação

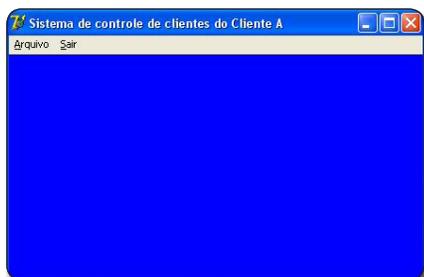


Figura 5. Formulário principal em azul devido à diretiva ClienteA

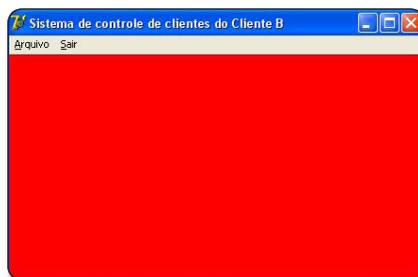


Figura 6. Formulário principal em vermelho devido à diretiva ClienteB

trabalharmos com *Build* ao final de cada alteração de condição.

Outro detalhe importante sobre diretivas é entendermos que os códigos que estão dentro das diretivas que não fazem parte da condição, ou seja, quando definimos que a diretiva de compilação ativa é a *ClienteA*, todos os códigos que estiverem dentro da condição *ClienteB* serão ignorados, e não serão carregados no executável.

Ao compilarmos podemos notar, pelos indicadores azuis, que o fluxo do programa não passará pelo código referente ao *ClienteA*. (ver Figura 7)

Dessa forma mesmo que houvesse um erro de sintaxe dentro da diretiva *ClienteA* não seria notado, nem mesmo o *Code Completion* funciona bem dentro de seções de diretivas desativadas.

Conclusão

Neste artigo entendemos o funcionamento básico das diretivas de compilação. É importante aproveitarmos ao máximo os recursos da ferramenta para podermos ganhar o máximo em produtividade e redução dos custos de nossos softwares.

O exercício criado foi simples, mas acredito que foi o suficiente para começarmos a trabalhar com diretivas. Feito isso liberem a sua imaginação e tirem proveito de todas as formas imagináveis. Um grande abraço a todos e bons códigos. ●

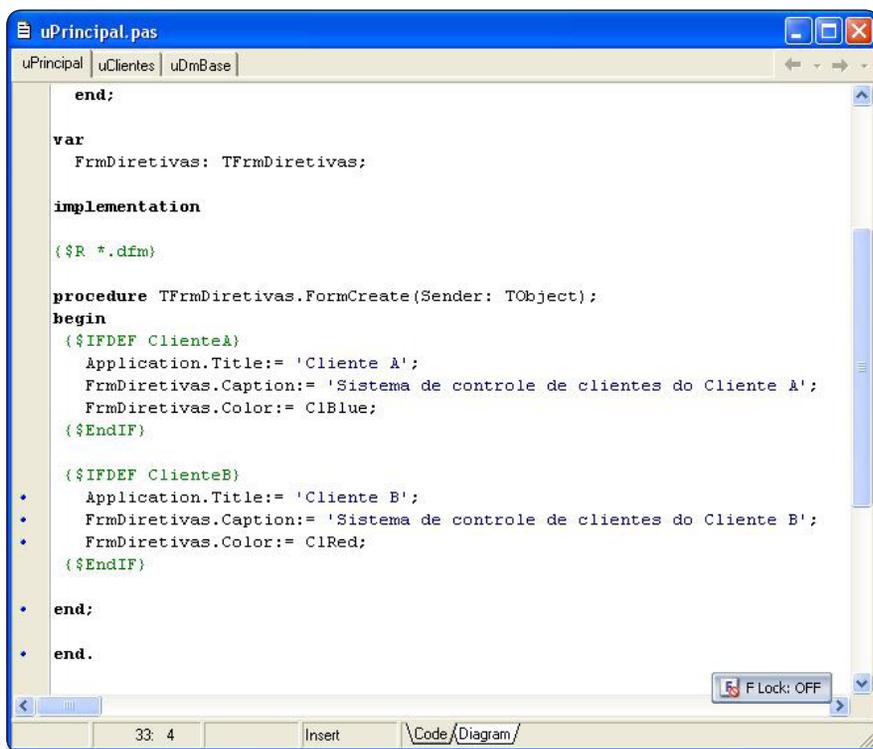


Figura 7. Indicação de fluxo de código