

Live Templates



ADRIANO SILVEIRA

(adr_silveira@yahoo.com.br)

é desenvolvedor certificado Borland Delphi 7. Bacharel em Sistemas de Informação pela Faculdade Cotemig em Belo Horizonte, trabalha como gerente de TI na Fácil Informática. Utiliza Delphi

como IDE há 7 anos. Trabalha com projetos cliente-servidor, multi-camadas, webservices e na construção de frameworks. Detém o título de campeão mineiro de robótica, além de ter ministrado cursos oficiais na TDS Borland Learning Partner e em cursos técnicos.

Desde a versão 3 do Delphi, podemos utilizar o recurso de *Code Templates*, que forneciam blocos de código comumente utilizados em nossos projetos. Contudo, eram bastante limitados se resumindo a fornecer uma cópia de um bloco de código “colado” no editor.

Além disso, o único recurso extra existente era a possibilidade de indicar o ponto onde o cursor seria posicionado no momento da utilização do *template* de código, através do character “|” (*pipe*). No Delphi 2006, o conceito de *Code Templates* foi alterado para *Live Templates*, isso porque diversos recursos foram adicionados, como veremos neste artigo.

Utilizando Live Templates

Para utilizar um *Live Template* pressione CTRL+J no ponto do código onde deseja inseri-lo. A lista com todos os *Live Templates* será exibida. Selecione o *template* desejado e pressione ENTER. Outra maneira de utilizar um *Live Template* é selecionar o menu *View>Templates* (**Figura 1**).

Para adicionar um *Live Template* através da janela, dê um duplo clique sobre o *template* desejado. Ao adicionar um *template* ao código já podemos perceber o primeiro novo recurso: é possível se movimentar entre pontos de código utilizando a tecla TAB.

Por exemplo, adicione o *template Class* e pressionando TAB, e veja que é possível alternar entre a declaração do nome da classe e sua ancestral.

CLUBEDELPHI PLUS!

Acesse agora o mesmo o portal do assinante ClubeDelphi e assista a uma vídeo-aula de Guinther Pauli que mostra como criar um *Live Template* para acesso pronto a um banco de dados Firebird.

<http://www.devmedia.com.br/articles/visualizacomponente2.asp?comp=2510>

Grupos de Templates

Devido ao fato do Delphi 2006 possuir múltiplas personalidades, os *Live Templates* foram agrupados de acordo com a linguagem (Delphi, C#, C++, etc.). Além desses, existe também o grupo IDE, fornecendo um único *Template*, utilizado na construção de novos.

Outra novidade interessante é o recurso de automação de escrita de código. Por exemplo, insira o *template* "for" em algum ponto do código. Repare que o ponto onde o cursor se inicia é exatamente o nome da variável que será utilizada na iteração.

Altere seu nome para "Aux" e pressione ENTER. Observe que o próprio IDE declara a variável *Aux*. Esse recurso é válido para outros *Live Templates* como *for in* e *try/finally*.

Conhecendo os Live Templates

Nesse ponto serão apresentados e descritos os *Live Templates* disponíveis para o Delphi. A maioria desses possui correspondentes para as linguagens C# e C++, como por exemplo: *case* do Delphi com *switch* do C# e do C++. Veja na **Tabela 1** a descrição dos *Lives Templates* presentes no Delphi 2006.

Código dos Live Templates

Em algumas situações, possivelmente, nenhum dos *Live Templates* seja compatível com alguma necessidade de código, assim é possível criar nossos próprios *templates*. Todos são escritos no padrão XML, como podemos ver o exemplo da **Listagem 1**.

Listagem 1. Exemplo de Live Templates

```
<?xml version="1.0" encoding="utf-8" ?>
<codetemplate
  xmlns="http://schemas.borland.com/Delphi/2005/codetemplates"
  version="1.0.0">
  <template name="tryf" surround="true" invoke="manual">
    <description>
      try finally
    </description>
    <author>
      Borland Software Corporation
    </author>
    <code language="Delphi" context="methodbody"
      delimiter="|" "><![CDATA[try
|selected||*||end|
finally
end;
]]>
    </code>
  </template>
</codetemplate>
```

Como não é o objetivo do artigo, não entraremos nos detalhes de padrões XML, concentrando-se apenas no que diz respeito à criação de *Live Templates*, pré-supondo que seja de conhecimento do leitor as regras e padrões XML.

Conforme visto na listagem anterior, a primeira linha contém o cabeçalho do arquivo. Após, a primeira *tag* é a *codetemplate*, que delimita o início e o término do *Live Template* e contém os seguintes atributos:

- *xmlns*: contém o *namespace* do XML;
- *version*: versão do *Live Template*.

Todo *Live Template* contém essa *tag* com os respectivos atributos. A primeira *tag* interna do *Live Template* é a *<tem-*

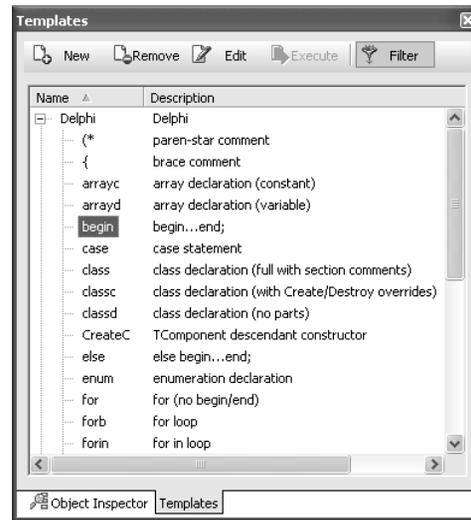


Figura 1. Janela Templates do Delphi 2006

plate> que define os seguintes atributos globais:

- *name*: nome do *Live Template*;
- *surround*: define se será disponibilizado na lista de *Surrounds* (veja box);
- *invoke*: define a forma como o *Live Template* será "invocado", que pode ser:
 - *auto*: pressionando SPACE ou TAB;
 - *manual*: pressionando TAB;
 - *none*: pressionando CTRL+J, CTRL+SPACE ou usando a janela *Template Viewer*.

A *tag description* é opcional e define uma descrição do *Live Template*. A *author* também é opcional e logicamente, define o criador do *Live Template*. A *tag* que realmente contém o código do *Live Template* é a *code*, com os seguintes atributos:

- *language*: obrigatório. Define a linguagem que o *Live Template* pertence, podendo ser: Delphi, CSharp, C, XML, HTML, Jsharp, IDE, SQL, IDL, JavaScript, CSS, INI, PHP.
- *context*: contexto de visibilidade do código, que pode ser:
 - *memberdecl*: declaração de membro, inserido entre a declaração da classe e seu término;
 - *methoddecl*: declaração de método;
 - *decl*: declaração de variável;
 - *typeddecl*: declaração de tipo (*type*);
 - *methodbody*: corpo de método;
 - *page*: reservado para uso futuro;
 - *file*: reservado para uso futuro;
 - *any*: inserido em qualquer contexto;

Sorround

Recurso incorporado na versão 2006, que insere um bloco de código selecionado para "dentro" de uma estrutura de código selecionada, como while, for e begin/end.

- *delimiter*: define o character delimitador de pontos de tabulação;
- *![CDATA]*: define o bloco de código “colado” no editor.

Para criar pontos de tabulação é preciso defini-los antes da *tag code*. Para criar um ponto de tabulação é preciso adicionar uma *tag point* com os seguintes atributos/sub-tags:

- *name*: nome do ponto de tabulação;
- *default*: o ponto que possuir esse valor como verdadeiro será o primeiro ponto selecionado;
- *editable*: define se o ponto poderá ser acessado durante a navegação;
- *text*: *tag* opcional que define um valor padrão para o ponto de tabulação;

Conjunto de Characters

Evite a utilização de characters acentuados. O interpretador não é *unicode* e erros podem ocorrer se usarmos characters acentuados.

- *hint*: texto que é exibido como *hint* no momento da entrada no ponto de tabulação.

Além de montar os pontos de tabulação é possível utilizar alguns scripts para validar o *Live Template*, declarar variáveis e invocar o *Code Completion*. Para utilizar um script é preciso adicionar uma *tag script* dentro ou fora de uma *point*.

Além disso, é preciso definir o momento da chamada de sua execução. Por exemplo, para executar a declaração de uma variável no momento da saída do *Live Template* (pressionando a tecla TAB), o seguinte código deve ser adicionado:

Live Template	Permite navegação	Declara variável	Descrição
(*	Não	Não	Insere um conjunto de characters para criação de bloco de comentários.
{	Não	Não	Mesma funcionalidade do anterior.
ArrayC	Sim	Não	Cria um array de “n” posições com valores constantes.
ArrayD	Sim	Não	Cria um array de “n” posições com valores variáveis.
Begin	Não	Não	Insere um bloco begin/end no código.
Case	Sim	Não	Insere uma instrução case simples.
Class	Sim	Não	Insere o corpo completo de declaração de classe, incluindo comentários.
ClassC	Sim	Não	Igual ao anterior somado à métodos construtores e destrutores.
ClassD	Sim	Não	Insere o corpo simples de declaração de classe.
CreateC	Não	Não	Insere declaração de um método construtor.
Else	Não	Não	Insere bloco de código com instrução else/begin/end.
Enum	Sim	Não	Cria o corpo de construção de um tipo enumerado.
For	Sim	Sim	Insere corpo de código da instrução for.
ForB	Sim	Sim	Igual ao anterior somado ao bloco de código begin/end.
ForIn	Sim	Sim	Insere corpo de código da instrução for in.
ForR	Sim	Sim	Insere corpo de código da instrução for reverso (downto).
Function	Sim	Não	Insere corpo de código para criação de funções.
If	Sim	Não	Insere corpo de código simples da instrução if.
IfB	Sim	Não	Igual ao anterior somado ao bloco de código begin/end.
IfE	Sim	Não	Igual ao anterior somado ao bloco de código else simples.
IfEB	Sim	Não	Igual ao anterior somado ao bloco de código else/begin/end.
Procedure	Sim	Não	Insere corpo de código para criação de procedures.
Region	Sim	Não	Cria uma region nomeada.
Repeat	Sim	Não	Insere corpo de código simples da instrução repeat/until.
Try	Sim	Sim	Insere corpo de código para criação dinâmica de objetos. Além de inserir código de criação, utiliza try/finally para destruição garantida do objeto.
TryE	Sim	Não	Insere corpo de código para tratamento de exceções incluindo declaração do objeto de exceção.
TryF	Sim	Não	Insere corpo de código simples da instrução try/finally.
Var	Sim	Sim	Insere código para declaração simples de variável.
While	Sim	Não	Insere corpo de código simples da instrução while.
WhileB	Sim	Não	Igual ao anterior somado ao bloco de código begin/end.

Tabela 1. Descrição dos Live Templates disponíveis para o Delphi 2006

```
<script language="Delphi" onenter="false" onleave="true" onvalidate="false">
DeclareVariable(|var|);
</script>
```

Neste exemplo temos a declaração da variável *var* que na verdade é um ponto de tabulação no momento da “saída” do *Live Template* (*onleave*). Existem alguns outros eventos, conforme podemos observar na **Tabela 2**.

Evento	Descrição
onvalidate	Ocorre antes do script ser inserido no código.
onenter	Ocorre quando um ponto de código recebe foco.
onleave	Ocorre quando um ponto de código perde foco.

Tabela 2. Eventos de script

Além de declarar variáveis, é possível executar outras ações dentro dos scripts (**Tabela 3**).

Função	Linguagem
InvokeClassCompletion	Delphi
PopulateCase(expression)	Delphi
DeclareVariable(variable)	Delphi
RemoveTemplate	Delphi
ValidateForTemplate	Delphi
GetClipboardContents	Delphi
VarDeclToAssignment	Delphi
Invoke_code_completion	C
Populate_case(\$expr\$)	C

Tabela 3. Funções de script

Criando Live Templates

Para criar um *Live Template*, acesse dentro do *Object Repository*, o grupo *Other Files* e em seguida o item *Code Template* (**Figura 2**).

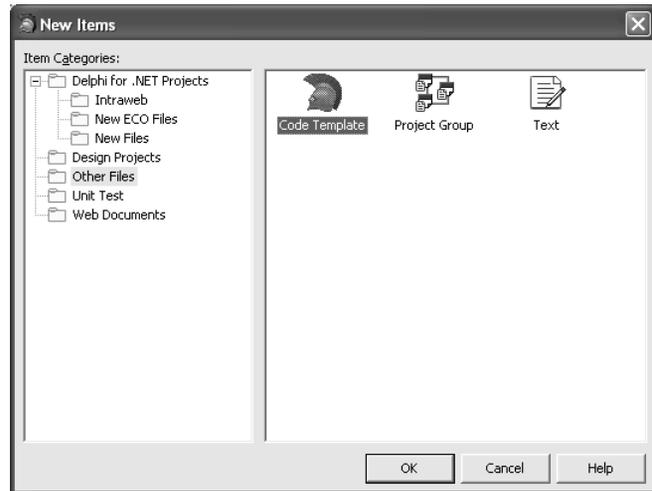


Figura 2. Criando um novo Template a partir do Object Repository

Identação

Para explicitar a indentação de uma linha de código dentro da sessão `![CDATA[]]`, utilize `|*|` “n” vezes antes de cada linha. Cada `|*|` produzirá um nível de indentação.

Navegação entre pontos

Para que o interpretador do BDS consiga efetuar a navegação entre os pontos de código (*point*), é necessário definir o *character delimitador*. Por padrão o character “\$” é utilizado. Contudo, esse pode gerar conflitos com outros códigos, portanto, deve-se definir outro. Normalmente utiliza-se o character pipe (“|”).

Ao confirmar a operação, uma nova janela de código será criada com o código inicial do *Live Template*, como podemos ver na **Listagem 2**.

Aceite pagamentos on-line em seu Website em minutos

Sem convênio. Sem taxa de adesão. Sem taxa mensal. Sem período mínimo.



BRpay.com.br
Pagamentos







Aceite pagamentos online em minutos

Amplie seus horizontes

Alcance novos compradores

A BRpay oferece um rápido, acessível e conveniente serviço de pagamentos online, para Web sites e comerciantes de qualquer porte e tamanho com um baixo custo. A BRpay não cobra taxas para abertura de conta, não cobra taxa de manutenção, não requer prazo mínimo de utilização ou volume mínimo. Você só paga uma pequena taxa sobre o que recebe, ou seja, se não recebe nada, então você não paga nada.

www.BRpay.com.br

Listagem 2. Código inicial do Live Template

```
<?xml version="1.0" encoding="utf-8" ?>
<codetemplate
  xmlns="http://schemas.borland.com/Delphi/2005/codetemplates"
  version="1.0.0">
  <template name="" invoke="manual">
    <description>
    </description>
    <author>
    </author>
    <code language=""><![CDATA[]]>
    </code>
  </template>
</codetemplate>
```

Definições Iniciais

O primeiro passo é definir o nome do *Live Template*. Atribua o valor "populalistview" para o atributo *name* da tag *template*. Em seguida atribua o valor "Delphi" para *language* em *code*. Esse atributo definirá o grupo de linguagens onde o *Live Template* participará e será exibido.

Por fim, crie um novo atributo "delimiter" com o valor "|" dentro da tag *code*. O mesmo definirá o caracter delimitador de "pontos de código". Por fim, preencha o conteúdo das tags *description* e *author* com os valores "Template para popular um ListView a partir de um DataSet" e "Clube Delphi", respectivamente. O resultado está na **Listagem 3**.

Listagem 3. Código do Live Template após as definições iniciais

```
<?xml version="1.0" encoding="utf-8" ?>
<codetemplate
  xmlns="http://schemas.borland.com/Delphi/2005/codetemplates"
  version="1.0.0">
  <template name="populalistview" invoke="manual">
    <description>
      Template para popular um ListView a partir de um DataSet
    </description>
    <author>
      ClubeDelphi
    </author>
    <code language="Delphi" delimiter="|"><![CDATA[]]>
    </code>
  </template>
</codetemplate>
```

Armazenamento

O BDS reconhece todos os arquivos XML do diretório *Application Data\Borland\BDS\4.0\Code_Templates* como *Live Templates*, desde que sigam as regras de sintaxe. Além desse, o diretório *Borland\BDS\4.0\Objrepos\Code_Templates\delphi* da instalação do BDS também é reconhecido.

Nesse último estão os *Templates* do BDS, então salve o nosso *template* nesse diretório, com o nome de "populalistview.xml".

Pontos de Código

Para utilizar o novo *template* é necessário adicionar os pontos de código contidos na **Tabela 4**.

Listagem 4. Código do Live Templates, após a adição dos pontos de código

```
<?xml version="1.0" encoding="utf-8" ?>
<codetemplate xmlns="http://schemas.borland.com/
  Delphi/2005/codetemplates" version="1.0.0">
  <template name="populalistview" invoke="manual">
    <description>
      Template para popular um ListView a partir de um DataSet
    </description>
    <author>
      Clube Delphi
    </author>
    <point name="NewItem">
      <text> NewItem </text>
      <hint>
        Nome do objeto "TListItem" que contera as instancias
        dos itens do "ListView"
      </hint>
    </point>
    <point name="NewColumn">
      <text> NewColumn </text>
      <hint>
        Nome do objeto "TListColumn" que contera as colunas do "ListView"
      </hint>
    </point>
    <point name="DataSet">
      <text> DataSet </text>
      <hint>
        Nome do objeto "TDataSet" que servira como fonte de dados
      </hint>
    </point>
    <point name="ListView">
      <text> ListView </text>
      <hint>
        Nome do objeto "TListView" que contera a lista de itens
      </hint>
    </point>
    <point name="Iterator">
      <text> i </text>
      <hint> Variavel de controle da iteracao </hint>
    </point>
    <code language="Delphi" delimiter="|"><![CDATA[]]></code>
  </template>
</codetemplate>
```

Listagem 5. Bloco de código do exemplo adicionado à tag Code

```
01 <code language="Delphi" delimiter="|"> <![CDATA[
02 {Abre o DataSet ou posiciona no primeiro registro}
03 if not |DataSet|.Active then
04   |*|DataSet|.Open
05 else
06   |*|DataSet|.First;
07   {Cria as colunas}
08   for |Iterator| := 0 to |DataSet|.Fields.Count -1 do begin
09     |*|NewColumn| := |ListView|.Columns.Add;
10     |*|NewColumn|.Caption := |DataSet|.Fields[|Iterator|].DisplayLabel;
11     |*|NewColumn|.AutoSize := True;
12   end;
13   {Efetua a iteracao nos registros}
14   while not |DataSet|.Eof do begin
15     |*|{Adiciona novo item}
16     |*|NewItem| := |ListView|.Items.Add;
17     |*|{Configura o "caption" do "item" com o primeiro campo}
18     |*|if |DataSet|.FieldCount > 0 then
19       |*|NewItem|.Caption := |DataSet|.Fields[0].AsString;
20     |*|{Cria os "subitens" com os outros campos}
21     |*|for |Iterator| := 1 to |DataSet|.Fields.Count -1 do
22       |*|NewItem|.SubItems.Add(|DataSet|.Fields[i].DisplayText);
23     |*|DataSet|.Next;
24   end;]]>
25 </code>
```

Conjunto de Caracteres

Reforçando a observação citada anteriormente, evite a utilização de caracteres acentuados. O interpretador não é unicode e erros podem ocorrer se usarmos caracteres acentuados.

+ de 80.000 membros cadastrados
+ de 15.000 exemplos com fontes
+ de 900 apostilas
+ de 4.000 dicas
Fórum Delphi
Artigos

TOTALMENTE GRÁTIS
www.delphi.eti.br

Um dos maiores sites de apoio a desenvolvedores Delphi do Brasil!!!

Ponto de Código	text	hint
NewItem	NewItem	Nome do objeto "TListItem" que conterá as instâncias dos itens do ListView.
NewColumn	NewColumn	Nome do objeto "TListColumn" que conterá as colunas do ListView.
DataSet	DataSet	Nome do objeto "TDataSet" que servirá como fonte de dados.
ListView	ListView	Nome do objeto "TListView" que conterá a lista de itens.
Iterador	Iterador	Variável de controle da iteração.

Tabela 4. Pontos de código do novo Template

A tag *hint* dos pontos de código descreve a funcionalidade de cada ponto. Após a inserção dos pontos de código, o *Live Template* estará semelhante à **Listagem 4**.

Código

Agora será adicionado o código do novo *Live Template* (na tag *code*), conforme a **Listagem 5** (a numeração é apenas para fins de explicação do código).

Da linha 4 à linha 6 o código abre o *DataSet* ou posiciona o cursor no primeiro registro, caso o mesmo já esteja aberto. Logo, esse novo *template* somente funcionará com *DataSets* bi-direcionais, como o *SimpleDataSet* ou *ClientDataSet*.

Da linha 8 à linha 11 são criadas as colunas do *ListView* de acordo com os *Fields* do *DataSet*. A linha 14 inicia a iteração dos registros do *DataSet*. A linha 16 instancia um novo *Listitem*, enquanto na linha 19 adicionamos o valor do primeiro *Field* para a propriedade *Caption* do *Listitem*.

As linhas 21 e 22 adicionam os valores dos outros *Fields* às colunas do *Listview*. Além disso, serão adicionados comentários ao código para facilitar o entendimento de quem vier a utilizar o *template*.

Scripts

Para facilitar ainda mais a escrita desse código, o *template* efetuará a declaração automática das variáveis/objetos utilizados no mesmo. Para isso, será utilizado o evento *onleave* do *template*. Isso significa que, no momento da saída (acionando a tecla TAB), o script será executado (**Listagem 6**).

Listagem 6. Script para declaração automática de variáveis/objetos

```
<script language="Delphi" onenter="false" onleave="true">
  DeclareVariable(|NewItem|);
  DeclareVariable(|NewColumn|);
  DeclareVariable(|Iterador|);
</script>
```

Nesse ponto o *template* já está concluído (**Listagem 7**).

Exemplo de utilização

Vamos criar uma pequena aplicação do tipo *VCL Forms Application - Delphi for Win32* para utilizar o *template*. Salve-a como desejar. No formulário vazio adicione os componentes e altere suas propriedades de acordo com a **Tabela 5**.

Análise de Tipos

O BDS é capaz de analisar os objetos utilizados nos pontos de código e declarar os tipos das variáveis e/ou objetos corretamente. Para isso, é preciso que as *units* necessárias estejam adicionadas a cláusula *uses* (nesse exemplo, temos que ter a *unit ComCtrls*).

Livros de Delphi



Delphi - Implementação de Algoritmos e Técnicas p/ Ambientes Visuais
Visual Books

R\$ 62,00

Delphi: para Todas as Versões
Ciência Moderna



R\$ 29,00



Estudo Dirigido de Delphi 2005
Érica

R\$ 78,50

Delphi 2005: Aplicações de Banco de Dados com InterBase 7.5 e MySQL 4.0.23
Érica



R\$ 149,50



Introdução ao Desenvolvimento de Aplicações em Delphi
Uniderp

R\$ 50,00



O Melhor da Delphi Line (CD 1 e 2)
Sistemas e Exemplos com fontes e Delphi
Ramos da Informática

10% desconto

De: **R\$ 134,00** Por: **R\$ 120,00**



Delphi e Relatórios - Treinamento a Distância (CD)2005
Clube Delphi

10% desconto

De: **R\$ 200,00** Por: **R\$ 180,00**



Guia do Desenvolvedor de Delphi for .NET
Makron Books

10% desconto

De: **R\$ 173,50** Por: **R\$ 156,15**



Delphi 8 Para Plataforma .NET: Curso Completo
Axcel

10% desconto

De: **R\$ 249,00** Por: **R\$ 224,10**



Sistema Comercial Integrado com Delphi 2005: Cadastro e Estoque
Érica

10% desconto

De: **R\$ 135,00** Por: **R\$ 121,50**



Frete Grátis
Para todo o Brasil!

3x no Visa e Mastercard
e **6x no Hipercard**

Outras facilidades de pagamento:



livrosdeprogramacao
.com.br

Listagem 7. Código completo do Live Template

```
<?xml version="1.0" encoding="utf-8" ?>
<codetemplate
xmlns="http://schemas.borland.com/Delphi/2005/codetemplates"
version="1.0.0">
  <template name="populalistview" invoke="manual">
    <description>
      Template para popolar um ListView a partir de um DataSet
    </description>
    <author>
      Clube Delphi
    </author>
    <point name="NewItem">
      <text>
        NewItem
      </text>
      <hint>
        Nome do objeto "TListItem" que contera
        as instancias dos itens do "ListView"
      </hint>
    </point>
    <point name="NewColumn">
      <text>
        NewColumn
      </text>
      <hint>
        Nome do objeto "TListColumn" que contera
        as colunas do "ListView"
      </hint>
    </point>
    <point name="DataSet">
      <text>
        DataSet
      </text>
      <hint>
        Nome do objeto "TDataSet" que servira como
        fonte de dados
      </hint>
    </point>
    <point name="ListView">
      <text>
        ListView
      </text>
      <hint>
        Nome do objeto "TListView" que contera a lista de itens
      </hint>
    </point>
    <point name="Iterador">
      <text>
        i
      </text>
      <hint>
        Variavel de controle da iteracao
      </hint>
```

```
</point>
<script language="Delphi" onenter="false" onleave="true">
  DeclareVariable(|NewItem|);
  DeclareVariable(|NewColumn|);
  DeclareVariable(|Iterador|);
</script>
<code language="Delphi" delimiter="|"> <![CDATA[
{Abre o DataSet ou posiciona no primeiro registro}
if not |DataSet|.Active then
  |*||DataSet|.Open
else
  |*||DataSet|.First;
  {Cria as colunas}
  for |Iterador| := 0 to |DataSet|.Fields.Count -1 do begin
  |*||NewColumn| := |ListView|.Columns.Add;
  |*||NewColumn|.Caption := |DataSet|.Fields[|Iterador|].DisplayLabel;
  |*||NewColumn|.AutoSize := True;
  end;
  {Efetua a iteracao nos registros}
  while not |DataSet|.Eof do begin
  |*||Adiciona novo item}
  |*||NewItem| := |ListView|.Items.Add;
  |*||Configura o "caption" do "item" com o primeiro campo}
  |*||if |DataSet|.FieldCount > 0 then
  |*|| |*||NewItem|.Caption := |DataSet|.Fields[0].AsString;
  |*|| {Cria os "subitens" com os outros campos}
  |*||for |Iterador| := 1 to |DataSet|.Fields.Count -1 do
  |*|| |*||NewItem|.SubItems.Add(|DataSet|.Fields[|i|. DisplayText);
  |*||DataSet|.Next;
  end;]]>
</code>
</template>
</codetemplate>
```

Conclusão

Como visto, os *Live Templates* adicionam muita produtividade na escrita de código. Além de possuir uma utilização simples, é possível criar nossos *templates*, maximizando ainda mais a produção de código. Com um pouco de criatividade é possível transformar códigos repetitivos em tarefas produtivas e eficazes, além de tornar os códigos um pouco mais padronizados entre diferentes desenvolvedores. ■

“ O Maior Portal para Desenvolvedores da América Latina.”



Acesse:

www.devmedia.com.br

