Delphi

# Code Insight

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Conclusão de código** | Exibe uma lista de propriedades, métodos e eventos quando você insere um nome de classe seguido por um ponto no Editor de Código. Você pode então selecionar um item e pressionar Enterpara adicioná-lo ao seu código.  Se essa opção não estiver marcada, você ainda poderá invocar a Conclusão de código pressionando Ctrl+Space. O valor padrão é ON (marcado).   |  |  | | --- | --- | | **Invocação automática** | Abre a lista de conclusão de código enquanto você digita o código Delphi ou C ++. C ++ também suporta o ->para invocar a conclusão de código. | | **Auto parêntese** | Adiciona automaticamente parênteses às chamadas de função quando você invoca a Conclusão de Código. | | **História** | Lembra o que você inseriu ou selecionou e coloca isso como a primeira entrada na lista de conclusão de código. | | **Mostrar palavras reservadas** | Inclui palavras-chave de idioma (somente para Pascal) na lista de conclusão de código. | |
| **Parâmetros de código** | Exibe os argumentos de uma chamada de método à medida que você os insere em seu código. Se esta opção não estiver marcada, você ainda pode chamar parâmetros de código pressionando Shift+Ctrl+Space. O valor padrão é ON (marcado). |
| **Conclusão do modelo de código** | Invoca a conclusão do modelo de código quando você pressiona SPACEdepois de iniciar um modelo existente. Quando essa opção está desativada, você deve pressionar TABpara invocar o preenchimento do modelo depois de digitar o nome do modelo. O valor padrão é ON (marcado). |

## Code Hint

O Code HINT (CTRL + SHIFT + H) é utilizado em cima de um código. Afim de mostrar informações sobre o mesmo.

## Code Template

O Code Template é utilizado para ser utilizado padrões de códigos do delphi.

Utilizado com CTRL + J.

## Code Folding

Imagine que você tenha necessidade de devidir os códigos em regiões.

($ REGION ‘INCLUIR’)

($ ENDREGION)

Todo e qualquer código entre esses comandos poderar ser “minimizado”, afim de organizar seu código.



## Sync Edit

Imagine que você tenha declarado seu código todo e precise alterar uma mesma informações em todos os códigos. Existe a possibilidade de se alterar tudo de uma vez!

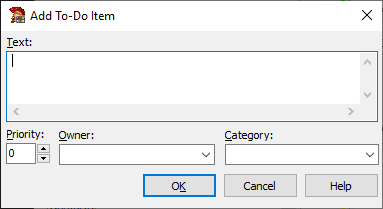
Selecione o código todo em questão que vai ser alterado e clique em CTRL + SHIFT + J.

Todas as palavras que se repetem vão ser enquadraculadas, alterando uma, altera todas!

## To Do List

É uma opção que quando você está trabalhando com vários programadores existe uma necessidade de deixar anotações para serem realizadas.

Clique onde há necessidade de se colocar uma anotação e aperte CTRL + SHIFT + T.



Ficando assim:

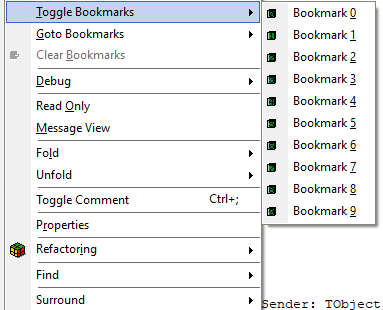


Podendo gerenciar os afazeres.

## BooksMarks

Serve para marcar determinadas linhas no código, para lembrar onde há necessidade de alterar algo.

Clique com o botão direito do mouse em determinada linha ao lado esquerdo.



Logo após, selecione um BookMark.

## Refactoring

O comando Refactoring é interessante para executar comandos de qualquer lucas do projeto.

Bastando clicar com o botão direito do mouse no local (em alguns casos selecionar o código todo) e ir na opção “Refactoring”.

Com ele você pode declarar de forma rápida, por exemplo, method, variáveis, fields, uses, entre outros.

# Mensagens de alerta

No delphi, existem várias formas de se criar uma alerta, ou mensagem, ao realizar determinado comando.

Exemplos:

## Show Message

**begin**  
  **// Show a simple message**  
  **showmessage**('Hello World');  
  
  **// Show a blank message**  
  **showmessage**('');  
  
  **// Split this into two lines**  
  **showmessage**('Hello '+#13#10+'World');  
**end;**

No código acima, o resultado é o seguinte:



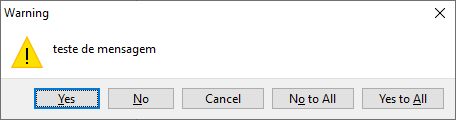




## MessageDlg

Um outro método é o message dlg, com a mesma sintaxe que o showmessage, porém, com opções mais detalhadas. (UTILIZE O SEGUNDO CODE INSIGHT PARA ENTENDER QUAIS CAMPOS DEVEM SER INFORMADOS DO CÓDIGO).

Ex: MessageDlg ('teste de mensagem', mtWarning, mbYesAllNoAllCancel, 0, mbYes) ;



No código acima, estamos criando um messagedlg teste, como aviso (mtWarning), com botões de yesAll, noALL, Cancel (mbYesAllNoAllCancel) sem nenhum help (0(sem o botão de ajuda ao apertar f1)) e o botão padrão (pré selecionado (mbYes))