

Estudo de Caso – Entendendo os Códigos

Este estudo de caso foi inspirado por alguém que queria uma explicação da amostra de código abaixo.

```
Sub Teste1()  
With Range("B2:B" & Cells(Rows.Count, 1).End(xlUp).Row)  
    .SpecialCells(4).Formula = "=R[-1]C"  
    .Value = .Value  
End With  
End Sub
```

Se você quiser realmente aprender como os modelos orientados a objetos funcionam com tão poucos códigos, desdobre tudo nos seus componentes individuais. Depois então, coloque-os juntos novamente. No meio disto, faça uso extensivo do XL VBA help, msdn.microsoft.com, e qualquer livro à sua disposição.

Mas antes de começarmos, destacaria uma coisa. Microsoft Office 2003 faz um trabalho absolutamente terrível de busca nos seus arquivos de ajuda (MSDN trabalha muito melhor). Procurar resultados que deveriam naturalmente estar no topo não estarão por lá; algumas vezes eles estarão próximos do final da lista, o que significa ter que rolar para baixo por várias páginas. Mas, é um esforço útil pois, tipicamente, o conteúdo do arquivo ajuda é muito bom.

Vamos começar com a primeira linha:

```
Range("B2:B" & Cells(Rows.Count, 1).End(xlUp).Row)
```

Exatamente como uma fórmula XL, temos que ler isto da esquerda para a direita e de dentro para fora. Quando algo tem parênteses, precisamos resolver o que está dentro do parênteses primeiro e daí então mover-se para fora. Caso contrário, vamos da esquerda para a direita.

OK, então, o par de parênteses para o *Range* é representado pelo primeiro e o último parênteses na linha. So, primeiro observariamos o que está dentro, ou em *"B2:B" & Cells(Rows.Count, 1).End(xlUp).Row*

A primeira parte parece familiar. Ela é o início de um identificador de range B2:B...mas, não precisamos de um número no final? Claramente, isto deverá vir do &... qualquer coisa.

Assim, vamos observar *Cells(Rows.Count, 1).End(xlUp).Row*. Oh, oh. Um outro conjunto de parênteses.

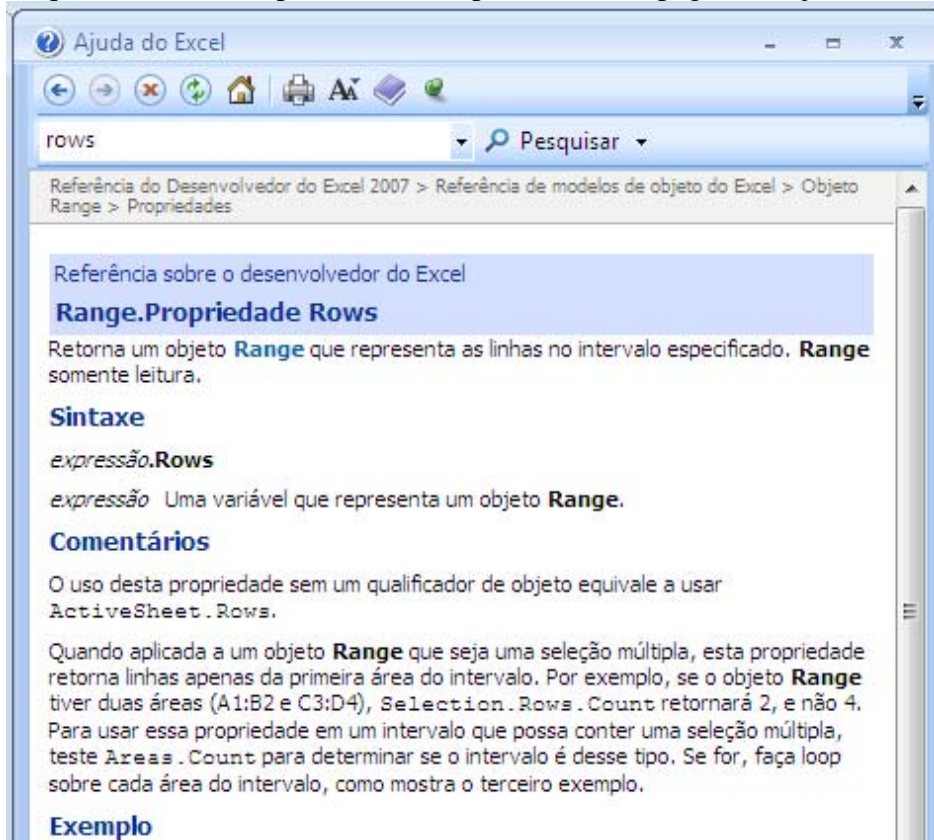
Pare aqui, precisamos ver o *Rows.Count*. Nenhum parêntese; portanto, vamos da esquerda para a direita.

Se visitarmos a ajuda XL VBA para a palavra chave 'rows' (sem as aspas simples) encontraremos a propriedade *Rows*. Esta é uma daquelas instâncias onde a palavra chave buscada não está próxima ao topo, não está nem na primeira página dos resultados mostrados. Role para baixo uma página e a Propriedade *Rows* está na direção do topo da segunda página (ver Figura 1).



Figura 1

Clique no link da Propriedade *Rows* para acessar a página de ajuda. Ela avisa:



Como sabemos se nossa referência é para o objeto application, para a planilha, ou para alguma outra coisa? Observe a sentença destacada na seção Comentários: "Usando esta propriedade sem um qualificador de objeto é equivalente a usar `ActiveSheet.Rows`." Colocando as duas sentenças destacadas juntas, percebemos que *Rows*, por si só, retorna um **objeto range** que representa todas as linhas da planilha ativa.

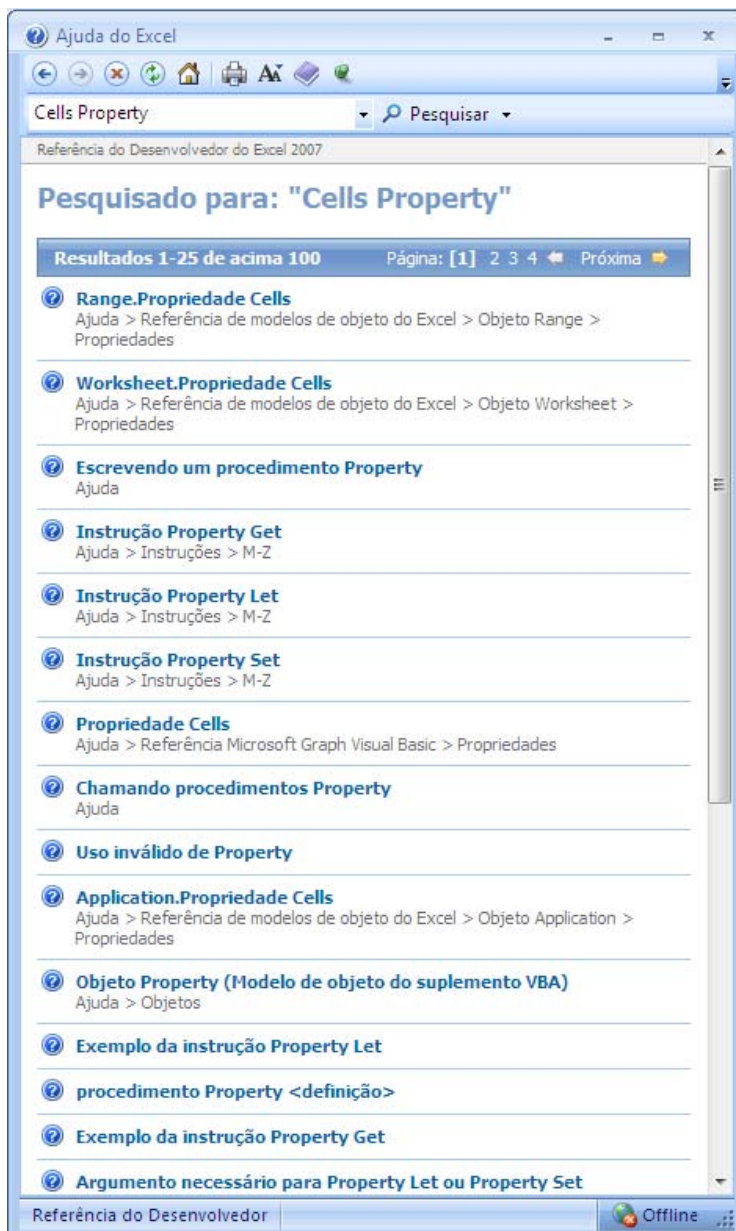
O uso das propriedades ou métodos inadequados tais como *Rows* acima e assim como *Cells* e *Range* podem ter – e frequentemente tem - conseqüências sem propósitos. É significativamente seguro sempre qualificar as referências. Ver capítulo xx – estudo de caso yy para mais coisas

OK, armado com o conhecimento que temos de um **objeto range** que representa todas as linhas na planilha ativa, o que faz a parte *.Count*? Se ela não é óbvia, visite-o na ajuda XL.



OK, então, *Rows.Count* retorna uma contagem do número de linhas na planilha – 65.536. Por que não usar exatamente este número? Porque uma versão futura do Excel poderá ter um número de linhas diferente!

Tudo bem. Volte ao nosso problema. Agora sabemos que *Cells (Rows.Count, 1)* é o mesmo que *Cells({número de linhas},1)*. O que uma propriedade *Cells* nos dá? Visite a ajuda XL VBA. Agora, este é um daqueles lugares onde a busca na ajuda Office 2003 realmente, engole. Buscar a palavra chave *células* é completamente inútil. Buscar 'propriedade cells' (sem as aspas) nos levará à Propriedade **Cells Property**; mas, ela é a última naquele tópico! Vamos à figura. Uma combinação de palavras chaves que estamos procurando é aproximadamente o último tópico que a Microsoft pensa que deveríamos realmente ler!



Referência sobre o desenvolvedor do Excel

Worksheet.Propriedade Cells

Mostrar tudo

Retorna um objeto **Range** que representa todas as células na planilha (e não apenas as células que estejam em uso no momento).

Sintaxe

expressão.Cells

expressão Uma variável que representa um objeto **Worksheet**.

Comentários

Como a propriedade **Item** é a *propriedade padrão* para o objeto **Range**, você pode especificar o índice de linha e de coluna imediatamente após a palavra-chave **Cells**. Para obter mais informações, consulte a propriedade **Item** e os exemplos desse tópico.

O uso desta propriedade sem um qualificador de objeto retorna um objeto **Range** que representa todas as células da planilha ativa.

Exemplo

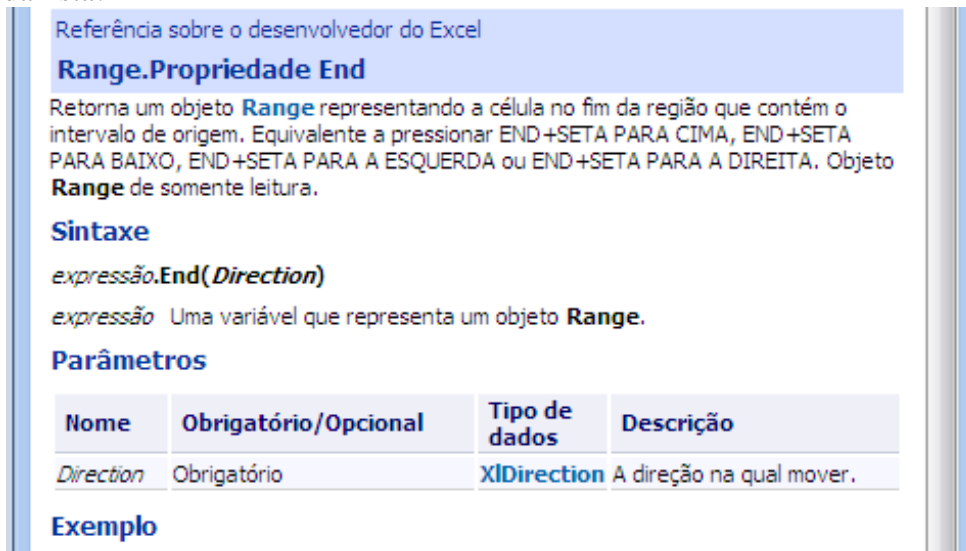
Existem três coisas que deveríamos notar. Primeiro, na seção Comentários encontramos "Usando esta propriedade um qualificador de objeto retorna um objeto Range que representa todas as células na planilha ativa". Segundo, clicar na "Propriedade Cells quando ela se aplica ao objeto Worksheet." Indica que o uso da *Cells* desqualificada dá-nos uma coleção de todas as células na planilha.

O terceiro item e final é notar o parágrafo acima da seção Comentários: "Devido a propriedade Item ser a propriedade *default* para o objeto Range, você pode especificar o índice linha e coluna imediatamente após a palavra chave Cells." O que isto significa? Basicamente que podemos usar o atalho Cells(linha, coluna) ao invés de Cells.Item(linha, coluna) para identificar uma célula específica.

Assim, agora que sabemos o que *Cells (Rows.Count, 1)* significa – a última célula na coluna 1, i.e., coluna A. Na versão atual do XL isto será a A65536, mas ela poderá ser alguma outra coisa numa versão futura.

OK, agora estamos pronto para cuidar de *Cells(Rows.Count,1).End(xlUp)*

Adivinhe o que temos a fazer? Entendeu. Entrar com a palavra chave *end* na busca da ajuda XL VBA e ela nos conduzirá à *propriedade End* – e para variar ela está no topo da lista!



Referência sobre o desenvolvedor do Excel

Range.Propriedade End

Retorna um objeto **Range** representando a célula no fim da região que contém o intervalo de origem. Equivalente a pressionar END+SETA PARA CIMA, END+SETA PARA BAIXO, END+SETA PARA A ESQUERDA ou END+SETA PARA A DIREITA. Objeto **Range** de somente leitura.

Sintaxe

expressão.End(*Direction*)

expressão Uma variável que representa um objeto **Range**.

Parâmetros

Nome	Obrigatório/Opcional	Tipo de dados	Descrição
<i>Direction</i>	Obrigatório	XlDirection	A direção na qual mover.

Exemplo

O primeiro parágrafo nos diz exatamente o que *End* retorna. Ela retorna um **objeto Range** que é o equivalente ao resultado de uma combinação de teclas particular. Clicando em *xlDirection* mostra a lista de valores válidos para *Direction*. Esperançosamente, é evidente ao leitor que *xlUp* corresponde a combinação de 2 teclas: END+UP ARROW. Para encontrar o que isto significa, entre com algum número em

A1:A5, Vá até a A65536, e pressione as 2 teclas: END+UP ARROW. Você descobrirá que ela levará você para a célula não vazia mais ao fundo na coluna A – o que faz sentido, suponho.

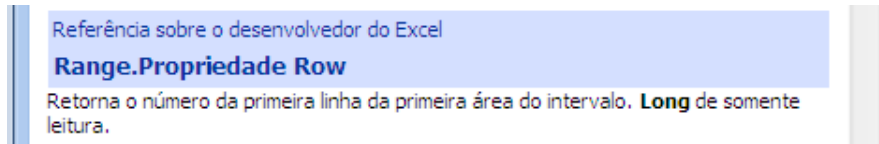
Aqui está um importante aviso. O que acontece se as células A65535 e A65536 não estiverem vazias? Para onde o END+UP ARROW nos levará? Ela é realmente a célula não vazia mais ao fundo (bottom) na coluna A?

Similarmente, e se a coluna estiver completamente vazia? A END+UP ARROW nos leva à célula não vazia mais ao fundo da coluna A?

Estas são as limitações da End() que o desenvolvedor deverá estar ciente delas. Quando usada apropriadamente e com precauções apropriadas, End() é uma propriedade muito útil.

Assim, agora que sabemos o que `Cells(Rows.Count,1).End(xlUp)` faz. Ela nos dá um **objeto range** que representa a célula não vazia mais funda da coluna A.

A seguir, termos `Cells(Rows.Count,1).End(xlUp).Row`. Podemos supor o que `.Row` faz mas ela não prejudica uma visita na ajuda VBA.

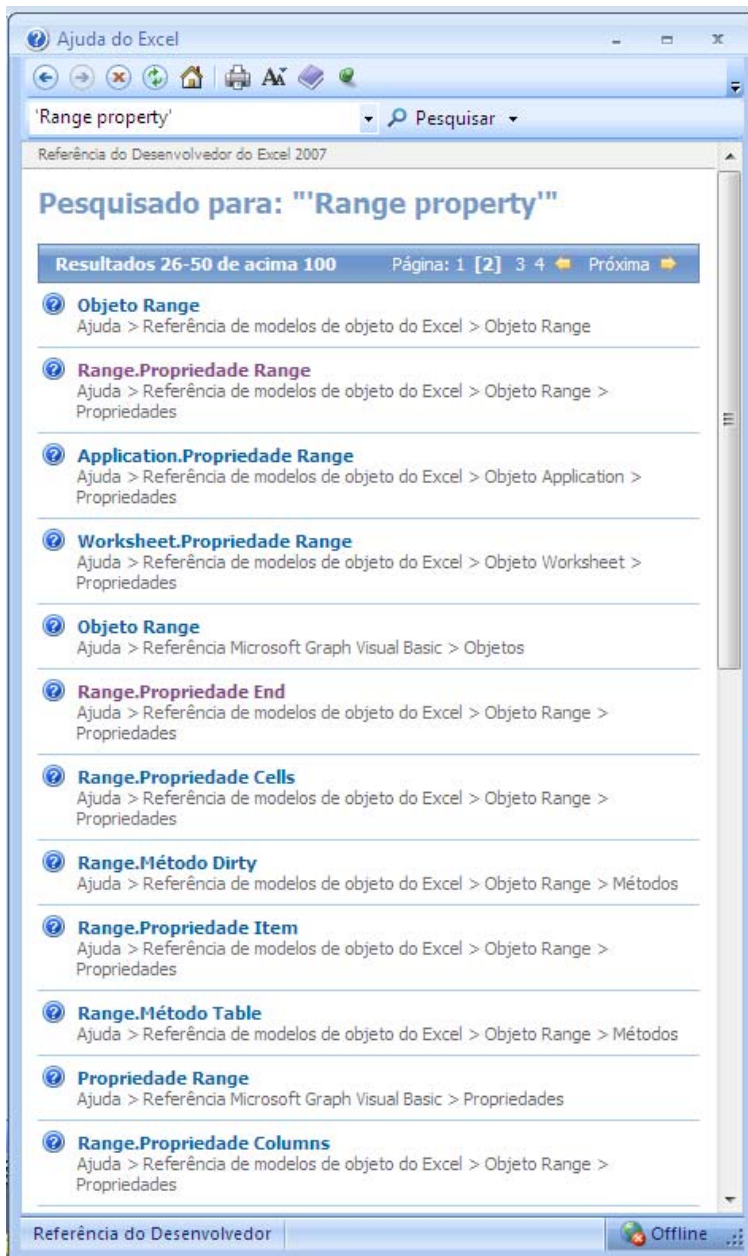


Ela nos dá um **número** que representa a linha da primeira célula no objeto range ao qual a propriedade é aplicada. No nosso caso, o range consiste de uma única célula (a célula não vazia mais no fundo da col. A). Assim, agora temos o número da linha da célula mais no fundo e não vazia da coluna A.

OK, trabalhamos de dentro para fora e da esquerda para a direita para obter o ponto de `Range("B2:B" & { número da linha da célula não vazia mais no fundo da coluna A})`

Se buscarmos na ajuda VBA por 'range' (por instante, você sabe o que fazer com as aspas, não sabe?), encontramos a 'propriedade range' perto do fundo da última mais uma página.

Clique no link e a ajuda VBA mostra:



Você será levado a concluir que estamos interessados no 2º item. Clique ali para ver:

Referência sobre o desenvolvedor do Excel

Range.Propriedade Range

Retorna um objeto **Range** que representa uma célula ou um intervalo de células.

Sintaxe

expressão.Range(*Cell1*, *Cell2*)

expressão Uma variável que representa um objeto **Range**.

Parâmetros

Nome	Obrigatório/Opcional	Tipo de dados	Descrição
<i>Cell1</i>	Obrigatório	Variant	O nome do intervalo. Deve ser uma referência no estilo A1 no idioma da macro. Pode incluir o operador de intervalo (dois-pontos), o operador de interseção (um espaço) ou o operador de união (uma vírgula). Também pode incluir sinais de dólar, mas eles são ignorados. Você pode usar um nome definido de local em qualquer parte do intervalo. Se usar um nome, será pressuposto que ele está no idioma da macro.
<i>Cell2</i>	Opcional	Variant	A célula no canto superior esquerdo e inferior direito do intervalo. Pode ser um objeto Range que contém uma única célula, uma coluna inteira ou uma linha inteira, ou pode ser uma seqüência de caracteres que nomeia uma única célula no idioma da macro.

Comentários

Quando usada sem um qualificador de objeto, esta propriedade é um atalho para `ActiveSheet.Range` (ela retorna um intervalo da planilha ativa; se o documento ativo não for uma planilha, a propriedade falhará).

Ela diz a você para seguir adiante: Retorna um objeto Range que representa a célula ou a range de células.

OK, então, de uma maneira ou outra obteremos um objeto range. O que mais?

expression.Range(Cell1, Cell2) onde *Cell1* é exigida e *Cell2* é opcional.

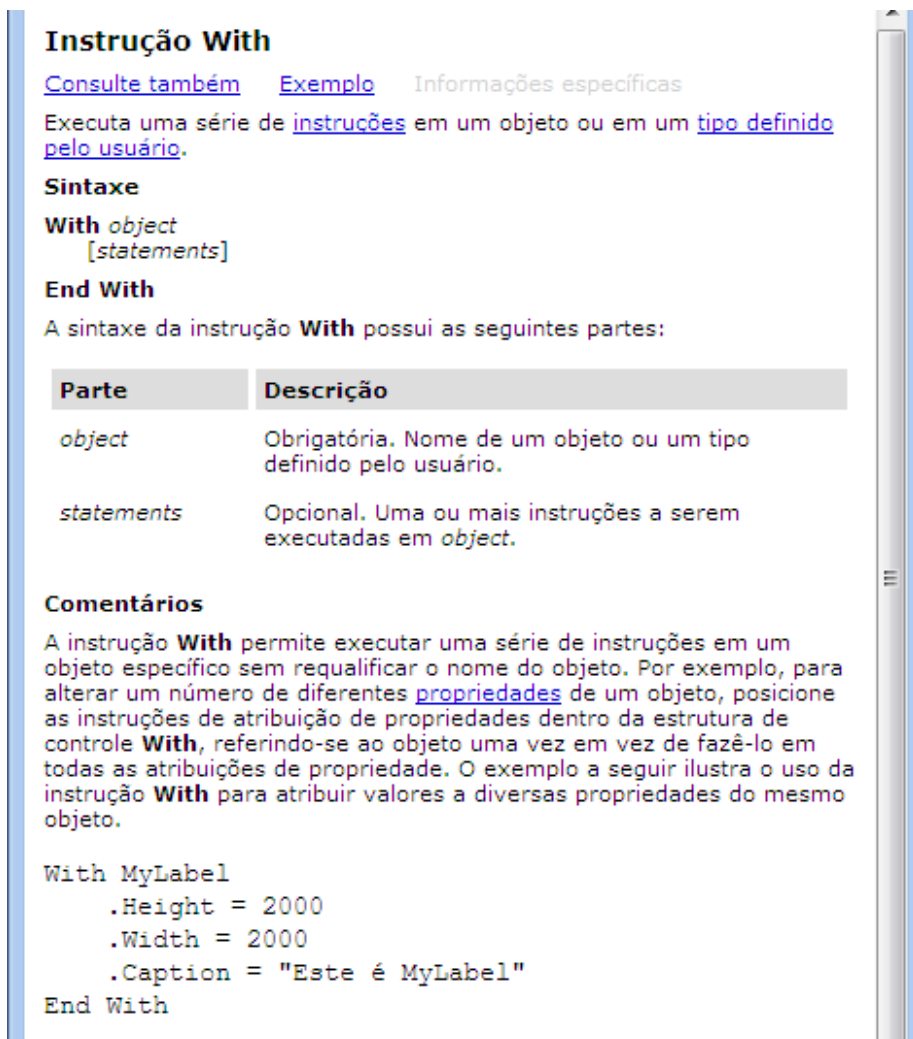
Como no caso de Rows e as propriedades Cells, na seção Comentários encontramos: "Quando usado sem um qualificador de objeto, esta propriedade é um atalho para `ActiveSheet.Range` (ela retorna um range da activesheet; se a activesheet não for a worksheet, a propriedade falha)." Assim, o uso da propriedade Range desqualificada significa que ela aplica à célula ativa.

Também, não estamos usando a parte opcional *Cell2*. Ainda mais, estamos usando a variante com o "operador range (dois pontos)."

Bem, quase sabemos o que a primeira linha no código significa: *Range (B2:B{número de linha da célula não vazia mais ao fundo da coluna A})* : todas as células na coluna B da linha 2 até a última linha que contém dados na coluna A.

Isto poderia ser feito diferentemente? Sim. Eu prefiro trabalhar com objetos ao invés de ir e vir entre objetos e seus atributos como feito acima. Por que converter o objeto range identificado por Cells (Rows.Count, 1).End(xlUp) para um número linha, concatenando-o com a string B2:B e daí convertendo isto para objeto range com a propriedade Range()? Poderíamos ter permanecidos com objetos até o fim e usado a variação *Range (Cell1, Cell2)* como em:

OK, volte para a próxima tarefa. Busque *With* no help e encontramos...oops, nada. Pesquise *with statement* e encontraremos o que estamos procurando.



Instrução With

[Consulte também](#) [Exemplo](#) [Informações específicas](#)

Executa uma série de [instruções](#) em um objeto ou em um [tipo definido pelo usuário](#).

Sintaxe

```
With object  
    [statements]
```

End With

A sintaxe da instrução **With** possui as seguintes partes:

Parte	Descrição
<i>object</i>	Obrigatória. Nome de um objeto ou um tipo definido pelo usuário.
<i>statements</i>	Opcional. Uma ou mais instruções a serem executadas em <i>object</i> .

Comentários

A instrução **With** permite executar uma série de instruções em um objeto específico sem requalificar o nome do objeto. Por exemplo, para alterar um número de diferentes [propriedades](#) de um objeto, posicione as instruções de atribuição de propriedades dentro da estrutura de controle **With**, referindo-se ao objeto uma vez em vez de fazê-lo em todas as atribuições de propriedade. O exemplo a seguir ilustra o uso da instrução **With** para atribuir valores a diversas propriedades do mesmo objeto.

```
With MyLabel  
    .Height = 2000  
    .Width = 2000  
    .Caption = "Este é MyLabel"  
End With
```

Basicamente, é uma maneira para evitar a repetição de uma referência ao objeto que está próximo à palavra chave *With*. Assim,

```
.SpecialCells().Formula=...  
.Value=.Value
```

São os equivalentes de

```
Range("B2...").SpecialCells().Formula=...  
Range("B2...").Value=Range("B2...").Value
```

A *With* poupa-nos de muita digitação – e, em certos casos, pode mesmo aumentar a velocidade de execução do código.

Continuando na veia acima, vamos encerrar a análise do resto do código. Buscando na ajuda VBA por *SpecialCells* dirigimos ao *método SpecialCells*:

Range.Método SpecialCells

Retorna um objeto **Range** representando todas as células que coincidem com o tipo e valor especificados.

Sintaxe

expressão.SpecialCells(*Type*, *Value*)

expressão Uma variável que representa um objeto **Range**.

Parâmetros

Nome	Obrigatório/Opcional	Tipo de dados	Descrição
<i>Tipo</i>	Obrigatório	xlCellType	As células a serem incluídas.
<i>Valor</i>	Opcional	Variant	Se <i>Type</i> for xlCellTypeConstants ou xlCellTypeFormulas , este argumento será usado para determinar quais tipos de células devem ser incluídas no resultado. Esses valores podem ser adicionados em conjunto para retornar mais de um tipo. O padrão é selecionar todas as constantes ou fórmulas, independente do tipo.

Valor de retorno

Intervalo

Comentários

xlCellType constants	Valor
xlCellTypeAllFormatConditions Células de qualquer formato	-4172
xlCellTypeAllValidation Células que possuem critérios de validação	-4174
xlCellTypeBlanks Células vazias	4
xlCellTypeComments Células contendo notas	-4144
xlCellTypeConstants Células contendo constantes	2
xlCellTypeFormulas Célula contendo fórmulas	-4123
xlCellTypeLastCell A última célula no intervalo usado	11
xlCellTypeSameFormatConditions Células com o mesmo formato	-4173
xlCellTypeSameValidation Células com os mesmos critérios de validação	-4175
xlCellTypeVisible Todas as células visíveis	12

Parece que estamos presos. O código lê *SpecialCells(4)*. O autor perversamente usou uma constante ao invés da mnemonic da lista acima. Podemos calcular como um valor constante 4 mapeia sobre um dos itens xlCellType? Sim, mas não é legal – e certamente não tão trivial quando se está usando o mnemonic na primeira vez. Contudo, ele pode ser feito.

Selecione o texto todo correspondente à descrição dos valores válidos para CellType. No VBE, abra a janela imediata (CTRL+G) e cole.

```

Immediate
xlCellTypeAllFormatConditions. Cells of any format
xlCellTypeAllValidation. Cells having validation criteria
xlCellTypeBlanks. Empty cells
xlCellTypeComments. Cells containing notes
xlCellTypeConstants. Cells containing constants
xlCellTypeFormulas. Cells containing formulas
xlCellTypeLastCell. The last cell in the used range
xlCellTypeSameFormatConditions. Cells having the same format
xlCellTypeSameValidation. Cells having the same validation criteria
xlCellTypeVisible. All visible cells

```

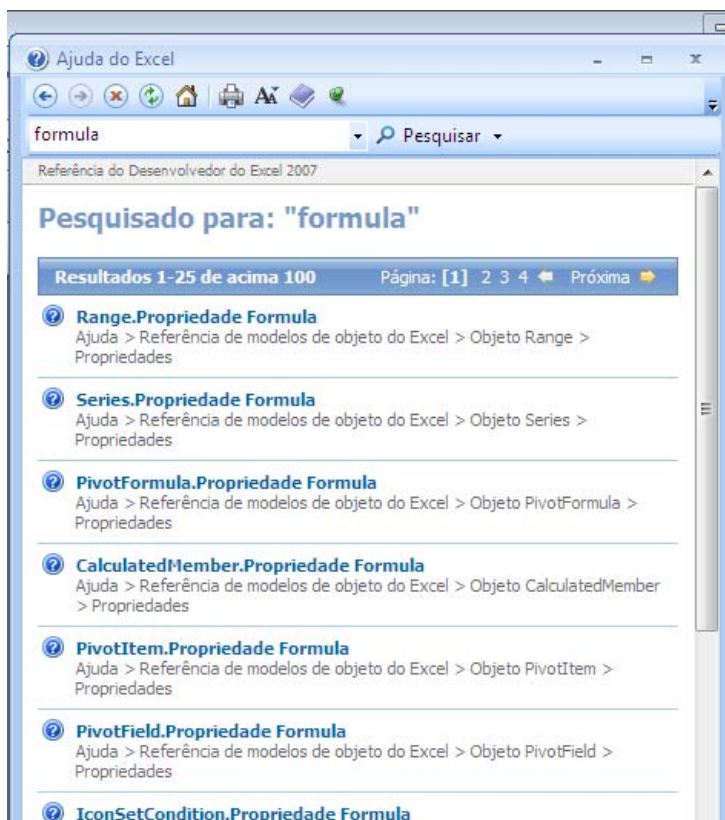
Agora, para cada linha, gaste uma ? na frente e delete a explicação após a palavra chave. Pressione ENTER e VBE lhe dará o valor constante. Pare quando você vir 4.

```
Immediate
?xlCellTypeAllFormatConditions
-4172
?xlCellTypeAllValidation
-4174
?xlCellTypeBlanks
4
xlCellTypeComments. Cells containing notes
xlCellTypeConstants. Cells containing constants
xlCellTypeFormulas. Cells containing formulas
xlCellTypeLastCell. The last cell in the used range
xlCellTypeSameFormatConditions. Cells having the same format
xlCellTypeSameValidation. Cells having the same validation criteria
xlCellTypeVisible. All visible cells
```

Bingo! Então, agora sabemos o que o autor realmente quis dizer – ou esperamos que isto é o que estava pensando!

Agora, movendo-se para *.Formula = "=R[-1]C"*

Ao visitar Formula na ajuda VBA leva-nos à propriedade Formula (mas note também que a presença da propriedade FormulaR1C1):



A propriedade Formula por si lê:

Referência sobre o desenvolvedor do Excel

Range.Propriedade Formula

Mostrar tudo

Retorna ou define um valor **Variant** que representa a fórmula do objeto em notação de estilo A1 e no idioma da macro.

Sintaxe

expressão.**Formula**

expressão Uma variável que representa um objeto **Range**.

Comentários

Esta propriedade não está disponível para fontes de dados OLAP.

Se a célula contiver uma constante, essa propriedade a retornará. Se a célula estiver

Então, parece que o autor está dirigindo algumas regras aqui. A propriedade Formula é proposta a ser usada com fórmulas de estilo A1, não com a variedade R1C1. Enquanto neste caso o Excel e/ou VBA acomodaram este desvio na documentação, em outras instâncias a exigência do estilo fórmula são reforçadas. Eu teria usado .FormulaR1C1=...

A parte final: *.Value=.Value*. Isto é um 'truque' no sentido que funciona para a Formula, FormulaR1C1, Valor, Valor2, e uma outras poucas propriedades que são mutuamente exclusivas. Atribuindo a um derrota o que tivesse sido entrado em qualquer uma das outras. Assim, *.Value=.Value* toma o valor atual da célula e o coloca de volta. Mas isto remove efetivamente a fórmula da célula!

Conclusão: Efetivamente, o autor entrou em cada célula vazia no range identificado da coluna B com o valor da célula acima nele. E, se a célula acima estivesse vazia, aquela célula teria o valor da célula acima dela de maneira que até o Excel encontrar uma célula não vazia.