Criando macros básicas no OpenOffice.org

Produção em massa

O conjunto de aplicativos do OpenOffice.org pode lançar mão de uma grande variedade de scripts e macros para automatizar tarefas repetitivas. O modo mais fácil é usar o dialeto Basic integrado. Este artigo ajudará você a se iniciar no mundo dessa linguagem surpreendentemente sofisticada.

por Oliver Frommel

Se você com freqüência se vê repetindo tarefas complexas e cheias de etapas no OpenOffice.org [1], já é hora de criar uma macro. O OpenOffice.org permite o uso de uma porção de opções de programação. A versão 1.1 introduziu o conceito de *pontes* (*bridges*), que permitem ao usuário acrescentar seu próprios programas em C, C++, Java ou Python. Na versão 2.0, que deve ser lançada a qualquer momento, o pacote básico do OpenOffice.org também reconhecerá a *Common Language Interface* (CLI), que permite aos usuários adicionar seus próprios programas em Javascript (também conhecido como ECMAScript/ISO 16262) e C#. Dentre todas as linguagens usadas com o OpenOffice.org, o Basic talvez seja a opção mais simples. Este artigo descreve como começar a criar macros para o OpenOffice.org em Basic.

Primeiros passos

A programação em Basic começa no menu `Ferramentas | Macros` (`Tools | Macros`), que tem duas opções:`Gravar Macro` (`Record macro`), um registrador de macros para uso interativo, e `Macro...`, que abre uma nova janela (**figura 1**) – por sinal, também acessível pela combinação de teclas ***[Alt]+[F11]***.

Figura 1: A janela de gerenciamento de macros em Basic do OpenOffice.org

Essa janela ajuda a organizar as macros que acompanham a distribuição do OpenOffice.org; ali você também pode mexer em suas próprias macros. Os programas em Basic são chamados de módulos e atribuídos a uma biblioteca (uma coleção de módulos); por padrão, `Module1` e `Standard`. Um novo módulo sempre conterá a função `Main`, que é basicamente uma caixa vazia sem nenhum código – uma função é como um "subprograma", vários subprogramas reunidos formam um módulo. Clicar em `Edit` leva ao editor e exibe a caixa vazia (**figura 2**).

Agora você pode usar todo o conjunto de recursos do Basic. Em `Ajuda | Conteúdo` (`Help | Contents`) você encontra um guia de referência do Basic no OpenOffice.org com uma lista de funções. Ela está no item `Macros e Programação | Referência de Comandos | Listagem alfabética` (`Macros and Programming | Commands | Alphabetical list`).

Adicione o seguinte código ao início da caixa do programa:

```
Sub Main
    Mensagem = "Agora são " & Time()
    MsgBox Mensagem, 0
End Sub
```

A palavra-chave `Sub` designa uma função, ou seja, um "pedacinho" do programa que faz alguma coisa em especial. Todo programa precisa de uma função principal; dentro dela as outras funções serão chamadas. Normalmente, a função principal de uma macro no OpenOffice.org é chamada `Main`. Na verdade, o OpenOffice.org não está lá muito interessado em nomes e começa a analisar o programa pela primeira função que encontra no módulo da macro. `Mensagem` é o que chamamos de variável: uma entidade com um nome e que guarda alguma coisa na memória. No caso, `Mensagem` é uma variável do tipo *String*, que guarda uma seqüência de caracteres qualquer. Seu nome é livre – na verdade há algumas restrições para a criação de nomes para variáveis, mas falaremos disso mais adiante.

Em nosso programa, a variável `Mensagem` inclui um trecho constante, `"Agora são "`, e a saída da função `Time()`, que dá a hora do dia. O operador `&` combina essas partes para formar um único conjunto. Você pode deixar de fora os parênteses da função `Time`; eles não têm efeito na saída.

Finalmente, a função `MsgBox` exibe uma caixa de diálogo. O primeiro parâmetro passa o texto a ser exibido e o segundo define o tipo do diálogo. O `0`, neste exemplo, quer dizer que a caixa de diálogo só mostrará um botão de OK. `1` adicionaria também um botão de cancelar; há variações com botões de `Sim/Não` e outras combinações. `MsgBox` também retorna um valor que indica em qual botão o usuário clicou, que poderia ser usado por outra função mais adiante no código caso fosse necessário. Nosso exemplo não faz nada com esse valor de retorno.

Este Documento

Precisamos de mais do que simples funções de Basic se quisermos acessar e manipular os documentos do OpenOffice.org em nosso programa, já que isso envolve acessar os objetos e interfaces

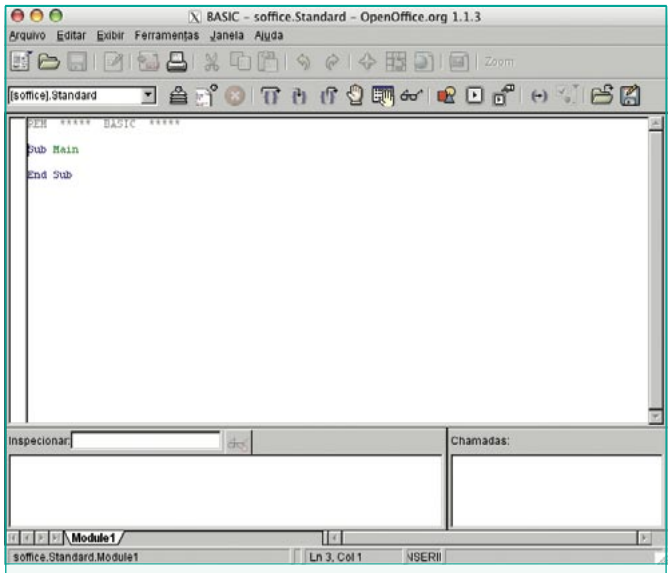

Figura 2: Quando você cria uma nova macro, o OpenOffice.org gera uma função *Main* vazia, como mostrado no editor integrado.

UNO (ver **Quadro 1: UNO complexo, Basic simples**). Nosso ponto de acesso a essa hierarquia de objeto é `ThisDocument`, uma palavra-chave que referencia o documento no qual o script está sendo rodado – pode ser um documento de texto, uma planilha ou mesmo um desenho. `ThisDocument` é o objeto pai que fornece métodos básicos para navegar na estrutura de árvore de um documento.

Mas antes de fazer isso, precisamos declarar algumas variáveis. A palavra-chave `Dim` nos ajudará com isso. O número de elementos numa lista (matriz) precisa ser posto entre parênteses, como em `(10)`. O Basic lida com essa convenção de forma diferente das outras linguagens de programação: o parâmetro não especifica o número de elementos, mas designa o índice mais elevado. Assim, a matriz unidimensional `Jurubeba(10)` fornece uma lista de 11 elementos, de `Jurubeba(0)` até `jurubeba(10)` – como você pode ver, usar letras maisúsculas ou minúsculas não faz diferença.

Após ter passado uma referência ao documento, o script confere se ele é mesmo um documento de texto. Para fazê-lo, precisamos do método `supportsService()` para a referência ao documento:

```
oDoc = ThisComponent
If oDoc.supportsService("com.sun.star.text.TextDocument")
Then
```

Além desse método, o objeto documento também tem uma função aparentemente promissora chamada `getText()`. A função não fornece o próprio texto do documento, mas uma referência ao serviço de texto que, por sua vez, tem diversos métodos, incluindo alguns para mover um cursor pelo texto (como `createTextCursor`).

O serviço de texto tem outra função chamada `createEnumeration()`, que relaciona os parágrafos num documento, mas isso ainda não oferece o texto em si. Pelo contrário, um parágrafo tem cerca de 150 propriedades que descrevem com precisão coisas como, por exemplo, espaçamento, tabulação e estilo.

Quadro 1: UNO complexo, Basic simples

O OpenOffice.org permite o uso de Basic diretamente em suas entranhas, o que elimina a necessidade de uma interface externa. A linguagem Basic em si não é exatamente uma teoria de física mecatrônica, mas você vai precisar de algum conhecimento avançado do Basic para poder trabalhar com a arquitetura complexa do OpenOffice.org – tanto que, sem ele, é capaz que não compreenda o texto deste quadro.

O OpenOffice.org tem uma interface de programação independente chamada UNO (*Universal Network Objects*). O UNO segue os atuais paradigmas de projeto de software (ou seja, serviços, interfaces e os chamados padrões de projeto ou *design patterns* [2]). Uma volumosa quantidade de documentação de desenvolvimento [3] e da API [4] atesta a importância do UNO na programação em OpenOffice.org.

O Basic do OpenOffice.org não permite o uso de todos os recursos do UNO, já que a própria linguagem Basic é muito simples. Por exemplo, o Basic não trabalha com tipos de dados complexos, como *hashes*, que atribuem valores a palavras-chave. Essa carência dificulta a programação daquilo que julguei que seria um exemplo muito simples: um script que contasse o número de ocorrências de uma palavra específica num texto. Um *hash* (ou matriz associativa) com a palavra a procurar como chave seria uma solução ideal para esse exemplo. Mas exigiria muita programação extra no Basic do OpenOffice.org; de fato, você precisaria implementar sua própria tabela de *hash*, coisa que está muito além do escopo deste artigo.

Ao contrário do Basic do OpenOffice.org, a interface do UNO é orientada a objetos. Esses pontos de vista antagônicos acabam fazendo despontar algumas peculiaridades: por exemplo, alguns métodos são mapeados diretamente para propriedades. Em outras palavras, um programador não precisa chamar uma função como `circle.radius()`, em vez disso pode usar diretamente o atributo `circle.radius`. Isso causa um pouco de confusão em aplicações práticas que usem ambas as notações. Nesses casos, documentação é tudo: sempre comente seu código, para não se perder depois quando tiver que fazer a manutenção.

Podemos acessar os elementos de texto normais em um parágrafo chamando `createEnumeration()`. Se o documento contiver uma tabela, o OpenOffice.org exibirá uma mensagem de erro porque o `createEnumeration` não reconhece elementos de tabela. Seria preciso adicionar algum tratamento de exceção nesse caso.

O método `String()` retorna o texto puro de um elemento. O Basic usa uma construção em laço que começa com `Do` e termina com `Loop` para analisar uma dada condição de parada, que pode ser especificada imediatamente depois de qualquer uma dessas palavras-chave. Se a parada ocorrer no final do laço, o script vai iterar pelos comandos dentro do laço ao menos uma vez.

Nossa macro de exemplo escreve em um arquivo os dados do texto analisados por esse método. O nome de arquivo é especificado pela variável `MeuArquivo`. O Basic tem um comportamento incomum no que toca ao acesso a arquivos: é preciso um número especial, ou manipulador (*handle*) para abrir um arquivo, e não o próprio nome do arquivo. A função `Freefile()` nos dá esse número. Podemos agora passar o número e o nome do arquivo para `Open()`, abrindo assim o arquivo para escrita:

```
Open MeuArquivo For Output As #NúmeroDoArquivo ➡
```

Quadro 2: Resolução avançada de problemas

O *debugger* (*depurador*, em Português, embora quase todos os programadores prefiram o termo em inglês) integrado só é útil para tipos simples de variáveis no Basic. Como os tipos UNO do OpenOffice.org não são mapeados diretamente para tipos do Basic, o depurador simplesmente exibirá pontos de interrogação para objetos UNO. Outros métodos dão mais informações, mas exigem mais trabalho de programação. Por exemplo, `Dbg_supportedInterfaces()` e `Dbg_methods()`, chamados como métodos de um objeto, são bastante úteis.

Há uma macro pronta do Basic que pode ajudar a simplificar a resolução de problemas. A macro exibe uma janela com os métodos e propriedades do objeto UNO que você precisa investigar. Ela é adequadamente chamada *Xray* [5] (raio-X) porque realmente permite olhar dentro de um objeto (**figura 3**). Após descompactar o arquivo Zip, abra o documento no OpenOffice.org e siga as instruções. Basicamente, tudo o que você precisa fazer é atribuir a macro ***Xray*** ao documento em que você está trabalhando (`Ferramentas | Macro | Organizador`, em seguida `Biblioteca`). Para radiografar um objeto, você precisa acrescentar uma linha parecida com `Xray.Xray oDoc` ao seu script.

Por acaso, o site do *Xray* tem outro documento útil: uma coleção de fragmentos de script [6] com as devidas explicações, uma cortesia de Andrew Pytonyak, autor de um famoso livro sobre macros no OpenOffice.org.

Listagem 1: Exportador de texto simples

```
Sub Main
   Dim oDoc As Object
   MeuArquivo = "/home/oliver/resultado.txt"
   oDoc = ThisComponent
   title$ = oDoc.DocumentInfo.Title
   If oDoc.supportsService("com.sun.star.text.TextDocument") Then
      NumeroArq = Freefile()
      Open MeuArquivo For Output As #NumeroArq

      oTexto = oDoc.getText()
      oParagrafo = oTexto.createEnumeration()
      Do While oParagrafo.hasMoreElements()
         oPar = oParagrafo.nextElement()
         oTextos = oPar.createEnumeration()
         Do While oTextos.hasMoreElements()
            oTexto = oTextos.nextElement()
            Print #NumeroArq oTexto.string
            If oTexto.string = "" Then
               Print #NumeroArq
            Endif
         Loop
      Loop
   Endif ' If oDoc.supportsService(..)
   Close #NumeroArq
End Sub\
```

O comando `Print` com o número do arquivo como primeiro parâmetro permite adicionar linhas ao arquivo: `Print #NúmeroDoArquivo "Olá pessoal!"`. Sem o número do arquivo, o OpenOffice.org abre uma caixa de diálogo ao encontrar o comando `Print`. Após adicionar as linhas, é preciso fechar o arquivo. Uma chamada à instrução `Close`, com o número do arquivo como argumento, cuida disso. A **listagem 1** mostra a macro completa.

Para rodar o script, clique no segundo botão a contar da esquerda na segunda fileira (ver **figura 2**). Esse botão roda o script no documento atual. Se o OpenOffice.org descobrir um erro de sintaxe, ele imediatamente gera um relatório ao executar o script e o exibe numa caixa de diálogo. Infelizmente, seguindo a tradição dos antigos interpretadores Basic as mensagens de erro são demasiado genéricas e não muito úteis para resolução de problemas (por exemplo: *Variável de objeto não atribuída*). Um "erro genérico" ocorre se você tentar rodar o script enquanto a janela de ajuda for seu documento atual.

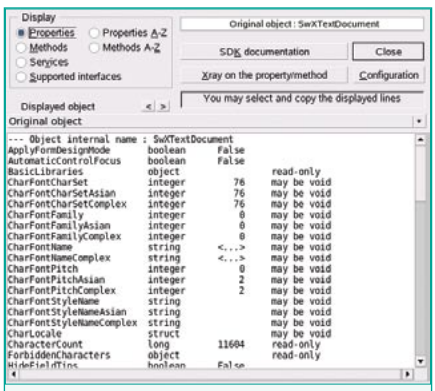

Figura 3: A macro *Xray* é útil para resolução de problemas. Ela exibe as propriedades do objeto UNO e Basic.

Os botões com as chaves permitem passear pelo código. Escolha um nome de variável e clique no botão com os óculos para visualizar seu valor no campo `Watch`, no canto inferior esquerdo da janela – novamente, aplicam-se as restrições mencionadas anteriormente. Confira o **Quadro 2: Resolução avançada de problemas** para mais dicas de depuração.

A interface UNO do OpenOffice.org oferece aos autores de scripts uma útil ferramenta de programação para aplicações de escritório. Mas programar no OpenOffice.org não é a experiência intuitiva que se poderia esperar. O sistema é tão complexo quanto o CORBA [7] ou o J2EE [8] e assume conhecimento de conceitos de desenvolvimento de software moderno como arquitetura de componentes e padrões de design.

Os pré-requisitos para programar macros no OpenOffice.org colocam a tarefa além do alcance da maioria dos usuários ocasionais, mas programadores amadores e profissionais encontrarão verdadeiros mundos a explorar. ■

Informações

[1] OpenOffice: http://www.OpenOffice.org/

[2] Padrões de projeto : http://en.wikipedia.org/wiki/Design_pattern_%28computer_science%29

[3] Guia de desenvolvimento: http://api.OpenOffice.org/docs/DevelopersGuide/DevelopersGuide.htm

[4] Guia de Referência do UNO: http://api.OpenOffice.org/docs/common/ref/com/sun/star/module-ix.html

[5] XRay: http://www.ooomacros.org/dev.php#101416

[6] Macros explicadas (em inglês): http://www.ooomacros.org/dev.php#91896

[7] CORBA: http://www.corba.org/

[8] J2EE: http://java.sun.com/j2ee/

[9] Tutorial do StarBasic da Sun: ftp://docs-pdf.sun.com/817-3924/817-3924.pdf