Dica

PyUNO

e as macros Python

por ***Noelson Duarte***

A partir desta edição, vamos apresentar uma série de artigos sobre a programação do OpenOffice.org com a linguagem Python.

A ponte entre Python e os componentes UNO, conhecida como PyUNO, foi incorporada ao OpenOffice.org 1.1. Esta versão incluiu, ainda, o núcleo da linguagem Python 2.3 em sua distribuição.

Na versão 2.0, Python passou a ser suportada pelo ambiente de macros. Isto facilitou o acesso aos objetos de entrada da API do OpenOffice.org e melhorou a integração entre o aplicativo e a linguagem. No momento, a integração resume-se à execução de macros. A edição do código fonte e a instalação das macros devem ocorrer fora do aplicativo.

Codificação

No BrOffice.org 2.1, o aumento ou a redução do tamanho da fonte só funciona se todo o conteúdo selecionado tiver o mesmo tamanho. Segue uma macro com funções para aumentar ou reduzir, em uma unidade, o tamanho da fonte do texto selecionado, incluindo múltiplas seleções.

• Antes de usar o seu editor favorito para digitar o código, tenha em mente os seguintes detalhes:

• Os arquivos **py** usam o fim de linha no estilo Unix (LF). Isto é uma causa de erro bastante comum. Portanto, configure o seu editor de texto adequadamente;

• Funções a serem executadas pela interface gráfica, não devem receber argumentos;

• Atualmente, apenas arquivos **py** localizados nos caminhos da variável PYTHONPATH podem ser importados.

Dica

```
# tamanhoFonte.py: macro para o BrOffice.org
# por Noelson Duarte, em 10/05/2007

def AlteraTamanhoFonte(valor):
   """ adiciona o valor ao tamanho da fonte do texto selecionado"""
   # o conteúdo selecionado deve conter apenas parágrafos
   # fora de objetos como tabelas, molduras, etc.
   #
   # obtem o modelo do documento
   oDoc = XSCRIPTCONTEXT.getDocument()
   # obtem a selecao
   oSel = oDoc.getCurrentSelection()
   # a seleção é um objeto TextRanges ?
   if (oSel.supportsService("com.sun.star.text.TextRanges")):
      # obtém cada uma das seleções
      for i in range(oSel.getCount()):
         # obtém o conteúdo da iésima seleção
         oCur = oSel.getByIndex(i)
         # enumera o conteúdo da iésima seleção
         oEnum = oCur.createEnumeration()
         # visita cada elemento da enumeração
         while (oEnum.hasMoreElements()):
            oTxt = oEnum.nextElement()
            # o elemento é um objeto Paragraph ?
            if (oTxt.supportsService("com.sun.star.text.Paragraph")):
               # obtém as porções dentro do parágrafo, todo
               # conteúdo de texto com formatação diferente
               oEnum2 = oTxt.createEnumeration()
               # visita as porções e altera o tamanho da fonte
               while (oEnum2.hasMoreElements()):
                  oTxtParte = oEnum2.nextElement()
                  tamFonte=oTxtParte.getPropertyValue("CharHeight")+valor
                  oTxtParte.setPropertyValue("CharHeight", tamFonte)
   return None

def IncrementaTamanhoFonte():
   """ adiciona 1 ao tamanho da fonte do texto selecionado"""
   AlteraTamanhoFonte(1)
   return None

def DecrementaTamanhoFonte():
   """ adiciona -1 ao tamanho da fonte do texto selecionado"""
   AlteraTamanhoFonte(-1)
   return None

g_exportedScripts = (IncrementaTamanhoFonte, DecrementaTamanhoFonte)
```

Dica

Vamos a uma breve discussão sobre o código diretamente relacionado com o ambiente de macros.

Inicialmente, os **comentários** no estilo """ serão exibidos como uma "dica", quando a macro for selecionada no diálogo **Seletor de Macro**.

A seguir, temos a variável XSCRIPTCONTEXT, que possui três métodos para acesso aos objetos de entrada da API do OpenOffice.org:

- **getDocument**, retorna o modelo do documento para o qual a macro foi chamada. Este é o objeto responsável pelos dados contidos no documento e deve ser usado em todas as situações envolvendo edição;

- **getDesktop**, retorna o objeto Desktop. Usado para gerenciar os documentos abertos, criar e carregar documentos, entre outras tarefas;

- **getComponentContext**, retorna o contexto UNO. É o objeto usado para obter o ServiceManager um objeto utilizado para criar outros objetos.

Segue, um trecho de código usando os dois últimos métodos:

```
ctx = XSCRIPTCONTEXT.getComponentContext()
smgr = ctx.ServiceManager
nomeSv = "com.sun.star.awt.UnoControlDialogModel"
# usa o service manager para criar um objeto
oDlg = smgr.createInstanceWithContext( nomeSv, ctx )
# ...
desktop = XSCRIPTCONTEXT.getDesktop()
sURL = "private:factory/swriter"
novoDoc = desktop.loadComponentFromURL( sURL,"_blank", 0, () )
```

Finalmente, a variável **g_exportedScripts** é usada para relacionar as funções visíveis na interface gráfica. Se não for definida, todas as funções do módulo serão exibidas.

No restante do código, acrescentei breves comentários sobre os principais comandos. Mas, os leitores ansiosos por informações sobre os objetos da API, podem visitar o Projeto Programação, no portal do BrOffice.org, para consultar a documentação e as macros disponíveis.

Instalação

O BrOffice.org, por padrão, organiza as suas macros em containeres, de acordo com os direitos de execução das mesmas. Temos três tipos:

- ***Minhas Macros***: aqui, encontramos as macros instaladas para um determinado usuário e devem ser instaladas no diretório do aplicativo do usuário, em:

```
<broo_user_install>/user/Scripts/python/biblioteca/
```

O caminho subordinado a **Scripts** deve ser criado pelo usuário.

- **Macros do BrOffice.org**: aqui, estão as macros compartilhadas por todos os usuários de uma instalação. São instaladas no diretório de instalação do aplicativo, em:

Dica

<broo_install>/share/Scripts/python/biblioteca/

O caminho subordinado a **python** deve ser criado. O BrOffice.org traz algumas macros, assim não será necessário a criação deste subdiretório.

Use o gerenciador de arquivos do seu sistema para inspecionar o caminho:

<broo_install>/share/Scripts/python da instalação do seu BrOffice.org.

- **Documento**: são gravadas no documento. Todos os usuários com direitos de acesso ao mesmo, podem executá-las. Atualmente, não existe nenhum auxílio para a criação de macros Python num documento e a instalação envolve a edição manual do mesmo.

Para instalar a macro no container **Minhas Macros** devemos:

alternar para a pasta <broo_user_install>/user/Scripts;

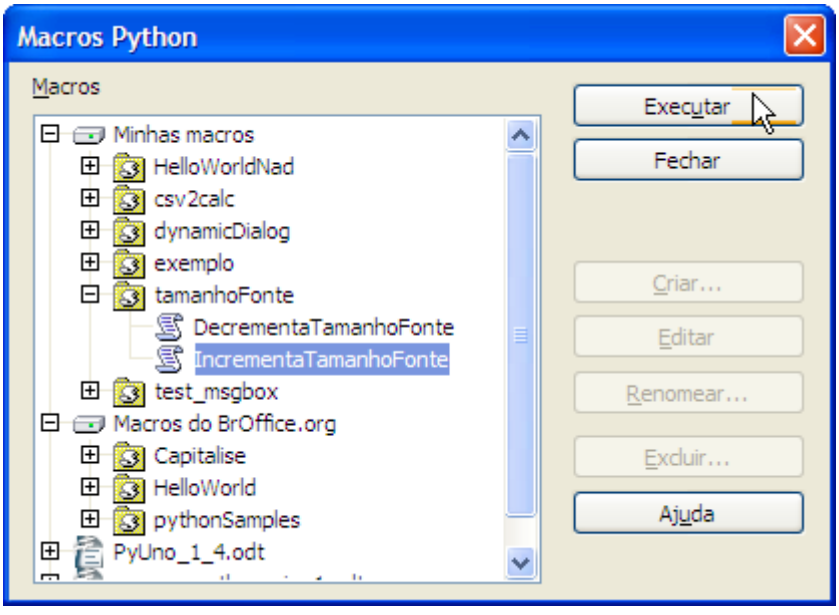
- criar a pasta **python**;
- criar uma pasta própria para a biblioteca, escolha um nome adequado (passo opcional);
- mover o arquivo Python para a pasta da biblioteca, se você a criou, senão mova para a pasta **python**;

Para o container **Macros do BrOffice.org**, o procedimento é similar, mas o caminho inicial deve ser:

<broo_install>/share/Scripts/python/

Para desinstalar uma macro, basta remover o arquivo Python do caminho padrão.

Execução



Podemos rodar a nossa macro de vários modos, o mais simples é através do diálogo **Seletor de Macro**, exibido pelo comando Executar Macros em Ferramentas | Macros. Todos os “scripts” registrados no aplicativo, independente da linguagem em que foi escrito, serão exibidos.

O segundo modo, via diálogo **Macros Python,** opera apenas com os “scripts” desta linguagem. Para exibir o diálogo, selecione Ferramentas | Macros | Organizar Macros | Python:

Dica

Note, na figura, a estrutura dos containeres citados no tópico anterior. Na minha instalação, optei por não criar uma pasta própria para o módulo. Neste caso, o nome da biblioteca tem o mesmo nome do arquivo sem a extensão.

Após a sua instalação, caso você não consiga ver as macros, revise o código fonte e certifique-se de ter gravado o arquivo usando o fim da linha no estilo Unix (LF).

Para macros usadas com freqüência, podemos configurar a interface gráfica, via ***Ferramentas | Personalizar*** e atribuir uma combinação de teclas ou um ícone numa barra de ferramentas, para disparar a sua execução.

Uma macro pode, ainda, ser disparada na ocorrência de um evento. Por exemplo, durante a abertura de um documento, ao clicar sobre uma figura ou hiperlink, etc. A atribuição de macros aos eventos do aplicativo ou documento deve ser feita no diálogo **Personalizar** guia **Eventos**. Os objetos que oferecem esta facilidade, possuem uma guia **Macros** em seu diálogo de propriedades, que permite a ligação de "scripts" aos seus eventos.

Por hoje é só. No próximo artigo, espero apresentar o desenvolvimento de "**add-ons**" com a linguagem Python. São suplementos com uma maior integração na interface gráfica, você pode acrescentar ícones, entradas no menu principal, mas ... vamos aguardar.