A

Comprendre l'API d'OpenOffice.org

Cette annexe offre une introduction à la principale source d'information officielle sur les mécanismes internes à OpenOffice.org. Tous ces documents, en anglais, sont accessibles à un utilisateur confirmé.

Qu'est-ce que l'API ?

L'API (Application Programming Interface) d'OpenOffice.org est un ensemble de points d'entrée permettant de manipuler OpenOffice.org – quoique ne couvrant cependant tous ses mécanismes. Sans être liée à un langage de programmation particulier, elle est accessible de manière privilégiée avec OOoBasic, mais on peut aussi utiliser des langages tels que Java, Python, Delphi, voire d'autres outils de script tel VBscript.

L'API est un système logiciel très complexe par son étendue et par ses concepts ; elle est composée de très nombreux objets héritant les uns des autres. Le développeur la verra comme un ensemble de méthodes, de fonctions, de propriétés, de structures de données. Elle est organisée en un arbre dont la racine est :

```
com.sun.star.
```

À partir de cette racine, elle se subdivise en modules, qui sont des groupements de logiciels, par exemple :

```
com.sun.star.text.
```

Tout élément de l'API se situe à un nœud de cet arbre, en juxtaposant les branches successives :

```
com.sun.star.text.WrapTextMode.PARALLEL
```

Un module se décompose en un service ou plusieurs, et parfois en sous-modules, comme pour le module `text`.

Un service comporte lui-même parfois des services (en anglais *included services*). Il peut avoir des propriétés. Ce sont des « vraies » propriétés (nous verrons plus loin qu'OOoBasic présente aussi des pseudo-propriétés). Un service possède en général une interface ou plusieurs. Tous les noms d'interfaces commencent par un X (majuscule). La documentation dit que le service *exporte* des interfaces, ce qui signifie qu'il les met à la disposition du programmeur.

Une interface contient des méthodes. Ce sont des sous-programmes utilisables à partir de l'objet considéré. Les méthodes utilisent des arguments et renvoient éventuellement un résultat (sous-programme fonction). Chaque argument et chaque résultat peut être :

- un objet API,
- une donnée d'un type simple (booléen, entier, flottant),
- une séquence de données ou d'objets (présenté en OOoBasic sous forme de tableau),
- une constante nommée, qui est un type entier dont les valeurs possibles sont définies avec des noms qualifiés,
- une structure de données, qui est un regroupement de données accessibles individuellement.

Comme l'API est indépendante du langage, le type simple indiqué pour une donnée doit être « traduit » dans le type de données le plus proche pour le langage de programmation utilisé. Cela peut poser quelques difficultés : par exemple, OOoBasic ne possède ni type `Byte`, ni type `hyper`.

Un objet API, en dehors des constantes et des structures, comporte un service ou plusieurs. Il y a lieu de distinguer les services pris en charge, directement utilisables, et les services disponibles, qui peuvent être invoqués avec la fonction `createInstance`.

L'API réelle et l'API selon OOoBasic

OOoBasic a plusieurs avantages par rapport aux autres langages :

- Il est le plus intégré à l'application.
- Il est facile à apprendre.
- Il est conçu pour simplifier l'accès aux primitives de l'API : il n'impose pas de respecter les majuscules et minuscules dans l'emploi des noms de routines et de propriétés.
- Il connaît les valeurs des constantes nommées.
- Il permet d'utiliser directement les interfaces d'un objet, contrairement à Java™.
- Il permet d'utiliser comme une pseudo-propriété le couple de méthodes `get` et `set` manipulant la même donnée interne, contrairement à Java.

Pour toutes ces raisons, les exemples en Java, nombreux dans la documentation, sont bien plus difficiles à lire que leurs équivalents en OOoBasic. À titre d'exemple, voici un extrait de code en Java qui modifie un mot dans un texte Writer et modifie le curseur pour écrire en gras.

```
mxDocCursor = mxDocText.createTextCursor();
XSentenceCursor xSentenceCursor = (XSentenceCursor)
 UnoRuntime.queryInterface(XSentenceCursor.class, mxDocCursor);
xSentenceCursor.gotoNextSentence(false);
XWordCursor xWordCursor = (XWordCursor)
 UnoRuntime.queryInterface(XWordCursor.class, mxDocCursor);
xWordCursor.gotoNextWord(false);
xWordCursor.gotoNextWord(true);
mxDocText.insertString(xWordCursor, "hello ", true);
XPropertySet xCursorProps = (XPropertySet)
 UnoRuntime.queryInterface(XPropertySet.class, mxDocCursor);
xCursorProps.setPropertyValue("CharWeight",
 new Float(com.sun.star.awt.FontWeight.BOLD))
```

Ce code Java nécessite trois variables supplémentaires pour gérer le curseur, une pour chaque interface nécessaire. Voici pour comparaison le code OOoBasic équivalent, qui n'utilise que la variable curseur :

```
mxDocCursor = mxDocText.createTextCursor
mxDocCursor.gotoNextSentence(false)
mxDocCursor.gotoNextWord(false)
mxDocCursor.gotoNextWord(true)
mxDocText.insertString(mxDocCursor, "hello ", true)
mxDocCursor.CharWeight = com.sun.star.awt.FontWeight.BOLD
```

Lorsqu'il existe deux méthodes complémentaires simples, l'une servant à affecter une valeur à une donnée interne, l'autre servant à obtenir la valeur de cette donnée, OOoBasic

combine les deux sous la forme d'une pseudo-propriété, utilisable comme une simple variable. Voici deux codages OOoBasic qui réalisent exactement la même chose :

```
Dim Taille As Object
' codage autorisé par OOoBasic
Taille = dessin.Size
Taille.Width = Taille.Width * 2
dessin.Size = Taille

' application stricte de l'API
Taille = dessin.getSize()
Taille.Width = Taille.Width * 2
dessin.setSize(Taille)
```

L'utilisateur OOoBasic ne voit qu'une propriété `Size`. En réalité, cette propriété n'existe pas, mais elle est un raccourci vers deux méthodes :

- `setSize(`valeur`)` qui modifie la taille (en anglais *size*) de l'objet ;
- `getSize()` qui renvoie la taille de l'objet.

Pour retrouver dans l'API la description d'une propriété d'objet, il est donc nécessaire d'ajouter `get` ou `set` s'il s'agit d'une pseudo-propriété. Certaines données internes `Xyz` peuvent être manipulées seulement par `getXyz`, ou seulement par `setXyz`. Dans ces cas, la pseudo-propriété est restreinte à la seule lecture ou écriture.

On accède aux vraies propriétés des objets de l'API de manière plus simple en OOoBasic que dans d'autres langages comme Java. Exemple :

```
' codage autorisé par OOoBasic
couleur = UneCellule.CellBackColor
UneCellule.CellBackColor = RGB(255,255,204)

' deuxième manière, plus complexe, et aussi valide en OOoBasic
' ici la casse de CellBackColor doit être respectée
couleur =  UneCellule.getPropertyValue("CellBackColor")
UneCellule.setPropertyValue("CellBackColor", RGB(255,255,204))
```

Un objet dans une collection est accessible en OOoBasic par une indexation, comme si la collection était un tableau. En réalité, OOoBasic fait appel à la fonction `getByIndex` de la collection :

```
' codage autorisé par OOoBasic
uneFeuille = monDocument.Sheets(1)

' deuxième manière, plus complexe, et aussi valide en OOoBasic
uneFeuille = monDocument.Sheets.getByIndex(1)
```

Il existe aussi un raccourci OOoBasic pour accéder par son nom à un objet de collection. Nous ne l'avons pas employé car il peut donner des expressions ambiguës.

```
' codage recommandé
uneFeuille = monDocument.Sheets.getByName("Total")

' deuxième manière, valide en OOoBasic, déconseillée
uneFeuille = monDocument.Sheets.Total
```

ALLER PLUS LOIN **Liaison Basic - UNO**

L'interfaçage entre OOoBasic et UNO (le concept logiciel d'OpenOffice.org) est décrit plus longuement dans le « Developer's Guide », chapitre 3.4.3.

Les fonctions Basic dédiées à l'API

Tout au long de cet ouvrage, nous avons utilisé plusieurs fonctions OOoBasic qui facilitent l'accès à l'API. En voici une liste plus systématique.

StarDesktop

Il s'agit d'un objet prédéfini qui est le service de base d'OpenOffice.org. Il est un raccourci pour :

```
Dim monOOo As Object
monOOo = createUnoService("com.sun.star.frame.Desktop")
```

ThisComponent

Représente l'objet document en cours ; il est en général équivalent à :

```
Dim monDocument As Object
monDocument = StarDesktop.CurrentComponent
```

Cependant, si l'EDI est en premier plan, `CurrentComponent` renverra l'EDI et la macro ne fonctionnera pas, alors que `ThisComponent` continue à renvoyer l'objet document. C'est pourquoi `ThisComponent` est préférable.

Notez que `ThisComponent` n'est pas une variable, mais un appel de fonction : s'il y a plusieurs documents OpenOffice.org ouverts, elle renvoie le document OpenOffice.org dont la fenêtre est actuellement en avant-plan. C'est pourquoi il est préférable de ne l'appeler qu'au début de la macro et de sauver le résultat dans une variable interne.

CreateUnoStruct

Sert à obtenir une structure UNO, par exemple :

```
Dim uneProp As Object
uneProp = CreateUnoStruct("com.sun.star.beans.Property")
```

Il est plus simple de définir directement la variable :

```
Dim uneProp As New com.sun.star.beans.Property
```

CreateUnoValue

Sert à créer une donnée quelconque pour la transmettre à l'API. Cette méthode est prévue pour des problèmes très particuliers de conversion Basic-UNO.

CreateUnoService

Permet d'obtenir un objet capable de fournir le service donné en argument :

```
Dim demandePasse As Object
demandePasse = CreateUnoService( _
                   "com.sun.star.task.InteractionHandler")
```

Cette fonction est un raccourci pour :

```
Dim outilService As Object, demandePasse As Object
outilService = GetProcessServiceManager
demandePasse = outilService.createInstance( _
                   "com.sun.star.task.InteractionHandler")
```

CreateUnoListener

Permet à un programme de s'enregistrer comme auditeur d'un ensemble d'événements. Voir le principe au chapitre 19 et un exemple au chapitre 17.

CreateUnoDialog

En fait, cette fonction n'est concevable que dans un environnement OOoBasic, puisqu'elle sert à créer une boîte de dialogue (voir chapitre 15).

GetProcessServiceManager

Renvoie un objet permettant d'obtenir un service initialisé, soit par défaut, soit avec des arguments. Le premier cas est réalisé plus simplement avec `CreateUnoService`. Le deuxième cas est de la forme :

```
Dim outilService As Object, unService As Object
Dim args(1) ' un ou plusieurs arguments
outilService = GetProtcessServiceManager
' - initialiser le tableau args() avant cette instruction -
unService = outilService.createInstanceWithArguments( _
                  "com.sun.star.xxx.yyy.zzz", args())
```

Il existe aussi une troisième forme d'initialisation :

```
createInstanceWithArgumentsAndContext
```

Ces deux dernières formes ne sont utilisées que dans des cas assez particuliers.

IsUnoStruct

Renvoie `True` si la variable est une structure UNO. Exemple de structure UNO : le descripteur renvoyé par `createReplaceDescriptor` (voir chapitre 11). Cette fonction permet de distinguer une structure d'un objet véritable, ou d'une donnée simple.

Les variables de structure UNO sont de vraies valeurs, et non des références comme les variables sur les objets. C'est la raison des recopies nécessaires pour effectuer une modification, comme ici :

```
dim rognure as object
rognure = monImage.GraphicCrop
rognure.Bottom = -2000
monImage.GraphicCrop = rognure
```

EqualUnoObjects

Les variables représentant des objets sont en fait des références sur l'objet lui-même. Il est donc possible d'avoir deux variables pointant sur le même objet. Cette fonction permet de le vérifier.

HasUnoInterfaces

Renvoie `True` si l'objet en premier argument prend en charge toutes les interfaces des arguments suivants. Cette fonction est parfois utile pour rechercher un objet d'un type particulier dans une collection. Un autre moyen est d'utiliser la fonction `supportsService` des objets pour vérifier s'ils prennent en charge un service particulier.

Comprendre les messages d'erreur OOoBasic

Les messages d'erreur d'exécution de OpenOffice.org Basic sont souvent assez peu compréhensibles. Voici quelques informations pour vous aider.

Évitez les erreurs

Vérifiez la syntaxe de chaque module que vous avez écrit ou modifié. Il suffit de cliquer sur le bouton Compiler dans l'EDI. Faites-le pour chaque module modifié, car seul celui affiché par l'EDI est analysé.

Si vous utilisez des instructions complexes, ou des notations pointées en cascade, il devient difficile de trouver la raison de l'erreur d'exécution. Décomposez l'instruction en plusieurs instructions successives, et repérez celle qui part en erreur. Remontez ensuite en vérifiant chaque variable ou valeur utilisée.

Beaucoup d'erreurs sont dues à une faute d'orthographe. Relisez, relisez, passez à autre chose, revenez-y et relisez encore.

Respectez la casse des caractères (majuscule/minuscule) pour les arguments en chaîne de caractères, ainsi que pour les constantes API. Ce codage est un exemple typique :

```
DescrTri(0).Name = "SortFields"     ' Attention à la casse !
DescrTri(0).Value =  ConfigTri()
DescrTri(1).Name = "Orientation"    ' Attention à la casse !
' Constante API sur la ligne suivante, attention à la casse !
DescrTri(1).Value = com.sun.star.table.TableOrientation.ROWS
DescrTri(2).Name = "ContainsHeader" ' Attention à la casse !
DescrTri(2).Value = true
```

Erreur : variable indéfinie

Le message signifie plutôt : variable non définie. Il apparaît si vous avez utilisé l'option de déclaration obligatoire des variables :

```
Option Explicit
```

Quelque part sur la ligne en erreur, une variable n'a pas été déclarée préalablement. Souvent, il s'agit d'une simple faute de frappe. Relisez encore une fois... Et bénissez l'`Option Explicit`.

Erreur : variable objet non paramétrée

Premier cas

Vous utilisez une propriété ou une méthode d'une variable objet, mais cette variable est un objet vide (`Null`). Exemple :

```
Dim tata as object, toto as object
tata = toto.truc
```

L'erreur est que la variable `toto` n'est pas un véritable objet, puisque nous avons oublié de lui affecter un objet API. Basic ne peut pas trouver un `truc` dans cet objet !

Contre-exemple, ceci fonctionne sans erreur :

```
Dim tata as object, toto as object
tata = toto
```

En effet, cela revient à affecter à `tata` une variable objet de valeur `Null`.

En pratique, le cas arrive lorsqu'une fonction API est mal utilisée, mais qu'elle se contente de renvoyer un objet `Null`, sans erreur d'exécution :

```
Dim MonDocument As Object, lesFeuilles As Object
Dim AdresseDoc As String, PropFich()

AdresseDoc = "private:factory/calc"
MonDocument = StarDesktop.LoadComponentFromURL( _
               AdresseDoc,"_blank",0, PropFich)
lesFeuilles = MonDocument.Sheets
```

La dernière ligne déclenche une erreur parce que `MonDocument` est `Null`. En effet, il y a une erreur dans l'appel de `LoadComponentFromURL` qui l'empêche de créer un nouveau document Calc : on a affecté une valeur incorrecte à la variable `AdresseDoc`. Cherchez l'erreur !

Deuxième cas

Vous appelez une fonction de l'API ; un des arguments de cette fonction doit être un objet API, mais vous transmettez un autre type (valeur numérique, chaîne de caractères...), comme ici sur la dernière ligne :

```
Dim MonDocument As Object, MonTexte As Object
Dim MonCurseur  As Object

MonDocument = ThisComponent
MonTexte = MonDocument.Text
MonCurseur= MonTexte.createTextCursorByRange(1234)
```

Si vous n'utilisez pas `Option Explicit`, l'erreur peut aussi être une faute de frappe dans le nom d'une variable passée en argument à la fonction API : ce nom est alors interprété comme une nouvelle variable de type `Variant`. Si vous aviez utilisé l'option, le message d'erreur aurait été plus clair.

Erreur : utilisation incorrecte d'un objet

Cela signifie que vous essayez d'affecter à une variable objet quelque chose qui n'est pas un objet, par exemple une valeur numérique, ou une chaîne de caractères comme ici l'élément `Author` :

```
Dim tata as object
tata = ThisComponent.DocumentInfo.Author
```

Erreur : propriété ou méthode introuvable

Ce message est assez explicite. Vous cherchez à utiliser dans un objet (qui existe bien) une propriété ou une méthode que l'objet ne possède pas. Souvent, c'est une faute typographique, parfois c'est une mauvaise connaissance de l'API comme ici, pour un document Writer :

```
Dim tata as object
tata = thisComponent.DrawPages
```

La propriété est `DrawPage`, sans s...
... alors qu'un document Draw possède effectivement une propriété `DrawPages` !

Si l'erreur provient de ce que vous cherchez à deviner une propriété d'un objet API, vous perdez votre temps. Relisez plutôt ce livre, et utilisez la macro Xray.

Erreur : la sous-procédure ou procédure fonctionnelle n'est pas définie

Dans l'instruction en faute, Basic pense que vous essayez d'appeler une fonction ou procédure qu'il ne connaît pas. Vérifiez l'orthographe de la procédure que vous voulez utiliser, et vérifiez qu'elle existe bien. Exemple :

```
sleep(1000)
```

L'instruction ci-dessus donne cette erreur, car elle n'existe pas en Basic (mais elle existe dans d'autres langages). L'équivalent Basic est :

```
wait(1000)
```

Erreur : runtime exception

Traduction : anomalie à l'exécution. Sur un appel de fonction API, vous déclenchez un message d'erreur contenant :

```
Type: com.sun.star.uno.RuntimeException
Message:xxxxxx.
```

Parfois la section Type indique un nom d'exception plus spécifique. Bien souvent la section Message ne contient rien. C'est la fonction API qui vous renvoie un diagnostic suite à ses propres contrôles. Vérifiez chacun des arguments et relisez la documentation de l'API.

La raison est parfois plus subtile :

```
Dim MonDocument As Object, MonTexte As Object
Dim MonCurseur  As Object, MonCadre As Object

MonDocument = ThisComponent
MonTexte = MonDocument.Text
MonCurseur= MonTexte.createTextCursor
MonCadre = MonDocument.createInstance("com.sun.star.text.TextFrame")
MonCadre.Width = 10400 ' 104 mm largeur
MonCadre.Height = 2530 ' 25,3 mm de haut
MonCadre.String = "hello"
```

Ci-dessus, il s'agit d'une erreur de méthodologie : l'API ne permet pas d'initialiser la propriété `String` du cadre avant de l'avoir inséré dans le texte.

Erreur : cannot coerce argument type during corereflection call !

Ce message en anglais est issu de l'API. Il n'est compréhensible que par un informaticien connaissant les mécanismes internes de OOoBasic. En termes courants, il signifie : vous m'avez transmis un argument dont le type n'est pas compatible avec celui que j'attends, je ne sais pas l'utiliser.

En pratique, un des arguments servant à appeler la méthode API est incorrect. À vous de trouver de quel argument il s'agit, et en quoi il est incorrect. Relisez la documentation de l'API ou le chapitre correspondant dans ce livre. Cet exemple essaie d'insérer un cadre dans un document Writer :

```
Dim MonDocument As Object, MonTexte As Object
Dim MonCurseur  As Object, MonCadre As Object

MonDocument = ThisComponent
MonTexte = MonDocument.Text
MonCurseur= MonTexte.createTextCursor
MonTexte.insertTextContent( MonCurseur, MonCadre, false)
```

L'erreur est qu'on a oublié d'initialiser la variable `monCadre`.

L'API avec d'autres langages de programmation

Les autres langages de script de la version 2

Nous avons signalé au chapitre 3 les différents langages de scripts utilisables avec la version 2 d'OpenOffice.org et le niveau de leur intégration dans l'application. Voici le point de vue du programmeur.

JavaScript et BeanShell

Ces langages très similaires à Java nécessitent comme lui d'invoquer l'interface des méthodes à utiliser. Nous n'avons pas précisé les noms des interfaces dans ce livre car c'est parfaitement inutile avec Basic (et Python). La conséquence est une certaine lourdeur du codage, il est plus difficile à écrire et à relire. En contrepartie, les exemples du Developer's Guide étant écrits pour la plupart en Java sont facilement transposables.

Avec ces langages, la casse doit être respectée tant pour les noms de méthodes et de propriétés que pour tous les noms symboliques du programme. Les pseudo-propriétés de Basic n'existent pas, vous devez utiliser les méthodes de l'objet. Les pseudo-indexations de collections permises par OOoBasic doivent être remplacées par l'usage de la méthode `getByIndex()` de l'objet collection. Pour des interactions simples avec l'utilisateur, vous devrez développer des équivalents de `MsgBox` et `InputBox`.

Python

Dans Python, la casse doit être respectée tant pour les noms de méthodes et de propriétés que pour tous les noms symboliques du programme. Les pseudo-indexations de collections permises par OOoBasic doivent être remplacées par l'usage de la méthode `getByIndex()` de l'objet collection. Pour des interactions simples avec l'utilisateur, vous devrez développer des équivalents de `MsgBox` et `InputBox`.

Dans l'état actuel (pré-version 1.9 ou version 2.0.0), il est nécessaire de créer et éditer les scripts Python dans un éditeur externe produisant un codage source avec des fins de ligne façon Unix, même sous MS-Windows. Le fichier doit être mis dans un sous-répertoire `user/Scripts/python/` ou `share/Scripts/python`. Dans le premier cas, le sous-répertoire `python/` doit être créé par le développeur. Dans un document, le script doit être inséré en manipulant la structure du fichier document. Actuellement, le `sys.path` fourni ne permet d'importer que des modules situés dans `{installation}/program/` et ceux du `python-core`.

Tout ceci semble décourageant, pourtant les scripts obtenus gèrent l'API aussi facilement qu'en OOoBasic : pas d'invocation d'interface, utilisation possible des pseudo-propriétés. De plus, contrairement à Basic, il est possible avec Python de créer de véritables compo-

sants UNO intégrés à OpenOffice.org. Ils sont alors utilisés comme de nouveaux services, même dans une macro Basic.

Le programmeur profite des qualités de Python : l'indentation obligatoire facilite la relecture, les variables ne sont pas déclarées mais leur usage est contrôlé, elles peuvent changer de type dynamiquement, les algorithmes sont plus simples grâce aux fonctions puissantes intégrées dans Python et ses modules principaux, la gestion des erreurs est celle des langages modernes, et la programmation objet facilite la conception de programmes complexes.

Pour montrer la similarité de programmation API, voici le portage en Python de la portion de codage Basic indiqué plus haut, section « L'API réelle et l'API selon OOoBasic » :

```
mxDocCursor = mxDocText.createTextCursor()
mxDocCursor.gotoNextSentence(False)
mxDocCursor.gotoNextWord(False)
mxDocCursor.gotoNextWord(True)
mxDocText.insertString(mxDocCursor, "hello ", True)
mxDocCursor.CharWeight = uno.getConstantByName("com.sun.star.awt.FontWeight.BOLD")
```

Avant de vous lancer, lisez ces pages fondamentales en anglais sur le site udk :

- conversions entre types API et types Python, création de composant OpenOffice.org en Python : http://udk.openoffice.org/python/python-bridge.html ; cette page est aussi valide avec la version 1.1 d'OpenOffice.org ;
- mise en place de scripts Python et sorties de messages d'erreur dans OpenOffice.org 2.0 : http://udk.openoffice.org/python/scriptingframework/index.html

Piloter OpenOffice.org avec Microsoft COM

Sous MS-Windows, il est possible, à partir d'un langage de programmation indépendant, d'utiliser COM pour piloter l'application OpenOffice.org et manipuler des documents en utilisant son API telle qu'elle a été décrite dans cet ouvrage. Le SDK de la version 2.0 d'OpenOffice.org fournit plusieurs exemples :

- dans `examples/CLI/` : langages VB.NET et C#,
- dans `examples/OLE/` : langages Delphi et VbScript.

Dans la section CodeSnippets de OOoForum (http://www.oooforum.org/forum/) vous trouverez notamment le fil de discussion Using COM for OOo with different languages, qui présente des exemples de programmation COM avec divers langages dont VB, C#, Python, LotusScript, Delphi.

Mises à part l'initialisation de la connexion OLE et la syntaxe propre au langage, le cœur des instructions peut être remarquablement similaire à ce qui a été exposé. Voici l'équivalent en langage Delphi™ du codage exemple cité plus haut en Java et OOoBasic :

```
const
  _awtFontWeightBOLD = 150.000000;
var
  mxDocText, mxDocCursor : Variant;
// extrait du codage
  mxDocCursor:= mxDocText.createTextCursor;
  mxDocCursor.gotoNextSentence(false);
  mxDocCursor.gotoNextWord(false);
  mxDocCursor.gotoNextWord(true);
  mxDocText.insertString(mxDocCursor, 'hello ', true);
  mxDocCursor.CharWeight:= _awtFontWeightBOLD;
```

Comme nous n'avons pas un accès direct aux constantes nommées, nous devons définir celles dont nous avons besoin. Les valeurs de ces constantes sont lisibles dans les fichiers de l'arborescence `/idl` du SDK. Les constantes nommées « enum » sont implicitement numérotées à partir de zéro.

En Delphi, les noms des propriétés employés comme ci-dessus ne sont pas sensibles à la casse. Il en est de même pour les noms de méthodes. Les pseudo-propriétés sont aussi utilisables. Les tableaux réels, comme `ElementNames` fournis par certains objets, peuvent être indexés normalement. En revanche, les pseudo-indexations de collections permises par OOoBasic doivent être remplacées par l'usage de la méthode `getByIndex()` de l'objet collection.

La boîte à outils Delphi_OOo disponible en français sur le site fr.OpenOffice.org facilite la programmation OOo grâce à :

- une « unité » Delphi contenant des fonctions simplifiées d'accès à l'API,
- une « unité » listant toutes les constantes API et un programme pour la mettre à jour à partir d'un SDK,
- une « unité » apportant les fonctionnalités de Xray,
- des exemples de programmation,
- un mode d'emploi avec des conseils pour convertir une macro Basic en équivalent Delphi.

RESSOURCE

Le fil de discussion « Using COM for OOo with different languages » donne des exemples en Visual Basic, C, C++, C#, Python, Perl, Ruby, TCL et Delphi.

La documentation de l'API (Software Development Kit)

L'ensemble de la documentation est appelé SDK (Software Development Kit), conçu à l'origine pour un environnement de développement en Java ou en C++. Il se compose de diverses parties, d'où sa taille respectable :

- le « Developer's Guide », qui est un hypertexte expliquant la conception de l'API OpenOffice.org ;
- une version PDF imprimable du « Developer's Guide », plus de 1000 pages pour ceux qui veulent détruire les forêts, ou pour imprimer quelques pages d'un chapitre ;
- la référence IDL (Interface Definition Language) qui est un gigantesque hypertexte documentant (presque) tous les objets (au sens le plus général) de l'API ;
- les fichiers `*.idl` ayant servi à constituer la référence ; ce sont des fichiers texte où on peut lire les valeurs des constantes nommées ;
- une référence pour le développement en Java ;
- une référence pour le développement en C++ ;
- des exemples (repris dans le « Developer's Guide ») ;
- la spécification des formats XML utilisés ;
- divers outils de développement.

Le SDK est disponible pour consultation en ligne, à la page http://api.openoffice.org/ où vous trouverez aussi la dernière version à télécharger.
Il existe une version du SDK pour OOo 1.1.0 et une autre pour OOo 2.0. Dans cette dernière, vous trouverez aussi des informations concernant les versions 1.1.2 et suivantes, qui ne sont pas intégrées au SDK 1.1.0

Dans sa version hypertexte, le « Developer's Guide » contient de nombreux liens vers la référence IDL, et réciproquement, ce qui rend une lecture interactive souvent préférable à une lecture imprimée. D'ailleurs, la référence IDL est disponible seulement en version HTML.

La principale difficulté lorsqu'on étudie cette documentation est qu'elle est conçue pour un développeur Java ou C++ chargé de faire évoluer OpenOffice.org ou d'en réaliser une variante. Ce point de vue est intimement mélangé avec les descriptions des fonctionnalités disponibles, ce qui en rend la lecture assez difficile. D'autre part, et principalement dans le « Developer's Guide », les exemples sont donnés en langage Java, notablement plus lourd que OOoBasic.

La référence IDL est rédigée par les programmeurs eux-mêmes lors du développement, et compilée ensuite automatiquement. L'ennui, c'est que les programmeurs sont rarement intéressés par l'écriture de la documentation. Aussi est-elle parfois décevante par son aspect répétitif et ses lacunes.

Comment s'y retrouver ?

Nous vous conseillons d'installer le SDK sur votre ordinateur, pour un accès plus rapide, et d'utiliser un navigateur Internet capable d'afficher de multiples pages accessibles sous forme d'onglets, par exemple Mozilla Firefox (http://frenchmozilla.org/).

Dans une installation standard sous MS-Windows, la page d'entrée se trouve sur votre disque à l'adresse `C:\OpenOffice.org1.1_SDK\index.html`. En fait, cette page est assez peu utilisée. Vous commencerez plutôt par la page « Developer's Guide » ou la page « IDL Reference », qui sont accessibles depuis la première.

La page d'introduction au « Developer's Guide » vous renvoie vers le sommaire (*Table of contents*) de ce qui est en fait un véritable livre. Beaucoup de recherches générales partiront de ce sommaire. Comme certaines pages sont très lentes à charger sur un navigateur, il peut être plus pratique d'afficher la version PDF du « Developer's Guide ».

La première page de la référence IDL liste les différents modules qui composent le module `Star`, l'ensemble de l'application. Tout est ensuite décomposé en un arbre avec de nombreuses ramifications. La position d'une page dans cet arbre est rappelée en haut à gauche des pages, par exemple `::com::sun::star::`, et l'organisation des répertoires contenant les pages HTML reflète exactement cette hiérarchie. Lorsque nous indiquons une référence de documentation comme `com.sun.star.drawing.LineProperties`, vous afficherez la page correspondante en suivant, depuis la première page de l'IDL, le lien intitulé `drawing`, puis dans la page obtenue le lien intitulé `LineProperties`, comme on peut le voir sur la figure A-1.

L'autre moyen d'accès à l'IDL, à partir de n'importe laquelle de ses pages, est d'utiliser le lien Index, sur la première ligne du haut de la page. Il vous affiche la page `A` d'un dictionnaire. Si vous cherchez la documentation sur `LineProperties`, affichez la page `L` et recherchez ce mot dans celle-ci, en début de ligne. Vous repérerez la ligne :

```
LineProperties - service ::com::sun::star::drawing:: .LineProperties
```

Ici, il s'agit d'un service dont la page de description est accessible par lien hypertexte. Dans bien des cas, vous obtiendrez plusieurs lignes avec le même nom, car plusieurs types d'objets ayant une propriété ou une méthode similaire ont normalement le même nom. Ce sera à vous de déterminer lequel correspond à votre contexte, grâce aux autres informations de la ligne.

Un dernier moyen, très rapide, d'accéder directement à la bonne page de l'IDL consiste à utiliser l'outil XRay, dont nous avons déjà parlé.

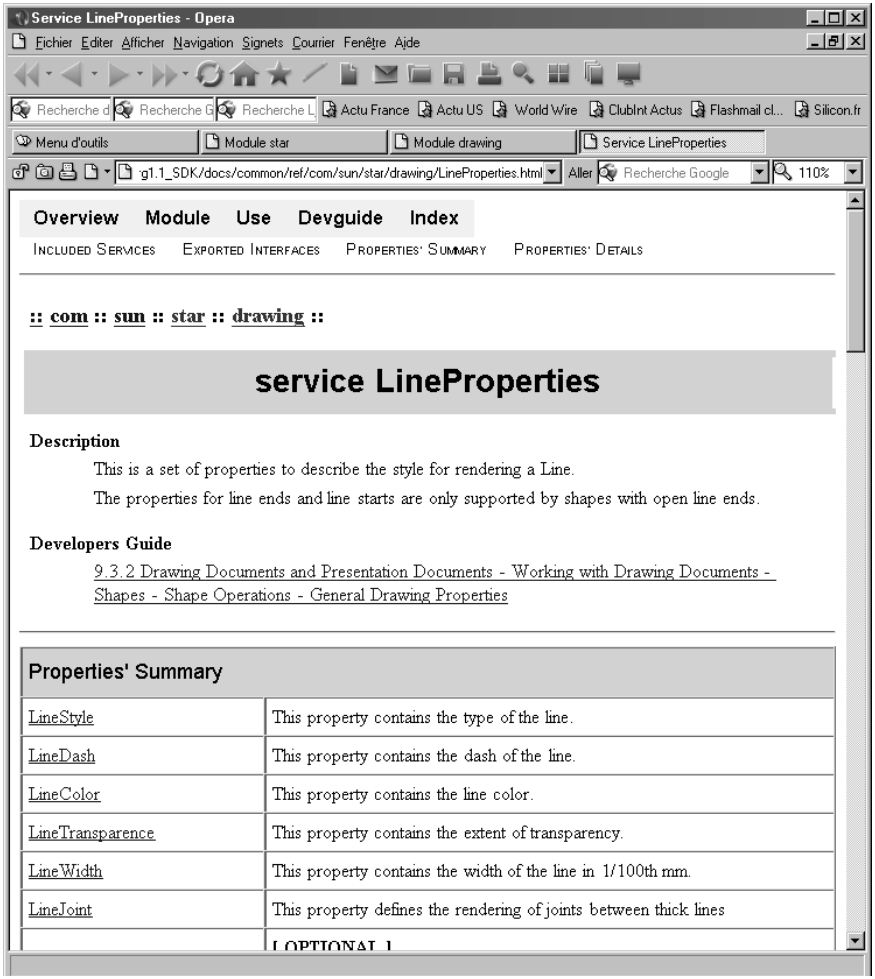


Figure A–1
Une page de la référence IDL

Conclusion

Une compréhension de l'API elle-même hors du contexte des macros et une connaissance des sources d'information disponibles nous semblent nécessaires pour parvenir à en tirer parti.

L'annexe suivante offre un panorama de routines utilitaires, qui complète utilement les techniques vues au chapitre 19.

B

Routines utilitaires

Les sous-programmes présentés dans cette annexe seront souvent utilisés dans vos projets de macros. Certains sont soumis à la licence LGPL, décrite sur le site http://www.gnu.org/copyleft/lesser.html.

LICENCE LGPL

Une traduction non officielle de la licence LGPL est disponible sur le site http://www.linux-france.org/article/these/licence/lgpl/lgpl_monoblock.html. Cette licence stipule que vous pouvez utiliser ces routines dans vos programmes et les améliorer, à condition de maintenir les indications de licence et d'auteur initial.

La première partie des routines décrites ici ont déjà été utilisées dans divers chapitres de ce livre. Elles peuvent servir de base à des constructions plus élaborées, notamment sur le plan du traitement d'erreurs.

Tableaux de propriétés

Certains objets de l'API se présentent sous la forme d'un tableau de propriétés dont la liste est prédéfinie. C'est le cas des descripteurs de tri pour Calc et Writer.

La routine `printProps` affiche les noms de ces propriétés et leur rang dans le tableau. La fonction `getPropVal` renvoie la valeur de la propriété dont le nom est en argument. La routine `setPropVal` affecte une valeur à une propriété.

```
rem CodeAnnexeB-01.sxw   bibli : Proprietes Module1
Option Explicit

' affiche les noms de propriétés du tableau descr
Sub printProps(descr As Variant)
Const Titre = "Tableau de propriétés"
Dim x As Long, liste As String
for x = 0 to UBound(descr)
  liste = liste & "Index " & x & " : " _
                &  descr(x).Name & chr(13)
next
MsgBox(liste, 0, Titre)
End Sub


' affecte la valeur valProp à la propriété nomProp
Sub setPropVal(descr As Variant, _
               nomProp As String, valProp As Variant)
Const Titre = "Tableau de propriétés"
Dim x As Long
for x = 0 to UBound(descr)
  if descr(x).Name = nomProp then
    descr(x).Value = valProp
    Exit Sub
  end if
next
MsgBox("Propriété inconnue : " & nomProp, 16, Titre)
End Sub


' renvoie la valeur de la propriété nomProp
Function getPropVal(descr As Variant, nomProp As String)_
                    As Variant
Const Titre = "Tableau de propriétés"
Dim x As Long
for x = 0 to UBound(descr)
  if descr(x).Name = nomProp then
    getPropVal = descr(x).Value
    Exit Function
  end if
next
MsgBox("Propriété inconnue : " & nomProp, 16, Titre)
End Function
```

Collections

Certaines collections d'objets ne permettent pas un accès direct par le nom d'objet. La fonction `getIndexByName` recherche dans la collection l'objet dont le nom est passé en argument. En cas de succès, elle renvoie l'index correspondant ; en cas d'échec, elle affiche une erreur et renvoie un index hors-limites.

```
rem CodeAnnexeB-01.sxw   bibli : Collections Module1
Option Explicit

' renvoie l'index de l'élément ayant le nom donné en argument
Function getIndexByName(collection As Object, _
                        leNom As String) As Long
Const Titre = "Collection"
Dim x As Long
for x = 0 to collection.Count -1
  if collection(x).Name = leNom then
    getIndexByName = x ' renvoyer l'index correspondant au nom
    Exit Function
  end if
next
MsgBox("Nom inconnu : " & leNom, 16, Titre)
getIndexByName = -100 ' valeur d'index hors-limites
End Function
```

Coordonnées de cellules

L'API fournit des fonctions pour convertir en adresse numérique une « adresse utilisateur » de cellule ou de zone de cellules. Les fonctions de cette bibliothèque réalisent l'inverse, sur toute l'étendue des adresses possibles. La version 2 d'OpenOffice.org devrait repousser les limites de Calc au-delà de 32 000 lignes. Ces macros peuvent facilement être adaptées, mais plusieurs fonctions Basic pourront aussi être modifiées par cette évolution.

La fonction `adrZoneString` renvoie la chaîne de caractères correspondant aux coordonnées de la zone de cellules passée en argument. La fonction `adresseString` effectue le même travail pour une adresse de cellule. Les autres fonctions sont utilisées par les deux précédentes.

Notez que ces routines effectuent un contrôle de vraisemblance sur les coordonnées fournies. En cas d'anomalie, la chaîne renvoyée comportera des caractères « ?? ».

```
rem CodeAnnexeB-01.sxw   bibli : Cellules Module1
Option Explicit

' convertit une adresse de zone de cellules en adresse textuelle
Function adrZoneString(maDoc As Object, adrZone As Object) As String
Dim resu As String
resu = maDoc.Sheets(adrZone.Sheet).Name & "." & _
               alphaXY(adrZone.StartColumn, adrZone.StartRow)
if (adrZone.StartColumn <> adrZone.EndColumn) or _
   (adrZone.StartRow <> adrZone.EndRow) then
  resu = resu & ":" & alphaXY(adrZone.EndColumn, adrZone.EndRow)
end if
adrZoneString = resu
End Function


' convertit une adresse de cellule en adresse textuelle
Function adresseString(maDoc As Object, adrCellule As Object) As String
adresseString = maDoc.Sheets(adrCellule.Sheet).Name & "." & _
               alphaXY(adrCellule.Column, adrCellule.Row)
End Function

' convertit une coordonnée XY en coordonnée alphanumérique
Function alphaXY(X As Long, Y As Long) As String
if (Y>=0) and (Y<32000) then
  alphaXY = lettreColonne(X) & CStr(Y +1)
else
  alphaXY = lettreColonne(X) & "??"
end if
End Function


' convertit numéro de colonne 0...255 en lettres A...IV
Function lettreColonne(n As Long) As String
Select Case n
Case > 255
  lettreColonne = "??"
Case < 0
  lettreColonne = "??"
Case < 26
  lettreColonne = _Lettre(n)
Case Else
  lettreColonne = _Lettre((n \ 26) -1) & _Lettre(n Mod 26)
End Select
End Function

' fonction interne
Function _Lettre(p As Long) As String
_Lettre = chr(Asc("A") +p)
End Function
```

Rechercher un objet par son nom

La fonction `findObjectByName` est inspirée des travaux de Danny Brewer. Cette fonction recherche dans une page de dessin un objet dont le nom est donné en argument. En cas d'échec, la fonction renvoie la valeur `Null`.

```
rem CodeAnnexeB-02.sxw   bibli : Dessin Module1
Option Explicit

' retrouve un objet à partir de son nom
Function FindObjectByName(unePage As Object, _
         nomObj As String, Optional service As String) As Object
Dim objX As Object, x As Long
  For x = 0 To unePage.Count - 1
    objX = unePage(x)
    If objX.Name = nomObj Then
      if IsMissing(service) then
        FindObjectByName = objX ' objet trouvé
        Exit Function
      else
        if objX.supportsService(service) then
          FindObjectByName = objX ' objet trouvé
          Exit Function
        end if
      end if
    EndIf
  Next
End Function ' renvoie Null en cas d'échec
```

Le paramètre `service` est optionnel. Il permet de ne rechercher qu'un objet proposant le service indiqué. En effet, une page de dessin peut contenir différentes sortes d'objets, et vous pourriez par exemple obtenir une image ayant le nom du dessin que vous cherchez (par programmation, on peut donner le même nom à plusieurs objets d'une page). En vérifiant que l'objet obtenu reconnaît un service caractéristique du type d'objet recherché, nous effectuons une vérification supplémentaire. Le tableau B-1 indique quel service caractérise un objet.

Tableau B–1 Services caractéristiques

Type d'objet recherché	Service caractéristique
Dessin, sauf 3D	`com.sun.star.drawing.LineProperties`
Dessin 3D	`com.sun.star.drawing.Shape3DScene`
Image	`com.sun.star.drawing.GraphicObjectShape`
Objet OLE2	`com.sun.star.drawing.OLE2Shape`

Par exemple, pour rechercher seulement une forme dessinée appelée « F3 », nous écrirons :

```
Dim sv As String
sv = "com.sun.star.drawing.LineProperties"
maForme = FindObjectByName(maPage, "F3", sv)
```

En effet, le service `Shape`, trop général, est également proposé par une image. En revanche, vous pouvez être plus précis et exiger par exemple une `EllipseShape`.

Rechercher un diagramme par son nom

La fonction `FindChartByObjName` recherche dans une feuille Calc un objet diagramme à partir du nom de forme défini par l'interface utilisateur. Elle utilise la fonction précédente, en recherchant un objet `OLE2`. Toutefois, pour être certain qu'il s'agit d'un diagramme, elle vérifie que le sous-objet `Model` propose le service de diagramme. La fonction renvoie la valeur `Null` en cas d'échec.

```
rem CodeAnnexeB-02.sxw   bibli : Dessin Module1
Option Explicit

' retrouve un objet diagramme à partir du nom de l'objet
Function FindChartByObjName(laFeuille As Object, _
                            nomDiag As String) As Object
Dim dessin As Object, sv As String

sv = "com.sun.star.drawing.OLE2Shape"
dessin = FindObjectByName(laFeuille.Drawpage, nomDiag, sv)
if not IsNull(dessin) then
  if dessin.Model.supportsService(_
              "com.sun.star.chart.ChartDocument") then
    FindChartByObjName = dessin.Model
  end if
end if
End Function ' renvoie Null en cas d'échec
```

Redimensionner une image

Le sous-programme `resizeImageByWidth` redimensionne une image à une largeur donnée, en gardant ses proportions. Il obtient de l'objet image l'objet `GraphicObjectFillBitmap`, qui donne accès aux informations sur l'image elle-même. Dans ce dernier objet, la structure `Size` nous donne la taille de l'image mesurée en pixels. Une règle de trois nous permet d'en déduire les dimensions en 1/100 de millimètres.

```
rem CodeAnnexeB-02.sxw   bibli : Images Module1
Option Explicit

Sub resizeImageByWidth(uneImage As Object, largeur As Long)
Dim leBitMap As Object, Proportion As Double
Dim Taille1 As New com.sun.star.awt.Size

LeBitmap = uneImage.GraphicObjectFillBitmap
Taille1 = LeBitMap.Size ' taille en pixels !
Proportion = Taille1.Height / Taille1.Width
Taille1.Width = largeur ' largeur en 1/100 de mm
Taille1.Height = Taille1.Width * Proportion
uneImage.Size = Taille1
End Sub
```

Vous pourrez facilement réaliser sur ce modèle un sous-programme redimensionnant selon une hauteur donnée.

Autre variation, le sous-programme `resizeImageByDPI` redimensionne l'image en respectant une densité de points, donnée en DPI.

```
rem CodeAnnexeB-02.sxw   bibli : Images Module2
Option Explicit

Sub resizeImageByDPI(uneImage As Object, DPI As Long)
Dim leBitMap As Object, Proportion As Double
Dim Taille1 As New com.sun.star.awt.Size
Const pouce = 2540 ' longueur en 1/100 de mm

LeBitmap = uneImage.GraphicObjectFillBitmap
Taille1 = LeBitMap.Size ' taille en pixels !
Proportion = pouce / DPI
Taille1.Width =  Taille1.Width * Proportion
Taille1.Height = Taille1.Height * Proportion
uneImage.Size = Taille1
End Sub
```

Traduire un nom de style

Le nom d'un style obtenu par exemple avec la propriété `ParaStyleName` est le nom interne, en anglais. Le nom affiché dans le styliste est dans la langue locale.

La fonction `getLocaleStyleName` renvoie le nom localisé correspondant à un nom interne. Elle emploie trois arguments :

- l'objet document (qui contient les styles),
- le nom de la famille de styles,
- le nom anglais du style.

Cette fonction ne donnera pas de bons résultats dans Impress pour les styles de la famille `Standard` (voir le chapitre 13).

```
rem CodeAnnexeB-03.sxw   bibli : NomsStyles Module1
Option Explicit

' renvoie le nom localisé d'un style
Function getLocaleStyleName(leDoc As Object, _
                 fam As String, nomStyle As String) As String
Dim uneFamille As Variant
Dim desStyles As Object, unStyle As Object

on Error Goto pbStyle
desStyles = leDoc.StyleFamilies.getByName(fam)
unStyle = desStyles.getByName(nomStyle)
getLocaleStyleName = unStyle.DisplayName
On Error Goto 0
exit function

pbStyle:
  On Error Goto 0
  getLocaleStyleName = "????"
End Function
```

Nous avons utilisé un traitement d'erreur pour renvoyer des points d'interrogation sur les cas d'échec, notamment si le nom de famille de styles ou le nom de style est inconnu.

Le document comporte un exemple de routine utilisant la fonction. Vous remarquerez que si vous entrez un nom localisé, la fonction renvoie ce même nom, grâce à la souplesse de `getByName`.

Rappel des routines déjà décrites dans les chapitres

Dialogue

La fonction `Dialogue` crée un objet dialogue connaissant le nom de la boîte de dialogue et sa bibliothèque. Le sous-programme `CenterDialog` centre un nouveau dialogue par rapport à un dialogue père. Voir le chapitre 16 pour ces deux routines.

Tableur et base de données

Les fonctions suivantes se trouvent dans le fichier `CalcSQL.sxc` qui se trouve dans le répertoire regroupant les exemples du chapitre 17.

La fonction `CALCSQL1` sert à effectuer dans une cellule Calc une requête SQL sur une base de données. Elle renvoie un tableau de résultats.

La fonction `CALCSQL2` importe le résultat d'une requête SQL depuis une base de données. Le résultat est obtenu dans un tableau de cellules d'une feuille du tableur.

Rechercher la forme d'un contrôle de formulaire

La fonction `FindCtrlShapeByName` recherche dans une page de dessin une forme correspondant à un contrôle de formulaire dont le nom est donné en argument. Voir le chapitre 18.

Création et décompression d'un fichier ZIP

Plusieurs sous-programmes de gestion de fichier ZIP sont décrits dans le chapitre 19.

Les routines suivantes n'ont pas été utilisées dans les chapitres précédents, mais elles sont d'une incontestable utilité.

Conversion date-heure vers heure, minute, seconde

Les fonctions Basic `Hour`, `Minute`, `Second` sont incorrectes sur les versions d'OpenOffice.org antérieures à 1.1.4, et corrigées à partir de celle-ci et sur la version 2.0.0. Pour les utilisateurs des versions posant problème, nous proposons le sous-programme `Time_HMS` qui réalise une conversion correcte. Il a été testé sur les 86 400 valeurs possibles de la partie heure.

Le sous-programme `Time_HMS` reçoit une valeur de date-heure au format interne, et en extraie les valeurs d'heure, minute, seconde.

- Argument 1 : date-heure
- Argument 2 : variable qui recevra la valeur d'heure (0 à 23)

- Argument 3 : variable qui recevra la valeur de minute (0 à 59)
- Argument 4 : variable qui recevra la valeur de seconde (0 à 59)

Codage du sous-programme :

```
rem CodeAnnexeB-04.sxw   bibli : Standard Module1
Option Explicit

'Convertit une heure de type Date en heure, minute, seconde
Sub Time_HMS(temps As Date, _
         hr As Integer, mn As Integer, sec As Integer)
Dim s2 As Long, m2 As Long, h2 As Long
Dim d3 As Double, s3 As Long

d3 = CDbl(temps) ' conversion en Double
d3 = d3 - Fix(d3) ' garder la partie fractionnaire
s3 = CLng(d3 * 86400) ' convertir le temps en secondes
' le calcul se poursuit avec des entiers Long
s2 = s3 Mod 60
s3 = (s3 -s2) / 60
m2 = s3 Mod 60
h2 = (s3 -m2) / 60
hr = h2
mn = m2
sec = s2
End Sub
```

Exemple d'utilisation :

```
rem CodeAnnexeB-04.sxw   bibli : Standard Module2
Option Explicit

Sub Exemple_Time_HMS()
Dim t1 As Date
Dim h1 As Integer, m1 As Integer, s1 As Integer

t1 = Now
Time_HMS(t1, h1, m1, s1)
MsgBox(CStr(t1) & chr(13) & _
       "    " & h1 & ":" & m1 & ":" & s1)
End Sub
```

Traitement des chaînes de caractères longues

Sur les versions 1.1 et antérieures d'OpenOffice.org, les instructions OOoBasic `InStr`, `Left`, `Right` et `Asc` ne traitent pas de manière correcte les chaînes de caractères de plus de

32 767 caractères. Elles sont corrigées à partir de la version 2.0.0. Pour les utilisateurs des versions posant problème, le document `CodeAnnexeB-07.sxw` contient dans la bibliothèque `LongString` des fonctions de remplacement : `InStrL`, `LeftL`, `RightL` et `AscL`. Elles utilisent les mêmes arguments que les fonctions d'origine.

Remplacer un motif partout dans une chaîne de caractères

La fonction `RemplaceChaine`, disponible sur la page Internet http://fr.openoffice.org/index.html a pour but de remplacer une séquence de caractères par une autre, chaque fois qu'elle apparaît dans une chaîne de caractères. Un exemple trivial consiste à remplacer les caractères de séparation dans un chemin de fichier. Les arguments de la macro sont, dans l'ordre :

- la chaîne initiale,
- la séquence à rechercher (chaîne de caractères),
- la séquence qui doit la remplacer (chaîne de caractères),
- un indicateur mis à `True` pour que la recherche distingue la casse,

La fonction renvoie une chaîne de caractères modifiée.

Cette macro est codée afin de fonctionner aussi sur des chaînes de plus de 32 767 caractères.

Trier un tableau de données

La fonction `TriShell`, disponible sur la page Internet http://fr.openoffice.org/index.html effectue le tri d'un tableau (`Array`) de `Variant`. Elle est facilement modifiable pour trier une structure quelconque.

Le cœur de la macro est la comparaison des éléments et leur échange. Adaptez-le à vos besoins, par exemple une comparaison de `String` tenant compte de la casse.

Ainsi que le précise le texte du document contenant la macro, des théoriciens ont étudié avant nous comment optimiser un tri, alors c'est l'occasion de ne pas réinventer la roue (en plus mal). La méthode *Shell* est très efficace sans nécessiter un codage récursif. Elle est probablement assez rapide pour vos besoins en Basic, et pas plus compliquée à employer qu'un tri *Bulle*, méthode évidente mais très lente.

Obtenir des adresses URL de fichiers et répertoires

Le document `CodeAnnexeB-05.sxw` contient plusieurs fonctions facilitant l'analyse d'adresses de chemins de fichiers. Elles sont écrites en anglais pour être comprises du plus grand nombre.

Toutes ces fonctions (voir le tableau B-2) utilisent des adresses au format URL. Toutes les adresses de répertoires doivent se terminer par le caractère `/`.

Tableau B–2 Fonctions d'adresses URL

Fonction	Argument	Résultat
`getDirectory`	Chemin d'un fichier	Chemin du répertoire contenant le fichier
`getParentDir`	Chemin d'un répertoire	Chemin du répertoire parent
`getFullFileName`	Chemin d'un fichier	Nom et extension du fichier, exemple : `monfichier.sxw`
`getFileNameOnly`	Chemin d'un fichier	Nom du fichier sans l'extension, exemple : `monfichier`
`getFileExt`	Chemin d'un fichier	Extension du fichier, avec le point, exemple : `.sxw`

Copier-coller avec le presse-papiers

C'est avec un peu de scrupules que nous présentons ces deux macros, qui ont été réalisées avec l'enregistreur de macros ; c'est cependant une solution simple et efficace et nous en avons amélioré le code obtenu.

```
rem CodeAnnexeB-06.sxw   bibli : PressePapier Module1
Option Explicit

Sub CopierDansPressePapier()
dim fenetreDoc as object, dsp as object
fenetreDoc = StarDesktop.CurrentFrame
dsp = createUnoService("com.sun.star.frame.DispatchHelper")
dsp.executeDispatch(fenetreDoc, ".uno:Copy", "", 0, Array())
End Sub

Sub CollerDepuisPressePapier()
dim fenetreDoc as object, dsp as object
fenetreDoc = StarDesktop.CurrentFrame
dsp = createUnoService("com.sun.star.frame.DispatchHelper")
dsp.executeDispatch(fenetreDoc, ".uno:Paste", "", 0, Array())
End Sub
```

Les deux macros doivent être lancées sans utiliser le bouton de l'EDI. En premier plan doit s'afficher une fenêtre gérée par OpenOffice.org, en général celle d'un document ouvert. Notez que même la fenêtre de l'EDI conviendra !

La première macro copie dans le presse-papiers la zone sélectionnée par l'utilisateur (ou sélectionnée par macro puis rendue visible pour l'utilisateur, comme expliqué dans la partie Writer). Pour cela, elle récupère la fenêtre courante affichée par OpenOffice.org (alors que le codage de l'enregistreur de macros récupère la fenêtre courante du document). L'utilisation du service `DispatchHelper` est reprise du code de l'enregistreur de macros, le nom de la variable ayant été changé pour raison de mise en pages.

La deuxième macro copie le presse-papiers dans la fenêtre OpenOffice.org en premier-plan, en écrasant éventuellement une zone sélectionnée. L'utilisateur peut afficher une autre application OpenOffice.org, alors que le codage original ne le permettrait pas. La seule différence avec la première macro réside dans le deuxième argument de la méthode `executeDispatch`.

À NOTER L'enregistreur de macros

Utiliser l'enregistreur de macros est quelquefois frustrant, mais cela peut être un moyen simple de couvrir une fonctionnalité difficile ou impossible à implémenter avec l'API.

La liste des commandes du dispatch est disponible sur internet aux adresses suivantes :

- pour la version 1.1 d'OpenOffice.org :
  - http://framework.openoffice.org/files/documents/25/1042/commands_11beta.html
- pour la version 2.0 d'OpenOffice.org :
  - http://framework.openoffice.org/files/documents/25/2570/commandsReference.html

Les commandes de la forme .uno:xxxxx sont dans la première colonne des tableaux. L'utilisation des constantes numériques (ID) de cette page est à proscrire. Elles sont à usage interne et peuvent changer à tout moment.

Conclusion

Les macros exposées dans ce chapitre sont directement opérationnelles. Elles illustrent la notion de « ré-utilisabilité » qui permet de gagner du temps et de l'énergie en pérennisant les développements. Il est inutile de ré-écrire plusieurs fois le même code pour effectuer la même action. Usez et abusez de telles routines ; elles constitueront progressivement une boîte à outils adaptée à vos besoins, et vous permettront de rester concentré sur l'objectif premier de la macro que vous êtes en train de concevoir.

Voyons maintenant quelles richesses sont disponibles sur l'Internet.

C

Ressources disponibles sur Internet

De nombreuses informations, outils et compléments sur la programmation OpenOffice.org sont disponibles sur Internet. Ils sont une source toujours renouvelée de savoir et d'inspiration.

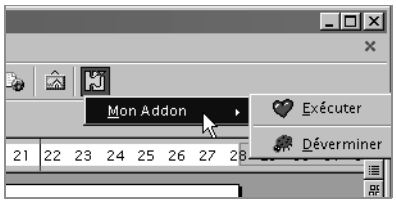
> **ATTENTION** **Lecture critique**
>
> De nombreuses macros sont disponibles sur Internet, mais elles ne doivent pas être considérées comme une référence absolue. Elles peuvent contenir des erreurs, ou être améliorées, ou correspondre aux premiers essais d'un programmeur ou à des extraits d'un codage plus complexe. Cherchez donc toujours à comprendre les principes utilisés.

Diffuser vos macros avec un add-on

Installer des macros sur un système et ajouter des icônes pour les déclencher est une tâche intimidante pour un utilisateur ordinaire. Pour un utilisateur confirmé, répéter ces opérations sur plusieurs ordinateurs devient lassant et sujet à erreurs. La méthode des *add-on* est prévue pour modifier la configuration d'une installation OpenOffice.org. En particulier, elle permet d'installer une bibliothèque de macros, d'ajouter des sous-menus ou des boutons avec icônes (à des endroits réservés pour cet usage) et même de prévoir des

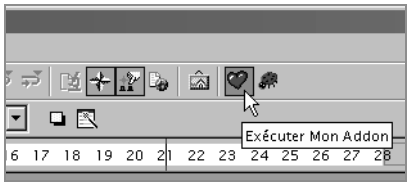
libellés en différentes langues. La figure C-1 montre un sous-menu créé pour accéder à l'*add-on*.

Figure C–1
Sous-menu pour un add-on



La figure C-2 montre deux boutons ajoutés sur la barre d'outils de la version 1.1 d'OpenOffice.org. Dans la version 2.0, chaque *add-on* est accessible dans une barre d'outil supplémentaire qui peut être juxtaposée aux autres.

Figure C–2
Nouveaux boutons pour un add-on



Ceci est expliqué dans le document « *HowTo* - Comment diffuser vos macros avec un *Add-on* ». Le document est lui-même un outil (réalisé par macros) qui sert à créer l'*add-on* sous forme d'un fichier ZIP.

- http://fr.openoffice.org/Documentation/How-to/indexht.html, section programmation Basic, document n°4.

Avec la version 1.1 d'OpenOffice.org, le fichier *add-on* doit être installé en exécutant le programme `pkgchk` à partir de la ligne de commande. La méthode à suivre est indiquée dans le document, mais certains utilisateurs ont quelques difficultés à le faire correctement, aussi d'autres développeurs ont-ils cherché une méthode plus simple pour diffuser un *add-on*. Parmi ceux-ci, Didier Lachièze a réalisé le « *HowTo* - Installeur de macros ». D'autres travaux sont en cours.

- http://fr.openoffice.org/Documentation/How-to/indexht.html, section programmation Basic, document n°6.

Avec la version 2.0 d'OpenOffice.org, l'installation d'un *add-on* est beaucoup plus simple, car on peut utiliser le Gestionnaire de packages, qui est accessible à partir du menu Outils. La figure C-3 montre un *add-on* déjà installé.

Contrairement à la version 1.1 d'OpenOffice.org, un add-on peut être ajouté dynamiquement à OpenOffice.org même si celui-ci est en cours d'execution : cliquer sur le bouton Ajouter et choisir le fichier `.zip` de l'add-on. Le gestionnaire de packages peut aussi être exécuté en ligne de commande par l'intermédiaire de la commande `unopkg`.

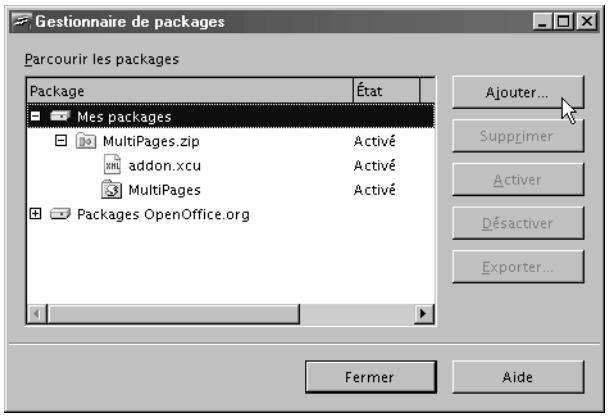


Figure C–3
Le Gestionnaire de packages

Introspection et documentation avec Xray

L'API offre des fonctions d'introspection et de core reflection permettant d'obtenir à l'exécution de nombreuses informations sur les objets manipulés. Toutefois, ces fonctions sont assez complexes à utiliser.

L'outil Xray, réalisé avec des macros, met en forme ces informations et permet d'étudier les sous-objets. Il est capable de retrouver dans l'IDL la documentation du sous-objet, si elle existe. Les auteurs utilisent intensivement Xray pour étudier la structure des objets et lire leur documentation dans l'API. Sans Xray, cet ouvrage ne serait pas aussi détaillé.

- http://fr.openoffice.org/Documentation/How-to/indexht.html
  section programmation Basic, document n°5.

Xray se présente sous la forme de deux bibliothèques de macros à installer dans le conteneur `soffice` (dans Mes Macros pour la version 2.0 d'OpenOffice.org). Pour étudier un objet, il est nécessaire d'ajouter une instruction dans le codage, pour appeler Xray :

```
monDocument = thisComponent
lesFeuilles = monDocument.Sheets
maFeuille = lesFeuilles.getByName("Janvier")
xray.xray maFeuille
```

L'expression signifie : appeler la macro `xray` qui se trouve dans la bibliothèque `xray` et lui transmettre en argument l'objet `maFeuille`. On obtient le panneau de la figure C-4, où on visualise les propriétés disponibles pour l'objet analysé. En positionnant le curseur sur la ligne d'un élément complexe comme `CellProtection`, il suffit de cliquer le bouton Xray pour afficher le contenu de ce sous-objet. L'opération peut être répétée sur différents sous-objets, à plusieurs niveaux.

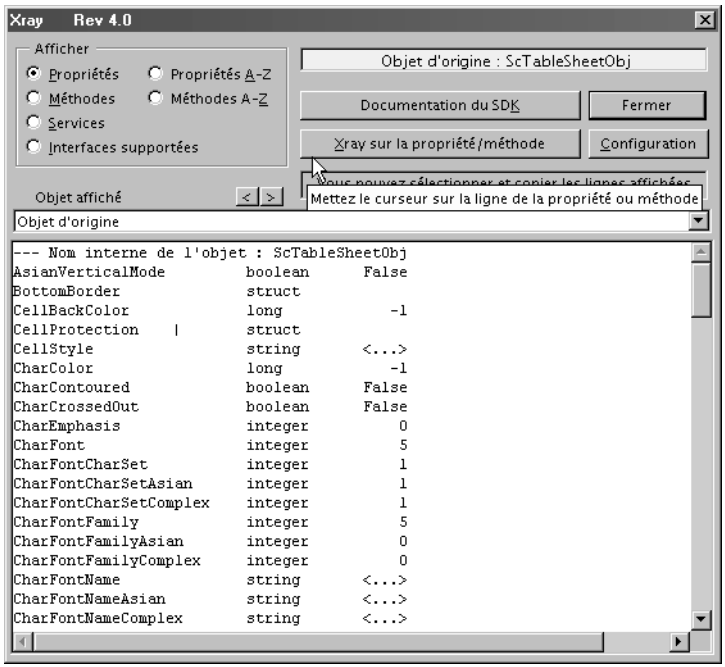


Figure C–4
Xray : propriétés d'une feuille Calc

En positionnant le curseur sur la ligne d'un élément, un simple clic sur le bouton `Documentation` permet de visualiser sur votre navigateur Internet la documentation API concernant l'objet sélectionné (voir figure C-5). Pour utiliser cette fonction, il faut toutefois avoir installé la documentation SDK sur l'ordinateur.

La figure C-6 montre les méthodes offertes par l'objet en analyse. On peut approfondir l'analyse sur une des méthodes à condition qu'elle ne comporte aucun argument ou encore accéder à sa documentation.

Xray liste les services proposés par l'objet et ceux disponibles par invocation, ou liste les interfaces prises en charge. Là encore, la documentation d'un service ou d'une interface est directement accessible.

Une version équivalente de Xray pour le langage Python est disponible. Naturellement appellée pyXray (voir figure C-7), elle a été réalisée par Laurent Godard et est disponible sur le site http://www.indesko.com/ . Cet outil est utilisable dès la version 1.1 d'OpenOffice.org.

Il faut une certaine expérience de l'API pour ne pas être perdu dans les informations fournies. Comme les objets API héritent pour la plupart des propriétés et méthodes d'un ou plusieurs autre(s) objet(s), et ainsi de suite, et que vous pouvez analyser un objet interne à un objet, vous retrouvez certaines méthodes et propriétés des « briques de base » d'OpenOffice.org, par exemple `setDelegator`, `getByName`, ou le tableau `ImplementationId`.

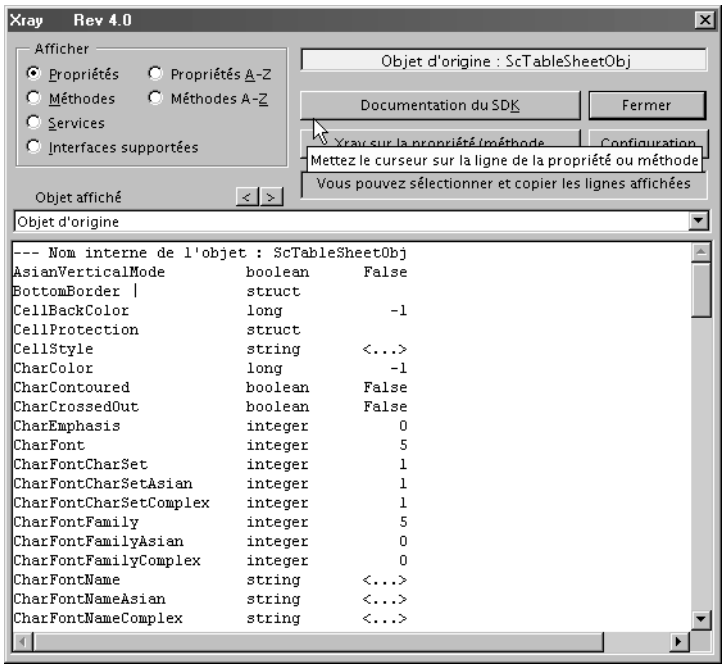


Figure C–5
Xray : voir la documentation d'un objet

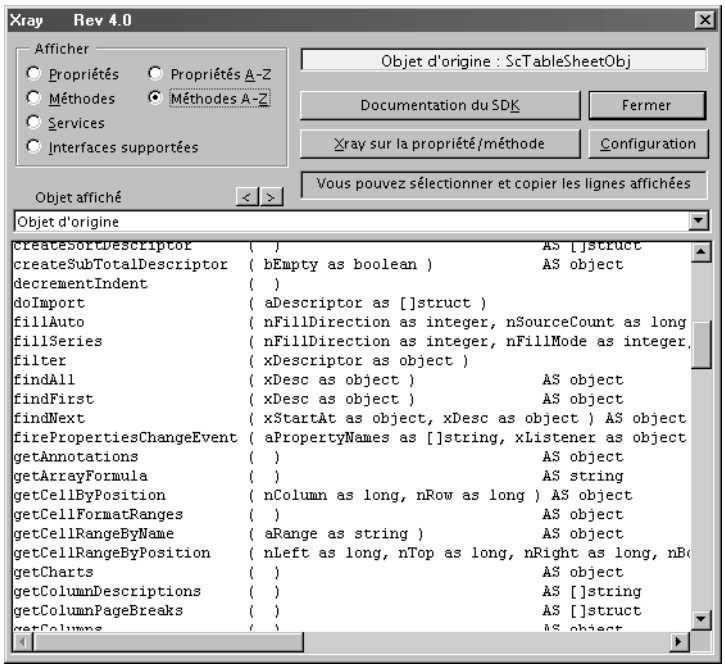


Figure C–6
Xray : méthodes d'une feuille Calc

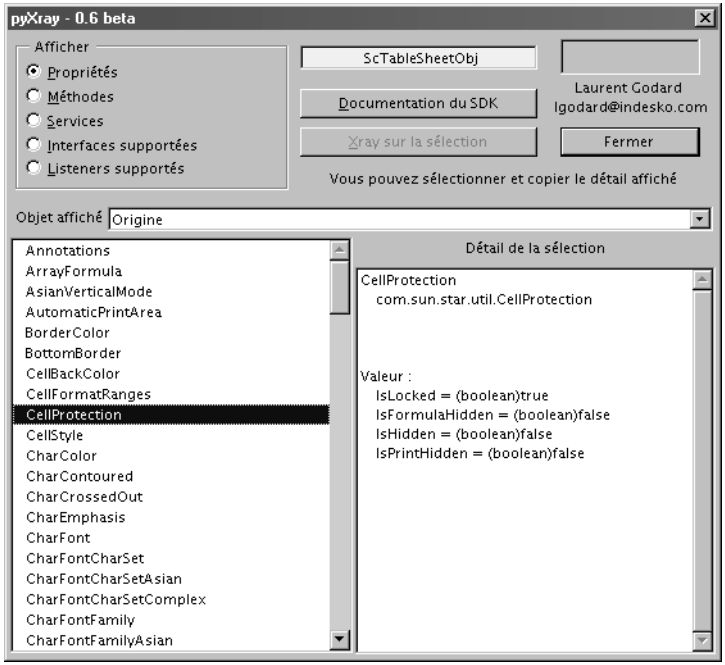


Figure C–7
Xray, version Python

Certaines propriétés d'objet sont listées mais non disponibles dans le contexte, d'autres renvoient un objet qui n'est autre que l'objet lui-même.

Lorsque vous demandez la documentation sur un élément d'un objet, la recherche peut échouer : la documentation manque, ou l'élément est un « fossile » des temps révolus, ou l'objet n'est pas officiellement utilisable.

Les bibliothèques de DannyB

DannyB est la signature de Danny Brewer. Il propose, sur la page Developers du site OOoMacros, des exemples et des routines réutilisables, dont :

- *Danny Brewer's Examples* (basé sur les réponses données dans OOoForum),
- *Danny's Draw Power Tools* (outils de macros pour Draw),
- *Danny's Library* (bibliothèque de l'auteur).

Il a publié aussi dans la section Code Snippets de OOoForum plusieurs bibliothèques personnelles, mises à jour au long de ses recherches ; notamment :

- Danny's Basic library
- Danny's Python Modules

- String utility functions for OOo Basic

▸ http://ooomacros.org/dev.php
Comme vous pouvez vous en douter, Danny Brewer écrit en anglais...

Les ressources que nous venons de passer en revue sont essentiellement destinées au développeur. Nous allons voir quelles sont celles mises à la disposition des utilisateurs.

Documents informatifs et exemples

La page des HowTo du site français d'OpenOffice.org contient une section intéressante sur la programmation Basic.

▸ http://fr.openoffice.org/Documentation/How-to/indexht.html

Une partie de ces « HowTo » fait double emploi avec notre ouvrage ou sont cités ailleurs, mais nous signalerons ceux apportant des informations complémentaires.

- « Dans la jungle de l'API » : une expérience vécue d'un essai de compréhension de l'API, racontée sur le mode humoristique.
- « Exemples pour BDD » : divers conseils de programmation Basic sur les bases de données et formulaires.
- « Manuel de programmation Basic StarOffice 7 » : ce document écrit par Sun Microsystems vous fournit une autre vision de la programmation de StarOffice, un dérivé d'OpenOffice.org. Vous y trouverez une description succincte, mais couvrant l'ensemble OOoBasic et API, avec quelques informations non décrites ici et des conseils pour un programmeur Visual Basic for Application.

La page Macros du site français d'OpenOffice.org met à disposition un grand nombre d'exemples de macros sélectionnés pour leur valeur explicative.

▸ http://fr.openoffice.org/Documentation/Macros/indexmac.html

Le document « Éléments de programmation des macros dans OOo » est une traduction française d'un document anglais écrit par Andrew Pitonyak. Il contient un grand nombre de « recettes » sur des aspects assez spécialisés de la programmation OpenOffice.org. La traduction étant assez en retard sur le document original, consultez plutôt la version anglaise si vous maîtrisez la langue de Mark Twain.

▸ Français : http://fr.openoffice.org/Documentation/Guides/Indexguide.html
Anglais : http://www.pitonyak.org/oo.php

Le site de langue anglaise OOoMacros contient de nombreuses macros, la plupart assez, voire très élaborées. On y trouve notamment des macros écrites par Danny Brewer, qui sont des modèles de bonne programmation. Vous y retrouverez aussi certaines macros françaises publiées en version anglaise.

- http://ooomacros.org/

Le site de langue anglaise OOoForum consacre la section Code Snippets à des exemples de codage, la plupart en OOoBasic. Ces codages ont été choisis pour leur intérêt pédagogique et technique. Danny Brewer est un des modérateurs de OOoForum.

- http://www.oooforum.org/forum/viewforum.php?f=11

Le projet API du site anglais OpenOffice.org, n'est pas en reste et publie les *Snippets* que lui fournissent les programmeurs API.

- http://codesnippets.services.openoffice.org/

Le projet DBA d'OpenOffice.org consacré aux bases de données (en anglais) offre un fichier ZIP contenant quelques macros d'accès aux bases de données. Décompressez ce ZIP et utilisez Outils > Macros > Macro pour insérer une bibliothèque, qui aura pour nom `DBATools`.

- http://dba.openoffice.org/downloads/DBATools.zip

Votre application OpenOffice.org elle-même est livrée avec plusieurs bibliothèques de macros dans `soffice`. Ces codages sont malheureusement très peu documentés et souvent anciens. On peut y glaner des séquences intéressantes, et même utiliser des macros telles quelles. Voyez notamment les bibliothèques `ImportWizard`, `Gimmicks` et `Tools`.

D'une manière plus générale, les solutions à base de macros citées dans le chapitre 1 du présent ouvrage peuvent aussi apporter des solutions à vos besoins.

Sites Web et forums

Parmi les sources d'information, nous citerons des forums en langue française, anglaise ou autres. Rappelons quelques principes communs à tous les forums :

- Commencez par observer les échanges de messages avant de participer activement ; ceci afin de cerner le domaine des discussions et le niveau technique.
- Ne posez pas une question qui a déjà reçu une réponse récemment ; il existe des moyens de recherche dans chacun des forums.

- Restez poli, évitez le bavardage intempestif, soyez bref et clair dans votre demande.
- Il est très mal élevé de poser dans un forum une question dans une autre langue que celle pour laquelle il est créé.
- Rappelez-vous que ces forums sont animés par des bonnes volontés, qui ne sont pas rétribuées pour cela.

À RETENIR S'inscrire à un forum

Les forums cités sont pour la plupart des listes de distribution de messages (*mailing lists*). Pour savoir comment s'y abonner (gratuitement), envoyer un message, se désabonner, il suffit d'envoyer un courrier électronique vide à l'adresse xxx-info@yyyy, par exemple prog-info@fr.openoffice.org. Un robot vous renverra un message explicatif, en anglais. Comme c'est un robot, inutile d'être poli avec lui, et inutile d'espérer une réponse personnalisée.

En français

Le site français d'OpenOffice.org, http://fr.openoffice.org/ est probablement le plus vivant et le plus complet de tous les sites nationaux d'OpenOffice.org. Nous en avons cité plusieurs pages. Ce site est géré par Sophie Gautier, qui est aussi la modératrice des forums en langue française. Les questions (et réponses) concernant OOoBasic et l'API ont leur place sur la liste prog@fr.openoffice.org. Il arrive que certains messages soient signés de l'un ou l'autre des auteurs de ce livre.

- http://fr.openoffice.org/servlets/ProjectMailingListList

Le site OOoConv, créé par un des auteurs, présente plusieurs outils dont l'analyse sera instructive : OOoConv, FitOO, BatchConv.

- http://oooconv.free.fr/

En anglais

Le site http://www.oooforum.org/ a une vitalité semblable aux forums français d'OpenOffice.org, mais il n'est accessible qu'avec un navigateur Web. On y trouve en particulier :

- le forum Macros and API, supervisé par Danny Brewer, où vous trouverez de nombreuses informations dans les réponses déjà faites ;
- le pseudo-forum Code Snippets, déjà cité, qui n'est pas un vrai forum de discussion mais un répertoire d'exemples de codage.

La page http://www.openoffice.org/mail_list.html vous donne accès à tous les forums OpenOffice.org de langue anglaise. Le forum dev@api.openoffice.org est consacré aux questions sur l'API elle-même, posées par des programmeurs expérimentés. Les réponses sont souvent

apportées par les chefs développeurs d'OpenOffice.org dans la mesure de leur disponibilité. Abstenez-vous d'y poser des questions de débutant.

Pour les programmeurs aguerris, le site du projet UDK d'OpenOffice.org http://udk.openoffice.org/ est consacré aux développements permettant de programmer OpenOffice.org dans différents langages, dont OOoBasic. Il fournit différents liens et des pages explicatives. Ce projet possède un forum dev@udk.openoffice.org. Il traite de l'interfaçage de langages de programmation (OOoBasic, Python, Java, etc) avec l'API OpenOffice.org, et de OOoBasic.

Autres langues

Le site principal d'OpenOffice.org possède de plus en plus de subdivisions nationales, accessibles depuis la page principale (voir la liste déroulante Native Language Projects). Il faut reconnaître qu'elles n'offrent que peu d'informations nouvelles sur la programmation d'OpenOffice.org.

- http://www.openoffice.org/

Le site privé allemand Kienlein offre diverses informations de programmation d'OpenOffice.org pour accéder à des bases de données.

- http://kienlein.com/pages/oo.html

IssueZilla

IssueZilla, appelé aussi en abrégé IZ, est une base de données pour gérer les rapports d'anomalies et de demandes d'amélioration. Également connue depuis peu sous le nom de `IssueTracker`, IT en abrégé, elle est consultable sur Internet. Les rapports IssueZilla sont rédigés en anglais, afin d'être compréhensibles par des lecteurs du monde entier. Il est donc nécessaire d'avoir une connaissance de cette langue pour utiliser cette source d'information.

Rechercher un rapport dans IssueZilla

La page d'entrée pour rechercher quoi que ce soit dans IssueZilla est :

- http://www.openoffice.org/project/www/issues/query.cgi

Chaque rapport IssueZilla reçoit un numéro. Pour afficher un rapport dont on connaît le numéro, il suffit de remplir le champ en haut de la page et de cliquer sur le bouton Jump to Issue.

Si vous cherchez s'il existe un rapport sur un sujet donné, il faut remplir certains champs proposés par le formulaire. Plus vous remplissez de champs, plus la recherche se focalise. La difficulté est de se demander quels mots ont pu être utilisés dans les rapports déjà écrits. Pour éviter des recherches infructueuses, il est préférable de commencer par une recherche assez générale, un mot ou deux dans le titre du rapport, et de restreindre la recherche si elle renvoie un grand nombre de rapports. Souvent il faut essayer des mots synonymes, ou écrits différemment.

Rédiger un rapport

De nombreux rapports sont écrits chaque jour et les développeurs ont du mal à les analyser. Vous ne devriez en écrire que sur un sujet que vous estimez important et faire une recherche préalable pour éviter de dupliquer un rapport existant.

> On peut écrire un rapport pour seulement deux raisons : demander une amélioration ou signaler une anomalie. IssueZilla n'est pas un site pour demander une aide, voyez les forums pour cela.

Les demandes d'améliorations sont lues par les développeurs et classés avec un horizon de prise en compte éventuelle (target milestone). Ceci ne veut pas dire qu'elles seront introduites, car d'autres critères entrent en jeu (ressources, intérêt marketing, complexité).

Avant d'écrire un rapport d'anomalie, vérifiez très soigneusement qu'il s'agit bien d'une erreur de l'application OpenOffice.org et non pas d'une erreur de compréhension de votre part, ou une mauvaise configuration de votre ordinateur. Simplifiez au maximum les conditions d'apparition de l'anomalie. Joignez si possible un document démontrant l'erreur, par exemple une macro réduite au codage minimum nécessaire. En effectuant ce travail, vous analyserez mieux l'anomalie et souvent vous verrez qu'elle provient d'une erreur de votre part. Soyez le plus clair possible dans les explications, restez factuel. Pensez que les développeurs ont à lire des dizaines de rapports, en plus de leur activité habituelle.

L'écriture de rapports est réservée aux membres d'OpenOffice.org. Ce n'est pas une société secrète, n'importe qui peut devenir membre, gratuitement et sans engagement. Allez à la page http://www.openoffice.org/servlets/Join et remplissez le formulaire. Vous avez maintenant une identité dans le système et un mot de passe.

Pour créer un rapport, vous devez d'abord vous connecter au système, en allant à la page http://www.openoffice.org/servlets/TLogin. Puis, commencez à la page http://www.openoffice.org/issues/enter_bug.cgi. Une fois choisi le domaine principal, vous vous retrouvez devant un formulaire dont il faut remplir au mieux les cases. Ce n'est pas facile les premières fois. Le titre du rapport et le texte explicatif doivent obligatoirement être en anglais. Envoyez le rapport. Un numéro sera attribué automatiquement et vous recevrez un courrier électronique pour chaque évolution de ce rapport. Utilisez la page Web renvoyée pour ajouter éven-

tuellement un fichier. Plus tard, en réaffichant le rapport, vous pourrez ajouter de nouvelles informations, à condition de vous être identifié. Soyez très patient, un rapport peut rester plusieurs mois sans réponse...

Si votre anomalie est reconnue comme telle, la date prévisionnelle de correction sera indiquée dans le champ Target Milestone. Les ressources humaines étant toujours limitées, il est encore possible que cette date soit repoussée à plus tard lors d'une ré-analyse des priorités.

Partager la connaissance

Comme nous l'avons vu dans l'exposé précédent, la communauté est très active en ce qui concerne l'API et les macros. De nombreuses sources sont disponibles et il ne tient qu'à nous, acteurs et utilisateurs, d'enrichir notre connaissance commune.

Vous venez d'écrire une macro qui comble un manque d'OpenOffice.org ou simplement se révèle utile au jour le jour ; il est fort probable que cette macro puisse servir à quelqu'un d'autre. Comme nous apprenons tous de la lecture des macros publiées par d'autres, vous pouvez vous aussi participer en rendant public votre travail.

Vous serez accueillis avec enthousiasme sur les listes de diffusion francophones comme doc@fr.openoffice.org ou prog@fr.openoffice.org. Votre contribution y sera valorisée et ce, en échangeant juste quelques courriers électroniques. Pourquoi s'en priver ?

Conclusion

La richesse des ressources disponibles sur l'Internet, source d'informations dynamique et vivante, est illimitée et nous n'en présentons qu'une partie. N'hésitez pas à en tirer régulièrement profit.