# Programmation OpenOffice.org et LibreOffice

## Macros OOoBASIC et API

**Laurent Godard**

**Bernard Marcelly**

Préface de l'Aful


 ISBN : 978-2-212-13247-2

EYROLLES

# Préface

Depuis le début des années 2000, le monde de l'informatique assiste à une révolution attendue depuis des décennies par les professionnels : la standardisation.

Certes les premiers standards ont pris naissance dès les années 1960, sous l'impulsion des industriels tels IBM ou Bull mais, avec l'apparition massive des logiciels propriétaires dans les années 1980, l'informatique est devenue peu à peu une science exacte non écrite.

En effet, les éditeurs, au prétexte de protéger leurs investissements, ont pris l'habitude de livrer à leurs clients des suites logicielles d'une grande richesse fonctionnelle, mais sans fournir la moindre documentation sur le fonctionnement interne des composants, ni sur la manière dont ils communiquent entre eux.

Or en matière de communication, les experts savent parfaitement que l'absence de standards conduit inexorablement à l'émergence de monopoles. Ce qui devait arriver arriva, et la fin des années 1980 a vu se profiler la domination d'un système d'exploitation, d'une suite bureautique, d'un outil de CAO et de bien d'autres logiciels dans des domaines plus spécialisés.

Heureusement, quelques années plus tard, l'explosion d'Internet, une technologie construite entièrement sur des standards ouverts, a fait comprendre qu'il était possible d'imaginer des modèles économiques particulièrement lucratifs basés sur le partage de la connaissance.

C'est ainsi que des organisations telles que l'IETF, le W3C, OASIS et bien d'autres ont collaboré avec les principaux acteurs du marché pour spécifier des standards de communication et des formats d'échange. Beaucoup d'entre eux sont devenus des normes ISO et permettent tous les jours à de nouveaux arrivants de se positionner sur le marché, d'apporter un sang neuf et d'innover.

Parmi ces standards, ODF occupe une place de choix. Validé par OASIS en 2005 puis adopté comme norme par l'ISO en 2006, il s'agit du seul standard bureautique

qui soit totalement spécifié et effectivement adopté nativement par des éditeurs. Grâce à ce standard, une dizaine de suites bureautiques concurrentes, libres ou propriétaires, sont totalement compatibles entre elles. La plus connue d'entre elles, bien sûr, est OpenOffice.org.

Malheureusement l'existence d'un standard, si elle est nécessaire pour lutter contre un monopole, est loin d'être suffisante. Les alternatives ont surtout besoin d'être attractives pour donner envie aux utilisateurs de changer leurs habitudes. OpenOffice.org apporte aux utilisateurs l'ensemble des fonctionnalités qu'ils ont l'habitude de trouver dans une suite bureautique. Mais en tant que challenger, il faut proposer plus. Beaucoup plus. Il faut étendre ses fonctionnalités. C'est ici que ce livre joue un rôle essentiel.

Les API sont précisément un moyen d'étendre, à différents niveaux, les capacités d'un logiciel. Au fil de cet ouvrage, le lecteur se familiarisera avec les mécanismes qui permettent de prendre le contrôle du logiciel. Non seulement il deviendra possible de manipuler les documents par d'autres moyens que ceux proposés par la classique interface utilisateur, mais on pourra également effectuer des opérations impossibles à mettre en œuvre autrement que par programmation.

Pour s'en convaincre il suffit de consulter la liste déjà bien longue des extensions disponibles pour OpenOffice.org qui comprend par exemple une interface pour convertir un fichier PDF en feuilles de dessin, un générateur de rapports, un colorieur syntaxique, un générateur de formulaires, de codes barres, des fonctionnalités de conception assistée, et des centaines d'autres. Certes, nombre de ces extensions dépassent un peu les objectifs de cet ouvrage, mais il y a fort à parier que bon nombre de lecteurs, enthousiasmés par le potentiel et par la grande cohérence des API d'OpenOffice.org et motivés par le succès de leurs premières expérimentations avec l'interface Basic, auront envie de s'aventurer plus loin. Ils pourront alors apprendre à naviguer dans la documentation en ligne qui détaille l'interface Java, très similaire bien que moins abordable, bien sûr, mais ouvrant sur un univers qui ne sera limité que par l'imagination.

Mais il ne faudrait pas s'arrêter en si bon chemin. L'adoption d'OpenOffice.org par le plus grand nombre d'utilisateurs passe par la disponibilité d'un maximum d'extensions, et surtout par leur pertinence par rapport à des besoins réels. Aussi le développeur doit toujours garder à l'esprit que si son travail est utile pour lui ou pour ses collègues, il l'est certainement pour beaucoup d'autres. Et c'est là le principal succès que l'Association Francophone des Utilisateurs de Logiciels libres (AFUL) souhaite à cet ouvrage : parvenir à créer parmi ses lecteurs une dynamique qui leur donne vraiment envie de partager leur travail. Du code, en Basic ou dans n'importe quel autre langage, est avant tout de la connaissance formalisée, qu'il est toujours bon de diffuser.

Selon cette logique, le raisonnement est simple : plus ce livre aura de lecteurs, plus on pourra espérer voir remonter de nouvelles fonctionnalités pour OpenOffice.org sous

forme d'extensions, plus la suite deviendra attrayante, et plus on trouvera de documents au format ODF. La victoire de ce standard s'en trouvera alors largement facilitée.

Cette cohérence entre l'utilisation des standards et le partage de la connaissance est à la base des actions de l'AFUL qui défend depuis plus de douze ans l'utilisation des logiciels libres et l'adoption de standards ouverts.

OpenOffice.org est un logiciel libre qui, grâce au potentiel développé par l'existence de ses API, est susceptible de gagner là où toute l'industrie a finalement renoncé, c'est-à-dire redonner à l'utilisateur les trois atouts qui font la devise de l'AFUL : la Liberté de choisir sa suite bureautique, la Pérennité des documents grâce aux standards ouverts et l'Interopérabilité entre les suites concurrentes qui respectent la norme ODF.

En conclusion, il s'avère que dans la société de l'information, dans laquelle le XXI^e^ siècle nous a fait entrer, la maîtrise des formats et logiciels, canalisateurs de notre savoir et de sa diffusion n'est pas seulement une affaire de professionnels. L'Histoire n'est pas encore écrite et chacun peut devenir acteur, comme le montre d'ailleurs la naissance récente de LibreOffice, soutenue d'abord par une communauté d'utilisateurs et de développeurs, soucieux de préserver sur le long terme les fondamentaux d'une liberté égalitaire et fraternelle.

*Association Francophone des Utilisateurs de Logiciels libres (AFUL),*
*par la plume de Philippe Allart (trésorier)*
*et les lumières de Véronique Fritière et Jean Peyratout*

# Table des matières

CHAPITRE 5

## Les instructions de traitement ..... 163

CHAPITRE 13

## Les formulaires ..... 757

# Avant-propos

Rappelons en guise de préambule qu'OpenOffice.org, suite bureautique libre et gratuite, est constituée des modules habituels de traitement de texte, de tableur, de présentation, ainsi que de modules de dessin et d'édition de formules mathématiques. Tournant aussi bien sous Windows et Linux que sous Mac OS X, elle peut être utilisée en lieu et place de la suite Microsoft Office.

> **À NOTER** **OpenOffice.org et ses dérivés**
>
> OpenOffice.org est une base à partir de laquelle diverses variantes ont été dérivées. Sun MicroSystems avait commercialisé StarOffice. Depuis, Sun a été absorbé par Oracle, qui commercialise Oracle Open Office, version non libre dont le nom crée une ambigüité... Parmi les versions libres et gratuites, certaines sont adaptées à des distributions Linux, et NeoOffice est plus orienté Mac. Plus récemment, LibreOffice, qui se veut plus indépendant et plus évolutif, est porté sur tous les systèmes d'exploitation. Soulignons que LibreOffice est loin de constituer un simple produit dérivé. Bien au contraire, il s'agit là d'une séparation majeure sous l'égide de l'OpenDocument Foundation, et nombreux sont les contributeurs qui prennent position pour cette nouvelle initiative. Nous vous recommandons donc de suivre son évolution.
> Tous ces produits dérivés d'OpenOffice.org témoignent d'une dynamique d'évolution fonctionnelle tout en assurant la pérennité des documents bureautiques produits manuellement ou par programmation.
> Le contenu de ce livre s'applique généralement à toutes ces variantes et nous en resterons à la dénomination générique OpenOffice.org.

## À qui s'adresse ce livre ?

Vous êtes un utilisateur de la suite OpenOffice.org 2 ou 3 (ou un dérivé comme LibreOffice) et vous connaissez bien ses nombreuses possibilités. Cependant, dans certains cas, vous souhaitez simplifier des manipulations répétitives. Le langage de macros Basic d'OpenOffice.org, intégré dans la suite, peut répondre à votre besoin. Il est conçu pour être simple d'emploi, tout en étant puissant.

Avec ce livre, vous apprendrez à programmer la suite OpenOffice.org, par exemple pour ajouter une fonctionnalité personnelle déclenchée par un bouton sur une barre d'outils ou par un raccourci. Vous pourrez même automatiser des traitements, par exemple effectuer des modifications dans toute une série de documents Writer, ou lire une base de données pour en extraire des informations et les insérer dans un document Writer.

Ce livre est-il accessible à un débutant en programmation ? Les connaissances de base de la programmation sont exposées dans les premiers chapitres. Nous avons aussi inclus des conseils et des bonnes pratiques, qui vous éviteront bien des déboires. À chaque étape, nous avons choisi des exemples volontairement simples, chacun focalisé sur le point à expliquer. Évidemment, le chemin sera plus ardu et plus long si vous en êtes à vos tout débuts : avancez très progressivement, en écrivant de nombreux petits programmes pour vous approprier les concepts de base. Ne craignez pas les erreurs, elles sont source de connaissances. Progressivement, vos programmes s'enrichiront et deviendront toujours plus utiles, à votre grande satisfaction.

Si vous avez l'expérience de la programmation, la première partie vous semblera facile. Méfiez-vous cependant des analogies avec d'autres langages, car chacun a ses particularités. Si vous connaissez la programmation orientée objet, vous comprendrez plus facilement les principes de l'API OpenOffice.org, qui sera utilisée à partir de la troisième partie ; mais ce n'est pas indispensable.

Si vous êtes dans un service informatique chargé d'automatiser des processus utilisant la suite OpenOffice.org ou StarOffice, ou de migrer des applications écrites pour la suite MS-Office, cet ouvrage vous économisera de très nombreuses heures de recherche et de tâtonnements et accélérera la phase d'apprentissage. Comme la mémoire humaine a ses limites, vous souhaiterez garder ce livre à portée de main. Il est cependant probable que vous rencontrerez des besoins non décrits ici, mais la base de connaissances acquises vous facilitera l'étude de l'API OpenOffice.org. Même si votre projet n'emploie pas Basic, celui-ci peut vous aider à trouver plus rapidement comment utiliser telle ou telle fonctionnalité et en déduire le codage équivalent dans votre langage.

**Précaution**

Pour tirer parti de ce livre, il est recommandé de bien connaître les possibilités qu'OpenOffice.org offre au niveau de son interface utilisateur, dans les domaines sur lesquels vous souhaitez intervenir par macro. En effet, autant éviter un développement si quelques manipulations résolvent votre problème ou le simplifient. Savez-vous bien utiliser les styles de paragraphe, de caractère, de page ? les modèles de document ? la recherche générique ? le copier-coller avec ou sans formatage ? l'utilisation des bases de données ? Savez-vous ce qu'est un signet ? N'hésitez pas à lire ces excellents livres donnant toutes les astuces pour être productif sous OpenOffice.org.

- 🕮 *OpenOffice.org 2.2 efficace* de Sophie Gautier, Christian Hardy, Frédéric Labbé, Michel Pinquier, éditions Eyrolles 2007.
- 🕮 *OpenOffice.org 3.2 efficace* de Sophie Gautier, Gilles Bignebat, Christian Hardy et Michel Pinquier, éditions Eyrolles 2010.

# Contenu de l'ouvrage

Vous trouverez une description complète et précise du Basic OpenOffice.org, rédigée pour être compréhensible pour un débutant en programmation, tout en apportant des informations indispensables au programmeur expérimenté.

Vous apprendrez comment utiliser facilement l'interface de programmation d'application (API) pour lire, écrire, modifier les documents OpenOffice.org, accéder aux bases de données, et dialoguer avec l'utilisateur. L'API d'OpenOffice.org est extrêmement riche et parfois complexe. Il nous a fallu privilégier les sujets les plus courants et les vérifier chacun par des macros. Si vous ne trouvez pas la réponse à une question dans cet ouvrage, c'est peut-être qu'il s'agit d'un cas rare et particulièrement difficile à réaliser.

L'API est une interface indépendante du langage de programmation. À cet égard, les descriptions des fonctionnalités sont valides pour divers environnements : les langages de script intégrés à OpenOffice.org (Basic, Python, JavaScript, BeanShell, cités au chapitre 1), le pilotage par un langage externe via COM (abordé au chapitre 14), voire même dans des composants développés en Java ou C++. Un développeur d'applications pourra facilement transposer dans un autre langage les exemples Basic donnés dans cet ouvrage. L'annexe A décrit les principes de l'API, son utilisation par le Basic OpenOffice.org, et comment aller plus loin avec Xray et la documentation de l'API et les outils d'introspection (Xray et similaires).

Notre souci a été d'être clair et progressif, sans trop entrer dans des considérations théoriques. Nous avons pour cela créé des centaines d'exemples de macros Basic, avec des documents spécialement configurés pour chaque essai. Tous les exemples ont été testés, et pour la plupart retestés, sur les versions récentes. Ils sont mis à votre disposition en téléchargement libre sur le site www.editions-eyrolles.com – cela vous épargnera l'effort de frappe et les erreurs de ressaisie. Ces exemples peuvent servir de modèles pour créer rapidement de nouvelles macros, en quelques copier-coller.

Prenez votre temps en lisant les explications et les exemples : chaque phrase est importante. Nulle prétention littéraire pour ce texte technique. N'hésitez pas à relire des passages que vous pensez connaître.

La source de documentation étant presque exclusivement en anglais, nous avons choisi d'aider le lecteur peu familier de cette langue en traduisant les termes importants et en utilisant des noms de variables en français. À l'usage, le fait d'employer des noms français facilite beaucoup l'assimilation, même si on lit couramment l'anglo-américain.

Ce livre n'est pas une simple traduction de documents anglais, ni une collection d'astuces récoltées sur les forums. Il est une synthèse de connaissances et présente de

façon directement utile beaucoup d'informations peu connues, mal documentées ou non documentées. Nous signalons notamment des anomalies de fonctionnement, des limitations, ou des erreurs de documentation afin de vous éviter les difficultés que nous avons rencontrées.

Nous mettons aussi à votre disposition en téléchargement un grand nombre de routines utilitaires pour simplifier de nombreux codages. Leur liste est donnée à l'annexe B.

L'interface utilisateur des versions successives présente inévitablement de petites modifications par rapport à notre version de travail. De plus, la forme des icônes peut varier d'une distribution à l'autre, selon le système d'exploitation, ou la configuration choisie par l'utilisateur. Cela ne devrait pas vous empêcher de retrouver l'équivalent de ce qui est imprimé dans ce livre.

La **première partie** montre l'utilité de la programmation OpenOffice.org, et présente les langages de script et les diverses manières de déclencher un script. Vous verrez comment utiliser l'enregistreur de macro, et pourquoi il est finalement assez limité. Contrairement à la concurrence, OpenOffice.org supporte plusieurs langages de script ; nous en faisons une comparaison. Le but de la programmation OpenOffice.org est souvent d'ajouter des fonctionnalités. Les extensions sont un moyen de les diffuser facilement, nous en montrerons les possibilités ainsi que les outils pour les produire.

Dans la **deuxième partie**, nous décrivons le langage OOoBasic. Vous faites connaissance avec l'environnement de développement intégré, et vous l'utilisez pour écrire et exécuter votre première macro. Vous apprenez les diverses manières d'exécuter une macro. Même si vous connaissez déjà un Basic (par exemple Visual Basic™ qui lui est proche), parcourez les chapitres de cette partie. En cas de problème d'exécution, relisez-la, elle contient bien des détails importants et souvent non décrits dans la documentation officielle.

À partir de la **troisième partie**, vous apprenez à écrire ou modifier des documents OpenOffice.org : Writer, Calc, Draw, etc. Vous aurez besoin d'utiliser l'API, cœur de la programmation OpenOffice.org dans quelque langage que ce soit. Mais nous éviterons les exposés théoriques rébarbatifs pour nous concentrer sur les solutions à des besoins réels. OpenOffice.org réutilise des concepts généraux dans chaque type de documents, mais avec des variations propres à chacun. Nous avons regroupé les principes communs dans le chapitre « Les documents OpenOffice.org », puis décrit les aspects spécifiques aux documents Writer, Calc et Draw/Impress dans les chapitres suivants. Dans chacun de ces chapitres, il n'est nul besoin d'effectuer une lecture complète : après avoir acquis les notions de base, utilisez ensuite le livre comme une référence, et n'approfondissez que les sujets qui vous sont utiles.

La **quatrième partie** va au-delà des manipulations de documents pour vous permettre de construire des applications élaborées. Vous y apprenez à afficher des dialogues tout à fait semblables à ceux des applications classiques, à utiliser une base de données et des formulaires élaborés. Des aspects plus transversaux sont ensuite traités : gestion de la configuration, gestion de fichiers depuis l'API, utilisation du Dispatcher, interaction avec le monde MS-Windows, et diverses méthodes bien utiles.

Les **annexes** présentent de nombreuses informations complémentaires. Nous expliquons ce qu'est l'API et comment en obtenir des informations pour aller encore plus loin. Nous présentons ensuite une liste de routines utilitaires dont certaines ont été utilisées pour simplifier nos exemples. Nous signalons enfin les ressources Internet incontournables pour qui souhaite se tenir à jour : il s'agit de forums où chercher assistance, ou de sites fournissant des exemples de macros, des documents explicatifs et des outils.



# Changements par rapport à la précédente édition

Avec cette nouvelle édition, totalement revue et réorganisée, nous avons ajouté des notions qui n'avaient pas été décrites, tenu compte des modifications et ajouts apportés par les versions successives d'OpenOffice.org 2 et 3, amélioré de nombreux codages, mis à jour les références d'adresses Internet, et signalé des outils apparus depuis la version précédente de notre livre.

Pour alléger la lecture, nous avons supprimé ce qui était spécifique de l'ancienne version 1 d'OpenOffice.org, et les limitations propres à la version initiale 2.0. Sauf indication contraire dans le texte, ce qui est décrit est aussi valable pour les dernières versions 2 (versions 2.3.1 et plus récentes). Les copies d'écran sont faites avec la version 3.

### Principaux ajouts

Chapitre 1 : les extensions.

Chapitre 2 : information sur la compatibilité VBA.

Chapitre 3 : type `Byte`, sous-type `Decimal`, type `Collection`, type défini par l'utilisateur.

Chapitre 4 : boucle `For Each`.

Chapitre 5 : opérateur `Like`, fonctions `DateDiff`, `DateAdd`, `DatePart`, `Format`.

Chapitre 7 : description complète des options d'export PDF, valeurs d'encodage de caractères pour import et export CSV et texte, propriétés de document définies par l'utilisateur.

Chapitre 8 : surlignement, statistiques du document, tableaux irréguliers en version 3, actualiser un document Writer.

Chapitre 9 : fusion de cellules, filtrage, fonction matricielle, liens vers un autre classeur, exemples de *Listener*.

Chapitre 10 : méthodes et propriétés pour gérer un diaporama en cours.

Chapitre 11 : ajout dynamique des contrôles de dialogue, dialogues et messages multilingues.

Chapitre 12 : valeurs d'encodage de caractères pour une base plate, connexion à une base non enregistrée, publipostage plus détaillé.

Chapitre 13 : description systématique des contrôles de formulaire, ajout dynamique des contrôles de formulaire, macros dans un document Base, ouvrir et fermer un formulaire depuis un autre formulaire de fichier Base.

Chapitre 14 : lire et modifier la configuration OpenOffice.org, écrire et lire un fichier texte encodé, formatage par l'API, exemples d'utilisation du Dispatcher.

## Documents disponibles en téléchargement

Sur le site web des éditions Eyrolles, www.editions-eyrolles.com, vous trouverez la fiche de ce livre, où vous pourrez télécharger gratuitement un fichier Zip. Ce fichier se décompacte dans un répertoire `MacrosLivre` comprenant autant de sous-répertoires que de chapitres donnant des exemples de macros. Les macros d'un chapitre se trouvent dans des documents OpenOffice.org, ainsi que les fichiers associés éventuels. La référence du fichier contenant la macro est indiquée en première ligne de chaque macro reproduite dans cet ouvrage.

# 1

# Les scripts dans OpenOffice.org

Le terme macro évoque plutôt le langage Basic, qui sera d'ailleurs notre principal langage de programmation dans ce livre. Mais il n'est pas le seul, comme nous allons le voir. On devrait maintenant employer le terme plus général de script, mais les habitudes ont la vie dure, et le terme macro est employé partout dans les interfaces utilisateur.

OpenOffice.org offre différents langages de programmation (langages de script), contrairement à la concurrence. Sa structure permet même d'en rajouter d'autres. Nous serons amenés à signaler quelques concepts avancés, qui seront plus clairs après lecture du reste du livre.

## De l'automatisation d'OOo à l'application d'entreprise

Avant de vous lancer dans l'aventure, vous vous demandez peut-être ce qu'on peut bien réaliser d'intéressant avec OOoBasic et l'API d'OpenOffice.org. Eh bien, tout est fonction du besoin. Une « bonne » macro est une macro qui satisfait un besoin, qu'il soit récurrent ou ponctuel. Il n'est pas nécessaire de bâtir un environnement applicatif complet (même si cela est tout à fait possible) et quelques lignes suffisent parfois à rendre des services inestimables au quotidien.

Les macros d'OpenOffice.org permettent d'adapter le logiciel à un besoin spécifique, avec cet avantage indéniable que dans le cas des macros OOoBasic, tout est déjà intégré et prêt à l'utilisation. OOoBasic offre un cadre d'exécution commun pour élaborer des additifs logiciels. Une macro « arrivant » sur un poste est certaine de retrouver ce cadre de travail, et ce quelle que soit la plate-forme utilisée.

## Des macros pour les utilisateurs d'OpenOffice.org

Les utilisations des macros sont multiples. On peut bien sûr intervenir directement sur un document en cours pour reproduire une tâche répétitive ou fastidieuse, mais aussi fédérer plusieurs documents pour des traitements transversaux. Bien des utilisateurs ont leurs propres macros, non publiées, qui leur font gagner du temps dans leurs activités quotidiennes, depuis l'application d'un style de caractère en cliquant sur une simple icône jusqu'à la mise en forme de plusieurs documents à la fois.

Une macro d'intérêt général peut être distribuée sous la forme d'une extension. Une extension est un fichier reconnu par OpenOffice.org qui permet de lui ajouter facilement une nouvelle fonctionnalité.

## Des applications à part entière pour l'entreprise

Un nombre croissant d'entreprises et d'administrations développent des applications internes basées sur OOoBasic et l'API d'OpenOffice.org. Des outils internes à l'API peuvent notamment permettre d'envisager une utilisation à travers un réseau voire Internet. Là encore, de nombreuses fonctionnalités sont présentes en interne.

Par exemple, si un important fonds documentaire est disponible dans un certain format et qu'il devient nécessaire d'en effectuer une migration pour obtenir une version PDF des documents, les fonctionnalités d'import/export le permettent.

Si des données sont éparpillées dans plusieurs sources et qu'il devient nécessaire de les fédérer voire d'en construire des graphes à intervalles donnés, l'accessibilité à l'API de Calc va pouvoir répondre au besoin.

Si un mailing requiert des interventions particulières ou s'il devient nécessaire de récupérer des données dans des documents contenant des champs utilisateurs afin de les consolider, là encore, l'API et les macros peuvent être utilisées.

Enfin, les macros peuvent servir à faire de petits scripts simples complètement déconnectés du contexte bureautique, comme des « moulinettes » sur des fichiers texte.

# Les macros et la sécurité

La puissance des macros comporte un revers : des individus peuvent écrire des documents anodins contenant des programmes conçus dans un but malveillant. Les utilisateurs de MS-Outlook, MS-Word et MS-Excel en savent quelque chose. En réalité, il est heureusement rare de récupérer de tels documents, mais beaucoup plus courant qu'un correspondant de bonne foi vous envoie un document avec une macro de son cru, et que celle-ci provoque des dégâts dans votre PC ou dans votre configuration OpenOffice.org. Ainsi, d'une manière générale, un document destiné à être diffusé devra être lisible sans macro.

Dans OpenOffice.org, l'utilisateur définit les conditions d'exécution des macros à partir du menu Outils>Options>OpenOffice.org>Sécurité. La figure 1-1 reproduit ce panneau.

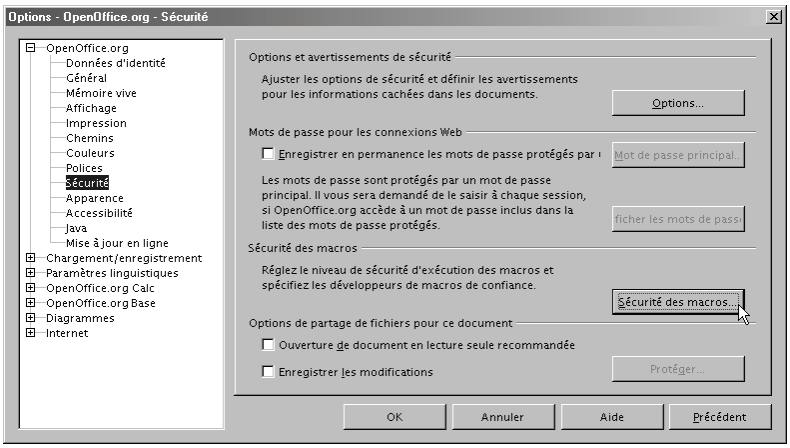



**Figure 1–1**
Entrée vers le panneau de sécurité des macros

En fait, tout se passe dans le panneau qui apparaît en cliquant le bouton Sécurité des macros, qui concerne tous les scripts, pas seulement Basic. Ce panneau comporte deux onglets, le premier est reproduit à la figure 1-2. Les textes explicatifs de chaque niveau ne sont pas tous corrects, nous allons voir exactement ce qu'il en est.

## Les différents niveaux de sécurité

Ces niveaux de sécurité ne concernent que les macros contenues dans un document. Les macros intégrées dans votre exemplaire d'OpenOffice.org sont supposées inoffensives (il s'agit des macros hébergées dans les zones Mes macros et Macros OpenOffice.org, et des extensions installées).

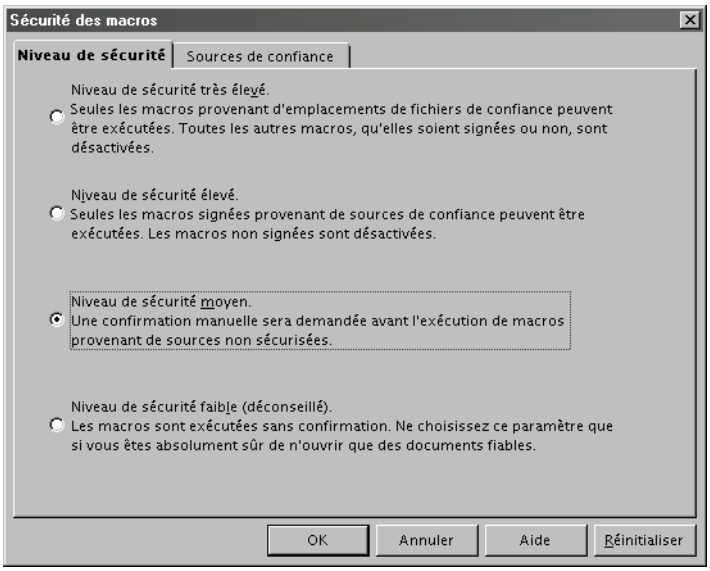



**Figure 1–2**
Onglet Niveau de sécurité des macros

## Niveau faible

Ce niveau de sécurité autorise toute macro, quelle que soit l'origine du document. Ne l'utilisez que si vous aimez vivre dangereusement.

## Niveau moyen

Le niveau de sécurité moyen vous avertit si le document contenant la macro ne se trouve dans aucun de vos répertoires de confiance (voir l'onglet Sources de confiance). Une bonne sécurité consiste à déclarer quelques répertoires de confiance, ceux où vous savez que certains documents nécessitent des macros. Quand vous récupérez un document inconnu, placez-le dans un répertoire ordinaire. Si le document contient des macros, OpenOffice.org vous en avertit (figure 1-3 pour des macros signées ou figure 1-4 pour des macros non signées) en vous donnant plusieurs possibilités d'action :

- Fermer la fenêtre (case X) désactive l'exécution des macros.
- Dans le cas de la figure 1-3, si vous cochez la case Toujours faire confiance aux macros provenant de cette source, les macros seront activées, le certificat sera ajouté dans la liste des certificats de confiance (figure 1-7) et la question ne vous sera plus posée pour les documents avec des macros portant la même signature, quel que soit leur emplacement. Ne cochez la case qu'après avoir cliqué sur le bouton Afficher les signatures. En effet, le certificat peut être invalide, ou le niveau de confiance faible s'il est délivré gratuitement par Internet.

- Dans le cas de la figure 1-4, ou si vous ne cochez pas la case de la figure 1-3, vous pouvez activer ou non l'exécution des macros pour cette fois-ci. Le fait de les désactiver n'empêche absolument pas d'ouvrir le document, ni de visualiser les instructions avec l'éditeur de macros (que nous verrons au chapitre 2).

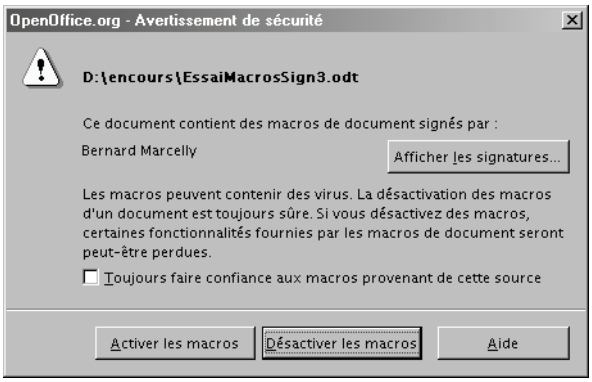



**Figure 1–3**
Niveau moyen, avertissement de macro signée

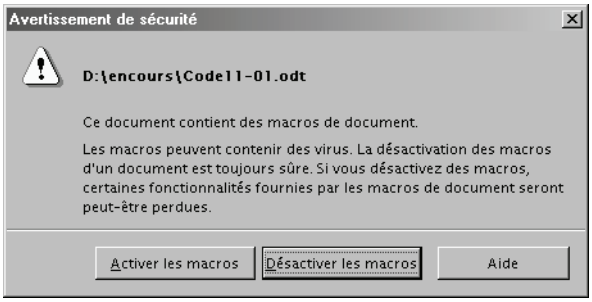



**Figure 1–4**
Niveau moyen, avertissement de macro

## Niveau élevé

Ce niveau autorise l'exécution des macros dont le document se trouve dans un des répertoires de confiance. Si le document se trouve en dehors de ces répertoires, OpenOffice.org considère deux cas :

- Soit les macros ne sont pas signées : elles sont systématiquement désactivées et OpenOffice.org affiche le message d'avertissement de la figure 1-5.

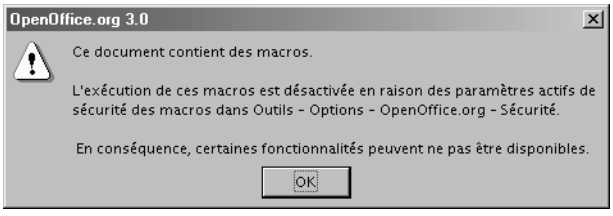



**Figure 1–5**
Niveau élevé, avertissement de macro

- Soit les macros sont signées, vous obtenez alors le message de la figure 1-6.

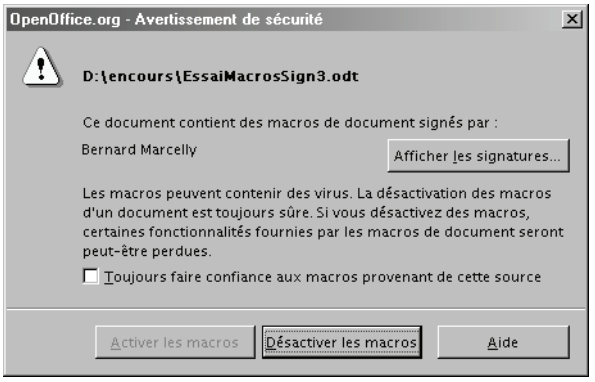



**Figure 1–6**
Niveau élevé, avertissement de macro signée

Si vous cochez la case Toujours faire confiance aux macros provenant de cette source, vous accédez au bouton Activer les macros. Après ouverture du document, le certificat sera ajouté dans la liste des certificats de confiance (figure 1-7) et la question ne vous sera plus posée pour les documents avec des macros portant la même signature, quel que soit leur emplacement. Ne cochez la case qu'après avoir cliqué sur le bouton Afficher les signatures. En effet, le certificat peut être invalide, ou le niveau de confiance faible s'il est délivré gratuitement par Internet. Si vous ne cochez pas la case, l'exécution de macros du document est désactivée.

### Niveau très élevé

La sécurité se base exclusivement sur les répertoires de confiance. Les macros de documents situés en dehors de ces répertoires sont systématiquement désactivées et OpenOffice.org affiche le message d'avertissement de la figure 1-5.

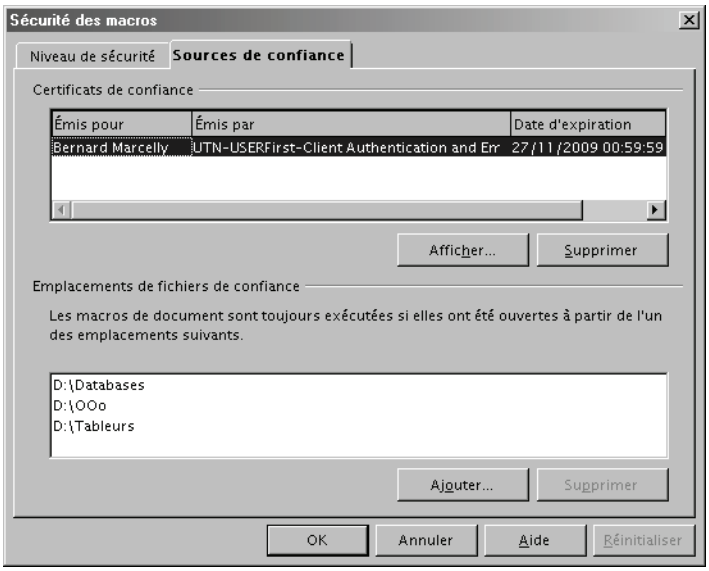
## Les sources de confiance

L'onglet Sources de confiance est reproduit à la figure 1-7.

La zone du haut liste les certificats de confiance qui représentent des signatures acceptables pour les macros. Lorsque, sur le message des figures 1-4 ou 1-6, vous cochez Toujours faire confiance aux macros de cette source, le certificat correspondant est automatiquement ajouté dans la liste des certificats de confiance.

La zone du bas liste les répertoires de confiance. Pour chacun, la confiance s'étend à toute l'arborescence de sous-répertoires qu'il contient. Évitez de mettre la racine d'un disque principal car tout le disque serait alors considéré comme de confiance, ce qui n'aurait plus de signification.

**Figure 1–7**
Onglet Sources de confiance




## Les signatures numériques

La création ou l'importation de certificats permettant de valider une signature numérique nécessite d'autres logiciels comme Firefox et autres navigateurs Internet. Nous ne détaillerons pas ici les procédures.

> **Se documenter**
>
> Dans l'aide (F1), cherchez dans l'index Signature et Utilisation des signatures numériques.

Un document OpenOffice.org peut être certifié numériquement par un ou plusieurs certificats. Notez qu'avec la version 3 d'OpenOffice.org les documents à signer doivent être sauvegardés au format ODF 1.2. On certifie un document avec le panneau obtenu par Fichier>Signatures numériques. Une fois certifié OpenOffice.org calcule une signature numérique sur le document. Cette signature de document est indépendante de la signature éventuelle des macros du document. De manière similaire, les macros d'un document peuvent être signées avec le panneau de la figure 1-8, obtenu avec le menu Outils>Macro>Signature numérique.

Le processus est le suivant :

1. Sauvez votre document, sans le fermer.
2. Ouvrez le panneau de la figure 1-8 et ajoutez un ou plusieurs certificats.
3. Fermez le document *sans* le sauver !

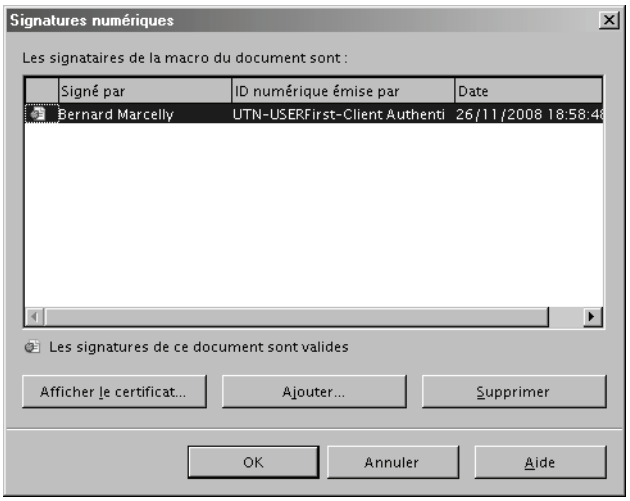



**Figure 1–8**
Signatures numériques des macros

Le document peut être copié ou déplacé, il gardera ses signatures.

Plus tard, après toute modification du document, répétez exactement le processus, car la sauvegarde supprime les signatures pour éviter qu'un tiers ne modifie les macros en gardant une apparence de sécurité.

# L'enregistreur de macros

L'enregistreur de macros enregistre les séquences de manipulations de l'utilisateur sous forme d'une macro, ce qui permet ensuite de reproduire à volonté la même séquence. La méthode est très simple et ne nécessite pas de connaissances de programmation.

## Comment enregistrer une macro ?

L'enregistrement est déclenché en cliquant sur Outils>Macros>Enregistrer une macro. À partir de cet instant, toutes les actions sur le document OpenOffice.org contenu dans la même fenêtre sont enregistrées. Vous remarquerez une petite fenêtre en avant-plan : elle vous permet de terminer l'enregistrement.

Ayant cliqué sur cette fenêtre de terminaison de macro, un autre panneau apparaît. Choisissez dans quelle bibliothèque et quel module vous souhaitez sauvegarder votre macro. Nous expliquerons plus loin ces termes et ce panneau. Essentiellement, vous avez le choix entre un module d'une bibliothèque disponible en permanence dans

OpenOffice.org et une bibliothèque propre au document en cours. En général, spécialement pour un débutant, il suffit de choisir la bibliothèque `Standard` du document en cours et de cliquer le bouton **Enregistrer** (enregistrer la macro). Un nouveau panneau apparaît, qui vous demande de choisir le nom du module, par exemple `Module1`. Maintenant, votre macro est écrite dans le module et elle a pour nom `Macro1` ou un nom similaire. Vous pouvez changer ce nom dans l'éditeur de macros qui s'affiche.

Notre document exemple `Code-02-01.odt` contient dans sa bibliothèque `Standard` une séquence réalisée avec l'enregistreur de macros : aller à la fin du document, écrire un texte et terminer le paragraphe. Voici le codage obtenu (les lignes blanches sont omises).

```
sub BonjourEnregistreur
rem ----------------------------------------------------------------------
rem define variables
dim document   as object
dim dispatcher as object
rem ----------------------------------------------------------------------
rem get access to the document
document   = ThisComponent.CurrentController.Frame
dispatcher = createUnoService("com.sun.star.frame.DispatchHelper")
rem ----------------------------------------------------------------------
dispatcher.executeDispatch(document, ".uno:GoToEndOfDoc", "", 0, Array())
rem ----------------------------------------------------------------------
dim args2(0) as new com.sun.star.beans.PropertyValue
args2(0).Name = "Text"
args2(0).Value = "L'enregistreur de macros vous salue !"
dispatcher.executeDispatch(document, ".uno:InsertText", "", 0, args2())
rem ----------------------------------------------------------------------
dispatcher.executeDispatch(document, ".uno:InsertPara", "", 0, Array())
end sub
```

Le document exemple contient d'autres scripts réalisant exactement la même chose en Basic, JavaScript, BeanShell, Python. En comparant les codages, vous constaterez que l'enregistreur de macro produit un codage très différent des autres : en effet il n'utilise pas les fonctions de l'API, mais seulement un mécanisme appelé *dispatch*.

## Un outil limité

L'enregistreur de macros souffre de limitations assez sévères :

- Il n'est disponible que sous Writer et Calc.
- Il ne sait que « mimer » des actions de l'utilisateur, et encore, pas toutes.
- Il utilise des commandes peu documentées (les « slot ID »).
- Il ne permet pas d'écrire des macros interactives.

- Il produit un codage Basic non optimisé qui est assez difficile à lire, sans rapport avec un « vrai » codage Basic OpenOffice.org.

Par ailleurs, mais c'est le principe d'un tel enregistreur, il ne peut produire que du codage linéaire (c'est-à-dire qu'il est incapable de faire par exemple une boucle pour répéter une action sur une liste d'objets ou de choisir entre plusieurs alternatives).

Pour avoir plus de possibilités, il faut écrire soi-même les instructions de la macro, ce qui nécessite de connaître un langage de programmation et l'API OpenOffice.org. C'est la voie qui est développée dans ce livre.

Il est cependant des cas où l'enregistreur de macros nous sera utile : lorsque l'API ne permet pas certaines manipulations que peut réaliser l'enregistreur, ou seulement au prix de développements complexes. Il est alors possible de résoudre la difficulté en combinant un codage Basic avec les instructions créées par l'enregistreur.

# Les différents langages de script

Depuis la version 2, OpenOffice.org intègre plusieurs langages de script, et pas seulement Basic. Chaque langage a ses avantages et ses défauts, et un programmeur expérimenté préfèrera celui qui est le plus adapté à son projet, ou même à une partie du projet. Nous appellerons macro ou script tout programme réalisé avec un de ces langages.

Certains détails de cette section sont destinés aux lecteurs ayant acquis une bonne connaissance de la programmation avec OpenOffice.org.

## Basic OpenOffice.org

Basic sera notre langage de développement dans ce livre, mais nous ne l'aborderons qu'avec le chapitre 2. Pour vous donner un avant-goût, voici une petite macro `BonjourBasic`, que vous trouverez dans la bibliothèque `Library1` du document exemple `Code-02-01.odt`. Elle réalise l'équivalent de l'exemple de l'enregistreur de macros.

```
Sub BonjourBasic
Dim monDocument As Object, monTexte As Object, monCurseur As Object
monDocument = ThisComponent
monTexte = monDocument.Text
monCurseur = monTexte.createTextCursor
monCurseur.gotoEnd(false)
monTexte.insertString(monCurseur, "Basic et l'API vous saluent !", false)
monTexte.insertControlCharacter(monCurseur, _
com.sun.star.text.ControlCharacter.PARAGRAPH_BREAK, false)
End Sub
```

## Particularités des autres langages de script

Les langages autres que Basic sont gérés par le Scripting Framework, qui fait l'objet d'un chapitre complet dans le *Developer's Guide* (documentation en anglais) disponible en ligne à l'adresse suivante :

http://wiki.services.openoffice.org/wiki/Documentation/DevGuide/OpenOffice.org_Developers_Guide

Une variable prédéfinie `XSCRIPTCONTEXT` est disponible dans un script. Cet objet expose trois méthodes :

- `getDocument()` renvoie l'objet document en cours.
- `getDesktop()` renvoie l'objet application OpenOffice (équivalent du Basic `StarDesktop`).
- `getComponentContext()` renvoie le contexte, nécessaire pour appeler certaines méthodes.

Les dialogues, que nous verrons au chapitre 11, peuvent être appelés par un script quelconque, et les événements de dialogue peuvent aussi être traités par un script.

Il est plus facile de développer un script non Basic dans Mes macros, quitte à le transférer ensuite dans un document avec adaptation éventuelle.

## Java compilé

OpenOffice.org peut exécuter des scripts en Java compilé (fichiers `.jar`). Mais il n'existe pas de panneau Macros correspondant, et leur installation devra être faite manuellement par un programmeur confirmé ou par le biais d'extensions.

## JavaScript

Langage bien connu des créateurs de sites web, JavaScript est utilisé ici comme langage de programmation indépendant de toute page web. Chaque macro JavaScript est contenue dans un fichier portant l'extension `.js`, stocké dans un répertoire bibliothèque. Le panneau Macros JavaScript de la figure 1-9 est obtenu par le menu Outils>Macros>Gérer les macros>JavaScript. Ici la macro se trouve dans la bibliothèque `Library3` du document `Code01-01.odt`.

Vous pouvez créer une nouvelle bibliothèque ou une nouvelle macro dans celle sélectionnée. Le bouton Éditer affiche le contenu de la macro dans la fenêtre de l'éditeur OpenSource Rhino (voir figure 1-10). Il offre des possibilités d'évaluation de variables, de pas-à-pas et de point d'arrêt ; mais il n'y a ni coloration syntaxique ni aide en ligne sur les instructions. Il souffre actuellement de défauts rédhibitoires (Issue 70176, Issue 70215) qui conduisent à le déconseiller et préférer un éditeur séparé pour modifier le fichier.

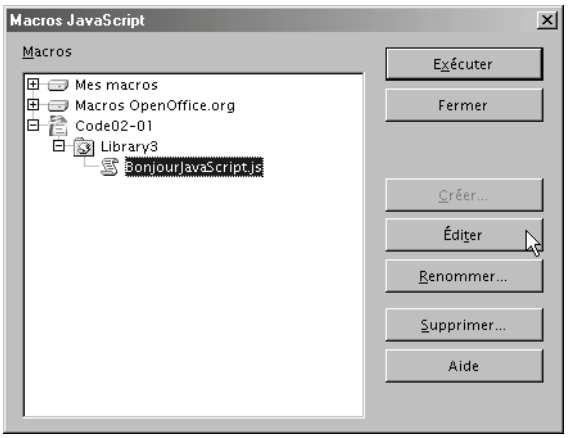



**Figure 1–9**
Panneau Macros JavaScript

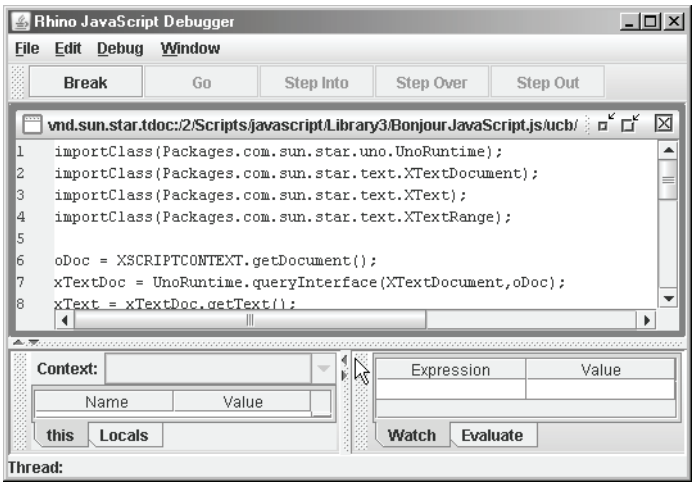



**Figure 1–10**
Éditeur de JavaScript

Quand vous créez une nouvelle macro JavaScript, OpenOffice.org écrit automatiquement un codage de type HelloWorld. S'il peut servir d'exemple sous Writer, il est inutilisable sous Calc, Draw, etc.

Voici le codage du script `BonjourJavaScript.js`. Comme pour Java, il est nécessaire d'obtenir explicitement chaque interface dont on utilise une méthode. Par contre il n'y a pas de typage de données.

```
importClass(Packages.com.sun.star.uno.UnoRuntime);
importClass(Packages.com.sun.star.text.XTextDocument);
importClass(Packages.com.sun.star.text.XText);
importClass(Packages.com.sun.star.text.XTextRange);
```

```
oDoc = XSCRIPTCONTEXT.getDocument();
xTextDoc = UnoRuntime.queryInterface(XTextDocument,oDoc);
xText = xTextDoc.getText();
xTCursor = xText.createTextCursor();
xTCursor.gotoEnd(false);
xText.insertString( xTCursor, "JavaScript vous salue ! " , false);
xText.insertControlCharacter(xTCursor,
Packages.com.sun.star.text.ControlCharacter.PARAGRAPH_BREAK, false);
```

Les arguments passés à une macro JavaScript sont récupérés dans une variable globale prédéfinie `ARGUMENTS` qui est un tableau de valeurs de type `Object`. Par exemple, ici on récupère l'objet événement transmis par le déclenchement d'un bouton de formulaire :

```
evt = ARGUMENTS[0];
```

**Pour aller plus loin**

Site web de Rhino, en anglais :
- http://www.mozilla.org/rhino/

Un site web français sur JavaScript :
- http://www.toutjavascript.com/

## BeanShell

BeanShell est une sorte de Java interprété et plus simple au niveau des déclarations. Chaque macro BeanShell est contenue dans un fichier portant l'extension `.bsh`, stocké dans un répertoire bibliothèque. Le panneau Macros BeanShell de la figure 1-11 est obtenu par le menu Outils>Macros>Gérer les macros>BeanShell. Ici la macro se trouve dans la bibliothèque `Library2` du document `Code01-01.odt`.

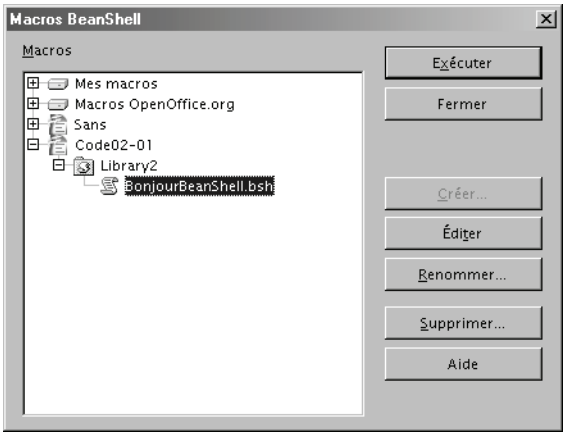



**Figure 1–11**
Panneau Macros BeanShell

Vous pouvez créer une nouvelle bibliothèque ou une nouvelle macro dans celle sélectionnée. Le bouton Éditer affiche le contenu de la macro dans la fenêtre de l'éditeur BeanShell (voir figure 1-12). C'est un éditeur rustique qui n'offre ni police à espacement fixe, ni coloration syntaxique, et aucun outil de mise au point.

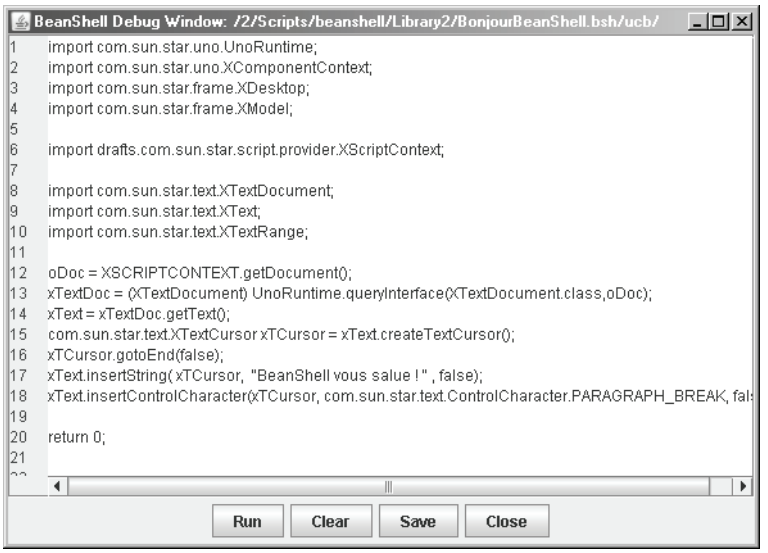



**Figure 1–12**
Éditeur de BeanShell

Comme avec JavaScript, lorsque vous créez une nouvelle macro BeanShell, OpenOffice.org présente automatiquement un codage type « HelloWorld ». Il peut servir d'exemple sous Writer, mais il est inutilisable sous Calc, Draw, etc.

Voici le codage du script `BonjourBeanShell.bsh`. Comme pour Java, il est nécessaire d'obtenir explicitement chaque interface dont on utilise une méthode. Le principal intérêt de BeanShell réside dans sa capacité à utiliser des codages Java tout en profitant d'un langage plus souple pour les types de données.

```
import com.sun.star.uno.UnoRuntime;
import com.sun.star.uno.XComponentContext;
import com.sun.star.frame.XDesktop;
import com.sun.star.frame.XModel;

import drafts.com.sun.star.script.provider.XScriptContext;

import com.sun.star.text.XTextDocument;
import com.sun.star.text.XText;
import com.sun.star.text.XTextRange;
```

```
oDoc = XSCRIPTCONTEXT.getDocument();
xTextDoc = (XTextDocument) UnoRuntime.queryInterface(
  XTextDocument.class,oDoc);
xText = xTextDoc.getText();
com.sun.star.text.XTextCursor xTCursor = xText.createTextCursor();
xTCursor.gotoEnd(false);
xText.insertString( xTCursor, "BeanShell vous salue ! " , false);
xText.insertControlCharacter(xTCursor,
  com.sun.star.text.ControlCharacter.PARAGRAPH_BREAK, false);

return 0;
```

Les arguments passés à une macro BeanShell sont récupérés dans une variable globale prédéfinie `ARGUMENTS`, qui est un tableau de valeurs de type `Object`. Par exemple, ici on récupère l'objet événement transmis par le déclenchement d'un bouton de formulaire :

```
evt = (ActionEvent) ARGUMENTS[0];
```

**Plus d'informations**

- http://www.beanshell.org/

## Python

Python est un langage Open Source, très puissant et original. Les points remarquables, par rapport à Basic, sont les suivants :

- 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 incluses dans Python et ses modules principaux.
- La gestion des erreurs est celle des langages modernes.
- La programmation objet facilite la conception de programmes complexes.

Python permet la création de composants UNO, contrairement à Basic. Un composant UNO ajoute un service API qui peut être utilisé par tout autre langage.

Chaque macro Python est une fonction déclarée dans un fichier ayant l'extension `.py`, stocké dans le sous-répertoire `Scripts/python/` ou un sous-répertoire de celui-ci. Le panneau Macros Python de la figure 1-13 est obtenu par le menu Outils>Macros>Gérer les macros>Python. Ici la macro se trouve dans la bibliothèque `Library3` du document `Code01-01.odt`. Un fichier source peut comporter plusieurs fonctions appelables.

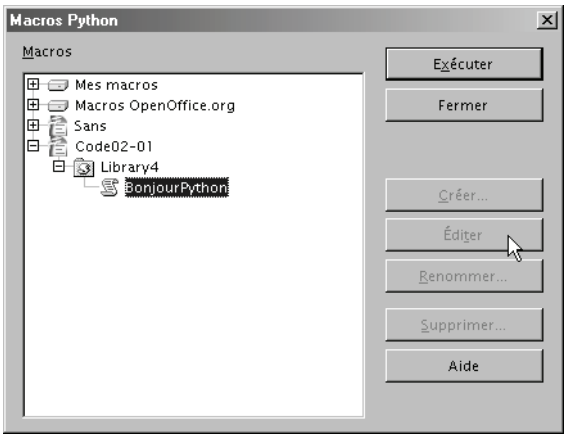



**Figure 1–13**
Panneau Macros Python

Vous remarquerez sur la figure 1-13 que les boutons Créer, Éditer, Renommer, Supprimer sont inactifs ; ce qui ne laisse en fait que la possibilité d'exécuter un codage existant. OpenOffice.org n'offre pas encore d'éditeur pour Python. Le développeur de macros Python doit donc utiliser un éditeur externe pour modifier son fichier et travaillera dans Mes macros. Il n'y a pas d'outil de mise au point, et les messages d'erreur apportent malheureusement peu d'informations utiles.

Les programmeurs Python noteront qu'il est nécessaire d'utiliser l'interpréteur Python intégré à OpenOffice.org, et que l'IDLE n'est pas non plus utilisable pour exécuter un script dans OpenOffice.org. Ces limitations sont dues au manque de développeurs. Souhaitons que quelques développeurs ou des entreprises généreuses proposent leur aide à la communauté OpenOffice.org afin de mieux intégrer ce langage. Dans l'état actuel, les avantages de Python dans OpenOffice sont plutôt réservés aux programmeurs expérimentés sachant de surcroît lire l'anglais.

Voici le codage du script `BonjourPython`. L'utilisation de l'API OpenOffice.org est très semblable à Basic.

Faites attention à ne pas oublier deux particularités de Python : respecter la casse (majuscules/minuscules) pour les noms de variables, et mettre des parenthèses vides dans tout appel d'une méthode sans argument.

```
from com.sun.star.text.ControlCharacter import PARAGRAPH_BREAK

def BonjourPython( ):
    """Ecrit un texte dans le document Writer"""
    monDocument = XSCRIPTCONTEXT.getDocument()
    monTexte = monDocument.Text
    monCurseur = monTexte.createTextCursor()
    monCurseur.gotoEnd(False)
```

```
    monTexte.insertString(monCurseur, "Python vous salue !", False)
    monTexte.insertControlCharacter(monCurseur, PARAGRAPH_BREAK, False)
    return None
```

**Pour plus d'informations**

Deux pages web de référence, en anglais, à lire attentivement pour développer des macros en Python :
- http://wiki.services.openoffice.org/wiki/Python_as_a_macro_language
- http://udk.openoffice.org/python/python-bridge.html/

Le site web Python, en anglais :
- http://www.python.org/

L'Association francophone Python :
- http://www.afpy.org/

# Exécuter une macro depuis OpenOffice.org

Une macro s'intègre dans les mécanismes de votre exemplaire d'OpenOffice.org, ajoutant ainsi une fonctionnalité, soit dans le contexte d'un type de document, soit pour tous les documents. Vous pouvez alors la déclencher de différentes façons.

## Exécuter une macro depuis le menu Outils

Avec le menu Outils>Macros>Exécuter la macro vous obtenez le panneau reproduit à la figure 1-14.

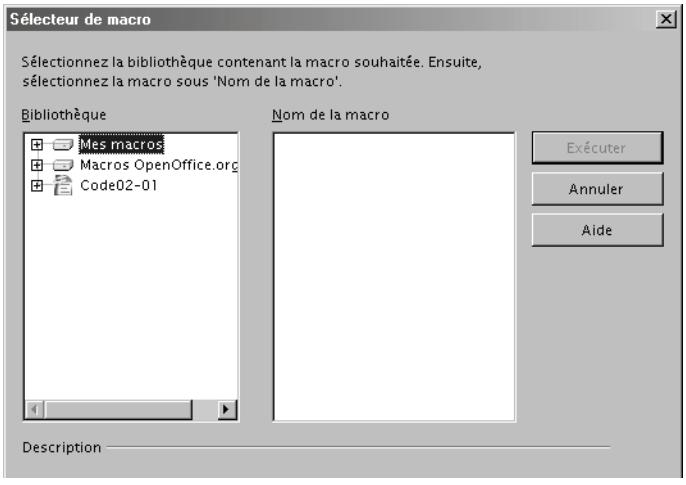



**Figure 1–14**
Panneau du Sélecteur de macro

Remarquez qu'il existe trois arbres contenant chacun des bibliothèques de macros :

- Mes macros correspond aux macros que vous avez ajoutées et qui sont disponibles pour toute application OpenOffice.org et depuis tout document.
- Macros OpenOffice.org contient des bibliothèques de macros fournies par OpenOffice.org. Dans une installation réseau, l'administrateur peut y ajouter des bibliothèques qui seront alors disponibles pour tous les utilisateurs OpenOffice.org.
- Code02-01 est le nom d'un document que nous avons ouvert. Un document peut aussi contenir des bibliothèques de macros.

Développez l'arborescence jusqu'à trouver le module contenant votre macro, sélectionnez-la, et cliquez sur le bouton Exécuter. Dans la figure 1-15 nous avons choisi une macro Basic contenue dans Mes macros.

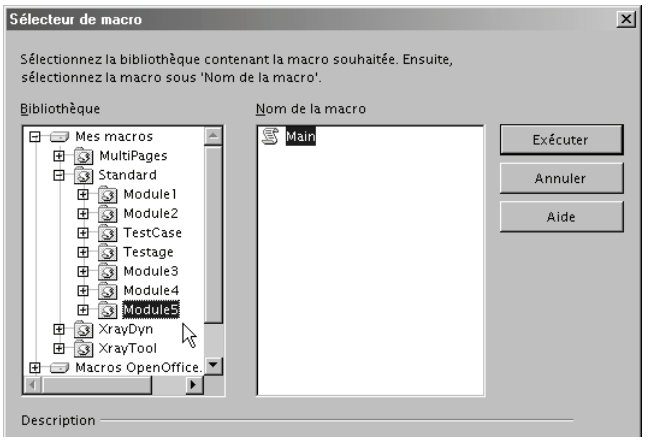



**Figure 1–15**
Sélection d'une macro Basic

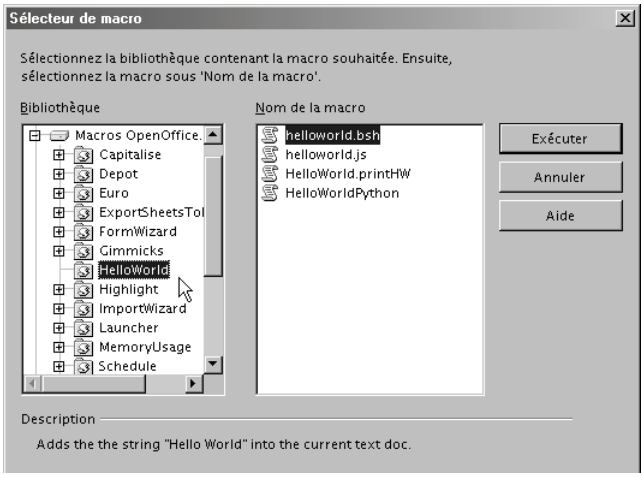
Ce même panneau permet de choisir une macro écrite dans un autre langage que Basic. Il en existe quelques-unes fournies avec l'installation, dans la branche Macros OpenOffice.org. La figure 1-16 vous en montre plusieurs :

- `helloworld.bsh` est une macro BeanShell ;
- `helloworld.js` est une macro JavaScript ;
- `HelloWorld.printHW` est une macro en Java compilé ;
- `HelloWorldPython` est une macro en Python.

Vous remarquerez que, sauf pour les macros Basic, il est possible d'afficher un commentaire descriptif de la macro. Le document exemple `Code01-01.odt`, disponible dans le Zip téléchargeable sur le site des éditions Eyrolles, contient lui aussi des macros écrites dans ces langages.

Vous retrouverez la même structure arborescente dans les autres manières de déclencher une macro par l'interface utilisateur.

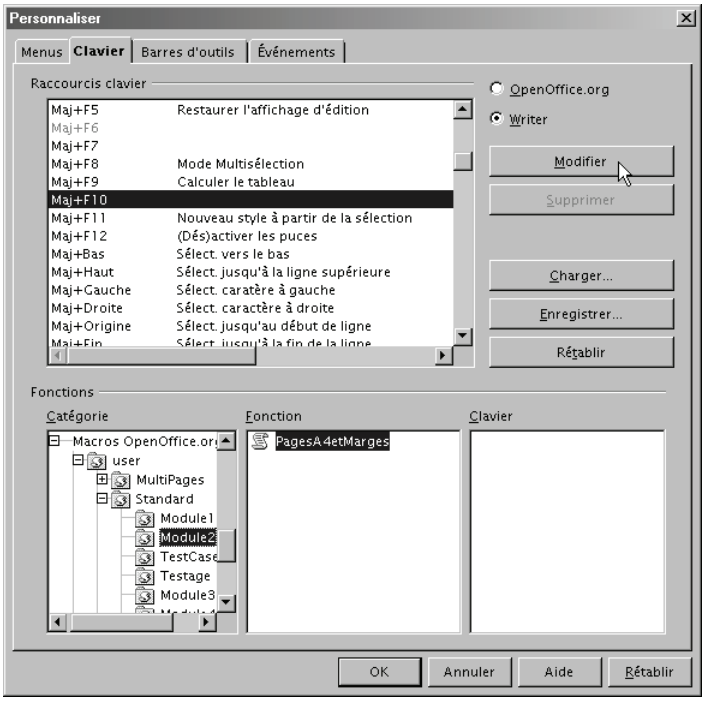
**Figure 1–16**
Macros écrites
en divers langages




## Exécuter une macro depuis un raccourci clavier

Avec le menu Outils>Personnaliser vous obtenez le panneau Personnaliser, qui comporte un onglet Clavier. Cet onglet est représenté sur la figure 1-17.

**Figure 1–17**
Ajouter un raccourci clavier



Pour affecter une macro à un raccourci clavier afin de pouvoir ensuite l'exécuter d'une simple combinaison de touches, il faut commencer par déterminer si la macro sera disponible pour une application particulière (ici tous les documents Writer), ou pour toute application OpenOffice.org (Calc, Draw, Writer, etc). Selon ce choix, le cadre Raccourcis clavier présente la liste des raccourcis possibles, en indiquant ceux déjà utilisés. Choisissez un raccourci inutilisé ou supprimez un raccourci déjà attribué.

Passez ensuite à la minuscule fenêtre Catégorie. Il faut utiliser les ascenseurs pour l'explorer. Dans la partie inférieure sont regroupées les macros dans la branche Macros OpenOffice.org. Cliquez sur les signes + et vous trouvez :

- `user` : correspond à l'arborescence Mes macros vue avec le menu Outils.
- `share` : correspond à l'arborescence Macros OpenOffice.org du menu Outils.
- `un nom de document` : regroupe les macros éventuellement présentes dans un document actuellement ouvert. Attention, il faut jamais utiliser une macro de document ! En effet, le raccourci sera valable pour tout document.

En cliquant successivement pour développer l'arborescence, vous finissez par faire apparaître dans la fenêtre Fonctions une liste de noms de macros. Une fois que le bon raccourci et la bonne commande sont tous deux sélectionnés dans les panneaux du haut et du bas, cliquez sur le bouton Modifier. La touche ou combinaison de touches s'affiche dans la fenêtre Clavier et le nom de la macro s'inscrit en face du raccourci. Cliquez sur OK pour refermer la boîte de dialogue.

Dans votre document, saisissez un paragraphe, puis testez la macro en l'invoquant par son raccourci. À l'appel du raccourci clavier, la macro s'exécute.

## Exécuter une macro avec un bouton de barre d'outils

Vous pouvez également lancer votre macro en cliquant sur un nouveau bouton dans une des barres d'outils.

Cliquez sur la flèche descendante tout au bout à droite sur la barre d'outils. Dans le menu contextuel, choisissez Personnaliser la barre d'outils. Vous obtenez le panneau de la figure 1-18.

Choisissez d'enregistrer dans OpenOffice.org ou dans une des applications (ici Writer). Cliquez sur la position de votre futur bouton, puis cliquez sur Ajouter. Vous obtenez le panneau de la figure 1-19.

Le choix de la macro s'effectue comme pour un raccourci clavier. Vous aurez sans doute envie d'affecter une icône à ce nouveau bouton. Dans le panneau de la figure 1-18 cliquez sur le bouton Modifier puis choisissez Changement d'icône dans le menu.

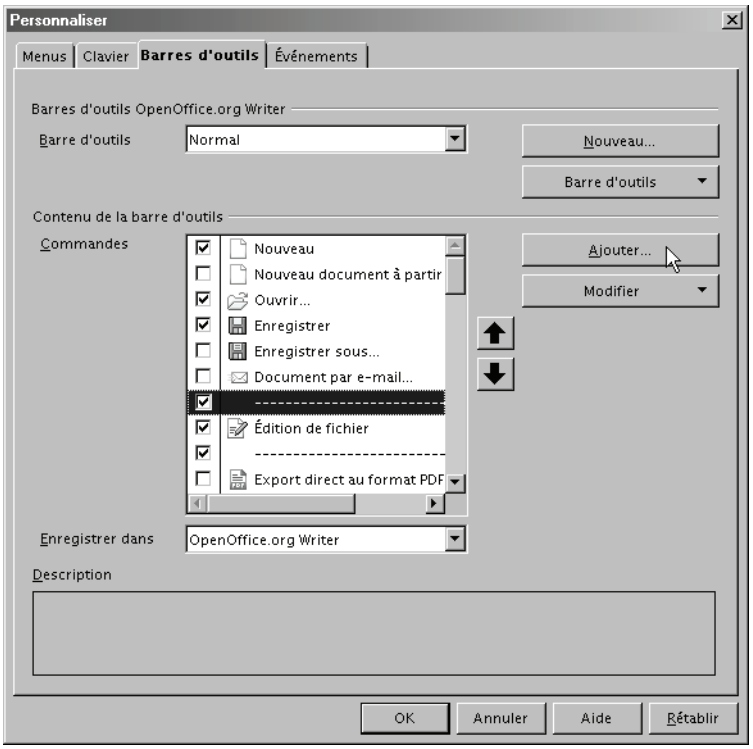



**Figure 1–18**
Personnaliser
une barre d'outils

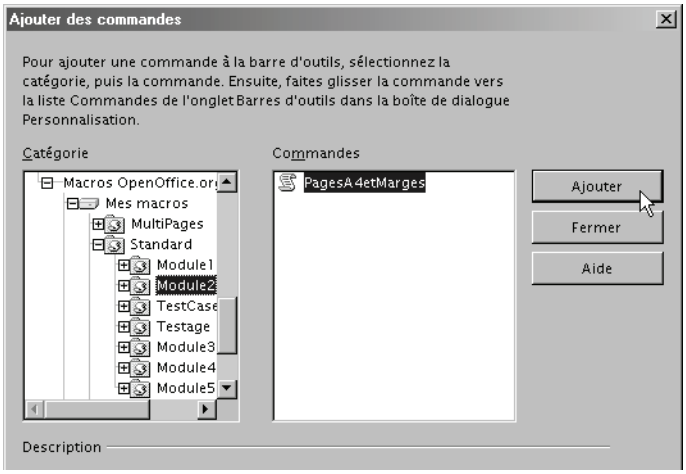



**Figure 1–19**
Ajouter une macro
sur une barre d'outils

## Exécuter une macro par une entrée de menu

OpenOffice.org vous permet de modifier les éléments dans une liste de menu. Il est ainsi possible d'ajouter un élément qui déclenchera une macro. Avec le menu Outils>Personnaliser vous obtenez le panneau Personnaliser, qui comporte un onglet Menus, représenté sur la figure 1-20.

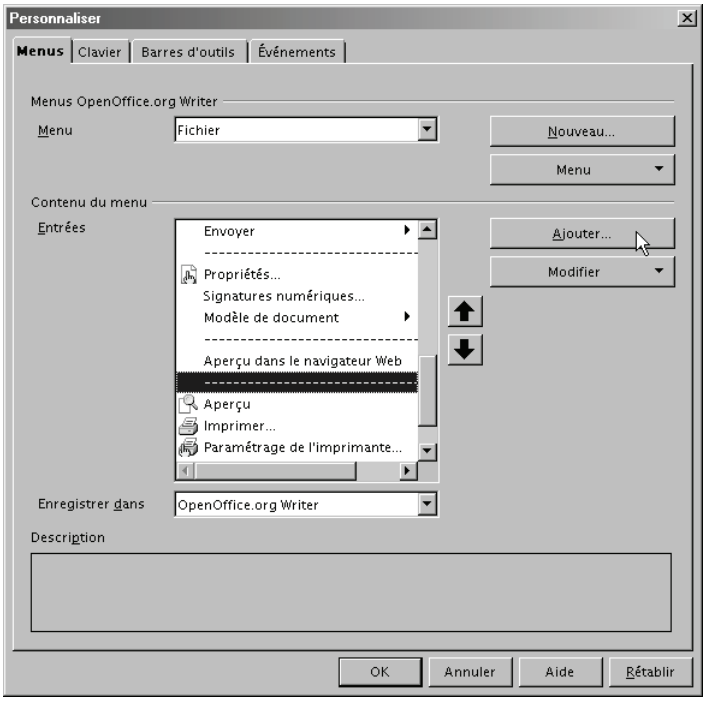



**Figure 1–20**
Personnaliser une entrée de menu

Choisissez d'enregistrer l'entrée de menu dans l'application (ici, Writer) ou dans un document ouvert. Le reste de la procédure est identique à celle présentée à la section précédente.

## Exécuter une macro depuis une extension

Une extension peut apporter une nouvelle barre d'outils, ou modifier une barre d'outils existante, ou encore ajouter des entrées de menu ; tout dépend du concepteur de l'extension. Une fois installée, les macros de l'extension peuvent être déclenchées avec les nouveaux boutons ou les nouvelles entrées de menu.

## Exécuter une macro depuis un document

Un simple bouton de formulaire déposé sur une page de document Writer, Calc ou Draw permet de déclencher une macro. Les contrôles de formulaire utilisent très fréquemment des macros, comme nous le verrons au chapitre 13. Une forme dessinée dans Writer, Calc, Draw, Impress peut déclencher une macro lorsque l'utilisateur clique dessus.

Sur une page de document Writer, vous pouvez insérer un champ permettant de déclencher une macro par hyperlien. Pour cela, dans le menu Insertion>Champs>Autres, cliquez sur l'onglet Fonctions et dans le cadre Type de champ, cliquez sur Exécuter la macro. Choisissez votre macro, puis tapez le texte du lien dans la zone Annotation, et insérez.

Dans Calc, il est possible de créer (avec une macro) votre fonction de calcul pour l'utiliser dans une formule de cellule. Nous verrons au chapitre 9 comment procéder. Notez également que le contrôle de validité d'une cellule peut appeler une macro en cas d'entrée non valide.

## Exécuter une macro sur un événement

Pour certains usages, il est parfois intéressant de déclencher automatiquement une macro sur un événement tel que l'ouverture ou la fermeture d'un document. Pour cela, nous utilisons l'onglet Événements du panneau Personnaliser (voir la figure 1-21).

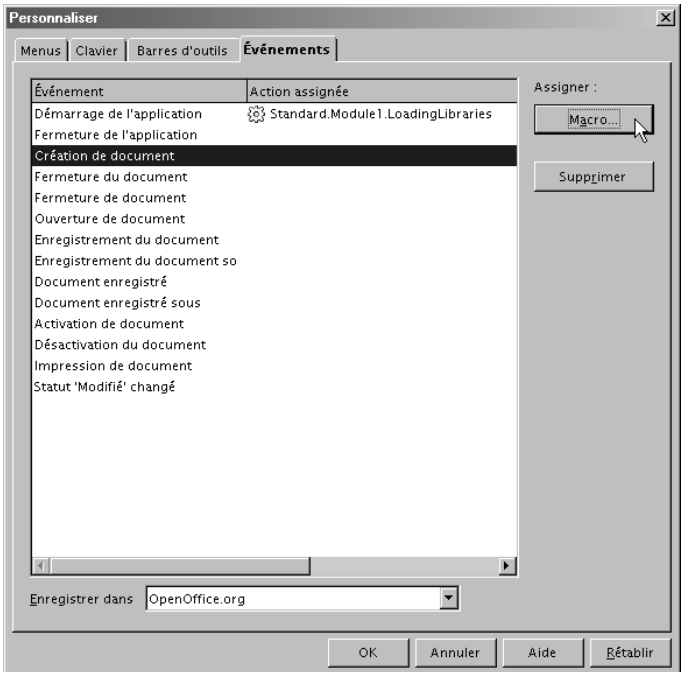



**Figure 1–21**
Déclencher une macro sur un événement

Déterminez si l'affectation doit concerner OpenOffice.org ou le document ouvert. Choisissez l'événement. Cliquez sur le bouton Macro. Le panneau Sélecteur de macro apparaît, la suite est identique aux cas précédents. La figure montre qu'une macro est déjà affectée à l'événement Démarrage de l'application.

Bien d'autres événements peuvent faire l'objet d'un traitement par macro, par exemple un clic de souris, une touche enfoncée, etc.

# Exécuter une macro en ligne de commande

Il est parfaitement possible de lancer OpenOffice.org pour simplement exécuter une macro, sans passer par l'interface utilisateur. La syntaxe diffère selon le système d'exploitation. Nous décrirons surtout l'appel d'une macro Basic sous MS-Windows, puis nous verrons comment exécuter des macros d'autres langages. Enfin, nous signalerons les particularités propres à Linux.

Cette section nécessite des connaissances de base sur les programmes en ligne de commande (MS-DOS ou Unix Shell). Nous supposons en outre que vous avez une connaissance minimale du Basic OpenOffice.org, sinon nous vous conseillons de vous reporter auparavant au chapitre 2.

## Sous Windows

### Lancer une macro Basic résidente

On appelle « résidente » une macro qui se trouve dans une des bibliothèques de Mes macros ou Macros OpenOffice.org, et qui est donc indépendante de tout document. Pour l'exécuter, il faut indiquer successivement les éléments suivants dans la ligne de commande :

1. Le chemin d'accès et le nom de l'exécutable d'OpenOffice.org (`soffice.exe`). Notons que ce chemin dépend du système d'exploitation, de la version et des conditions d'installation d'OpenOffice.org.
2. Le paramètre optionnel `-headless` si on souhaite empêcher l'affichage de tout message à destination de l'utilisateur.
3. Une séquence de caractères qui se présente sous la forme du mot `macro` suivi du nom de la bibliothèque, du nom du module, du nom de la macro, et de ses arguments :

```
macro:///maLib.monModule.maMacro(Arg1, Arg2,...)
```

**Ressources**

Dans le Zip téléchargeable, vous trouverez dans le dossier des macros de ce chapitre, les fichiers correspondants à nos exemples.

Dans Mes Macros, créez la bibliothèque `maBibli1` et le module `monModule`. Dans ce dernier, écrivez la macro suivante, ou faites un copier-coller de la même macro qui se trouve dans le fichier `Code01-02.odt` :

```
' écrit un fichier texte comportant n fois le caractère c
Sub bavard(c As String, n As Integer)
Dim f As Integer
f = FreeFile

open "C:\essaiMacro.txt" for Output As f
Write #f, Time & " bavard a dit : " & String(n, c)
close #f

msgbox "bavard a terminé"
end sub
```

Cette macro écrit l'heure courante et un texte dépendant de la valeur de ses arguments. Adaptez éventuellement le chemin du fichier résultat `essaiMacro.txt` à votre propre configuration.

Réalisez ce fichier batch (fichier `LanceBavard1.bat`) contenant ceci (l'instruction principale doit être écrite en une seule ligne).

```
rem lancement de la macro
"C:\Program Files\OpenOffice.org 3\program\soffice.exe"
 ➥ "macro:///maBibli1.monModule.bavard(A, 20)"
pause
```

Modifiez si besoin le chemin d'accès à `soffice`, il dépend de la version OpenOffice.org utilisée. Mettez ce fichier batch dans un répertoire où OpenOffice.org autorise les exécutions de macros.

Arrêtez l'application OpenOffice.org, y compris le lanceur. Exécutez le fichier batch. La fenêtre MS-DOS va s'ouvrir, puis le logo de démarrage d'OpenOffice.org va s'afficher et, après un certain temps, le message « bavard a terminé » apparaîtra dans un petit panneau. Cliquez sur OK, puis terminez l'exécution du batch en appuyant sur une touche. Vérifiez que le fichier `essaiMacro.txt` est bien écrit.

Réalisez un deuxième fichier batch, (`LanceBavard2.bat`) contenant cette variante :

```
rem lancement de la macro
"C:\Program Files\OpenOffice.org 3\program\soffice.exe"
 ➥ -headless "macro:///maBibli1.monModule.bavard(B, 15)"
pause
```

Arrêtez l'application OpenOffice.org, y compris le lanceur. Exécutez le fichier batch. La fenêtre MS-DOS va s'ouvrir et, après un certain temps, affichera l'attente. Aucun autre message n'est apparu. Terminez l'exécution du batch en appuyant sur une touche. Vérifiez que le fichier `essaiMacro.txt` est bien écrit avec cette deuxième commande.

Lorsque l'argument `-headless` est employé, les éventuelles boîtes de dialogue de OOo, comme la confirmation de lancement des macros, reçoivent la réponse par défaut.

Ouvrez un document OpenOffice.org quelconque et relancez le même fichier batch `LanceBavard2.bat`. Cette fois-ci, le message « bavard a terminé » s'affiche, malgré l'option `-headless`.

### Arguments d'appels de la macro

Comme vous l'avez constaté, le premier argument est considéré comme une chaîne d'un seul caractère ; le deuxième est aussi une chaîne de caractères, mais Basic la convertit en une valeur numérique valide pour le paramètre `n` de la macro.

Une chaîne de caractères est transmise telle quelle, sans ajouter de guillemets, sous Windows XP. Sous d'autres versions de MS-Windows, il peut être nécessaire d'ajouter des guillemets. Il n'est pas possible de transmettre une chaîne de caractères comportant un guillemet ou une virgule, et les lettres accentuées sont modifiées.

On peut transmettre à la macro les arguments d'appel du fichier batch, par exemple :

```
"macro:///maBibli1.monModule.bavard(%1, %2)"
```

### Lancer une macro Basic contenue dans un document

Si le batch appelle une macro contenue dans un document OpenOffice.org la méthode diffère sur plusieurs points :

- On doit ajouter le chemin d'accès au document s'il n'est pas déjà chargé.
- L'argument `macro` doit comporter le nom court, sans extension, du document, sous cette forme (attention au nombre de caractères /) : `macro://monDoc/maLib.monModule.maMacro(Arg1, Arg2,...)`
- Le document ne se fermera pas automatiquement.

Sur le dernier point, il est possible d'écrire à la fin de la macro une instruction fermant le document, voir le chapitre 7.

Réalisez ce fichier batch (`LanceBavard3.bat`) :

```
rem lancement de la macro
"C:\Program Files\OpenOffice.org 3\program\soffice.exe"
 ➥ "Code01-02.odt" "macro://Code01-02/Standard.monModule.bavard(C,20)"
pause
```

Mettez ce batch ainsi que le fichier `Code01-02.odt` (disponible dans le Zip téléchargeable) dans un même répertoire, qui doit être le répertoire courant pour la ligne de commande.

Arrêtez l'application OpenOffice.org, y compris le lanceur. Exécutez le fichier batch. La fenêtre MS-DOS va s'ouvrir, puis le document va s'afficher et, après un certain temps, le message « bavard a terminé » apparaîtra dans un petit panneau. Cliquez sur OK. Le document reste ouvert. L'exécution du batch ne reprend qu'à la fermeture d'OpenOffice.org. Vérifiez que le fichier `essaiMacro.txt` est bien écrit. L'option `-headless` donnerait le même résultat.

Si le document contenant la macro est déjà chargé, le fichier batch est un peu simplifié (`LanceBavard4.bat`) :

```
rem lancement de la macro
"C:\Program Files\OpenOffice.org 3\program\soffice.exe"
 ➥ "macro://Code01-02/Standard.monModule.bavard(D,20)"
pause
```

Chargez au préalable le document `Code01-02.odt`. Exécutez le fichier batch. La fenêtre DOS va s'ouvrir et, après un certain temps, le message « bavard a terminé » apparaîtra dans un petit panneau. Cliquez sur OK. Le batch reprend son exécution (et pause). Le fichier reste ouvert.

### Lancer un script autre que Basic

Pour lancer un script d'un langage quelconque, il faut employer le Scripting Framework, qui a deux limitations quand on l'utilise depuis un batch :

- On ne peut lancer qu'un script « résident » (situé dans Mes macros ou dans Macros OpenOffice.org). Il est impossible de lancer un script d'un document.
- On ne peut pas transmettre d'argument au script.

L'argument du programme `soffice.exe` est une chaîne de caractères dont la structure générale est :

```
vnd.sun.star.script:alpha?language=beta&location=gamma
```

Respectez les majuscules et minuscules pour tous les caractères. Nous employons les mots `alpha`, `bêta`, `gamma` pour désigner des termes que nous allons expliciter.

- Le mot `bêta` est à remplacer par le nom du langage de script, exemples : `Java`, `JavaScript`, `BeanShell`, `Python`.
- Le mot `gamma` est à remplacer par le terme `user` si le script se trouve dans Mes macros, ou par le terme `share` si le script se trouve dans Macros OpenOffice.org.
- Le mot `alpha` est à remplacer par une séquence dépendant du langage.

Pour un script JavaScript ou BeanShell créé avec l'interface utilisateur, `alpha` se compose du nom de bibliothèque, d'un point, et du nom du fichier avec son extension.

```
HelloWorld.helloworld.bsh
```

Pour un script Python, `alpha` se compose du chemin vers le fichier python, en adressage relatif par rapport au répertoire `python/`, ensuite un caractère `$`, et enfin le nom de la fonction à appeler.

```
tata/exemple.py$maFonction
```

Quand un script est appelé par commande, il reçoit un argument. Cet argument n'a aucune utilité, mais il faut en tenir compte dans la déclaration de la fonction appelée (Python ou macro Java). Cet argument parasite n'a pas de conséquence sur les macros JavaScript ou BeanShell, car elles ne sont pas déclarées comme des fonctions.

Voici un exemple de ligne batch (`HelloJavaScript.bat`) appelant le script JavaScript `Hello World` qui se trouve dans Macros OpenOffice.org. Ouvrez un nouveau document Writer avant de lancer le batch car ce script écrit un texte dans le document Writer courant.

```
"C:\Program Files\OpenOffice.org 3\program\soffice.exe"
 ➥ "vnd.sun.star.script:HelloWorld.helloworld.js?
 ➥ language=JavaScript&location=share"
```

Basic peut lui aussi être lancé via le Scripting Framework, mais cela n'a pas d'intérêt, et implique une syntaxe encore différente.

Pour terminer sur ce point, signalons enfin que l'appel d'un script d'une extension installée utilise un argument plus complexe.

### Sous Linux

Le principe est le même que sous MS-Windows. Les chemins suivront évidemment la syntaxe Unix, les arguments d'un shell seront obtenus par `$1` etc.

Il faut ajouter dans le fichier shell une redirection vers un terminal X. Faites attention aux guillemets.

```
export DISPLAY=":0.0"
/usr/bin/soffice -headless "macro:///maBibli1.monModule.bavard("$1,
15")"
```

## Les extensions

### De quoi s'agit-il ?

Le concept d'extension remplace et améliore considérablement le concept d'add-on des versions anciennes d'OpenOffice.org tout en restant compatible avec lui. Une extension est un fichier qui permet d'étendre les fonctionnalités d'OpenOffice.org (par exemple l'export au format LaTeX). Le nom d'un fichier extension a maintenant pour extension `.oxt`, mais les anciennes versions utilisent `.zip` ou `.uno.pkg`.

Vous pouvez créer des extensions vous-même, ou profiter de celles qui sont disponibles sur un site web. Soyez prudent dans ce dernier cas : une extension peut potentiellement, comme toute macro, provoquer des dégâts, intentionnels ou non. En installant une extension, vous acceptez implicitement son exécution, quel que soit le niveau de sécurité que vous avez choisi. La communauté OpenOffice.org a créé un site pour regrouper les extensions disponibles, ce qui facilite leur évaluation et réduit les risques :

http://extensions.services.openoffice.org/

Vous y trouverez de très nombreuses extensions téléchargeables, la plupart gratuites, certaines payantes. Bien entendu, la qualité et l'utilité des extensions est très variable.

Une extension peut se limiter à installer un fichier de configuration. C'est le cas des dictionnaires écrits pour OpenOffice.org.

Une extension est souvent un moyen pour diffuser facilement une bibliothèque de scripts sur différents postes de travail. Elle peut simplement contenir une seule macro Basic, ou un ensemble de macros, mais ce n'est pas limitatif : tous les langages de script que nous avons vus peuvent être utilisés, ainsi que des bibliothèques de code objet compilé. Si nécessaire, une extension peut comporter des parties codées avec différents langages de programmation et plusieurs bibliothèques.

Une extension peut apporter un composant UNO, créé en Java, C++ ou Python. Ce composant peut être un nouveau service qui enrichit l'API, ou un add-in pour Calc, ou encore un add-in pour diagrammes.

En pratique, les extensions élaborées nécessitent de modifier l'interface utilisateur afin de les rendre plus conviviales. Aussi, une extension peut :

- ajouter une barre d'outils avec plusieurs boutons ayant des icônes spécifiques ;
- ajouter ou supprimer des boutons dans des barres d'outils existantes ;
- ajouter une entrée ou un sous-menu dans l'entrée Outils>Add-ons ;
- ajouter ou supprimer de nouvelles entrées ou même une arborescence de sous-menus dans les menus principaux (Fichier, Édition, Affichage, etc.) ;
- ajouter des pages qui seront intégrées au système d'aide OpenOffice.org ; la page correspondant au contexte étant appelée comme pour une fonctionnalité de l'application.

Ces ajouts et modifications peuvent être propres à chaque application d'OpenOffice.org concernée (Writer, Calc, Writer-HTML, etc). Par exemple, un bouton n'apparaît que pour Calc, un autre pour Writer, Calc, Draw et Impress.

Ajoutons que, pour permettre une diffusion internationale, tout le système d'extension est multilingue : il est possible de préparer tous les textes (menus, pages d'aide, etc.) pour plusieurs langues. S'ils sont disponibles, les textes apparaîtront automatiquement dans la langue de l'interface utilisateur.

D'autres mécanismes facilitent la gestion d'extensions d'envergure professionnelle :

- un identifiant unique, spécifique à l'extension (pour éviter les conflits de noms entre auteurs d'extensions) ;
- un numéro de version ;
- un nom « commercial » ;
- une icône symbolisant l'extension dans le gestionnaire des extensions ;
- un texte descriptif affiché par le gestionnaire des extensions ;
- un texte de la licence du produit, affiché à l'installation ;
- un contrôle d'adéquation entre l'extension et le poste utilisé (version d'OpenOffice.org, plate-forme matérielle) ;
- l'ajout de pages d'options spécifiques dans le menu Outils>Options ;
- un système de recherche de mise à jour vers une version plus récente de l'extension ;
- des liens vers le site de l'éditeur du logiciel ;
- un lien vers un site affichant les notes de livraison.

## Installer une extension

Quand une extension est installée sur un seul poste, les scripts sont visibles dans la section Mes Macros. Dans une installation OpenOffice en réseau, le responsable réseau a la possibilité d'installer une extension en mode partagé, accessible depuis tous les postes. Les scripts de l'extension apparaissent alors dans Macros OpenOffice.org. Sur un système multi-utilisateur comme Windows XP Home, depuis OpenOffice.org version 3.0 vous pouvez installer l'extension pour un utilisateur ou pour tous.

### Installation depuis OpenOffice.org

Le gestionnaire des extensions permet d'ajouter ou de supprimer très facilement des extensions. Vous y accédez par le menu Outils>Gestionnaire des extensions. Le panneau obtenu (figure 1-22) liste les extensions déjà installées. Celles comportant un petit cadenas sont installées pour tous les utilisateurs.

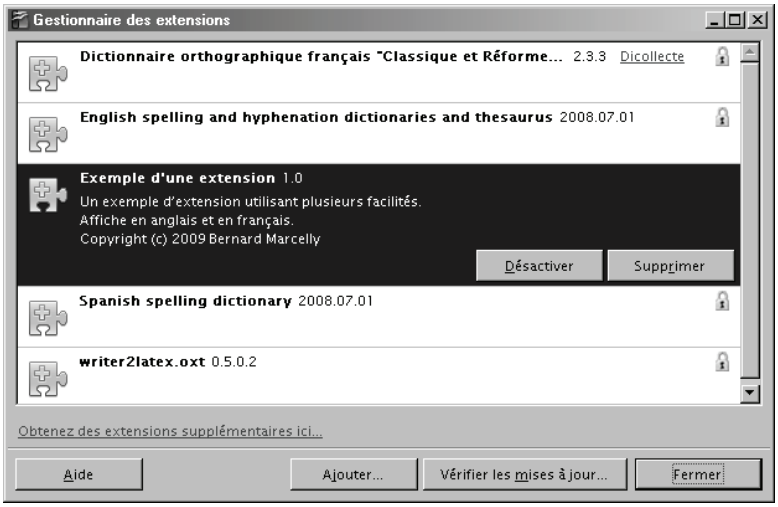



**Figure 1–22**
Le gestionnaire des extensions

Il suffit de cliquer sur le bouton Ajouter et de choisir le fichier extension. Le panneau qui apparaît (figure 1-23) vous donne le choix de l'installer pour un seul utilisateur ou pour tous les utilisateurs. Le même panneau sert à supprimer une extension ou à voir s'il existe des mises à jour pour les extensions installées.

À partir de la version 3.0, il est possible d'installer une extension en la lançant depuis un gestionnaire de fichiers, par exemple avec un double-clic.

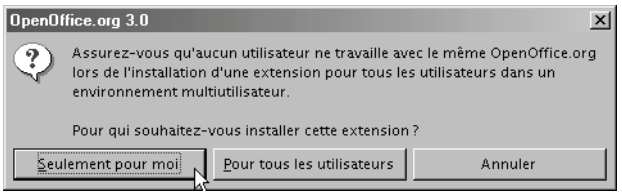



**Figure 1–23**
Installer une extension

### Installation depuis la ligne de commande

Le responsable informatique qui préfère l'installation en ligne de commande, lancera la commande `unopkg` qui se trouve dans le sous-répertoire `program/` du répertoire d'installation d'OpenOffice.org. Dans ces exemples des principaux modes de lancement nous n'avons pas écrit le chemin complet du fichier `monExtension.oxt`.

```
*** ajouter une extension pour un utilisateur
unopkg add monExtension.oxt
*** ajouter une extension pour tous les utilisateurs
unopkg add --shared monExtension.oxt
*** supprimer une extension à partir de son identifiant unique
unopkg remove xxx.yyyy.zzz.monExtension
unopkg remove --shared xxx.yyyy.zzz.monExtension
*** lancer unpkg en mode interactif
unopkg gui
*** interactif, en désignant le fichier (OOo 3.0 minimum)
unopkg gui monExtension.oxt
*** lister les options de unopkg
unopkg -h
```

Attention, pendant l'installation pour tous les utilisateurs le responsable informatique doit s'assurer que personne d'autre n'utilise OpenOffice.org.

## Concevoir une extension

Une extension se présente sous la forme d'une archive Zip qui regroupe différents fichiers. Réaliser une extension comportant toutes les possibilités que nous avons décrites représente parfois un effort plus important que le codage du script, en particulier pour une extension multilingue avec pages d'aide. Une telle extension comporte souvent des dizaines de fichiers. La plupart d'entre eux sont des fichiers XML dont les contenus sont codifiés selon des syntaxes précises.

Certains développeurs créent une extension en se basant sur une extension existante. Pour cela, il faut :

1. dézipper l'archive ;
2. modifier les fichiers XML manuellement avec un éditeur ;

3 ajouter et remplacer les fichiers propres à la nouvelle extension (bibliothèque Basic, fichiers images, etc) ;
4 et enfin zipper le dossier.

> **POUR LES EXPERTS Documentation officielle**
>
> Les extensions sont décrites techniquement dans le *Developer's Guide*, section *Extensions*. Ces pages, en anglais, sont disponibles à cette adresse :
>
> ‣ http://wiki.services.openoffice.org/wiki/Documentation/DevGuide/Extensions/Extensions

En pratique créer ainsi une extension est pénible et long si on veut profiter des possibilités existantes. Si vous ne connaissez pas bien la syntaxe des différents sous-fichiers de l'extension, l'échec est assuré. Nous présentons ici trois manières plus efficaces de créer une extension.

## Exporter une extension minimale

Un codage Basic réalisé dans une bibliothèque Basic autre que `Standard` est exportable en tant qu'extension simple, en cliquant sur un bouton dans le gestionnaire de macros Basic, comme sur la figure 1-24. Une fois l'export effectué, supprimez la bibliothèque Basic existante, puis installez l'extension, ce qui rétablira la bibliothèque.

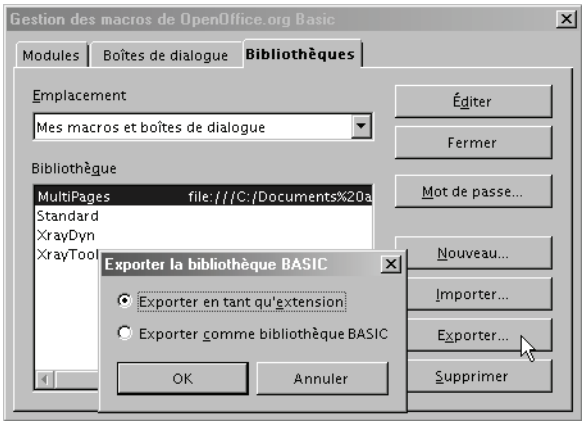



**Figure 1–24**
Créer une extension à partir d'une bibliothèque Basic

Ainsi, il est facile d'ajouter ou de supprimer une bibliothèque de macros pour différents utilisateurs. Ceux-ci pourront les employer dans leurs codages ou configurer manuellement leur poste pour les appeler d'un clic sur un bouton de barre d'outils, ou par un raccourci clavier. En contrepartie on n'utilise aucune des possibilités avancées décrites plus haut.

## L'outil BasicAddonBuilder

L'outil Open Source BasicAddonBuilder a été élaboré par Paolo Mantovani. Il sert à créer une extension à partir d'une bibliothèque de macros Basic, et à ajouter une barre d'outils et des menus. Un Assistant très bien conçu facilite sa mise en œuvre (voir les figures 1-25 et 1-26).

**Figure 1–25**
BasicAddonBuilder, étape 1

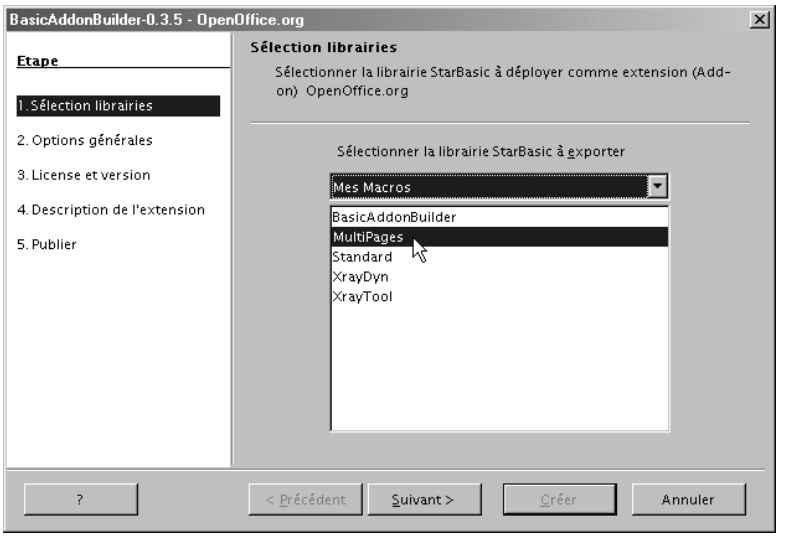
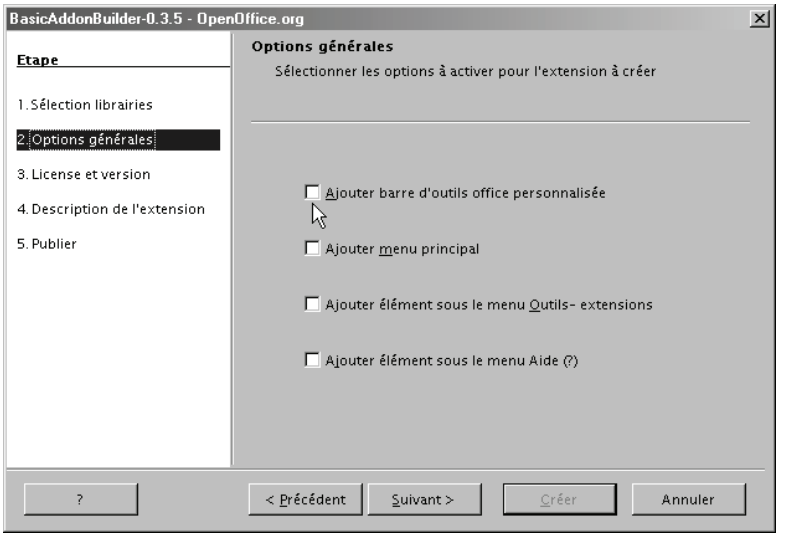



**Figure 1–26**
BasicAddonBuilder, étape 2



Pour la petite histoire, BasicAddonBuilder est une extension, réalisée en plusieurs langues, dont le français. Cet outil est idéal pour une extension comportant des macros Basic, une barre d'outils et quelques menus. L'inconvénient est que, dans la version actuelle, si vous voulez modifier quelque chose vous devez ré-exécuter toutes les étapes de l'Assistant sans vous tromper, ce qui peut devenir pénible.

Pour télécharger BasicAddonBuilder, rendez-vous à la page :

http://wiki.services.openoffice.org/wiki/Extensions_Packager

## L'outil Extension Compiler

Un des auteurs de ce livre a créé l'outil Open Source Extension Compiler, seulement disponible en anglais. Il permet de réaliser tous types d'extensions, pas seulement Basic, et supporte tous les mécanismes connus, ce qui implique en contrepartie un effort de compréhension plus important. Pour le télécharger, rendez-vous à l'adresse suivante :

http://wiki.services.openoffice.org/wiki/Extensions_Packager

L'outil se présente sous la forme d'un modèle de document Writer (extension `.ott`). En l'ouvrant par un double-clic, vous obtenez un nouveau document que vous allez personnaliser. Commencez par le sauver sous le nom de votre future extension, et dans un répertoire dédié. Dans ce répertoire (ou dans des sous-répertoires), vous devrez mettre toutes les bibliothèques que votre extension apporte (Basic, Python, etc), les fichiers images, et tous autres fichiers spécifiques qu'elle utilise. Ainsi, tous les éléments nécessaires à la création de l'extension se trouvent dans le répertoire dédié.

Ce document est à la fois un outil réalisé en Basic, la documentation de l'outil, et le support sur lequel vous décrivez votre extension. Vous allez écrire dans ce document :

- Une succession linéaire de macro-instructions (en fait des appels de sous-programmes Basic) dans une macro déjà préparée ; le document décrit la syntaxe d'utilisation de chaque macro-instruction.
- Si vous le souhaitez, un ou plusieurs textes destinés à être affichés par l'extension (descriptions, licences, pages d'aide), en une ou plusieurs langues pour chacun.

Voici à titre d'exemple le début des instructions Basic nécessaires pour compiler une extension multilingue.

```
beginDescription("org.bmarcelly.ExtExample", "1.0", "user")
  beginDependencies
    setOOoDependency("2.4", "OpenOffice.org 2.4.0 minimal")
  endDependencies
  setExtensionDescription("en, fr")
  setLicense("en, fr", "", "user", True)
  setDisplayName("en", "Example of an extension")
  setDisplayName("fr", "Exemple d'une extension")
```

```
  setHelp("en, fr")
endDescription

beginAnnexes
  useLibrary("Basic", "basTest4/")
endAnnexes

beginAddonUI
  beginAddonMenu
    beginMenu
      beginTitles("Writer")
        setTitle("Calling scripts", "en")
        setTitle("Appel de scripts", "fr")
      endTitles
      beginMenuItems
        beginCommand
          beginTitles
            setTitle("Basic : display time", "en")
            setTitle("Basic : afficher l'heure", "fr")
          endTitles
          setURL("Basic", "basTest4", "Module1", "WhatTimeIsIt")
        endCommand
        beginCommand
          beginTitles
            setTitle("Basic : Hello", "en")
            setTitle("Basic : Hello", "fr")
          endTitles
          setURL("Basic", "basTest4", "Module1", "HelloBas")
          setImage("Images/greenPage")
        endCommand
        addSeparator
rem --- etc... ---
```

La compilation s'effectue en lançant l'exécution du programme Basic. Tous les fichiers de configuration nécessaires sont créés, l'ensemble est zippé, et le fichier extension est obtenu après quelques secondes. S'il détecte une erreur grave, le compilateur fournit un message diagnostic et stoppe. Il suffit alors de corriger la ligne en cause et de relancer la compilation.

Le compilateur signale par un message d'avertissement les incompatibilités potentielles entre les options choisies et le numéro de version minimal d'OpenOffice.org accepté. En effet, comme les fonctionnalités liées aux extensions ont été introduites au fil des versions successives d'OpenOffice.org, il faut éviter qu'un utilisateur n'installe une extension sur une version incompatible.

Si vous décidez par la suite d'améliorer votre extension, il vous suffira de reprendre le document, d'y apporter les modifications nécessaires et de le recompiler pour obtenir la nouvelle version de l'extension.

# Conclusion

Après ce tour d'horizon des scripts dans OpenOffice.org, il est temps d'aborder Basic, le langage le plus couramment utilisé, ainsi que son environnement de développement intégré. Ceci nous permettra de voir le contenu d'une macro, de l'éditer et de l'exécuter.

Avant exécution, il vous faudra copier les fichiers nécessaires dans un répertoire de travail de votre ordinateur, là où, justement, leur exécution est autorisée.

## Remerciements

Nous remercions tous les intervenants français et étrangers dans les divers forums OpenOffice.org, soit qu'ils aient apporté leur pierre à la compréhension du produit, soit qu'ils nous aient incités à approfondir tel ou tel domaine grâce à la grande diversité de leurs questions.

Nous remercions tous ceux qui ont bien voulu publier leurs macros, car leur lecture est une source d'information inestimable autant pour les débutants que pour les programmeurs confirmés.

Enfin, merci à Sun, racheté par Oracle, pour avoir laissé en Open Source la suite OpenOffice.org, donnant ainsi à quiconque la possibilité d'étudier la structure d'un logiciel moderne et de très grande envergure. Nous espérons que cet esprit sera conservé.

Merci enfin aux développeurs d'OpenOffice.org pour leur souci constant de rendre automatisable et accessible par l'API tous leurs développements internes.