Mikaël Bidault

Couvre
Excel 2019
et
Office 365

Programmation EXCEL avec VBA

Compatible avec toutes les versions d'Excel

2e édition

Éditions EYROLLES

Programmation EXCEL avec VBA

2e édition

Maîtrisez la programmation VBA et tirez pleinement profit d'Excel. Cet ouvrage détaille les principes de la programmation orientée objet, le langage VBA et Visual Basic Editor, l'environnement de programmation d'Excel.

Intégrez pleinement vos programmes dans le ruban d'Excel. Vous apprendrez à créer des onglets personnalisés pour vos programmes VBA avec Custom UI Editor.

« **EXCELlez** ». Le dernier chapitre offre l'occasion de réviser l'ensemble des connaissances acquises lors de votre lecture, en développant un programme complet de génération de feuilles de paie Excel qui interagit avec Word.

Distribuez vos applications Excel de façon professionnelle. Créez un programme d'installation avec Inno Setup pour distribuer vos applications Excel simplement.

Compléments web

Tous les exemples des programmes du livre sont en téléchargement sur notre site Internet www.editions-eyrolles.com/dl/0067786.

À qui s'adresse cet ouvrage ?

- Aux utilisateurs d'Excel désireux d'améliorer leur productivité
- Aux développeurs qui souhaitent créer et distribuer des solutions Excel sûres et efficaces
- Aux personnes qui veulent s'initier à la programmation via le tableur de Microsoft

Au sommaire

Éditeur et développeur indépendant, **Mikaël Bidault** développe des compléments Word et Excel pour différentes sociétés. Il est le créateur de l'Articho, un complément VBA pour Word dédié à l'édition print et numérique (www.articho.eu).

www.editions-eyrolles.com

Programmation Excel avec VBA

2e édition

DANS LA MÊME COLLECTION

É. Sarrion. – **React.js.**
N° 67756, 2019, 358 pages.

R. Goetter. – **Grid Layout.**
N° 67683, 2019, 144 pages.

C. Blaess. – **Solutions temps réel sous Linux.**
N° 67711, 3e édition, 2019, 320 pages.

C. Pierre de Geyer, J. Pauli, P. Martin, E. Daspet. – **PHP 7 avancé.**
N° 67720, 2e édition, 2018, 736 pages.

H. Wickham, G. Grolemund. – **R pour les data sciences.**
N° 67571, 2018, 496 pages.

F. Provost, T. Fawcett. – **Data science pour l'entreprise.**
N° 67570, 2018, 370 pages.

J. Chokogoue. – **Maîtrisez l'utilisation des technologies Hadoop.**
N° 67478, 2018, 432 pages.

H. Ben Rebah, B. Mariat. – **API HTML 5 : maîtrisez le Web moderne !**
N° 67554, 2018, 294 pages.

W. McKinney. – **Analyse de données en Python.**
N° 14109, 2015, 488 pages.

E. Biernat, M. Lutz. – **Data science : fondamentaux et études de cas.**
N° 14243, 2015, 312 pages.

SUR LE MÊME THÈME

D.J. David. – **VBA pour Excel 2010, 2013 et 2016.**
N° 14457, 2016, 324 pages.

N. Barbary. – **Excel expert**.
N° 13692, 2e édition, 2014, 444 pages.

J.-M. Lagoda, F. Rosard. – **Réaliser des graphiques avec Excel**.
N° 56425, 2016, 128 pages.

Mikaël Bidault

Programmation Excel avec VBA

Compatible avec toutes les versions d'Excel

2e édition

Éditions EYROLLES

ÉDITIONS EYROLLES
61, bd Saint-Germain
75240 Paris Cedex 05
www.editions-eyrolles.com

Table des matières

CHAPITRE 2

CHAPITRE 3

CHAPITRE 7

CHAPITRE 8

CHAPITRE 9

CHAPITRE 10

Introduction

Visual Basic pour Applications, VBA, est la solution de programmation intégrée aux applications de la suite Office. Grâce à VBA, l'utilisateur d'Excel tire pleinement profit du tableur de Microsoft en en développant les fonctionnalités pour ses besoins spécifiques. Maîtriser VBA, c'est à coup sûr améliorer sa productivité.

L'intégration dans Excel de Visual Basic pour Applications, un *environnement de développement intégré* complet et professionnel, remonte à sa version 97. Office 2013, Office 2016 et Office 2019 intègrent la version 7.1 de Visual Basic, tandis qu'Office 2010 propose la version 7.0 et que XP, 2003 et 2007 fournissent Visual Basic 6.3. Entre ces versions, les différences sont quasi-inexistantes.

Cet ouvrage traite de la programmation des versions 97 à 2019 d'Excel. Sauf exceptions signalées, les explications et les exemples proposés sont valides pour toutes les versions d'Excel. En effet, des unes aux autres, il n'y a pas eu de révolution. Le modèle d'objets s'est affiné et les nouvelles fonctions d'Excel, apparues au cours des différentes versions du logiciel, peuvent également être manipulées via la programmation VBA. Cependant, le langage, la gestion des programmes, l'environnement et les outils au service du développeur – bref, tout ce que vous devez savoir pour programmer Excel et que cet ouvrage se propose de vous apprendre – restent inchangés d'une version à l'autre.

Donc, sachez que vous pourrez appliquer les connaissances acquises lors de la lecture de ce livre, aussi bien avec Excel 2003 sous Windows XP qu'avec la version 2016 et un système Windows 10. Mieux, les programmes développés pour Excel 97 fonctionnent avec toutes les versions ultérieures du tableur et, dans la très grande majorité des cas, les programmes développés dans Excel 2016 devraient fonctionner avec les versions antérieures.

Dans cet ouvrage, vous découvrirez les différentes méthodes de création de projets VBA pour Excel, Visual Basic (le langage de programmation proprement dit) et les outils de développement et de gestion intégrés de Visual Basic pour Applications. Votre initiation à la programmation VBA se fera au moyen d'exemples de programmes détaillés et commentés.

Définition

Vous rencontrerez le terme *projet* tout au long de cet ouvrage. C'est ainsi que l'on nomme un ensemble de programmes développés avec Visual Basic pour Applications.

Compléments VBA et compléments Office

Avec Office 2013, Microsoft a introduit un nouveau type de compléments, les compléments Office. Contrairement à ceux développés en VBA, les compléments Office ne sont pas installés sur l'ordinateur de l'utilisateur, mais hébergés sur un serveur distant à partir duquel ils s'exécutent. Ils sont développés à partir de technologies Web, telles que HTML 5, JavaScript, CSS 3, XML et des API REST.

En termes d'expérience utilisateur, les compléments Office s'apparentent à des applications mobiles auxquelles on s'abonne via l'Office store et s'exécutent systématiquement dans un panneau qui leur est dédié. Cependant, si vous souhaitez développer des solutions professionnelles dans le cadre d'une entreprise et non dans le but de les commercialiser via l'Office store, VBA reste presque toujours la solution la plus simple et la plus souple à mettre en œuvre et à déployer.

VBA, pour quoi faire ?

Excel offre des possibilités très étendues. Pourtant, quelle que soit la puissance de ses fonctions, elles ne peuvent répondre à toutes les situations. La programmation VBA est la solution de personnalisation offerte par Excel, afin d'ajouter des caractéristiques, des fonctions et des commandes qui répondent précisément à vos besoins.

La programmation VBA peut être définie comme la *personnalisation d'un logiciel afin de s'assurer gain de temps, qualité des documents et simplification des tâches complexes ou fastidieuses.* Voici quelques exemples de ce que permettent les programmes VBA :

- Combiner un nombre indéterminé de commandes. Nous sommes souvent amenés à répéter ou à associer certaines commandes plutôt que d'autres et à ignorer certaines fonctionnalités selon l'usage personnel que nous avons d'un logiciel. VBA permet d'associer un nombre illimité de commandes à une seule. Vous pouvez ainsi ouvrir simultanément plusieurs documents Excel stockés dans des dossiers ou sur des serveurs différents, y insérer des données spécifiques et leur appliquer des mises en forme adaptées, en exécutant une seule commande créée en VBA.
- Ajouter de nouvelles commandes et de nouvelles fonctions à Excel – par exemple, une fonction personnalisée qui calcule les taxes à retenir sur un salaire (ou, mieux, les primes à y ajouter), etc. Vous pouvez, en outre, attacher vos programmes VBA à des raccourcis clavier et à des commandes d'onglets afin d'en améliorer l'accessibilité.
- Automatiser des actions répétitives. Nous sommes parfois amenés à répéter certaines opérations plusieurs fois sur un même document ou à réitérer des traitements spécifiques. Un programme VBA peut, par exemple, mettre en forme des cellules dans un classeur Excel, effectuer des séries de calculs, etc.

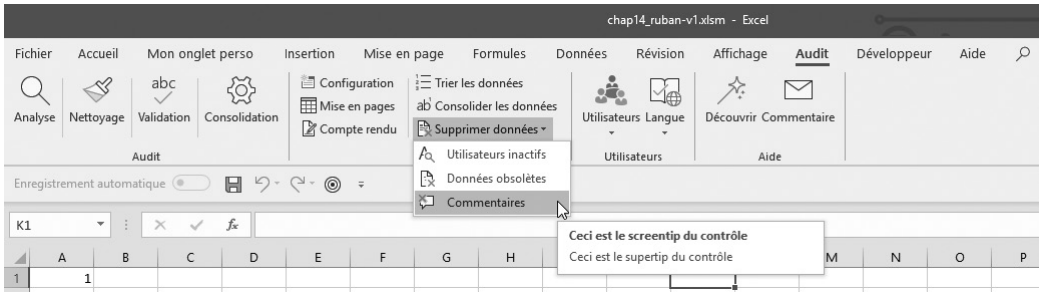

Figure 1 – *Vous pouvez affecter les macros VBA à des commandes sur des onglets personnalisés.*

- Modifier et améliorer les commandes d'une application. Les commandes Excel ne sont pas toujours adaptées à nos besoins ou présentent parfois des limitations gênantes. Un programme VBA peut modifier, brider ou compléter les commandes d'une application. Vous pouvez ainsi intégrer dans un tableau le nom de l'utilisateur, le nombre de pages imprimées et l'imprimante utilisée chaque fois qu'une impression est lancée à partir d'Excel.
- Faire interagir les différentes applications Office. Un programme VBA sait exploiter des données issues de fichiers générés par d'autres programmes et interagir avec ceux-ci de façon transparente pour l'utilisateur. Vous pouvez ainsi créer une commande qui envoie automatiquement le classeur Excel ouvert en fichier joint dans un courriel Outlook à des destinataires définis ou qui génère un rapport Word à partir de données Excel et l'imprime.
- Créer des interfaces personnalisées. Les programmes VBA peuvent ramener des tâches complexes à la simple information de champs dans des boîtes de dialogue personnalisées pour l'utilisateur final, simplifiant ainsi considérablement le travail de celui-ci, tout en vous assurant qu'aucun oubli ou fausse manipulation n'aura lieu.

Visual Basic pour Applications permet le développement de solutions adaptées à vos besoins. Les outils que vous apprendrez à manier vous permettront de développer des programmes simples, sans écrire la moindre ligne de code, comme des programmes complets intégrant une interface utilisateur adaptée.

La fonction d'un programme VBA peut être d'automatiser une tâche répétitive. Cependant, vous pouvez aussi créer très vite un petit programme VBA pour faire face à une nécessité immédiate ; par exemple, afin de généraliser un traitement exceptionnel à l'ensemble d'un document.

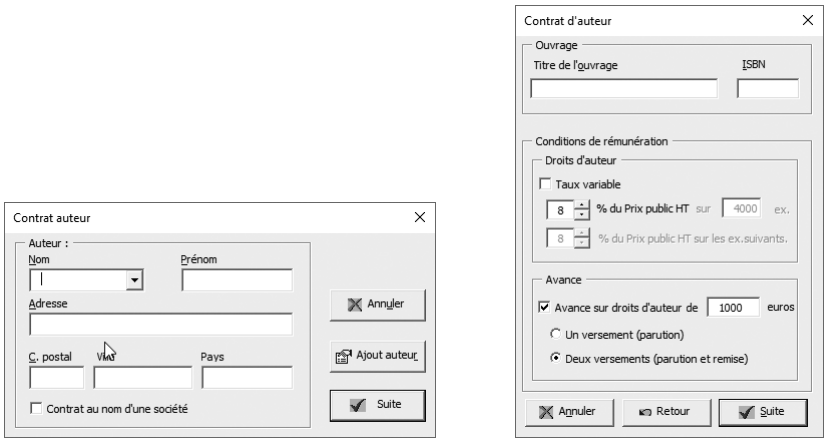

Figure 2 – *Visual Basic pour Applications vous permet de développer des interfaces utilisateur évoluées.*

Des programmes

Les projets *VBA* sont des programmes ou *macros* écrits dans le langage Visual Basic. Si vous ne possédez aucune expérience préalable en programmation, ne vous inquiétez pas : cet ouvrage aborde le développement de projets VBA à travers l'enregistrement de macros. Lorsque vous l'activez, l'Enregistreur de macro mémorise chacune de vos actions. C'est votre programmeur personnel : vous utilisez simplement les commandes d'Excel et il se charge de traduire les actions exécutées en instructions Visual Basic. Il vous suffit ensuite d'exécuter la macro pour répéter l'ensemble des commandes enregistrées.

Définition

Le terme macro désigne le regroupement d'un ensemble de commandes en une seule. On parle parfois de macrocommande pour désigner un programme qui se résume à l'exécution d'une série de commandes, sans égard pour le contexte. Des macros plus évoluées peuvent répéter des opérations en boucle ou afficher des boîtes de dialogue qui autorisent une interaction avec l'utilisateur. Ces programmes se comporteront différemment en fonction des informations entrées ou de l'état du document sur lequel elles s'exécutent.

Le terme projet est plus large. Il désigne l'ensemble des éléments constituant vos programmes VBA. Il s'agit toujours de macros, mais à celles-ci peuvent s'ajouter des feuilles – qui constituent une interface utilisateur permettant de récolter des informations de tout type –, des modules de classe et autres friandises que vous découvrirez tout au long de cet ouvrage.

L'enregistrement de macros constitue sans aucun doute le meilleur moyen de se familiariser avec la programmation en Visual Basic. Ainsi, sans connaître le langage – les instructions qui le composent et la façon dont elles sont structurées –, vous pouvez créer des programmes VBA et en visualiser ensuite le code.

Une application hôte et des projets

Visual Basic pour Applications est un environnement de développement calqué sur Visual Basic, une solution de développement d'applications Windows. Les structures de contrôle du langage sont les mêmes, et l'environnement proprement dit (Visual Basic Editor) est pour ainsi dire identique à celui de Visual Basic. Cependant, contrairement à Visual Basic, Visual Basic pour Applications est conçu… *pour des applications*. Cela signifie que, tandis que les programmes Visual Basic sont autonomes, les programmes VBA ne peuvent être exécutés qu'à partir d'une application intégrant cet environnement de développement – Excel ou une autre application.

Lorsque vous développez un programme VBA, vous l'attachez à une application. Il s'agit de l'*application hôte* du programme. Plus précisément, vos programmes VBA sont attachés à un document (un fichier ou un modèle Word, une feuille de calcul Excel, une présentation PowerPoint…) spécifique à l'application hôte. L'ensemble des programmes VBA attachés à un document constitue un projet. Un projet regroupe des macros, mais peut également intégrer des interfaces utilisateur, des déclarations système, etc. Un projet constitue en fait la partie VBA d'un document. Si cet ouvrage ne traite que de la programmation pour Excel, sachez qu'un programme VBA peut être attaché à une autre application. Les concepts et les outils que vous découvrirez au long de cet ouvrage sont valides pour toutes les applications de la suite Office. Pour exécuter une macro VBA, vous devez avoir accès au document auquel elle est attachée. Vous pouvez choisir de rendre certaines macros disponibles à partir de n'importe quel document Excel ou en limiter l'accessibilité à un classeur Excel spécifique. La disponibilité des programmes VBA est abordée au chapitre 2.

Un langage de programmation

Les projets VBA sont développés dans le langage de programmation Visual Basic. Vous découvrirez par la pratique la structure de ce langage et apprendrez rapidement à en discerner les composants et les relations qu'ils entretiennent. Comme nous l'avons dit précédemment, l'enregistrement de macros constitue une excellente initiation à Visual Basic. C'est sous cet angle que nous vous ferons découvrir ce langage.

Visual Basic est un langage de *programmation orientée objet* (POO). Nous présenterons donc les concepts de ce type de programmation. Vous apprendrez ce qu'on appelle un objet, une propriété, une méthode ou un module de classe. Vous verrez comment conjuguer ces éléments pour créer des applications Excel souples et puissantes. Visual Basic pour Applications constitue une bonne approche de la programmation pour le néophyte.

VBA intègre un grand nombre d'instructions, grâce auxquelles vous développerez des macros qui identifient très précisément l'état de l'application et des documents et reproduisent l'exécution de la plupart des commandes disponibles dans l'application hôte.

Vous verrez que certaines instructions sont spécifiques à Excel, par exemple celles qui affectent une formule à une cellule. Vous n'utiliserez probablement qu'un nombre limité de ces instructions, en fonction de votre usage personnel d'Excel ou des besoins de votre entreprise. Cependant, certaines apparaîtront presque toujours dans vos macros. C'est par exemple le cas de la propriété `Range`, qui renvoie un objet Excel tel qu'une cellule ou une plage de cellules.

D'autres instructions sont communes à l'ensemble des applications Office, notamment celles qui règlent le comportement d'une macro : réaliser des opérations en boucle, induire des réactions face à certains paramètres, afficher des boîtes de dialogue simples (figures 3 et 4) ou développer des interfaces utilisateur évoluées (figure 1), etc. Ce sont ces instructions qui constituent véritablement ce qu'il est convenu d'appeler *le langage Visual Basic*. Vous aurez besoin d'y faire appel dès que vous voudrez créer un programme interactif, capable de se comporter différemment selon le contexte. Pour la plupart, ces instructions ne peuvent être générées par enregistrement de macros et doivent donc être éditées manuellement dans Visual Basic Editor.

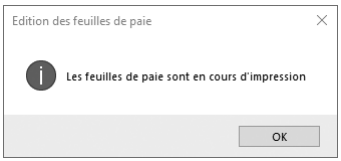

Figure 3 – *La fonction VBA MsgBox affiche une boîte de dialogue.*

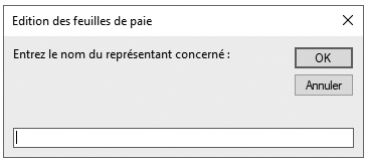

Figure 4 – *Il existe une version VBA et une version Excel de la fonction InputBox.*

Cet ouvrage ne se veut pas un dictionnaire du langage, mais un guide qui vous enseignera le développement de projets VBA de qualité. Vous apprendrez à enregistrer, modifier, exécuter et déboguer des macros, à créer des interfaces utilisateur ainsi qu'à gérer vos projets VBA. Vous découvrirez, à travers les exemples de cet ouvrage, un certain nombre d'instructions spécifiques à la *hiérarchie d'objets* d'Excel, qui vous familiariseront avec la logique de ce langage.

Définition

La hiérarchie d'objets d'une application, encore appelée modèle d'objets, est le rapport qu'entretiennent entre eux les différents objets d'une application. Ce concept ainsi que les notions spécifiques aux langages orientés objet seront développés au chapitre 1, « Notions fondamentales de la programmation orientée objet ».

Ce livre présente et illustre d'exemples commentés l'ensemble des structures de contrôle qui servent à créer très simplement des macros évoluées. Nous vous fournirons les bases du langage Visual Basic. Elles suffisent pour créer une infinité de macros et répondre à vos besoins spécifiques.

Lorsque les principes du développement de projets VBA vous seront acquis et que vous créerez vos propres macros, il vous arrivera sûrement d'avoir besoin d'instructions que vous n'aurez pas rencontrées lors de la lecture de cet ouvrage ; vous pourrez alors utiliser l'Enregistreur de macro ou encore les rechercher dans l'aide de Visual Basic pour Applications ou dans l'Explorateur d'objets – étudié au chapitre 4. Vous verrez que l'aide de VBA fournit une référence complète du langage, facilement accessible et consultable.

Si vous n'avez aucune expérience de programmation, peut-être ce *Visual Basic* vous apparaît-il comme un langage barbare ou inaccessible. Ne vous inquiétez pas : le développement de projets VBA ne requiert ni expérience préalable de la programmation, ni connaissance globale du langage. Contentez-vous, au cours de votre lecture, d'utiliser les fonctions nécessaires aux exercices et que nous vous détaillerons. Cet ouvrage propose un apprentissage progressif et concret : vous développerez vos premiers projets VBA dès les premiers chapitres.

Un environnement de travail

VBA dispose d'un environnement de développement à part entière : Visual Basic Editor.

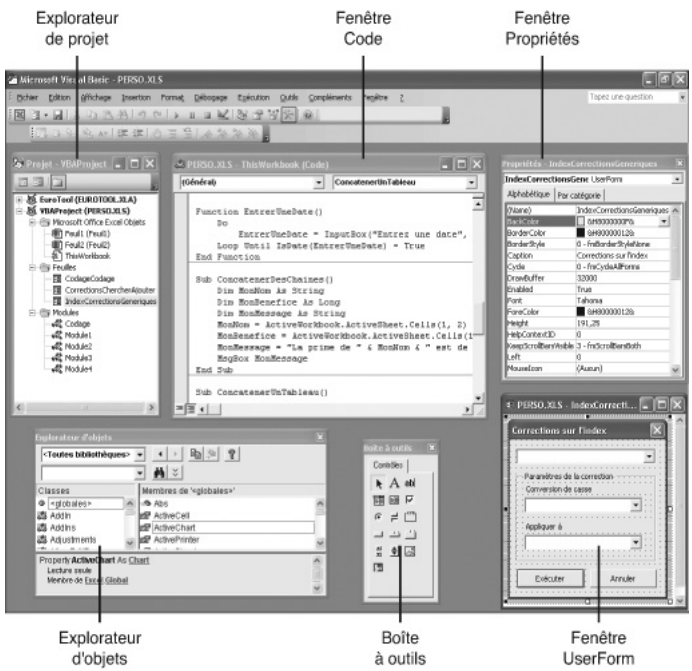

Figure 5 – *Visual Basic Editor est l'environnement de développement de Visual Basic pour Applications.*

Visual Basic Editor est l'*environnement de développement intégré* des applications Office. Il permet de visualiser et de gérer les projets VBA, d'écrire, de modifier et de déboguer les macros existantes, de visualiser comment les commandes propres à une application Office sont traduites en langage Visual Basic et inversement. C'est un outil de débogage de vos projets VBA d'une grande efficacité. Il propose nombre d'outils pour tester les macros et en étudier le comportement. Vous pouvez ainsi exécuter les commandes de la macro pas à pas, en suivre le déroulement, insérer des commentaires dans le texte de la macro, etc. Enfin, cet environnement intègre des outils très intuitifs, dédiés au développement d'interfaces graphiques.

Vous apprendrez dans cet ouvrage à utiliser les nombreux outils de Visual Basic Editor à toutes les phases de développement d'un projet VBA.

Conventions typographiques

Afin de faciliter la lecture, nous avons adopté dans cet ouvrage un certain nombre de conventions typographiques. Lorsqu'un mot apparaît pour la première fois, il est composé en *italique*. Les programmes et les mots-clés du langage Visual Basic apparaissent dans `une police à chasse fixe`. Lorsque, dans un programme, un mot signale une information attendue dans le code, celui-ci apparaît en *`italique`*.

Lorsqu'une ligne de code ne peut être inscrite sur une seule ligne de l'ouvrage, cette flèche (➥) en début de ligne indique que le texte est la suite de la ligne précédente.

Par ailleurs, vous rencontrerez au long de cet ouvrage différents types de notes, matérialisées par des encadrés.

Info

Ces rubriques apportent un complément d'information en rapport avec le sujet traité. Leur lecture n'est pas indispensable, mais elles vous aideront à mieux cerner le sujet.

Définition

Vous trouverez sous ces rubriques la définition de termes techniques spécifiques à la programmation VBA.

Attention

Ces rubriques vous mettent en garde contre les risques inhérents à telle ou telle commande ou manipulation.

Rappel

Il est parfois nécessaire de se rafraîchir la mémoire. Lorsqu'un sujet fait appel à des connaissances acquises plusieurs chapitres auparavant, cette rubrique vous les remémore brièvement.

Astuce

Sous cette rubrique, vous trouverez des trucs pour aller plus vite et travailler plus efficacement.

Conseil

Nous vous faisons ici part de notre expérience, en vous prodiguant des conseils qui vous aideront à développer des projets VBA de qualité.

Codes sources des exemples du livre

Les exemples du livre sont proposés en téléchargement sur le site des éditions Eyrolles. Vous pouvez ainsi tester tous les exemples à partir des fichiers Excel qui les intègrent. Cela vous évitera également de saisir le code dans Visual Basic Editor. Avant de poursuivre la lecture de ce livre, téléchargez les exemples à l'adresse suivante : http://www.editions-eyrolles.com/dl/0067786.

Première partie

Découvrir la programmation Excel

1

Notions fondamentales de la programmation orientée objet (POO)

Visual Basic est un langage de programmation *orienté objet*. En tant que tel, il repose sur des concepts communs à tous les langages de POO. Avant de vous lancer dans la programmation pour Excel, il est important de vous familiariser avec ces concepts et le vocabulaire qui les décrit. Plus concrètement, ce chapitre vous fera découvrir les différents composants de Visual Basic en tant que langage orienté objet et comment ils s'articulent pour créer des programmes VBA puissants.

Vous ne trouverez pas dans ce chapitre de programmes VBA. Il est destiné à vous donner les bases et la terminologie sur lesquelles nous nous appuierons tout au long de cet ouvrage. Alors, patience ! Les connaissances qu'il vous apportera permettront d'appréhender vos premiers programmes dès le chapitre 2.

Comprendre le concept d'objet

Comme pour tous les langages de POO, les *objets* sont le fondement de Visual Basic. Quelle que soit la fonction d'un programme VBA, presque toutes les actions qu'il exécute s'apparentent à la modification d'objets.

Les ouvrages présentant la POO le font presque toujours par analogie avec les objets de la vie réelle. Nous ne dérogerons pas à cette règle. La programmation orientée objet repose en effet sur une structure qui rappelle, par de nombreux points, les objets de la vie courante et les rapports qu'ils entretiennent. Cette analogie rend simples et faciles d'accès des concepts qui, abordés de façon abstraite, vous apparaîtraient probablement obscurs.

Objets et collections d'objets

Dans la vie, un objet peut être tout et n'importe quoi. Ce qui caractérise un objet, c'est son existence physique, ses propriétés spécifiques, son comportement et les actions que l'on peut exécuter sur celui-ci. Une voiture est un objet. Lorsque vous parlez de l'objet `Voiture`, vous pouvez faire référence à un objet abstrait (« Je vais acheter une voiture ») comme à une voiture bien concrète (« Regarde un peu ma belle 504 verte »). Les objets que vous utiliserez dans vos programmes VBA répondent à une même définition.

Dans le premier cas, vous évoquez un objet `Voiture` imprécis et pourtant tout le monde comprend de quoi vous parlez. Il vous suffit de prononcer le mot « voiture » pour que chacun imagine et visualise un véhicule bien spécifique, en fonction de ses goûts, de ses aspirations, de ses souvenirs, etc. Cependant, en tant qu'objet `Voiture`, elle possède un certain nombre de *propriétés* (une carrosserie, des roues, un moteur) et autorise un certain nombre de *méthodes* (démarrer, freiner, tourner) qui permettent d'en maîtriser le comportement.

Ce sont ces *propriétés* et ces *méthodes*, communes à toutes les voitures, qui définissent l'objet `Voiture`. Elles sont sous-entendues, évidentes et essentielles. Il existe donc des milliers de voitures différentes, toutes reconnaissables par un certain nombre de caractéristiques communes définies dans le concept (l'objet) `Voiture`. En POO, cet objet abstrait est appelé la *classe* `Voitures` et est la définition formelle des objets `Voiture` (leurs propriétés et leurs méthodes). Il s'agit du modèle à partir duquel vous pouvez imaginer et créer des milliers de voitures différentes. L'ensemble des véhicules appartenant à la classe `Voitures` (parce qu'ils possèdent les propriétés et les méthodes définies dans cette classe) est appelé la *collection d'objets* `Voitures`.

Info

Une collection porte le nom pluriel des objets qu'elle rassemble.

Ainsi, la collection `WorkBooks` renvoie tous les objets `Workbook`, soit tous les classeurs ouverts, la collection `Sheets`, toutes les feuilles d'un objet `WorkBook`, la propriété `Worksheets`, toutes les feuilles de calcul d'un objet `Workbook`, etc. La section « Le modèle d'objets d'Excel » située en fin de chapitre vous fera découvrir les objets Excel les plus importants.

Définition

Le terme Classe désigne la définition commune d'un ensemble d'objets (qu'est-ce qu'une voiture ?), tandis qu'une Collection désigne l'ensemble des objets appartenant à une classe (toutes les voitures en circulation).

Lorsque vous parlez d'acheter la Peugeot 504 verte de vos rêves, vous évoquez une voiture concrète, bien spécifique. Vous créez une *instance* – on parle aussi d'une *occurrence* – de l'objet `Voiture`. Elle possède toutes les propriétés de la classe `Voitures`, mais ces propriétés sont attachées à des valeurs précises. La carrosserie est verte, la vitesse maximale est de *x* km/h, etc. Vous pouvez maîtriser le comportement de votre voiture à l'aide des méthodes définies dans la classe `Voitures` (`Accélérer`, `Freiner`), mais l'effet précis de ces méthodes est étroitement lié aux propriétés de votre véhicule. La puissance du moteur ne permet pas d'atteindre 200 km/h (mais

vous pouvez décapoter !) ; les freins ne sont pas équipés du système ABS, il faut donc telle distance pour freiner, etc.

Un programme VBA peut ainsi créer une feuille de calcul Excel en appliquant la méthode `Add` (ajouter) à la collection `WorkBooks` et déterminer les propriétés de ce classeur (son nom, ses options de protection, le nombre des feuilles qui le composent, etc.)

Info

Lorsque vous créez une instance, cet objet possède toutes les propriétés et méthodes définies dans la classe. Ce principe essentiel de la programmation orientée objet est appelé instanciation.

Le grand intérêt de la programmation orientée objet, c'est qu'il n'est pas indispensable de savoir comment fonctionne un objet pour l'utiliser. Lorsque vous achetez une voiture, vous n'avez pas besoin de savoir comment la carrosserie et le moteur ont été fabriqués, ni comment les différents composants sont assemblés ; vous vous contentez de choisir un modèle, une couleur, etc. Il vous suffit de connaître les méthodes propres à la classe `Voitures` pour l'utiliser. Avec VBA, lorsque vous créez une instance d'un objet, vous en définissez les propriétés sans vous préoccuper de la façon dont celles-ci seront appliquées. Il en va de même pour les méthodes que vous utilisez pour maîtriser le comportement d'un objet. Lorsque vous tournez la clé de contact, le moteur de la voiture démarre, sans que vous ayez à vous soucier du détail des événements et des technologies mises en œuvre.

VBA permet, par exemple, de créer des interfaces graphiques pour vos programmes, en déposant simplement les objets dont vous avez besoin (cases à cocher, zones de texte, boutons de commandes), sur une feuille. Ces objets ont des comportements spécifiques que votre programme exploitera, sans que vous ayez besoin de vous soucier de leur mécanisme interne.

Application hôte et modèles d'objets

Lorsque vous développerez des programmes VBA, vous agirez sur des objets qui varieront en fonction des actions que vous souhaitez que votre programme exécute. Vous définirez et associerez ces objets de façon à créer une application complète. Là encore, l'analogie avec les objets de la vie courante est révélatrice. Les objets que nous utilisons sont généralement ordonnés selon leurs fonctions. Lorsque vous souhaitez vous laver, vous vous dirigez vers la salle de bains ; il s'agit du lieu consacré à la toilette. Vous y trouvez un certain nombre d'objets tels que savon, gant de toilette, dentifrice, brosse à dents, etc. Vous utilisez le savon avec le gant de toilette, le dentifrice avec la brosse à dents, et vous pouvez faire une toilette complète.

Si vous souhaitez manger, c'est dans la cuisine que vous vous orienterez. Vous y trouverez quelques objets disponibles dans la salle de bains (savon, robinet, placard). Vous ne devriez cependant pas y trouver de brosse à dents, ni aucun des objets spécifiques à la toilette. En revanche, vous pourrez utiliser le four, ouvrir le réfrigérateur et utiliser tous les objets spécifiques de la cuisine.

Les applications du Pack Office sont comparables aux pièces de votre maison. Lorsque vous choisissez de développer un projet VBA, vous choisissez une *application hôte*. Il s'agit de l'application Office qui contient les objets sur lesquels vous souhaitez agir. C'est dans cette dernière que vous

développerez vos programmes, et c'est uniquement à partir de cette application qu'ils pourront être exécutés. Si vous souhaitez travailler sur des textes, vous choisirez d'entrer dans Word. Pour faire des calculs, vous savez que c'est dans Excel que vous trouverez les objets dont vous avez besoin. Access sert au développement et au maniement des bases de données, et PowerPoint à la création de présentations.

Cependant, à l'image des pièces de votre maison, les applications Office ne sont pas hermétiques. Vous pouvez parfaitement vous préparer un plateau repas dans la cuisine et choisir de manger au lit. De façon semblable, des projets VBA évolués sont capables d'utiliser des objets de différentes applications Office. Un programme développé dans Excel peut utiliser des données stockées dans une base de données Access ou des objets Word pour imprimer un courrier qui accompagnera une facture, et envoyer un message Outlook de confirmation.

Vous devez choisir une application hôte pour votre projet. Deux critères doivent la déterminer :

- Votre programme sera plus performant et plus simple à développer si l'application hôte est celle dans laquelle s'exécute l'essentiel des instructions du programme.
- La présence du programme dans l'application hôte doit être logique, et l'utilisateur final doit y accéder facilement puisque le programme ne pourra être exécuté qu'à partir de celle-ci.

Info

Tous les projets développés dans cet ouvrage seront hébergés dans Excel. Pour accéder aux objets d'une application autre que l'hôte, vous utiliserez la technologie Automation. L'accès aux objets d'une autre application est traité au chapitre 6.

L'application est donc la pièce dans laquelle votre programme s'exécutera. Elle est composée d'un certain nombre d'objets – constituant une *bibliothèque* – dont les rapports sont précisément définis. Les objets d'une application et les rapports qu'ils entretiennent sont représentés sous la forme d'un organigramme. Tout en haut de l'organigramme se trouve l'application (la pièce dans laquelle sont rangés tous les objets). Viennent ensuite les classes d'objets de premier niveau de l'application, auxquelles sont liés d'autres objets ou classes, et ainsi de suite. On appelle cette structure le *modèle d'objets* ou la *hiérarchie de classes* de l'application. La figure 1-1 représente ce qui pourrait être un modèle d'objets sommaire de l'application Salle de bains.

Info

Pour la plupart, les éléments d'Excel peuvent être manipulés dans Visual Basic pour Applications en tant qu'objets. Un classeur, une feuille de ce classeur, une cellule ou une boîte de dialogue Rechercher sont autant d'objets manipulables dans un programme Visual Basic.

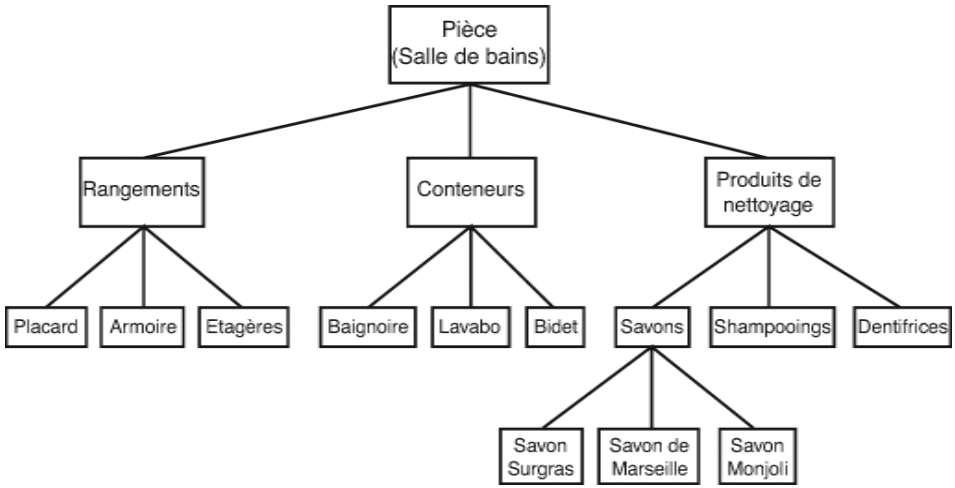

Figure 1-1 – *L'ensemble des objets d'une application est structuré selon un modèle d'objets qui en définit les rapports et la hiérarchie.*

Au sommet du modèle se trouve la pièce – l'application. Tous les objets auxquels vous pouvez accéder y sont contenus. Si l'on établit un modèle d'objets pour l'ensemble des pièces de la maison, on retrouvera toujours l'objet `Pièce` au sommet du modèle. De la même façon, au sommet des modèles d'objets des applications Office, se trouve l'objet `Application`.

Viennent ensuite les classes situées immédiatement sous l'objet `Pièce`. Plus on progresse dans le modèle, plus les objets sont précis et donc spécifiques de la pièce ou de l'application. Dans Excel par exemple, sous l'objet `Application` se trouve la collection (ou classe) `Workbooks` qui englobe tous les objets `Workbook`, c'est-à-dire tous les classeurs Excel ouverts. Sous l'objet `Workbook` se trouve la classe `Worksheets`, qui englobe tous les objets `Worksheet` (toutes les feuilles de calcul) de l'objet `Workbook` désigné.

Astuce

Pour accéder à l'aide en ligne des objets Excel, affichez l'Aide de VBA à partir du menu « ? », puis sélectionnez la commande Référence VBA d'Excel. Vous apprendrez à accéder à Visual Basic Editor au prochain chapitre.

Notez que l'appartenance des objets à des branches distinctes du modèle ne signifie pas qu'ils ne peuvent pas interagir. L'objet `Savon de Marseille` peut se trouver sur l'étagère et vous pouvez utiliser la méthode `Déplacer` pour le mettre dans l'objet `Baignoire`, comme dans l'objet `Lavabo`.

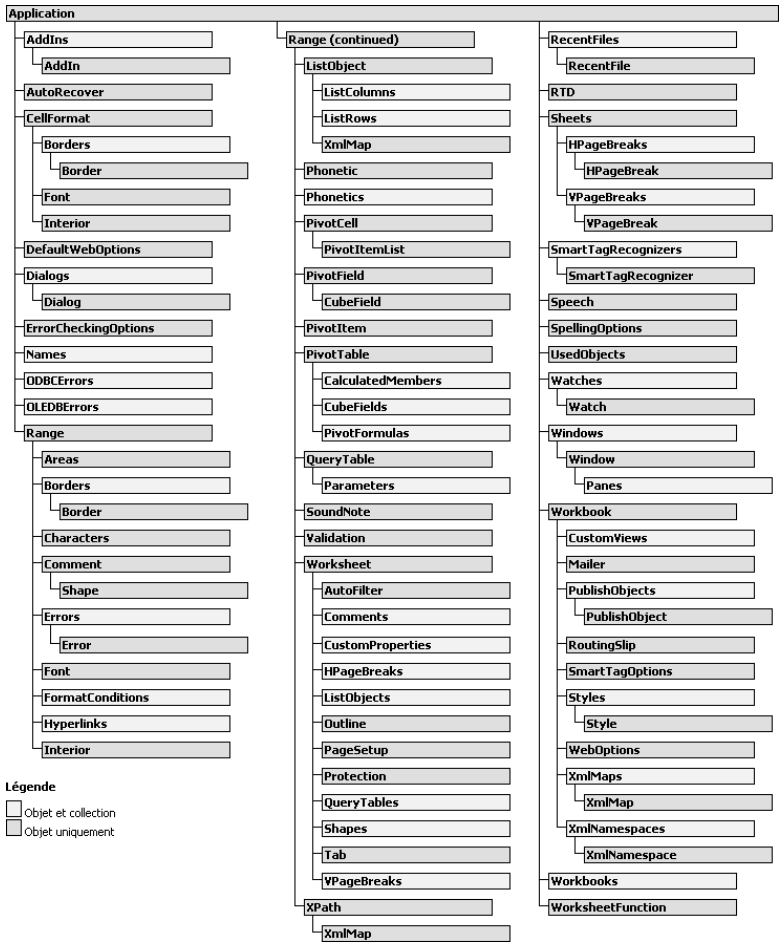

Figure 1-2 – *Le modèle d'objets d'Excel.*

Un objet peut en englober d'autres ; il est alors qualifié de *conteneur*. C'est le cas de l'objet `Application`, mais c'est aussi vrai pour beaucoup d'autres objets du modèle d'Excel. Par exemple, un `Workbook` contient des `Worksheet` (feuilles de calcul), contenant eux-mêmes des `Range` (cellules et plages de cellules).

Accéder aux objets

Le modèle détermine le chemin à emprunter pour accéder à un objet. Pour vous laver les dents, vous devez d'abord accéder à votre brosse à dents. Même si le processus est inconscient, vous identifiez l'objet `Brosse à dents` par son emplacement : il est situé dans la salle de bains, parmi les

objets et produits de toilette. De la même façon, en Visual Basic, vous devez identifier un objet avant de pouvoir agir dessus (appliquer l'une de ses méthodes ou modifier la valeur de l'une de ses propriétés). Lorsque vous souhaitez vous laver les dents, vous pensez et suivez inconsciemment les étapes suivantes :

- aller à la salle de bains ;
- se diriger vers les produits de toilette ;
- choisir parmi ceux-ci la brosse à dents et le dentifrice et s'en saisir.

Pour accéder à un objet Excel, vous opérerez selon le même mode, c'est-à-dire en partant du haut de la hiérarchie et en progressant dans celle-ci jusqu'à atteindre l'objet voulu.

Le point est utilisé comme séparateur entre les différentes collections et objets que l'on rencontre avant d'atteindre l'objet voulu. La référence à un objet précis d'une collection se fait selon la syntaxe suivante :

```
Nom_Collection("Nom_Objet")
```

Le code VBA permettant d'accéder à l'objet `Dentifrice` serait :

```
Piece.ProduitsHygiene("Dentifrice").Prendre
```

La première partie du code permet d'accéder à l'objet `Dentifrice` ; l'expression identifiant un objet est appelée *référentiel d'objet*. La méthode `Prendre` est ensuite appliquée à cet objet afin de s'en saisir.

Le code Visual Basic activant la feuille de classeur Excel, nommée MaFeuille et située dans le classeur MonClasseur.xlsm (à condition que celui-ci soit ouvert), serait :

```
Application.Workbooks("MonClasseur.xlsm").Sheets("MaFeuille").Activate
```

On accède à l'objet `Workbook` MonClasseur de la collection `Workbooks` (tous les classeurs ouverts), puis à la feuille nommée MaFeuille de la collection `Sheets` (toutes les feuilles de l'objet MonClasseur). Une fois le chemin d'accès à l'objet indiqué, on lui applique la méthode `Activate`.

Info

Outre son nom, chaque objet est identifié par une valeur d'indice représentant sa position dans la collection. Cette valeur peut être utilisée pour renvoyer un objet précis selon la syntaxe suivante :

```
Nom_Collection(IndexObjet)
```

où *IndexObjet* représente la position de l'objet dans la collection. L'instruction suivante :

```
Workbooks(2).Activate
```

active le classeur Excel apparaissant en deuxième position dans la liste des fenêtres du bouton Changer de fenêtre de l'onglet Affichage.