Vk

Zeste de savoir

Réutiliser du code en VBA sur Excel

vendredi 23 août 2024

Table des matières

Introduction

Il est souvent inutile de réinventer la roue.

Lorsque c'est possible, reprendre des éléments (code générique, formulaire, feuille de classeur, …) ou installer une bibliothèque nous fera gagner un temps précieux en création comme en maintenance.

À travers ce billet, nous allons nous pencher sur des moyens de réutiliser du code.

1. Exports et imports

Pour commencer, nous pouvons réutiliser des fichiers sources ainsi que des feuilles.

C'est une solution intéressante pour reprendre de petits éléments d'un projet à un autre.

1.1. Exporter et importer des fichiers sources

Il est donc possible d'exporter et d'importer des fichiers sources, que ce soit un *UserForm* (extension `.frm`), un module de classe (extension `.cls`) ou tout simplement un module (extension `.bas`).

1.1.1. Exporter

Pour exporter, il faut sélectionner l'élément dans l'arborescence du projet et cliquer sur "Exporter un fichier…" depuis le menu contextuel ou depuis le menu "Fichier", ou encore via le raccourci CTRL + E.

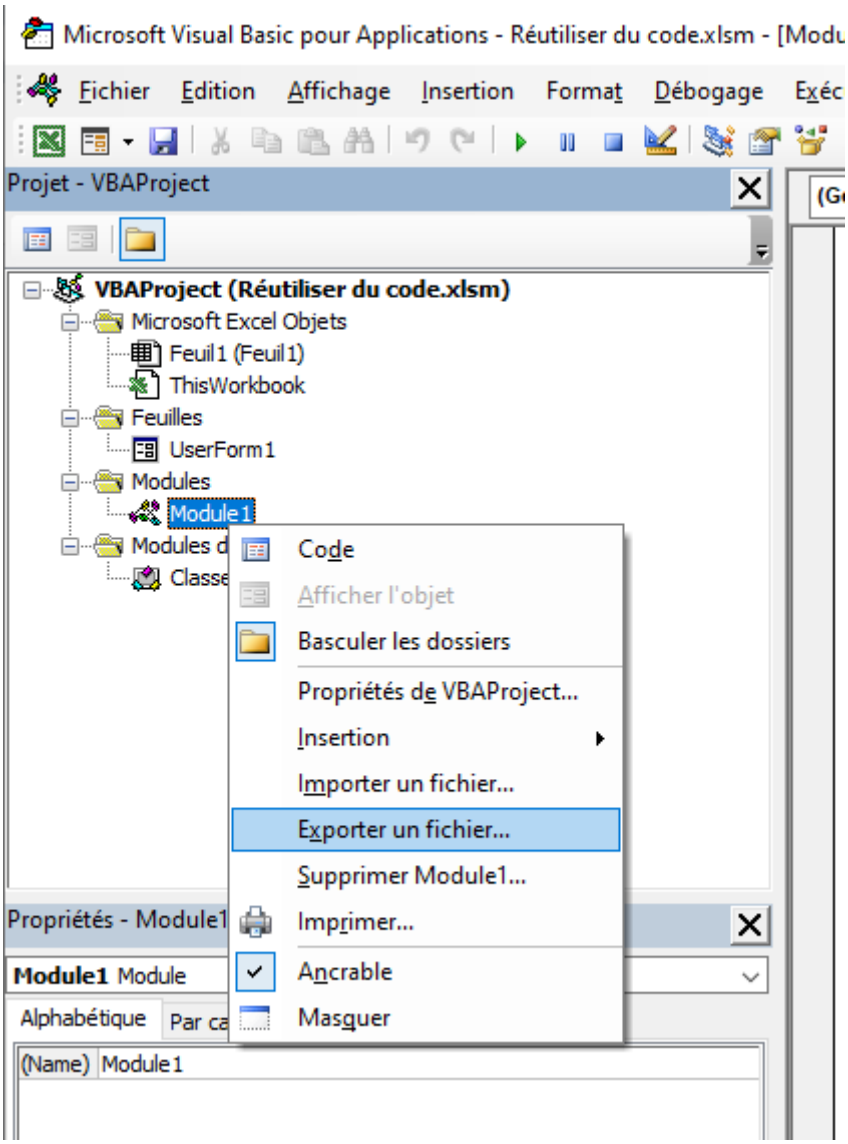

FIGURE 1.1. – Exporter le fichier depuis le menu contextuel (CTRL + E).

Nous pouvons ensuite choisir où et sous quel nom stocker le fichier.

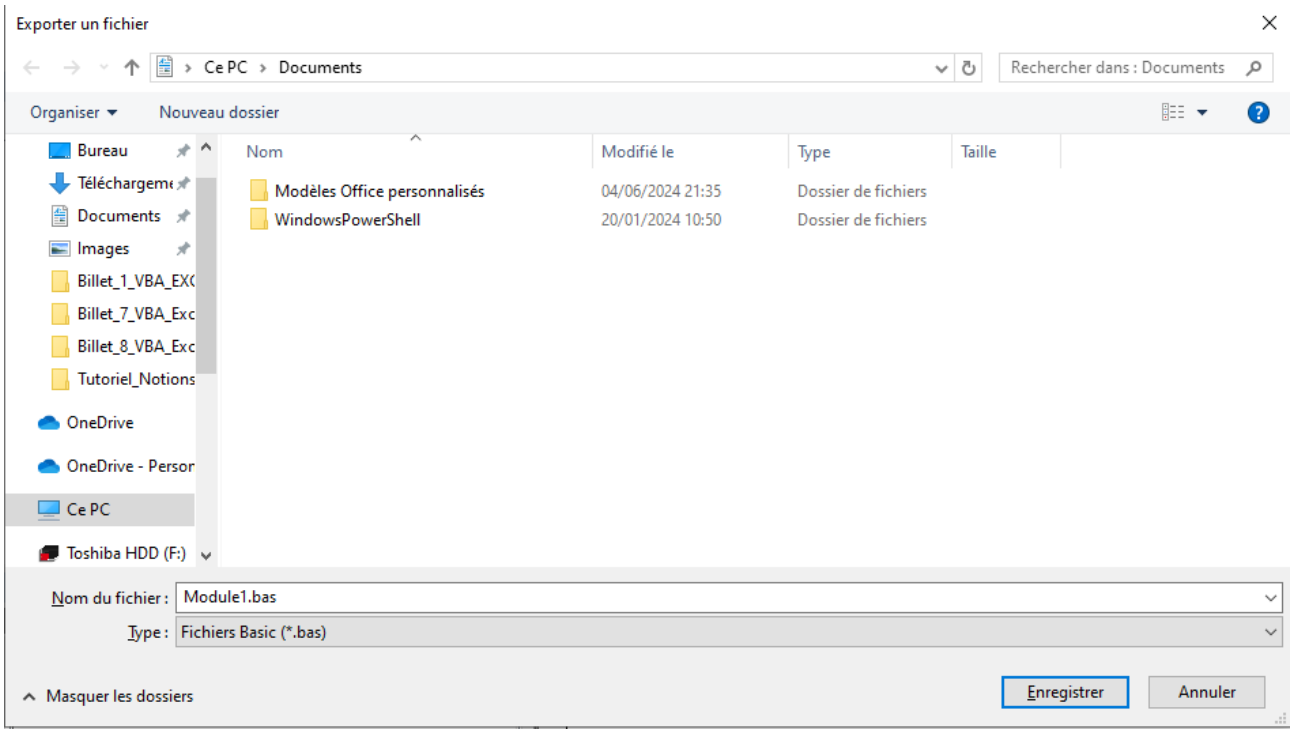

FIGURE 1.2. – Écriture du fichier à exporter.

1.1.2. Importer

Pour importer, il faut cliquer sur "Importer un fichier..." depuis le menu contextuel ou depuis le menu "Fichier", ou encore via le raccourci CTRL + M.

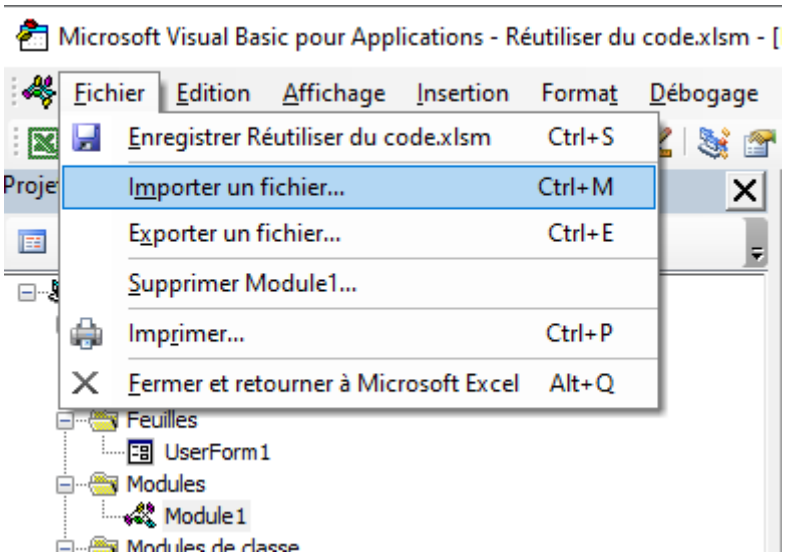

FIGURE 1.3. – Importer le fichier depuis le menu Fichier (CTRL + M).

Un explorateur de fichiers s'ouvre alors, nous permettant de sélectionner le fichier souhaité.

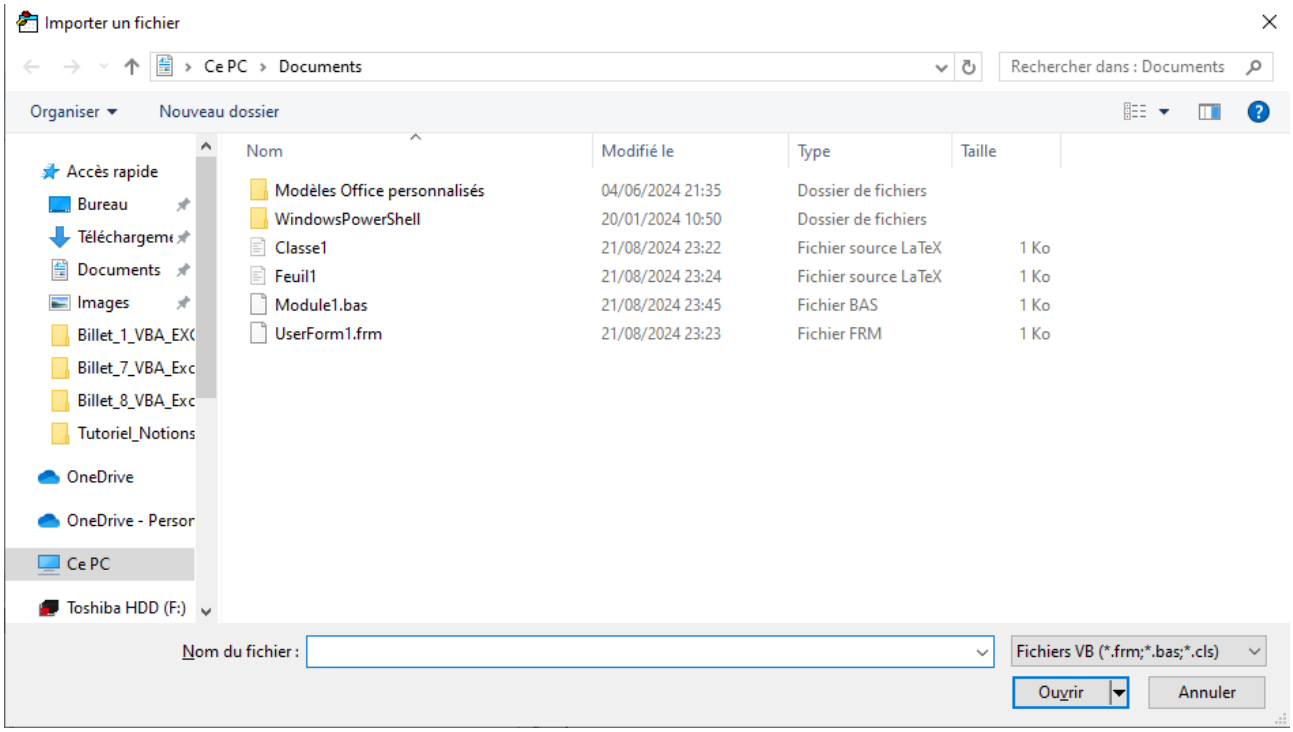

FIGURE 1.4. – Sélection du fichier à importer.

Nous pouvons constater que le fichier à importer a été ajouté dans l'arborescence du projet.

i

Dans le cas où le projet contient déjà un fichier source du type et du nom indiqué, VBE numérotera automatiquement le fichier importé de manière à éviter un conflit de nommage.

1.2. Exporter et importer des feuilles

Le cas des feuilles est un peu particulier et ne fonctionne pas avec la méthodologie vue ci-dessus.

Pour cet exemple, nous partirons d'une feuille de calcul contenant le code suivant :

```
1  Option Explicit
2
3  Private Sub Worksheet_Activate()
4      MsgBox "Bonjour !"
5  End Sub
6
7  Private Sub Worksheet_Deactivate()
8      MsgBox "Au revoir !"
9  End Sub
```

Pour réutiliser cette feuille, il faut la déplacer (en veillant à cocher la case "Copie" dans ce cas) dans le classeur souhaité.

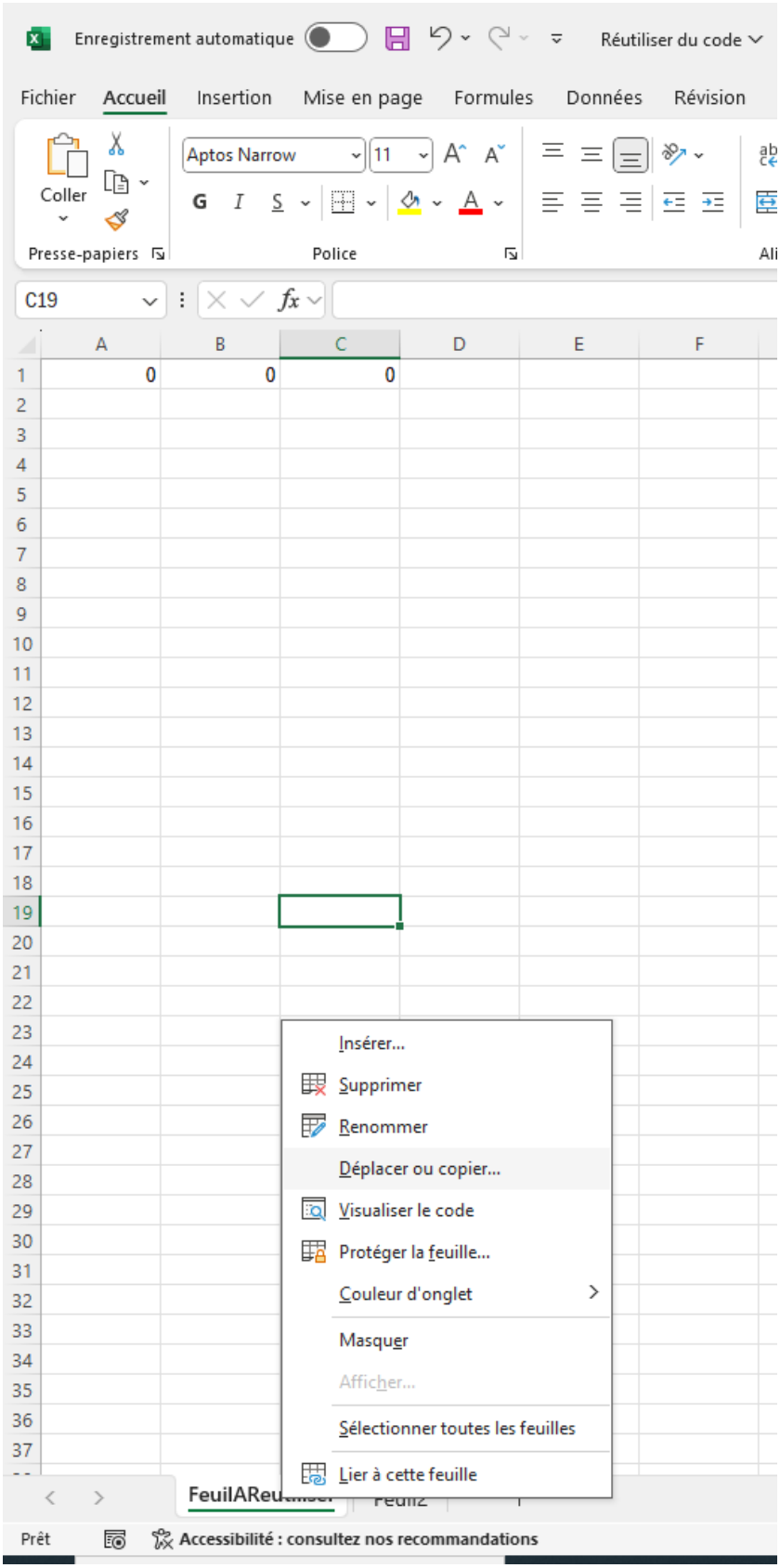

FIGURE 1.5. – Choix "Déplacer ou copier..." pour la feuille voulue.

Il faut ensuite choisir la destination (cadre orange ci-dessus) et choisir l'option de copie (cadre rouge ci-dessous) afin d'éviter de retirer la feuille du classeur d'origine.

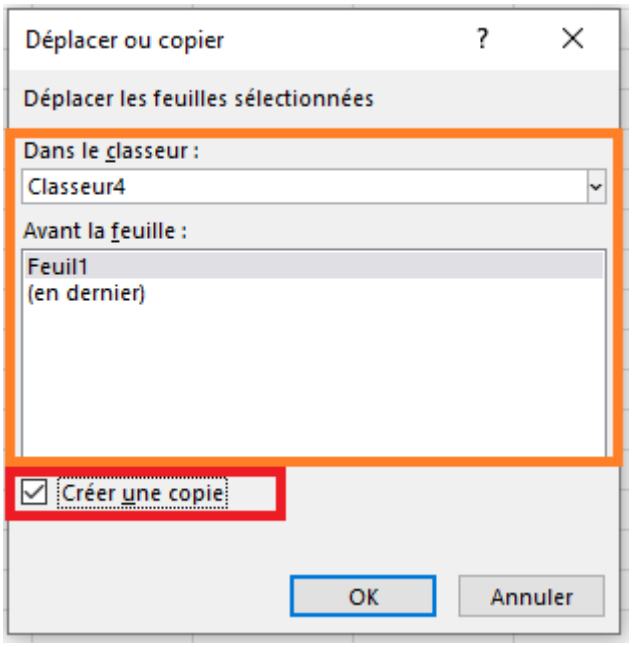

FIGURE 1.6. – Fenêtre déplacer ou copier.

Cela fait, nous pouvons vérifier que l'onglet est bien présent dans notre nouveau classeur et que le code fonctionne.

> Dans le cas où le projet contient déjà une feuille du nom indiqué, la feuille sera automatiquement numérotée pour éviter un conflit de nommage.

À travers cette première section, nous avons vu comment exporter et importer des fichiers VBA et comment copier des feuilles.

2. Bibliothèques

Les bibliothèques sont un autre moyen de réutiliser du code.

Elles servent à étendre un projet en installant des fonctionnalités.

2.1. Références du projet VBA

Nous avons eu l'occasion de parler de la fenêtre des références au fur et à mesure de ces pages, en particulier pour les dictionnaires en *liaison anticipée*. Pour rappel, cette fenêtre est accessible via le menu "Outils" du VBE.

Durant nos projets, nos recherches peuvent nous mener à devoir installer telle ou telle référence. C'est donc ici que ça se passe !

Lors du partage d'un classeur avec macros, les références cochées ne sont pas toujours disponibles pour certaines versions d'Excel. Il y a donc parfois une adaptation à faire pour le destinataire (cocher une référence plus ancienne par exemple).

2.2. Macros complémentaires Excel

C'est une solution intéressante pour des éléments plus conséquents ou pour persister. Par contre, il n'est pas possible de reprendre des feuilles de cette manière.

Il existe différentes natures de complément (Excel, COM, ...), mais nous nous concentrerons sur celles d'Excel.

2.2.1. Ajouter et retirer une macro complémentaire

Pour ajouter et retirer des compléments Excel, nous devons passer par la fenêtre adéquate. Cela se fait soit par le bouton "Compléments Excel" du ruban "Développeur", soit par le bouton "Atteindre" du menu "Compléments" dans les options du fichier.

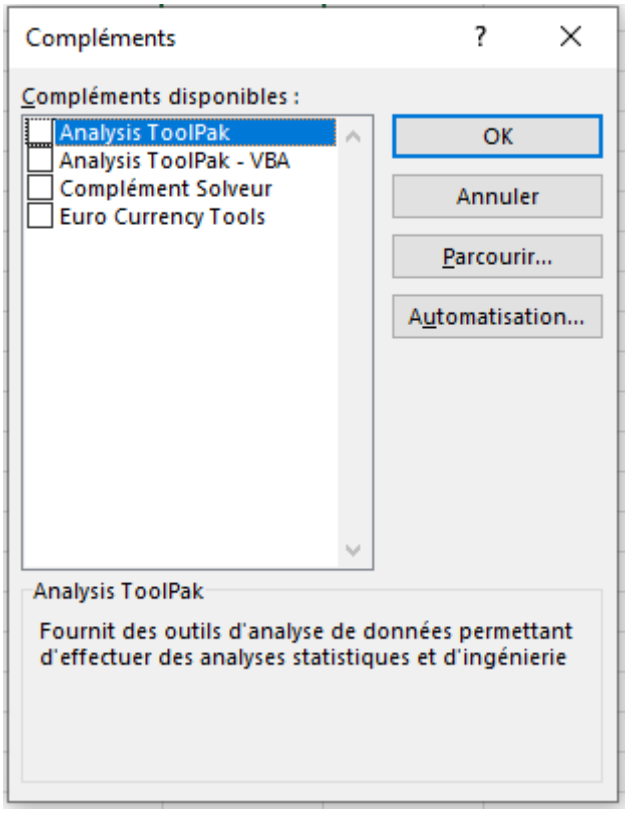

FIGURE 2.7. – Fenêtre compléments.

L'installation d'un complément peut s'accompagner d'une modification de l'interface Excel (ruban, menu contextuel, ...), permettant son usage.

2.2.2. Créer une macro complémentaire

Pour créer une macro complémentaire, il faut suivre différentes étapes.

2.2.2.1. Créer un nouveau classeur

Nous commençons en créant un nouveau classeur correspondant à notre macro complémentaire.

2.2.2.2. Développer le contenu de la macro complémentaire

Ensuite, nous devons développer nos fonctionnalités.

Pour cet exemple, j'ai ajouté un `UserForm` dont j'ai changé la propriété `Caption` par "Macro complémentaire" ainsi que la propriété `Name` par "UserForm_MacroComplementaire"

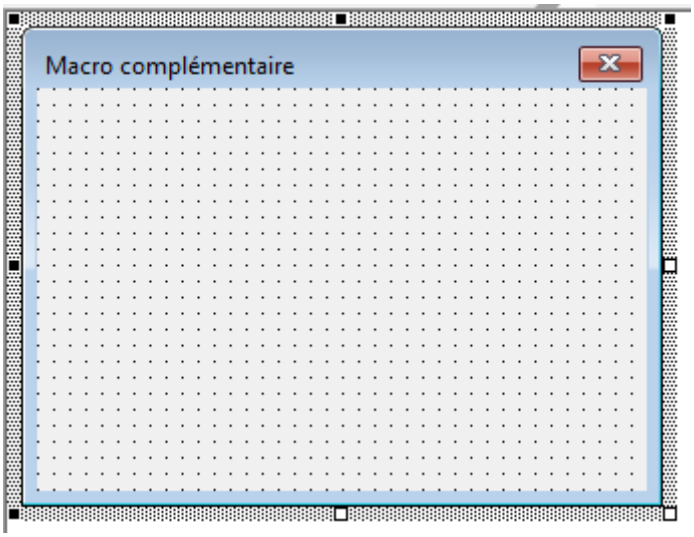

FIGURE 2.8. – UserForm titré pour la macro complémentaire.

Ensuite, j'ai ajouté un peu de code :

Module MacroComplementaire

```
1  Option Explicit
2
3  Public Function Operation(ByVal dNombre1 As Double, ByVal dNombre2
       As Double) As Double
4      Operation = dNombre1 + dNombre2
5  End Function
6
7  Public Function Ouvre_UserForm_MacroComplementaire()
8      UserForm_MacroComplementaire.Show
9  End Function
```

ThisWorkbook

```vba
Option Explicit

Private Sub Workbook_AddinInstall()
    MsgBox "Merci d'avoir ajouté l'AddIn"
End Sub

Private Sub Workbook_AddinUninstall()
    MsgBox "À une prochaine !"
End Sub
```

2.2.2.3. Protéger le code VBA (optionnel)

Cette étape est optionnelle, mais selon les cas nous pouvons vouloir protéger le code de la macro complémentaire avec un mot de passe robuste.

Il suffit de passer par l'onglet "Protection" de la fenêtre "Propriétés du projet" (menu "Outils" du VBE) pour cela.

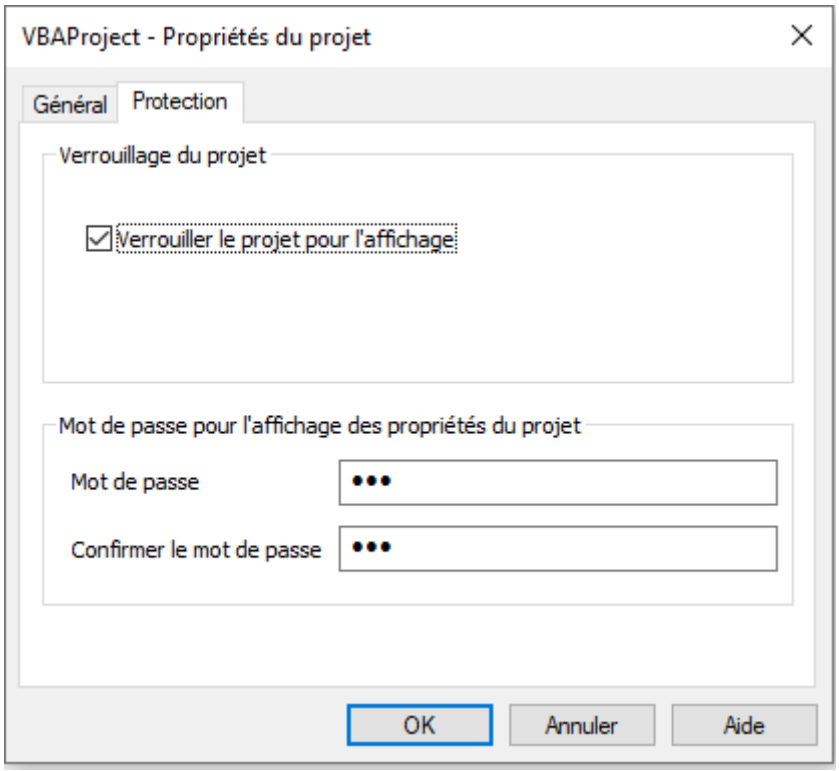

FIGURE 2.9. – Verrouillage du projet VBA.

2.2.2.4. Renseigner les informations du document

Maintenant, nous allons renseigner un titre et une description pour notre macro complémentaire.

Depuis Excel, il faut cliquer sur "Fichier" puis "Informations" et lister toutes les informations possibles en cliquant sur "Afficher toutes les propriétés".

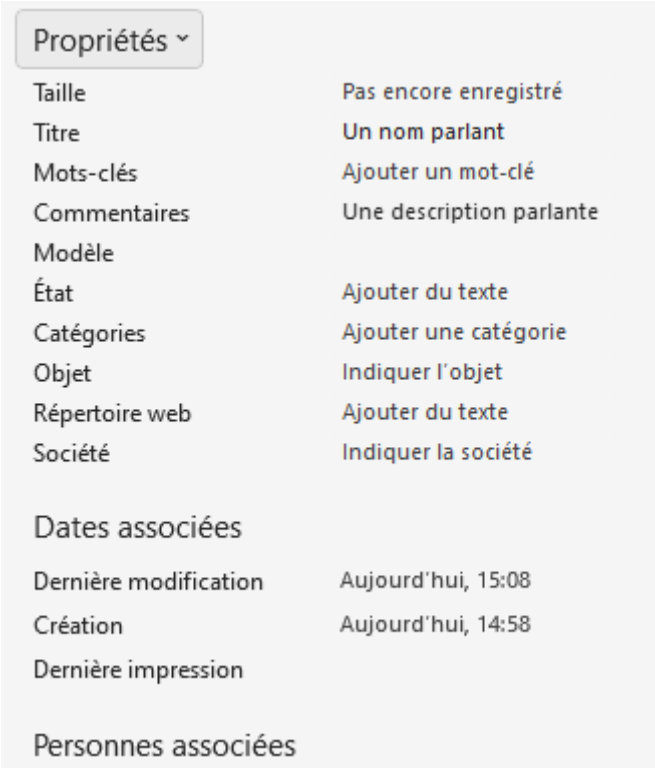

FIGURE 2.10. – Renseigner un titre et une description dans les informations du fichier.

Le titre se met dans la propriété `Titre` et la description dans la propriété `Commentaires`.

2.2.2.5. Mettre la propriété IsAddin à True

Une fois le code écrit et validé, nous pouvons passer la propriété `IsAddin` de l'objet Workbook à `True` depuis la fenêtre des propriétés.

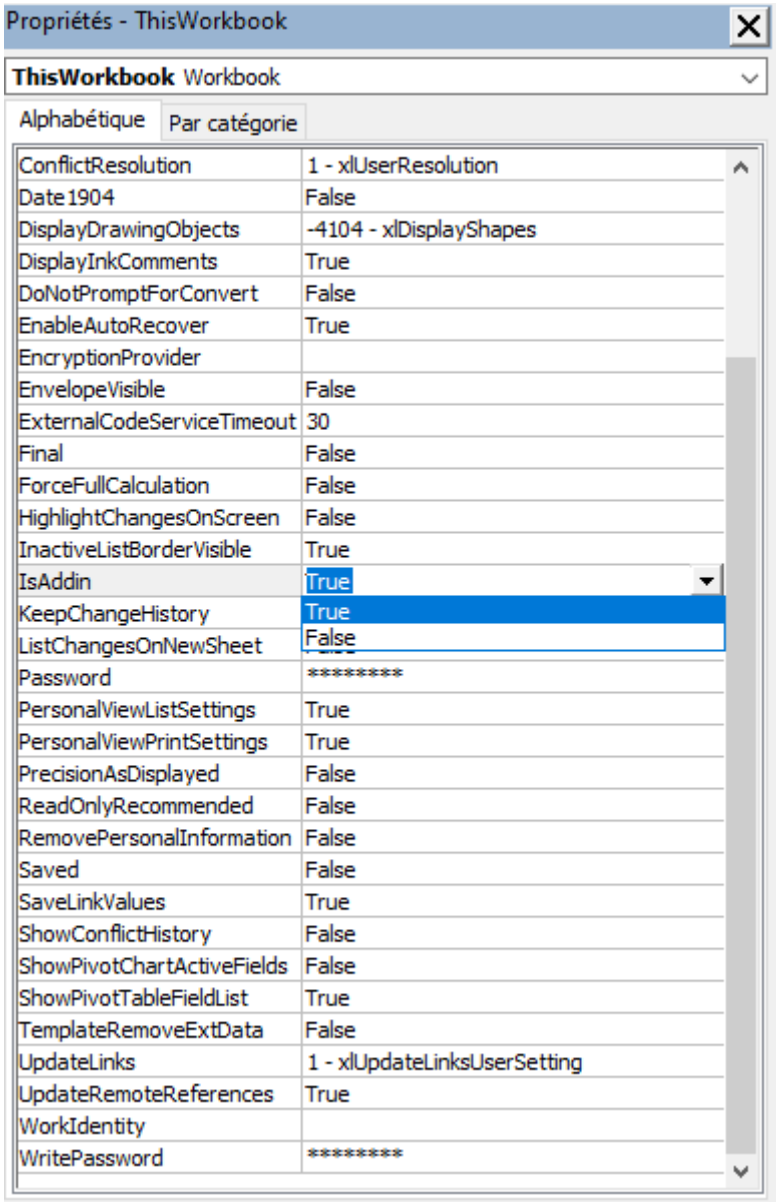

FIGURE 2.11. – Propriété IsAddin pour l'objet Workbook.

Cela va permettre de démarquer le classeur de type macro complémentaire des autres. Il fera alors parti de la collection `AddIns` et non plus de `Workbooks`.

En conséquence, nous ne voyons plus les feuilles de ce classeur et il n'est plus possible d'accéder aux informations du classeur, d'où la nécessité de faire cette étape à la fin.

2.2.2.6. Enregistrer le résultat via le VBE avec l'extension `.xlam`

Il est temps d'enregistrer le résultat final via le VBE avec le type "Complément Excel" (extension `.xlam`). L'explorateur de fichiers nous propose par défaut un emplacement dédié aux compléments.

Nous pouvons observer la spécificité de l'icône.

FIGURE 2.12. – Icône macro complémentaire.

J'ai placé cette étape à la fin ici, mais il est préférable d'enregistrer régulièrement (avant de basculer le classeur en macro complémentaire) au risque de perdre son travail malencontreusement.

Une fois le résultat enregistré, nous sommes en mesure de l'ajouter dans nos classeurs comme étudié précédemment. Remarquez le déclenchement de l'événement lorsque nous ajoutons et retirons notre macro complémentaire.

Si nous effectuons des modifications dans notre macro complémentaire, celles seront alors répercutées. Par exemple, si nous changeons la somme en soustraction dans notre fonction `Operation`, le résultat sera modifié en relançant la fonction. Autre exemple, si nous modifions notre `UserForm`, nous aurons bien accès à la nouvelle version :

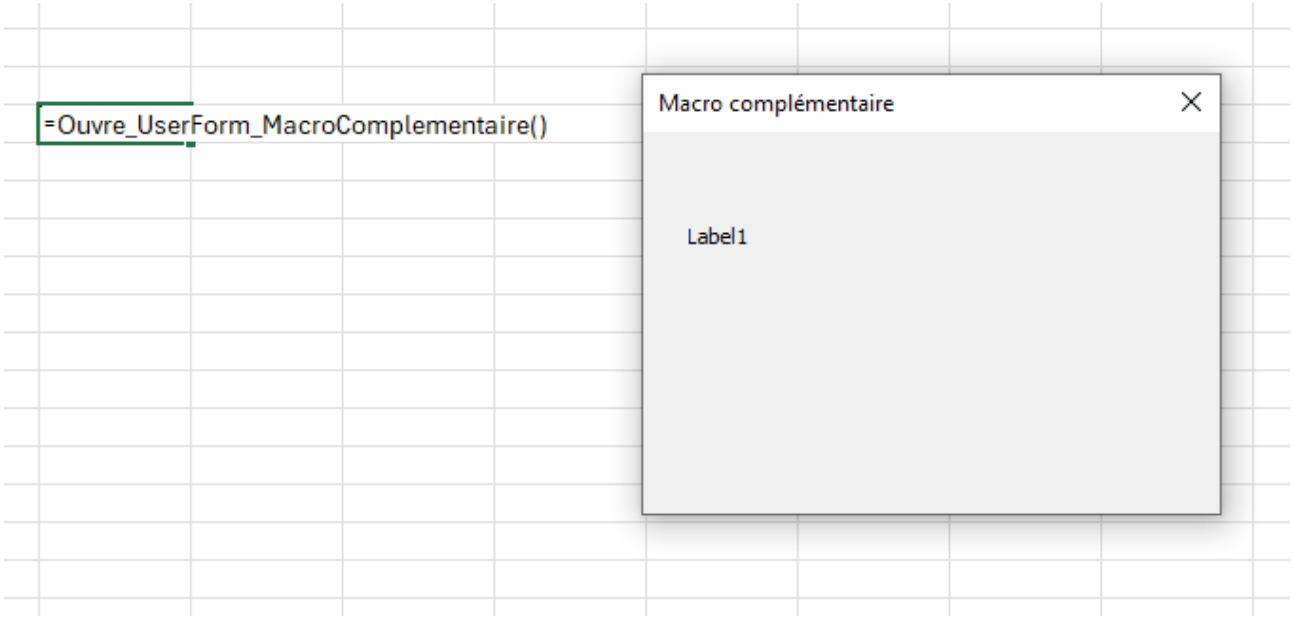

FIGURE 2.13. – Notre UserForm de macro complémentaire après modifications.

Au fil de cette section, nous avons vu comment nous servir des bibliothèques pour gagner en simplicité et en temps.

3. Classeur de macros personnelles

Le classeur de macros personnelles, aussi appelé `PERSONAL.XLSB`, est comme son nom l'indique, personnel.

Il permet de stocker du code (donc pas de réutilisation de feuilles par ce biais) pour tous ses classeurs. C'est en quelque sorte une bibliothèque à soi.

3.1. Enregistrer des macros dans le classeur personnel

Durant l'enregistrement d'une macro, il est possible de choisir le classeur de macros personnelles comme destination.

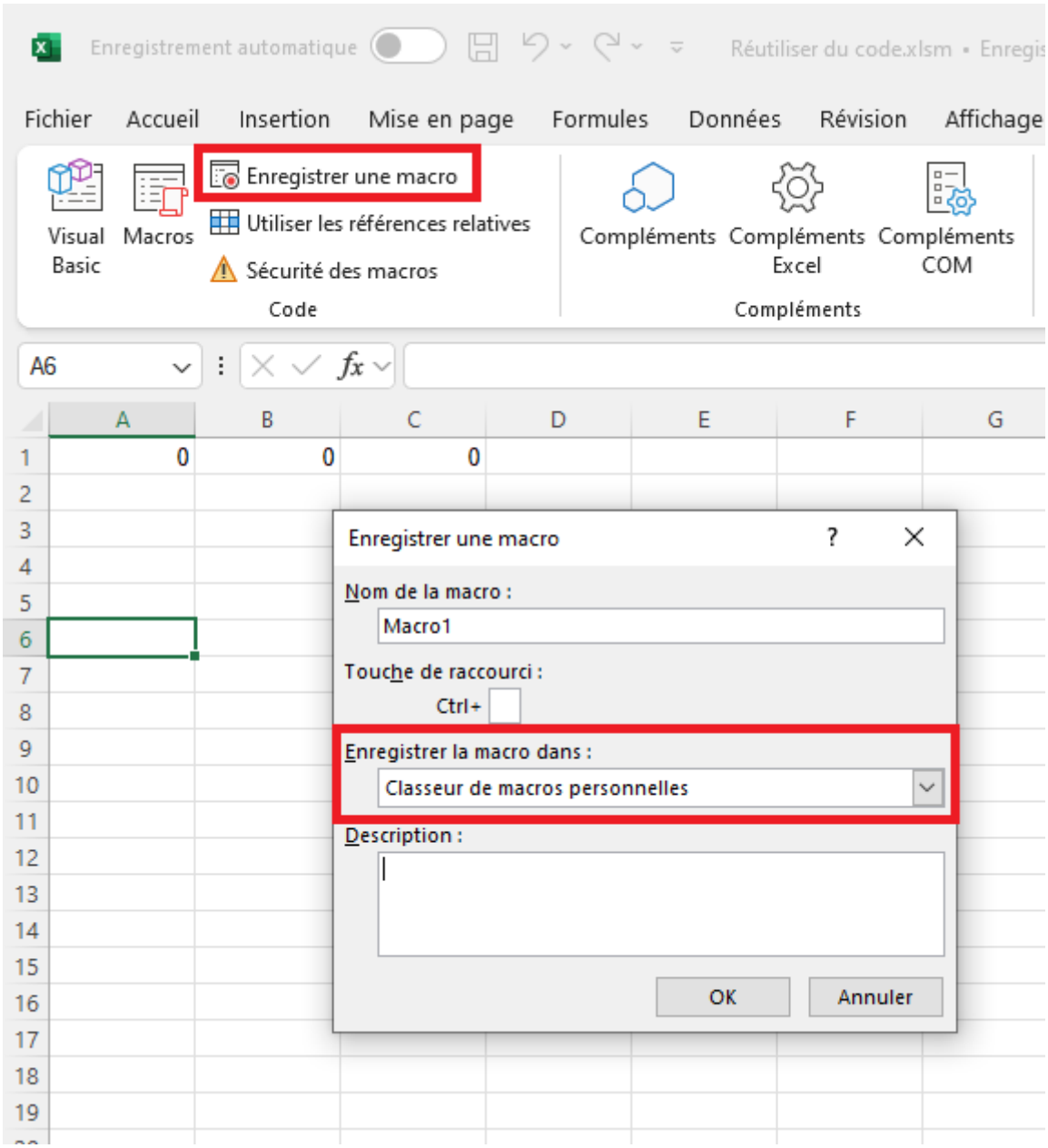

FIGURE 3.14. – Enregistrer une macro pour le classeur personnel.

L'enregistrement lancé, nous effectuons quelques opérations et y mettons fin.

Nous pouvons constater qu'un fichier `PERSONAL.XLSB` a bien été ajouté dans l'arborescence (si ce n'était pas déjà votre cas) avec notre code :

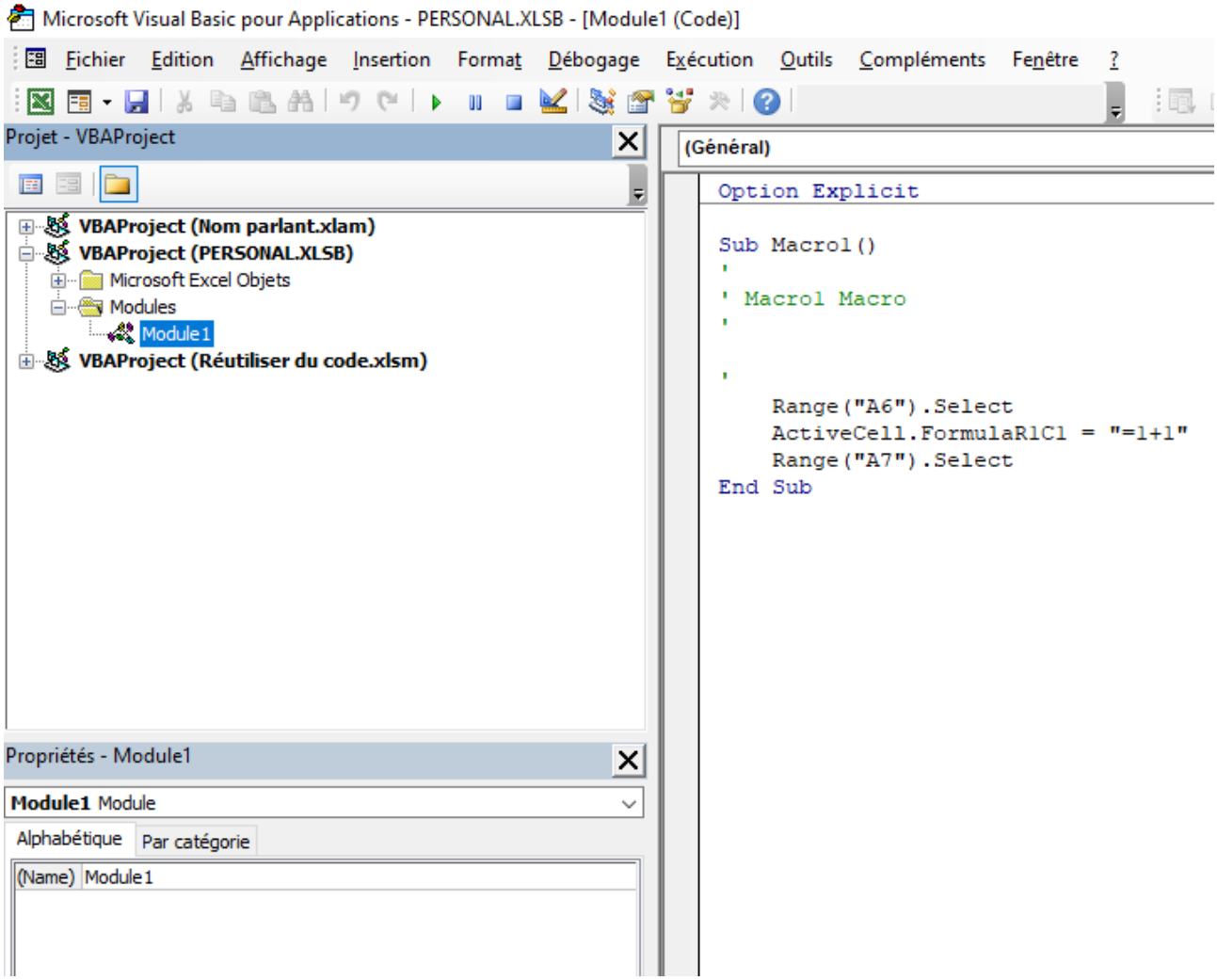

FIGURE 3.15. – Fichier PERSONAL.XLSB avec le code ajouté.

3.2. Éditer le classeur personnel

Nous pouvons éditer ce classeur directement dès qu'existant (enregistrez une macro comme expliqué juste au-dessus pour le créer sinon).

Pour l'édition, nous pouvons le trouver dans l'arborescence de projet de l'éditeur VBE ou l'ouvrir via le chemin suivant (attention aux éléments masqués dans l'explorateur de fichiers) : C:\Users**VotreNomUtilisateur**\AppData\Roaming\Microsoft\Excel\XLSTART.

Voilà tout pour ce classeur spécial.

Conclusion

Pendant ce billet, nous avons étudié la réutilisation de code par le biais des exports/imports, des bibliothèques et du classeur `PERSONAL.XLSB`.

> Pour des raisons de sécurité, soyez vigilant sur l'origine du code que vous souhaitez utiliser lorsque vous n'en êtes pas l'auteur.

Quelques ressources :

— La documentation pour l'installation de références
— Livre Programmation VBA pour Excel pour les nuls (Excel 2010, 2013 et 2016), de John Walkenbach
— Cette page