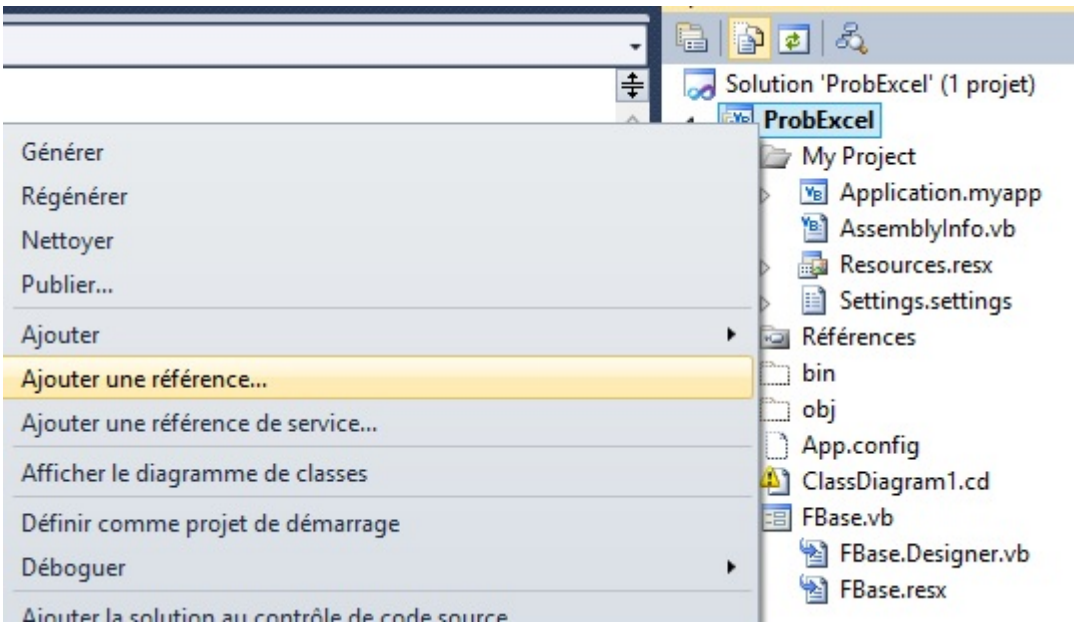
Je ne doute pas un instant du fonctionnement de ton code et moi, je ne suis pas arrivé au même résultat en évitant les références de type COM … ☹

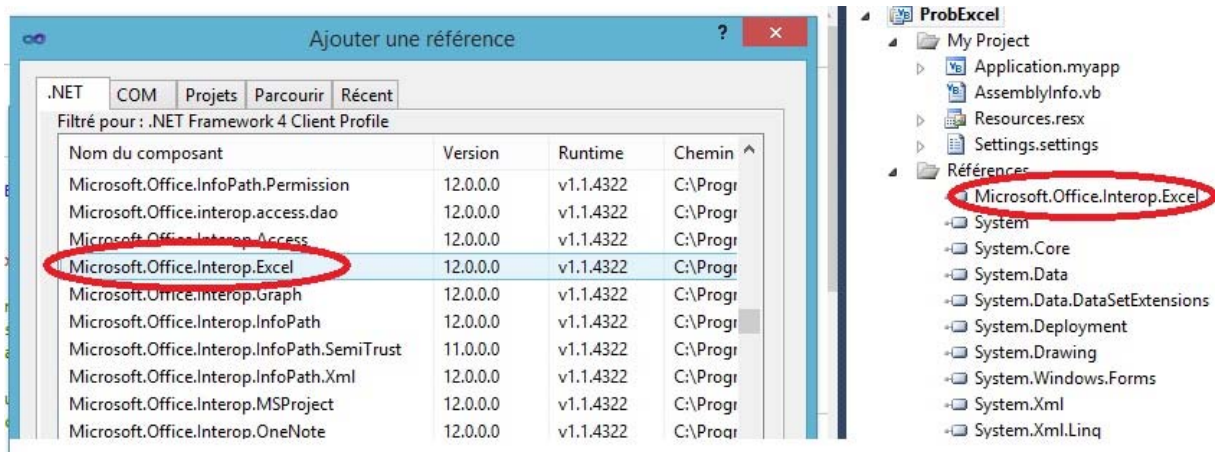
La notion de « code managé » concerne en fait les objets managés et la gestion de la mémoire : « Le ramasse-miettes ou garbage collector réalise la récupération des ressources allouées et devenues inutiles. Son travail est transparent pour le programmeur qui ne doit pratiquement jamais s'en occuper.
Il existe toutefois des cas où il ne suffit pas de compter sur le ramasse-miettes. Ce dernier libère en effet automatiquement la mémoire allouée à un objet managé lorsque cet objet n'est plus utilisé, mais il est impossible de prévoir quand il va le faire.
De plus, le ramasse-miettes n'a pas connaissance des ressources non managées telles que les handles de fenêtres et les fichiers et flux ouverts.
Les objets managés sont ceux dont le code est spécifiquement conçu pour le Framework et sont, de ce fait, contrôlés par le CLR (Common language runtime). A l'inverse notamment, des composants appartenant à d'anciennes versions de VB se retrouvent parmi les objets non managés. Lorsqu'il doit ajouter une référence dans un projet, la fenêtre de dialogue qui s'ouvre au programmeur possède plusieurs onglets dont NET et COM. Les références proposées sous l'onglet NET sont des ressources managées. Celles de l'onglet COM ne le sont pas.
Le programmeur peut parfois souhaiter que la libération d'une ressource soit immédiate. Il doit par ailleurs absolument libérer les ressources qui sont inconnues du ramasse-miettes.
Plusieurs moyens existent pour libérer explicitement les ressources en association avec le ramasse-miettes. Ce sont les méthodes **`Close`**, **`Dispose`** et **`Finalize`**.
L'étude des destructeurs est abordée un peu plus loin dans ces pages, mais il faut déjà noter qu'il est préférable d'utiliser la méthode **`Close`** quand l'objet concerné l'implémente. Cette méthode appelle la méthode **`Dispose`** sans lui passer de paramètres. Si l'objet ne possède pas cette méthode ou s'il est nécessaire de paramétrer le destructeur, il faut utiliser la méthode **`Dispose`** adéquate. La méthode **`Finalize`** de la classe **`Object`** ne doit jamais être appelée explicitement, elle est exécutée automatiquement lors de la destruction de l'objet. »

L'espace de nom **`Microsoft`**.**`VisualBasic`**, a été créé pour permettre aux programmeurs VB 6.0 de retrouver bon nombre de leurs outils favoris, il donne notamment la commande Shell nécessaire à l'exécution de programmes externes (à noter que cette commande peut être remplacée par la commande Start).
Cette librairie est référencée par défaut dans tout nouveau projet. Afin de se prémunir d'éventuelles incompatibilités futures, le programmeur doit n'utiliser les outils de **`Microsoft`**.**`VisualBasic`** que lorsqu'ils n'ont pas de remplaçant dans DotNet. Pour la plupart des applications, le programmeur peut même désactiver l'importation automatique de cette librairie. (cf. ExtraitVisualBasic.pdf ci-joint).

Comment choisir entre .Net et Com lorsqu'on ajoute une référence. ? Solution en images :



Sous l'interface de VisualStudio 2010, les onglets sont explicites :



Sous l'interface de VisualStudio 2015, il faut choisir « Assembly » et commander une recherche (sur office, par ex.) pour trouver Microsoft.Office.Interop.Excel :