



Dotnet France
Technologies Sharepoint, SQL Server & .NET

Association Dotnet France

Le .NET Compact Framework Windows Mobile

HEROGUEL Quentin

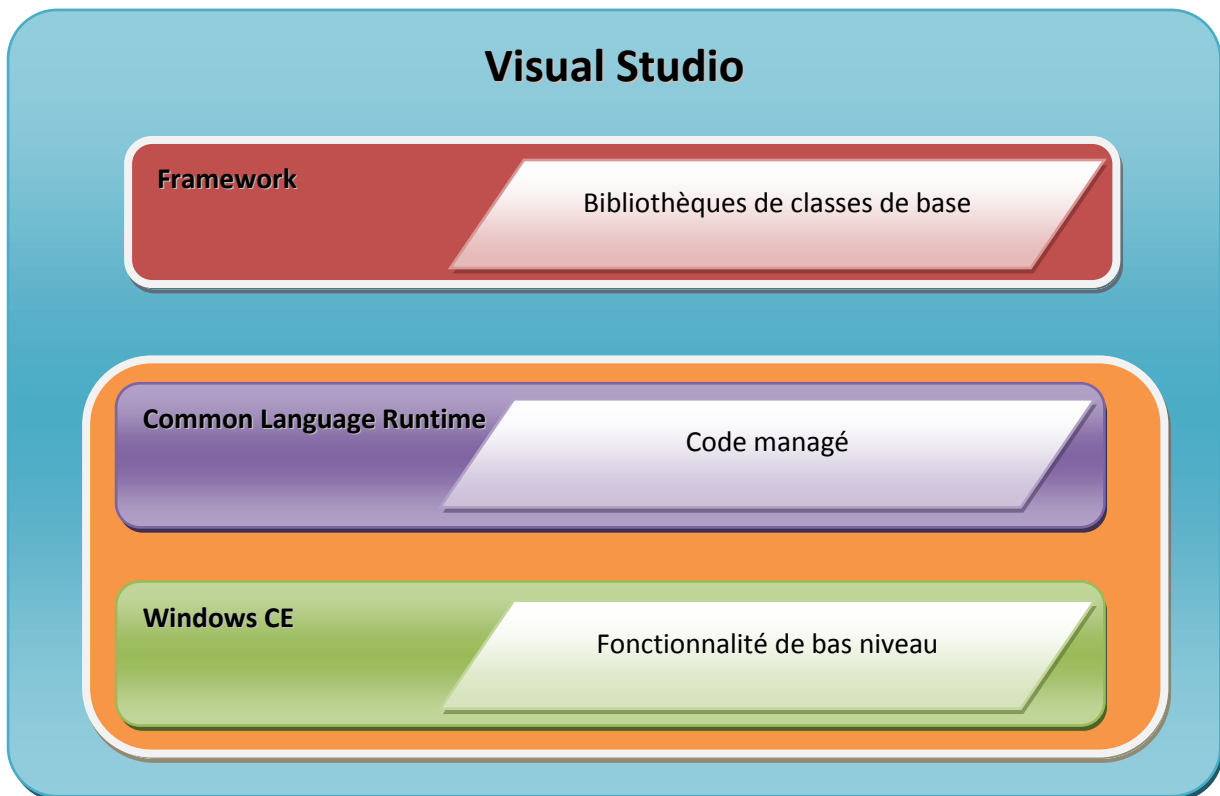
Sommaire

1	Le .NET Compact Framework	3
1.1	Vue d'ensemble rapide de l'architecture du .NET Compact Framework.....	3
1.2	Windows CE.....	3
2	Différence entre le .NET Compact Framework et le .NET Framework.....	4
3	Le Common Language Runtime (CLR)	5
3.1	Schéma général du CLR	5
3.2	Le Garbage Collector	6
3.3	Le domaine d'application	6
3.4	L'hôte de domaine d'application.....	7
3.5	La vérification d'exécution du type sécuritaire	7
4	L'assembly	8
4.1	Définition.....	8
4.2	Statique ou dynamique ?	8
4.3	Les noms fort <i>d'assemblies</i>	8
4.4	Le Global Assembly Cache.....	12
5	Les bibliothèques de classes.....	14
5.1	Vue d'ensemble.....	14
5.2	Créer une bibliothèque de classe.....	15
6	Conclusion	17

1 Le .NET Compact Framework

Avant tout il faut savoir que le .NET Compact Framework est un sous ensemble du .NET Framework, c'est aussi une version allégée du .NET Framework. Par exemple l'ASP.NET est totalement inutile pour les PDA ou les Smartphones puisqu'ils ne servent pas de serveur. Le .NET Compact Framework contient uniquement des fonctionnalités qui lui sont propres mais aussi les fonctionnalités les plus utilisées par le .NET Framework. Cependant il ne comporte pas les fonctions redondantes présentes dans le .NET Framework, c'est-à-dire que si vous possédez plusieurs façons de faire en programmation avec le .NET Framework il n'existera qu'une façon de faire pour le .NET Compact Framework. Il permet aussi de faciliter le transfert d'application bureautique aux périphériques.

1.1 Vue d'ensemble rapide de l'architecture du .NET Compact Framework



1.2 Windows CE

Le système d'exploitation Windows CE permet au .NET Compact Framework d'avoir en plus des fonctionnalités principales des fonctionnalités spécifiques aux périphériques. Ces fonctionnalités sont de plusieurs types et il comporte aussi des assemblées particulières comme le design pour Windows Form ou les services Web pour le Web Form.

2 Différence entre le .NET Compact Framework et le .NET Framework

	.NET Compact Framework
Domaine d'applications	Ne prend pas en charge les assemblies issues d'un code indépendant d'un domaine dans le but de l'utiliser pour plusieurs domaines d'applications.
ASP.NET	Ne prend pas en charge l'ASP.NET.
Classes et Types	Étant un sous-ensemble du .NET Framework, le .NET Compact Framework comporte des ressources limitées.
Common Language Runtime	Le CLR comporte les mêmes fonctionnalités que celui en .NET Framework (MSIL, JIT, garbage collector,...). Cependant il ne comporte pas les fonctions redondantes, c'est pourquoi il ne représente que 12% du CLR du .NET Framework.
Contrôles	Le .NET Compact Framework comporte la plupart des contrôles Windows Form du .NET Framework Complet et aussi des contrôles spécifiques aux périphériques mobiles.
Évènements	Il comprend les évènements les plus utilisés (par exemple GotFocus).
Langages	Le Visual Basic et le C# sont les seuls langages disponibles pour le .NET Compact Framework.
Threads	Quatre types de threads sont disponibles dans le .NET Compact Framework : le thread pour l'application principale, le thread pour gérer le temps (minutage, délais, ...), un thread pour les modifications de l'interface TCI/IP, et le thread pour la gestion des objets (garbage collector).
XML	Le .NET Compact Framework prend en charge les DOM XML (Document Object Model). Cependant, étant une version allégée, il ne comporte pas certains composants XML comme la Classe XMLDataDocument ou encore la transformation XSL (eXtensible Stylesheet Language Transformation).

3 Le Common Language Runtime (CLR)

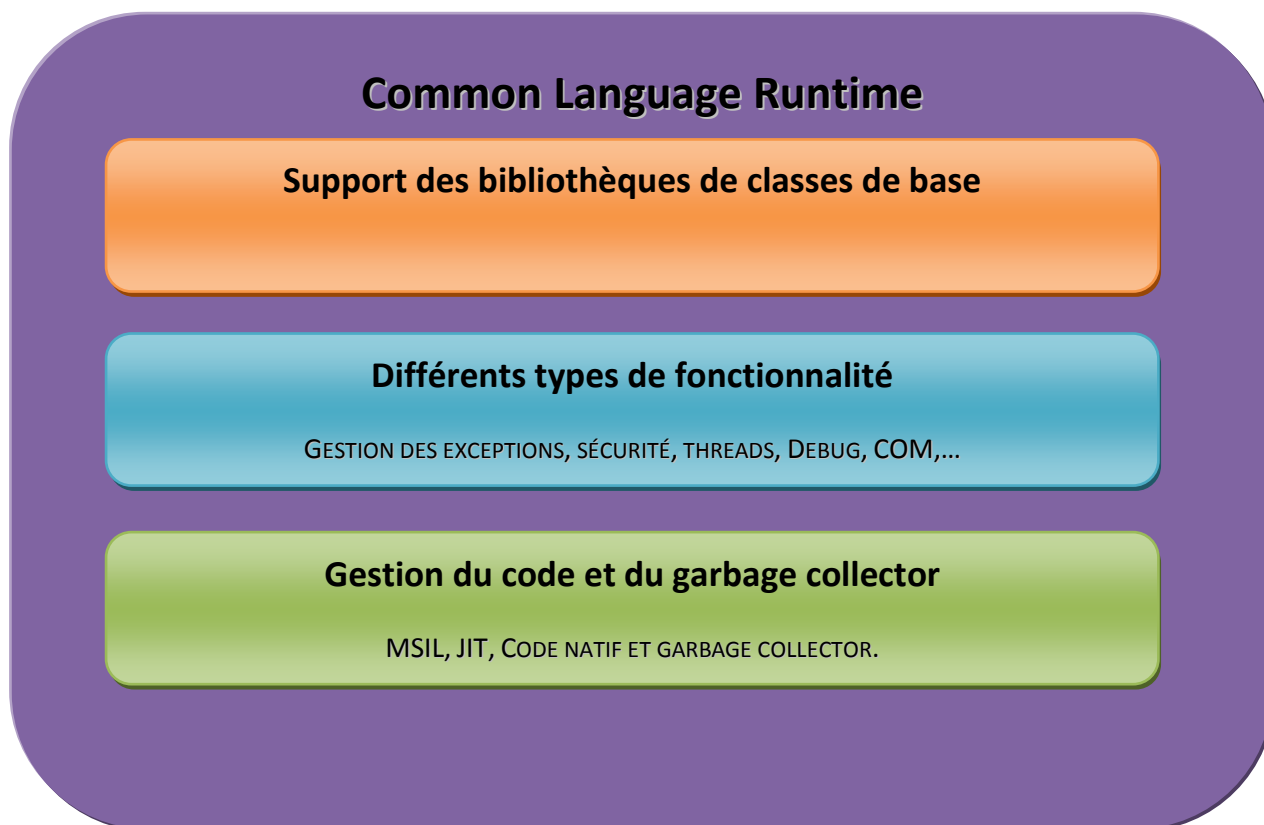
Le *Common Language Runtime* (CLR) est un environnement d'exécution managé. Il se compose de deux parties distinctes : un moteur d'exécution et des bibliothèques de classes qui sont la base de toute application. Tout code que vous créez avec un compilateur de langage ciblant le *runtime* est appelé code managé.

Tout simplement le CLR permet de manager le code natif, l'allocation de mémoire ou le *garbage collector* et le chargement des classes. Il intègre aussi l'intégration inter-langage, la gestion inter-langage des exceptions, ce qui permet ainsi d'avoir une interaction entre les composants ou bien intègre encore les services de débogage, une sécurité améliorée, la prise en charge du déploiement enfin une performance accrue.

Le CLR fonctionne avec le *Microsoft Intermediate Language* (MSIL) qui résume la compilation du programme dans un langage de bas niveau (en code IL). Ensuite le code IL est compilé en code binaire par le "Just-In-Time" ou JIT pour finalement être exécuté par la machine.

Ainsi grâce au CLR, l'environnement d'exécution peut supporter plusieurs langages différents (C#, VB.NET, C++,...).

3.1 Schéma général du CLR



3.2 Le Garbage Collector



Le *garbage collector* (ou bien appelé familièrement “ramasse-miettes”) gère l’allocation et la libération de la mémoire. Lorsque vous créez un nouvel objet, le *runtime* alloue la mémoire suffisante pour ce nouvel objet. Une fois que l’objet n’est plus utilisé, le *garbage collector* va dé-allouer la mémoire utilisée ; la rendant ainsi disponible pour d’autres objets.

3.3 Le domaine d’application

Un domaine d’application s’apparente à un système d’exploitation sauf qu’il est entièrement sous le contrôle de la CLR. Chaque application .NET fonctionne dans un domaine d’application et la CLR manage les ressources afin que celles-ci soient toutes utilisées par l’application au moment de l’exécution. Il vérifie aussi lors de l’opération qu’elles soient libérées lorsque l’application se termine. Pour simplifier, le domaine d’application est un ensemble englobant tous les threads d’un même processus. Enfin la CLR peut gérer plusieurs domaines d’application mais un domaine d’application gère une seule application.

Le domaine d’application présente plusieurs avantages :

- Le code managé doit obligatoirement passer par un processus de vérification qui assure la sécurité de l’application ce qui est très important en Windows Mobile. En effet le processus de vérification garantit que le code d’application est type-safe (de type sécuritaire) c’est-à-dire qu’une application ne peut pas exécuter. Le type-safe est une propriété d’un langage (C#, VB.NET) qui vérifie les types de valeurs et les types à la compilation de l’application. Ainsi la CLR permet d’obtenir un large niveau d’isolement des applications, comme si les processus de l’application étaient séparés. De plus les défauts d’adresse mémoire ne peuvent pas être produits, donc l’échec d’une application ne peut pas affecter une autre application.
- Vous pouvez créer une nouvelle application dans un nouveau domaine d’application, et vous pouvez stopper une application dans un autre domaine sans gérer les processus (un arrêt ou une mise en marche d’un processus).
- Par défaut il existe un domaine d’application par processus, c’est-à-dire que le processus est synchrone et donc il met en suspend le prochain domaine d’application jusqu’à l’arrêt du domaine d’application qui précède. Par conséquent une application qui s’exécute dans un domaine d’application peut appeler un autre domaine d’application. Cependant une application peut démarrer une nouvelle tâche et créer un nouveau domaine d’application dans cette tâche permettant ainsi de faire fonctionner deux domaines d’application simultanément. En général vous n’aurez pas besoin de créer un nouveau domaine d’application.

3.4 L'hôte de domaine d'application

Étant donné que chaque application s'exécute dans un domaine d'application et chacune d'entre elle a besoin de charger les *assemblies* et de créer aussi le domaine d'application approprié. Pour cela il existe l'hôte de domaine d'application qui remplit ces fonctions. Il récupère toutes les informations utiles à l'exécution de l'application dans le code natif. Il a aussi pour principe de charger le CLR dans le processus d'exécution avec toutes ces informations, de créer le domaine d'application et charge le code d'application dans le domaine d'application. Il existe plusieurs types d'hôtes d'applications, et celui qui sera utilisé dans le .NET Compact Framework est l'hôte *shell* : ce sont des fichiers en .exe qui s'exécutent à partir du *shell*.

Remarques :

- les futures versions du .NET Compact Framework comprendront une interface de programmation d'applications (API) pour charger le CLR dans un processus.
- les fonctionnalités de l'hôte de domaine d'application se situent dans la classe `System.AppDomain`.

3.5 La vérification d'exécution du type sécuritaire

Le vérificateur d'exécution du type sécuritaire est une des parties les plus importantes du .NET Compact Framework. Il permet de garantir que l'exécution du code est sûre, améliorant ainsi la sécurité. En effet le vérificateur fait partie du JIT *compiler* (Just In Time). Donc avant toute exécution de méthodes dans le .NET *assemblies*, le compilateur JIT va compiler le code IL en code natif pour que la vérification d'exécution soit établie.

De plus le vérificateur s'assure que tous les types présents dans le code sont correctement déclarés et utilisés. Cela signifie donc que le code .NET ne peut pas être mis en échec en raison de l'une des caractéristiques suivantes : utilisation de variables non initialisées, mauvaise utilisation de pointeurs,...

4 L'assembly

4.1 Définition

L'*assembly* (ou assemblage) est une unité de déploiement de base dans le .NET. C'est le résultat entre la compilation et la liaison du code. Il s'agit donc d'un ensemble de codes, de métadonnées (données servant à définir ou décrire d'autres données) et de ressources (ex : images .png, chaînes de caractères ...). Les métadonnées sont un concept récent dans le .NET et dans les applications Microsoft, elles permettent de donner des informations supplémentaires qui indiquent de quelle manière il faut exécuter le code. Ainsi toutes les informations d'une *assembly* qui sont utilisées au moment de l'exécution sont stockées dans un manifeste (*assembly manifest*) qui contient donc la description du contenu de l'*assembly* : types de données, nom, version, autre *assembly*, sécurité.

Les deux plus gros avantages de l'*assembly* sont la fin de problèmes de versions ("versioning problems") et des conflits dll. Les *assemblies* ont été conçues pour simplifier le déploiement des applications et pour résoudre ces problèmes.

4.2 Statique ou dynamique ?

Il existe deux types d'*assemblies* : les *assemblies* statiques et les *assemblies* dynamiques. Les *assemblies* statiques (très utilisées en C++) sont en fait des fichiers contenant codes « semi-exécutables ». Ces codes représentent le programme compilé en partant de la source mais ne contiennent pas tous les liens pouvant être fait vers d'autres *assemblies*. Il faudra donc compiler un autre programme (qui ne soit pas une *assembly* statique) qui utilise notre librairie statique pour effectuer tous les liens requis et ainsi générer un exécutable digne de ce nom.

Les *assemblies* dynamiques sont utilisées de façon dynamique à l'exécution du programme. C'est-à-dire que les *assemblies* ne sont pas associées au fichier exécutable du programme. On obtient deux fichiers distincts : les *assemblies* et l'exécutable.

Dans le cas du .NET, ce sont des *assemblies* dynamiques qui sont utilisées. Il existe plusieurs moyens de créer des *assemblies* dynamiques notamment avec Visual Studio 2008 en employant les outils de l'espace de nom [System.Reflection.Emit](#) ou encore en créant une bibliothèque de classes.

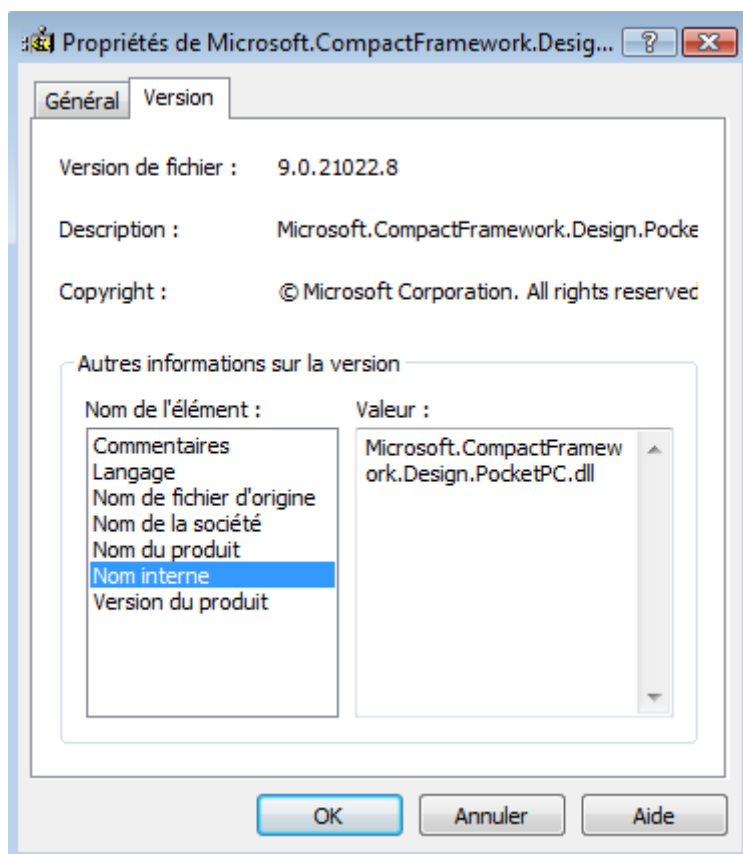
Remarque : vous pouvez obtenir une liste des fichiers et des *assemblies* du .NET Compact Framework en cliquant sur ce [lien](#).

4.3 Les noms forts d'*assemblies*

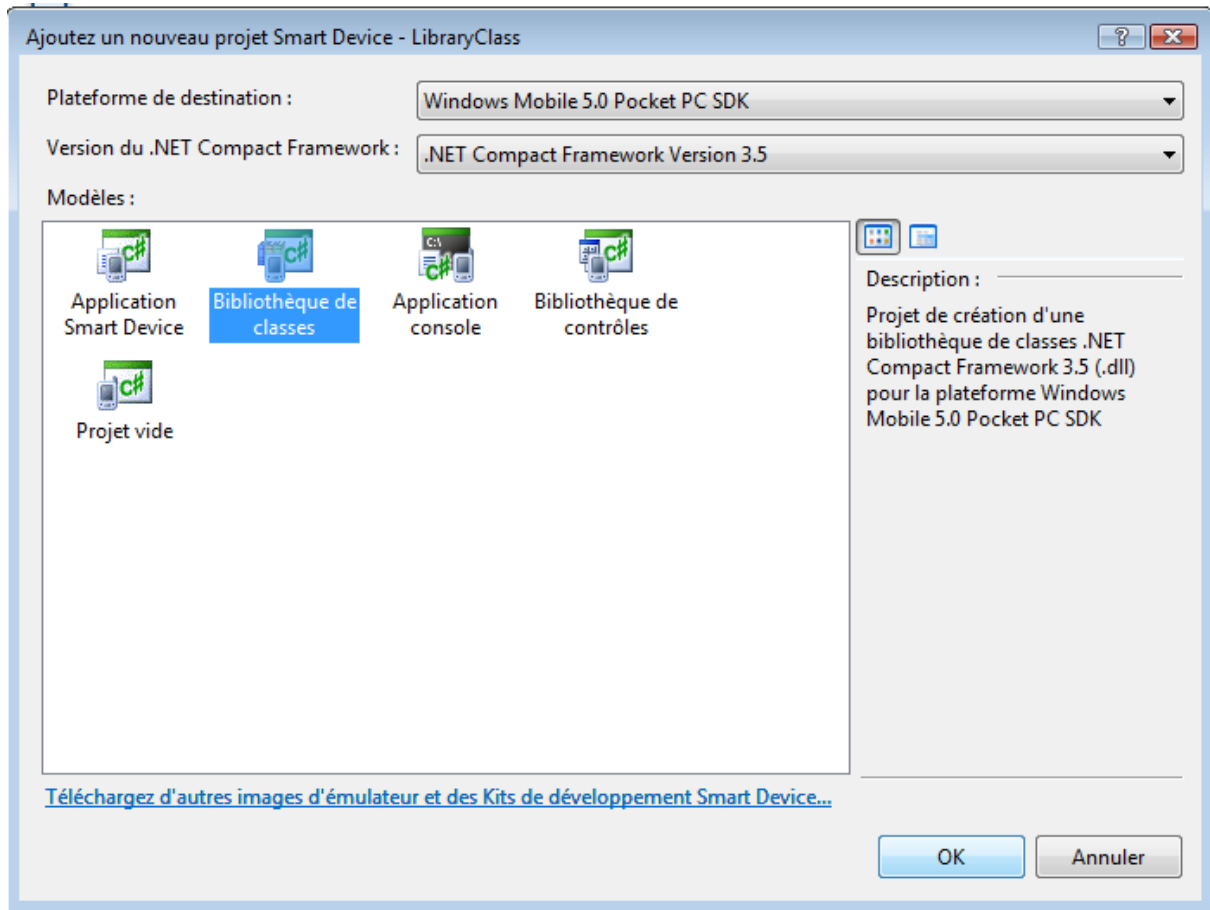
La principale différence entre les *assemblies* du .NET Framework Complet et ceux du .NET Compact Framework étant que ces dernières sont signées avec une paire de clés de noms forts différente de celle du .NET Framework. Ceci permet au CLR de distinguer les *assemblies* des deux types de .NET Framework. Nous allons donc vous parler des *assemblies* avec un nom fort.

L'identité d'une *assembly* se constitue d'un nom fort regroupant plusieurs informations (versions, nom textuel,...). Ces noms forts sont eux-mêmes composés d'une clé publique et d'une signature numérique. Ils présentent deux avantages utiles à la sécurité de vos *assemblies* : assigner un nom fort à vos *assemblies* protège votre application du *versioning* et l'affectation d'un nom. En effet l'affectation d'un nom s'appuie sur une paire de clés unique ce qui rend le nom unique.

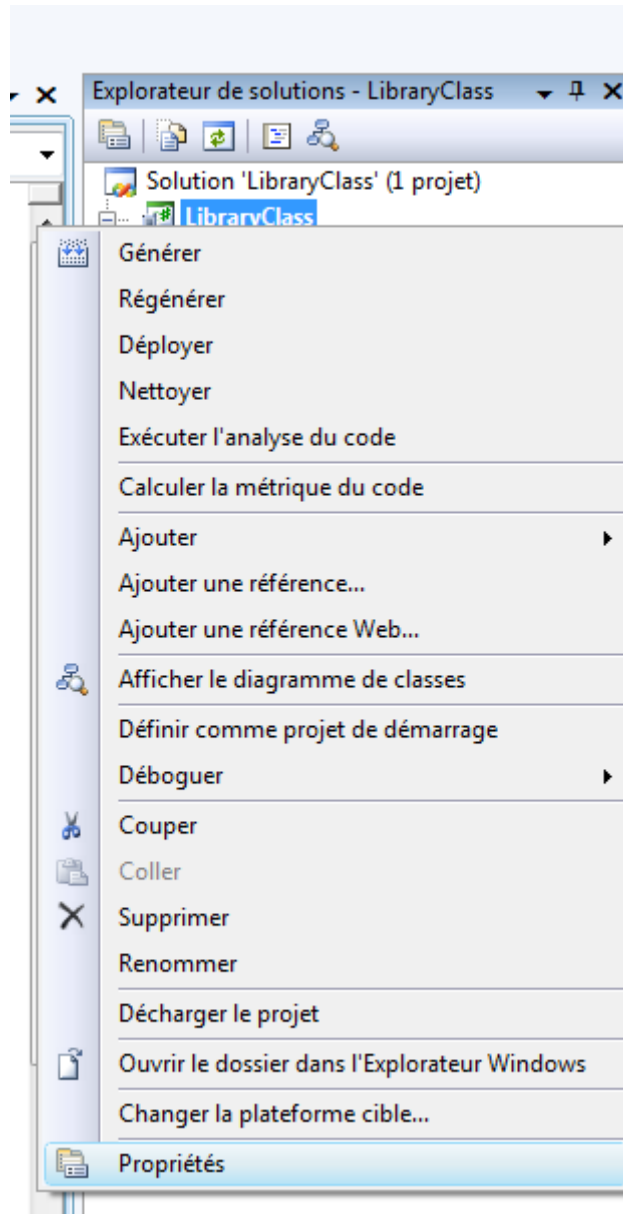
Exemple :



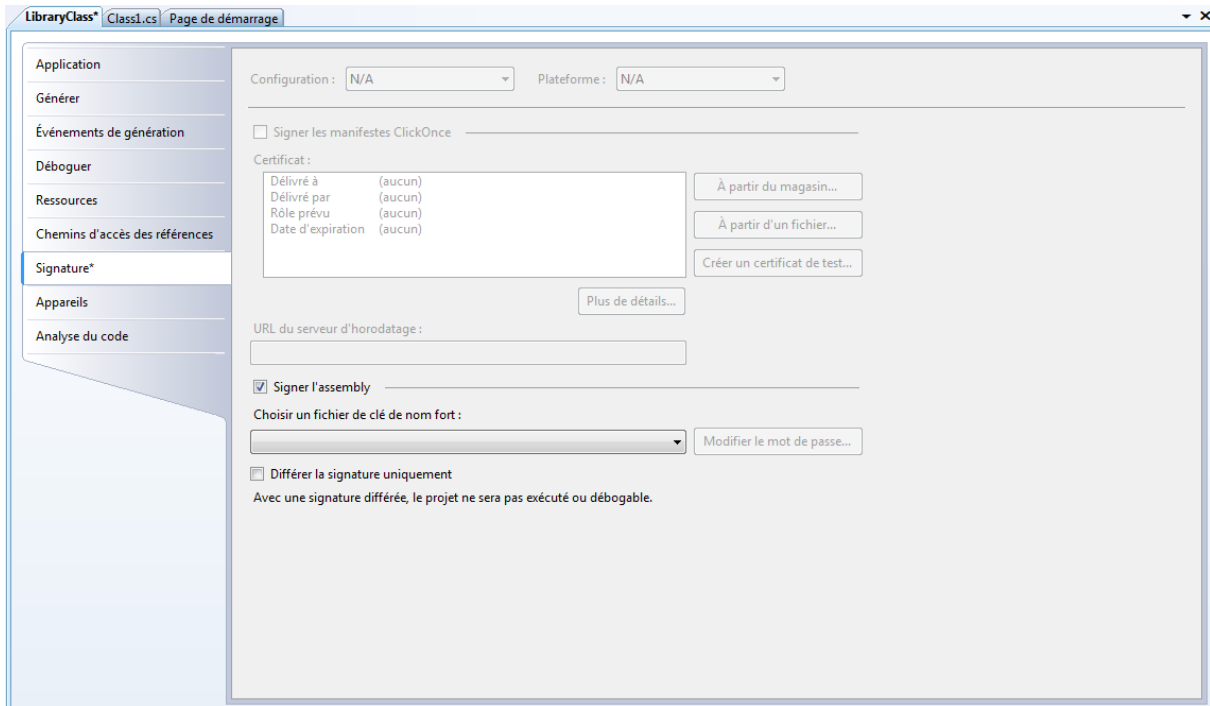
Pour cela vous devez créer une Bibliothèque de classes Windows Mobile. Sélectionnez dans le menu Fichier → Nouveau → Projet. Choisissez votre langage puis le type de projet (*SmartDevice*) et enfin validez la sélection d'une bibliothèque de classes comme dans l'exemple ci-dessous :



Ensuite vous devrez aller dans l'explorateur de solutions (si celui-ci n'est pas ouvert, sélectionnez dans le menu Affichage → Explorateur de solutions ou bien utilisez la commande Ctrl+W+S) et faire un clic-droit sur votre projet comme le montre le screenshot suivant pour notre projet *LibraryClass* :



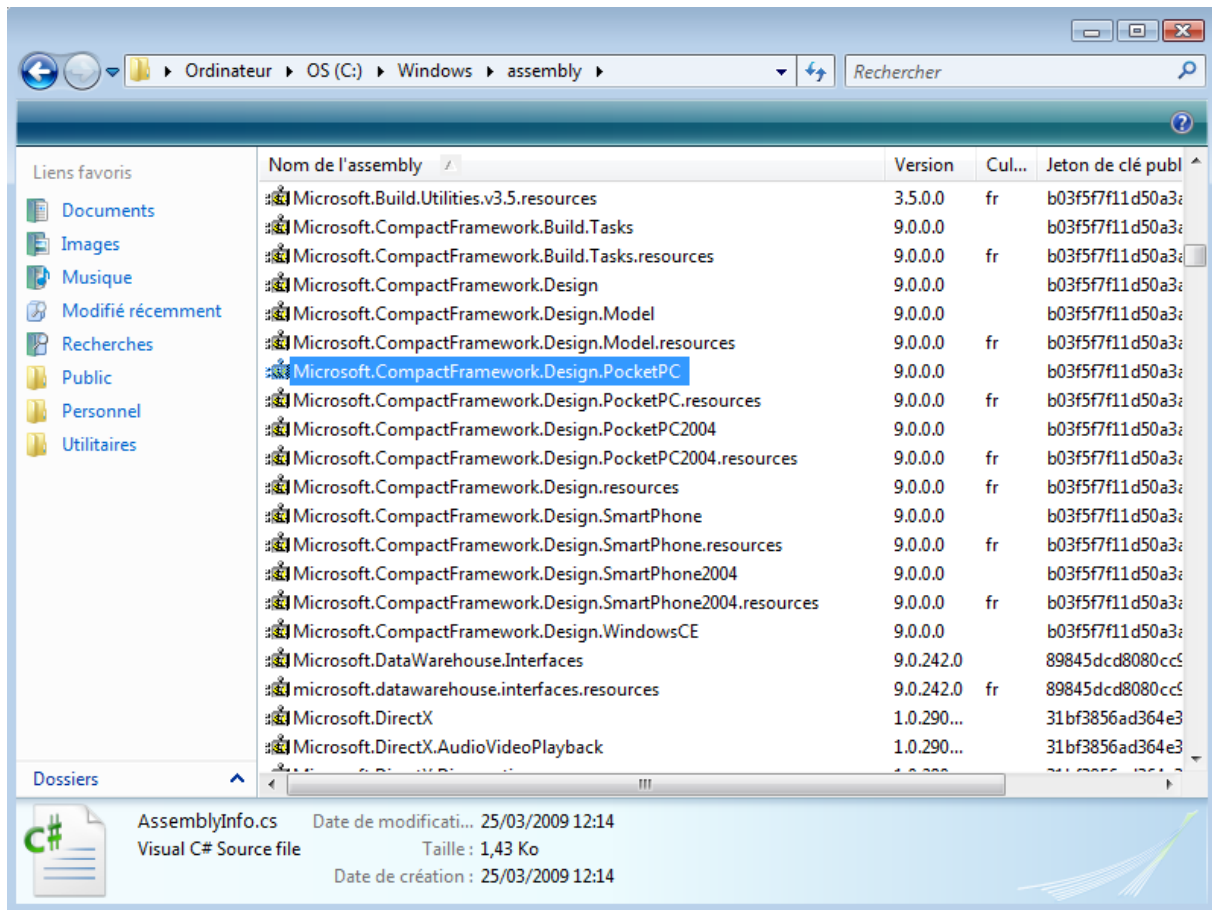
Vous aurez ainsi généré un nouvel onglet qui vous affichera toutes les propriétés de votre projet. Vous remarquerez une propriété nommée *Signature*. Sélectionnez-la et vous accéderez aux options vous permettant de générer un nom fort en choisissant un fichier de clé (Deux options : *Nouveau* ou *Parcourir*).



4.4 Le Global Assembly Cache

Le Global Assembly Cache (GAC) rassemble toutes les assemblies destinées à être partagées sur l'ordinateur pour différentes applications. Le Global Assembly Cache est un cache du CLR et vous pouvez déployer un assembly de plusieurs façons dans le GAC :

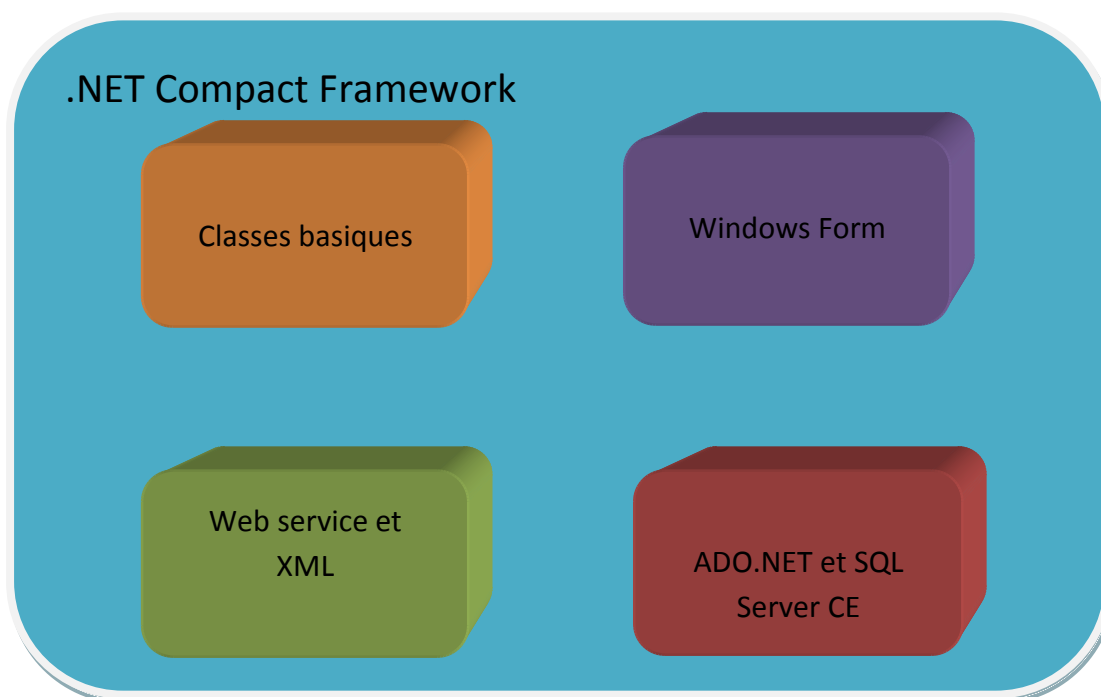
- L'option par défaut qui n'est autre qu'un programme d'installation fonctionnant avec le GAC.
- En utilisant l'explorateur Windows. Le dossier se situe par défaut dans C:\Windows\assembly. De cette façon lorsque vous copierez/glisserez vos assemblies dans le dossier vous les partagerez avec les autres applications.



5 Les bibliothèques de classes

5.1 Vue d'ensemble

Le plus intéressant dans le .NET Compact Framework de Windows Mobile pour les développeurs sont les bibliothèques de classes. Elles permettent aux développeurs d'obtenir un large choix de fonctionnalités quelque soit le langage de programmation choisi. De plus le .NET Compact Framework contient la moitié des classes du .NET Framework. Ainsi le .NET Compact Framework peut paraître assez limité cependant il permet de toucher à un grand nombre de technologies. Il contient par exemple des classes de Windows Form ainsi que des classes de Web service mais certaines fonctionnalités ne sont pas disponibles sous le .NET Compact Framework (comme exécuter un serveur Web sur vos périphériques mobiles). Ci-dessous vous trouverez un schéma permettant d'avoir une vue d'ensemble sur les types de bibliothèques de classes :

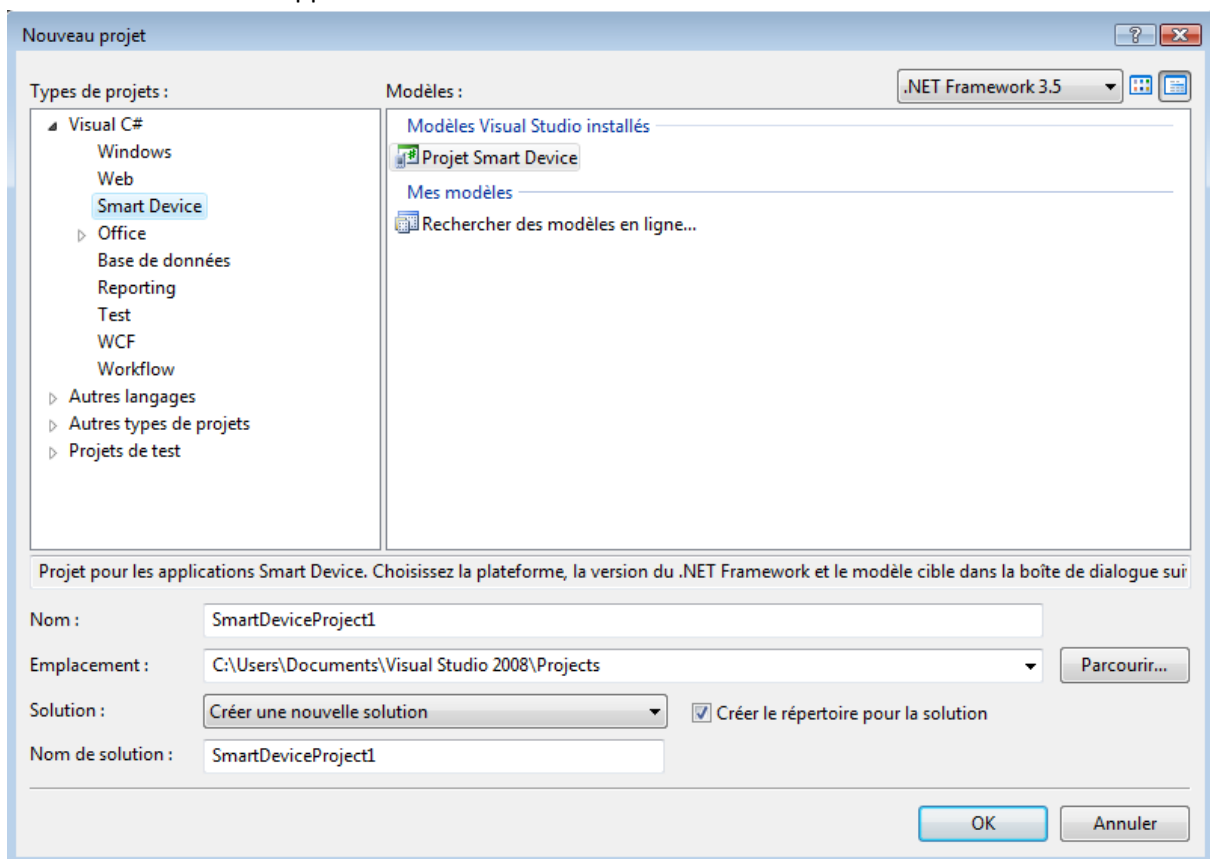


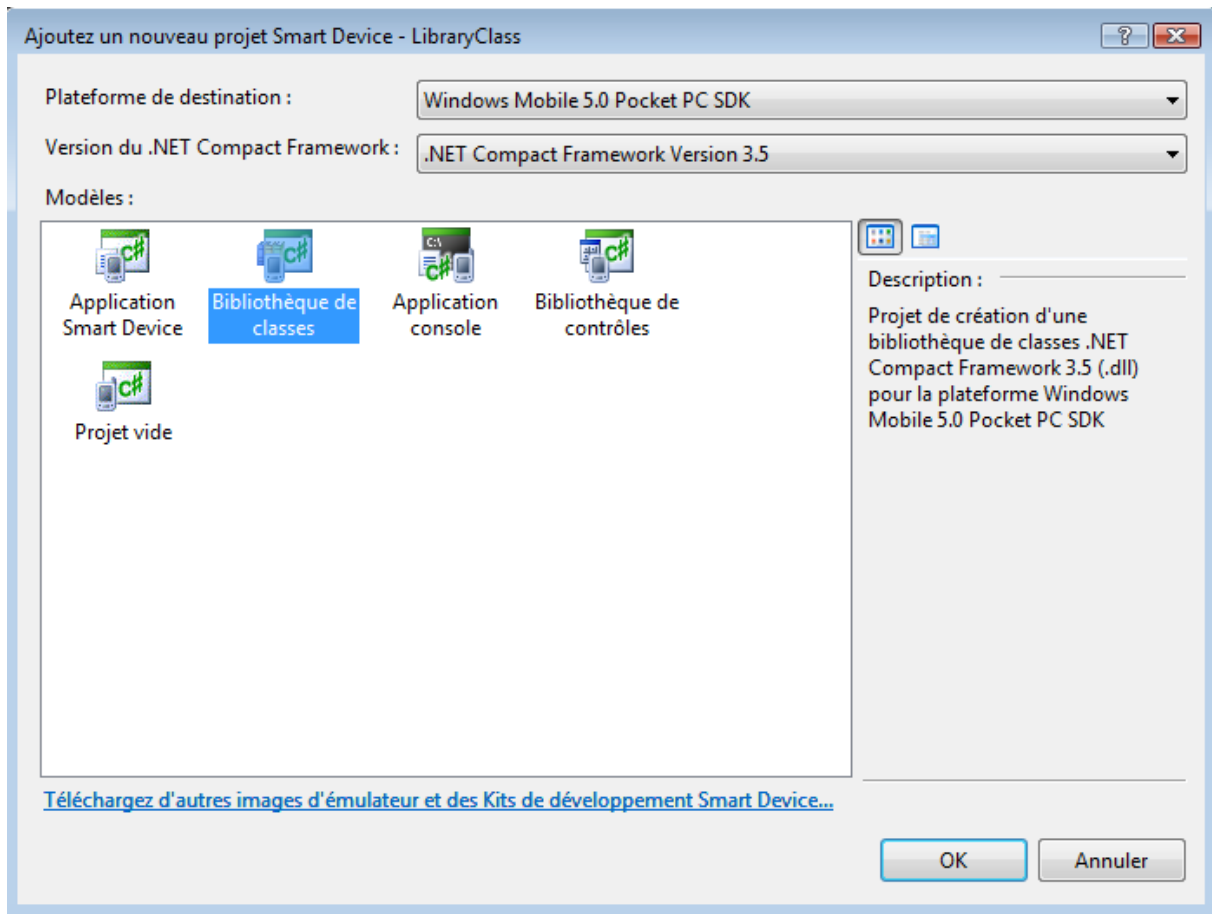
Remarque : Lorsque vous compilez un programme en langage IL, vous créez une assembly. De ce fait, une assembly peut aussi bien être une bibliothèque de classes (avec l'extension .dll) qu'un exécutable (avec l'extension .exe).

5.2 Créer une bibliothèque de classe

Allez dans le menu Fichier de Visual Studio 2008 et sélectionnez Nouveau →Projet (ou Ctrl+Maj+N). Sélectionnez votre langage et votre type de projet dans la partie correspondante.

Enfin choisissez Bibliothèque de classes comme modèle de projet et définissez la plateforme de destination ainsi que la version du .NET Compact Framework. Ensuite validez et vous aurez ainsi créé un projet de bibliothèque de classes dans lequel vous pourrez créer vos propres classes afin de les utiliser sur d'autres applications.





6 Conclusion

Ainsi vous avez vu dans ce chapitre, une vue d'ensemble du .NET Compact Framework en Windows Mobile. Vous avez pu comprendre comment fonctionne le .NET Compact Framework, quels sont ses principaux composants et quelle est la différence entre le .NET Framework et le .NET Compact Framework. De plus la création de vos propres bibliothèques de classes vous permet d'obtenir des fonctionnalités personnalisées en plus de celles présentes dans le .NET Compact Framework et de les partager avec d'autres utilisateurs.