Migration du code existant vers .Net 4 et Visual Studio

La nouvelle version de .Net et de Visual Studio, attendue par toute la communauté, est enfin là malgré quelques semaines de retard. Vous vous posez la question de savoir ce que vont devenir vos applications existantes ? Comment réaliser une migration réussie ? Avec quels outils ? Quelles sont les modifications à effectuer ? Les points délicats à connaître ? Les fonctionnalités devenues obsolètes ? Comment tester votre nouvelle application ? Cet article va répondre à toutes vos questions sur la migration.

Cette nouvelle version, qui accompagne Visual Studio 2010 (complètement réécrit en WPF), est majeure, car elle apporte une nouvelle version de la CLR dont voici quelques atouts. Elle permet tout d'abord de développer avec de nouveaux langages (F# mais aussi IronPython et IronRuby). Une nouvelle librairie nommée MEF (Managed Extensibility Framework) permet également de développer simplement une application extensible. Le traitement parallèle de l'information est aussi possible, grâce à PLINQ notamment. Enfin, la plupart des briques existantes (Entity Framework, WorkFlow Foundation, ...) ont été améliorées et la sortie de Silverlight 4 a été quasiment simultanée. Tout cela sera disponible une fois la migration de votre application effectuée !

Compatibilité des versions

Le Framework .NET 4 est compatible avec les applications qui ont été développées avec les versions antérieures (1.1, 2.0, 3.0 et 3.5), à l'exception de quelques changements qui ont permis d'améliorer la sécurité, la conformité aux normes, l'exactitude, la fiabilité et la performance. La compatibilité des versions du Framework .NET signifie qu'une application (ou un composant) développée sous une version antérieure est censée fonctionner avec les versions supérieures. De fait, un code source écrit avec une version du Framework .NET devrait compiler sur les versions futures. De même, les binaires qui s'exécutent doivent avoir un comportement identique entre les versions.

La migration simple...

Une fois la nouvelle version du Framework installée, grâce au nouveau processus d'exécution Side-By-Side (SxS), pas besoin de recompiler vos anciennes assemblies pour les utiliser avec vos nouvelles applications ! Il est maintenant possible pour une même application d'avoir différentes versions de la CLR s'exécutant en même temps et pouvant communiquer entre elles [Fig.1] .

Si vous voulez quand même forcer l'utilisation de la version 4 de la CLR, il vous faudra rajouter un fichier de configuration spécifiant la (les) version(s) à utiliser lors de l'exécution. Ce fichier doit lister un ou plusieurs éléments `<supportedRuntime version="vX.X" />`, le premier étant la version préférée. Le tableau ci-dessous montre les valeurs à utiliser dans votre fichier selon la (les) version(s) souhaitée(s) :

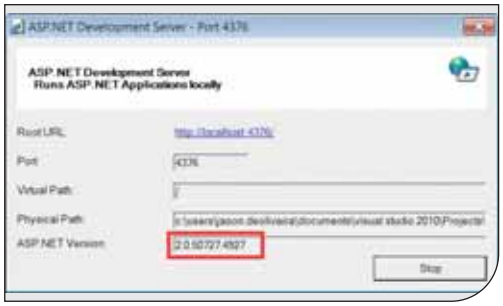
Version du Framework .NET	Version
4	v4.0
3.5	v2.0.50727
2.0	v2.0.50727
1.1	v1.1.4322
1.0	v1.0.3705

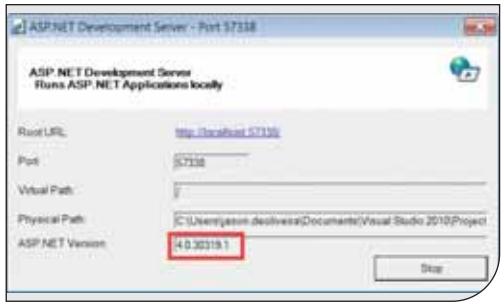
Et voici un exemple de fichier dans le cas de migration vers .NET 4 :

```
<?xml version="1.0" encoding="utf-8" />
<configuration>
  <startup>
    <supportedRuntime version="v4.0" />
  </startup>
</configuration>
```

Bien entendu, cela peut vite devenir fastidieux si vous devez migrer toutes les applications d'une même machine en même temps. Heureusement, il n'est pas nécessaire de créer un fichier par application. Il est possible de toutes les migrer d'un seul coup en modifiant la valeur de la clé « OnlyUseLatestCLR » de la base de registre de Windows [Fig.2] . Pour cela :

- Ouvrez un éditeur de base de registre Windows (exemple : regedit.exe)
- Allez dans [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\.NETFramework]





egilia®

LEARNING

LE SPÉCIALISTE DE LA
FORMATION CERTIFIANTE
EN INFORMATIQUE
ET MANAGEMENT

Faire de vos succès
notre réussite

www.egilia.com

CONTACTEZ NOS CONSEILLERS FORMATION

N°National 0 800 800 900

APPEL GRATUIT DEPUIS UN POSTE FIXE

ANVERS . LIEGE . PARIS . LYON . LILLE . AIX-EN-PROVENCE .
STRASBOURG . RENNES . BRUXELLES
TOULOUSE . BORDEAUX . GENEVE . LAUSANNE . ZURICH .

- Mettez à jour la valeur de la clé OnlyUseLatestCLR=dword: 00000001
- Pour les systèmes 64 bits, mettez également à jour la clé dans [HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\.NET Framework]

Pour désactiver cette fonctionnalité, il vous suffit de remettre l'ancienne valeur de la clé OnlyUseLatestCLR=dword:00000000.

L'assistant de conversion

Comme pour le passage de Visual Studio 2005 à Visual Studio 2008, Microsoft propose avec Visual Studio 2010 un assistant de conversion pour migrer une solution (et donc tous ses projets) ou un projet donné au format VS 2010 [Fig.3]. La solution, ou le projet converti sera créé au même emplacement que l'existant, mais vous pourrez effectuer une sauvegarde avant la conversion. Lorsqu'un projet ne peut être converti, il est marqué comme indisponible dans l'Explorateur de solutions, jusqu'à la résolution des incidents. Pour migrer un ensemble de projets ou de solutions, il est possible d'automatiser leur conversion en créant un fichier batch.

Ce qu'il faut savoir…

La plupart des changements qui ont affecté le Framework .NET 4 ne nécessitent pas de modification du code de vos applications. Cependant, si vous rencontrez des problèmes, sachez qu'il existe des solutions de contournement prévues par Microsoft. En voici quelques exemples.

Core

L'accès aux fichiers de configuration par le Framework a été modifié. Si votre fichier de configuration d'application est nommé par exemple MonApplication.config, renommez-le en MonApplication.exe.config.

ASP.NET

Lors de la mise à jour des applications ASP.NET 3.5 vers ASP.NET 4 grâce à l'assistant de conversion, Visual Studio 2010 modifie automatiquement le fichier Web.config de l'application ASP.NET 3.5 pour contenir les paramètres appropriés à la nouvelle version. Si vous voulez désactiver ces nouveaux paramètres, voici ce qu'il faut rajouter dans le fichier Web.config :

- Le rendu de certains contrôles a été modifié. Pour désactiver le nouveau mode de rendu, rajoutez le paramètre suivant :

```
<pages controlRenderingCompatibilityVersion="3.5" />
```

- Le mode de validation des requêtes http a été amélioré (protection contre les attaques cross-site scripting). Pour rétablir le comportement précédent, rajoutez le paramètre suivant :

```
<httpRuntime requestValidationMode="2.0" />
```

- Le nouveau paramètre ClientIDMode qui permet de spécifier l'attribut ID pour les éléments HTML, est paramétré par défaut afin d'assurer la compatibilité avec les versions précédentes. Cependant, si l'on utilise le pool d'applications IIS du Framework .NET 4, il faut rajouter le paramètre suivant pour désactiver le mode par défaut :

```
<pages ClientIDMode="AutoID" / >
```

D'autre part, les méthodes UrlEncode et HtmlEncode ont été modifiées. Vérifiez que leur comportement est bien celui attendu.

Enfin, l'analyseur de pages (pour les ASPX et les ASCX) est plus strict que celui des versions précédentes. Il est donc conseillé de corriger les balises invalides.

Windows Presentation Foundation (WPF)

Une exception de sécurité sera générée sous Firefox pour les applications utilisant un contrôle WebBrowser ou un contrôle Frame avec du HTML et qui s'exécutent en Partial Trust dans la zone Internet sur des sites qui ne sont pas de confiance.

Pour résoudre ce problème, exécutez l'application en Full Trust, ajoutez le site de l'application dans les sites de confiance ou exécutez l'application dans Internet Explorer. Le parsing XAML a été modifié. Les attributs en XAML ne peuvent maintenant avoir qu'un seul point. Les attributs suivants sont valables :

```
<button Background="Red"/>
<button Button.Background="Red"/>
```

L'attribut suivant n'est plus valable :

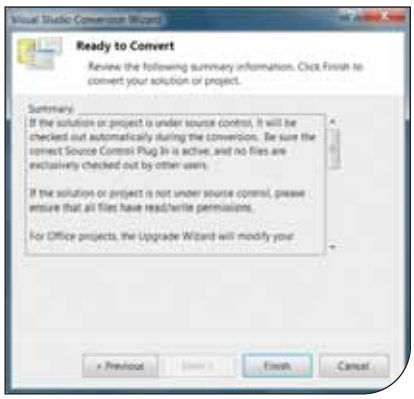
```
<button Control.Button.Background="Red"/>
```

Il vous faudra donc corriger les attributs XAML qui ont plus d'un point.

Les fonctionnalités obsolètes

Chaque nouvelle version du Framework .NET ajoute de nouveaux types et de nouveaux membres qui fournissent des fonctionnalités nouvelles. Les types existants et leurs membres changent également. Par exemple, certains types sont remplacés car la technologie qu'ils implémentent est supplantée par une nouvelle technologie ; et certaines méthodes sont remplacées elles aussi par de nouvelles qui sont plus pratiques ou plus complètes.

Le Framework .NET et le Common Language Runtime s'efforcent de respecter une compatibilité descendante (qui permet aux applications développées avec une version antérieure du Framework .NET de continuer à fonctionner sur la nouvelle version). Au lieu de supprimer directement un type ou un membre, le Framework



indique qu'il ne devrait plus être utilisé, en le marquant comme obsolète (ou déprécié) avec l'attribut ObsoleteAttribute. Cela permet de prévenir les développeurs de sa disparition future et leur permettre d'adapter le code existant. Toutefois, ce dernier continue à fonctionner dans la nouvelle version du Framework .NET.

Pour prévenir de l'utilisation de fonctions obsolètes, le compilateur peut émettre un message d'avertissement ou même d'erreur (dans ce cas, la compilation échoue et la correction est obligatoire). Cependant, les applications dans ce cas et déjà compilées avec succès, seront toujours exécutées avec succès. Seule la tentative de recompiler l'application échouera. Le message d'avertissement du compilateur qui signale les types obsolètes propose souvent une solution appropriée. Si ce n'est pas le cas, il vous faudra modifier votre code en supprimant l'utilisation du type obsolète.

Voici quelques exemples de types / membres qui ont été dépréciés.

Types obsolètes

- Les types présents dans l'espace de noms System.Data.Oracle ont été dépréciés. La politique de Microsoft est de ne plus fournir de provider pour Oracle et de laisser les éditeurs fournir leur propre provider (ODP.NET notamment).
- L'authentification via Passport n'est plus supportée, Live ID étant désormais utilisé. Tous les types associés sont donc obsolètes.
- Les fonctionnalités liées aux mails, présentes dans System.Web.Mail sont à remplacer par celles de System.Net.Mail.
- L'assembly System.Web.Mobile.dll permettant de développer des applications pour mobiles a été dépréciée.
- Côté Xml, XmlDataDocument et XslTransform (remplacé par XslCompiledTransform) ont été dépréciés.

Membres obsolètes

- Certaines méthodes de LINQ ont été modifiées pour autoriser l'utilisation de PLINQ, qui permet une utilisation optimale des multiprocesseurs et processeurs multi-cœurs.
- Pour Entity Framework, la fonction ApplyPropertyChanges est remplacée par ApplyCurrentValues, la fonction SaveChanges prend en paramètre une énumération au lieu d'un booléen.
- Plusieurs méthodes de System.Diagnostics.Process sont remplacées par des méthodes équivalentes en 64 bits.
- Des membres déjà obsolètes depuis le Framework 2.0 n'ont pas été supprimés (vous ne devriez déjà plus les utiliser !).

Ex : ConfigurationSettings.AppSettings (remplacé par ConfigurationManager.AppSettings) ou Page.SmartNavigation (remplacé par Page.SetFocus et PageMaintainScrollPositionOnPostback).

Stratégies de tests

Toutefois, en pratique, il existe des cas pour lesquels la compatibilité ne fonctionne pas. Par conséquent, il est indispensable de tester l'application ou les composants après migration sous la nouvelle version. Plusieurs types d'applications sont à tester :

- Applications client utilisant le code managé
- Applications Web
- Add-in COM
- Applications ClickOnce

Par exemple, pour les applications client, les scénarios de tests comprennent les étapes suivantes :

- Tester l'installation : il s'agit de s'assurer que l'installation de l'application fonctionne encore correctement.
- Vérifier le comportement de l'application : il s'agit de tests fonctionnels.

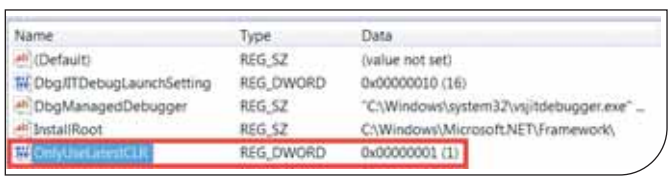
Ces tests sont à effectuer avant l'installation du Framework 4, l'application tournant alors sous l'ancienne version de .NET, ainsi qu'après la migration vers le Framework 4 via le fichier de configuration ou la modification de la clé. Par exemple, pour les applications Web, les scénarios de tests comprennent les étapes suivantes :

- Tester que les applications Web précompilées avec l'ancienne version .NET fonctionnent avec le Framework 4.
- Tester que le code source est compatible entre les anciennes et nouvelles versions pour les applications Web en compilation dynamique.

Le dernier point à tester (et un des plus importants) pour tout type d'application qui l'utilise : la sérialisation. Si le format de sérialisation de données a changé entre les versions du Framework .NET, les applications peuvent ne pas fonctionner correctement. En effet, une application écrite en Framework 2.0 peut ne pas être capable de lire les données sérialisées avec le Framework 4, et vice versa. Le Framework .NET contient des dizaines de classes de sérialisation. Par conséquent, la sérialisation est un domaine qui est particulièrement important à tester !

Conclusion

Nous avons donc vu qu'effectuer la migration n'était pas forcément le plus difficile, étant donné que le Framework 4 est la version de .NET dont la compatibilité descendante est la plus aboutie jusqu'à présent. In-Proc SxS nous garantit en effet que l'installation du nouveau Framework ne nuira à aucune application existante et que tout ce qui est déjà installé sur l'ordinateur fonctionnera aussi bien qu'auparavant, ce qui est un gros soulagement pour les utilisateurs, comme pour les professionnels de l'informatique. Il faut ensuite tirer parti des nouveautés du Framework 4 ! Certains vont pouvoir être utilisées au fur et à mesure dans le projet (comme l'utilisation du late binding, des tuples, de PLINQ, des arguments nommés ou de la programmation par contrat). Mais pour d'autres, il va être nécessaire sur le long terme de reprogrammer une partie du code existant. Par exemple, une couche de données basée sur Entity Framework devra être modifiée pour profiter des améliorations de la V4 (POCO, transparent lazy loading, ...). Bien entendu, si vous n'avez pas le temps de migrer, cela ne vous empêche pas de développer vos nouveaux projets en Framework 4 tout en utilisant l'ancien Framework pour vos autres applications.



■ Jason **De Oliveira**
Solutions Architect chez Winwise, 13 ans d'expérience dans le développement logiciel.
Son Blog: http:// jasondeoliveira.com

■ Christophe **Heral**
7 ans d'expérience dans le monde du développement logiciel. Il est actuellement Consultant/Formateur .NET au sein de la société Winwise.
christophe.heral@winwise.com