



www.programmez.com

programmmez!

LE MAGAZINE DU DÉVELOPPEMENT

100% Pratique !

Tout savoir sur le développement .Net

Les outils • Les frameworks • C# ou VB.Net, le choix du langage

Mobilité

Développer
des applications
Windows Mobile
avec le
Compact
Framework

Bureautique

Développer avec
MS Office 2007

SGBD

Le databinding
de Windows

Actu

Expression :
Microsoft investit
le design web

Web

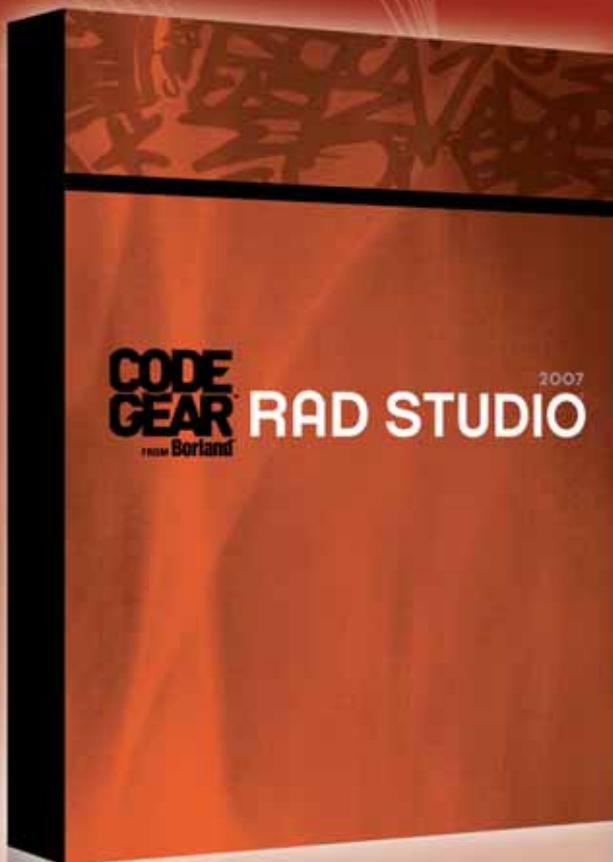
Silverlight :
la révolution de
l'interface graphique



À venir : .Net 3.5, Visual Studio 2008...

M 02104 - 8 H - F: 4,95 € - RD





Découvrez RAD STUDIO 2007

**(Delphi, Delphi.NET et
C++ Builder)
et ses nouvelles fonctionnalités...**

**Support de Vista, support de .NET, ASP.NET 2.0,
nouveau système de bases de données Blackfish...
et des tas d'autres nouveautés !**

CodeGear™ RAD Studio 2007* Architecte est le seul EDI intégrant tous les composants nécessaires pour modéliser et développer des applications natives Windows et .NET 2.0 pour et sous Windows 2000, XP et Vista. Il réunit Delphi®, C++Builder® et des technologies ECO et de modélisation UML dans le même environnement intégré. Il permet de créer des sites Web AJAX dynamiques avec ASP.NET 2.0 et la bibliothèque VCL pour le Web et de construire des applications avancées pour bases de données avec l'architecture de prochaine génération DBX4. Les nouveaux objets ECO™ IV (Enterprise Core Objects) permettent de modéliser et construire des applications pour bases de données sans se soucier de la base sous-jacente.



OPENWAY, distributeur des produits CODEGEAR (division de Borland) depuis 6 ans, devient le représentant unique de la marque en France. Aujourd'hui, nous vous offrons Expertise et Valeur Ajoutée pour vous accompagner dans vos projets.

codegear@open-way.com

www.blog.codegearfrance.com



> Actu

L'actualité en bref 4

> Outils

Expression : Microsoft investit le design web ! 7

> Événements

.Net 3.5, Visual Studio 2008 et consorts 8

> Découverte

Office : développement autour d'une plate-forme utilisateur 12

> Reportage

Centre technique pour les technologies Microsoft 16

> Dossier : Tout sur le développement .Net

Présentation des différentes versions du Framework.Net 19

L'architecture de .Net 21

C# ou VB.Net, le choix du langage 22

La boîte à outils des développeurs 24

Poster : Espaces de noms .Net 2.0 26

Formations et certifications 28

> Technique

Les entrées/sorties avec le .Net Framework 29

> Mobilité

Développement avec le .Net Compact Framework et Windows Mobile 33

> C#

Les bases de LINQ avec LINQ To Objects 39

> SGBD

Le databinding avec WPF 43

> Développement Web

Silverlight : la révolution de l'interface graphique web 46

Rédaction : redaction@programmez.com

Directeur de la Rédaction : Jean Kaminsky

Rédacteur en Chef : François Tonic

Ont collaboré : Van Thomas Le, D. Frontigny, D. Bendejacq, M. Mezil, M. Szablowski, B. Carcel, M. Plasse, S. Leroux, A. Zanchetta, Y. Faure, T. Lebrun, M. Benmostapha, J. Vanderroost, V. Flevez, X. Vanneste.

Photo de couverture : © Sagel & Kranefeld/zefa/Corbis

Maquette : AJE Conseils

Publicité : Régie publicitaire, K-Now sarl. Pour la publicité uniquement : Tél. : 01 41 77 16 03. coordination@programmez.com

Editeur : Go-02 sarl, 6 rue Bezout - 75014 Paris Coordination@programmez.com - Dépôt légal : à parution - Commission paritaire : 0707K78366 - ISSN : 1627-0908 - Imprimeur : ETC - 76198 Yvetot

Directeur de la publication : Jean-Claude Vaudecrane

Abonnement : Programmez 22, rue René Boulanger, 75472 Paris Cedex 10 - abonnements.programmez@groupe-gli.com Tél. : 01 55 56 70 55 - Fax : 01 55 56 70 20 - du lundi au jeudi de 9h30 à 12h30 et de 13h30 à 17h00, le vendredi de 9h00 à 12h00 et de 14h00 à 16h30. **Tarifs abonnement** (magazine seul) : 1 an - 11 numéros France métropolitaine : 45 € - Etudiant : 39 € - CEE et Suisse : 51,83 € Algérie, Maroc, Tunisie : 55,95 € Canada : 64,33 € Tom : 79,61 € - Dom : 62,84 € Autres pays : nous consulter. **PDF :** 30 € (Monde Entier) souscription en ligne.

Programmez!
LE MAGAZINE DU DÉVELOPPEMENT



Soyez net sur .Net

À son lancement, la plate-forme .Net avait laissé une impression mi-figue, mi-raisin. Comment la comprendre ? Comment la positionner ? Microsoft avait surtout misé sur

la partie serveur au printemps 2000. Et il a fallu attendre quelques mois avant de voir réellement émerger .Net au niveau client et développeur. .Net 2 a constitué une étape importante car il a réellement lancé .Net en entreprise et chez les développeurs. Plus mature, .Net 2 constitue une base solide. .Net 3 a juste rajouté de nouvelles briques, sans modifier les langages. La prochaine étape importante sera .Net 3.5 qui sortira dans un peu plus de 4 mois. Pour la première fois, depuis plusieurs années, Microsoft touche les langages C# et VB.Net, avec notamment l'apparition de Linq, un langage de requêtage. Et encore au-delà, c'est l'intégration des langages dynamiques, voire dans un futur incertain, du fonctionnel. D'ailleurs, la DLR de Silverlight 1.1 montre la voie de cette évolution pas si éloignée que cela, peut-être dans un .Net 4.0 en 2009 ou 2010 ? Et qui sait, peut-être verra-t-on le premier framework .Net (hormis le Micro Framework) se passer de tout système d'exploitation avec une fonction bootstrap. Car l'avenir de .Net, dans son ensemble, n'est pas, n'est plus lié à Windows.

En sept ans, .Net a donc considérablement évolué. Il fonctionne sous Windows, sous MacOS X et Linux avec le projet Mono, sur les terminaux mobiles avec le Compact framework et dans les systèmes enfouis avec le Micro Framework. Les outils ont suivi cette même évolution. La version 2005 avait été une étape importante pour la maturité de la plate-forme de développement, suite à une v2003 lourde et pas toujours stable. La cuvée 2008 s'annonce mineure sur les nouveautés, mais importante sur la partie .Net 3 et Web.

À Programmez, nous nous intéressons à .Net depuis ses débuts ! C'est tout naturellement, que nous avons créé ces hors séries 100 % .Net ! Programmez .Net s'adresse à tous les développeurs .Net ou s'intéressant à la plate-forme. Vous y trouverez des articles de codage (C#, VB, Web, SQL Server, etc.), des interviews, toute l'actualité .Net, les événements ! Que vous soyez expert ou nouveau converti, Programmez .Net est fait pour vous ! Tous les deux mois, faites le plein de .Net !

Programmez .Net est *votre* magazine. Vous voulez partager une astuce de programmation, votre expérience ou tout simplement écrire sur une fonction .Net, contactez-nous sur redaction@programmez.com

■ **François Tonic**
Rédacteur en chef
ftonic@programmez.com

COMPOSANT

Infragistics renforce son offre WPF



L'éditeur de composants .net propose aux développeurs depuis quelques semaines une version gratuite d'une datagrid pour WPF. Il s'agit de la solution NetAdvantage for WPF Express. Elle inclut une version (gratuite) totalement fonctionnelle du XamDataGrid. Ce composant peut être utilisé comme tableur, pour réaliser des rapports, des tableaux, des listings, etc. À l'origine, il était disponible uniquement aux souscripteurs d'un abonnement MSDN. D'autre part, l'éditeur a décidé de renforcer sa prise en compte de .Net 3.0, et notamment de WPF, en dévoilant une application de référence : Tangerine. Ce projet doit montrer les meilleures pratiques autour de WPF, sur les guides d'interface, le codage, etc. Tangerine est une librairie virtuelle. Il est possible de personnaliser le modèle de données et on dispose d'un moteur de recherche, d'une navigation, de vue multiple d'un objet, etc. Tangerine est librement téléchargeable. Site : www.infragistics.com

ATELIER

PowerBuilder 11 se met à .Net

L'environnement de développement PowerBuilder continue lui aussi d'évoluer. Actuellement, en version 11, il prend maintenant en compte .Net et permet de déploiement des applications PowerBuilder en .Net (webform et windows form). .Net est aussi supporté dans le déploiement d'objets non visuels (les NVO). Il est aussi possible d'importer les assemblies ou encore de créer un DataWindow sur un web services (pour la source de données). L'éditeur le présente comme une alternative à Visual Studio. La prochaine version est prévue courant 2008 avec PowerBuilder 11.1, et une prise en compte toujours plus accrue de .Net. Ce support est stratégique car il permet de conforter le parc installé de PowerBuilder et de pérenniser l'investissement en prenant en compte les nouvelles technologies.

SERVEUR

Home Server arrive dans les maisons

Le serveur personnel signé Microsoft, Home Server, est disponible en version finale depuis septembre dernier, les premières machines arrivent sur le marché. Home Server se base sur Windows Server 2003 et rajoute des fonctions spécifiques pour la configuration, la sauvegarde logicielle des informations, l'accès à distance, etc. L'administration a été simplifiée au maximum pour "aider" l'utilisateur à installer et à paramétrer les différentes fonctions. Cependant, Home Server, dans sa première ver-

sion, demande quelques connaissances réseaux et d'administration. Un serveur n'est pas un système normal... Côté développeur, Microsoft propose depuis ce printemps un Home Server SDK qui a évolué avec le développement de Home Server. Il s'agit d'un jeu d'API pour créer et déployer des Add-in dans la console Home Server et capable d'interagir avec les éléments du système. On peut avec ce SDK : rajouter une "console tab", une page paramètres dans la console, ajouter site web sur le serveur, envoyer des notifications sur les ordinateurs du réseau. Home Server SDK est en code managé et s'appuie sur le SDK de Windows Server 2003. Le déploiement se fait par un fichier MSI. Ce jeu d'API permet d'accéder aux dossiers partagés, aux ordinateurs clients, au dossier application du serveur ainsi qu'aux disques durs, au backup, au système de notification... Le développement se déroule sous Visual Studio, avec une simple version Express, et le framework .Net 2.0. On peut coder en C#, VB, C++. Aujourd'hui plusieurs dizaines d'add-in sont disponibles.

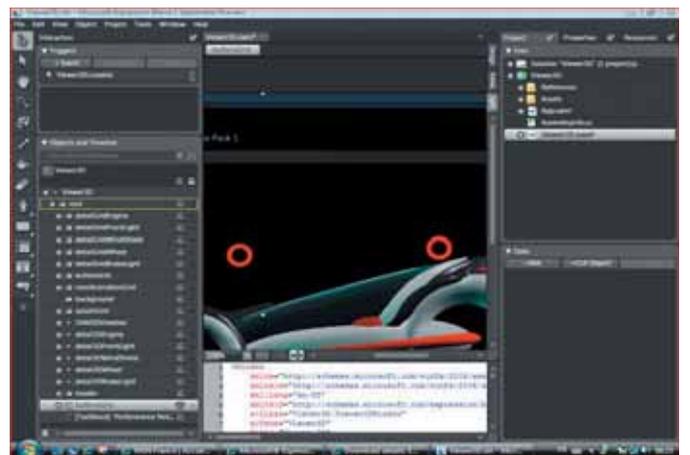
DOCUMENTATION

Le projet Sandcastle dans une nouvelle version

Microsoft a livré une nouvelle version technique du projet Sandcastle. Il s'agit d'un outil de documentation, dans le style de MSDN, sur les sources des assemblies et pouvant aussi intégrer les commentaires XML. Il supporte .Net 2 et les génériques et se compose de deux éléments : le MrefBuilder et Build Assembler.

IDE

Expression Blend 2 continue d'évoluer



Microsoft travaille activement à la version 2.0 d'Expression Blend alors que la version 1.0 vient de sortir. Cette version constitue une étape importante car elle permet d'intégrer la gamme Expression avec Silverlight 1.1 qui assurera le support .Net et de la DLR (une CLR prenant en compte les langages dynamiques) et la compatibilité avec Visual Studio 2008. On peut ainsi construire des projets WPF et il supporte aussi le futur .net framework 3.5.

De nombreux éléments seront améliorés dans cette mouture : fonctions de refactoring, vue mixte améliorée éditeur XAML et design, remplacement de l'actuel storyboard par un storyboard Picker (une petite étiquette indiquera le nom du storyboard sélectionné), possibilité d'embarquer dans les projets les polices de caractères. Blend 2 sortira avant la fin de l'hiver (février ou mars 2008 sans doute).



Microsoft®

Visual Studio® 2005

avec

msdn®

Microsoft® Developer Network



*Voici l'occasion de vous équiper !
Profitez maintenant d'une réduction supplémentaire de*

-15%*

De plus, grâce à votre abonnement MSDN Recevez GRATUITEMENT avant tout le monde !



Microsoft®
Visual Studio® 2008

"ORCAS"



Microsoft®
Windows Server® 2008

"LONGHORN"



Microsoft®
SQL Server® 2008

"KATMAI"

En effet, grâce aux mises à jour mensuelles ***vous recevez avant tout le monde les versions Beta et finalisées*** des logiciels contenus dans les abonnements MSDN.
Contient déjà Windows Vista et Office 2007 (en version Premium)

Visual Studio 2005 Pro avec MSDN Pro



520 €^{HT} par an et par développeur
Soit 1040 €^{HT} pour l'abonnement 2 ans

Visual Studio 2005 Pro avec MSDN Premium



1185 €^{HT} par an et par développeur
Soit 2370 €^{HT} pour l'abonnement 2 ans

Soit environ 50 % d'économie par rapport au prix boîte

*Promotion valable jusqu'au 31 décembre 2007 sur toute nouvelle licence Open de Visual Studio 2005 avec un abonnement MSDN Premium ou MSDN Professional. Valable dans tous les modes de licences Open (Open Business, Open Value et Open Value Souscription) Abonnement d'une durée de 2 ans

COMSOFT direct

Bechtle's Software Specialist



Microsoft®
GOLD CERTIFIED

Partner

Contactez vite nos spécialistes MSDN au 04 89 87 22 45
msdn@sosdevelopers.com • www.sosdevelopers.com/msdn.htm

NOUVEAUTÉS

2008 : préparez-vous !

Attention, chaud devant ! L'ambition de Microsoft est proportionnelle au défi informatique d'aujourd'hui et de demain. Et le dernier trimestre 2007 a débuté avec la disponibilité de la gamme Expression, en Français. La gamme comprend : Expression Web, Blend, Media et Design. Une suite est disponible sous le nom Expression Studio (voir page 7).

Le premier semestre 2008 sera intense : Windows Server 2008, SQL Server 2008, Visual Studio 2008, framework .Net 3.5 (voir page 8), et surtout Silverlight 1.1 et sans doute la sortie très proche des nouvelles versions de la gamme Expression. Silverlight



1.1 est très attendu car cette moulture étend considérablement les possibilités de Silverlight, avec le support de .Net et de la DLR pour les langages dynamiques. Ce sera le véritable coup d'envoi de l'offre RIA de Microsoft et d'applications web nouvelle génération. Sur le front des services en ligne, Live arrivera avec de nouvelles versions d'API et de services. On verra ainsi apparaître Office Live 2.0. Avec la disponibilité récente de Windows Live et d'API enfin stabilisées, l'offre Microsoft commence réellement à proposer une alternative aux solutions Google. Côté mobilité, en 2008 ou 2009, Windows Mobile verra une nouvelle version majeure arriver et surtout des services Live mobiles ! Pour le moment, l'éditeur ne communique pas sur son contenu. Mais un effort sera fait pour synchroniser la sortie de Windows Mobile et de Windows CE.

S+S = SaaS

Avec l'arrivée massive des services en ligne et de la transformation des logiciels en services, dans la mouvance SaaS, Microsoft se devait de présenter une approche globale. Pour se diff-

rencier, l'éditeur propose la stratégie : Software + Services. S + S doit coupler les logiciels et les services dans un ensemble unique quel que soit le terminal et le mode d'utilisation (connecté et déconnecté). Au cœur de cette approche, 4 services majeurs : services aux personnes (Popfly, Virtual Earth, etc.), services aux développeurs, services aux entreprises et les services aux intégrateurs. Le rôle du développeur est d'ailleurs central pour Microsoft qui propose un kit de développement Windows Live ou encore des services Web 2 comme Streaming Silverlight. Mais la réussite de S+S passe par le dynamisme de l'écosystème, d'où un appel du

pied aux partenaires. Reste à définir un modèle économique équitable entre Microsoft et les fournisseurs. Un des exemples donnés pour montrer les capacités du S + S est la communication unifiée avec différents modes d'accès (web mail, mailer, accès vocal, terminaux mobiles), au cœur, les serveurs Exchange et Office Communications Server. Le futur Live sera-t-il capable de contrer l'offensive Google sur les services et applications en ligne ?

Et Microsoft soutient Moonlight

L'éditeur a aussi annoncé la reconnaissance officielle de Moonlight menée avec le projet Mono. Il s'agit de porter sous Linux, Silverlight 1.1. Si techniquement, cela est possible, le travail n'en demeure pas moins important à fournir. Microsoft apportera un soutien technique, des ressources. Avec Moonlight, les applications Silverlight fonctionneront sous Windows, MacOS X, Linux !

Agenda .Net

TechEd Développeurs du 5 au 9 Novembre 2007, Barcelone, Espagne

La plus importante des conférences de Microsoft dans la zone EMEA (Europe, Moyen-Orient, Afrique) se tient cette année de nouveau à Barcelone. Elle a pour but de fournir aux développeurs une formation technique poussée, des informations et des ressources pour leur permettre de construire des logiciels avec les outils de développement de Microsoft pour les technologies actuelles et futures.

4 000 développeurs sont attendus. Attention, les conférences se tiennent en anglais. L'inscription coûte 2 245 € HT <http://www.mseventseurope.com/teched/07/developers/Content/Pages/Default.aspx>

TechDays 2008 du 11 au 13 février

: 3 jours de conférences, plusieurs centaines de sessions techniques, des dizaines d'exposants, présence des communautés. Plus de 10 000 personnes attendues ! Avec présentation des nouveaux outils

27 février 2008 : lancement mondial de Windows Server 2008, SQL Server 2008 et de Visual Studio 2008 !

Mix Paris 2008 juin : déclinaison française de l'événement Web 2 / RIA de Microsoft.

Nouveautés livres

.Net 3.0 et Expression tiennent la vedette chez Dunod :

- **Prendre en main Microsoft Expression Design** : les auteurs font découvrir Design et les manipulations de base des fonctions, très visuel et pratique. Orienté designer.

- **Initiation à Microsoft Expression Blend** : comment maîtriser les animations Blend, comment utiliser au mieux XAML et les relations avec Visual Studio ?

Et aussi : **WPF et Silverlight, Windows PowerShell kit d'administration, Ecrire du code sécurité pour Windows Vista, ASP.Net Ajax**. Ces ouvrages seront disponibles d'ici fin novembre.

Expression : Microsoft investit le design web !

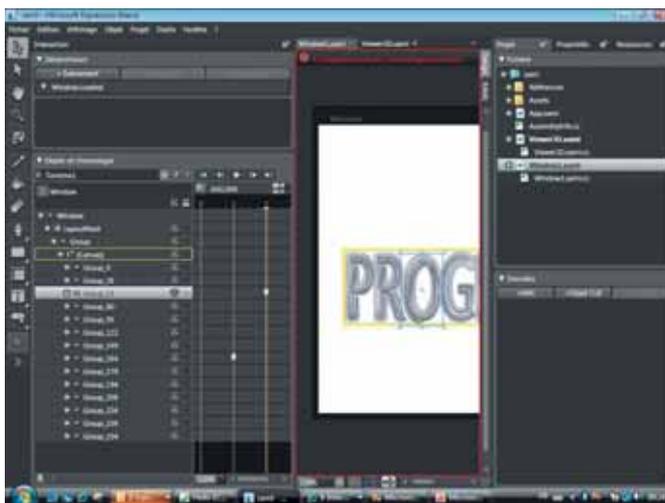
Depuis cet été, la gamme Expression est disponible en version 1.0 et en Français ! Première incursion dans le design web, l'éditeur tente le pari de se faire une place sur un marché largement dominé par Adobe. Dans le même temps, les développeurs auront enfin un éditeur web à la hauteur du web 2.0 avec Expression Web, successeur de FrontPage.

Expression Blend

Souvent décrit comme le "Flash Killer", Blend est l'outil de création d'interfaces riches, interactives. Il s'appuie sur XAML et s'intègre étroitement avec Design pour la partie graphique, visuelle. Il peut être utilisé par un designer avancé ou un développeur web. L'interface de l'outil est écrite en XAML / WPF. L'outil est plutôt réactif, à défaut d'être immédiatement intuitif. Une phase d'apprentissage est nécessaire même si un concepteur Flash retrouvera assez rapidement ses réflexes. On réalise en quelques étapes une puissante interface animée. Les éléments graphiques sont créés dans Design, que l'on importe sans difficulté, puis on les manipule dans Blend, par événement et/ou en définissant les animations. Si on veut aller plus loin, il faut passer par le code XAML. On peut l'éditer dans Blend mais ce n'est pas un éditeur de code. C'est pour cela que Visual Studio 2005 Standard est inclus en standard dans Expression Studio. L'outil possède un potentiel énorme. Blend 2, actuellement en pré-version, franchit une nouvelle étape en intégrant Silverlight. L'absence de celui-ci est l'un des points faibles de Blend 1.0.

Expression Design

Design est un signe fort envers les graphistes et designers web ! Et un moyen de faire collaborer le développeur et le designer dans un projet unique. Il permet de créer des objets graphiques, des interfaces pour les applications desktop (de type WPF) et web.



L'outil Blend

Totalement vectoriels, les objets créés sont ensuite manipulables dans Blend avec une souplesse peu commune. Les designers pourront laisser libre cours à leur imagination. Les options disponibles comme les effets sont d'une efficacité redoutable ! Vous pouvez exporter vos créations directement en XAML.

Expression Media / Encoder

Expression Media est le seul outil fonctionnant sous Windows et MacOS X. Il s'agit d'un gestionnaire professionnel de contenus comme l'est un Adobe Bridge dans la CS3. Il permet d'organiser, de chercher les éléments graphiques, vidéos, animations. Très pratique quand on gère des milliers d'images.

Mais Media est aussi une gamme d'outils qui va s'agrandir rapidement. Media Encoder est l'environnement permettant de créer des présentations, des galeries web et vidéo en quelques

minutes. On peut y encoder des vidéos et les publier en format Silverlight, par exemple dans Silverlight Streaming ! Il deviendra rapidement le compagnon de la gamme Expression et des développeurs / designers silverlight. Actuellement en pré-version.

Expression Web

Successeur de FrontPage, il est le nouvel outil de développement de sites web de Microsoft. Il se veut plus respectueux des standards Web. Il prend en compte les CSS (1.0, 2.0 et 2.1), XHTML, les guides d'accessibilité du W3C. Les CSS constituent le gros morceau d'Expression Web avec un nouvel éditeur plus graphique et plus simple pour créer et modifier des CSS. Notons que quelques parties de l'éditeur sont restées en anglais. Sur la partie contenu riche, les flux RSS ont été remis à plat pour en faciliter l'intégration dans les pages. XSLT est présent par défaut. Sur la partie serveur, ASP.Net a été largement modi-

Expression Studio

Pour diffuser rapidement ses nouveaux outils, Microsoft propose un package complet : Expression Web, Blend, Design, Media, Visual Studio 2005 Standard et un DVD de tutoriaux. Le prix officiel est de 799 euros.

fiée. On bénéficie de toutes les avancées de la v2, comme les master pages, les contrôles ou encore le data binding "automatique".

Sur le côté extensibilité, nous ne disposons pas de plug-in comme pour Dreamweaver. Il y a tout de même la possibilité d'ajouter des templates. D'autre part, Expression Web se limite principalement aux technologies Microsoft (pas de PHP par exemple) et l'intégration d'applications Silverlight, de code XAML n'est pas encore disponible.

Expression n'est pas (encore) un concurrent frontal aux solutions Adobe. Mais avec la version 2.0, l'intégration de Silverlight et avec Visual Studio 2008, la situation changera. Le fait de pouvoir intégrer les développeurs et les designers constitue bien une étape importante dans le futur du développement web et desktop. Microsoft réfléchit en ce moment à proposer un programme MSDN pour les designers autour d'Expression. La communauté va jouer un rôle important et l'éditeur ne manquera pas de les inciter à promouvoir cette gamme.

■ François Tonic

.Net 3.5, Visual Studio 2008 et consorts

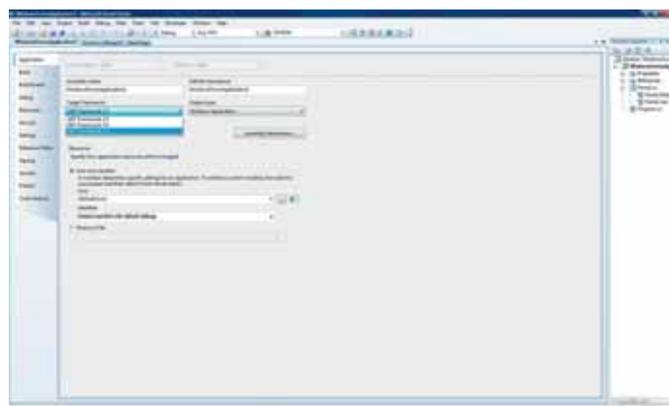
Dans 5 mois, le développeur .net / Windows aura la possibilité d'utiliser un nouveau .net, un nouveau Visual Studio. La sortie de .Net 3.0 avait semé le trouble car de fait, les langages n'avaient pas bougé, ni la CLR 2.0, seules de nouvelles bibliothèques se rajoutaient à l'ensemble. On peut dire que .Net 3.5 est le véritable .Net 3.0 ! Microsoft a introduit des modifications, parfois profondes, dans les langages. " Il s'agit d'une release majeure " confirme Eric Mittelette (Relation Technique Développeur - Microsoft France).

Sur les outils, Microsoft a décidé de synchroniser le framework et Visual Studio qui arrivera en version 2008. La sortie de Windows Vista avait semé une petite zizanie chez les développeurs car Visual Studio 2005 n'était pas idéal pour le nouveau système. " Avec Vista, nous avons un nouveau SDK et les nouveautés WPF, WCF, etc. On a aussi UAC et Visual Studio n'en est pas toujours un bon compagnon, comme sur les connexions base de données, " poursuit E. Mittelette. Même si VS 2008 demeure une évolution assez importante, il s'agit d'une version mineure comparée aux nouveautés du framework et des langages.

.Net 3.5 : des langages qui évoluent !

Cette future version permet d'unifier le framework entre les nouveautés des langages, .Net 2 et .Net 3. Ainsi, si la CLR évolue peu, on bénéficie de C# 3.0, de VB 9, d'un nouveau langage de requêtage : Linq, qui se place au-dessus des langages. On bénéficie aussi d'un nouvel ADO.Net et enfin, se rajoutent les couches WPF, WCF, Workflow et CardSpace.

Un des gros avantages de C# 3.0 et de VB 9 est de réduire considérablement la taille du code. Cela permet certes de gagner en productivité à l'écriture, mais cela permet surtout de gagner en lisibilité et donc en maintenance. Prenons un comparatif avec les commentaires. Dans un programme si chaque ligne est commen-



tée alors le commentaire pollue plus qu'autre chose. En revanche, si les commentaires ne sont utilisés que pour certains points difficiles ou pour justifier certains choix techniques, ils sont très intéressants et on gagne en maintenance. Les nouveautés de C# 3.0 et VB 9 vont permettre de réduire le code " simpliste ".

Automatic properties

Les automatic properties constituent un excellent exemple de réduction du code. Dans le cas d'une propriété qui ne ferait qu'encapsuler un champ, le code du *get*

et du *set* n'apporte rien. De plus, le champ lui-même ne sert à rien puisqu'il y a la propriété. Ce code est d'ailleurs tellement simple qu'il peut être généré directement par refactoring. Aussi, le compilateur C# 3.0 se propose de faire le travail pour vous.

Object initializer

Avec C#3.0 (et VB9), il sera possible d'instancier une classe et d'initialiser cette instance en une seule instruction, sans modifier le code de la classe à instancier.

Collection initializer

Tout comme les object initializers,

les collection initializers permettent d'instancier et d'initialiser en une seule instruction. Les collection initializers s'appliquent à n'importe quelle classe implémentant `IEnumerable<T>` et possédant une méthode `Add`.

Anonymous type

Les anonymous types sont utilisés pour définir des types anonymes comme leur nom l'indique. Mais qu'est-ce qu'une classe anonyme ? En fait, c'est une classe qui se limite à des propriétés toutes en *get / set public* et qui sera générée par le compilateur. Comme nous le verrons tout à l'heure, les anonymous type sont surtout utilisés par LINQ pour faire une vue.

Lambda expression

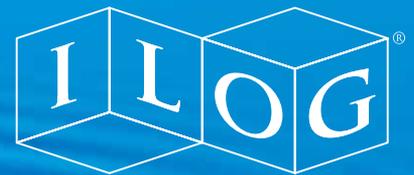
Les lambda expressions sont une simplification des délégués anonymes.

Partial methods

Les partial methods permettent d'appeler une méthode ne renvoyant rien, en n'ayant défini que sa signature. Si la méthode n'est pas redéfinie dans la classe, son appel ne fera rien.

L'intérêt des partial methods repose essentiellement dans les partial class générées par un designer. Prenons l'exemple du designer associé à LINQ To SQL. Il va générer des classes partielles d'entités contenant des propriétés associées aux colonnes de la table SQL associée à la classe. Avec les partial methods, il va être possible, depuis un autre fichier associé à cette même classe, de valider la valeur donnée au set de ces propriétés.

**Maintenant vos applications
les plus exigeantes auront
un impact visuel maximum**



Changing the rules of business™

**Nouveau ILOG Diagrammer
pour ASP.NET AJAX et Windows forms**

www.ilog.com/products/diagrammernet

LINQ

Le requêtage SQL est différent du requêtage XML. LINQ permet d'unifier le requêtage. De plus, il est ouvert et n'importe qui peut se créer un provider LINQ, il y a d'ailleurs beaucoup d'exemples sur le web. Avant LINQ, la plupart du temps, l'accès aux données se faisait via un String. Par conséquent, la validité de ces derniers ne s'effectuait qu'à l'exécution. Avec LINQ, la vérification se fait dès la compilation. De plus, l'IntelliSense propose une aide très appréciable lors de l'écriture de la requête. Nous reviendrons rapidement sur Linq.

Compact framework 3.5

Notons aussi que le Compact Framework évolue en même temps afin d'être synchronisé. De nombreuses nouveautés :

- intégration d'un sous-ensemble de WPF
 - disponibilité partielle de Linq (Linq to Objects, Linq to XML, Linq to DataSet)
 - apparition de nouveaux outils comme un CLR Profiler (analyse de la heap), outil de configuration pour les logs de diagnostics
- D'autre part, on bénéficie d'une meilleure compression, d'une partie audio améliorée, le meilleur support de Windows Form, certificats de type Client-Side ! Sous VS 2008, on bénéficiera de nouveaux émulateurs et tous les terminaux supportant Compact Framework 2 seront supportés par cette version.

Visual Studio 2008

Sans doute le morceau le plus attendu ! Ce sera le premier IDE Microsoft réellement natif Vista, hormis Expression, ce qui devrait simplifier le développement natif en C++. On bénéficiera d'un design XAML (nom de code : cider), " même si Blend reste l'environnement de référence " précise Éric Mittele. Le développement Office est désormais totalement supporté avec l'intégration de VSTO. Ce sera bien entendu la plateforme de référence pour créer des applications Vista. Les fonctions de test ont été renforcées, notamment sur le test unitaire, les tests de montée en charge. Le tuning sera aussi présent.

DLR, avenir de VB, multicore

Avec l'annonce prématurée de Silverlight 1.1, alors que la 1.0 n'était même pas disponible, Microsoft a dévoilé la disponibilité prochaine de la DLR. Il s'agit d'une CLR .net capable d'exécuter des langages dynamiques tels que Ruby ou VB dynamique. La DLR ne sera pas disponible dans .Net 3.5. Le premier à en bénéficier sera Silverlight 1.1. Mais elle sera tôt ou tard intégrée par défaut dans le framework, sans doute dès la version 4.0 ! Si on ne parle pas aujourd'hui de C# dynamique, pourquoi ne pas l'envisager à terme. Là aussi se pose la question de l'avenir de VB sur

.Net. Quelle valeur ajoutée a-t-il par rapport à C# ? Aujourd'hui, des discussions s'articulent sur la présence de VB. Faut-il le garder comme langage à part entière ou alors le mettre dans un atelier de développement de type WinDev ? Enfin, le multicore est au cœur des stratégies Microsoft. Déjà aujourd'hui, l'éditeur travaille à la possibilité de paralléliser les requêtes Linq avec le projet PLINQ et des bibliothèques de type Parallel FX Library. Cela permet de simplifier le codage d'ap-

plication pour fonctionner sur plusieurs cœurs et éviter de se soucier de la gestion des threads. La vectorisation devrait se réaliser dans le code et non dans le compilateur comme on peut l'avoir avec GCC 4.x.

■ François Tonic

En collaboration avec

■ Matthieu Mezil - Ingénieur

Consultant chez Winwise

Email : matthieu.mezil@winwise.fr

Site : www.winwise.fr

brèves

IDE

CodeGear RAD STUDIO 2007 disponible

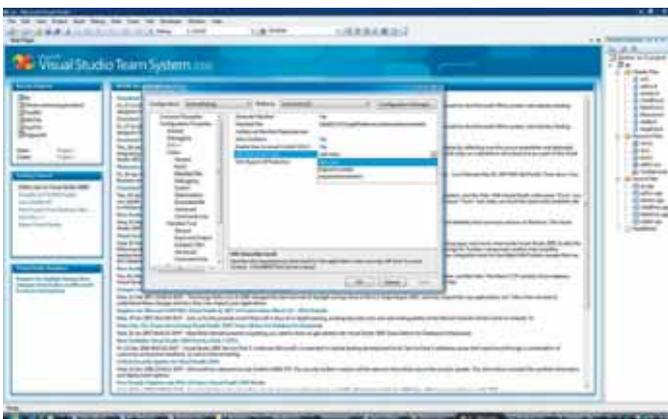
CodeGear RAD Studio 2007 est une mise à jour significative pour les développeurs Delphi et C++. Il supporte le développement natif Windows ainsi que .NET dans un même environnement. Développez des applications Web, multi niveaux (n-tiers), pour bases de données sous Windows 2000, XP ou Vista et déployez-les ensuite librement sur la plate-forme Windows de votre choix. Les principales fonctionnalités sont :

- Développement des applications natives Windows compatibles Vista – avec support des thèmes Windows, des effets de transparence, et des nouvelles boîtes de dialogue Vista.
- Développement AJAX avec la VCL pour le web.
- Nouveau système MSBUILD pour simplifier le process de gestion des " builds ".
- Rationalisation des connexions aux bases de données avec la nouvelle architecture DBX 4.
- Blackfish™ SQL, une base de données SQL entièrement administrée, portable et intégrable qui peut être déployée librement avec votre application.
- Améliorations C++ Builder : tests unitaires C++ intégrés (support de DUnit), meilleures performances de construction intra-EDI (désormais plus ou moins aussi rapides que par ligne de commande), Indy 10...
- Support des Generics pour Delphi .Net
- Support de .Net 2

COMPOSANT

Ilog sort Diagrammer

L'éditeur français propose Diagrammer pour .Net. Il s'agit d'un outil de création de diagrammes simples et complexes que l'on intègre dans les Win et Web Forms. On peut créer toutes sortes de représentations. Il est compatible avec ASP.Net Ajax, et dispose de trois outils d'édition pour construire des éditeurs de modélisation, de processus métiers et de diagrammes UML. Il dispose aussi d'un SDK incluant des exemples et est accessible à tous les langages .Net. Site : www.ilog.fr



L'expertise C# pilotée par le modèle UML

Objecteering 6 optimise MDA et UML2 pour générer un code C# d'un haut niveau d'expertise : il maximise la productivité et la qualité des développements C#.

Comment tirer parti au mieux de la modélisation UML à des fins de production automatisée d'un code de qualité, maintenu en cohérence avec le modèle ?

L'approche MDA qui consiste à exploiter le modèle par des mécanismes de transformation répond précisément à cette problématique en assurant également la traçabilité entre le code généré et les modèles. Avec Objecteering 6, Objecteering Software met à disposition des développeurs C# une nouvelle génération d'outils de développement guidé par le modèle, en s'appuyant sur les dernières avancées de MDA de UML2.1 et de C#.

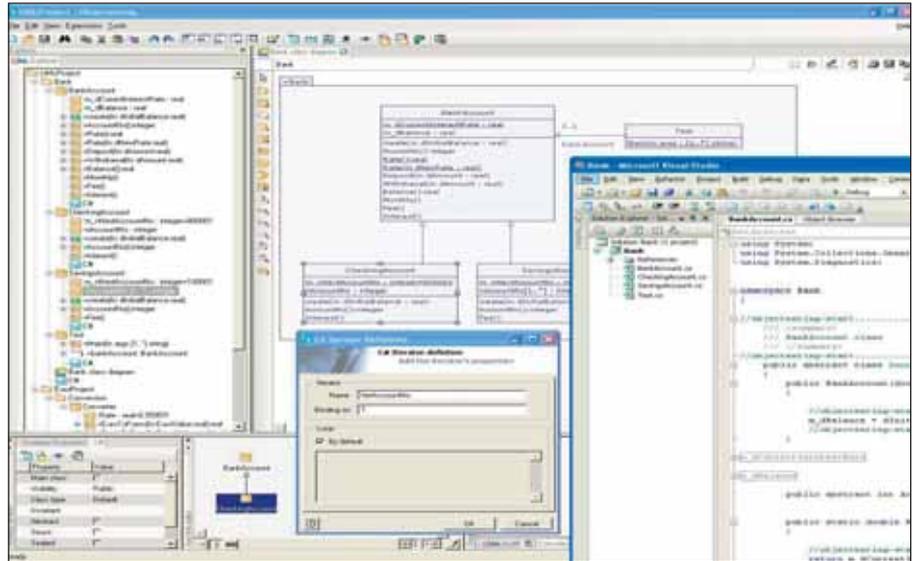
Un haut niveau de services de guidage et d'automatisation dédiés C#

Objecteering 6 guide et automatise le développement C# en apportant la puissance de UML et MDA à la plate-forme Visual Studio .Net, de Microsoft. Il apporte :

- Une production automatisée du code C#, toujours cohérente avec le modèle,
- Le support de patterns de conception pour plus de productivité,
- Une production de documentation .Net automatique, pour une aide en ligne HTML, MSDN ou Visual Studio.
- Un support complet de C# 2.0 à partir de UML2
- Une génération automatisée de la chaîne de production (Makefile) à partir du modèle
- Un support dédié du travail en équipe, avec des " composants de modèles " assurant la diffusion de parties applicatives C#/UML vers d'autres équipes.
- Un reverse engineering C# (source et .net assembly) pour reprendre les applications existantes
- Un paramétrage puissant et aisé, à l'aide du générateur, permettant d'adapter la génération de code C# à son contexte avec la technologie MDA.

La richesse de UML2 traduite en C#

Objecteering 6 exploite au mieux les nouvelles capacités du standard UML 2.1. Les templates UML2 permettent ainsi efficacement de modéliser et générer les generics C#. La gestion des références d'assemblies, couplée à un service de contrôle de cohérence sur leur bonne utilisation permet de gérer finement les modes d'import C# avec leurs diverses options d'emploi. Les associations sont pleinement exploitées,



Objecteering 6 : Diagramme de classes, assistant C# et code généré.

pour déduire les librairies de container appropriées. Les attributs sont supportées à partir du modèle. Une assistance experte C# apporte un gain de qualité et de productivité au niveau du modèle. Par exemple, des règles de nommage par défaut sont supportées et la gestion des event, delegate, property, indexer et C# container est intégrée à l'outil.

Une gestion de cohérence modèle/code permanente

Avec Objecteering 6, le modèle est le code : le développeur complète le modèle UML par l'algorithme des méthodes, tout en étant guidé dans l'outil. La modélisation est intégralement exploitée, l'utilisateur n'ayant jamais à reformuler le modèle en C#. Les compléments de code sont attachés aux éléments du modèle, stockés dans le référentiel. A chaque instant la cohérence du modèle UML et du code C# est garantie. Le développeur peut ainsi éditer son application au niveau modèle sous Objecteering ou au niveau code sous Visual Studio: quand le modèle est modifié, Objecteering régénère le code de façon incrémentale et quand le code est modifié, il assure la mise à jour du modèle via son référentiel.

Une ergonomie dédiée au programmeur C#

Au sein du modèle UML, le développeur bénéficie d'une interface utilisateur spécifique C# : il retrouve les notions C# (iterator, override, sea-

led, etc.), dispose de boîtes de saisie dédiée, et de diagrammes UML spécifiques. Il dispose de services dédiés C# (Ndoc, patterns, etc.) pour rendre son modèle encore plus productif.

Les gains MDA au-delà du code

Architecturé depuis l'origine pour une approche MDA optimisée, Objecteering permet de combiner les avantages du modèle, son indépendance des plateformes cibles et la productivité en programmation. Son support multi-langages permet de réaliser des développements combinant C#, C++, Java, SQL, etc. Le support intégré de l'analyse des besoins et du dictionnaire permet d'assurer une traçabilité maintenue depuis les exigences jusqu'au code dans un même environnement.

Objecteering
SOFTWARE

Objecteering Software, éditeur de l'atelier Objecteering 6, est le spécialiste français UML/MDA pour le développement d'applications guidé par le modèle.

Sa suite d'outils couvre le cycle de vie de la gestion des exigences jusqu'au déploiement de l'application pour les cibles Java/J2ee, C#.Net, C++, SQL, Corba et Fortran.

Pour plus d'informations :

www.objecteering.com

info@objecteering.com

Tél. : 01 30 12 16 60



Office 2007 : développement autour d'une plate-forme utilisateur

Office System dans sa version 2007 a étendu sa portée, version après version, pour proposer aujourd'hui un ensemble d'outils de collaboration (OneNote, Groove, Outlook, SharePoint, Communicator), de gestion (Project) ou de publication avec Publisher. Voyons dans cet article comment procéder pour développer autour et avec cette suite bureautique.

Avec l'arrivée de technologies et d'outils comme VSTO, Open XML et Visual Studio 2008, Office est devenu une véritable plate-forme sur laquelle les entreprises peuvent proposer des solutions à l'endroit le plus stratégique pour une entreprise : directement dans les applications critiques dont se servent les utilisateurs quotidiennement.

L'intégration forte proposée par l'extensibilité de la plate-forme Office et l'intégration des outils de création dans les environnements de développement Visual Studio 2005 et 2008 font d'Office System une solution de choix pour la création et le déploiement de solutions métiers sur les postes utilisateurs.

Dans cet article, nous nous limiterons à une présentation de VSTO dans sa dernière édition (VSTO 2005 Second Edition), du nouveau format Open XML et de l'utilisation de Visual Studio 2008 dans ce contexte. A l'heure de la rédaction de cet article, Visual Studio 2008 est encore en version Bêta 2 et Open XML dans sa version ECMA-376 v1.0.

Etendre Office 2007 avec VSTO

L'arrivée de VSTO 2005 SE – Visual Studio Tools for Office – a permis aux développeurs .NET d'être capables de développer des add-in, véritable extension des applications clientes, pour les plates-formes Office 2003 et 2007. Cette version de VSTO offre ainsi la possibilité de créer des extensions pour Word, Excel, PowerPoint, Visio, Project et Outlook, tout en constatant que seul InfoPath dans son édition 2007 est supporté (cf. Figure 1 - Les projets VSTO disponibles dans Visual Studio 2008).

Bien que faisant l'objet d'une extension à Visual Studio 2005, l'environnement de développement VSTO est aujourd'hui intégré directement dans Visual Studio 2008. On regrettera néanmoins dans cette dernière version, l'absence de création automatique de projet de déploiement, ce qui rend la tâche encore plus ardue qu'elle ne l'était avec Visual Studio 2005.

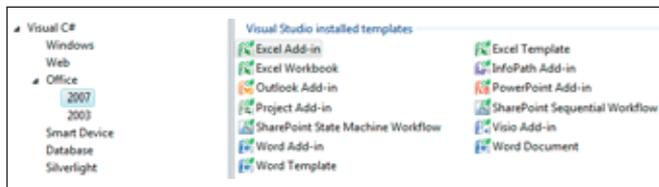


Figure 1 - Les projets VSTO disponibles dans Visual Studio 2008

Vous remarquerez la possibilité de créer des documents évolués (possédant des extensions en code managé) directement depuis Visual Studio, cependant ce point ne sera pas traité dans cet article tout comme la possibilité de manipuler Office avec COM. Pour vous convaincre de la

facilité de créer des extensions à vos clients Office favoris (ici Outlook par exemple), voici l'exemple d'une méthode qui crée un mail et l'envoie depuis votre client Outlook :

```
using Outlook = Microsoft.Office.Interop.Outlook;

using Office = Microsoft.Office.Core;
...
// Création d'un mail dans la boîte d'expédition
Outlook.MAPIFolder outBoxfolder = this.GetNamespace("MAPI").GetDefaultFolder(Outlook.OlDefaultFolders.olFolderOutbox);
Outlook.MailItem mail = (Outlook.MailItem)outBoxfolder.Items.Add(Outlook.OlItemType.olMailItem);
// Ajout de l'adresse d'un destinataire
mail.Recipients.Add("julien@wygwam.com");
// On formate le sujet et le corps du message.
mail.Subject = String.Format("Addin Office avec VSTO");
mail.Body = String.Format("Bonjour {0}{1}{1} Hello {0}{1}{1}", "Programmez!", Environment.NewLine);
// On ne veut pas afficher la fenêtre de rédaction du mail
mail.Display(false);
// On envoie le courriel
mail.Send();
```

L'ajout d'une barre de menu, la manipulation du contenu d'un document Word ou Excel sont tout aussi simples à réaliser. Du moins tant que l'on ne s'écarte pas trop du modèle objet proposé par VSTO. Par exemple, il faudra descendre au niveau de COM pour pouvoir demander l'en-tête original d'un mail, néanmoins pour les tâches dites courantes, VSTO se prête bien à cet exercice.

Déploiement d'un add-in VSTO avec Visual Studio

Avec l'arrivée de Visual Studio 2008, le mode de déploiement privilégié est Click Once, par conséquent vous ne retrouverez plus de projet de déploiement lors de l'ajout d'un projet d'add-in dans votre solution. Néanmoins, cela ne vous empêche pas pour autant d'ajouter manuellement un projet de déploiement et de continuer de la même façon que dans Visual Studio 2005.

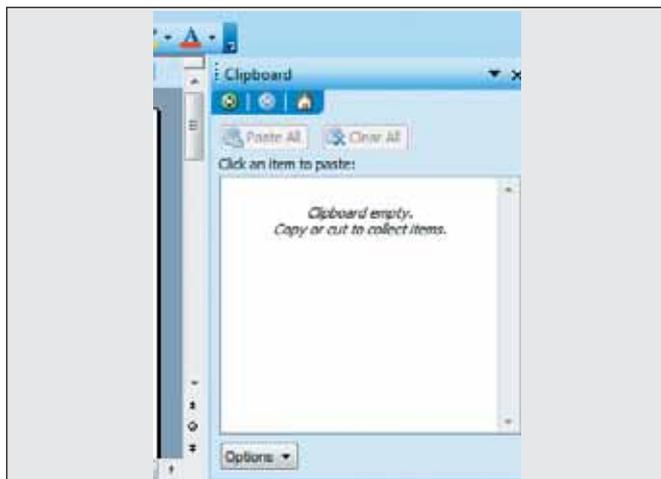
Etendre l'interface utilisateur

L'interface utilisateur reste un critère important et nécessaire pour proposer les fonctionnalités d'un produit aux utilisateurs. Pour cela Office

dispose de plusieurs éléments pour proposer les services à un utilisateur : les action pane, les smart tags, la barre d'outils et le ruban.

Les Action Pane

Les actions pane sont de véritables panneaux latéraux proposant aux utilisateurs une interface assez évoluée. Néanmoins, il faut éviter de créer des solutions nécessitant d'avoir plus d'un panneau ouvert à la fois car ceux-ci empiètent directement sur la zone de travail de l'utilisateur



Les SmartTags

Ces éléments, disponibles pour Word et Excel, permettent d'afficher un menu pop-up proposant des actions en fonction d'un certain contenu de document ou d'une feuille de calcul. L'image suivante présente un exemple de Smart Tag dans le domaine de la finance qui propose des actions lorsque l'utilisateur saisit des symboles financiers (ici MSFT) :



Exemple du Smart Tag Financial Symbol

La création d'un Smart Tag avec VSTO est rapide et simple. Voici l'élaboration d'un Smart Tag proposant d'aller sur le site du magazine :

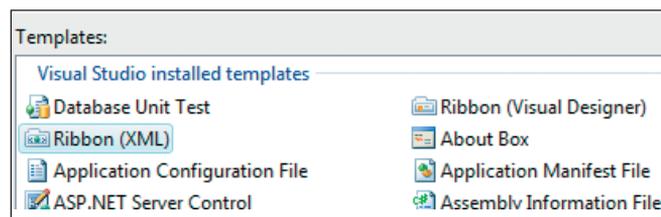
```
using Microsoft.Office.Tools.Word;
...
// Création du Smart Tag
SmartTag smartTagDemo = new SmartTag(
"www.microsoft.com/Demo#ProgrammezSmartTag", "Programmez Smart Tag");
// On spécifie les termes à reconnaître.
SmartTagDemo.Terms.Add("Programmez!");
// Création d'une action à exécuter
Action monAction = new Microsoft.Office.Tools.Word.Action("Ouvrir le site");
// Ajoute l'action au Smart Tag
smartTagDemo.Actions = new Action[] { monAction };
// On attribue des événements à l'action
monAction.BeforeCaptionShow += new BeforeCaptionShowEventHandler(...);
```

```
monAction.Click += new ActionClickEventHandler(...);
//On ajoute le Smart Tag au document
this.VstoSmartTags.Add(smartTagDemo);
```

Ribbon X

Office 2007 offre un nouveau système d'interface appelé le " ruban ". Ce système représente une manière efficace d'organiser les commandes en groupe de façon à les retrouver rapidement. Cette approche donne aussi l'avantage d'offrir un environnement fortement contextuel des commandes.

VSTO et son intégration dans Visual Studio 2008 supporte bien la création d'interface de type Ribbon X comme le montrent les captures d'écran suivantes :



Ajout d'élément RibbonX dans un projet VS 2008



Conception de ruban à l'aide d'un designer

Ribbon X avec Open XML

Il est possible d'embarquer la définition d'une interface Office directement dans un document Open XML à l'aide de sa structure flexible (utilisation d'Open Packaging Conventions) et de ses capacités d'extensibilité.

La garantie d'interopérabilité : le format Open XML

Créer des applications avec la plate-forme Office est intéressant pour les entreprises dont le socle applicatif des logiciels bureautique est essentiellement Office. Néanmoins, il est aussi important de pouvoir travailler dans des environnements hétérogènes permettant ainsi à d'autres applications ou types d'applications (analyseurs, générateurs, etc. ...) de pouvoir travailler de concert. C'est ce que propose le standard Open XML (ECMA-376) en spécifiant la structure des documents Word, Excel et PowerPoint de façon ouverte. Open XML étant principalement basé sur XML et le format de compression ZIP, quasiment toutes

Découverte

les plates-formes de développement (.NET, Java, PHP, ...) et systèmes d'exploitation actuels sont capables de le générer, le manipuler et l'exploiter.

Par exemple on pourrait imaginer une solution dans un environnement hétérogène suivant :



Cette solution ne poserait aucun problème pour fonctionner malgré sa nature multi plate-forme (Windows/Linux) et multi technologique (.NET/Java) grâce à ce nouveau format de documents bureautique.

Développer avec le format Open XML

L'arrivée de .NET 3 a permis d'apporter de nombreuses API complémentaires en plus des technologies WF, WCF, WPF et CardSpace. C'est le cas des APIs de Vista et notamment de l'assembly WindowsBase.dll et de son espace de nom System.IO.Packaging qui permet de manipuler la structure des documents Open XML (et accessoirement XPS).

Voici un exemple d'utilisation de l'API pour récupérer l'intégralité du contenu d'un document Word avec cet API :

```
const string typeRelationDocument = "http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDocument";
// Ouverture du paquet en lecture.
using (Package monPackage = Package.Open(nomFichier, FileMode.Open, FileAccess.Read)){
// Obtention de la partie principale du document (workbook.xml, document.xml, presentation.xml).
foreach (System.IO.Packaging.PackageRelationship relation in monPackage.GetRelationshipsByType(typeRelationDocument)) {
// Il ne doit y avoir qu'une seule partie de contenu dans le package.
Uri uriDocument = PackUriHelper.ResolvePartUri(new Uri("/", UriKind.Relative), relation.TargetUri);
PackagePart partieDocument = monPackage.GetPart(uriDocument);
XmlDocument doc = new XmlDocument();
doc.Load(partieDocument.GetStream());
MessageBox.Show(doc.InnerXml);
// Une seule partie de contenu, donc nous sortons maintenant.
break;
}
}
```

En plus de cette API livrée 'Out of the box' par .NET 3, Microsoft propose un nouveau SDK permettant de manipuler directement le contenu – soit les parties – d'un document sans se soucier de la structure interne d'un fichier Open XML. N'oubliez pas d'installer ce SDK (lien en fin d'article) et de référencer l'assembly Microsoft.Office.DocumentFormat.OpenXml.

Voyons un exemple rapide de lecture de la partie de commentaires (opé-

ration identique à celle réalisée dans l'exemple précédent) d'un document Word :

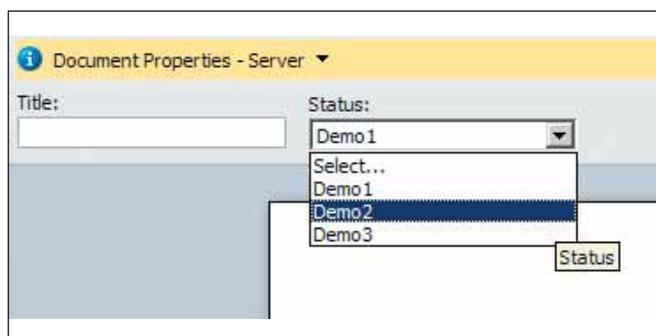
```
using Microsoft.Office.DocumentFormat.OpenXml.Packaging;
...
string comments = null;
using (WordprocessingDocument wordDoc = WordprocessingDocument.Open(document, true)) {
try { wordDoc.Validate(null); }
catch (Exception) { return null; }
MainDocumentPart mainPart = wordDoc.MainDocumentPart;
CommentsPart commentsPart = mainPart.CommentsPart;
using (StreamReader streamReader = new
StreamReader(commentsPart.GetStream())) {
comments = streamReader.ReadToEnd();
}
}
```

Comme on le constate, le SDK Open XML apporte une nouvelle couche au-dessus de l'espace de nom System.IO.Packaging en fournissant une abstraction et un typage fort des parties d'un document Open XML. Cela rend le développement de solutions autour des documents bureautiques encore plus productif et plus intuitif. Un plus qui devrait se bonifier avec le temps puisque cette version est pour le moment encore en Technical Preview. En dehors de tout développement avec la plate-forme Office (VSTO & consort), en utilisant les API mises à disposition dans .NET 3 et le SDK Open XML, vous avez la capacité de développer des solutions complexes et robustes de façon simple et rapide. Finalement, et en plus de sa richesse fonctionnelle, le format Open XML réussit le pari de réconcilier Office avec les outils tiers et les environnements non-Windows.

La collaboration avec SharePoint

La collaboration des employés d'une entreprise est un facteur essentiel de productivité. Afin d'optimiser cette collaboration, SharePoint est devenu un produit incontournable surtout avec l'arrivée de MOSS 2007 offrant une solution de portail.

L'intégration de fonctionnalités SharePoint dans Office avec notamment Word, Excel, PowerPoint, Outlook, OneNote, Groove et InfoPath est une avancée significative dans le domaine de la collaboration. Vous allez pouvoir, par exemple, publier et créer vos documents dans des bibliothèques de documents, synchroniser et partager vos documents de façon triviale, et tout cela à partir de votre client Office. Vous allez pouvoir enrichir vos documents avec des métadonnées directement depuis votre client afin de pouvoir identifier et caractériser les documents dans vos bibliothèques SharePoint :



Excel Services

MOSS 2007 apporte également Excel Services qui fournit aux entreprises le moteur de calcul de Excel. Ainsi vous n'avez plus besoin d'avoir un Excel installé sur votre serveur (pratique non recommandée) pour pouvoir traiter les feuilles de données Excel. Noter que Excel Services est accessible via des appels de Web Service, ce qui permet une fois de plus d'accroître l'ouverture et les possibilités offertes par Office. Excel Services permet aussi et surtout d'afficher des documents Excel dans votre navigateur Internet depuis MOSS, ce qui permet de manipuler les documents Excel directement depuis cette interface. Mais ne vous y trompez pas, Excel Services est dans sa première version et il cible particulièrement l'accès multi utilisateur. Ce n'est en rien un outil de création de documents Excel ou de calcul intense. Il y a donc quelques limitations à sa manipulation, mais nul doute que ce nouveau 'produit' à de l'avenir devant lui.

L'autre avantage d'Excel Services tient aussi dans le fait que la logique métier des feuilles est complètement cachée à l'utilisateur. Ainsi, vous pouvez proposer un calcul d'hypothèque ou de prévisions financières sans avoir à exposer vos algorithmes de calcul et les données utilisées. Noter que Team System utilise également SharePoint, dans sa version 2003 seulement, pour stocker les documents inhérents à votre méthodologie de développement.

L'intégration d'InfoPath

InfoPath est un produit destiné principalement à la saisie de données afin de produire des documents structurés en XML. La nouveauté aujourd'hui avec MOSS 2007 est la possibilité de saisir les informations sans l'aide d'un client InfoPath avec les InfoPath Forms.

Vous aurez aussi la possibilité de traiter les données saisies dans InfoPath par des workflows déployés sur un serveur SharePoint. A moins d'utiliser les workflows fournis par défaut dans SharePoint, ces workflows peuvent être conçus principalement de deux façons bien distinctes. La première est l'utilisation de SharePoint Designer qui permet aussi de créer des modèles de site ou de modifier un site (SharePoint Designer représente une énorme évolution du produit FrontPage). La seconde est l'utilisation de Windows Workflow Foundation avec Visual Studio pour pouvoir gérer les cas complexes de workflow ou de traitement métier.

La gestion de projet avec Project

Project est un produit qui existe depuis de nombreuses années, et Microsoft ne pouvait pas s'arrêter en si bon chemin avec l'arrivée en 2003 du produit Project Server. Aujourd'hui, Project Server 2007 offre de nouvelles possibilités à ses utilisateurs en proposant d'un part une intégration évidente avec le client Project, souvent utilisé par le chef de projet, et d'autre part un accès par le web pour ses utilisateurs et autres gestionnaires. Sans surprise, Project Server travaille de concert avec SharePoint Services afin de fournir une gestion plus simple et harmonieuse des différents projets.

SharePoint Services

SharePoint Services permet aux entreprises de proposer un environnement de collaboration puissant, robuste et extensible à ses utilisateurs avec une simple licence Windows Server 2003. Non loin d'être un produit fédérateur des ressources en entreprises, SharePoint Services est aussi un environnement de partage et de support pour les produits Office.

Choisissez votre version en fonction de vos besoins

Office est une suite bureautique de premier choix, néanmoins les outils puissants nécessitent souvent de s'affranchir d'un coût de licence, qui multiplié par le nombre d'utilisateurs peut être significatif et peser dans la balance lors de la décision d'utiliser ou de migrer vers Office.

Dans le cas d'Office 2007, il vous faudra compter sur une version Standard pour bénéficier du quatuor Word, Excel, PowerPoint, Outlook ; d'une version Professionnel pour avoir Publisher et Access en plus ; ou d'une édition Entreprise ou Intégrale pour également jouir de InfoPath, Groove, Communicator et certaines fonctionnalités avancées.

Conclusion

Office 2007 est pour sûr une plate-forme intéressante pour le développement, elle permet d'adapter et d'étendre les outils quotidiens des utilisateurs pour y intégrer, par exemple, les solutions métiers d'une entreprise. Les possibilités offertes par Office avec SharePoint et Open XML permettent de pousser l'intégration de solutions directement au sein des applications et des systèmes sans que les utilisateurs aient besoin de quitter leur environnement. A l'instar des utilisateurs, la plate-forme Office permet aux développeurs .NET de pouvoir créer des solutions puissantes et performantes sans quitter leur environnement de travail quotidien. Office est une plate-forme unifiée pour la production, le développement et la collaboration. Un produit riche qui reste souvent, et malheureusement, peu ou mal exploité.

Références

- Site officiel Microsoft Office : <http://www.microsoft.com/france/office/2007/default.aspx>
- Le blog officiel de l'équipe de développement VSTO : <http://blogs.msdn.com/vsto2/>
- Programmer les applications Office : [http://msdn2.microsoft.com/en-us/library/e7e2s44a\(VS.80\).aspx](http://msdn2.microsoft.com/en-us/library/e7e2s44a(VS.80).aspx)
- SDK Open XML : <http://www.microsoft.com/downloads/details.aspx?FamilyId=AD0B72FB-4A1D-4C52-BDB5-7DD7E816D046&displaylang=en>
- Télécharger les snippets Open XML pour Visual Studio : <http://www.microsoft.com/downloads/details.aspx?displaylang=fr&FamilyID=8d46c01f-e3f6-4069-869d-90b8b096b556>

■ Julien Chable

Collaboration & Portal Team

Wygwam – <http://www.wygwam.com>

CONCOURS DE DEVELOPPEMENT OFFICE

Un concours est organisé par le site www.logitheque.com, en partenariat avec Microsoft France, destiné à mettre en valeur l'inventivité et la créativité des utilisateurs d'Office 2007, du 15 octobre au 16 décembre.

Les internautes votent pour la réalisation Office 2007 qu'ils préfèrent.

Ce concours doit faire la promotion des trésors d'inventivité, trouvailles et perles réalisées par les utilisateurs d'Office 2007, professionnels ou amateurs. Tout

utilisateur d'Office 2007 peut en effet présenter ses inventions et réalisations. Le concours tourne autour de documents en langue française (tableaux Excel, présentations PowerPoint, bases Access...etc.) créés obligatoirement à partir d'Office 2007 ou de versions anciennes d'Office mises à jour.

Les réalisations Office 2007 participant au concours seront téléchargeables à partir du 15 octobre sur www.logitheque.com. 100 prix sont offerts, le premier étant doté de 5000 euros.

Centre technique pour les technologies Microsoft

Depuis quelques mois, au cœur de Paris, dans les locaux de Microsoft France rue de l'université, les entreprises et éditeurs peuvent venir tester, expérimenter, vérifier leur migration vers les technologies .Net et Microsoft. Comment fonctionne ce centre ?

Reportage au cœur du MTC (Microsoft Technology Center).

Inauguré il y a quelques mois, le MTC Paris fait partie d'un "réseau" d'une quinzaine de centres identiques à travers le monde, dont un tiers se situe en Europe. Le MTC a trois grandes missions : accompagner le client pour les projets critiques / stratégiques avec des briefings de stratégie, connaître les contraintes du client, voir ensuite les solutions possibles. 2e mission : travailler sur l'architecture de l'application (soit Microsoft, soit un partenaire). 3e mission : réaliser un prototype. Cela prend entre 1 à 3 semaines en général et on passe par la définition d'un modèle sur la faisabilité.

Le MTC possède de nombreux benches sur la validation des projets, la migration. Dès le départ, les équipes tiennent compte de l'architecture définie pour tester la migration, définir les règles. Mais au-delà des technologies Microsoft, le MTC sait s'intégrer avec des technologies et langages non Microsoft, comme PHP ou Java / Java EE. Il faut prendre l'existant et savoir composer avec celui-ci quand c'est nécessaire. L'infrastructure du MTC est à la hauteur du travail à fournir : plus de 100 serveurs, 40 stations de travail, du 32 et 64-bit, du SAN, un réseau gigabit, etc.

Atos et Compuware en expertise

Une des forces du MTC est de s'appuyer sur des expertises extérieures et notamment sur celles de Compuware et d'Atos. Compuware s'occupe de la partie outil avec, par exemple,



DevPartner. Les outils ont été évalués et certifiés par le centre technologique. Atos, pour sa part, est l'intégrateur, l'expertise technique. Les équipes d'Atos travaillant au MTC s'occupent beaucoup du processus de migration, de l'accompagnement de projet comme passer de DNA à .Net.

Du workshop à la finalisation

Le MTC n'est pas dédié aux développeurs indépendants pour tester leurs projets, mais plutôt pour les entreprises et notamment les grands comptes afin de préparer un projet de migration. Elles sollicitent alors le MTC pour une expertise et avoir une analyse pertinente sur la faisabilité de la migration. L'entreprise passe tout d'abord par un "workshop" permettant de définir le cadre et l'ampleur de la migration. Occasion aussi pour voir et définir l'architecture, les règles utilisées. Ce travail permet ensuite de créer des scénarios. Ces scénarios aident à identifier les risques, les architectures définies. Le workshop est donc une visualisation des enjeux du projet. Pour ce faire, les experts du MTC possèdent un guide visuel pour bien comprendre ce que signifie migration, ce que le client ne comprend pas toujours. L'analyse est une étape vitale pour la réussite d'une migration. L'important est de définir la stratégie afin ensuite de la dérouler phase après phase. On passe alors par la phase des enjeux, l'audit puis enfin la planification. À noter que des questionnaires sont soumis aux clients pour qualifier l'opportunité d'une migration. On ne migre pas pour migrer. De l'aveu même des responsables du MTC, il arrive que des projets s'arrêtent là. Autre élément

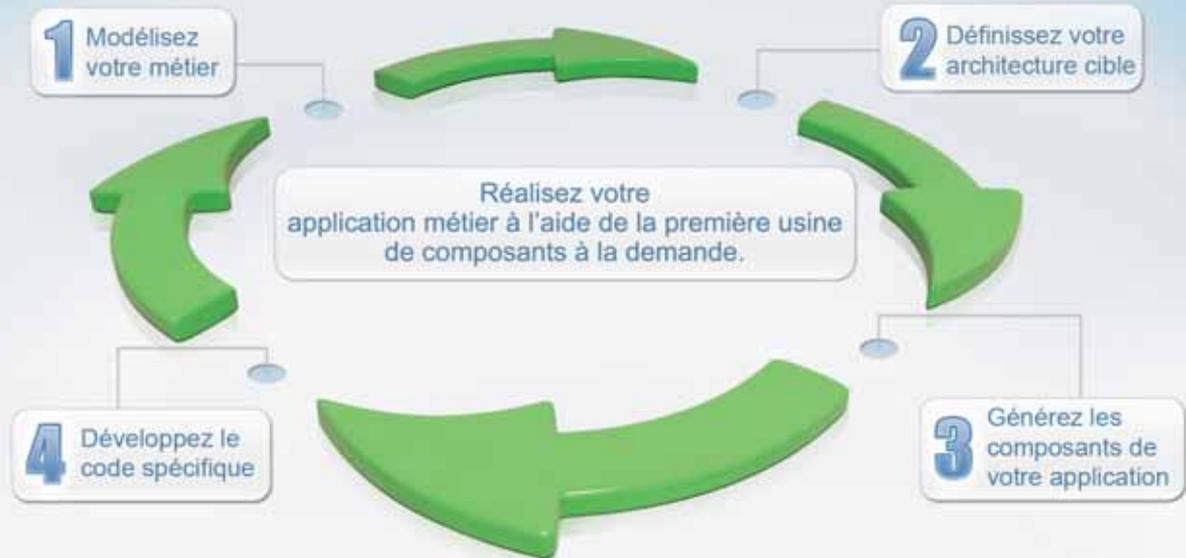
à ne pas oublier. Migrer le code c'est bien, mais il ne faut pas négliger les développeurs afin de les former, de les mettre à niveau sur les nouvelles technologies.

Qu'est-ce qui peut pousser une entreprise à migrer ? Les raisons s'avèrent multiples. On citera : aller vers le web 2, la fin de support des outils et langages utilisés, le changement de version de langage ou le changement de langage, de plate-forme, ou encore les contraintes réglementaires.

C'est durant l'analyse que les outils Compuware jouent un rôle prépondérant. Ainsi, en utilisant une analyse CAQS, on audite le code, on passe en revue l'interface. Au final, on possède une vision du code, de sa qualité. Au niveau de l'application, une cartographie est réalisée de l'ensemble des composants. CAQS permet donc d'obtenir une photographie de l'état de l'application dans son ensemble et cela facilite ensuite le travail dans DevPartner afin de créer les règles de migration. Une des difficultés est de coordonner les contraintes client au processus de migration. Mais Atos estime qu'à 80 - 90 %, on peut déléguer la migration à un intégrateur. Bien entendu, un tel projet a besoin d'un cadre. C'est pour cela qu'il existe le contrat de migration qui fixe le travail, les limites, le scénario, etc.

Et le prix d'une telle prestation ? Pas de tarif fixe, tout est à la carte, selon le projet. Il y a tout d'abord l'évaluation du projet puis la phase de discussion pour chiffrer les coûts de l'expertise, des outils et du temps. Là encore, il s'agit de sur-mesure.

■ François Tonic



CodeFluent.com

La fabrique en ligne est totalement gratuite pour les modèles métier comprenant jusque 30 entités.



Inscrivez-vous dès aujourd'hui sur : <http://www.codefluent.com>

0€

CodeFluent

Version complète Développeur donnant accès aux codes sources des composants générés.



Prix public : ~~2490€ht~~

Prix spécial lecteurs de "Programmez" : **1990€ht**

CodeFluent est une usine logicielle qui automatise la création des composants de votre application métier .NET selon les meilleures pratiques des experts de la plate-forme Microsoft.



SoftFluent est une société spécialisée dans le développement sous plate-forme Microsoft. Elle compte déjà de nombreux clients prestigieux tels qu'ILOG, VCS Timeless, TF1 ou Microsoft France et Europe. Contactez-nous sur info@softfluent.com

Softfluent recrute !

Venez assouvir votre passion du développement logiciel au sein d'une équipe dynamique !



TOUT savoir sur le développement .Net

Le Framework • L'architecture • C# ou VB.Net, le choix du langage • La boîte à outils

Depuis son lancement en 2002, le Framework n'a cessé d'évoluer pour être, à l'heure actuelle, utilisé par 65% des développeurs en France. Mais qu'est-ce que le Framework .NET au juste ?



Le Framework .NET est un composant utilisable sur les systèmes d'exploitation Windows, qui a pour but de faciliter la tâche des développeurs, en termes de développement, de débogage, de déploiement, etc. Certains le considèrent simplement comme la réponse de Microsoft au Java de Sun. En effet, l'un des deux composants de ce Framework .NET est la CLR (Common Language Runtime), équivalent de la JVM. Je ne m'étendrai pas plus sur ces deux composants car ils seront détaillés un peu plus loin dans le dossier réservé à la technologie .NET. Le Framework .NET, accompagné de Visual Studio, représente une solution complète et unifiée pour développer, déployer et exécuter des applications de tous types : applications Windows, applications Web, Services Web, applications mobiles, etc.

L'offre de Microsoft pour le développement en .Net comprend :

- Une interface de développement (IDE) et des outils d'aide au développement (actuellement Visual Studio 2005) qui permettent de transformer le code d'un programme en langage intermédiaire. Visual Studio 2005 se décline en plusieurs versions : (Express, Standard, Professionnelle...).
- Une plate-forme d'exécution (Framework .Net) fournissant notamment des fonctions système,

de gestion de la mémoire, de gestion des données et de communication (intranet et internet). Elle permet d'exécuter les programmes compilés en langage intermédiaire.

- Et depuis un an, un système de gestion du travail en équipe : Visual Studio Team System. Il permet d'industrialiser le processus de développement de vos applications. Le système est composé de plusieurs éditions indépendantes qui correspondent aux différents rôles des acteurs au sein d'un projet et d'un serveur de gestion de données.

Le modèle général de .NET se compose de 3 parties qui s'articulent autour de nombreux composants, de la gestion de la sécurité à la gestion de flux XML en passant par l'interopérabilité et l'accès aux données :

- Présentation (Client riche, Pages web, Mobile)
- Communication (Services Web, Protocole TCP/IP, http, ...)
- Données (Base de données, Modèle relationnel, XML...) etc.

Certains n'hésitent pas à demander : " Pourquoi utiliser .NET plutôt que Java ou une autre technologie ? ". Il serait très simple de leur répondre : " Et pourquoi pas ? ". En effet, vous souhaitez développer des Web Services ? Des programmes fonctionnant à la fois sur Windows et Linux ? Vous connecter à une base de données Oracle ou à un annuaire LDAP ? .Net le

peut, et de façon très simple. Il n'y a pas, selon moi, de véritable réponse à cette question : certains utilisent .NET car c'est une technologie à la mode, d'autres car c'est un produit Microsoft ou encore un concurrent de Java, etc. Bref, c'est au cas par cas et seuls des besoins très spécifiques vous feront choisir l'une ou l'autre de ces technologies. On a beaucoup reproché à la technologie .NET de n'être opérationnelle, au départ, que sous Windows. Cependant, on remarque que cette technologie a su marquer des points auprès des développeurs qui n'ont pas hésité à en créer une implémentation sous Linux. Même si certains restent sur leur position en affirmant que .NET ne peut pas rivaliser avec Java, le simple fait que la technologie ait été portée sous Linux montre bien que bon nombre de développeurs s'y intéressent et sont prêts à envisager, un jour ou l'autre, de faire leurs développements en .NET sous Linux. Avec la technologie .NET, Microsoft aurait-il trouvé un bon concurrent à Java ? C'est fort possible, si l'on regarde l'évolution de l'utilisation de .NET face à Java au cours des dernières années, mais pour en être sûr, il faudra encore attendre quelques années....

■ Van Thomas LE - Winwise
Expert en technologie .NET
vanthomas.le@winwise.fr



Framework .NET : présentation des différentes versions

Pendant les 5 années d'existence de .Net, Microsoft a su l'imposer dans les entreprises pour les développements d'applications sur la plate-forme Windows, aussi bien sur les serveurs que sur les postes clients. Au fil des versions, les équipes de Microsoft à Redmond ont fait évoluer le Framework .Net autour de ses principaux objectifs comme la simplicité d'utilisation, la performance ou la sécurité.

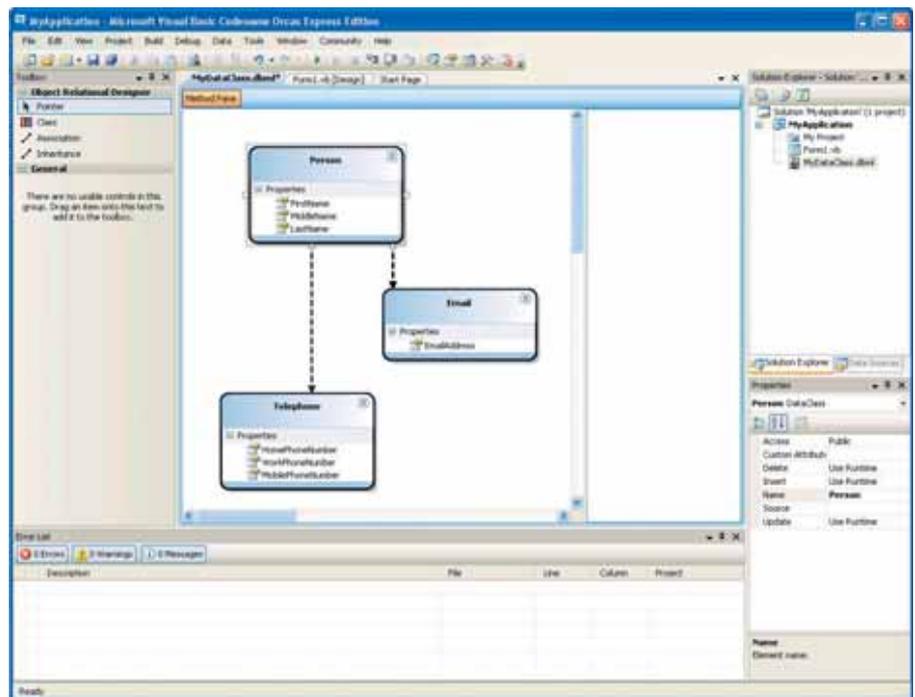
Objectifs du Framework .Net

Unification du développement

L'homogénéisation des développements est l'objectif principal du Framework .Net. Que ce soit sur les types d'applications ou même les types de plates-formes visées, la solution proposée par Microsoft permet de répondre à tous les cas de figures. En effet, avec le Framework .Net, il est possible de réaliser des applications orientées serveurs, comme les services Windows, les applications Web (ASP.Net) ou les Web Services (hébergées dans IIS, Internet Information Services), mais aussi des applications plus "classiques" sur les postes clients (WinForms). La grande force du Framework .Net est d'intégrer toutes ces possibilités en une seule et même plate-forme de développement, tout en assurant une parfaite communication entre chaque type d'application. Ainsi, une solution complexe basée entièrement sur la technologie .Net est possible, par exemple la gestion des stocks d'une entreprise. Celle-ci pourrait être réalisée par l'intermédiaire d'un Web Services pour extraire et mettre à jour les informations dans la base de données et plusieurs applications clientes communiquant avec le Web Service, tel qu'un site ASP.Net, une application WindowsForms ou même une application sur Pocket PC, via le Compact Framework .Net (qui est une version allégée et spécialisée du Framework .Net, exploitable au mieux sur cette plate-forme).

Simplicité de développement

L'autre objectif est la simplicité de développement. Et pour cela, le Framework .Net a plus d'un atout. Tout d'abord, il s'appuie sur une programmation orientée objet, tout comme le C++ ou le Java. Il est possible de développer des applications utilisant la technologie .Net avec plusieurs langages de programmation. Tout d'abord le C# (qui a été créé spécialement



pour cette plate-forme), le VB.Net qui permet aux développeurs VB de passer simplement à l'environnement .Net, mais aussi le J# (pour les développeurs Java désireux de passer à .Net) ou le C++. A ceux-ci s'ajoute un grand nombre d'autres langages comme COBOL.Net, Fortran.Net, Delphi.Net, Python : il suffit simplement qu'un compilateur soit capable de générer du code MSIL (Microsoft Intermediate Language). Les applications .Net sont des applications managées : le code MSIL qui les décrit est compilé lors de l'exécution afin d'être optimisé à la plate-forme sur laquelle est lancée l'application. De plus, de cette façon, la gestion de la mémoire est simplifiée car c'est la CLR (Common Language Runtime, équivalent de la JVM de Java) qui s'occupe de la gestion et libération de celle-ci lorsqu'elle n'est plus utilisée. De cette façon, le code est plus sécurisé et, par conséquent, le risque d'erreur de l'application est limité (il est, par exemple,

impossible d'accéder à une ressource à laquelle l'application n'est pas censée avoir accès !). Autre avantage du Framework .Net : sa librairie de classes est très importante. A priori, cela ne représente rien mais grâce à elle, les développeurs possèdent une boîte à outils complète (le framework .NET 2.0 comporte pas moins de 28 600 classes) pour réaliser leurs applications et cela dans de nombreux domaines. Que ce soit pour lire des données sur le disque dur, accéder à une base de données, manipuler des fichiers XML ou même communiquer en réseau pour échanger des données avec d'autres applications, toutes ces fonctionnalités sont fournies par défaut dans le Framework .Net. Ce Framework intègre également en natif un système de gestion de sécurité du code. Il est tout à fait possible de limiter les actions d'un programme .Net : on peut ainsi empêcher l'exécution de code voulant accéder à des ressources auxquelles il n'est pas censé avoir accès.

Enfin, l'interopérabilité avec l'existant est un point important, car il permet de continuer d'utiliser des composants COM ou des DLL non-managés au sein des applications .Net et cela, simplement. Inversement il est aussi possible d'utiliser des composants .Net dans des applications non-managées.

Les différentes versions du Framework .Net

Depuis 2002 et la sortie du Framework .Net en version 1.0, Microsoft a fait évoluer sa plateforme de développement et cela généralement en parallèle de son outil de développement phare qu'est Visual Studio. En effet, lors de la sortie de la version 1.0 du Framework .Net, Microsoft lança Visual Studio .Net : cette version était la toute première à permettre la réalisation d'applications utilisant la technologie .Net.

Framework .Net 1.1

Cette version du Framework .Net apporta la stabilité nécessaire pour que la plate-forme .Net soit suffisamment fiable pour être utilisée sans risque par les entreprises. Comme pour la version précédente du Framework, la sortie de la version 1.1 s'est accompagnée de la sortie d'une nouvelle version dédiée de l'IDE de Microsoft avec Visual Studio 2003.

Framework .Net 2.0

Le Framework .Net 2.0 fut une étape majeure dans l'évolution de la plate-forme de développement de Microsoft. Car en plus d'une nouvelle version de l'IDE (Visual Studio 2005), qui apporta un lot important de nouvelles fonctionnalités (classes partielles, méthodes anonymes, etc.), le framework .Net fut lui aussi mis à jour de façon très poussée. De nom-

breux points ont été revus au niveau de la sécurité, mais surtout au niveau des performances globales du CLR. Les langages de développement se sont vus ajouter de nouveaux mots clés et donc de nouvelles fonctionnalités. De plus, de nouveaux outils ont été rajoutés à la librairie du Framework, mais le framework 2.0 apporta aussi (et surtout !) une façon nouvelle de développer des applications ASP.Net.

Framework .Net 3.0

Ce Framework n'est pas réellement une nouvelle version, en effet il s'appuie à 100% sur la version 2.0, dont il conserve l'intégralité des fonctionnalités sans les modifier. Cependant, il apporte de nouvelles briques à la librairie du Framework, en y intégrant :

- **WCF** : Windows Communication Foundation, qui ajoute une nouvelle couche de communication simplifiée et unifiée au sein du Framework .Net
- **WPF** : Windows Presentation Foundation, qui est la nouvelle couche de présentation gérant les animations, un rendu vectoriel et aussi la 3D. A terme, WPF devrait remplacer la technologie WindowForms pour la création d'applications Windows
- **WF** : Windows Workflow Foundation, qui comporte des outils et un moteur de gestion de workflows
- **WCS** : Windows CardSpace qui permet aux utilisateurs de prouver leur identité de façon sécurisée et simplifiée sur Internet via des cartes d'identité numériques.

La version 3.0 du Framework .Net est livrée par défaut dans Windows Vista afin de généraliser l'utilisation de ces nouvelles technologies dans les prochaines applications mais vous pouvez tout à fait la télécharger pour Windows XP et Windows 2003.

Framework .Net 3.5

Le Framework 3.5 sera la nouvelle version qui arrivera officiellement au début de l'année 2008 avec Visual Studio 2008 (il est actuellement en version Bêta 2). En plus des technologies ajoutées avec la version 3.0 du Framework, elle ajoutera une nouvelle version de C# et de VB et surtout la technologie LINQ (Language INtegrated Query) qui permettra d'obtenir une nouvelle approche pour manipuler les données quelle que soit leur source : une base de données, un fichier XML ou tout simplement un objet .Net en mémoire.

A côté de ces versions majeures du Framework .Net, d'autres frameworks ont évolué aussi. C'est le cas du Compact Framework qui en est actuellement à la version 2.0 et qui a évolué jusqu'à cette version, au même rythme que son grand frère. Il existe également le Micro Framework .NET, utilisé pour le développement d'applications sur processeurs ARM7/9, que l'on retrouve dans les téléphones portables capteurs, robotique, etc.

Conclusion

Le Framework .Net est devenu essentiel dans la stratégie de Microsoft, à tel point que des produits essentiels comme SQL Server 2005 ou même Windows Vista l'intègrent nativement. Cela, sans compter sur les produits qui en dérivent, comme SharePoint depuis sa première version ou même Biztalk à partir de la version 2004. De plus, le Framework .Net ne se limite plus à la simple plate-forme Windows, en effet il est possible maintenant d'exécuter des applications .Net sous l'environnement Unix/Linux grâce au projet Mono.

■ **Davy FRONTIGNY** - Winwise
Expert en technologie
davy.frontigny@winwise.fr

Prochain numéro : **parution 12 décembre**



Hors Série .Net

Spécial Expression

Tout ce qu'il faut savoir sur la gamme (Blend, Web, Design, Media). Comment les mettre en œuvre ? Les fonctions et astuces.

.Net 3.5 / Visual Studio 2008

Toutes les nouveautés !

L'architecture de .NET

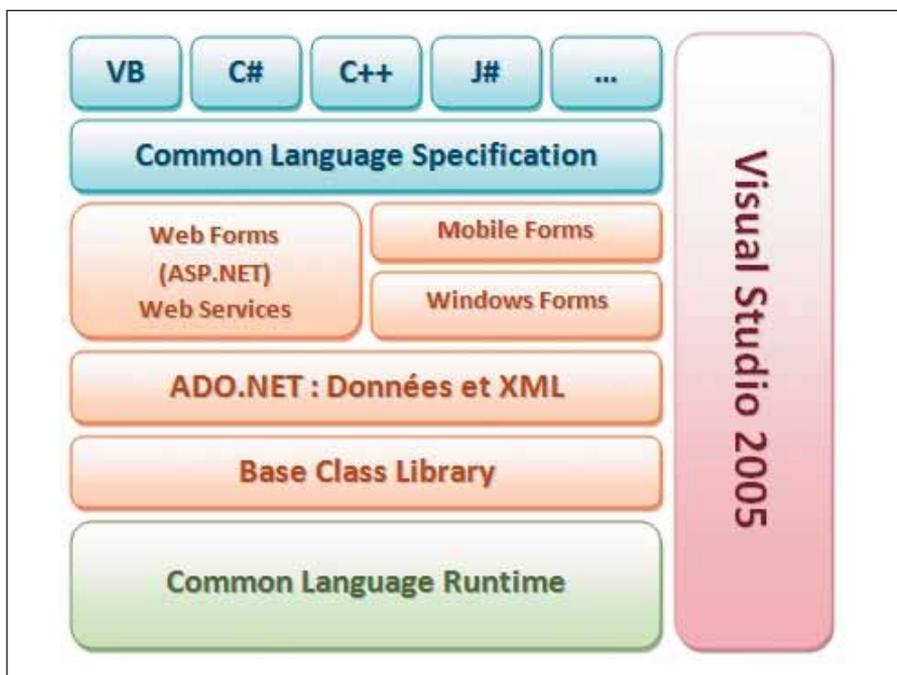
.NET est architecturé autour de deux composants principaux : l'environnement d'exécution appelé CLR (Common Language Runtime) et la bibliothèque de classes de base (BCL, Base Class Library). Ces deux éléments ont pour rôle d'optimiser, faciliter et accélérer le processus de développement de tout type d'application.

Le Common Language Runtime (CLR)

Le CLR (Common Language Runtime) est le premier composant du .NET Framework. C'est lui qui est en charge de l'exécution du code et qui fournit quelques services essentiels. Son rôle est un peu comparable à la machine virtuelle Java (JVM).

Contrairement à du code C/C++ non managé, dont la compilation génère des fichiers PE (Portable Executable) en code natif, le code .NET managé n'est pas directement compilé en code natif. En effet, le CLR utilise un langage intermédiaire appelé MSIL (Microsoft Intermediate Language) pour compiler le code managé en assemblies .NET. Les assemblies .NET sont encapsulées dans des fichiers PE classiques, à la seule différence qu'elles contiennent des métadonnées et attributs que le runtime utilise lors de leur exécution pour instancier les classes contenues, appeler des méthodes, appliquer la sécurité etc. Le code MSIL ainsi généré par le CLR est à son tour compilé (et optimisé en fonction de la plateforme) en langage machine au moment de son exécution via un compilateur JIT (Just-In-Time). Ce mécanisme de langage intermédiaire a l'avantage de rendre interopérables entre elles des assemblies développées en C#, VB.NET, Pascal, Cobol ou tout autre langage ciblant le runtime. En plus de prendre en charge l'exécution de code managé, l'interopérabilité entre les codes managés et non managés permet de continuer à utiliser d'anciens composants COM et DLL au sein du code managé.

Mais les services que propose le runtime ne s'arrêtent pas là. En effet, celui-ci est responsable de la gestion de la mémoire et décharge ainsi la tâche du développeur. Le runtime utilise pour cela un système appelé garbage collector (GC ou "ramasse-miettes") qui gère de manière autonome la durée de vie des objets managés via leur allocation et désallocation. Cette gestion automatique de la mémoire permet notamment d'éliminer les problèmes classiques de fuite de mémoire ou de non



libération d'un objet après sa consommation. Attention cependant, les ressources non managées (handle de fichier, pointeur unsafe) requièrent un nettoyage explicite de la part du développeur car le GC n'a aucune information sur les ressources non managées et n'est donc pas en mesure de les libérer.

Le runtime définit également un système de type commun (CTS ou Common Type System), afin que des classes définies dans plusieurs langages puissent communiquer entre elles. Elles ont besoin d'un ensemble de types de données communs. C'est l'objet même du CTS : définir des types de données que le runtime comprend, quel que soit le langage de développement utilisé dans l'application.

La bibliothèque de classes de base (BCL)

Le deuxième composant du framework, la bibliothèque de classes de base, est une collection complète orientée objet qui s'intègre parfaitement au runtime et rassemble des centaines de classes, interfaces et types réutilisables. Cette bibliothèque permet d'accéder

aux différentes fonctionnalités du système et joue le rôle d'API pour le développeur.

Les types du .NET Framework couvrent une multitude de fonctions : la représentation des types de données de base et des exceptions, l'encapsulation des structures de données, la gestion des entrées/sorties, l'accès aux informations de types chargés, l'accès aux données ou encore la création d'interfaces utilisateur riches.

La bibliothèque de classes prend en charge deux catégories de types : les types valeur et les types référence. Les types valeur contiennent directement leurs données (ex.: System.Int32, float, struct) et sont alloués sur la pile. Les types référence, eux, stockent une référence à l'adresse mémoire de la valeur et sont alloués sur le tas (exemple : System.DateTime ou System.String). Il est de coutume de les comparer aux pointeurs en C/C++.

■ **Damien BENDEJACQ** - Winwise
Expert en technologie .NET
damien.bendejacq@winwise.fr

C# ou VB.NET : choisir le bon langage

Avant d'analyser les différences existant entre C# et VB.NET, il est important de rappeler l'historique de ces deux langages.

La première version de Visual Basic (VB) est sortie en 1991. Celle-ci s'inspirait fortement du BASIC mais apportait une révolution à travers son environnement de développement d'interface graphique. Il y a eu 6 versions de VB avant que Microsoft ne sorte le Framework .NET. En 2002, Microsoft lance le .NET 1.0. Il intègre plusieurs langages dont Visual Basic .NET et C# 1.0. Tous ces langages s'appuient sur le même runtime : la CLR (Common Language Runtime). Cependant, il n'y a pas que la syntaxe qui diffère entre eux. Depuis VB4, Visual Basic faisait des efforts pour devenir un langage Orienté Objet (OO) avec l'apparition des classes puis des interfaces en VB5. Avec VB .NET, VB est devenu un vrai langage OO avec héritage, polymorphisme, encapsulation, etc... Le C#, quant à lui, apparaît avec le Framework .NET (il s'agit du langage qui a été développé pour .NET), il est très inspiré du langage à la mode du moment : Java. VB étant, pour beaucoup de développeurs, un langage destiné aux débutants (le B de B.A.S.I.C. signifie d'ailleurs Beginner). En 2008, Microsoft sortira le framework 3.5 avec la 3e version de C# et la 9e version de VB. L'historique est important car il permet d'expliquer certaines "étrangetés" du VB .NET par un problème de compatibilité avec l'existant.

Deux langages bien distincts

C# est devenu le langage phare de Microsoft, le CLR étant d'ailleurs développé avec VB.NET n'est cependant pas amené à disparaître dans les années à venir. Il y a bien chez Microsoft deux équipes de développement qui font chacune des choix techniques. C'est encore le cas aujourd'hui, chaque langage a ses propres spécificités.

Les délégués anonymes, par exemple, introduits par C#2.0, sont une fonctionnalité qui n'existe pas dans VB .NET 2005, le mot clé "yield" non plus. De même, alors qu'une harmonisation des nouveautés aurait été envisageable, il n'en est rien : "XML Litteral" sera, a priori, spécifique à VB9 et ne devrait pas être supporté par C#3.0. La syntaxe des "extension methods" (méthodes d'extension) avec C#3.0 est radicalement différente de celle de

VB9. Pourquoi cela ? Parce que chaque équipe a fait des choix afin de "coller" le plus possible à la philosophie du langage. Si les deux langages étaient identiques, il serait inutile d'en avoir deux.

VB .NET est un langage beaucoup plus verbeux que C#. Les propriétés en lecture seule illustrent bien cela. En C#, il suffit de ne coder que l'accessor "get". En VB .NET, il faudra, bien entendu coder l'accessor "get", mais il faudra également préciser que la propriété est en lecture seule avec le mot clé "ReadOnly".

L'instruction "for" diffère complètement en VB .NET par rapport à C#. C# a le "for" version "while" qui fait une initialisation, vérifie une condition et fait une instruction. En VB .NET, le "for", c'est celui du BASIC. C'est-à-dire une boucle de telle valeur à telle valeur, avec éventuellement un pas différent de 1 mais la borne max n'est jamais réévaluée. C'est peut-être plus simple, mais attention au parcours des éléments d'une liste dont des éléments seraient supprimés dans la boucle. En bouclant comme ceci :

```
For index As Integer = 0 To maCollection.Count - 1
  ...
End For
```

Il y a un risque car la propriété "Count" ne sera pas réévaluée.

Citons encore quelques autres différences :

- VB .NET n'est pas sensible à la casse contrairement à C#.
- L'espace de nom des fichiers de ressources est différent : "My.Resources" pour VB .NET. De façon générale, VB .NET possède un espace de nom un peu fourre-tout : l'espace de nom My.
- En VB .NET, le mot clé "Optional" permet de préciser la valeur par défaut d'un paramètre et ainsi de s'en passer à l'appel (dans le cas où la valeur souhaitée est celle par défaut bien entendu).
- La gestion des espaces de nom est différente (C# fait abstraction de l'assembly contrairement à VB .NET).
- VB .NET permet d'éviter le double typage lors d'une instanciation avec la syntaxe "As New". Cet argument perd de son intérêt face à C# 3.0 et les "local variables type inference". En effet, avec les "local variables type inference", le compilateur va automatique-

ment déduire le type de l'objet instancié (Object dans l'exemple). Dans l'exemple suivant : `var list = new List<string>();` list sera de type `List<string>`.

- VB ne possède qu'un seul opérateur = contrairement à C# qui possède un opérateur d'affectation et un opérateur d'égalité. De plus en C#, le = (affectation) retourne une valeur, ce qui peut s'avérer pratique.
- L'implémentation des interfaces est différente. En C#, on peut soit déclarer le membre public, soit le déclarer de façon explicite. En VB, il n'y a pas de notion d'implicite / explicite. Il faudra sur chaque membre dire qu'il implémente un membre de l'interface. Le plus de VB, c'est que ce membre pourra être Private (équivalent à explicite) mais aussi Protected, Friend (internal en C#) ou, bien sûr, Public. De plus, en VB, le membre pourra avoir un nom différent de celui de l'interface.
- VB .NET a la notion d'occultation avec le mot clé "Shadows".
- La distinction entre les propriétés et les méthodes est beaucoup plus claire en C# qu'en VB .NET, notamment dans la gestion des parenthèses.
- Pour définir un "custom event" en VB .NET, il faudra redéfinir les méthodes "AddHandler", "RemoveHandler" et "RaiseEvent". Alors qu'en C#, il n'est pas possible de redéfinir "RaiseEvent".

Si vous souhaitez avoir la liste complète des différences entre C# et VB.NET, il vous suffit de vous rendre à l'adresse suivante : http://www.codeproject.com/dotnet/vbnet_c_difference.asp

Un des avantages de VB .NET, surtout pour un développeur débutant, repose dans la syntaxe très facile à apprendre et à comprendre. Par exemple, l'héritage se définit avec le mot clé "Inherits" (au lieu de ":" en C#).

De plus, VB .NET fait la distinction entre l'héritage et l'implémentation d'interface (mot clé Implements) contrairement à C# qui utilise de nouveau ":".

La syntaxe Function / End Function, If [...] Then / End If, très proche d'une écriture algorithmique, facilite la relecture. En C#, un bloc se ferme systématiquement avec l'accolade fermante, ce qui oblige parfois à réfléchir pour savoir quelle accolade ferme quel bloc.

La boîte à outils du développeur .NET

Vous venez de faire vos premiers pas dans le monde du développement .Net. ? Suite à la lecture de ce dossier, une idée a germé et vous décidez dès à présent de réaliser votre première application .Net. Motivé mais peut-être un peu trop pressé, vous vous jetez dans l'écriture de votre œuvre. Stop !!!

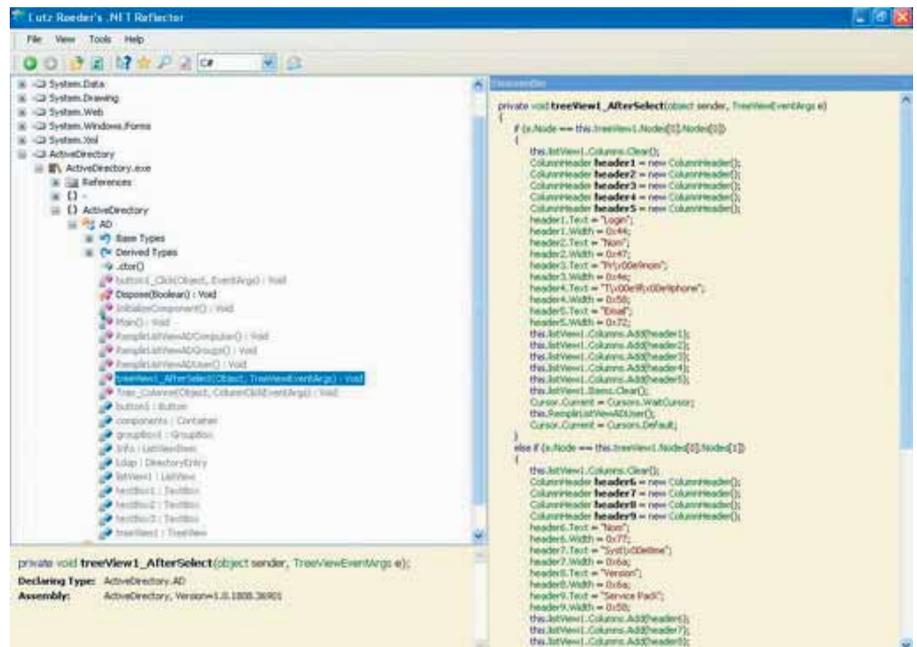
Un bon ouvrier ne travaille bien qu'avec de bons outils : prenez un peu de temps pour découvrir ce que la communauté .Net vous a concocté pour travailler plus efficacement.

Les éditeurs.

À la lecture de ce dossier, vous avez fait connaissance avec la plate-forme .net et vous avez écrit vos premières lignes de C#, VB.Net ou J#. Vous avez peut-être utilisé un simple éditeur de texte mais vous devrez faire appel à un véritable environnement de développement, comprenant des outils de compilation, de recherche, de conception voire d'automatisation et de génération pour mener à bien votre projet. Les environnements de développement peuvent désormais être séparés en deux grandes catégories, les gratuits et les payants. Si Visual Studio Professionnel et Team System sont obligatoires pour un développement à but commercial, la gamme Express permet dorénavant de s'équiper d'un IDE puissant, disposant de nombreuses fonctionnalités, à moindre frais puisqu'ils sont gratuits. Il ne vous en coûtera qu'un enregistrement et le téléchargement de la version adéquate, en fonction du langage que vous avez choisi ou du type d'application que vous souhaitez créer.

L'indispensable

Si vous avez retenu la leçon, vous savez désormais que le code que vous écrirez en .net ne sera pas directement exécuté sur la machine de vos clients mais passera par une machine virtuelle qui prendra en entrée le résultat d'une compilation .Net, à savoir du code IL. Ce code est un peu trop verbeux et en comprendre le sens est une activité longue et fastidieuse. Cependant, un outil (ou plutôt l'OUTIL) permet de parcourir ce code facilement puis de repasser celui-ci sous une représentation en code managé. Cet outil, Reflector (<http://www.aisto.com/roeder/dotnet/>) de Lutz Roeder est un indispensable pour tous les développeurs .Net. En effet, celui-ci permet de prendre connaissance des rouages internes



d'une librairie ou d'un composant que vous souhaitez utiliser et dont la documentation technique laisse à désirer. Les aficionados des produits en Bêta, pauvres en documentation, ne jurent que par lui.

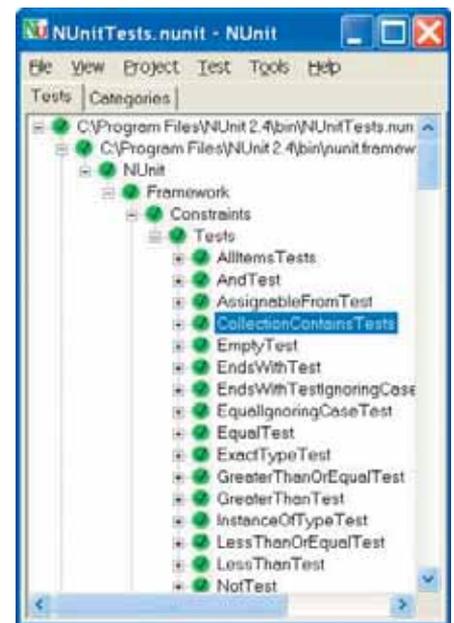
De plus, le fait de pouvoir désassembler les applications dont vous vous servez est une excellente façon d'apprendre à réaliser certaines actions.

Tester

Fier de vos premières lignes de code, vous en oublieriez presque vos bonnes intentions d'origine. Vous aviez promis d'écrire du code de qualité, testé, vérifié, contrôlé. Depuis 2002, une foule d'outils adaptés au développement d'application .Net permettent d'optimiser la qualité du résultat. Concernant les tests unitaires, impossible de ne pas parler de NUnit (<http://www.nunit.org/>), framework d'écriture et d'exécution des projets de test.

Par ailleurs, qui n'a pas un jour déclaré des variables au nom pour le moins brumeux, les "temp", "mavariabile" ? Qui n'a jamais hurlé à la vue de ce code (personnel ou d'un tiers) au moment de la maintenance ?

Bien entendu, il nous arrive d'avoir à coder de



cette façon pour tenter une approche ou un algorithme et tout le monde sait et comprend les dangers d'une telle démarche : l'oubli. Un outil devrait être votre pense-bête : FxCop (<http://www.getdotnet.com/Team/FxCop/>).



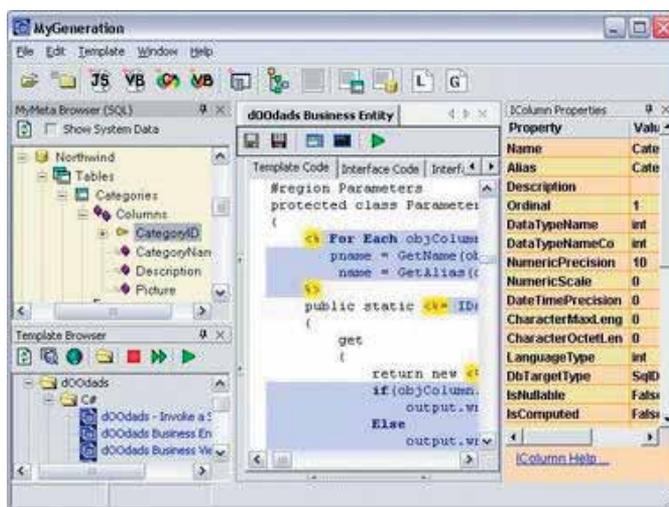
FXCop est un outil d'analyse statique de code, qui permettra de vérifier le bon respect de règles de codage dans votre projet. Ces règles sont configurables et extensibles, mais les règles fournies par défaut devraient déjà vous prendre un certain temps pour que votre application les respecte toutes.

Documenter

Vous avez peut-être découvert les fonctionnalités des langages .Net permettant de documenter vos classes et vos routines. Si vous avez pris soin de saisir l'intégralité des champs de cette documentation, il est temps de capitaliser cet effort et de transformer ce pauvre fichier XML généré par la compilation en une documentation technique que vous pourrez fièrement mettre en ligne ou livrer à vos utilisateurs. Cette transformation pourra se faire à l'aide de l'outil SandCastle (<http://www.microsoft.com/downloads/details.aspx?FamilyId=E82EA71D-DA89-42EE-A715-696E3A4873B2&displaylang=en>), encore en Bêta, mais entièrement fonctionnel, qui vous générera des fichiers CHM ou un site web HTML au look MSDN. SandCastle n'est rien d'autre qu'un ensemble de fichiers de transformation XML et sa configuration n'est pas des plus aisée. Heureusement, vous trouverez sur CodePlex, SandCastleHelpFileBuilder (<http://www.codeplex.com/SHFB>), un projet permettant de créer à l'aide d'une interface graphique le fichier de définition SandCastle qui correspond à vos attentes.

Protéger

Comme nous venons de le voir, il est possible de désassembler du code .net pour en voir le contenu et ainsi en comprendre le fonctionnement. Si vous n'êtes pas du genre prêteur ou que certains de vos algorithmes méritent que vous en conserviez la propriété, vous envisagez peut-être de protéger votre code de ce désassemblage. Vous devez donc faire appel à un obfuscateur, un outil qui modifiera le code afin d'en rendre la compréhension extrêmement difficile une fois désassemblé. Pour la plate-forme .net, Dotfuscator (<http://www.preemptive.com/products/dotfuscator/>) est la référence.



Construire

Ne serait-ce qu'à la lecture de cet article, vous conviendrez qu'une application moderne ne se compose pas d'une série de fichiers de code source à compiler. Les éléments de configuration, les tests, les données de base, voilà réellement de quoi sera constitué votre projet. Malheureusement, plus vous multipliez les éléments, plus vous aurez besoin d'organisation et de temps pour construire et livrer votre application. Pourquoi ne pas passer alors un peu de temps à automatiser cette construction. Si les premiers processus et outils de builds étaient constitués d'un ensemble de batch exécutant les différentes applications, aujourd'hui, de véritables Framework dédiés aux différentes plates-formes existent. Si vous utilisez le Framework 2.0 pour votre application, l'installation du kit de développement a entraîné l'installation de MSBuild. MSBuild est un framework de construction, basé sur des fichiers de scripts XML, définissant les différentes étapes et les différentes tâches qui seront exécutées lors de la construction d'une version de votre application. Si vous ne disposez pas encore du framework 2.0 et souhaitez tout de même automatiser vos constructions, vous pourrez utiliser Nant (<http://nant.sourceforge.net>), version dotnetisée du célèbre Ant, depuis longtemps connu comme une référence en la matière.

Déployer

Votre projet est terminé, vous l'avez peaufiné, testé, contrôlé, documenté... Vous souhaiteriez maintenant pouvoir en faire profiter la terre entière et voir sur vous retomber les lau-

riers de la gloire. Mais vous ne pouvez pas vous contenter d'une simple archive au format Zip et souhaiteriez fournir un véritable package d'installation. Si les packages MSI disponibles à partir de Visual Studio ne vous conviennent pas et que vous souhaitez en étendre les fonctionnalités, il existe WIX (<http://sourceforge.net/projects/wix/>). Le principe de WIX est de fournir une description des étapes d'installation d'un logiciel sous forme de fichiers XML qui, une fois compilés, permettent de créer le programme d'installation de vos rêves. Accompagnez le

de WixEdit (<http://wixedit.sourceforge.net/>) pour en tirer le meilleur parti.

Pour finir, pourquoi ne pas citer quelques outils, pas forcément dédiés à la plate-forme .net mais qui pourront vous rendre un fier service. En matière de génération de code, CodeSmith reste la référence mais sa version complète, payante, n'est pas à la portée de toutes les bourses. MyGeneration (<http://www.mygenerationsoftware.com/>) vous permettra d'obtenir des résultats assez proches à moindre coût.

WinMerge (<http://winmerge.org/>), comparateur performant de fichiers est un indispensable sur une machine de développement moderne.

Voilà donc un aperçu de tout ce que vous pourrez installer sur votre machine pour coder et développer facilement et efficacement. Bien entendu, la liste de ces outils n'est pas exhaustive et vous trouverez moult produits similaires et complémentaires sur les différents sites communautaires. Et pourquoi ne pas utiliser un de ces sites pour mettre en avant votre travail ? Codeplex (<http://www.codeplex.com>) paraît être aujourd'hui la meilleure solution (Team Foundation Server, suivi des versions, des changements, recensement des anomalies, wiki, visibilité importante...) pour promouvoir un développement que vous souhaitez partager, voire faire évoluer de manière collaborative en faisant participer les visiteurs.

■ Mathieu SZABLOWSKI - Winwise
Expert en technologie .NET
mathieu.szabowski@winwise.fr

L'information permanente

- L'actu quotidienne de Programmez.com
 - La newsletter hebdo la synthèse des informations
- Abonnez-vous, c'est gratuit ! www.programmez.com

System.Drawing

Bitmap
Brush
Brushes
Color
Font
FontFamily
Graphics
Icon
Image
ImageAnimator
Pen
Pens
Region

System.Drawing.Design

UITypeEditor

System.Drawing.Printing

PageSettings
PrintDocument
PrinterSettings

System.Media **NEW!**

SoundPlayer **NEW!**
SystemSounds **NEW!**

System.Windows.Forms

BindingSource **NEW!**
Button
CheckBox
CheckedListBox
ColorDialog
ComboBox
ContextMenuStrip **NEW!**
DataGridView **NEW!**
DateTimePicker
FlowLayoutPanel **NEW!**
FolderBrowserDialog
FontDialog
Form
GroupBox
ImageList
Label
LinkLabel
ListBox
ListView
MaskedTextBox **NEW!**
MenuStrip
MessageBox
MonthCalendar
NumericUpDown
OpenFileDialog
PictureBox

PrintDialog
ProgressBar
RadioButton
RichTextBox
SplitContainer **NEW!**
StatusStrip **NEW!**
TabControl
TableLayoutPanel **NEW!**
TextBox
TextRenderer
ToolStrip
ToolStripContainer **NEW!**
TrackBar
TreeView
UserControl
WebBrowser **NEW!**

System.Windows.Forms.Design

ControlDesigner
DocumentDesigner
UITypeEditor

System.Windows.Forms.VisualStyles

VisualStyleRenderer **NEW!**

System.Web

HttpApplication
HttpContext
HttpCookie
HttpRequest
HttpResponse
HttpRuntime
HttpServerUtility
HttpWorkerRequest
IHttpHandler
IHttpModule

System.Web.Caching

Cache
SqlCacheDependency **NEW!**

System.Web.Compilation

BuildProvider **NEW!**
ClientBuildManager **NEW!**

System.Web.Configuration

WebConfigurationManager **NEW!**

System.Web.Hosting

ApplicationManager **NEW!**
HostingEnvironment **NEW!**
VirtualPathProvider **NEW!**

System.Web.Management

WebBaseEvent **NEW!**

System.Web.Profile **NEW!**

ProfileBase **NEW!**
SqlProfileProvider **NEW!**

System.Web.Security

FormsAuthentication
FormsAuthenticationTicket
FormsIdentity
Membership **NEW!**
Roles **NEW!**

System.Web.Services

WebMethodAttribute **A**
WebService
WebServiceAttribute **A**

System.Web.Services.Description

ServiceDescription

System.Web.Services.Discovery

DiscoveryClientProtocol

System.Web.Ser

SoapDocumentM
SoapHttpClient

System.Web.Ses

HttpSessionSta

System.Web.UI

ClientScriptMa
Control
ICallbackEvent
IDataSource
Page
MasterPage
UserControl

System.Web.UI

HtmlControl
HtmlForm
HtmlButton
HtmlInputCont

data and xml

System.Data

DataColumn
DataRelation
DataRow
DataSet
DataTable
DataView

System.Data.Common

DbCommand
DbCommandBuilder **NEW!**
DbConnection

DbConnectionStringBuilder
DbDataAdapter
DbDataReader **NEW!**
DbProviderFactories **NEW!**
DbProviderFactory **NEW!**

System.Data.Odbc

OdbcCommand
OdbcConnection
OdbcConnectionStringBuilder **NEW!**
OdbcDataAdapter

OdbcDataReader
OdbcParameter
OdbcPermission
OdbcTransaction

System.Data.OleDb

OleDbCommand
OleDbConnection
OleDbConnectionStringBuilder **NEW!**
OleDbDataAdapter
OleDbDataReader
OleDbParameter

OleDbPermission
OleDbTransaction

System.Data.OracleClient

OracleCommand
OracleConnection
OracleConnectionStringBuilder **NEW!**
OracleDataAdapter
OracleDataReader
OraclePermission
OracleTransaction

System.Data.Sql

SqlNotification **NEW!**
SqlDataSourceEnumerator **NEW!**

System.Data.SqlClient

SqlBulkCopy **NEW!**
SqlClientPermission
SqlCommand
SqlConnection
SqlConnectionStringBuilder **NEW!**
SqlDataAdapter
SqlDataReader

SqlDependency **NEW!**
SqlParameter
SqlTransaction

System.Messaging

IMessageFormatter **I**
Message
MessageQueue

System.Xml

XmlAttribute
XmlDocument
XmlElement
XmlReader
XmlReaderS
XmlWriter
XmlWriterSe

System.Xml.S

XmlSchemaS

fundamentals

System

Action **NEW! G**
AppDomain
Array
Attribute
BitConverter
Boolean
Byte
Char
CLSCompliantAttribute **A**
Comparison **NEW! G**
Console
ConsoleKeyInfo **NEW!**
Convert **NEW! G**
Converter
DateTime
Decimal
Delegate
Double
Enum
Environment
EventArgs
EventHandler
EventHandler **NEW! G**
Exception
FlagsAttribute **A**
IComparable **I**
IComparable **NEW! G I**
IConvertible
IDisposable
IEquatable **NEW! G I**
IFormattable
Int32
Int64
Math
Nullable **NEW!**
Nullable **NEW! G**

Object
ObsoleteAttribute **A**
Predicate **NEW! G**
SerializableAttribute **A**
String
Type
Uri
UriBuilder
UriParser **NEW!**
Version

System.CodeDom

CodeExpression
CodeObject
CodeStatement

System.CodeDom.Compiler

CodeCompiler
CodeGenerator
CodeParser

System.Collections

ArrayList
Hashtable
IComparer
IComparable
IList
SortedList

System.Collections.Generic **NEW!**

Dictionary **NEW! G**
EqualityComparer **NEW! G**
ICollection **NEW! G I**
IDictionary **NEW! G I**
IEnumerable **NEW! G I**

IEqualityComparer **NEW! G I**
IList **NEW! G I**
LinkedList **NEW! G**
List **NEW! G**
SortedList **NEW! G**

System.Collections.ObjectModel

Collection **NEW! G**
ReadOnlyCollection **NEW! G**

System.ComponentModel

BackgroundWorker **NEW!**
BindingList **NEW! G**
Component
TypeConverter
PropertyDescriptor

System.Configuration

Configuration **NEW!**
ConfigurationManager **NEW!**

System.Deployment.Application **NEW!**

ApplicationDeployment **NEW!**

System.Diagnostics

Debug
EventLog
PerformanceCounter
Process
Stopwatch **NEW!**
Trace
TraceListener
TraceSource

System.DirectoryServices

DirectoryEntry
DirectorySearcher

System.DirectoryServices.ActiveDirectory **NEW!**

ActiveDirectorySite **NEW!**
DirectoryServer **NEW!**
Domain **NEW!**
Forest **NEW!**

System.DirectoryServices.Protocols

DirectoryConnection **NEW!**
DirectoryRequest **NEW!**
DirectoryResponse **NEW!**

System.EnterpriseServices

ContextUtil
ServiceConfig
ServiceComponent
ServiceDomain

System.Globalization

Calendar
CompareInfo
CultureAndRegionInfoBuilder **NEW!**
CultureInfo
DateTimeFormatInfo
NumberFormatInfo
RegionInfo
TextInfo

System.IO

Directory
DriveInfo **NEW!**
DriveType **NEW!**
File
FileStream
Path
Stream
StreamReader
StreamWriter
UnmanagedMemoryStream **NEW!**

System.IO.Compression **NEW!**

DeflateStream **NEW!**
GZipStream **NEW!**

System.IO.IsolatedStorage

IsolatedStorage

System.IO.Ports **NEW!**

SerialPort **NEW!**

System.Net

Dns
FileWebRequest
FtpWebRequest **NEW!**
HttpListener **NEW!**
HttpWebRequest
WebClient

System.Net.Mail **NEW!**

MailMessage **NEW!**
SmtpClient **NEW!**

System.Net.NetworkInformation **NEW!**

NetworkChange **NEW!**
NetworkInterface **NEW!**
Ping **NEW!**

System.Net.Security **NEW!**

NegotiateStream **NEW!**
SslStream **NEW!**

System.Net.Sockets

NetworkStream
Socket
TcpClient
TcpListener
UdpClient

System.Reflection

Assembly
ConstructorInfo
EventInfo
FieldInfo
GenericParameterAttributes **NEW!**
MemberInfo
MethodBase
MethodInfo
ParameterInfo
PropertyInfo

System.Reflection.Emit

AssemblyBuilder
ConstructorBuilder
EnumBuilder
EventBuilder
FieldBuilder
MethodBuilder
ParameterBuilder
PropertyBuilder
DynamicMethod **NEW!**
TypeBuilder

System.Resources

ResourceManager
UltimateResourceFallbackLocation **NEW!**

System.Runtime.CompilerServices

InternalsVisibleToAttribute **NEW! A**

System.Runtime.InteropServices

ComVisibleAttribute **A**
SafeHandle **NEW!**

System.Runtime

ISerializable
OptionalField
OnDeserializing
OnDeserializing
OnSerialized
OnSerialized

System.Runtime.Serialization

BinaryForma

System.Security

SecurityMan
SecureString

System.Security.Cryptography

AccessRule
FileSecurity
FileSystemA
ObjectSecur
RegistryAcc
RegistrySec

System.Security.Cryptography

SHA1
MD5
TripleDES
Rijndael

System.Security.Cryptography

EnvelopedPK
SignedPkcs7
X509Chain
X509Store

Microsoft® .NET Framework 2.0

Commonly Used Types and Namespaces

Microsoft
.NET Framework

www.msdn.microsoft.com/netframework

Microsoft
Visual Studio 2005



Microsoft®

© 2005 Microsoft Corporation. All rights reserved. Microsoft, Visual Studio, and the Visual Studio logo are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

tools

Microsoft.Build.BuildEngine NEW!

Engine NEW!
Project NEW!

Microsoft.Build.Framework NEW!

IEventSource NEW!
ILogger NEW!
ITaskItem NEW!
OutputAttribute NEW! A
RequiredAttribute NEW! A

Microsoft.Build.Utilities NEW!

Logger NEW!
Task NEW!
TaskItem NEW!
TaskLoggingHelper NEW!
ToolTask NEW!

exceptions

System.Exception

System.SystemException
BadImageFormatException
System.ArgumentException
System.ArgumentNullException
System.ArgumentOutOfRangeException
System.FormatException
System.IndexOutOfRangeException
System.InvalidOperationException
System.ObjectDisposedException
System.Net.NetworkInformation.PingException NEW!
System.Net.WebException
System.NotSupportedException
System.PlatformNotSupportedException
System.OperationCanceledException NEW!
System.OutOfMemoryException
System.TimeoutException NEW!
System.UnauthorizedAccessException
System.Collections.Generic.KeyNotFoundException NEW!
System.Configuration.ConfigurationException
System.Data.DataException
System.Data.ConstraintException
System.Data.NoNullAllowedException
System.Data.ReadOnlyException
System.Data.RowNotInTableException
System.Drawing.Printing.InvalidPrinterException NEW!
System.EnterpriseServices.RegistrationException
System.IO.InvalidDataException NEW!
System.IO.IOException
System.IO.DirectoryNotFoundException
System.IO.FileNotFoundException
System.IO.FileLoadException
System.IO.PathTooLongException
System.Management.ManagementException
System.Runtime.InteropServices.InvalidComObjectException
System.Runtime.InteropServices.ExternalException
System.Data.Common.DbException
System.Data.Odbc.OdbcException
System.Data.OleDb.OleDbException
System.Data.OracleClient.OracleException
System.Data.SqlClient.SqlException
System.Web.HttpParseException
System.Web.HttpUnhandledException
System.ComponentModel.Win32Exception
System.Net.NetworkInformation.NetworkInformationException NEW!
System.Net.Sockets.SocketException
System.Runtime.Remoting.RemotingException
System.Runtime.Serialization.SerializationException
System.Security.SecurityException
System.Security.Authentication.AuthenticationException
System.Security.Cryptography.CryptographicException
System.Threading.ThreadAbortException
System.Transactions.TransactionException
System.Transactions.TransactionAbortedException
System.Xml.XmlException
System.Xml.Schema.XmlSchemaException

System.Web.UI.WebControls
AccessDataSource NEW!
Content NEW!
ContentPlaceHolder NEW!
CreateUserWizard NEW!
DetailsView NEW!
FormView NEW!
GridView NEW!
Login NEW!
LoginName NEW!
LoginStatus NEW!
LoginView NEW!
Menu NEW!
ObjectDataSource NEW!
SiteMapDataSource NEW!
SiteMapPath NEW!
SqlDataSource NEW!
TreeView NEW!
Wizard NEW!
XmlDataSource NEW!

System.Web.UI.WebControls.WebParts NEW!

SqlPersonalizationProvider NEW!
WebPartManager NEW!
WebPartZone NEW!
WebPart NEW!
WebPartConnection NEW!

System.Xml.Serialization

IXmlSerializable
XmlSerializer

System.Xml.XPath

XPathNavigator
XPathExpression

System.Xml.Xsl

XslCompiledTransform NEW!
XsltArgumentList

System.Security.Permissions

SecurityPermission

System.Security.Principal

NTAccount NEW!
SecurityIdentifier NEW!
WindowsIdentity

System.ServiceProcess

ServiceBase

System.Text

Encoding
StringBuilder
UnicodeEncoding
UTF8Encoding

System.Text.RegularExpressions

Match
Regex

System.Threading

EventWaitHandle NEW!
Interlocked
Monitor
Mutex
ReaderWriterLock
Semaphore NEW!
Thread
WaitHandle

System.Transactions

TransactionScope NEW!
CommittableTransaction NEW!
Transaction NEW!

System.Security.Cryptography
EventWaitHandle NEW!
Interlocked
Monitor
Mutex
ReaderWriterLock
Semaphore NEW!
Thread
WaitHandle
System.Transactions
TransactionScope NEW!
CommittableTransaction NEW!
Transaction NEW!



Formations et Certifications

L'arrivée du framework .NET a donné à beaucoup l'envie de se lancer dans le développement. Pour cela, une bonne formation est nécessaire afin d'acquérir de bonnes bases dès ses premiers pas. Trois possibilités s'offrent alors : la formation classique, les livres et Internet.

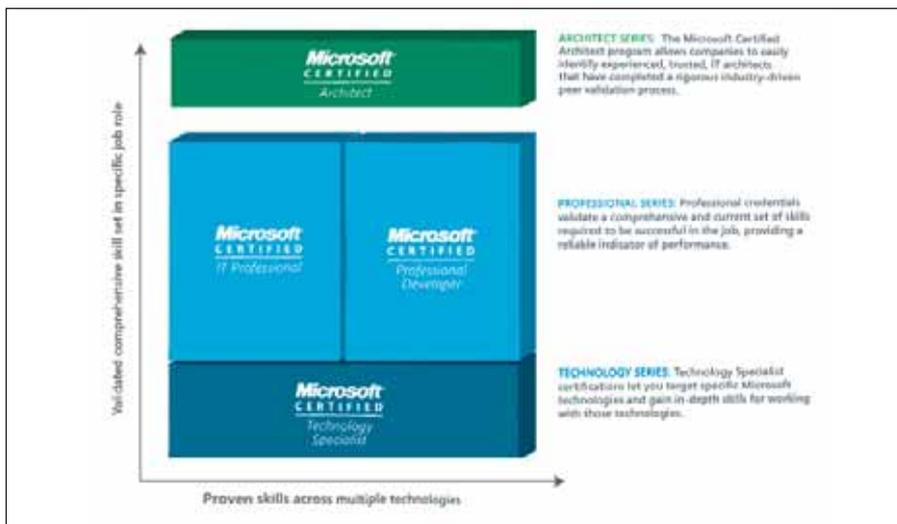
Microsoft propose un ensemble de cours officiels (les MOC, Microsoft Official Curriculum), dispensés dans des centres de formations agréés (les CPLS, Certified Partners for Learning Solutions) sélectionnés pour la qualité de leurs formateurs, leur savoir-faire dans le domaine de la formation et leur engagement à respecter les critères définis par Microsoft. Des formateurs indépendants certifiés dispensent aussi ce type de formations Microsoft. L'idée de disposer d'un pool de formateurs compétents techniquement et pédagogiquement est un gage de qualité important vis-à-vis des formations officielles.

Formations développeurs

Concernant le développement, Microsoft propose une douzaine de formations répartie en deux groupes. Le premier correspond à des formations génériques (introduction à Visual Studio 2005, Visual Studio 2005 pour les développeurs VB6, ...), le second à des Workshops plus thématiques: développement Web, WindowsForms, applications distribuées et accès au données. Chaque thème comprend deux formations, une " Core " (2 jours) et une " Advanced " (3 jours). Les deux conseils que l'on peut donner sont :

- suivre les deux formations pour le thème qui vous intéresse
- ne pas hésiter à bombarder le formateur de questions durant les exercices car ces formations font la part belle aux travaux pratiques, qui composent bien souvent plus de 70% du stage.

En parallèle de l'offre officielle de Microsoft, une offre complémentaire, fournie aussi bien par des centres CPLS que des centres indépendants, s'est développée : ces formations non officielles sont proposées pour approfondir un thème précis, répondre à un besoin particulier dans le cadre d'un projet, présenter une nouvelle technologie du marché informatique. Par exemple, Winwise propose des formations autour des framework 3.0/3.5 (WCF, WPF, WF, C# 3.0, LINQ) ou de Silverlight, Microsoft ne proposant pas à ce jour de cur-



sus de formation dans ces domaines dans le cadre de ses cours officiels.

Pour compléter l'ensemble, il existe des formations en ligne et des formations données à distance. Les formations en ligne sont dispensées par le biais du site Microsoft eLearning (<https://www.microsoftlearning.com/>). Les formations à distance, quant à elles, se font en collaboration avec un centre de formation certifié et sous l'égide d'un formateur. Celui-ci présente la partie théorique pendant quelques heures disséminées pendant la semaine (à distance, via Webcam interposée) et le stagiaire est libre de compléter les exercices à son gré. Ce domaine de formation reste tout de même assez peu développé.

La formation littéraire

A côté de cela, vous avez des ouvrages de tout type autour de vos technologies favorites pour vous auto-former. Même si nombre d'entre eux sont complets, ils ne peuvent remplacer l'expérience terrain apportée par un formateur, ni se substituer à une expérience réelle acquise sur un projet. Une fois une bonne expérience acquise il peut être intéressant pour vous de valider vos acquis.

Microsoft propose un système de certification dont le premier niveau, Microsoft Certified Technology Specialist (MCTS), valide les connaissances techniques :

- MCTS .NET Framework 2.0 Web Applications
- MCTS .NET Framework 2.0 Windows Applications
- MCTS .NET Framework 2.0 Distributed Applications

Chacun correspond au passage de deux certifications : une générique commune et une thématique.

Le deuxième niveau, Microsoft Certified Professional Developer (MCPD), entérine les compétences métier :

- MCPD Web Developer
- MCPD Windows Developer
- MCPD Enterprise Applications Developer

D'autres certifications dédiées aux développeurs existent autour de technologies Microsoft (Sharepoint, SQL Server, etc.). Ces certifications exigent de réelles connaissances terrain et nécessitent des mois de pratique, Microsoft privilégiant l'expérience à la théorie. Les centres de formations proposent d'ailleurs des offres combinées formation + certifications.

■ **Béatrice CARCEL** - Winwise
Responsable Pôle formation
beatrice.carcel@winwise.fr

■ **Mathieu PLASSE** - Winwise
Consultant/Formateur sur les technologies .NET
mathieu.plasse@winwise.fr

Les entrées/sorties avec le .NET Framework

De tout temps et dans tout programme informatique, dès que les micro-ordinateurs ont été pourvus d'un système de fichiers, les développeurs ont créé des bibliothèques pour y accéder. Je vous propose donc dans cet article un voyage au cœur des entrées/sorties avec VB.NET.

Dans notre travail de développeur, nous avons régulièrement besoin d'accéder au système de fichiers des postes de travail et autres serveurs sur lesquels nos programmes s'exécutent. La raison en est multiple ; pour certaines applications nous écrivons des logs, pour d'autres nous voudrions exporter ou importer des données, ou encore compresser des données pour les retourner à un tiers. Pour nous aider dans cette tâche, Microsoft a implémenté, au sein de .NET une API complète qui rassemble toutes les classes nécessaires à la gestion des entrées/sorties. Cet API est regroupée au sein d'un espace de noms nommé System.IO. Après avoir étudié le contenu de cet espace de noms ainsi que la façon de tirer parti des éléments du système de fichiers de nos PC, je vous propose, dans un second temps, de regarder plus spécifiquement la manière dont nous pouvons manipuler des fichiers. Pour ce faire, nous nous appuierons sur la création d'un outil permettant de compresser, décompresser et diviser des fichiers en plusieurs parties distinctes. Mais commençons d'abord par faire connaissance avec l'élément fondateur des entrées/sorties.

L'espace de noms System.IO

System.IO regroupe plusieurs classes, structures et énumérateurs. Il peut être vu comme regroupant les fonctionnalités principales suivantes dans la gestion des entrées/Sorties :

- Un jeu de classes permettant la navigation et la gestion du système de fichiers,
- Un jeu de classes utilisant les flux pour lire, écrire et compresser des fichiers,
- Des classes transverses comme les exceptions spécifiques.

Si System.IO est l'espace de noms principal, il possède des sous-espaces de noms dont :

- System.IO.Compression qui contient les classes de gestion de la compression,
 - System.IO.IsolatedStorage qui contient les classes permettant la gestion d'un espace de stockage isolé,
 - System.IO.Ports qui contient les classes nécessaires à la gestion du port série.
- Maintenant que nous avons une vision globale, regardons plus en détail la façon dont la navigation dans le système de fichiers est mise à notre disposition.

La Navigation dans le système de fichiers Windows

Plusieurs classes sont dédiées à la navigation dans le système de fichiers. Le tableau ci-dessous les répertorie et en donne une brève définition.

Classes	Définitions
Gestion des disques	
DriveInfo	Gestion des informations d'un disque
Gestion des répertoires	
Directory	Gestion des répertoires du système de fichiers
DirectoryInfo	Informations spécifiques d'un répertoire
Gestion des fichiers	
File	Gestion des fichiers
FileInfo	Informations spécifiques d'un fichier
Classes transverses	
Path	Gestion des chemins d'accès
FileSystemWatcher	C'est un observateur qui permet de notifier des modifications dans le système de fichiers comme l'ajout d'un fichier dans un répertoire.

La différence entre les classes suffixées par *info* et celles qui ne le sont pas réside dans le fait que la deuxième classe n'effectue pas de vérifications des droits sur les fichiers à chaque opération, ce qui est le cas de la première. Nous pouvons appliquer la règle suivante pour nous aider : pour une ou deux opérations mieux vaut utiliser les classes non suffixées par *info* pour de nombreuses opérations les autres classes sont plus appropriées. Désormais, passons à la pratique. Avant toute manipulation des entrées/sorties vous devez impérativement dans l'en-tête de vos fichiers sources, importer l'espace de noms System.IO comme suit :

```
Imports System.IO
```

Gestion des disques par l'exemple

La récupération des disques disponibles sur le poste de travail est réalisée avec la classe DriveInfo de la façon suivante :

```
Dim internalDrives As DriveInfo() = DriveInfo.GetDrives()
```

Les informations d'un disque sont accessibles aussi avec la classe DriveInfo. La méthode suivante prend un disque en paramètre et affiche les informations de ce dernier dans la console. Nous pouvons comme cela connaître son nom via la propriété Name ou encore sa taille via la propriété TotalSize :

```
Public Shared Sub DisplayDriveInformation(ByVal aDrive As DriveInfo)
    If aDrive.IsReady Then
        Console.WriteLine("Le nom du disque est : {0}", aDrive.Name)
        Console.WriteLine("La taille Totale du disque est : {0}", aDrive.TotalSize)
        Console.WriteLine("L'espace disponible du disque est : {0}", aDrive.TotalFreeSpace)
        Console.WriteLine("Le format du disque est : {0}", aDrive.DriveFormat)
    End If
End Sub
```

Gestion des répertoires par l'exemple

La création d'un répertoire est certainement l'une des premières actions que l'on souhaite effectuer. Pour ce faire, on utilise la méthode CreateDirectory de la classe Directory :

```
Directory.CreateDirectory("c:\Bewise\Articles\IO_VB_NET")
```

La suppression d'un répertoire, quant à elle, s'effectue via la méthode Delete de la classe Directory :

```
Directory.Delete("c:\Bewise\Articles\Programmez\IO_VB_NET")
```

Enfin, nous pouvons lister les fichiers d'un répertoire via la méthode GetFiles de la classe Directory toujours :

```
Directory.GetFiles("c:\Bewise\Articles\Programmez\IO_VB_NET", "*", SearchOption.AllDirectories)
```

La classe `DirectoryInfo` nous permet de récupérer diverses informations d'un répertoire. La méthode ci-dessous, qui prend en paramètre un répertoire, affiche son nom et sa date de création par exemple :

```
Public Shared Sub DisplayDirectoryInformation(ByVal oneDirectory As DirectoryInfo)
    Console.WriteLine("Le nom du répertoire est : {0}", oneDirectory.Name)
    Console.WriteLine("Date de création du répertoire : {0}", oneDirectory.CreationTime.ToShortDateString())
    Console.WriteLine("Nombre de fichier contenue dans le répertoire : {0}", oneDirectory.GetFiles("*.*", SearchOption.AllDirectories).Length)
    Console.WriteLine("Le répertoire parent est : {0} ", oneDirectory.Parent.Name)
End Sub
```

Gestion des fichiers par l'exemple

L'une des premières actions que l'on souhaite réaliser dans la manipulation d'un fichier c'est l'ouvrir. Vous trouverez ci-dessous le moyen d'ouvrir un fichier, s'il existe bien sûr :

```
If File.Exists("c:\test.txt") Then
    File.Open("c:\test.txt", FileMode.Open)
End If
```

La création d'un fichier est aussi parmi les actions courantes ; elle est réalisée de cette façon :

```
Dim fstream As FileStream = File.Create("c:\test.txt")
```

Nous avons aussi la possibilité de copier un fichier via la méthode `Copy`:

```
File.Copy("c:\test.txt", "c:\test2.txt")
```

Et enfin, une fois toutes ces manipulations opérées, il ne nous reste plus qu'à savoir supprimer un fichier comme présenté ci-dessous :

```
File.Delete("c:\text2.txt")
```

Nous venons de voir comment manipuler un fichier physique. Le chapitre suivant nous présente, pour sa part, la manière dont nous pouvons les modifier et interagir avec leur contenu.

Analyse des mécanismes de création et de modification des fichiers

Nous avons vu que la gestion des fichiers démarre à l'aide de la classe `File` qui met à notre disposition des méthodes statiques pour manipuler des fichiers physiques. Mais qu'en est-il lorsque nous souhaitons utiliser le contenu du fichier ? Et notamment le contenu de différents types comme du texte, du binaire ? Pour répondre à ce besoin, le .NET Framework met à notre disposition les flux de données.

Nous en avons déjà un aperçu, dans l'exemple sur la création d'un fichier, nous utilisons un `FileStream` en retour de la méthode `File.Create`. Le `FileStream` est en réalité un flux de données correspondant à un fichier. Je vous propose dans ce chapitre de vous présenter ces flux (`Stream` en anglais) et les moyens fournis pour les manipuler.

Définition, type et organisation des flux

Un flux est une séquence d'octets qui peut alors être représenté sous la forme d'un tableau d'octets comme ci-dessous :

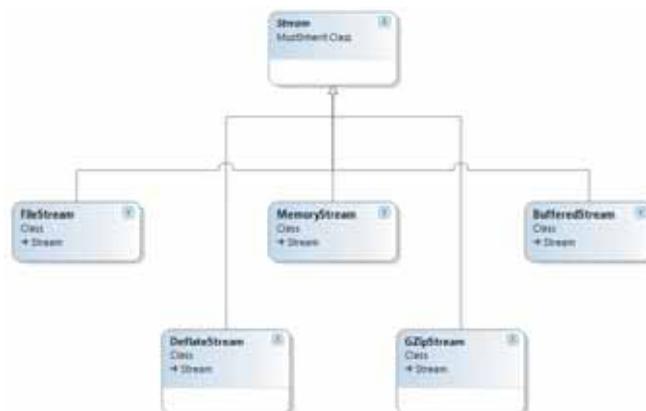
```
01110001 | 01110001 | 01110001 | 01110001 | 01110001 | 01110001 | 01110001 | 01110001 | 01110001
```

Les Flux dans System.IO

Il existe plusieurs types de flux dans l'espace de noms `System.IO`. Le tableau suivant les décrit :

Nom de la classe	Description
<code>Stream</code>	Classe abstraite dont tous les flux héritent
<code>FileStream</code>	Flux de données représentant un fichier physique (image, texte, xml, etc..)
<code>MemoryStream</code>	Flux de données en mémoire
<code>BufferedStream</code>	Flux de données représentant un Buffer (stockage temporaire d'octets)
<code>DeflateStream</code>	Flux compressé de type Deflate
<code>GZipStream</code>	Flux compressé de type GZip

Tous ces flux ont un comportement commun qui vient de la classe abstraite `Stream`. Le schéma suivant présente la hiérarchie basée sur cette classe : 140



La classe `Stream` nous fournit des méthodes permettant l'exploitation de la séquence d'octets qu'elle contient. Ces dernières permettent de naviguer, écrire et lire dans la séquence. Toutes ces méthodes sont implémentées dans les classes concrètes.

ReadByte : Lit un octet et avance la position en conséquence



WriteByte : écrit un octet et avance la position en conséquence



Comme nous pouvons le constater, nous avons à notre disposition le moyen d'écrire ou lire dans le flux mais cela reste complexe car nous manipulons des octets. On peut alors se demander comment écrire simplement une information de type texte ? Fort heureusement pour nous, le .NET Framework met à notre disposition des classes dédiées à l'écriture et la lecture dans les flux que nous appelons des `Reader` et des `Writer`. Etudions ces classes plus en détail à présent.

Les Reader et Les writer

Le principe des Reader et des Writer est toujours le même :

- Il nécessite un flux,
- Il s'applique sur ce flux en les prenant en paramètre de leur constructeur.

Nous nous intéresserons ici au reader et writer associés à l'écriture et la lecture de fichier. Nous pouvons en recenser 2 importants :

- Le StreamReader,
- Le StreamWriter.

StreamReader est une classe facilitant la lecture de fichier texte à partir d'un FileStream.

Elle permet donc de lire un fichier octet par octet, mais surtout de le lire ligne par ligne en considérant chaque ligne comme une chaîne de caractères. L'exemple suivant illustre ce fonctionnement. Tout d'abord la création d'un flux de données correspondant au fichier " c:\test.txt "

```
Dim fileText As IO.FileStream = File.OpenRead("c:\test.txt")
```

Ensuite, création d'un StreamReader qui s'appuie sur ce flux de données et nous permettra une lecture simplifiée du fichier :

```
Dim reader As StreamReader = New StreamReader(fileText)
```

Enfin, nous pouvons lire les informations contenues dans le fichier en s'appuyant sur le StreamReader. Dans l'exemple ci-dessous, La lecture se fait ligne par ligne via la méthode ReadLine. A chaque lecture de ligne, on enrichit un StringBuilder. La fin du flux est identifiée par la propriété EndOfStream:

```
Dim sb As New StringBuilder
While Not reader.EndOfStream
    sb.Append(reader.ReadLine())
End While
```

Il est nécessaire de fermer le StreamReader, après traitements, pour ne pas bloquer les ressources en mémoire, c'est le rôle de la méthode Close.

```
reader.Close()
```

A l'image du StreamReader, le StreamWriter a pour rôle de faciliter l'écriture d'un fichier texte à partir d'un FileStream. Il permet d'écrire une chaîne de caractères ou encore une ligne complète avec un retour à la ligne. L'exemple suivant illustre ce fonctionnement. Comme vous l'avez compris les deux premières étapes sont la création d'un flux de données sur le fichier et la création d'un writer basé sur ce flux :

```
Dim fileText As IO.FileStream=File.Open("c:\Nouveau.txt", FileMode.Create)
Dim writer As StreamWriter = New StreamWriter(fileText)
```

Ensuite, nous utilisons la méthode WriteLine du writer pour écrire une chaîne de caractères :

```
writer.WriteLine("Hello!")
```

Le StreamWriter doit être fermé via la méthode Close pour libérer les ressources et le fichier.

```
writer.Close()
```

Il existe de nombreux autres reader et writer comme par exemple TextReader, TextWriter ou encore le BinaryReader et le BinaryWriter que nous verrons plus loin. Pour illustrer ce que nous avons vu jusqu'alors, je vous propose de créer un outil de compression.

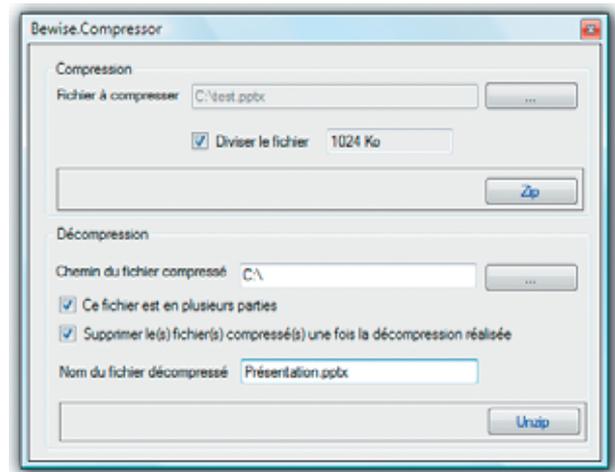
Création du programme Bewise.compressor

Les fonctionnalités de notre application sont :

- La compression du flux de données dans un fichier physique avec comme extension " bew " via un flux de données GZipStream.
- La scission du fichier compressé en plusieurs parties (la taille sera fixée à 1 Mo).
- La décompression du flux de données en récupérant les différentes parties s'il y a lieu.

L'IHM de notre outil

Comme le montre l'écran suivant, l'interface utilisateur de notre application sera réalisée en windows forms et permettra le paramétrage de l'outil.



La compression

On crée en premier lieu une copie du fichier source identifié par la variable FilePathSource vers le fichier à compresser identifié par la variable FilePathToCompress

```
File.Copy(FilePathSource, FilePathToCompress)
```

On crée ensuite un flux de données, qui sera compressé, identifié par la variable fileCompressComplete

```
Dim fileCompressComplete As FileStream = File.Open(FilePathToCompress, FileMode.OpenOrCreate, FileAccess.ReadWrite)
```

Ensuite, afin de réaliser la compression, on utilise un flux de données GZipStream qui s'appuiera sur le FileStream précédent.

```
Dim gzipStream As IO.Compression.GZipStream = New IO.Compression.GZipStream(fileCompressComplete, Compression.CompressionMode.Compress)
gzipStream.Close()
```

Une fois ces instructions exécutées, le fichier identifié par la variable FilePathToCompress avec l'extension " Bew " sera compressé. Il ne nous reste plus qu'à scinder ce fichier en plusieurs parties.

Le découpage du fichier

L'algorithme utilisé est le suivant :

- Création des flux vers le fichier source à scinder
- Récupération de la taille de chaque partie
- Parcours séquentiel du fichier source, récupération des octets équivalents à la taille désirée et création du nouveau fichier qui contiendra

Technique

les données lues précédemment.

• On itère l'algorithme précédent pour chacun des morceaux de fichier à créer. On crée donc un flux vers le fichier compressé identifié par la variable `FilePathToCompress`

```
Dim fileToSplit As IO.FileStream = File.Open(FilePathToCompress, FileMode.OpenOrCreate, FileAccess.Read)
```

On crée ensuite un Reader de type binary qui s'appuie sur le FileStream précédent

```
Dim reader As BinaryReader = New BinaryReader(fileToSplit)
```

On évalue la taille restante du fichier lu pour définir le nombre de fichiers à créer

```
Dim sizeRestant As Int32 = reader.BaseStream.Length - reader.BaseStream.Position
```

On crée une règle de nommage pour les morceaux de fichier, ils correspondront donc à cette règle " NomDuFichier ".Part " count ".bew comme implémenté ci-dessous :

```
Dim filePart As String = ".Part"  
Dim count As Int32 = -1
```

Tant qu'il y a des données à lire, nous créons un fichier (partie du fichier initiale) et nous écrivons les octets lus à l'aide d'un BinaryWriter. Enfin, nous réaffectons la taille restante en fonction du nombre d'octets lus.

```
While sizeRestant > 0  
  
    If sizeRestant > size Then  
        sizeRestant = size  
    End If  
  
    Dim value As Byte() = reader.ReadBytes(sizeRestant)  
  
    'création du fichier  
    count += 1  
    Dim FileName As String = filePart & count.ToString() + ".bew"  
    Dim fileToCreate As IO.FileStream = File.Open(Path.Combine("c:\",  
FileName), FileMode.CreateNew, FileAccess.Write)  
    Dim writer As New BinaryWriter(fileToCreate)  
    writer.Write(value)  
    writer.Flush()  
  
    writer.Close()  
    fileToCreate.Close()  
  
    sizeRestant = reader.BaseStream.Length - reader.BaseStream.Position  
End While  
  
reader.Close()  
fileToSplit.Close()
```

La décompression

Un prérequis au bon fonctionnement de notre application est que tous les morceaux de fichier devront être dans un même répertoire. Ils seront identifiés par leur nom. *Première étape de la décompression* : le rassemblement des morceaux de fichier si nécessaire. On crée une variable `internalFile` correspondant au fichier une fois décompressé

```
Dim internalFile As String = Path.Combine(FilePath, fileName)
```

Nous récupérons ensuite la liste des parties de fichiers, si nécessaire, en utilisant un filtre correspondant à la règle de nommage définie plus haut

```
Dim files As String() = Directory.GetFileSystemEntries(FilePath, "*part*.bew")  
Array.Sort(files)
```

Puis on effectue le rassemblement des morceaux de fichier dans un seul et même fichier

```
For index As Int32 = 0 To files.Length - 1  
  
    'Création d'un flux sur un fichier données  
    Dim filePart As FileStream = File.Open(files(index), FileMode.Open,  
FileAccess.Read)  
    Dim reader As New BinaryReader(filePart)  
    Dim value As Byte() = reader.ReadBytes(filePart.Length)  
  
    'Ouverture du fichier à décompresser  
    Dim fileToDecompress As FileStream = File.Open(internalFile, FileMode.Append,  
FileAccess.Write)  
    Dim writer As New BinaryWriter(fileToDecompress)  
    writer.Write(value)  
  
    'fermeture des flux  
    reader.Close()  
    writer.Close()  
    filePart.Close()  
  
Next
```

Seconde étape de la décompression : la décompression du fichier réuni. Pour décompresser le fichier, nous utilisons un flux de données GZipStream en mode décompression

```
Dim FileComplete As FileStream = File.Open(Path.Combine(FilePath,  
fileName), FileMode.OpenOrCreate, FileAccess.ReadWrite)  
  
Dim gzipStream As IO.Compression.GZipStream = New GZipStream(  
FileComplete, Compression.CompressionMode.Decompress)  
gzipStream.Close()
```

Nous avons donc bien réalisé notre outil en nous appuyant entièrement sur les classes fournies par le .NET Framework et plus particulièrement ceux de l'espace de noms `System.IO`

Conclusion

L'espace de noms principal pour la gestion des entrées/sorties avec le .NET Framework se nomme `System.IO`. On peut diviser ce dernier en deux grands pôles. Tout d'abord, un modèle objet permettant la gestion du système de fichiers : disque, répertoire et fichier. Puis un mécanisme de flux de données permettant de manipuler du contenu. Les informations des flux sont gérées via des Reader et Writer facilitant la lecture et l'écriture dans ces flux. Les flux de données sont aussi utilisés dans la gestion des communications TCP, des connexions via des sockets ou encore le pilotage du port série. Cela fera certainement l'objet d'un autre article.

■ **Sacha LEROUX**

Sacha.leroux@bewise.fr

Centre de compétences Team System - Bewise



Développement avec le .NET Compact Framework et Windows Mobile

Dès les premières versions des PDA sous Windows CE (appelés Palm Sized PC ou Handheld PC), Microsoft a proposé une interface de programmation (API) ainsi que des outils de développement proches des versions utilisées par les développeurs Windows.

Ainsi, le développeur disposait de :

- l'API Windows CE (et ses implémentations dans les systèmes dérivés de Windows CE comme Pocket PC ou maintenant Windows Mobile) est grosso-modo l'ensemble minimal (1) des fonctions Win32 permettant de couvrir les services offerts par Windows CE
- les premiers outils de développement pour Windows CE étaient soit des Add-on à Visual C++ (au temps de VC++5.0) soit des clones de Visual C++ ou Visual Basic.

Avec .NET et Visual Studio .NET 2005, la fusion des deux environnements de développement est maintenant achevée :

- Visual Studio .NET 2005 incorpore en standard les fonctions de développement pour toutes les plates-formes mobiles de Microsoft (à la fois pour le développement en code managé et pour le développement natif en C++),
- le .NET Compact Framework rend disponible sur plate-forme mobile un sous-ensemble important du .NET Framework :
 - les langages VB.Net et C# sont entièrement supportés (y compris avec les classes génériques),
 - un grand nombre de classes .NET sont disponibles, même si elles n'ont généralement qu'un sous-ensemble de leur équivalent sur desktop ou serveur.

Cet article va illustrer un certain nombre de concepts de programmation sous Windows Mobile à l'aide du .NET Compact Framework 2.0 en utilisant un exemple concret : la gestion d'une base de données de livres.

Contexte et architecture

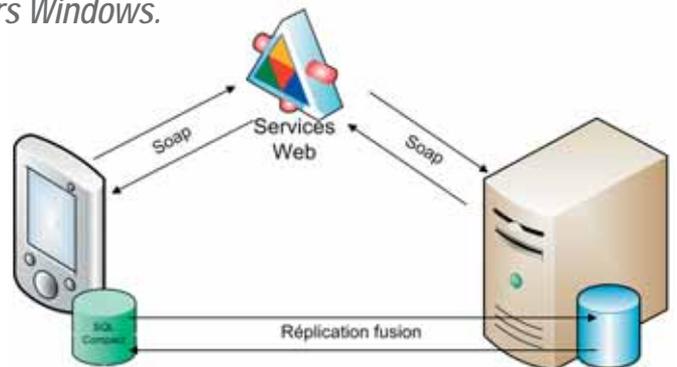
Pourquoi gérer ses livres sur mobile ? Essentiellement pour avoir un sujet de programmation ! Mais ça peut par exemple se justifier dans le cas d'un amateur de bandes dessinées qui possède de nombreuses séries incomplètes : avoir la liste des livres dans sa poche peut éviter d'acheter des doublons.

Cette application sera communicante :

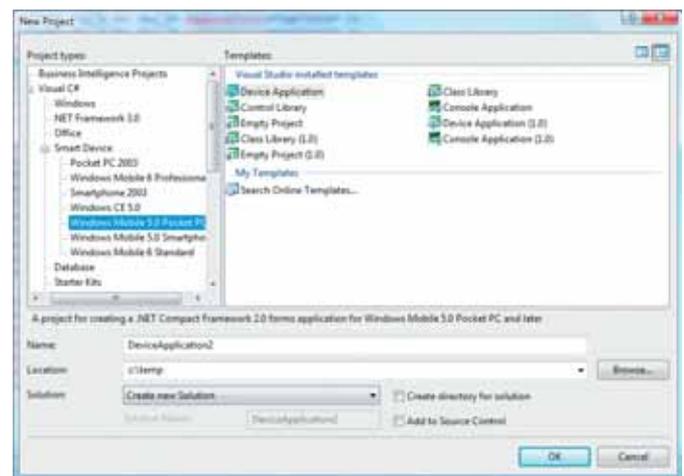
- elle va s'appuyer sur des services web pour récupérer les informations sur les livres (auteurs, couvertures, etc.),
- et la base de données sera synchronisée avec une base SQL Server sur un desktop ou un serveur : une application " jumelle " sur desktop permettra aussi l'enrichissement des données.

Les projets Windows Mobile sous Visual Studio .NET 2005

Les projets pour Windows Mobile se trouvent regroupés dans une catégorie " Smart Devices " sous le langage choisi. Sous cette catégorie, on



trouve ensuite toutes les plates-formes mobiles pour lesquelles un kit de développement a été installé : Visual Studio .NET 2005 vient avec le SDK Pocket PC 2003, les autres kits peuvent être téléchargés et installés indépendamment. Ils complètent l'installation de Visual Studio, apportent des exemples ainsi que des émulateurs permettant de déboguer les applications sans avoir le matériel réel.

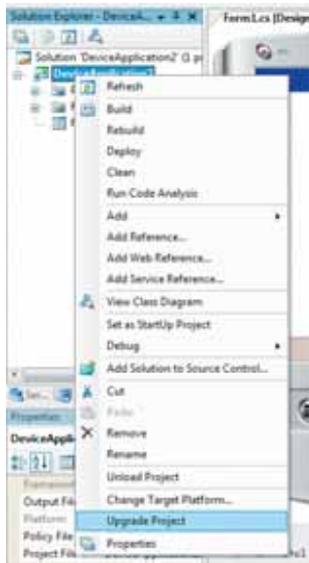


Une fois le projet créé, il a une structure similaire à son équivalent sur desktop.

Remarque : on voit avec les types de projet notés " (1.0) " qu'il est toujours possible de cibler le .NET Compact 1.0. Dans la mesure où le .NET Compact Framework 2.0 est plus rapide et plus complet, quel est l'intérêt de cibler l'ancienne version ? Eventuellement le déploiement, car la plupart des matériels en circulation incorporent le .NET Compact Framework 1.0 en ROM alors que ce n'est pas encore le cas pour le 2.0.

Il faut cependant noter qu'une application peut être par la suite " upgra-

(1) Lorsque Win32 propose plusieurs fonctions redondantes ce qui est assez fréquent, l'API de Windows CE ne retient que la fonction la plus riche, ce qui préserve les possibilités offertes au programmeur tout en réduisant les besoins mémoire des matériels sous Windows CE.



dée " vers le .NET CF 2.0 alors que l'opération inverse n'est pas possible. L'upgrade d'un projet est une opération irréversible et il est donc préférable de sauvegarder une copie des sources lors de cette opération.

En ce qui concerne le choix de la plate-forme, il n'est pas exclusif : un projet peut contenir, cibler plusieurs plates-formes et il est aussi possible d'en ajouter par la suite.

Le choix d'une plate-forme détermine quels assemblies pourront être référencés et quel émulateur sera lancé pour le débogage et définit aussi quelques symboles

pouvant être utilisés pour une compilation conditionnelle : " PocketPC " ou " Smartphone " .

Partage de code entre les différentes plates-formes

Bien qu'un certain nombre de classes soient communes entre le .NET Compact Framework et le .NET Framework " standard ", il est généralement préférable de ne pas chercher à partager des assemblies entre les deux plates-formes : les applications développées pour le .NET Compact Framework fonctionnent sur desktop mais l'inverse n'est pas vrai ; il est donc préférable de créer des assemblies différents pour les deux environnements, mais il est alors possible de partager les sources :

- créer le premier projet (par exemple " Windows / Class Library ") et y ajouter les fichiers
- créer le deuxième projet (" Smart Device / Windows Mobile 5.0 Pocket PC / Class Library " et y ajouter les fichiers avec " Add Existing Item " puis " Add as Link " :



L'utilisation de cette dernière option permet d'avoir le même fichier C# partagé entre les projets .NET Framework et .NET Compact Framework.

A l'intérieur des fichiers ainsi partagés, il est possible d'utiliser les directives de compilations définies par Visual Studio lorsque le code est dépendant de la plate-forme, soit avec la directive #if/#endif soit avec l'attribut Conditional :

```
private void AfficheNotification(string message) {  
    #if Smartphone  
        MessageBox.Show(message);  
    #else  
        Microsoft.WindowsCE.Forms.Notification notif  
            = new Microsoft.WindowsCE.Forms.Notification();  
        notif.InitialDuration = 2;  
        notif.Visible = true;  
    #endif  
}
```

Ou par exemple :

```
[Conditional("PocketPC")]  
void AfficheClavier(bool affiche) {  
    Microsoft.WindowsCE.Forms.InputPanel panel = new Microsoft.  
WindowsCE.Forms.InputPanel();  
    panel.Enabled = affiche;  
}  
  
private void txtTitre_GotFocus(object sender, EventArgs e) {  
    AfficheClavier(true);  
}  
  
private void txtTitre_LostFocus(object sender, EventArgs e) {  
    AfficheClavier(false);  
}
```

En ce qui concerne l'interface utilisateur, il faut savoir que depuis Windows Mobile 2003 Second Edition, les Pocket PC ou les Smartphones n'ont plus nécessairement tous la même résolution et peuvent éventuellement voir leur affichage tourné de 90°.

Pour l'essentiel, le .NET Compact Framework gère ces modifications, le principe d'utilisation est le même que pour les applications Windows Standard : on paramètre le redimensionnement et le remplacement des objets graphiques à l'aide de leur propriété Anchor. Cette adaptation fonctionne même lorsque le formulaire est déjà affiché.

L'utilisation des différents émulateurs permet de vérifier le comportement des fenêtres en fonction des différentes résolutions ou avec les rotations. Il faut noter que les dimensions limitées des écrans vont parfois entraîner l'affichage de barres de défilement (a priori dans une seule direction) :



Base de données

La base de données de référence sous Windows Mobile est SQL Server. Auparavant, il s'agissait d'une version spécifique mais la version actuelle, appelée SQL Server Compact Edition, est aussi disponible sur Desktop où elle se présente sous la forme d'une DLL facilement redistribuable avec une application.

Là encore, SQL Compact partage un certain nombre de concepts et en particulier une bonne partie du langage de programmation (Transact-SQL) avec les autres versions de SQL Server. Comme les classes spécifiques à SQL Compact (SqlCeConnection, SqlCeCommand, etc.) implémentent les mêmes interfaces que les classes SqlConnection ou



OleDbConnection, il est possible de partager une bonne partie du code entre l'application mobile et l'application desktop.

L'ouverture de la connexion à la base de données se fait en précisant le chemin de celle-ci :

```
SqlCeConnection cnx = new SqlCeConnection(
    @"Data Source=\Storage Card\SqlData\Livres.
sdf");
```

Une fois, la connexion établie, il est possible d'accéder de manière uniforme à SQL Server, Access (ou toute autre base accessible via OleDb) ou Sql Server Compact. Voici par exemple, le code d'insertion d'une nouvelle série de livres dans la base de données de test :

```
protected static void AddParameterWithValue(IDbCommand cmd, DbType type,
    string name, object value) {
    IDbDataParameter parameter = cmd.CreateParameter();
    parameter.ParameterName = name;
    parameter.DbType = type;
    if (value == null) {
        parameter.Value = DBNull.Value;
    } else {
        parameter.Value = value;
    }
    cmd.Parameters.Add(parameter);
}

public void Sauver(Serie serie) {
    using (IDbCommand cmd = _cnx.CreateCommand()) {
        cmd.Transaction = _trs;
        cmd.CommandType = CommandType.Text;
        if (serie.New) {
            cmd.CommandText = "insert into SERIE(ID, TITRE) values(@ID, @TITRE)";
        } else {
            cmd.CommandText = "update SERIE set TITRE=@TITRE where ID=@ID";
        }
        AddParameterWithValue(cmd, DbType.Guid, "@ID", serie.Id);
        AddParameterWithValue(cmd, DbType.String, "@TITRE", serie.Titre);
        int nbLignes = cmd.ExecuteNonQuery();
    }
}
```

Un des aspects les plus intéressants de SQL Compact est la synchronisation. En effet, beaucoup d'applications mobiles d'entreprises sont basées sur le scénario suivant : l'utilisateur télécharge des données sur son mobile le matin, il effectue ensuite une " tournée " pendant laquelle il enrichit les données récupérées, puis en fin de journée il met à jour la base centrale avec les données saisies.

SQL Mobile offre trois possibilités principales de synchronisation :

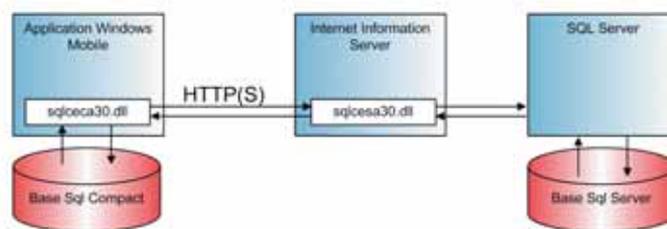
- Réplication fusion (merge replication) avec SQL Server
- Remote Data Access avec SQL Server
- Remote Data Access avec une base Access.

Note : Cette dernière possibilité n'est disponible qu'à l'état de CTP actuellement mais est prometteuse pour les utilisateurs particuliers.

La réplication fusion se base sur un des mécanismes de réplication de base de données offert par SQL Server. Le principe de la réplication fusion est le suivant : les bases répliquées sont synchronisées périodiquement, lors de ces synchronisations, les mises à jour effectuées sur chaque base sont transmises et appliquées à l'autre base. En cas de conflit, aucune donnée n'est perdue et les données en conflit sont enregistrées sur la base SQL Server de référence, sur laquelle il est ensuite possible de procéder à leur résolution; même en cas de conflit, les bases du mobile et du serveur sont identiques après la réplication. Ce type de réplication est bien adapté aux bases connectées épisodiquement.

Le Remote Data Access est plus spécifique aux applications mobiles (alors que la réplication fusion est utilisée dans de nombreux scénarios). Le mobile crée des tables localement à partir de requêtes SQL exécutées sur le serveur ; les modifications effectuées localement sur le mobile peuvent par la suite être rejouées sur le serveur SQL. Il n'y a pas vraiment de notion de synchronisation : on se contente de rejouer sur le serveur les ordres exécutés en mode déconnecté, les modifications effectuées entre temps sur le serveur ne sont pas redescendues sur le mobile lors de la synchronisation.

Dans les deux cas, l'architecture de synchronisation est la suivante : l'agent SQL " client " discute avec un agent SQL " serveur " qui est une DLL ISAPI hébergée dans Internet Information Server. C'est cette DLL et non l'agent " client " qui se connecte au serveur SQL. L'intérêt de cette architecture est évident dans les scénarios de type extranet car ceci évite de devoir exposer le serveur SQL principal sur internet.



La configuration de cette réplication se fait selon les étapes suivantes :

1. Création de la réplication fusion de la base SQL Server,
2. Configuration du répertoire IIS hébergeant l'agent de réplication SQL Compact.

Cette configuration peut être faite à l'aide d'un assistant ou manuellement. Le même répertoire peut servir à plusieurs bases de données.

La sécurisation de la réplication supporte toutes les options SQL et IIS :

- Authentification Windows ou SQL Server au niveau de SQL Server,
- Authentification basique, intégrée, etc. au niveau de IIS.

Dans le cadre de l'application exemple, un compte Windows spécifique a été créé (" SqlMobileUser "), IIS effectue cette authentification et prend l'identité correspondante pour effectuer l'accès à SQL Server. Le paramétrage de la réplication dans l'application mobile est le suivant :

```
SqlCeReplication repl = new SqlCeReplication();
// URL de l'agent de réplication sur IIS
repl.InternetUrl = "http://barney2003R2/sqlmobile/sqlcesa30.dll";
// Informations d'authentification
repl.InternetLogin = "SqlMobileUser";
repl.InternetPassword = motDePasse;
// Nom du serveur SQL
repl.Publisher = "barneyhpvista";
```

```
// Nom de la base répliquée
repl.PublisherDatabase = "Livres";
// Mode d'authentification auprès de SQL Server
repl.PublisherSecurityMode = SecurityType.NTAAuthentication;
// Nom de la réplication
repl.Publication = "Livres";
// Nom du client
repl.Subscriber = "PocketBD";
// Chaîne de connexion sur le PPC
repl.SubscriberConnectionString = @"Data Source=...";
```

La base SQL Compact peut être créée lors de la première synchronisation, il suffit pour cela d'ajouter l'option suivante au paramétrage précédent :

```
repl.AddSubscription(AddOption.CreateDatabase);
```

Enfin, la réplication peut être déclenchée de manière synchrone ou asynchrone. Dans le cas d'une réplication asynchrone, on peut préciser des fonctions de rappel permettant de suivre l'avancement de la synchronisation.

```
// Réplication synchrone
repl.Synchronize();
// Réplication asynchrone (AsyncReplicationData est une structure propre
// à l'application)
AsyncReplicationData data = new AsyncReplicationData(repl, target, event
Handler);
repl.BeginSynchronize(base_SyncCompletion, base_StartUpload, base_
StartDownload, base_Synchronisation, data);
```

Les principaux types de données de SQL Server sont supportés aussi par SQL Mobile (à l'exception notable des documents XML et des objets .NET) et peuvent être répliqués.

Ainsi, la petite application exemple stocke les couvertures des livres enregistrés dans la base dans des champs de type IMAGE qui sont répliqués sans difficulté. Dans la mesure où ces images sont de taille raisonnable, il est assez aisé de les manipuler :

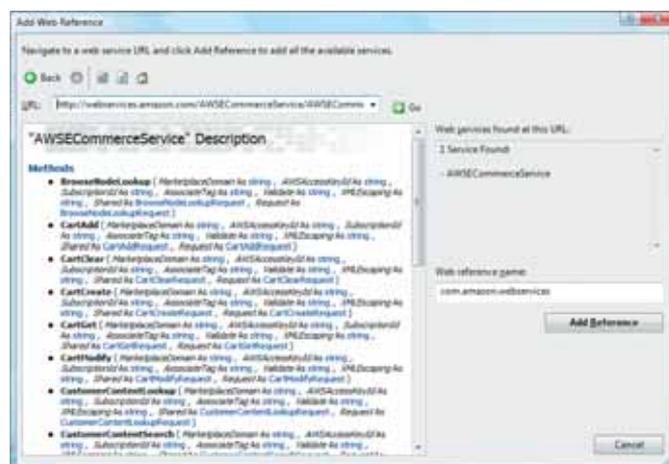
```
public void SauverImage(string id, byte[] image) {
    using (IDbCommand cmd = _cnx.CreateCommand()) {
        cmd.Transaction = _trs;
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = "insert into IMAGES(ID, BITMAPLEN, BITMAP)
values(@ID, @BITMAPLEN, @BITMAP)";
        AddParameterWithValue(cmd, DbType.String, "@ID", id);
        AddParameterWithValue(cmd, DbType.Int32, "@BITMAPLEN",
image.Length);
        IDataParameter paramBlob = cmd.CreateParameter();
        paramBlob.ParameterName = "@BITMAP";
        paramBlob.DbType = DbType.Binary;
        paramBlob.Value = image;
        cmd.Parameters.Add(paramBlob);
        cmd.ExecuteNonQuery();
    }
}
```

```
public byte[] ChargerImage(string id) {
    byte[] retour = null;
    using (IDbCommand cmd = _cnx.CreateCommand()) {
        cmd.Transaction = _trs;
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = "select BITMAPLEN, BITMAP from IMAGES
where ID=@ID";
        AddParameterWithValue(cmd, DbType.String, "@ID", id);
        using (IDataReader reader = cmd.ExecuteReader()) {
            if (reader.Read()) {
                int totalLength = 0;
                int len = reader.GetInt32(0);
                retour = new byte[len];
                while (totalLength < len) {
                    int lu = (int)reader.GetBytes(1, totalLength, retour, totalLength,
len - totalLength);
                    totalLength += lu;
                }
            }
        }
        return retour;
    }
}
```

A noter : dans le cas des images, la structure de la base a été adaptée à une limitation du Transact-SQL supporté par SQL Server Compact Edition. En effet, la fonction datalength qui serait bien pratique pour allouer le tableau d'octets lors de la lecture d'une image n'est pas disponible sous SQL Server Compact Edition : on a donc ajouté la colonne BITMAPLEN qui n'alourdit pas trop la table et permet de simplifier le code C#.

Appels de Services Web

Le Compact Framework supporte les appels de services web avec SOAP à l'aide des mêmes classes que la version .NET "classique". Notre application exemple s'appuie sur les services web proposés gratuitement par Amazon (certains services sont payants mais pas l'interrogation simple du catalogue). La génération de proxy se fait de manière habituelle :



Les Web Services d'Amazon étant assez riches, on peut encapsuler leurs appels dans des proxies dont le code peut être partagé entre la version desktop et la version mobile de l'application. Voici l'essentiel du code de recherche par titre :



```

public List<ResumeLivre> ParTitre(string critere, string page, out int nb
TotalResults, out int nbPages) {
    try {
        ItemSearch webRequest = new ItemSearch();
        ItemSearchRequest search = new ItemSearchRequest();
        webRequest.SubscriptionId = SubscriptionId;
        search.ResponseGroup = new string[] { "Small" }; //Large
        search.SearchIndex = "Books";
        search.Title = critere;
        search.ItemPage = page;
        webRequest.Request = new ItemSearchRequest[] { search };
        ItemSearchResponse webResponse = _service.ItemSearch(webRequ
est);
        Items response = webResponse.Items[0];
        int nbResults = response.Item.Length;
        nbTotalResults = SafeInt(response.TotalResults);
        nbPages = SafeInt(response.TotalPages);
        List<ResumeLivre> retour = new List<ResumeLivre>();
        for (int i = 0; i < nbResults; i++) {
            Item item = response.Item[i];
            ...
        }
    }
}

```

Ces web services ne récupèrent pas directement les images mais leurs URLs (en plusieurs tailles). Pour récupérer et stocker localement ces images, il est possible d'utiliser les classes habituelles WebRequest et WebResponse et aussi d'effectuer ces appels de manière asynchrone :

```

// Récupération asynchrone de l'image
WebRequest request = WebRequest.Create(_details.ImagePath);
request.BeginGetResponse(TelechargerImage, request);
...
private void TelechargerImage(IAsyncResult iar) {
    WebRequest request = iar.AsyncState as WebRequest;
    using (WebResponse response = request.EndGetResponse(iar)) {
        using (Stream responseStream = response.GetResponseStream()) {
            _imageData = new byte[response.ContentLength];
            int total = 0;
            while (total < _imageData.Length) {
                int lus = responseStream.Read(_imageData,
                total, _imageData.Length - total);
                total += lus;
            }
            this.Invoke(new VoidDelegate(AfficherImage));
        }
    }
}

```

La méthode TelechargerImage étant appelée de manière asynchrone, elle va s'exécuter sur un thread différent du thread gérant l'interface utilisateur.

Sur Windows Mobile comme sur les autres versions de Windows, il est préférable de toujours manipuler une fenêtre et ses composants à l'aide du même thread (certaines opérations peuvent être effectuées par d'autres threads mais ce n'est pas le cas général). L'appel à "this.Invoke" dans l'extrait ci-dessus permet d'appeler AfficherImage sur le thread de l'interface utilisateur.

```

private void AfficherImage() {
    if (pictureBox1.Image != null) {
        pictureBox1.Image.Dispose();
    }
    using (MemoryStream ms = new MemoryStream(_imageData)) {
        pictureBox1.Image = new Bitmap(ms);
    }
}

```

Gestion des ressources

Windows Mobile est un environnement puissant mais malgré tout bien plus contraint en termes de ressources (CPU et mémoire) que les versions habituelles de Windows. Il est donc nécessaire de gérer la libération des ressources avec la plus grande rigueur. En particulier, l'utilisation de l'interface IDisposable et de la structure de contrôle using associée doit être systématique :

- Lorsqu'une classe encapsule une ressource coûteuse (connexion base de données, fichier, objet natif), elle doit implémenter l'interface IDisposable.
- Lorsqu'on utilise une telle classe, on doit utiliser using.

Dans notre exemple, l'objet encapsulant les accès aux données conserve à la fois une connexion à SQL Server CE et un objet transaction (certaines opérations de mise à jour sont réparties sur plusieurs tables et nécessitent donc l'utilisation d'une transaction pour garantir l'intégrité de la base). Il implémente donc l'interface IDisposable :

```

public interface IDataAccess : IDisposable {
    // Méthodes sur les séries
    List<Serie> ListeSeries();
    void Sauver(Serie serie);
    ...
    // Validation / Annulation transaction
    void Commit();
    void Rollback();
}

```

Le destructeur et la méthode Dispose annulent la transaction si elle n'a pas été validée puis la libèrent ainsi que la connexion SQL associée :

```

~DataAccessHelper() {
    Dispose(false);
}

public void Dispose() {
    Dispose(true);
    GC.SuppressFinalize(this);
}

protected void Dispose(bool disposing) {
    if (!_disposed) {
        if ((_trs != null) && (!_committed)) {
            _trs.Rollback();
        }
        if (disposing) {
            // Libération des objets managés
            if (_trs != null) _trs.Dispose();
        }
    }
}

```



```
if (_cnx != null) _cnx.Dispose();
}
_disposed = true;
}
}
```

Si la gestion rigoureuse des objets d'accès aux bases de données est assez classique, les développeurs négligent en revanche assez fréquemment les objets Winform. Or un formulaire Winform encapsule des ressources Windows et implémente pour cette raison l'interface IDisposable. Oninstanciera donc les formulaires à l'intérieur de structures using :

```
private void mitParcourir_Click(object sender, EventArgs e) {
    using (FrmLivre dlg = new FrmLivre()) {
        dlg.Local = true;
        dlg.Livres = _livres;
        dlg.ShowDialog();
    }
}
```

Une alternative souvent employée sous Windows Mobile est le recyclage des objets graphiques : en effet, le code ci-dessus instancie un nouveau formulaire à chaque fois qu'on veut afficher la fenêtre de parcours des livres d'une série puis libère la mémoire une fois la fenêtre fermée. Il est donc envisageable de conserver le formulaire dans une variable globale et de le réutiliser lors des prochains affichages.

Dans le même ordre d'idée, certaines applications pour Windows mobile définissent un User Control par écran et affichent tous ces contrôle dans un formulaire unique qui reste affiché en permanence.

Un des intérêts de cette approche est de ne faire apparaître qu'une seule entrée correspondant à l'application dans la liste des tâches en cours d'exécution. La copie d'écran ci-contre montre ce qui se passe quand on empile des fenêtres, même s'il s'agit de boîtes de dialogue modales.

Cette approche est cependant un peu plus délicate à mettre en œuvre au niveau de la gestion de la mémoire et en particulier de la suppression des objets stockés dans le User Controls.

Il n'y a pas de solution idéale pour l'ensemble des situations et c'est au développeur de faire les bons choix d'implémentation en fonction de la

complexité de ses formulaires et de leur nombre : on gagnera à ne pas supprimer un formulaire complexe affiché souvent. En revanche, il faut penser à libérer explicitement les objets " métier " conservés en tant que données membre de tels formulaires et qui ne servent plus.

Par exemple, le formulaire de recherche sur internet de notre exemple conserve en tant que membre les résultats de la recherche courante afin de pouvoir les passer à la fenêtre de visualisation des détails :

```
public partial class FrmRecherche : Form {
    ...
    private List<ResumeLivre> _resultat;
    ...
}
```

Si on voulait rendre ce formulaire réutilisable en le masquant au lieu de le détruire, il faudrait alors mettre la variable resultat à null pour permettre la collecte de cette collection inutile lorsque le formulaire n'est pas affiché.

Conclusion

Au travers d'un certain nombre de mécanismes, on a vu que la programmation .NET sur Windows Mobile était assez proche de la programmation Winform " classique " et qu'une bonne partie du code pouvait être réutilisée entre ces différents types d'applications. Windows Mobile associée au .NET Compact Framework est la plate-forme idéale pour développer des solutions mobiles qui peuvent fonctionner de manière autonome (avec une éventuelle connexion régulière pour échanger des données avec une base de référence) ou connectées. Le futur .NET 3.5 et Visual Studio .NET 2008 continueront dans cette voie en offrant un .NET Compact Framework 3.5 riche en nouveautés.

■ **Alain Zanchetta**

Consultant Architecte dans la division Services de Microsoft France



La BOUTIQUE

TÉLÉCHARGEZ LES ANCIENS NUMÉROS
ET LES ARTICLES EN PDF.

VOUS POUVEZ ÉGALEMENT VOUS ABONNER EN LIGNE
www.programmez.com



Les bases de LINQ avec LINQ To Objects

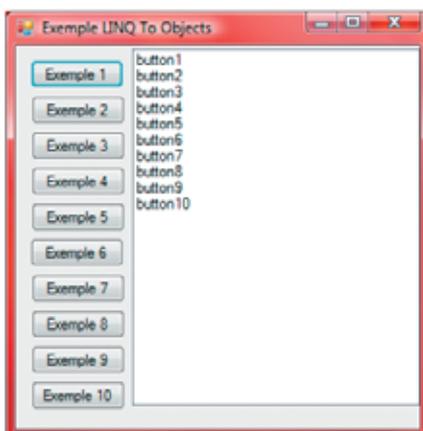
Le framework .NET en version 3.5 révolutionne les langages .NET en leur apportant la capacité de requêter des ensembles de données. Explications sur Language INtegrated Query avec son premier descendant : LINQ To Objects.

L'objectif de LINQ est de donner aux langages la capacité de traiter les données de manière *ensembliste* comme le fait SQL. Puisque les requêtes sont écrites au niveau du langage de programmation, elles peuvent porter sur différentes sources de données : base de données bien sûr avec LINQ To SQL et LINQ To Entities, mais aussi XML avec LINQ To XML ou n'importe quelle source comme les collections d'objets avec LINQ To Objects.

Écrire une requête simple

C# et VB intègrent de nouveaux mots-clés pour écrire ces requêtes : ce sont les expressions de requête. Voici, par exemple, comment récupérer, dans un formulaire, les noms des contrôles qui commencent par "button", triés par index de tabulation :

```
IEnumerable<string> controles =
    from Control c in this.Controls
    where c.Name.StartsWith("button")
    orderby c.TabIndex
    select c.Name;
```



LINQ To Objects permet de requêter n'importe quelle collection (appelée séquence) grâce à la clause *from*, du moment qu'elle implémente *IEnumerable<T>*. Si elle implémente simplement *IEnumerable*, alors il est nécessaire de spécifier le type de l'élément (appelé variable

d'itération). C'est pour cela que dans l'exemple nous spécifions le type de *c* : *Control*.

A la manière d'une requête SQL, il est possible de filtrer par une expression quelconque derrière la clause *where*. Le tri s'effectue de manière tout aussi classique avec la clause *orderby* sur une ou plusieurs clés.

De manière simplifiée voici la forme d'une requête en C# :

```
IEnumerable<type> resultat =
    from [Type] element in collection
    [where condition]
    [orderby expression [ascending | descending], ...]
    select expression;
```

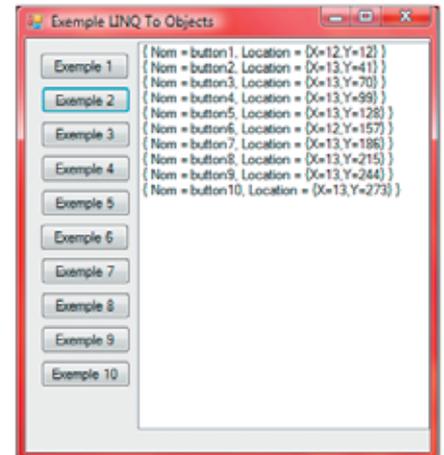
La clause *select* est à la fin de la requête pour faciliter l'IntelliSense et doit renvoyer une valeur. Celle-ci peut être simple (une chaîne de caractère par exemple) ou constituée de plusieurs valeurs (une chaîne de caractère et un entier par exemple). Le retour de la requête est un *IEnumerable<T>* où *T* est le type de l'expression retournée par la clause *select*. Dans notre exemple, c'est le nom du contrôle, donc une chaîne de caractères. Cependant, comment retourner le nom du contrôle ainsi que sa position ? Il serait nécessaire de créer un type avec deux propriétés (Nom et Location) et l'affecter en une seule expression. Les types anonymes ont justement cette vocation.

Intérêt du type anonyme et du typage implicite des variables locales

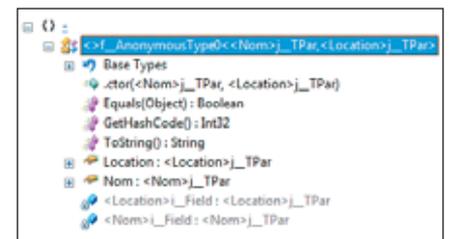
Pour récupérer le nom et la position du contrôle, nous pouvons écrire ceci :

```
from Control c in this.Controls
where c.Name.StartsWith("button")
orderby c.TabIndex
select new {Nom = c.Name, c.Location};
```

Le compilateur va se charger de créer une classe héritant d'*Object* avec les propriétés *Nom* et *Position*. Il va même implémenter une redéfinition de *ToString*, *Equals* et *GetHashCode*. La classe est construite et une instance



est initialisée à partir des valeurs. Voici ce qui est généré :



Voici la syntaxe simplifiée des types anonymes en C# :

```
New {[PropertyName1 = ]element.property1,
     [NomPropriete2 = ]element.property1...}
```

Si aucun nom de propriété n'est spécifié et que l'objet est initialisé à partir d'un objet, c'est le nom de sa propriété qui est utilisé.

Mais maintenant comment typer le retour de cette requête ? Il faudrait connaître le nom du type anonyme. Ce qui est pour le moins difficile. Il est pour cela possible d'utiliser le mot-clé *var* en C#.

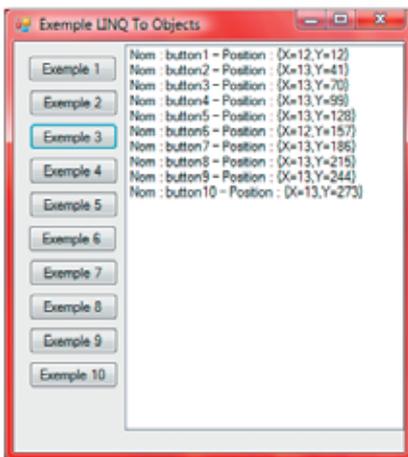
```
var controles =
    from Control c in this.Controls
    select new {Nom = c.Name, c.Parent};
```

Il est important de comprendre que cette déclaration n'enlève par le typage fort de la variable. Si, dans notre exemple nous

essayons d'affecter une chaîne de caractères à " resultat " nous n'y arriverons pas ; non pas parce qu'une exception est levée mais parce que la compilation elle-même échoue. Ce mot-clé n'est utile que pour le développeur. Le compilateur connaissant le nom du type anonyme fait " simplement " le remplacement. Ce qui permet à Visual Studio 2008 de proposer l'IntelliSense sur ces variables.

Le type étant défini, il est par exemple possible de l'utiliser dans une boucle :

```
foreach (var element in controles) {
    listBox1.Items.Add("Nom : " + element.Nom
        + " - Position : " + element.Location);
}
```



Comment s'exécute la requête ?

Nous avons vu la syntaxe pour créer une requête. Mais techniquement comment cette exécution est-elle réalisée ? Pourquoi mon type doit-il être un IEnumerable<T> ?

L'expression de requête, vue plus haut, n'est qu'une forme simplifiée d'appels à des méthodes. Notre requête s'écrit ainsi :

```
IEnumerable<string> controles = this.Controls.
    Where(...).OrderBy(...).Select(...);
```

La question qui se pose : comment sont arrivées ces méthodes ? Le type IEnumerable<T> est-il modifié ? En fait, il est plutôt étendu grâce aux méthodes d'extension.

Intérêt des méthodes d'extension

Les méthodes Where, OrderBy etc. ne sont pas réellement sur le type IEnumerable<T> mais dans une classe nommée Enumerable. L'astuce consiste à les déclarer de telle manière qu'elles indiquent étendre le type

IEnumerable<T>. Notre appel ressemble donc plutôt à :

```
IEnumerable<type> resultat =
    Enumerable.Select(Enumerable.OrderBy(
        Enumerable.Where(collection, ...), ...)...
```

Il est important donc de comprendre que les méthodes ajoutées à IEnumerable<T> ne sont pas des méthodes d'IEnumerable<T> mais d'un autre type. Elles ne sont vues comme des méthodes IEnumerable<T> que par les développeurs. A la compilation, les appels sont transformés. Pour écrire une méthode d'extension en C#, il faut que celle-ci soit marquée comme statique et que le premier paramètre soit précédé du mot clé *this* correspondant au type étendu. Voici la déclaration du Where par exemple :

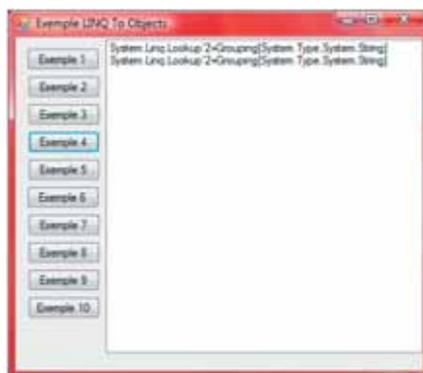
```
public static IEnumerable<T> Where<T>(this
    IEnumerable<T> source, ...);
```

Requêtes riches : regroupement et jointures

Tout ceci est fort intéressant, mais nous ne nous sommes occupés que de requêtes simples. Dans la pratique nous allons devoir regrouper les données et travailler sur plusieurs jeux de données.

Nous pouvons par exemple regrouper nos contrôles par type ainsi :

```
var controles =
    from Control c in this.Controls
    group c by c.GetType();
```



La clause *group* se positionne en lieu et place du *select* :

```
group expression by expression
```

Si le regroupement souhaite être filtré, trié etc. il est nécessaire d'utiliser la clause *into* pour

enchaîner les requêtes. Si, dans notre exemple, nous ne souhaitons récupérer de façon groupée que les noms des contrôles qui ne sont pas de type Button, il faudrait écrire ceci :

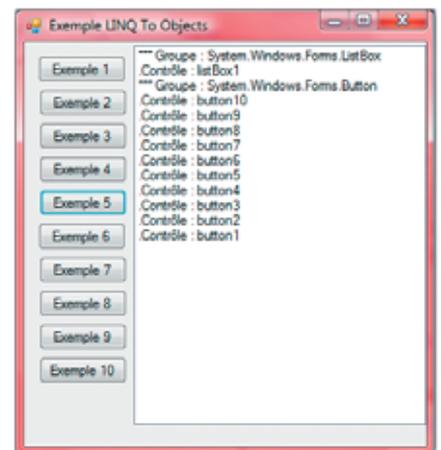
```
var controles =
    from Control c in this.Controls
    group c.Name by c.GetType() into g
    where g.Key != typeof(Button)
    select g ;
```

La syntaxe simplifiée en C# est la suivante :

```
requete into identificateur requete
```

Nous pouvons alors les afficher ainsi :

```
var controles =
    from Control c in this.Controls
    group c.Name by c.GetType() into g
    select g ;
foreach (var g in controles) {
    listBox1.Items.Add("*** Groupe : " + g.Key.
        FullName);
    foreach (string nom in g) {
        listBox1.Items.Add(" .Contrôle : " + nom);
    }
}
```



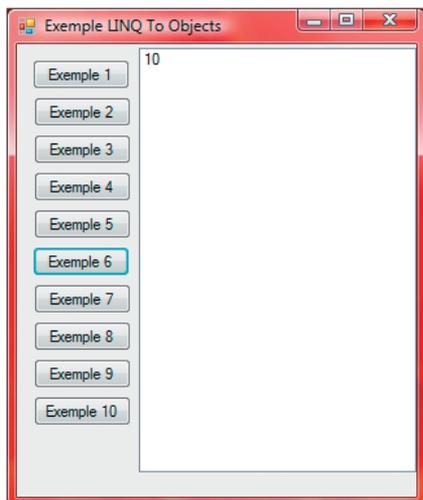
Notez au passage que la propriété *Key* est disponible sur le regroupement (le type du contrôle dans notre exemple). Le regroupement est de type *IGrouping<K, T>* qui hérite d'*IEnumerable<T>* et ajoute cette propriété *Key* bien utile.

Qui dit regroupement dit fonctions d'agrégation. Nous retrouvons les fonctions d'agrégation classiques étendant le type *IEnumerable<T>* :

Instruction	Définition
All	vrai si tous les éléments satisfont une même condition
Any	vrai si un des éléments satisfait une condition
Average	moyenne des éléments de la séquence
Count	nombre d'élément de la séquence
First	premier élément de la séquence
Last	dernier élément de la séquence
Max	valeur maximale de la séquence
Min	valeur minimale de la séquence
Sum	somme des éléments de la séquence

Dans notre cas, nous pouvons compter le nombre de contrôles de chaque type ainsi :

```
var controles =
    from Control c in this.Controls
    group c by c.GetType() into g
    where g.Key != typeof(Button)
    select g.Count();
```

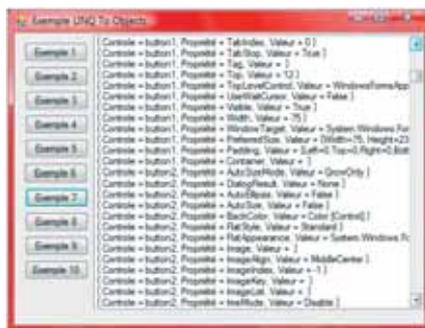


Le second besoin est de travailler sur plusieurs jeux de données. Trois cas se présentent :

1. Réaliser un produit cartésien (équivalent du CROSS JOIN en SQL)

Les requêtes peuvent tout simplement prendre plusieurs clauses *from* (et le *where* et *orderby* correspondant). Toutes les combinaisons sont alors exécutées à la manière de *foreach* imbriqués. Pour récupérer les valeurs de toutes les propriétés des différents contrôles du formulaire, nous pouvons écrire la requête suivante

```
var controles =
    from Control c in this.Controls
    orderby c.TabIndex
    from p in c.GetType().GetProperties()
    select new {
        Controle = c.Name,
        Propriété = p.Name,
        Valeur = p.GetGetMethod().Invoke(c, null) };
```



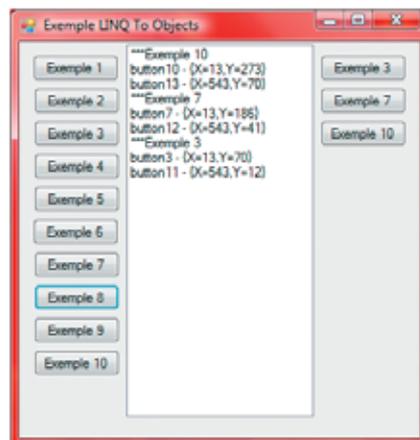
La syntaxe simplifiée en C# est la suivante :

```
from [Type1] element1 in collection1
[where condition1]
[orderby expression1 [ascending | descending], ...]
[
    from [Type2] element2 in collection2
    [where condition2]
    [orderby expression2 [ascending | descending], ...]
    ...]
select expression;
```

2. Réaliser une jointure simple (équivalent de l'INNER JOIN en SQL)

C'est le cas le plus courant : faire un lien entre les jeux de données. A la manière de SQL, LINQ introduit la clause *join*. Pour trouver tous les boutons qui ont le même nom et une position différente, nous pouvons écrire :

```
var controles =
    from Control c1 in this.Controls
    join Control c2 in this.Controls on c1.Text equals c2.Text
    where c1.Left == 13 && c2.Left > 13
    select new { Controle1 = c1, Controle2 = c2 };
```



La syntaxe simplifiée en C# est la suivante :

```
from [Type1] element1 in collection1
join [Type2] element2 in collection2 on element1.propriete1 equals element2.propriete2
```

3. Réaliser une jointure gauche (équivalent de LEFT JOIN)

Ce cas permet de récupérer tous les éléments de la séquence de gauche (même si aucun élément dans la séquence de droite ne peut être joint) et de les regrouper à la manière du *groupby* que nous avons vu. Si l'on veut récupérer tous les contrôles de gauche et pour ceux qui ont un homologue à droite l'afficher cela s'écrit ainsi :

```
var controlesGauche =
    from Control c1 in this.Controls
    where c1.Left == 13
    select c1;
var controlesDroite =
    from Control c1 in this.Controls
    where c1.Left > 13
    select c1;
var controles =
    from c1 in controlesGauche
    join c2 in controlesDroite on c1.Text equals c2.Text into g
    select new { Controle1 = c1, Groupe = g };
foreach (var element in controles)
{
    listBox1.Items.Add("*** Controle gauche : " + element.Controle1);
    foreach (var controle in element.Groupe)
    {
        listBox1.Items.Add(". Controle droite : " + controle);
    }
}
```



La syntaxe simplifiée en C# est la suivante :

```
from [Type1] element1 in collection1
join [Type2] element2 in collection2 on element1.propriete1 equals element2.propriete2
into g
...
```

Notez qu'un ensemble de fonctions permettant de travailler sur les jeux de données est fourni :

Instruction	Définition
Distinct	uniquement les éléments différents
Concat	les éléments de deux séquences
Union	les éléments différents de deux séquences
Intersect	les éléments communs à deux séquences
Except	les éléments sauf ceux précisés
ElementAt	un élément particulier
Take	les x premiers éléments de la sequence
Skip	les éléments après le xième
TakeWhile	les éléments jusqu'à un en particulier
SkipUntil	les éléments à partir d'un en particulier
Reverse	les éléments dans l'autre sens
SelectMany	tous les éléments " aplatis "

Intérêt des Expressions Lambda

Il nous reste à comprendre le passage des expressions du *Where*, *Select* etc. Ce ne sont en fait que des instructions passées aux méthodes (*Where*, *Select* etc.) qui seront invoquées lors de l'exécution de la méthode. Pour passer ces fonctions en paramètre d'une méthode, qui l'invoquera, on utilise habituellement, le mécanisme de délégué. Cependant, la syntaxe du délégué est un peu lourde (pour l'exemple précédent il serait nécessaire de déclarer et instancier trois délégués et de créer les trois méthodes s'y rapportant). Le délégué anonyme, introduit avec le framework 2.0, permet de simplifier l'écriture mais reste encore assez verbeux car il nécessite notamment de préciser le type des paramètres.

Une façon encore plus concise existe désormais grâce à une autre nouveauté : les expressions lambda. Toute méthode ayant un argument de type *Func* peut être utilisée avec une expression lambda. La méthode *Where* est par exemple déclarée ainsi :

```
public static IEnumerable<T> Where<T>(this
IEnumerable<T> source, Func<T, bool> predicat);
```

L'écriture de l'expression lambda est très légère. Pour reprendre notre exemple, nous pouvons réécrire notre première requête ainsi :

```
IEnumerable<type> resultat =
    Enumerable.Select(
        Enumerable.OrderBy(
            Enumerable.Where(
                collection,
                c => c.Anchor != AnchorStyles.
None),
                c =>
c.TabIndex),
        c => c.Name);
```

Vous voyez que l'expression lambda peut se résumer en *C#* à *parametre1 => expression*

car les types des paramètres peuvent être déduits de la signature de la méthode.

Plus généralement, les syntaxes simplifiées en *C#* sont les suivantes :

- Si vous n'avez qu'une expression à retourner :

```
((Type parametre1, Type parametre2, ...) =>
expression
```

- Si vous avez plusieurs instructions à exécuter avant de retourner un résultat :

```
((Type parametre1, Type parametre2, ...) =>
{instructions})
```

Exécution différée

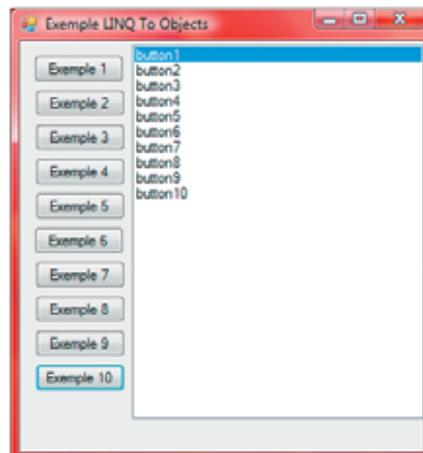
Il est important de noter que nous avons câblé la requête en lui donnant des bouts de codes via les expressions lambda et que son exécution se fait lors de l'itération et non à la déclaration de la requête.

Si nous voulons exécuter " toute " la requête nous pouvons utiliser les méthodes suivantes :

Instruction	Définition
ToArray	un tableau contenant les éléments
ToList	une <i>List<T></i> contenant les éléments
ToDictionary	un <i>Dictionary</i> clé/valeur - relation 1/1
ToLookup	un <i>Lookup</i> clé/valeurs - relation 1/n

Ceci est utile pour le databinding qui n'accepte pas d'*IEnumerable* mais des *IList*. Voici comment lier le résultat de ma requête à ma *listbox* :

```
IEnumerable<string> controles =
    from Control c in this.Controls
    orderby c.TabIndex
    orderby c.TabIndex
    select c.Name;
listBox1.DataSource = controles.ToList();
```



elle-même itère sur le retour de la méthode *Where*. Chacune des itérations invoque l'expression lambda et retourne l'itération courante si nécessaire grâce à un *yield* et rend donc la main à l'appelant. Ceci permet d'économiser au maximum les traitements à effectuer.

Ce mécanisme est très intéressant lorsque l'on travaille sur des objets en mémoire, mais lorsque l'on souhaite transformer la requête LINQ en requête spécifique (en SQL par exemple) cela pose des problèmes. En effet, cela nécessiterait de récupérer toute la table en mémoire ou de travailler avec un curseur sur la base de données. Ni l'une, ni l'autre des solutions n'est acceptable.

Heureusement, nos expressions lambda peuvent se compiler classiquement en délégués mais aussi en un arbre d'expression.

Intérêt des arbres d'expression

L'arbre d'expression est une structure de données permettant de stocker de manière abstraite les expressions à effectuer sous la forme d'un opérande gauche, un opérateur et un opérande droit. Ceci permet de construire une requête en mémoire et de la transformer par exemple en SQL pour l'exécuter en une seule fois et récupérer le résultat final. Tandis que la forme " délégué " construit au fur et à mesure de l'itération le résultat, la forme " arbre d'expression " réalise l'exécution et retourne le résultat dès le début de l'itération (en fait lors du *GetEnumerator*).

Pour cela, la méthode demandant une expression lambda sous forme d'un arbre d'expression doit être typée non pas avec le délégué *Func<>* mais avec la classe *Expression<Func<>>*. Les providers LINQ utilisant cette fonctionnalité vont alors implémenter non pas l'interface *IEnumerable<T>* jusqu'ici utilisée mais l'interface *IQueryable<T>* qui en dérive et permet de gérer les spécificités à ce mode d'exécution. Ce n'est pas le cas de LINQ To Objects mais par exemple de LINQ To XML ou LINQ To SQL. Mais ceci est un autre sujet...

Conclusion

LINQ To Objects offre à l'aide d'une syntaxe concise mais puissante tous les outils pour requêter finement les données les plus utilisées dans nos programmes : les objets en mémoire.

■ Yann Faure – Regional Director
Yann.faure@bewise.fr - Bewise

Le Databinding avec Windows Presentation Foundation (WPF)

Le Databinding représente un des concepts les plus importants de la programmation Windows, qu'il s'agisse de développement Web ou WindowsForms. Au travers de cet article, nous allons voir comment celui-ci a évolué pour être rendu plus simple et plus performant, en terme de développement WPF.

Le Databinding est la technique utilisée pour lier des éléments de votre interface graphique à des données simples ou complexes. Le but étant de faciliter la mise à jour de l'un ou l'autre (ou même des deux) lorsqu'une modification est interceptée par le système. Cette technologie, qui au premier abord peut sembler magique de par son automatisation totale, repose en fait sur un framework très riche et très puissant.

Databinding en XAML

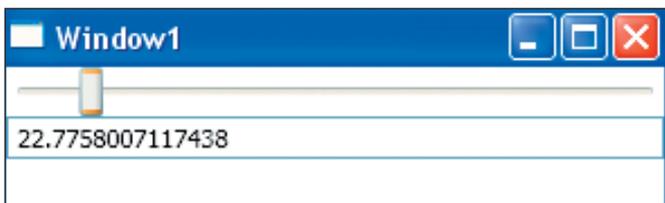
Pour réaliser une opération de Databinding (liaison de données) en XAML, c'est extrêmement simple : il suffit d'utiliser la bonne syntaxe, autrement dit : {Binding}. Ensuite, il faut savoir qu'il y a plusieurs types de binding. Le premier consiste à lier un élément de l'interface graphique à un autre élément de l'interface. Pour cela, il faut ajouter deux paramètres à notre syntaxe :

- ElementName, qui est utilisé pour indiquer la référence vers l'élément source
- Path, qui sert à indiquer quelle propriété de l'élément doit être utilisée pour réaliser le binding

Au niveau du code, cela donne donc quelque chose comme ceci :

```
<Slider x:Name="slider" Minimum="0" Maximum="200" />
<TextBox x:Name="tb" Text="{Binding ElementName=slider, Path=Value}" />
```

Si vous exécutez l'application, vous obtenez quelque chose comme ceci :



Si vous déplacez le slider, vous remarquerez que la valeur dans la zone de texte est modifiée également. A l'inverse, si vous saisissez une valeur dans la zone de texte, et que vous cliquez sur " Tabulation " (autrement dit dès que le slider perd le focus), la position du slider changera.

Un autre type de binding existant est le binding sur objet. Imaginez que vous ayez une classe, nommée ProductCollection (comme son nom l'indique, il s'agit d'une collection/liste de Product) : (voir figure suivante) Nous allons devoir, à présent, indiquer à notre application qu'elle peut utiliser cette collection comme source de données. Pour cela, nous

```
public class Product
{
    public int ProductId { get; set; }
    public string ProductName { get; set; }
    public double ProductPrice { get; set; }

    public Product(int id, string name, double price)
    {
        this.ProductId = id;
        this.ProductName = name;
        this.ProductPrice = price;
    }
}

public class ProductCollection : ObservableCollection<Product>
{
    public ProductCollection()
    {
        this.Add(new Product(1, "Livres", 50.0));
        this.Add(new Product(2, "Jeux", 250.0));
        this.Add(new Product(3, "CD", 75.15));
        this.Add(new Product(4, "DVD", 125.5));
    }
}
```

allons utiliser un ObjectDataProvider, en lui passant en paramètre le type de l'objet que nous voulons utiliser comme source de données. Pour cela, il suffit d'utiliser la syntaxe suivante :

```
<ObjectDataProvider x:Key="products" ObjectType="{x:Type local:ProductCollection}" />
```

Une fois cette opération effectuée, il nous reste à indiquer quel sera le DataContext à utiliser par l'application (ou par un élément en particulier). Le DataContext est un concept nouveau, introduit avec Windows Presentation Foundation, qui peut-être apparenté à la propriété DataSource que l'on retrouvait avec .NET 2 : il s'agit de la définition de la source de vos données. Pour indiquer ce DataContext, là encore vous allez devoir faire appel à du Databinding :

```
<ListBox DataContext="{Binding Source={StaticResource products}}" />
```

Et maintenant me direz-vous ? Et bien c'est terminé ! Enfin, pas tout à fait. En effet, il nous reste à indiquer à notre ListBox quelle est la source de données à utiliser pour afficher les éléments. Nous allons utiliser le DataContext, donc nous allons faire du binding sur l'élément ListBox lui-même :

```
<ListBox DataContext="{Binding Source={StaticResource products}}"
ItemsSource="{Binding}" />
```

Il est à noter que vous pourrez également rencontrer la syntaxe suivante, identique à ce que j'ai écrit :

```
<ListBox DataContext="{Binding Source={StaticResource products}}"
ItemsSource="{Binding Path=." />
```

Pour peaufiner un peu, on applique un DataTemplate (template appliqué à des données et non des contrôles) aux éléments de notre ListBox :

```
<ListBox DataContext="{Binding Source={StaticResource products}}"
ItemsSource="{Binding}" ItemTemplate="{StaticResource productTemplate}" />
```

productTemplate étant défini de la manière suivante :

```
<DataTemplate x:Key="productTemplate">
  <StackPanel Orientation="Horizontal">
    <TextBlock Text="{Binding Path=ProductName}" />
    <TextBlock Text=" " xml:space="preserve" />
    <TextBlock Text="{Binding Path=ProductPrice}" />
  </StackPanel>
</DataTemplate>
```

Bien sûr, vous n'êtes pas obligé d'utiliser un ObjectDataProvider pour faire votre binding. Rien ne vous empêche d'utiliser, directement, la collection que vous avez créée :

```
<local:ProductCollection x:Key="products" />
```

Quoiqu'il arrive, le résultat est toujours le même :



DataBinding par le code

Une des puissances de Windows Presentation Foundation, est que tout ce que l'on fait en XAML (Extensible Application Markup Language) peut être refait avec du code (attention, l'inverse n'est pas vrai : vous ne pouvez, par exemple, pas instancier un Web Service depuis votre code XAML). Ainsi, même le binding peut-être refait. Par exemple, si l'on prend le cas du binding sur élément, le code est simple à mettre en œuvre :

```
Binding elementBinding = new Binding();
elementBinding.ElementName = this.slider.Name;
elementBinding.Path = new PropertyPath("Value");

this.tb.SetBinding(TextBox.TextProperty, elementBinding);
```

Comme vous pouvez le constater, cela ressemble très fortement à la version XAML du binding. La seule différence réside dans l'utilisation de la méthode SetBinding : qui sert à indiquer quelle Dependency Property

doit être bindée (car je le rappelle, seules les Dependency Properties peuvent-être bindées, animées, transformées, etc....).

Dans le cas du binding sur objet, le code est encore plus simple :

```
Binding productItemsBinding = new Binding();
productItemsBinding.Source = this.TryFindResource("products");

this.lb.SetBinding(ListBox.DataContextProperty, productItemsBinding);
```

Là encore, on remarque l'inévitable appel à la méthode SetBinding. Cependant, on voit que l'on n'utilise plus la propriété ElementName ou Path : on passe directement par la propriété Source, qui est utilisée pour indiquer la source de données du binding !

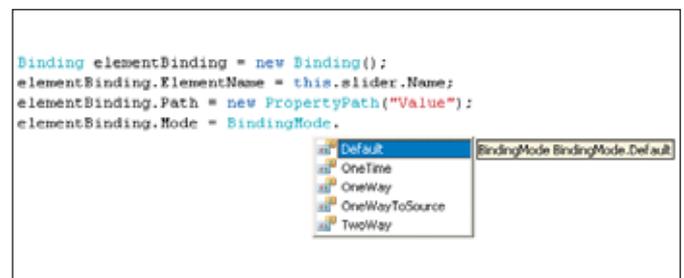
Comme vous le constatez, la réalisation d'une opération de Databinding, par le code, est quelque chose de réellement très simple à mettre en œuvre. Certes, cela nécessite plus de lignes de code que la version XAML, pas ou peu réservée aux développeurs purs et durs, mais le résultat est tout aussi identique.

Les différents modes de mise à jour

Lorsque l'on réalise une opération de binding, on a la possibilité d'indiquer quel sera le mode de mise à jour. Il existe 4 (sans compter la valeur par défaut) modes de mise à jour disponibles :

- **BindingMode.TwoWay** : Il s'agit du mode de binding par défaut. Utilisez ce mode pour indiquer que vous souhaitez mettre à jour l'interface ou la source quand l'un ou l'autre change. Dans le cas précédent, cela revient à indiquer que si la valeur du slider change, la valeur dans la TextBox change aussi, et vice-versa (si la valeur de la TextBox change alors la position/valeur du slider change aussi).
- **BindingMode.OneWay** : Permet d'indiquer que l'interface doit être mise à jour uniquement lorsque la source change. Concrètement, si la position du slider est modifiée, alors la valeur dans la TextBox sera modifiée également. Cependant, vous aurez beau saisir une valeur dans la zone de texte, cela ne modifiera pas la position du slider.
- **BindingMode.OneWayToSource** : Permet de notifier que la source doit être mise à jour lorsque l'interface change. C'est l'inverse de la propriété précédente : Si vous saisissez une valeur dans la zone de texte, alors la position du slider changera. Mais changer manuellement la position du slider ne modifiera pas la valeur inscrite dans la zone de texte !
- **BindingMode.OneTime** : Utilisé pour mettre à jour l'interface uniquement au démarrage de l'application ou lorsque le DataContext est modifié.

Pour utiliser l'un ou l'autre de ces modes de synchronisation, là encore, c'est extrêmement simple : il suffit de spécifier la propriété " Mode " de votre binding :



Les différents triggers de mise à jour

La propriété UpdateSourceTrigger est très utile dans le cas où vous souhaitez avoir un contrôle fin du binding : en effet, elle vous permet d'indiquer quand l'objet lié doit être mis à jour.

Comme dans le cas du mode de binding, cette propriété peut prendre 3 valeurs :

- **PropertyChanged** : Il s'agit de la valeur par défaut. Cette valeur permet d'indiquer au système que l'objet doit être mis à jour dès que la propriété est modifiée.
- **LostFocus** : Cette valeur sert à indiquer que l'objet doit être mis à jour lorsque le contrôle perd le focus.
- **Explicit** : Utiliser cette valeur pour avoir le contrôle total sur le moment de la mise à jour. En effet, grâce à cette valeur, votre objet ne sera mis à jour qu'après un appel à la méthode BindingExpression.UpdateSource !

```
Binding elementBinding = new Binding();
elementBinding.ElementName = this.slider.Name;
elementBinding.Path = new PropertyPath("Value");
elementBinding.Mode = BindingMode.Default;
elementBinding.UpdateSourceTrigger = UpdateSourceTrigger.
```



Dans le troisième cas, pour pouvoir faire l'appel à BindingExpression.UpdateSource, il faut d'abord récupérer la BindingExpression associée à la Dependency Property Text. Pour cela, il faut utiliser le code suivant :

```
this.tb.GetBindingExpression(TextBox.TextProperty).UpdateSource();
```

L'avantage de cette technique est évident: vous contrôlez le moment de la mise à jour de l'objet. Ainsi, vous pouvez très bien permettre à vos utilisateurs de modifier les propriétés d'un objet avec une interface graphique : le fait est que tant que vous n'aurez pas appelé cette méthode, BindingExpression.UpdateSource, les changements ne seront pas persistés sur votre objet !

Les Converters

Faire du DataBinding, c'est bien, mais souvent on se retrouve confronté à des situations de ce genre (fictif car, pour être honnête, cet exemple fonctionne tout de même) : on a un objet Product qui a des propriétés telles que le nom, le prix, etc. Lorsque l'on essaye de lier la propriété Price à une zone de saisie (TextBox), on reçoit une exception. Pourquoi ? Tout simplement car on ne peut lier une propriété de type double (le prix) à la propriété Text d'une zone de saisie.

Autre exemple : votre objet Product possède une propriété Image. Bien souvent, cette image est représentée par une chaîne de caractères. Or, lorsque vous faites votre binding pour lier cette propriété Image à un contrôle du type PictureBox ou Image, vous recevez une erreur car la source de ce type de contrôle doit être une image, pas une chaîne de caractères !

Pour régler ce genre de problèmes, il faut passer par des Converters. Ce sont des classes qui vont vous servir à convertir une valeur dans un autre type (typiquement, convertir un double en chaîne de caractères, convertir un chemin de fichier en une image, etc.).

Pour les utiliser, il vous faut une classe qui implémente l'interface IValueConverter. Cette interface met à votre disposition 2 méthodes :

Convert (pour convertir une valeur dans un autre type) et ConvertBack (méthode inverse).

```
public class DoubleToStringConverter : IValueConverter
{
    public object Convert(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)
    {
        object o = value as double?;
        if (o != null)
        {
            return String.Format(CultureInfo.InvariantCulture, "{0:###.###}", o);
        }
        return new StringConverter("Error: Invalid conversion");
    }
    public object ConvertBack(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)
    {
        return new StringConverter("Error: Invalid conversion");
    }
}
```

Pour utiliser ce convertisseur, rien de plus simple :

```
elementBinding.Converter = new DoubleToStringConverter();
```

Et si vous souhaitez faire la même chose en XAML (ce qui est tout à fait possible), il va vous falloir écrire un peu plus de code : on commence déjà par déclarer notre Converter dans les ressources de notre application :

```
<conv:DoubleToStringConverter x:Key="PriceConverter" />
```

Ensuite, on applique le binding à notre TextBox, en lui spécifiant cette instance du converter :

```
<Slider x:Name="slider" Minimum="0" Maximum="200" />
<TextBox x:Name="tb" Text="{Binding ElementName=slider, Path=Value, Converter={StaticResource PriceConverter}}" />
```

Et le tour est joué ! Comme vous pouvez le constater, la syntaxe est certes un peu plus lourde à écrire mais permet aux IDE comme Cider, l'éditeur d'interfaces WPF disponible avec Visual Studio, d'avoir un rendu en temps réel au sein de l'application (tout du moins à partir de la version Bêta).

Conclusion

Les opérations de binding sont une coutume employée par bon nombre de développeurs. En effet, la simplicité de mise en œuvre et l'efficacité de cette technique a suffisamment fait ses preuves pour que tout le monde l'utilise. Avec Windows Presentation Foundation, cette technique est devenue plus performante et plus simple à implémenter : une bonne raison de ne pas passer à côté lors de développements avec cette nouvelle technologie, qui ne cesse de faire parler d'elle !

Ressources

- Webcast sur le Databinding : <http://www.microsoft.com/france/vision/db/msdn/X00263/>
- Le blog de Beatriz Costa : <http://www.beacosta.com/>
- Le blog de l'équipe en charge de la rédaction du SDK de WPF : <http://blogs.msdn.com/wpf/sdk/archive/tags/Data+Binding/default.aspx>
- Data Binding sur la MSDN : <http://msdn2.microsoft.com/en-us/library/ms750612.aspx>

■ **Thomas LEBRUN**
 Consultant/Formateur - MVP C#
 Email : thomas.lebrun@winwise.com
 Blog: <http://blogs.developpeur.org/tom>

Silverlight : la révolution de l'interface graphique web



Microsoft se lance sur le marché de l'animation pour les sites web, dans le but de concurrencer la technologie Flash (et accessoirement Flex) d'Adobe. Silverlight est la dernière technologie de l'éditeur, disponible en version finale depuis début septembre.

Le pouvoir de Silverlight, connu anciennement sous le nom de Windows Presentation Foundation Everywhere (WPF/E), réside dans sa capacité de créer des interfaces web enrichies (RIA : application interactive riche, par exemple : Animation, 2D, 3D, interactivité, etc.) ainsi que dans la gestion du contenu multimédia. Et dans la prochaine version, d'intégrer les langages .Net et dynamiques (via un runtime dynamique, la DLR).

La fonctionnalité multi navigateur (IE, Firefox, Safari) et multi plate-forme (Windows, MacOS X, et Linux via le projet Moonlight récemment soutenu par Microsoft et développé par Novell sous la houlette de Miguel de Icaza), de Silverlight sont des avantages par rapport au WPF de .Net 3.0, centré sur le desktop. De plus, la gestion des droits par DRM (gestion des droits numériques) est incluse dans la version RC actuelle. Par exemple, on peut télécharger une vidéo vue sur Youtube grâce à des plug-in, chose qui ne sera plus possible avec Silverlight. Il répond à une attente : avoir un outil de gestion de contenu qui puisse tirer parti des outils dont on dispose déjà, permettant ainsi de disposer d'une interface graphique " évoluée ". En outre, Silverlight permet de toucher une grande partie des plates-formes avec des coûts de déploiement peu élevés.

Gestion du contenu Multimédia, qu'est ce que cela signifie concrètement ?

Silverlight est compatible avec les vidéos déjà présentes sur Internet en s'appuyant sur le format WMV (Window Media Vidéo). Bien sûr grâce à la gamme Expression, et à Expression Média / Media Encoder, en particulier (pour l'encodage et la conversion de vidéo), on pourra convertir les types de vidéo QuickTime ou encore AVI pour ensuite pouvoir les utiliser dans Silverlight. Grâce aux fichiers WMV intégrés dans Windows Media Server, il est possible d'avoir de la vidéo. La fonction streaming de silverlight permet de contrôler la vidéo sur le serveur à partir du client. Microsoft offre d'ailleurs gratuitement de l'espace de streaming à travers <http://silverlight.live.com/>

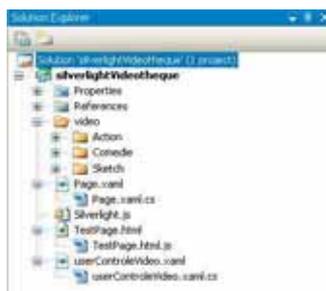
Silverlight accepte les types de format standard comme WMV, WMA, MP3 (attention : le format MPEG 4 n'est pas supporté, ndr). Grâce à Expression Media on peut créer des chapitres ou même ajouter son propre template de lecteur vidéo. Silverlight est donc pour le moment, le seul produit à pouvoir offrir une solution complète permettant d'exploiter toute la puissance de Windows Media ainsi que de donner une application véritablement interactive et riche.

Grâce à la plate-forme Silverlight, Microsoft vient de transformer les développeurs traditionnels en développeurs d'interface graphique riche, sans modifier la manière de travailler.

Création d'un UserControl dynamique

Pour comprendre comment ajouter un UserControl dynamique à notre page.xaml qui est notre page principale, nous utiliserons un exemple technique. Cet exercice va nous permettre de cerner la manière de travailler avec Silverlight. L'exercice consistera à faire " apparaître " sans streaming, différentes catégories de vidéos, en fonction de la catégorie choisie. Nous aurons 9 vidéos qui se lanceront sur l'écran avec un effet de rotation. Ces dernières seront en lecture de manière simultanée. La sélection d'une vidéo la fera apparaître automatiquement en plein écran

et retirera les autres de l'écran. Tout d'abord, nous allons ouvrir Visual Studio, sélectionner le framework 3.5 ainsi que le langage à utiliser, ici en l'occurrence C#. Ensuite nous allons choisir un type de projet : Silverlight Project. Nous constatons alors qu'il y a différents fichiers générés automatiquement : (voir ci-contre)



- Page.xaml : Le rendu de notre page HTML par défaut (On peut changer cela si on le désire).
 - Silverlight.js : Vérifie la présence d'un plug-in sur la machine cliente.
 - TestPage.html : Contient la page principale en l'occurrence Page.xaml.
- Afin de pouvoir créer un UserControl, nous allons ajouter au niveau de notre projet un élément de type SilverlightUserControl. Maintenant voyons ce qui se trouve à ce niveau-ci.

```
public class UserControl1 : Control
{
    public UserControl1()
    {
        System.IO.Stream s =
            this.GetType().Assembly.GetManifestResourceStream("silverlight
Videotheque.UserControl1.xaml");
        this.InitializeFromXaml(new System.IO.StreamReader(s).ReadTo
End());
    }
}
```

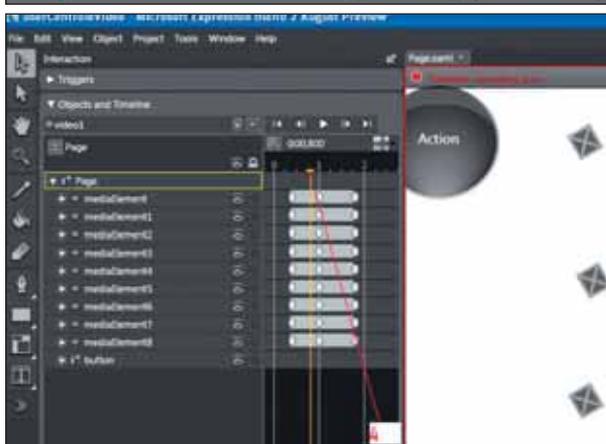
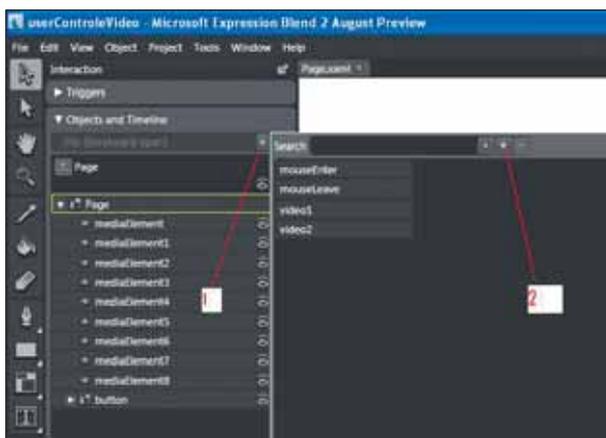
Nous constatons que notre UserControl hérite d'un contrôle qui se trouve au niveau du framework, pour pouvoir récupérer les différents éléments se trouvant au niveau du xaml. Le code ci-dessus qui a été généré, provoque une initialisation du xaml qui renvoie un FrameworkElement pour pouvoir gérer tous nos éléments.



Création de notre userControl

Pour réussir notre exercice, il faut compléter notre userControl avec des médias et des animations, nous allons donc utiliser Microsoft Expression Blend 2. Nous allons créer un bouton grâce au contrôle Ellipse et on mettra juste derrière ce contrôle 9 mediaElements. Nous pouvons commencer notre animation. Tout d'abord, créer un Storyboard, de cette manière nous avons un nouveau timeline qui se crée (il s'agit là de notre animation). En déplaçant l'axe de notre ligne du temps à l'endroit voulu, nous pouvons alors créer notre animation en déplaçant les différentes vidéos sur la surface de travail. Voilà : nous avons créé notre animation qui place nos 9 vidéos sur l'écran.

Pour créer l'animation qui retire les 9 vidéos de l'écran nous allons faire de même mais en sens inverse : donc création d'un timeline en positionnant les 9 vidéos à l'endroit où elles se sont mises en fin d'animation. Après avoir fait cela, nous déplaçons la ligne de temps plus loin et à cet instant nous remettons les 9 vidéos derrière notre bouton. Voilà comment créer des animations avec Blend. Maintenant, voyons ce qu'il a généré comme code xaml au niveau de notre userControl...



Avant cette explication, il faut savoir que nous n'avons pas mis tout le code qui a été généré dans le xaml de notre userControl (car le code est trop long). Les sources se trouvent sur le site du magazine.

Le code qui a été généré pour notre bouton, est un contrôle de type Ellipse (cercle) ou l'on a changé les propriétés des couleurs pour lui donner un effet de dégradé. Ensuite, nous avons rajouté par dessus notre Ellipse un contrôle de type textBlock afin de pouvoir lui donner un texte. Tous cela est mis au niveau d'un canvas appelé bouton grâce à la propriété x:Name="bouton" du canvas.

Mais qu'est ce qu'un canvas ?

Un canvas est tout simplement un panel où nous avons la possibilité de stocker des objets. Voyons maintenant la déclaration d'un des 9 mediaElements.

```
<MediaElement RenderTransformOrigin="0.5,0.5" x:Name="mediaElement" Width="28" Height="23" Canvas.Left="62" Canvas.Top="112" Source="300_Trailer.wmv" AutoPlay="False">
  <MediaElement.RenderTransform>
    <TransformGroup>
      <ScaleTransform ScaleX="1" ScaleY="1"/>
      <SkewTransform AngleX="0" AngleY="0"/>
      <RotateTransform Angle="0"/>
      <TranslateTransform X="0" Y="0"/>
    </TransformGroup>
  </MediaElement.RenderTransform>
</MediaElement>
```

Ici nous voyons que Blend crée notre mediaElement en lui donnant le nom "mediaElement" ainsi que la propriété autoPlay à False, pour que la vidéo ne puisse pas démarrer à l'apparition du contrôle. Il le positionne derrière le bouton, à l'emplacement Width et Height mais aussi au niveau du canvas principal de notre userControl, nous pouvons voir aussi qu'il nous place les propriétés de transformation :

- ScaleTransform : qui agrandit ou rétrécit notre objet sur un axe x et y
- SkewTransform : qui déforme notre objet sur un angle x ou y
- RotateTransform : qui fait pivoter notre contrôle par rapport à l'angle donné
- TranslateTransform : qui déplace notre contrôle sur un axe x ou y.

Sans oublier nos animations qui placent et retirent nos vidéos de l'écran, à savoir

```
<Storyboard x:Name="video1"> qui place les vidéos sur l'écran
<Storyboard x:Name="video2"> qui retire les vidéos de l'écran
```

Explication du code behind qui se trouve au niveau de notre userControl

```
FrameworkElement _root;
public userControlVideo(int top,String titre,List<String> liste)
{
    System.IO.Stream s =
        this.GetType().Assembly.GetManifestResourceStream("silverlightVideotheque.userControlVideo.xaml");
    _root=this.InitializeFromXaml(new System.IO.StreamReader(s).ReadToEnd());
}
```

Développement Web

Afin de pouvoir manipuler nos éléments qui se trouvent au niveau de notre code xaml qui a été généré par Blend, il faut déclarer une variable de type FrameworkElement comme montré ci-dessus.

Au niveau du constructeur

```
_monButton = _root.FindName("button") as Canvas;  
_monTexte = _root.FindName("texte") as TextBlock;  
_animationVideo1 = _root.FindName("video1") as Storyboard;  
_animationVideo2 = _root.FindName("video2") as Storyboard;
```

Ceci fait, nous pouvons commencer à récupérer nos éléments en lui indiquant `_root.FindName(String)` afin qu'il puisse faire une recherche au niveau de notre `frameworkElement` de l'élément demandé, sans oublier d'indiquer le type de ce dernier (as Type).

Donc pour notre projet :

- Button : variable `_monbutton` de type `canvas` déclaré en global.
- Text du bouton : variable `_monText` de type `TextBlock` déclaré en global.
- Animation qui place les vidéos sur l'écran : variable `_animationVideo1` de type `Storyboard` déclaré en global.
- Animation qui retire les vidéos de l'écran : variable `_animationVideo2` de type `Storyboard` déclaré en global.

```
private void initialisationDesVideo() {  
  
    String nameFind = "mediaElement";  
    listeVideo = new Dictionary<string, MediaElement>();  
  
    for (int i = 0; i < _video.Length; i++) {  
        if (i == 0) // test si i==0 pour le premier MediaElement  
        {  
            _video[i] = _root.FindName("mediaElement") as MediaElement;  
            _video[i].Source = new Uri(listeVideoName[i], UriKind.Relative  
OrAbsolute);  
            _video[i].MouseLeftButtonUp += new MouseEventHandler  
(_video1_MouseLeftButtonUp);  
            listeVideo.Add("mediaElement", _video[i]);  
        }  
        else {  
            nameFind = "mediaElement";  
            nameFind = nameFind + i;  
            _video[i] = _root.FindName(nameFind) as MediaElement;  
            _video[i].Source = new Uri(listeVideoName[i], UriKind.Relative  
OrAbsolute);  
            _video[i].MouseLeftButtonUp += new MouseEventHandler  
(_video1_MouseLeftButtonUp);  
            listeVideo.Add(nameFind, _video[i]);  
        }  
    }  
}
```

Afin de pouvoir gérer de manière optimale la récupération des 9 vidéos, l'évènement click et l'assignation de source des vidéos, nous allons créer une méthode `initialisationDesVideo()` qui sera appelée dans notre constructeur.

Dans cette méthode il y a un tableau de `MediaElement` qui a été créé en variable globale (`_video[9]`). On le parcourt en assignant chaque

`mediaElement` récupéré de notre `FrameworkElement` à chaque index de notre tableau de vidéo. Ensuite nous pouvons assigner les sources de vidéo qui sont passées en paramètre au niveau du constructeur dans une liste de type `String` (qui contient nos sources).

Ensuite, nous pouvons ajouter l'évènement `click` à toutes nos vidéos pour pouvoir les mettre en plein écran via la méthode `_video1_MouseLeftButtonUp`.

Enfin, nous pouvons les ajouter à notre (liste) `Dictionary` qui a comme clef un `String` (le nom de la vidéo) et comme objet la vidéo. Afin de pouvoir gérer toutes les vidéos en une seule fois, exemple : en fin d'animation (qui place toutes les vidéos sur l'écran) nous pourrions faire appel à une méthode qui traitera notre liste de vidéo, pour soit les démarrer ou les arrêter en fonction du paramètre de type booléen entrant.

Au niveau du constructeur

```
_monButton.MouseLeftButtonUp += new MouseEventHandler(_mon  
Button_MouseLeftButtonUp);  
_animationVideo1.Completed += new EventHandler(_animationVideo1  
_Completed);
```

Afin de lancer l'animation qui place les vidéos sur l'écran, nous devons assigner un évènement de type `click` à notre bouton pour pouvoir demander le lancement de l'animation. Pour gérer le démarrage des vidéos en fin d'animation (comme expliqué dans l'exemple ci-dessus) nous devons assigner un évènement en fin d'animation à notre animation (qui place les vidéos sur l'écran).

```
void _monButton_MouseLeftButtonUp(object sender, MouseEventArgs e)  
{  
    if (flag == false)  
    {  
        _animationVideo1.Begin();  
        flag = true;  
    }  
    else  
    {  
        playOrStopAllVideo(false);  
        _animationVideo2.Begin();  
        flag = false;  
    }  
}  
  
void _animationVideo1_Completed(object sender, EventArgs e)  
{  
    playOrStopAllVideo(true);  
}
```

Voici l'évènement `click` du bouton : pour commencer nous devons déclarer une variable de type booléen en global afin de pouvoir gérer si nous avons cliqué pour la 1re fois ou pour la 2e fois sur notre bouton. Si nous avons cliqué pour la 1re fois sur notre bouton, alors nous démarrons l'animation qui place les vidéos sur l'écran, sans oublier de mettre la variable de type booléen à `true` (disant que nous avons cliqué sur le bouton). En fin d'animation il y a l'évènement de fin d'animation qui s'enclenche. Cet évènement démarre ainsi toutes les vidéos en appelant la méthode `playOrStopAllVideo(true)` en lui passant le paramètre booléen



à True afin de lui demander de démarrer toutes les vidéos. Si nous cliquons pour la 2e fois sur notre bouton, nous demandons de stopper toutes les vidéos et de démarrer l'animation qui retire toutes les vidéos de l'écran sans oublier, bien entendu, de mettre notre variable booléenne à false.

```
void _video1_MouseLeftButtonUp(object sender, MouseEventArgs e)
{
    _videoTempForFullScreen=((MediaElement)sender);
    playOrStopAllVideo(false);

    if (BrowserHost.IsFullScreen == false)
    {
        _animationVideo2.Begin();
        BrowserHost.IsFullScreen = true;
        _videoTempForFullScreen.Play();
    }
}
```

Afin de pouvoir gérer le fullScreen en cliquant sur une des vidéos mise sur l'écran nous devons, comme montré ci-dessus, assigner l'évènement click sur toutes nos vidéos.

Voyons de plus près à quoi ressemble cet évènement. Tout d'abord, nous récupérons le mediaElement envoyé via le sender en l'affectant dans une variable globale videoTempForFullScreen de type mediaElement. Ensuite, nous testons si le BrowserHost est en fullScreen ou non. S'il ne l'est pas, alors nous pouvons demander de démarrer l'animation qui retire toutes les vidéos de l'écran (nous verrons par la suite ce qui se passe en fin d'animation), sans oublier bien entendu de demander le fullScreen en mettant BrowserHost.IsFullScreen à True, maintenant nous pouvons démarrer la vidéo qui est mise en fullScreen.

```
void _animationVideo2_Completed(object sender, EventArgs e)
{
    if (_videoTempForFullScreen != null)
    {
        _videoTempForFullScreen.SetValue(Canvas.TopProperty, topTemp);
        _videoTempForFullScreen.SetValue(Canvas.LeftProperty, 0);
        _videoTempForFullScreen.Height = BrowserHost.ActualHeight;
        _videoTempForFullScreen.Width = BrowserHost.ActualWidth;
    }
}
```

Voyons maintenant ce qui se passe en fin d'animation (qui retire les vidéos de l'écran), nous testons la variable temporaire de type mediaElement qui est déclarée en global si elle est à nul (afin de savoir sur quelle vidéo nous avons cliqué). Si ce n'est pas le cas, alors nous pouvons commencer la gestion du fullScreen par rapport à cette vidéo en la déplaçant tout en haut à gauche du browser sur la ligner zéro (grâce à la variable topTemp qui a été passée par le constructeur). Et en lui indiquant ces propriétés, Height et Width deviennent BrowserHost.ActualHeight et BrowserHost.ActualWidth pour pouvoir l'agrandir à la taille du browser qui a été mis en fullScreen au préalable.

Au niveau du constructeur

```
BrowserHost.FullScreenChange += new EventHandler(BrowserHost
```

```
_FullScreenChange);
```

```
void BrowserHost_FullScreenChange(object sender, EventArgs e)
{
    if (BrowserHost.IsFullScreen == false)
    {
        if (_videoTempForFullScreen != null)
        { //remet la video a sa place initiale ainsi que son format
            _videoTempForFullScreen.Height = 23;
            _videoTempForFullScreen.Width = 28;
            _videoTempForFullScreen.SetValue(Canvas.TopProperty, 112);
            _videoTempForFullScreen.SetValue(Canvas.LeftProperty, 62);
            _videoTempForFullScreen.Stop();
            _animationVideo1.Begin();
        }
        _videoTempForFullScreen = null;
    }
}
```

Maintenant voyons comment retirer le fullScreen du BrowserHost.

Avant de commencer nous devons assigner l'évènement du fullScreen comme déclaré ci-dessus dans le constructeur.

Une fois appuyé sur Escape, BrowserHost.IsFullScreen est mis directement à False. Donc on teste si BrowserHost.IsFullScreen est à false, alors nous redimensionnons la vidéo qui a été mise en fullScreen à sa taille initiale (qui se trouve dans le xaml). Ensuite, nous la replaçons derrière le bouton. Puis nous stoppons la vidéo et demandons de démarrer l'animation (qui met toutes les vidéos sur l'écran). Bien entendu, il ne faut pas oublier de mettre la variable temporaire de type mediaElement déclarée en global à nul afin de ne pas avoir de confusion avec d'autre instance de ce userControle. L'enclenchement de cet évènement se fait sur tous les userControle qui comportent cet évènement.

Comment intégrer ce userControle dynamiquement à notre Page.xaml

```
public void Page_Loaded(object o, EventArgs e)
{
    // Required to initialize variables
    InitializeComponent();

    maListe = new List<userControleVideo>();

    actionListe=new List<String>();
    actionListe.Add("video/Action/300_Trailer.wmv");
    actionListe.Add("video/Action/Copying_Beethoven_Trailer_
FANTASTIC____.wmv");
    actionListe.Add("video/Action/Disturbia_Trailer.wmv");
    actionListe.Add("video/Action/Fantastic_Four_Rise_of_the_
Silver_Surfer_Trailer_2.wmv");
    actionListe.Add("video/Action/harry_potter_en_the_order_of_
the_phoenix_trailer_.wmv");
    actionListe.Add("video/Action/Jindabyne_trailer.wmv");
    actionListe.Add("video/Action/New_Die_Hard_Trailer_.wmv");
    actionListe.Add("video/Action/Premonition_Trailer.wmv");
    actionListe.Add("video/Action/White_Noise_The_Light_
```



```
_Trailer.wmv");  
  
        userControleVideo us = new userControleVideo(-1,"Action",action  
Liste);  
        us.SetValue(Canvas.TopProperty, 1);  
        us.SetValue(Canvas.LeftProperty, 1);  
        Children.Add(us);  
        maListe.Add(us);  
        us.MouseLeftButtonUp += new MouseEventHandler(monButton_  
MouseLeftButtonUp);  
    }
```

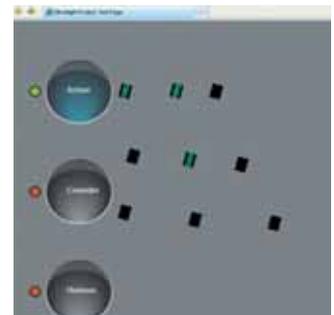
Il faut savoir qu'au niveau du constructeur de la Page.xaml nous n'avons mis qu'une instance de notre userControle alors que dans les sources il y en a trois. Nous allons commencer par créer une variable liste de type userControleVideo afin de pouvoir gérer les différents userControle (expliqué par la suite). Nous allons aussi déclarer une liste de type String qui comprendra toutes les sources des vidéos. Ensuite, on passe le paramètre int qui est affecté à notre variable topTemp au niveau du code Behind (C#) de notre userControle afin de pouvoir gérer le FullScreen. Donc nous passons -1, l'inverse du Canvas.TopProperty qui est l'emplacement Top du userControle. On indique aussi par la suite l'emplacement du Left de notre userControle par rapport au Canvas de notre Page.xaml. Enfin, nous passons en paramètre le titre de notre userControle, en l'occurrence " action ", ainsi que la liste de sources de nos vidéos. Maintenant nous pouvons l'ajouter à notre Page.xaml en indiquant Children.Add(userControle) sans oublier bien entendu de l'ajouter à notre liste de userControle déclarée en global afin de pouvoir gérer les différents userControle du même type.

Au niveau du constructeur

```
us.MouseLeftButtonUp += new MouseEventHandler(monButton_Mouse  
LeftButtonUp);  
us2.MouseLeftButtonUp += new MouseEventHandler(monButton_Mouse  
LeftButtonUp);  
us3.MouseLeftButtonUp += new MouseEventHandler(monButton_Mouse  
LeftButtonUp);
```

```
void monButton_MouseLeftButtonUp(object sender, MouseEventArgs e)  
{  
    tempUserControleVideo = ((userControleVideo)sender);  
  
    foreach (userControleVideo us in maListe)  
    {  
        if (us != tempUserControleVideo)  
        {  
            us.reinitialisationDuUserControle();  
        }  
    }  
    tempUserControleVideo = null;  
}
```

Voyons maintenant pourquoi avoir assigné un événement click sur chacune des instances de notre userControle. Nous l'avons assigné de telle manière que les vidéos d'un userControle (ex : action) sont mises sur l'écran. Lorsque nous cliquons sur le bouton d'un autre userControle (ex :



comédie) alors toutes les vidéos du userControle (action) se retirent de l'écran et laissent place aux vidéos du userControle (comédie) qui viennent se mettre sur l'écran. Afin de retirer les vidéos du userControle(action) une méthode Public a été mise au niveau du userControle qui réinitialise le

userControle à zéro, c'est-à-dire qui retire les vidéos de l'écran en les stoppant et en mettant la variable global booléen à False du userControle. En plein écran :



Conclusion

Avec cet exemple, nous avons un aperçu de la toute puissance de Silverlight, et la simplicité des UserControl. La manière de travailler et la logique de développement ne change pratiquement pas dans l'utilisation de Silverlight. Cette technologie annonce beaucoup de perspectives et d'idées pour le développement web.

Ressources

Afin que vous puissiez refaire cet exemple il vous faudra :

- Expression Blend
- Visual Studio 2008 bêta 2
- Plug in RC ou Alpha Refresh 1.1 pour le navigateur
- Tools Silverlight for VS2008 B2 permet d'intégrer un projet Silverlight dans Visual Studio.

La solution se trouve sur le site du magazine...

■ **Mohamed Benmostapha**
Consultant
en technologies .NET
mohamed.benmostapha@ict7.com

■ **Jonathan Vanderroost**
www.ict7.com
Consultant
en technologies .NET
jonathan.vanderroost@ict7.com

■ **Vincent Fievez**
Technical Director
vincent.fievez@ict7.com

Abonnez-vous

1 ÉCO

Recevez
le magazine
chaque mois

économisez **20 €**

11 Numéros

45 €

Au lieu de 65,45 €
(Prix au numéro)

(Prix France
métropolitaine)

-40%

2 Étudiant

Vous devez
justifier de votre
statut d'étudiant

Economisez **26 €**

11 Numéros

39 €

Au lieu de 65,45 €

(Prix au numéro)

(offre réservée
France
métropolitaine)

-30%

3 Numérique

Lisez
chaque mois
le magazine

Format PDF
(téléchargement)

11 Numéros

30 €

Tarif Monde entier

Inscription
en ligne uniquement

www.programmez.com



2,73€
le numéro

+ Abonnement INTÉGRAL

NOUVEAU

ACCÈS ILLIMITÉ aux ARCHIVES du MAGAZINE pour **1€** par mois !

Prix de lancement

Cette option est réservée aux abonnés pour 1 an au magazine, quel que soit le type d'abonnement (Éco, Numérique, Etudiant). Le prix de leur abonnement normal est majoré de 12 € (Prix de

lancement, identique pour toutes zones géographiques). Pendant la durée de leur abonnement, ils ont ainsi accès, en supplément, à tous les anciens numéros et articles /dossiers parus.

OUI, je m'abonne !

ou abonnement en ligne : www.programmez.com

ABONNEMENT 1 an ECO au prix de 45 € TTC. Tarif France métropolitaine.
Tarifs hors France métropolitaine : CEE et Suisse : 51,83 € - Algérie, Maroc, Tunisie : 55,95 € - Canada : 64,33 € - Tom : 79,61 € - Dom : 62,84 € - Autres : nous consulter

ABONNEMENT 1 an ETUDIANT (11 n°) : 39 € TTC. Offre limitée à la France métropolitaine. Photocopie de la carte d'étudiant obligatoire

+ SUPPLEMENT ABONNEMENT 1 an INTEGRAL au prix de lancement de 12 € TTC. (s'ajoute à une des formules d'abonnement)

MONTANT TOTAL DE L'ABONNEMENT : €

M. Mme Mlle Entreprise : Fonction :

Nom : Prénom :

Adresse :

Code postal : Ville :

Tél : E-mail :

Je joins mon règlement par chèque à l'ordre de Programmez ! Je souhaite régler à réception de facture

A remplir et retourner sous enveloppe affranchie à :
Programmez ! - Service Abonnements - 22 rue René Boulanger - 75010 Paris.
abonnements.programmez@groupe-gli.com

Programmez!
LE MAGAZINE DU DÉVELOPPEMENT

Offre limitée,
valable jusqu'au
31 décembre 2007

Le renvoi du présent bulletin implique pour le souscripteur l'acceptation pleine et entière de toutes les conditions de vente de cette offre.

Conformément à la loi Informatique et Libertés du 05/01/78, vous disposez d'un droit d'accès et de rectification aux données vous concernant.

Par notre intermédiaire, vous pouvez être amené à recevoir des propositions d'autres sociétés ou associations.

Si vous ne le souhaitez pas, il vous suffit de nous écrire en nous précisant toutes vos coordonnées.

ALLEZ PLUS LOIN DANS VOS DÉVELOPPEMENTS,
PARTEZ EN QUÊTE D'APPLICATIONS RICHES.

Votre potentiel, notre passion.™

Microsoft®

Votre défi : Créer des applis riches et dynamiques pour PC ou mobiles.

Vos armes : Utilisez Visual Studio et Windows Vista pour créer de la valeur et pas seulement des données. Plus d'informations sur releveztouslesdefis.com

