Hors-Série .Net

Programmez! HS

Programmez!

Avril / Mai 2008 - Hors-Série

www.programmez.com

LE MAGAZINE DU DÉVELOPPEMENT

Windows Live

Découvrez les API Live

Maîtrisez Virtual Earth, Live ID et Live Photo !

Web 2

Développez avec Silverlight 2.0 bêta !

Futur
F# : le prochain langage .Net ?

Multi-cœur
Optimisez vos codes avec Parallel FX

Architecture
Intéropérabilité .Net - Java 6 avec Workflow Foundation

Vista
Service Pack 1 et le développeur

.Net 3.5
Créez des documents OpenXML
Tout savoir ou presque sur LINQ

Serveur
Utiliser les nouvelles API de Windows Server 2008

Printed in France - Imprimé en France - BELGIQUE 5,45 € - SUISSE 10 FS - LUXEMBOURG 5,45 € - Canada 7,45 $ CAN - DOM Surf 5,90 € - TOM 780 XPF - MAROC 50 DH

Extrême Java

Seam

UML

Hibernate

valtech training

Gestion de projet

Scrum

XML

.Net

Au plus court vers vos nouvelles compétences

90 formations au développement logiciel

chez vous
ou à Paris, Toulouse, Lyon, Grenoble, Genève, Bruxelles, Luxembourg

Architecture et intégration

- Introduction au logiciel libre (1 jour)
- La persistance dans les applications Java (1 jour)
- Ingénierie logicielle objet (3 jours)
- Introduction aux architectures et technologies du Web (1 jour)
- Architectures .Net multi-niveaux (3 jours)
- Intégration d'applications (EAI, B2B) : les technologies et le projet (3 jours)
- Urbanisation du système d'information (2 jours)
- Architecture orientée service (3 jours)
- Architecture d'entreprise avec Java EE (4 jours)
- Du Mainframe au serveur d'applications (1 jour)

Développement Java et C++

- Introduction technique à Java (1 jour)
- Programmer en utilisant les aspects et les Design Patterns (3 jours)
- Java et la conception objet (5 jours)
- Développement d'un client riche avec SWT et Eclipse RCP (3 jours)
- Atelier Java avancé (5 jours)
- Eclipse, créer son environnement de développement intégré (2 jours)
- Programmation intensive avec Java (5 jours)
- Extrême Java (4 jours)
- Développer une application Corba (4 jours)
- L'essentiel de C++ et la conception objet (5 jours)
- Programmation efficace et avancée en C++ (5 jours)

Microsoft .Net

- C# et la conception objet (5 jours)
- Programmation avec Visual Basic .Net et conception objet (5 jours)
- Programmation intensive avec le Framework .Net (5 jours)
- Développement d'applications Web avec ASP.NET (5 jours)
- Développement d'applications Windows Forms sur la plate-forme .Net (5 jours)
- Développer des applications avec C# et le Framework .Net 3.0 (5 jours)

Gestion de projet

- Gérer des projets avec un processus itératif (4 jours)
- Les méthodes agiles de développement logiciel (1 jour)
- Le Processus Unifié de développement logiciel (2 jours)
- Du recueil des besoins aux exigences : rédiger le cahier des charges (2 jours)
- Gestion de projet (3 jours)
- Manager des hommes dans le cadre d'un projet (2 jours)
- Management de projet (5 jours)
- MSProject (3 jours)
- Gérer les projets agiles avec Scrum (2 jours)
- Gérer les projets agiles avec XP (2 jours)

Frameworks Java EE

- Concevoir et développer des EJB 2 (5 jours)
- Développer une application Java EE avec les EJB 3 (5 jours)
- Gestion de la persistance avec Hibernate (3 jours)
- Mise en œuvre du Framework Seam (3 jours)
- Développement avec le Framework Spring (3 jours)
- Gestion avancée de la persistance avec Hibernate (2 jours)

Analyse, conception et modélisation avec UML

- Introduction technique à l'analyse, la conception et la programmation objet (1 jour)
- Introduction à UML (1 jour)
- Concevoir avec les Design Patterns (5 jours)
- La modélisation métier avec UML (3 jours)
- Analyse et conception avec UML (5 jours)
- La modélisation des systèmes complexes avec UML 2 et SysML (3 jours)
- La modélisation efficace des exigences avec les cas d'utilisation (2 jours)
- Analyse orientée objet avec UML (2 jours)
- D'UML 1 à UML 2 : quoi de neuf, docteur ? (1 jour)
- Modéliser les besoins et analyser avec UML (4 jours)

Oracle

- Introduction technique (1 jour)
- Exploitation (4 jours)
- SQL (3 jours)
- PL / SQL (3 jours)
- Optimisation des requêtes (2 jours)
- Administration (5 jours)
- Tuning (3 jours)

XML et Web Services

- Introduction à la technologie XML (1 jour)
- Introduction aux technologies Web Services (1 jour)
- Développer avec XML (3 jours)
- Développer une application XSL (2 jours)
- Développer des applications Web Services en Java (3 jours)
- Développer des applications XML avec Java (2 jours)

Développement Web

- Développement de pages Web avec HTML, CSS et JavaScript (3 jours)
- Développement, déploiement et administration d'applications Web (Java EE) avec WebSphere (3 jours)
- Développement d'applications Web avec PHP (3 jours)
- Ajax, pour dynamiser le poste client (2 jours)
- Hacking des applications Web (1 jour)
- Développer des applications avec Adobe Flex 3 (5 jours)
- Conception d'applications Web d'entreprise avec Java EE, les Servlets, JSP et Struts (5 jours)
- Développement d'applications Web avec Ruby on Rails (3 jours)
- Développement d'applications Web avec JavaServer Faces (3 jours)
- JavaServer Faces avancé (2 jours)
- Utilisation du Framework Struts pour le développement d'applications Web (3 jours)
- Développer une application Web avec Ajax et le Google Web Toolkit (3 jours)
- Développer des applications pour Adobe Integrated Runtime (2 jours)

Stratégies de développement logiciel

- Test Driven Requirement ou la gestion des exigences dirigée par les tests (2 jours)
- Test Driven Development ou la programmation pilotée par les tests en Java (3 jours)
- Stratégie de test, vérification et validation (3 jours)
- Les fondements de l'IT Infrastructure Library (ITIL) (3 jours)
- Introduction au CMMI (3 jours)
- L'usine logicielle, des concepts à la pratique (3 jours)

www.valtech-training.fr
+33 (0)1 41 88 23 00 - +33 (0)5 62 47 52 00
info@valtech-training.fr

SOMMAIRE HORS-SÉRIE

> Actus
L'actualité en bref 6

> Evénements
Internet Explorer 8 : dès maintenant... ou presque 8

> Gros Plan
Windows Vista SP1 et les développeurs 9

> Serveur
Les nouvelles API de Windows Server :
API Application Recover et Restart 12

> Dossier : Windows Live
Introduction à Windows Live 14
Les API et services Live disponibles 16
Virtual Earth, une API surpuissante 17
Windows Live ID : demander le passeport 19
Tout savoir sur Windows Live Photo API 21

> Développement Web
Silverlight 2 bêta 1 : les applications RIA sont en marche 24

> Workflow
Médiations de protocoles REST/SOAP avec Workflow Foundation 29

> LINQ
Requêter avec LINQ : tout savoir ou presque 35

> Bureautique
Créez vos documents Open XML AVEC .Net et Linq 41

> F#
F#, un langage fonctionnel pour la plate-forme.Net 43

> Technique
Parallel FX : la parallélisation facile sous .Net ? 47

Édito

Soyez sur un nuage

2008 est décidément l'année " plus " de Microsoft, pour les développeurs. Le Mix08 à Las Vegas en mars dernier a permis de toucher les futures versions et évolutions de Silverlight, de la gamme Expression, Ces nouveautés ont été présentées lors de Remix08 à Paris.

Depuis quelques mois, chacun parle de " nuage informatique " ou cloud computing. Terme bizarre derrière lequel se cache l'utilisation des ressources du Web pour constituer des infrastructures externalisées et ne plus hébberger d'applications ou de services chez soi. Il s'agit ni plus ni moins de la notion de Software + Services ou de Saas. On a le choix du terme, mais il a le même sens. Depuis des semaines, Microsoft étoffe son offre sur le " nuage du web " avec Online Services. On dispose, en version bêta (et en France ce sera en 2009 seulement), d'une version d'Exchange, de Sharepoint et d'Office Live Meeting en ligne sous forme de services. À terme, il s'agit d'offrir les mêmes fonctionnalités dans notre nuage web que sur le serveur. Mais, fini l'installation, l'administration lourde, tout se passe en ligne ! Bref, c'est le retour de l'informatique à la demande, mais cette fois-ci, c'est du concret. Promis, juré, si je mens, je ne fais plus de .Net !

Et on ne s'arrête pas là ! SQL Server subit le même sort avec SQL Server Data Services. Aujourd'hui, les fonctionnalités disponibles sont réduites mais à terme, l'éditeur annonce que tout SQL Server sera dans le nuage sous forme de Software + Services ! Demain, on peut imaginer la même chose avec Visual Studio, MS Office, etc. On dispose déjà de Popfly qui montre comment un outil de développement peut fonctionner uniquement en ligne... Clairement, l'avenir, ou un des avenirs du développement, sera de développer dans et pour le nuage informatique. À vous de vous y mettre rapidement. Car le tournant est bien réel.

Cependant, nous sommes encore dans une approche bêta et bien peu de services (chez Microsoft et les concurrents) sont en version finale. Il faudra de longs mois pour finaliser tout cela. Et surtout, il faudra préciser comment interagir avec les différents services : Office Live, Windows Live, Online Services, les services tiers. Le rêve serait de pouvoir agréger tous ces services, d'utiliser tel service avec tel autre. Là, on serait réellement dans le Software + Services.

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

Programmez!

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 au Hors Série .Net : 20 € (5 numéros, tarif France métropolitaine) - PROGRAMMEZ! 1 an - 11 numéros France métropolitaine : 45 € - Etudiant : 39 € - Consultez les autres tarifs et abonnez-vous sur www.programmez.com

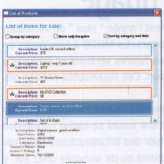
Rédaction : redaction@programmez.com
Directeur de la Rédaction : Jean Kaminsky - **Rédacteur en Chef :** François Tonic - **Ont collaboré :** F. Queudret, Y. Alet, A. Leroy, N. Bonnaert, L. Bar, A. Griffard, E. Vernié, S. Sfartz, I. R. Nzouetom, J. Chable, S. Leroux, F. Mazue.
Photo couverture : Microsoft - **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

Prochain numéro : 14 juin 2008

- **VB 9 :** *les meilleurs trucs et astuces pour maîtriser votre langage préféré*
- **Silverlight :** *déployer, développer avec Silverlight !*

LANGAGE

XAML pour Java : eFace enfonce le clou !



Depuis quelques semaines, l'éditeur Soyatec propose en téléchargement la nouvelle version de eFace, la v1.0.8. eFace est une solution pour pouvoir utiliser du code XAML dans des applications/environnements Java, Ajax, Swing, SWT. Il s'agit avant tout d'une plate-forme indépendante mais compatible avec XAML. Elle repose sur UPF : Universal Presentation Framework, une solution Java à la WPF, et s'utilise aussi bien avec NetBeans qu'Eclipse. Une solution qui mérite d'être connue !

Site : http://www.soyatec.com

STANDARD

XAML s'ouvre lui aussi !

Décidément, Microsoft se voit en éditeur interopérable. La dernière initiative en date concerne XAML présent dans WPF (.Net 3.x), Expression et Silverlight. XAML permet de décrire des interfaces utilisateurs. Aujourd'hui, ce sont 600 pages de documentation que les développeurs pourront consulter. Disponible gratuitement, sous format Microsoft Open Specification Promise, le développeur n'a pas à payer de frais et de licence pour accéder et mettre en œuvre XAML. Cette ouverture documentaire permettra, comme l'espère l'éditeur, d'étendre la présence de XAML qui jusqu'à présent reste très Windows.

Lien : http://go.microsoft.com/fwlink/?LinkId=113699

SYSTÈME

Singularity se dévoile un peu

Depuis quelques jours, Microsoft Research a mis sur la forge CodePlex un premier élément lié au projet Singularity, un projet de système d'exploitation radicalement différent de Windows qui pourrait être un des avenirs possibles de Microsoft. Cette brique est le Research Development Kit (ou RDK). On dispose de la documentation, des sources et de toute information sur comment compiler la chose ! Bien entendu, c'est un outil de virtualisation incontournable pour éviter de crasher son système de travail. On dispose d'un noyau managé (donc en .Net). Tout ceci reste expérimental, mais on peut penser que des morceaux de Singularity se retrouveront dans un futur Windows. Le RDK permet de se faire une idée des évolutions proposées par les laboratoires Microsoft sur les outils et langages. On y retrouve aussi la version actuelle de Singularity !

Site : http://www.codeplex.com/singularity

Agenda

Wygday 2008 : le 22 mai à Lille - www.wygday.wygwam.com
Wygwam, spécialisé sur les technologies .Net, organise une journée .Net pour les développeurs, DSI, chef de projets, architectes. De nombreuses sessions techniques sont prévues. Occasion aussi pour discuter, faire le point sur les dernières technologies.

IDE

Un peu d'Eclipse pour Microsoft

C'est Sam Ramji qui l'a dit, le 19 mars dernier, durant sa keynote à la grande conférence EclipseCon ! Sam Ramji est le grand gourou de la stratégie plate-forme Windows de Microsoft et notamment vers le monde open source. Sans être une réelle surprise, la session de Sam a permis de redire tout l'intérêt de Microsoft pour l'open source. L'interopérabilité fut de nouveau mise en avant. D'autre part, il a souhaité que Firefox soit encore plus optimisé et notamment sur le support de Media Player sous Vista. On apprenait aussi le bon fonctionnement d'Apache sur Windows Server. L'occasion aussi pour reparler des accords avec le monde PHP, Samba ou encore de l'intégration MySQL dans Visual Studio (via un plug-in). Concernant plus spécifiquement Eclipse, il a été question de collaboration entre Higgin et Cardspace sur la gestion et la fédération des identités et de leur sécurité. Le gros morceau fut SWT for WPF. Eclipse et Microsoft travaillent ensemble pour fournir un support de SWT pour la librairie d'interface et de présentation de .Net 3.x. Une première version a été introduite avec Eclipse 3.3, mais cela est jusqu'à présent assez limité et peu connu. Cependant, un gros travail reste à faire. SWT fonctionne actuellement sans trop de souci sur les couches natives Windows, mais pour faire le portage sur du WPF, c'est une autre histoire ! Notons que Sam n'a pas manqué d'humour en présentant une fiction : Supernova. La fusion de Visual Studio et d'Eclipse... Si pour le moment, Microsoft ne participe ni à la fondation, ni comme contributeur ou " lead " d'un projet, l'hypothèse n'a pas été écartée, tout comme celle d'un éventuel IDE C# sur Eclipse. Mais là, il faudra sans doute attendre...

DONNÉES

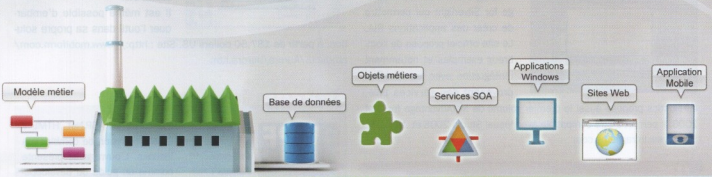
Microsoft dévoile SQL Server Data Services

Pour permettre un accès aux données n'importe où, n'importe quand, Microsoft a dévoilé SQL Server Data Services. SSDS vise l'informatique à la demande, chère à IBM avec haute disponibilité incluant un stockage de données à la demande et des requêtes via des web services. On paie ainsi le stockage réellement utilisé ainsi que les ressources utiles. La partie développement n'est pas oubliée avec des interfaces de programmation de type web services, Linq pour les requêtes, utilisation du *modèle Entity*.

C'est un changement important dans l'approche de la base de données qui confirme la volonté de Microsoft de proposer des services sur le principe du S+S, ou SaaS, selon le terme utilisé. Reste maintenant à intégrer l'ensemble des services et à trouver une cohérence à tout cela ! On peut le voir comme une extension de SQL Server.

La première fabrique logicielle entièrement pilotée par les modèles

CodeFluent est un produit de génie logiciel qui permet d'industrialiser la fabrication d'applications professionnelles manipulant des données sur la plate-forme .NET en automatisant la création des composants à partir d'une modélisation de votre métier.



CODEFLUENT PROCURE LES BÉNÉFICES SUIVANTS

Un gain très significatif de productivité par l'automatisation des tâches répétitives et la mise en œuvre du modèle métier selon une architecture orientée services.

La limitation du risque par la structuration du travail des développeurs autour d'une modélisation objet évolutive qui garantit une mise en œuvre selon les meilleures pratiques d'implémentation SOA sous plate-forme Microsoft.

Une maintenabilité et une qualité accrues grâce à l'approche intrinsèque de génération qui évite les erreurs et permet d'effectuer des mises à jour sur toutes les couches à l'aide d'une modification centralisée.

L'évolutivité de l'application car la prise en charge de nouvelles technologies et de nouvelles versions de plate-forme est assurée par la mise à disposition régulière de nouveaux producteurs (Windows Presentation Foundation, Office Business Applications, mobilité).

Les clients de **CodeFluent** gagnent sur leur marché

ENTREPRISES

118 218 LE NUMÉRO

118218 a choisi CodeFluent pour le développement accéléré et pérenne d'une application spécifique à son métier. 118218 dispose ainsi d'une application métier de type Web 2.0 tirant à la fois partie des technologies ASP.NET et Ajax.

ÉDITEURS DE LOGICIELS

VCS TIMELESS
Alternative retail software

VCS Timeless est un éditeur d'envergure internationale dans le secteur de la distribution spécialisée. La société a retenu CodeFluent pour le développement de son progiciel de gestion dédié à son secteur. Colombus.Next repose sur la technologie .NET 3.0 incluant WPF et WCF. La roadmap comprend le développement de nombreux modules sur plusieurs années.

SSII

SOGETI

Sogeti Application Consulting Services a sélectionné CodeFluent comme outil privilégié pour la réalisation d'applications spécifiques sous .NET. Sogeti propose à ses clients d'utiliser la fabrique logicielle CodeFluent pour la réalisation de leurs développements afin de garantir productivité, limitation du risque et maintenabilité.

DÉVELOPPEURS

CodeFluent EXPERT NETWORK

Richard Clark, 1er MVP .NET reconnu par Microsoft sur le marché français, a acquis la licence CodeFluent pour optimiser ses développements. Il est entré par ailleurs dans le réseau des experts CodeFluent, un label créé par SoftFluent pour reconnaître l'implication des développeurs dans la promotion et l'évolution du produit.

COMPOSANT

Infragistics se prépare à Silverlight 2.0

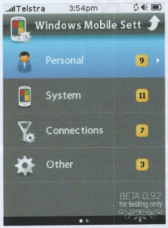
L'éditeur de NetAdvantage se prépare activement à la vague Silverlight 2.0 avec NetAdvantage for Silverlight qui permettra de créer des applications RIA. Le site officiel propose de nombreux exemples et notamment l'intégration avec les données provenant des services Live. Infragistics propose aussi le premier volume de NetAdvantage for .Net 2008. Il permet d'être compatible avec Visual Studio 2008 et .Net 3.5. Site : www.infragistics.com

IDE

MonoDevelop 1.0 : disponible !

La version finale de MonoDevelop est maintenant disponible après plusieurs années de développement. Cet IDE est avant tout destiné aux développeurs ASP.Net et Gnome sous Linux mais l'outil est bien entendu disponible sous Windows, MacOS X, etc. Il propose de nombreuses fonctionnalités : contrôle de code, intégration avec Unix toolchains, support Nunit, localisation, codage en différents langages, complétion (certains langages), refactoring (C#), designer graphique (C#). Il accepte les add-in tiers.

INTERFACE



PointUI rajeunit l'interface de Windows Mobile

Voilà une librairie d'interface pour Windows Mobile qui va faire du bruit ! Des développeurs indépendants mettent au point une interface qui n'est pas sans rappeler des conceptions de l'iPhone d'Apple. PointUI permet de contrôler et naviguer avec le doigt. À vous de tester ! Site : http://pointui.com/

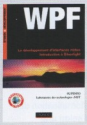
IDE

Aurora XAML Designer : tout pour XAML !

Mobiform crée l'éditeur Aurora XAML Designer qui permet de concevoir des interfaces XAML pour les applications desktop et web. Il est compatible avec .Net 3.5. Il gère les calques multiples, les graphiques 2D. Il repose sur une architecture extensible, accepte les contrôles WPF tiers, l'export vers divers formats graphiques, supporte le data binding, le zoom, etc.

Il est même possible d'embarquer l'outil dans sa propre solution. À partir de 187,50 dollars US. Site : http://www.mobiform.com/products/Aurora/aurora.htm

Le coin lecture



WPF

Difficulté : **
Editeur : Dunod
Auteur : collectif
Prix : 24,90 €

Enfin ! un bon ouvrage, en français, sur WPF, la nouvelle couche de présentation de .Net. Ce n'est pas un guide de référence mais une grande introduction à WPF : son architecture, son fonctionnement, sa mise en œuvre, ce qui est déjà bien. Avant tout didactique. Les auteurs (provenant du labo .Net de Supinfo) proposent une approche claire, illustrée, sans chercher à être exhaustive dans le code. On retrouve l'ensemble des éléments composants WPF : les contrôles, la 2D / 3D, XBAP, le layout, les templates, la gestion des ressources, les styles, la personnalisation des contrôles. On termine par un exemple complet d'application.

Expression Web

Difficulté : **
Editeur : Eni éditions
Auteur : Louise Villeneuve
Prix : 21,19 €

Vous ne connaissez pas encore Expression Web, le nouvel IDE Microsoft pour le développement web en ASP.Net, HTML ? C'est l'occasion de combler vos lacunes. L'ouvrage se présente comme un guide du développement dans lequel l'auteur revient sur les fonctions, l'environnement, les modules de l'outil. Très visuel, il devrait vous aider à rapidement mettre en œuvre ces fonctions dans vos projets. Une bonne introduction.

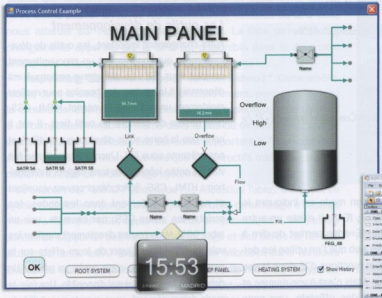
Windows Powershell

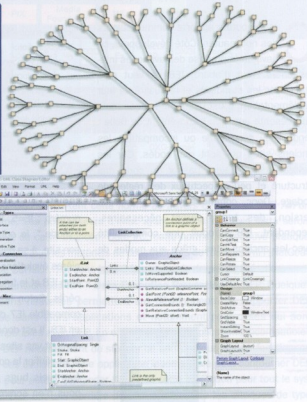
Difficulté : **
Editeur : Pearson
Auteur : collectif
Prix : 39 €

Powershell fait désormais partie de la panoplie des utilisateurs avancés et administrateurs de Windows. Ce puissant shell en ligne de commande permet de piloter, manipuler l'ensemble du système, plus finement qu'avec les interfaces graphiques.
Et la version 2.0 promet encore plus de possibilités, notamment avec une création de ses propres commandes, plus simple. Assez complet, le livre aborde les différents aspects de Powershell avec de nombreux codes et exemples, l'intégration avec .net, la sécurité, les objets COM, etc.

NOUVEAU

ILOG Diagrammer for .NET





- Diagrammes et tableaux de bord pour tous les types d'applications graphiques
- Algorithmes d'arrangement automatique de graphiques et de diagrammes
- Éditeurs graphiques prêts à être intégrés
- Intégration totale avec Microsoft Visual Studio
- Création d'applications riches ou Web avec Ajax
- Tableaux de bord avec animation en temps réel

Microsoft CERTIFIED Partner | ISV Software Solutions

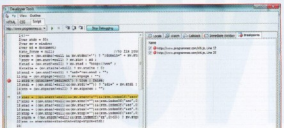
Optimized for Microsoft Visual Studio .net

http://diagrammerfornet.ilog.com

ILOG

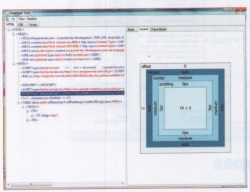
Changing the rules of business™

Web
Internet Explorer 8 : dès maintenant... ou presque



Annoncé en bêta pour début 2008, c'est finalement en mars dernier que Internet Explorer 8 est sorti en pré-version durant la conférence Mix08 de Las Vegas.

Cette mouture apporte comme toujours des nouveautés. Côté développeur, ce navigateur propose des avancées intéressantes. Tout d'abord, IE8 se veut plus strict avec les standards et la qualité de leur implémentation. C'est une bonne chose mais qui présente aussi un inconvénient de taille : rendre peu compatible ou incompatible les anciens sites qui sont mal codés. Car les navigateurs laissaient passer du code mal structuré et peu respectueux des règles de codage. Donc, comme avec toute nouvelle version, des tests seront nécessaires. Cependant, IE8 introduit une compatibilité avec les anciens sites, grâce aux modes de



rendu. Dans la b1, on dispose de trois rendus : **Quirks** (compatible avec les sites IE5), **standards IE7** et **standards IE8**. Par défaut, le mode IE8 est activé. Actuellement, pour passer d'un mode à un autre, il faut indiquer dans la page web le mode de rendu adéquat, par exemple :

```
<meta http-equiv="X-UA-Compatible" content="IE=7">
```

Dans ce cas, on demande à IE8 de passer en mode IE8. Mais on peut aussi passer par le serveur web, par l'intermédiaire du web.config de son IIS, selon la documentation officielle cela donne :

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <system.webServer>
    <httpProtocol>
      <customHeaders>
        <clear />
        <add name="X-UA-Compatible" value="IE=7" />
      </customHeaders>
    </httpProtocol>
  </system.webServer>
</configuration>
```

Il suffit de choisir son mode en indiquant le numéro : IE=5, IE=7, IE=8. Il existe un autre " mode ", le *edge*. IE=edge permet de dire à son site, à sa page web que l'on utilise les dernières évolutions et fonctions d'IE dans la version IE8 et le futur. Mais c'est à vos risques et périls. Selon les guides officiels, une page compatible IE7 doit passer sans modification en IE8. Il est donc conseillé de tester très rapidement vos sites sous IE8 bêta afin de garantir et vérifier la compatibilité et de détecter les éventuels bugs à fixer. Cependant, comme nous n'en sommes qu'à la première bêta, il faudra répéter les vérifications à chaque version. Vous pouvez aussi indiquer à chaque page du site d'être en mode IE=7 uniquement. La vérification sera importante pour les CSS, IE8 étant CSS 2.1.

HTML 5 et Ajax

Un important travail a été effectué sur Ajax et les premières implémentations de briques HTML5. Par exemple, dans une zone Ajax, la gestion du bouton " Back / Précédent " est assurée. On peut ainsi revenir à la situation précédente, par exemple sur une carte. Cette gestion fait partie de HTML5. Bien que tout HTML5 ne soit pas présent, on peut d'ores et déjà commencer à coder avec. Mais il est encore un peu trop tôt pour se faire une idée précise du support final.

Les outils de développement

Autre changement important, les outils de développement qui connaissent un renouvellement complet sous IE8 ! Le débuggeur embarqué est désormais à la hauteur des besoins pour repérer rapidement un problème Javascript, HTML, CSS, plus besoin de prendre un outil tiers. Il est à noter que la barre outils développeur de IE n'est pas présente sous IE8. L'accès aux outils se fait via une petite icône, avec trois onglets à disposition : HTML, CSS, Script. Vous pouvez visualiser et naviguer rapidement dans les codes, les scripts, les styles CSS, par exemple, voir un arbre DOM. Vous pouvez directement modifier les valeurs éditables et juger de leurs effets sur la page web. Et même avoir accès au offset. Un debuggeur Jscript est aussi disponible. Une fois le mode debug démarré, vous avez la possibilité de mettre des points d'arrêt et de suivre pas à pas. Le développeur web sera sans aucun doute ravi !

WebSlices : une des nouveautés

Le développeur marque une partie, une section de sa page web en l'indiquant comme webSlice. Ainsi, l'internaute accèdera à un panneau d'information contenant des données. Quand on souscrit à un webSlice, une nouvelle icône apparaît dans la barre d'outils près des favoris. Dans ce panneau s'affichent des images, du texte, des liens comme la météo. Le format webSlice décrit la souscription (selon la portion de la page web définie par le développeur). La déclaration se fait par une classe hslice. On peut aussi lui donner un temps défini d'activité, passé ce temps, le webSlice n'est plus disponible. Voici quelques-unes des nouveautés de IE8. Nous reviendrons prochainement plus en détail sur ce nouveau navigateur.

Pour en savoir plus :

http://code.msdn.microsoft.com/ie8whitepapers

François Tonic

Windows Vista SP1 et les développeurs

Avec la disponibilité successive de Windows Server 2008 et du Service Pack 1 de Vista, ce sont des améliorations de la sécurité, de la performance et l'ouverture à de nouveaux périphériques et standards qui sont d'ores et déjà disponibles. Le SP 1 de Vista bénéficie des développements réalisés pour Windows Server 2008 et apporte son lot d'améliorations mais aussi quelques nouveautés pour les développeurs. Visite guidée.

Le Service Pack 1 regroupe tous les correctifs de sécurité parus depuis la sortie officielle de Vista. En plus des évolutions, cela représente plusieurs centaines d'améliorations rendues disponibles aux utilisateurs qui avaient depuis longtemps exprimé leur frustration sur la performance globale du système. Dans cette partie, nous allons passer en revue les améliorations notables du SP1 sans nous attarder sur l'ensemble des correctifs. La liste complète des changements apportés par le SP1 est disponible dans le document officiel des changements notables (voir références).

• **" Hotpatching "**, autrement appelé " correctif à chaud ". Cette amélioration, dans la catégorie installation et déploiement, permet de réduire le nombre de redémarrages système lors de l'application d'un correctif à un composant Windows, même dans le cas où ce dernier est chargé et utilisé par un processus. Les correctifs supportant le " Hotpatching " sont déployés de la même manière que les correctifs traditionnels et ne déclenchent par de redémarrage système.

• Le support de **" exFAT "** (Extended File Allocation Table). exFAT est un système de fichiers conçu pour les périphériques de stockage à base de mémoire flash (clé USB, appareils photo, téléphones, PDA, etc.) et qui permet de supporter des volumes de stockage et des tailles de fichiers plus importants. exFAT avait été introduit par Microsoft dans Windows Embedded CE 6.0 pour contourner la faiblesse de NTFS sur ce type de périphérique étant donné l'importance des structures de données de ce système de fichiers (par exemple, NTFS prend 30Mo versus <1Mo pour exFAT sur une mémoire flash de 512Mo). exFAT permet de dépasser la limite des 4 Go des systèmes de fichiers FAT32 et de potentiellement supporter des stockages de 16 Exaoctets (264 octets).

• Ouverture aux applications de sécurité tierces parties à l'utilisation de l'API **" Kernel Patch Protection "** sur les versions 64 bits de Windows Vista SP1. Le support de cette API, aussi connue sous le nom de PatchGuard, permet aux éditeurs d'anti-virus, et autres applications de gestion de la sécurité, de fonctionner de manière homogène et bas niveau avec les API de sécurité de Windows, notamment en donnant la possibilité aux éditeurs de logiciels de sécurité de désactiver le " Centre de Sécurité Windows " pour proposer leurs propres solutions. Le KPP permet au Kernel de détecter les modifications faites aux ressources Kernel ou au code du Kernel par un programme non autorisé et d'en arrêter le processus. En protégeant l'intégrité du Kernel, le KPP est une barrière à la plupart des virus, rootkits et autres programmes malveillants qui tentent d'atteindre les structures et le code du Kernel.

• **Support natif de la spécification " draft " IEEE 802.11n** qui permet d'accroître le débit des réseaux sans fils jusqu'à 600Mbps (auparavant 54Mbps pour 802.11a/g). Les constructeurs de matériels Wi-Fi prévoyaient en 2006 de supporter un débit allant jusqu'à 300Mbps. L'une des améliorations notables de la norme 802.11n est le composant MIMO – Multiple-Input Multiple-Output qui permet d'améliorer le multiplexage spatial en répartissant jusqu'à 4 flux spatiaux entre les antennes émettrices et réceptrices. Les iMac et MacBook/MacBook Pro à base d'Intel

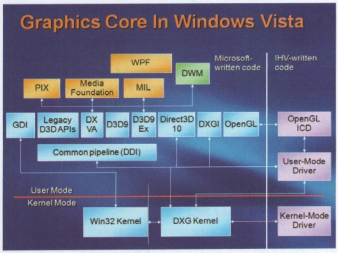


Fig.1

Core 2 Duo supportent cette norme au travers de la base Wi-Fi AirPort Extreme. Les possesseurs d'iMac ayant installé Windows Vista seront heureux de cette avancée. Côté PC, cette norme se retrouve pratiquement dans tous les matériels de constructeurs de routeurs WiFi (NetGear, D-Link, Buffalo…) et dans les PC/portables de dernière génération.

• L'extension des API **" Data Execution Prevention "** - DEP. Quelques fonctions nouvelles permettent aux développeurs de supporter DEP jusque dans leurs applications et notamment pour celles qui sont incompatibles avec DEP. En effet, certaines applications incompatibles avec DEP nécessitaient de stopper ce service pour pouvoir fonctionner. Avec ces évolutions, une application incompatible peut contourner DEP sans pour autant demander l'arrêt de cette protection pour tout le système. DEP est une barrière de protection contre les attaques de type " Buffer Overflow " qui consistent à exécuter du code dans des zones mémoire non permises.

• L'extension de **Direct3D avec la nouvelle version 10.1**. Direct3D 10.1, une évolution mineure de la version 10 et dont les spécifications étaient connues depuis longtemps déjà, a été intégrée au SP1 de Windows Vista. Cette nouvelle version apporte des optimisations qui ravironT les heureux possesseurs de cartes graphiques compatibles : La série HD 38xx des Radeon d'AMD & HD3000 series. Les plans de nVIDIA sont peu clairs sur le support de Direct3D10.1, le constructeur ayant annoncé qu'il supporterait directement la version 11 qui est déjà en cours de développement chez Microsoft. Direct3D est utilisé pour calculer et afficher les graphiques 3D dans les applications où la performance est importante (comme les jeux). Direct3D est utilisé par Windows Presentation Foundation pour le rendu des éléments 3D, mais basé sur la version Direct3D9Ex. Ces améliorations n'auront donc aucune incidence sur les applications WPF utilisant la 3D (Fig.1).

• **Améliorations des performances pour la copie de fichiers,** l'ouverture de fichiers volumineux, l'estimation du temps de copie de fichiers. Les modifi-

cations apportées concernent l'explorateur Windows mais aussi toutes les applications utilisant l'API **CopyFileEx**. Pour une information exhaustive sur les mécanismes de copie de fichiers dans Windows Vista, le blog de Mark Russinovich est un incontournable : http://blogs.technet.com/markrussinovich/ (voir l'article Inside Vista SP1 File Copy Improvements).

• **Modifications de Windows Search** pour permettre à d'autres moteurs de recherche de s'intégrer à Windows à la manière des moteurs de recherche Web dans Internet Explorer. Ceci permet à l'utilisateur de sélectionner un moteur de recherche pour Windows, différent de celui fourni par défaut par Microsoft (Windows Search).

• L'**alignement de Windows Vista sur Windows Server 2008** qui permet d'avoir des fichiers système communs aux deux versions. Vista SP1 bénéficie ainsi des optimisations de Windows Server 2008 sur le partage de fichiers, **Internet Information Server 7** et le support d'utilisateurs concurrents (notamment pour la prise de main à distance).

• Et encore beaucoup d'autres améliorations telles que le support du 64 bits pour le fournisseur OLEDB pour ODBC (MSDASQL), le support d'UEFI (Unified Extensible Firmware Interface) pour VISTA SP1 x64 ou encore un nouvel algorithme de compression pour les connexions RDP, etc.

Implémentation d'un moteur de recherche pour Windows Vista SP1

Comme nous l'avons vu, l'une des nouveautés intéressantes pour les développeurs est celle qui permet d'intégrer un moteur de recherche dans Vista SP1 à la place de celui fourni par Microsoft. Nous allons détailler les étapes et le code pour implémenter notre propre moteur de recherche, et le paramétrer pour qu'il soit utilisé lors d'une recherche depuis Windows ou l'explorateur.

Le procotole search

Pour permettre à d'autres moteurs de recherche de s'intégrer à Vista SP1, Microsoft a créé le protocole applicatif " search ". Ce protocole utilise une syntaxe qui permet de passer différents paramètres de requête à un moteur. Par défaut, les résultats d'une recherche sont affichés dans le Windows Search Explorer. Nous allons montrer comment utiliser les paramètres du protocole search pour afficher les résultats dans notre application. Ce nouveau protocole ne remplace pas le protocole " search-ms " qui continue d'être disponible aux applications pour exécuter des recherches sur l'index de Windows Search (voir : http://msdn2.microsoft.com/en-us/library/bb266520.aspx).

La syntaxe du protocole search est la suivante (encodée comme une url) :

```
search:parameter=value[&parameter=value]&
```

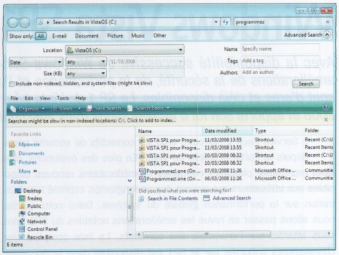
Les paramètres possibles sont les suivants : **query (la requête de l'utilisateur), crumb, subquery,** inputlocale, keywordlocale, syntax, stackedby, displayname. Vous trouverez le lien vers la référence du protocole search pour avoir les détails de chaque paramètre dans la section Références. Seuls les 3 premiers paramètres sont aujourd'hui utilisés par Windows Search. Ainsi, vous pouvez exécuter **(touche Windows+R)** la commande suivante :

```
search:query=programmez&crumb=location:C%3A%5C
```

Ce qui aura pour effet d'exécuter une recherche du mot " programmez " sur le répertoire racine " \ " de votre disque " C : ".

Notre moteur de recherche

Notre moteur de recherche est en fait une application qui devra s'enregistrer d'une certaine manière. Ainsi, nous allons créer une petite appli-



cation Windows Forms pour illustrer la récupération des paramètres d'une recherche et détaillerons l'installation de cette application.

Pour récupérer et afficher dans une liste les arguments d'appel de notre moteur, nous utiliserons le code suivant :

```
string[] args = Environment.GetCommandLineArgs();

paramsListBox.Items.Clear();
foreach (string arg in args)
{
  paramsListBox.Items.Add(arg);
}
```

Pour installer et enregistrer l'application, vous pouvez utiliser plusieurs méthodes telles qu'un programme Setup, une classe Installer, un script, etc. Notre application est installée dans le répertoire " C:\Program Files " et les informations utilisées pour l'enregistrement du moteur de recherche sont les suivantes :

```
HKEY_LOCAL_MACHINE\SOFTWARE\Classes\MySearchHelloWorld.
AssocProtocol.search
FriendlyTypeName = "My Search Hello World"
URL Protocol = ""

HKEY_LOCAL_MACHINE\SOFTWARE\Classes\MySearchHelloWorld.
AssocProtocol.search\shell\open\command
(Default) = "%ProgramFiles%\Mpoware\MySearchHelloWorld\MySearch
Engine.exe %1"

HKEY_LOCAL_MACHINE\SOFTWARE\RegisteredApplications
My Search Hello World = "Software\Mpoware\My Search Hello World
\Capabilities"

HKEY_LOCAL_MACHINE\SOFTWARE\Mpoware\My Search Hello World
\Capabilities
 ApplicationName = "My Search Hello World"
 ApplicationDescription = "My Search Hello World est un moteur de recherche
développé par Mpoware (encore plus puissant que Google & Live.com réunis! :-)"

HKEY_LOCAL_MACHINE\SOFTWARE\Mpoware\My Search Hello World
```

```
\Capabilities
\UrlAssociations
search = MySearchHelloWorld.AssocProtocol.search
```

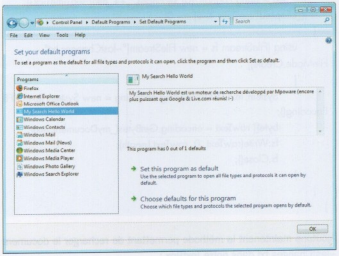
Configuration du moteur de recherche

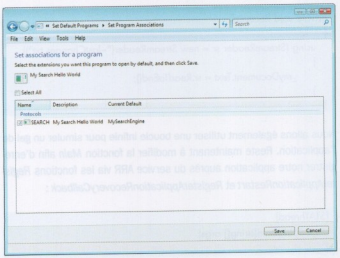
Votre nouveau moteur de recherche est développé, packagé et installé sur un Vista SP1. Il ne vous reste plus qu'à configurer Windows pour indiquer que vous souhaitez utiliser votre moteur lors d'une recherche depuis Windows. Il nous faut paramétrer notre programme comme étant le moteur de recherche par défaut de Windows (il existe 2 méthodes mais nous n'en détaillerons qu'une seule ici).

Paramétrage du programme par défaut pour le protocole search:

1. Cliquer sur **" Démarrer "**, puis **" Programmes par défaut "**.
2. Cliquer sur **" Paramétrer vos applications par défaut "**.
3. Dans la liste des programmes enregistrés, sélectionner le programme qui correspond à votre moteur de recherche (My Search Hello World dans notre exemple). Cliquer sur le bouton permettant d'associer les protocoles avec l'application (Choose defaults for this program).
4. Cocher le protocole SEARCH pour que votre application soit l'application exécutée par défaut lors d'une recherche, puis cliquer sur le bouton d'enregistrement du paramétrage.

Vous avez plusieurs manières de tester votre moteur de recherche : touche Windows + F, CTRL + F dans l'explorateur Windows, F3 dans le menu Démarrer ou l'explorateur Windows, ou plus communément la recherche depuis le menu Démarrer ou l'explorateur Windows.





Conclusion

Dans cet article, nous nous sommes intéressés aux améliorations importantes du service pack 1 de Windows Vista ayant un impact sur la sécurité, la performance, la robustesse et le déploiement. Bon nombre d'améliorations sont invisibles pour la plupart des développeurs, celles qui sont notables intéresseront les développeurs " bas niveau " (Kernel Patch Protection, DEP...), pour les autres, comme l'optimisation des performances d'Internet Information Server 7 par exemple, les développeurs en bénéficieront de facto. La nouveauté que nous avons mise en avant dans cet article concerne l'ouverture de la recherche dans Windows à d'autres moteurs de recherche que celui livré nativement par Microsoft (Windows Search). Ainsi, au travers d'un exemple simple, nous avons vu les possibilités d'intégration applicative du protocole de recherche. Il ne vous reste plus qu'à déployer le service pack 1 pour bénéficier de toutes ces améliorations. Pour cela, Microsoft a publié un guide de déploiement et une matrice accessible en ligne (http://technet.microsoft.com/en-us/windowsvista/cc137819 .aspx) mettant en correspondance le type de déploiement et l'outil recommandé (Windows Update, WSUS, SCOM 2007, WDS, etc.). Pour les entreprises qui ne souhaiteraient pas déployer automatiquement le SP1 via Windows Update, Microsoft fournit un outil (le Windows Service Pack blocker tool) qui permet de bloquer les mises à jour, et ce, pour Vista SP1, mais aussi, Windows 2003 SP2 et XP SP2. Dernier conseil avant de déployer : n'oubliez jamais de sauvegarder vos données et, si possible, votre système !

Références

- Document officiel des changements notables apportés par le Service Pack 1 de Windows Vista : http://www.microsoft.com/downloads/details.aspx?FamilyID=d69c4e1b-c81a-41be-b1f5-66e615ba5912&DisplayLang=en
- Extended FAT File System : http://msdn2.microsoft.com/en-us/library/aa914353.aspx
- Présentation de Direct3D 10 et 10.1 pour le SIGGRAPH 2007 (35 Mo) : http://www.microsoft.com/downloads/details.aspx?FamilyID=96cd28d5-4c15-475e-a2dc-1d37f67fa6cd&DisplayLang=en
- Utilisation du protocole search : http://msdn2.microsoft.com/en-us/library/bb776808.aspx

Frédéric QUEUDRET

CTO de la société Mpoware.
Accélérateur d'innovation. Société française d'édition de logiciels et de prestations de services sur la création d'outils de productivité pour les développeurs Visual Studio. http://www.mpoware.com/

Les nouvelles API de Windows Server : API Application Recover et Restart

Quel développeur n'a jamais essayé de mettre en place sur une application qu'il développe un système efficace de sauvegarde du travail effectué par l'utilisateur en cas de plantage ? La principale difficulté dans ce scénario est de pouvoir détecter que l'application ne répond plus, et de lui demander malgré tout d'exécuter une routine de sauvegarde/récupération puis d'éventuellement redémarrer en restaurant automatiquement les données.

Microsoft Windows Server 2008 propose de nouvelles API qui permettent de déléguer au système d'exploitation une partie de ce processus de récupération d'état de façon très efficace. Nous nous attarderons ici sur l'API Application Recover & Restart (ARR).

Application de test

Afin d'expliquer le fonctionnement de cette API, nous allons imaginer une simple application de traitement de texte créée avec le Framework .NET. Nous allons également ajouter un bouton qui provoquera un gel de l'application afin de simuler un plantage.

API Application Recovery & Restart

Une application peut utiliser l'API Application Recovery & Restart pour sauvegarder les données ainsi que son état avant qu'elle ne soit définitivement fermée suite à une exception non gérée ou un gel de l'interface. L'application peut alors être redémarrée et son état restauré si l'utilisateur le désire. Pour cela, il convient d'utiliser un certain nombre de fonctions :

- RegisterApplicationRestart : Inscrit l'application à redémarrer.
- RegisterApplicationRecoveryCallback : Précise la méthode à appeler pour sauvegarder les données et l'état de l'application au cas où l'application plante ou ne répond plus.
- ApplicationRecoveryInProgress : Indique à l'OS que l'application est en cours de sauvegarde.
- ApplicationRecoveryFinished : Indique que la sauvegarde est terminée.

Ces fonctions n'étant accessibles que dans la dll kernel32.dll, il est nécessaire de faire un peu d'interop pour y accéder depuis une application .NET :

```
[DllImport("kernel32.dll", CharSet = CharSet.Auto)]
public static extern uint RegisterApplicationRestart(string pwzCommand
Line, int dwFlags);
[DllImport("kernel32.dll")]
public static extern uint RegisterApplicationRecoveryCallback(APPLICATION
_RECOVERY_CALLBACK pRecoveryCallback, object pvParameter, int dw
PingInterval, int dwFlags);
[DllImport("kernel32.dll")]
public static extern uint ApplicationRecoveryInProgress(out bool pb
Cancelled);
[DllImport("kernel32.dll")]
public static extern uint ApplicationRecoveryFinished(bool bSuccess);
public delegate int APPLICATION_RECOVERY_CALLBACK(object pvParameter);
```

Pour plus d'informations sur les signatures de ces différentes fonctions, rendez vous sur cette page du site MSDN : http://msdn2.microsoft.com/en-us/library/aa373342(VS.85).aspx

La prochaine étape consiste à mettre en place dans l'application un système qui enregistrera périodiquement (ici toutes les 5 secondes) le texte présent dans notre TextBox. Pour ce faire, nous allons utiliser un simple Timer :

```
void _ellapsedTime_Tick(object sender, EventArgs e)
{
    _time.Text = string.Format("{0} secondes.",++_ellapsedTimeTicks);
    if ((_ellapsedTimeTicks % 5) == 0)
    {
        using (FileStream fs = new FileStream("~lastChanges.txt",
FileMode.Create))
        {
            System.Text.UTF8Encoding encoding = new System.Text.UTF8
Encoding();
            byte[] rawText = encoding.GetBytes(_myDocument.Text);
            fs.Write(rawText, 0, rawText.Length);
            fs.Close();
        }
    }
}
```

Ajoutons maintenant la méthode permettant de recharger le document ~lastChanges.txt dans notre TextBox :

```
private void Recover()
{
    using (StreamReader sr = new StreamReader("~lastChanges.txt"))
    {
        _myDocument.Text = sr.ReadToEnd();
    }
}
```

Nous allons également utiliser une boucle infinie pour simuler un gel de l'application. Reste maintenant à modifier la fonction *Main* afin d'enregistrer notre application auprès du service ARR via les fonctions *RegisterApplicationRestart* et *RegisterApplicationRecoveryCallback* :

```
[STAThread]
static void Main(string[] args)
{
```

```
Application.EnableVisualStyles();
Application.SetCompatibleTextRenderingDefault(false);
ApplicationRecovery.RegisterApplicationRestart("Crashed", 0);
ApplicationRecovery.RegisterApplicationRecoveryCallback(Recovery, null,
30000, 0);
Form1 myForm = new Form1(args.Length == 1 && args[0] == "Crashed");
Application.Run(myForm);
}
```

ApplicationRecovery.RegisterApplicationRestart("Crashed", 0) enregistre notre application auprès du service ARR de l'OS. Nous indiquons ici que la chaîne de caractère " Crashed " devra être passée en paramètre à la prochaine instance de l'application si elle est redémarrée, afin de lui indiquer dans quel contexte elle démarre. Le paramètre 0 indique à l'OS de redémarrer l'application quelle que soit la raison du plantage.

ApplicationRecovery.RegisterApplicationRecoveryCallback(Recovery, null, 30000, 0) permet de préciser le nom de la méthode à appeler en cas de plantage de l'application, c'est précisément dans cette méthode que la sauvegarde de l'état de l'application sera effectué. Dans notre cas, la sauvegarde de l'état de l'application étant déjà prise en charge avec notre Timer, la méthode Recovery n'aura aucun code supplémentaire pour sauvegarder l'application. Le deuxième argument, positionné à null, représente un " object " qui sera passé à la méthode Recovery si besoin. 30 000 représente en millisecondes le temps laissé à l'application pour indiquer à l'OS qu'elle est bien en train de sauvegarder son état. Passé ce délai, Windows considérera que l'application n'est plus en mesure de sauvegarder son état, et annulera cette procédure. Le dernier paramètre est réservé, et doit être égal à 0.

Remarquez enfin la modification de la signature du constructeur de la classe Form1 où nous passons maintenant en paramètre un booléen indiquant si l'application a été redémarrée suite à un plantage. Cette valeur sera utilisée par l'instance de Form1 pour savoir s'il est nécessaire d'appeler la méthode Recover avant affichage de la form.

```
public Form1(bool isRestarting)
{
InitializeComponent();
if (isRestarting)
{
Recover();
}
//Code d'initialisation ...
}
```

Bien que la méthode Recovery n'effectue dans notre application aucune action de sauvegarde, il est important de prêter attention au mécanisme de cette méthode en environnement réel.

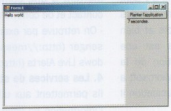
```
static int Recovery(object o)
{
Timer t = new Timer();
t.Interval = 1000;
t.Tick+=delegate(object sender, EventArgs e)
{
bool cancelled;
ApplicationRecovery.ApplicationRecoveryInProgress(out cancelled);
if (cancelled)
{
Console.WriteLine("Sauvegarde de l'état annulé par l'utilisateur.");
}
};
t.Start();
/*****
Code de sauvegarde de l'état l'application (éventuellement long)
*****/
t.Stop() ;
ApplicationRecovery.ApplicationRecoveryFinished(true);
return 0;
}
```

Le processus de sauvegarde peut être assez long en fonction de l'application, et ce processus peut lui-même planter. C'est la raison pour laquelle la fonction RegisterApplicationRecoveryCallback prévoit un délai accordé à la méthode de sauvegarde pour effectuer ses instructions. Si le processus de sauvegarde dure plus longtemps que prévu, il est nécessaire d'indiquer à l'OS que nous avons besoin de plus de temps pour terminer la sauvegarde, et ce aussi souvent que nécessaire. C'est précisément le rôle de la fonction ApplicationRecoveryInProgress utilize. Nous utilisons un Timer pour indiquer à l'OS, toutes les secondes, que le processus de sauvegarde est en cours et que nous avons besoin de plus de temps. Nous récupérons également en sortie un booléen indiquant si l'utilisateur a décidé d'interrompre cette procédure afin de stopper définitivement l'application.

Une fois la sauvegarde terminée, nous devons l'indiquer au service ARR via la fonction ApplicationRecoveryFinished.

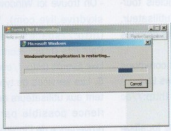
En exécutant notre application nous obtenons alors ce comportement

Lancement de l'application :



1 - Plantage de l'application

2 - Nous demandons à Windows 2008 de redémarrer l'application



3 - Notre application est restaurée dans l'état dans lequel elle était.

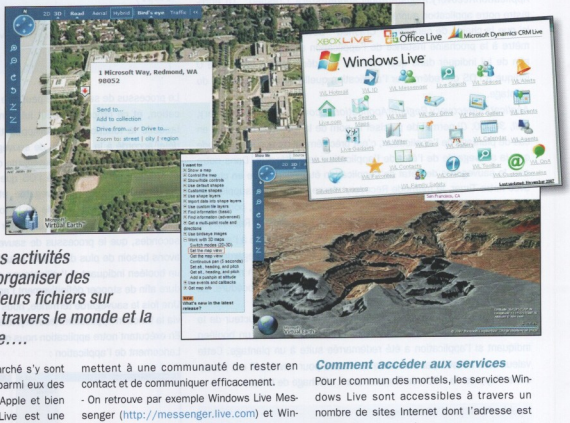
4 - Notez que l'application doit obligatoirement s'exécuter pendant au moins 60 secondes pour que son redémarrage soit proposé.

■ **Yann ALET** - *Développeur, Neos-SDI*

Recevez la Newsletter www.programmez.com/inscription.php

Développer avec Windows Live

Depuis maintenant quelques années, Internet regorge de services mis à la disposition des Internautes. Ces services leur permettent de gérer leur courrier électronique, gérer et interagir avec leurs amis, publier leurs photos, suivre les activités d'autres internautes, organiser des événements, stocker leurs fichiers sur Internet, téléphoner à travers le monde et la liste est encore longue….



Les acteurs majeurs du marché s'y sont tous attelés. On retrouve parmi eux des Yahoo!, Amazon, Google, Apple et bien entendu Microsoft. Windows Live est une plate-forme qui est la concrétisation de la vision "Software + Services" que Microsoft a annoncé à grand renfort de communication il y a 2 ans maintenant.

L'offre est composée non seulement de services en ligne, mais aussi de logiciels tournant sur la/les machine(s) de l'utilisateur. Windows Live se décline en plusieurs grandes catégories :

1. Les services d'identité : Ils sont communs à l'essentiel des services de Windows Live.

- On retrouve ici Windows Live ID (http://account.live.com).

2. Les services de recherche et de localisation : Ils permettent de rechercher de l'information en général.

- On retrouve ici Windows Live Search (http://search.live.com) et Windows Live Maps (http://maps.live.com).

3. Les services de communication : Ils permettent à une communauté de rester en contact et de communiquer efficacement.

- On retrouve par exemple Windows Live Messenger (http://messenger.live.com) et Windows Live Alerts (http://alerts.live.com).

4. Les services de stockage et de partage : Ils permettent aux utilisateurs de stocker de l'information en ligne et de la partager entre eux ou entre applications.

- On trouve ici Windows Live Skydrive (http://skydrive.live.com), Windows Live Spaces (http://spaces.live.com), Windows Live Contacts (http://contacts.live.com), Windows Live Events (http://events.live.com), etc.

5. Les services de type " média " : Ils permettent aux utilisateurs de vivre la meilleure expérience possible par du contenu riche en provenance de l'Internet

- On trouve ici Windows Live Silverlight Streaming (http://silverlight.live.com).

La plate-forme Windows Live peut fonctionner de manière transparente tant sur l'Internet, que sur les PC, téléphones mobiles et autres périphériques.

Comment accéder aux services

Pour le commun des mortels, les services Windows Live sont accessibles à travers un nombre de sites Internet dont l'adresse est structurée comme suit :

http://<Service>.live.com

En fonction de votre localisation géographique et de votre LiveID, le service va se présenter à vous dans votre propre langue et avec les partenaires spécifiques de votre pays. Certains services devant être exécutés sur la machine de l'utilisateur doivent être installés. Dans cette catégorie, on trouve Windows Live Messenger, Windows Live Writer, Windows Live Mail Desktop, etc. Beaucoup d'entre eux sont disponibles sur http://get.live.com/wl/config_all.

Les professionnels de l'informatique, quant à eux, disposent d'un ensemble d'interfaces (API ou "Application Program Interface") qu'ils peuvent utiliser pour intégrer, dans leurs applications, les services offerts par la plate-forme Windows Live. Ces interfaces sont :

- Soit de bas niveau, auquel cas le développeur dispose des interfaces brutes d'accès selon

les protocoles les plus adéquats (tels que REST, WebDAV, JSON, SOAP, etc.) et peut librement intégrer visuellement les services dans ses propres applications.

- Soit fonctionnellement très riches et faciles à mettre en œuvre, auquel cas le développeur dispose de Web Controls complètement prêts à l'emploi, faciles à mettre en œuvre, mais toutefois moins flexibles en terme de format visuel et de style.

Pour avoir accès à cet ensemble d'interfaces, il vous suffit de vous rendre sur http://dev.live.com (NDLR: En regardant le nom, on constate en effet qu'il s'agit bien là d'un Service ... pour les développeurs d'applications).

La notion d'identité

De nombreux services de la plate-forme Windows Live requièrent l'identification de l'utilisateur. La raison essentielle est en fait la personnalisation du service pour le rendre plus proche du mode de vie de son utilisateur (certains de nos lecteurs se souviendront du projet " Hailstorm " que Microsoft annonçait déjà en Mars 2001). L'identité est un élément clé de la plate-forme. Cette identité, connue sous le nom de LiveID, est simplement une adresse e-mail et un mot de passe. Toutes les adresses " @hot mail.com ", " @msn.com ", " @hotmail.fr ", etc. sont d'ores et déjà des LiveID. Toute adresse e-mail, quel que soit son domaine, peut devenir un LiveID. Les services ne sont donc pas uniquement disponibles aux utilisateurs ayant une adresse dans l'un des domaines de Microsoft. Quand on y réfléchit bien, il n'est pas rare pour un utilisateur d'avoir un LiveID principal et ... d'autres LiveID (publics ou non). Dans les faits, ce modèle colle très fort à la réalité car, en fonction du contexte, un utilisateur peut être successivement papa, employé de son entreprise, citoyen, patient, président du club de boules, trésorier du Kiwanis, etc., et pour chacun de ces rôles, adopte un comportement et un environnement social ajustés. La plate-forme Windows Live permet de très facilement passer d'une identité à une autre grâce au " Linked IDs " (ou " identifiants associés "). Dans les informations de compte liées à votre LiveID (http://account.live.com), vous avez la possibilité de créer une liste d'identités associées afin de faciliter le passage de l'une à l'autre.

Gratuit ou payant ?

Le modèle de l'Internet, comme tout modèle économique, requiert un flux d'argent car, après tout, offrir ces services est loin d'être gratuit. Toutefois, l'une des grandes différences spécifiques de l'Internet, c'est que ce n'est pas toujours l'utilisateur (ou bénéficiaire du service) qui paie. **Le modèle économique** de Windows Live est triple :

1. Un utilisateur bénéficie d'un service totalement **gratuit.** Dans ce cas, les revenus proviennent des messages publicitaires distillés aux utilisateurs au fur et à mesure de leurs manipulations.

2. Un utilisateur **paie** personnellement pour le service, se dégageant ainsi des messages publicitaires et, par le même fait, élargissant la richesse du service pour lequel il paie.

3. Un **partenaire** professionnel utilise la plate-forme Windows Live et ses services, sous sa propre marque, avec ses propres couleurs, moyennant rémunération à Microsoft. A titre d'exemple, et comme modèle d'inspiration, allez jeter un œil sur http://edu.live.com.

Le choix gratuit/payant peut être fait par l'utilisateur lui-même et ce, service par service. La liberté et la flexibilité sont très grandes.

Mais qu'en est-il des développeurs d'application? Doivent-ils payer une licence ? Doivent-ils faire payer leurs utilisateurs ?

En résumé :

- Pour les services qui requièrent l'identification de l'utilisateur, le paiement sera effectué sur la base du nombre de visiteurs uniques par mois et ce, seulement au-delà d'un certain volume (typiquement aujourd'hui 1 million d'utilisateurs uniques par mois).
- Pour les services qui ne requièrent pas l'identité de l'utilisateur, le paiement sera effectué sur la base du nombre de transactions effectuées ou de l'usage effectif du service et ce, seulement au delà d'un certain volume également (on parle ici d'un usage supérieur à plusieurs dizaines de milliers de transactions par jour).

Toutefois, la réponse exacte et la seule réponse valide à ces questions est donnée en long et en large sur http://dev.live.com/terms/licensing.aspx et http://dev.live.com/terms/tos.aspx.

Opportunité pour les développeurs

Vous l'aurez compris, la plate-forme Windows Live est bien conçue, accessible, disponible à des prix très démocratiques. Elle dispose de tout un ensemble d'interfaces qui permettent aux développeurs de l'intégrer dans leur propre application. Un exemple précis de ce type d'intégration est la Wygwam TV (http://tv.wygwam.com), solution packagée de Wygwam (http://www.wygwam.com), qui permet la création d'un site de type WebTV. Cette solution intègre :

- Windows Live ID, pour se connecter et sauvegarder des vidéos.
- Windows Live Silverlight Streaming : pour le streaming et l'hébergement gratuits de vidéos.
- Windows Live Maps : pour le positionnement des vidéos sur une carte géographique et donc permettre la géo-localisation.

Microsoft travaille également à la mise à disposition de services à très grande valeur ajoutée tels que Office Live (http://office.live.com), qui permettent à des TPE (Très Petites Entreprises), de gérer leur commerce en ligne. Ces services pourraient parfaitement être réalisés par vous-même pour créer de nouvelles offres intégrées à la plate-forme.

Dans les pages suivantes, nous allons vous montrer comment exploiter un certain nombre de ces services dans vos propres applications. Je vous souhaite beaucoup de succès... avec Windows Live.

Alain Leroy
Managing Partner
Wygwam Belux

Les API et services Live disponibles

Voici une liste non exhaustive des Services Live disponibles. Attention, certains ne le sont qu'en fonction de votre localisation.

Windows Live Partner
http://partner.live.com
Ce site permet de connaître l'étendue des services auxquels les partenaires Microsoft peuvent prétendre pour du développement autour de la plate-forme Windows Live.

Contrôle du partage d'information
http://consent.live.com
Ce site permet, essentiellement aux développeurs d'applications, de contrôler les autorisations de partage données à des tiers.

Admin Center
http://admin.live.com
Anciennement appelé "Custom Domains" (voir adresse dans cette liste), il est maintenant appelé "Windows Live Admin Center" et vous permet de gérer les LiveID dans vos domaines.

Live Events
http://events.live.com
Service Windows Live vous permettant de créer et de gérer vos événements (participants, agenda, etc.) et de partager les photos souvenirs.

Live Skydrive
http://skydrive.live.com
Votre "disque dur de l'Internet"... voici comment on peut qualifier Skydrive. Vous disposez d'un espace de stockage partageable ou privé de 500Mb parfaitement gratuit...

Live Homepage
http://home.live.com
La page dans laquelle les différents services Live que vous utilisez sont regroupés...

Live Hotmail
http://mail.live.com
Le fameux Hotmail que plus de 4,8 millions de personnes utilisent en Belgique ... mais à la sauce Windows Live. Fondamentalement revu et corrigé, c'est un réel outil exceptionnel.

Live Maps
http://maps.live.com
Un service vous donnant accès à une cartographie mondiale complète avec, en Belgique, la possibilité de rechercher des adresses, de trouver la route entre 2 points, etc.

Live Account Information
http://account.live.com
Vous y retrouvez toutes les informations relatives à votre Live account. Une page d'information bien pratique avec des trucs et astuces sur les actions les plus courantes.

Live OneCare
http://safety.live.com
Un anti-virus, anti-spam, anti-fishing gratuit.

Live Gallery
http://gallery.live.com
Un espace où vous pouvez trouver (et déposer) ces petites unités qui composent votre page sur Live.com et Live Spaces. Un répertoire impressionnant de modules très utiles...

Live Messenger
http://messenger.live.com
Pour chatter ou parler (son et image) avec tous ses contacts. On peut aussi l'utiliser pour téléphoner à ses amis non connectés à travers le monde.

Live.com
http://www.live.com
Votre home page sur le net. C'est l'endroit où vous rassemblez les informations que vous consultez jour après jour sur la grande toile.

Web Messenger
http://webmessenger.msn.com
Vous êtes-vous déjà retrouvé chez un ami ou sur une machine qui n'a pas Windows Live Messenger? Et bien, voici un endroit où vous pouvez trouver un Messenger qui fonctionne ? mais directement dans un navigateur (Firefox ou Internet Explorer).

Live Spaces
http://spaces.live.com
Espace de publication photos, billets, listes ...

Family Safety Settings
http://fss.live.com
Un service extraordinaire pour permettre aux parents de mieux protéger leurs enfants contre les éventuelles "légèretés" de la grande toile. Cela nécessite une petite mise en place, mais après, quelle tranquillité...

Live Alerts
http://alerts.live.com
Un service pour recevoir des notifications lorsque vos fournisseurs de contenu envoient une information à laquelle vous avez souscrit.

Live Labs
http://labs.live.com
Un petit espace d'expérimentation que Microsoft met à disposition du public pour se rapprocher un peu plus des résultats de ses recherches. On y retrouve notamment le projet Volta.

Live Microsoft Services Manager
http://msm.live.com
Ce service permet aux propriétaires de site, d'intégrer leur site avec la plate-forme Windows Live grâce aux LiveID. Ce site est le complément idéal du LiveID SDK. Réservé aux développeurs bien entendu.

Live Postmaster
http://postmaster.live.com
Tout ce que vous avez toujours rêvé de savoir à propos de Windows Live Hotmail est disponible ici (aussi des réponses à : Pourquoi mes mails sont-ils bloqués par Hotmail, politique Anti-Spam, Anti-Virus, etc.)

Live Search Club
http://club.live.com
Un petit service très sympa (et Américain) qui permet de jouer à plein de petits jeux. Attention, les prix ne sont valables que pour les clients américains!

Silverlight Streaming
http://silverlight.live.com
Un service permettant aux développeurs d'applications Silverlight de bénéficier de 4Gb sur Internet pour stocker du contenu riche (vidéo, etc.) à utiliser dans leurs applications. Ce service est toujours en Bêta et est réservé aux développeurs!

Live Expo
http://expo.live.com
Un site (Américain pour le moment) pour acheter et vendre ... tout ce qui peut l'être. Donc, un site de petites annonces.

Live ID
Pour utiliser la plate-forme Windows Live, puisqu'il s'agit de votre web personnel, il est nécessaire d'obtenir un identifiant que l'on appelle: Live ID.

Windows Live @ Edu
http://edu.live.com
Service spécialement destiné aux écoles supérieures et universités pour rester en contact avec leurs étudiants actuels et passés.

Live Product Upload
http://productupload.live.com/
Vous permet de publier votre catalogue de produits de façon à le rendre disponible à travers Live Product Search.

Live for Developers
http://dev.live.com
Tout ce qu'il faut connaître pour intégrer les services Live à vos propres applications.

Live Custom Domains
http://domains.live.com
Si vous possédez votre propre nom de domaine et que vous voulez obtenir des adresses e-mail GRATUITES pour votre nom de domaine... c'est ici qu'il faut venir! Tout simplement fabuleux pour les associations, les clubs, les groupes, etc.

Live Favorites
http://favorites.live.com
Vous connaissez les favoris de votre navigateur? Et bien, ces favoris-ci vous accompagnent partout où vous allez...

Live Search
http://search.live.com
Le moteur de recherche de la plate-forme Live.

Live Webmaster services
http://webmaster.live.com
Tout ce dont un Webmaster peut avoir besoin pour faire connaître son site par la plate-forme Windows Live.

Live Translator
http://translator.live.com
Un service de traduction automatisée pour un texte personnel ou une URL.

Virtual Earth : une API surpuissante

L'API Virtual Earth permet l'intégration de cartes interactives au sein de vos sites internet. On peut alors y retrouver des informations géographiques précises donnant une autre manière de visualiser du contenu.

L'intégration de l'API Virtual Earth est simple et son utilisation se fait à l'aide de JavaScript. Il est par ailleurs important de souligner les capacités de la nouvelle mouture de Visual Studio 2008 dans ce domaine, d'autant plus qu'un projet CodePlex auquel j'ai participé, permet d'avoir une Intellisense très utile pour l'utilisation de l'API. Ce projet nommé " Virtual Earth JavaScript Intellisense Helper " est disponible à l'adresse suivante : http://www.codeplex.com/VEJS Pour obtenir l'Intellisense, il suffit d'ajouter le dossier VEJS du projet CodePlex, dans votre solution (figure 1).

Fig.1

En insérant ensuite la référence vers le fichier dédié :

```
/// <reference path="VEJS/VeJavaScriptIntellisenseHelper.js" />
```

On obtient alors une aide à la saisie au sein du fichier JavaScript dans lequel est placée cette référence, WygScript.js dans notre cas (figure 2).

Tous les chemins mènent à ...

A travers cette partie, nous allons étudier l'utilisation basique du MapControl et la gestion des événements utilisateurs. Ainsi pour réaliser cette première application, nous allons utiliser une page HTML (statique) et un fichier JavaScript qui nous permettra d'établir notre code. L'architecture est présentée un peu plus tôt (figure 1).

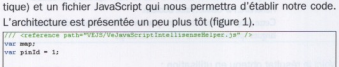


Fig.2

Définition du code HTML

Dans la page HTML, on ajoute très peu de contenu, parmi les éléments à ajouter, la référence vers le script de Virtual Earth, remarquez le paramètre permettant de définir le " market place " :

```
<script type="text/javascript" src="http://dev.virtualearth.net/mapcontrol
/mapcontrol.ashx?v=6&mkt=fr-fr"></script>
```

Et la DIV qui va accueillir le MapControl :

```
<div id="wygMap" style="position:absolute; width:500px; height:300
px;"></div>
```

Initialisation du MapControl

On définit dans notre fichier JavaScript, la méthode InitMap() qui se chargera d'initialiser notre MapControl. Cette méthode créée prend en paramètre une chaîne correspondant à l'id de la DIV dans laquelle sera insérée notre MapControl.

```
function InitMap(id)
{
    // Coordonnées du centre de la carte
    var mapCenter = new VELatLong(47, 3.4);

    // Initialisation du MapControl
    map = new VEMap(id);
    map.LoadMap(mapCenter);

    // Gestion du MapControl
    map.SetScaleBarDistanceUnit(VEDistanceUnit.Kilometers);

    // Gestion des événements
    map.AttachEvent("onclick",callback_MapClick);
    map.AttachEvent("onkeypress",callback_MapKeyPress);
}
```

On remarque dès lors, l'utilisation de plusieurs classes et méthodes de l'API Virtual Earth. Observons plus en détails la méthode LoadMap() :

```
VEMap.LoadMap(VELatLong, zoom, style, fixed, mode, showSwitch, tileBuffer);
```

Paramètres :

VELatLong	Centre de la carte.VELatLong
zoom	Niveau de zoom appliqué, par défaut 4. Entier de 1 à 19 (possibilité jusqu'à 21).
style	Style de carte (Aérien, Hybride, Route, Oblique).VEMapStyle enum.
fixed	Indique si la carte est interactive. Booléen.
mode	Indique le mode utilisé (Mode 2D ou 3D).VEMapMode enum.
showSwitch	Affichage ou non du panneau de contrôle de la carte. Booléen.
tileBuffer	Indique le nombre d'anneaux de tiles chargés autour de la vue courante. Par défaut : 0. Entier de 0 à 3.

Une autre méthode intéressante est celle de gestion des événements :

```
VEMap.AttachEvent(event, function);
```

Paramètres :

event	Nom de l'événement qui déclenche l'appel (onclick, onkeypress, onchangeview…). Chaîne de caractères.
function	Méthode à appeler en callback

Il est alors possible de récupérer de nombreuses informations à travers les paramètres des méthodes de callback associées aux événements.

Gestion de la souris et ajout d'élément

Ici nous allons capturer les clics droits qui définiront nos points de passage qui seront stockés dans un tableau de points.
Ceci est réalisé dans la méthode callback_MapClick() associée au clic sur le MapControl :

```
function callback_MapClick(e)
{
    if(e.rightMouseButton)
```

```
    {
        // Il s'agit d'un clic droit
        // > On ajoute comme point de passage

        // Récupération des coordonnées de clic - Mode 2D
        var x = e.mapX;
        var y = e.mapY;
        var pixel = new VEPixel(x, y);

        // Conversion des coordonnées : pixel > geographique
        var pointLatLong = map.PixelToLatLong(pixel);

        // Ajout d'une nouvelle punaise
        var shape = new VEShape(VEShapeType.Pushpin,pointLatLong);
        shape.SetTitle("Punaise n°" + pinId+1);
        shape.SetDescription("N°" + pinId+1 + " - Wouhou!");

        // Ajout de la punaise sur le layer de base
        map.AddShape(shape);

        //Ajout de la référence au tableau de points de passage
        pointPassage[pinId] = pointLatLong;
        pinId++;
    }
}
```

On peut récupérer plusieurs informations à travers le paramètre de la méthode, notamment sur les touches clavier pressées au moment du clic ou l'élément sur lequel le clic a été déclenché…
Ici on utilise uniquement la valeur de la propriété *rightMouseButton* afin de vérifier qu'il s'agit d'un clic droit.
Ensuite nous pouvons observer l'utilisation d'une classe particulière, la classe VEShape. C'est cette dernière qui définit les éléments ajoutés sur le MapControl que ce soient des punaises, des polygones ou des polylignes. Voici son constructeur :

```
var x = new VEShape(type, points);
```

Paramètres :

type	Type de l'élément (Pushpin, Polygon ou Polyline). VEShapeType enum.
points	Point ou tableau de point définissant les informations géographiques de l'élément. VELatLong ou VELatLong[].

Une fois initialisé, on peut éditer les propriétés de cet élément, pour cela on utilise les méthodes dédiées. Ici on met en œuvre uniquement le titre et la description qui apparaîtront dans la bulle d'information affichée au survol. Il existe de nombreuses autres propriétés et méthodes pour exploiter ces éléments et les customiser au besoin.

Gestion du clavier et calcul d'itinéraire

Une fois un ou plusieurs points de passage ajoutés à l'itinéraire, on peut appuyer sur la lettre " g " du clavier et calculer l'itinéraire :

```
function callback_MapKeyPress(e)
{
    if(e.keyCode == 71)
    {
        if(pointPassage.length != 0)
        {
            // Il s'agit de la lettre "g" (comme GO)

            // Point d'arrivée = Wygwam
            pointPassage[pinId] = pointWygwam;

            // On nettoie la carte
            map.Clear();

            // Définition des options de calcul d'itinéraire
            var options = new VERouteOptions();
            options.DistanceUnit = VERouteDistanceUnit.Kilometer;
            options.UseMWS = true;  // Permet l'internationalisation

            // Calcul de l'itinéraire
            map.GetDirections(pointPassage,options);
        }
    }
}
```

A travers le paramètre de la méthode de callback, on peut également récupérer différentes informations concernant les touches pressées ou d'autres informations sur l'état du MapControl. Ici, on s'intéresse uniquement à la propriété keyCode permettant de récupérer la touche pressée. Après avoir vérifié que l'on avait défini un ou plusieurs points, on nettoie la carte de tous les éléments. On initialise un objet de type VERouteOptions permettant de définir les options de calcul d'itinéraire à travers les propriétés de l'objet telles qu'ici :

DistanceUnit	Unité de distance utilisée pour le calcul de l'itinéraire. Par défaut : Mile. VERouteDistanceUnit enum.
UserMWS	Indique si on souhaite utiliser le Web Service MapPoint pour calculer l'itinéraire. Par défaut : Faux. Booléen. Cette propriété doit être à vrai pour avoir les itinéraires dans d'autres langues spécifiées dans la référence au script.

Voici le résultat obtenu en utilisation :

On peut ainsi voir une utilisation simple de l'API Virtual Earth, les possibilités de cette API s'étendent bien plus et vous permettront de réaliser des interfaces d'exploration de contenu innovantes. Une nouvelle version 7 sera annoncée très prochainement, avec toujours plus de fonctionnalités, l'utilisation de Silverlight et la technologie DeepZoom pour naviguer sur la carte.

■ **Nicolas Boonaert** - *Collaboration and Portal Team - Wygwam*

Windows Live ID : demander le passeport

Vous êtes probablement l'un des 380 millions d'utilisateurs de Live ID, anciennement connu sous le nom de " Microsoft Passport ". Aujourd'hui, vous pouvez bénéficier de Live ID sur votre propre site. L'utilisateur pourra ainsi se connecter sur votre site à l'aide de ses informations d'authentification (Login, password).

Microsoft vient de sortir son " Live ID Web Authentification SDK " mais voilà déjà que les forums de développeurs regorgent de questions en tout genre : *" Comment l'utiliser ? ", " Je ne peux pas récupérer l'email, aidez moi ! ", ...* Heureusement, cet article va répondre à toutes les questions que vous pouvez vous poser, la première d'entre elles étant probablement : *" A quoi ça sert ? ".* La réponse que je vous ferai sera : *" certainement pas à récupérer les informations d'un compte Live ID "* ou en tout cas, pas les informations personnelles d'un membre telles que son email, nom, prénom etc. Alors, dans ce cas, à quoi ça sert ? Et bien ce SDK va vous permettre de gérer l'authentification, et seulement cela. Là, certains d'entre vous pourraient douter de l'utilité d'un tel SDK et pourtant, rendre l'authentification plus facile aux internautes peut grandement favoriser l'utilisation de vos services. Nous allons montrer comment utiliser ce SDK au sein d'une application ASP.NET utilisant le SQL MemberShip Provider.

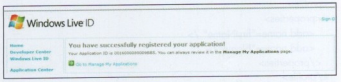
Live ID integré au MemberShip Provider

La première chose à faire est de télécharger le SDK. Dans ce dernier, vous trouverez un exemple d'utilisation. Nous allons partir de cet exemple et le faire fonctionner avec le SQL MemberShip Provider. La version actuelle du SDK est la 1.1. Pour le télécharger, rendez-vous à cette url : http://www.microsoft.com /downloads/details.aspx?familyid=24195B4E-6335-4844-A71D-7D39 5D20E67B&displaylang=en

Téléchargez le fichier MSI. Cette page reprend également différents exemples en Java, Ruby etc. Une fois installé, vous avez la possibilité d'ouvrir un projet d'exemple (par défaut dans C:\Program Files\Windows Live ID\WebAuth\Sample). Dans App_Code, nous avons une classe " WindowsLiveLogin ", c'est cette classe que nous allons utiliser pour s'authentifier à l'aide de notre Live ID. Cette classe à besoin de certaines informations que vous obtiendrez en enregistrant votre application web dans " Windows Live ID Application Center " (https://msm.live.com/app/default.aspx). Pour cela, cliquez sur " Register an Application ". Vous obtenez alors un formulaire que vous devez remplir

- **Application Name :** correspond au nom que vous voulez donner à votre application
- **Return RUL :** est l'url de la page qui sera renvoyée après s'être authentifié sur Live. C'est sur cette page que vous recevrez les différentes informations d'authentification.
- **Domain Name :** est le domaine sur lequel vous utilisez l'authentification par Live ID. Ce domaine doit bien entendu correspondre au domaine utilisé dans l'URL de retour.
- **Secret Key :** est une clef dont vous aurez besoin pour utiliser la classe WindowsLiveLogin.
- **Application verifier required :** laissez par défaut.

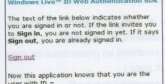
Acceptez les conditions d'utilisation et cliquez sur " submit ". Vous obtenez alors votre application ID qui est la deuxième information dont vous avez besoin pour initialiser le WindowsLiveLogin. Pour faire fonctionner



l'exemple fourni avec le SDK, vous devez renseigner ces différentes informations dans le web.config :

```
<appSettings>
        <add key="wll_appid" value="APPLICATION ID"/>
        <add key="wll_secret" value="SECRET KEY"/>
        <add key="wll_securityalgorithm" value="wsignin1.0"/>
</appSettings>
```

Après avoir correctement configuré votre application web, vous pouvez lancer le site et voir ce que cela donne. En cliquant sur le lien " sign in ", vous arrivez sur la page de connexion des services Live et après vous être authentifié, vous arrivez sur votre page avec une série d'informations. Ces informations vont nous être utiles afin de pouvoir obtenir



" l'UserUD " de l'utilisateur. Lorsque l'utilisateur revient sur votre site **il est déjà authentifié.** Le lien " sign in " est affiché à l'aide d'une iframe qui affiche le " contrôle " d'authentification Windows Live ID :

```
<iframe
    id="WebAuthControl"
    name="WebAuthControl"
    src="http://login.live.com/controls/WebAuth.htm?appid=<%=AppId%>&style=font-size%3A+10pt%3B+font-family%3A+verdana%3B+background%3A+white%3B"
    width="80px"
    height="20px"
    marginwidth="0"
    marginheight="0"
    align="middle"
    frameborder="0"
    scrolling="no">
</iframe>
```

Comment allons-nous faire pour transformer ce projet en un projet qui utilise le MemberShip Provider ? Nous allons prendre comme convention que le nom d'utilisateur et le mot de passe dans le MemberShip Provider seront, en fait, l'UserID obtenu à l'aide du jeton que l'authentification Live ID nous a envoyé. Pour obtenir cet UserID, rien de plus simple :

```
WindowsLiveLogin.User user = wll.ProcessToken(token);
string UserID = user.Id;
```

Rendez-vous maintenant sur l'outil de configuration de votre site internet pour activer le MemberShip Provider SQL. Une fois ceci fait, il ne reste plus qu'à implémenter la logique que nous avons décrite un peu plus haut. Nous allons donc utiliser l'authentification Live ID et utiliser le MemberShip Provider pour stocker les informations personnelles de l'utilisateur. Nous allons aussi utiliser les *profiles* pour stocker des informations supplémentaire. Dans le web.config on trouvera donc :

```
<profile>
  <properties>
    <add name="firstName" />
    <add name="Name"/>
  </properties>
</profile>
```

Attention ! vous devez changer la stratégie de mot de passe du MemberShip Provider car l'UserID ne comprend pas de caractère spécial. Pour ce faire, nous devons ajouter quelques lignes dans le web.config

```
<authentication mode="Forms"/>
<membership defaultProvider="MyProvider">
  <providers>
    <add
      name="MyProvider"
      type="System.Web.Security.SqlMemberShipProvider"
      minRequiredNonalphanumericCharacters="0"
      connectionStringName="LocalSqlServer"
      />
  </providers>
</membership>
```

Une fois ceci fait, il suffit de regarder si dans notre MemberShip nous avons un utilisateur dont le login et le password correspondent à L'UserID

- Si oui : on affiche les différentes informations en provenance de notre MemberShip (redirection vers Default.aspx)
- Si non : On permet à l'utilisateur de compléter son profil.

```
if (IsPostBack)
{
    WindowsLiveLogin.User user = wll.ProcessLogin(req.Form);

    HttpCookie loginCookie = new HttpCookie(LoginCookie);
    if (user != null)
    {
        loginCookie.Value = user.Token;

        if (user.UsePersistentCookie)
        {
            loginCookie.Expires = PersistCookie;
        }
    }
    else
    {
        loginCookie.Expires = ExpireCookie;
    }

    res.Cookies.Add(loginCookie);
    //res.End();
    UserId = user.Id;
    Session["UserID"] = UserId;
    MembershipUser user1 = Membership.GetUser(UserId);
    if (user1 != null)
        Response.Redirect("Default.aspx");
}
```

Le formulaire d'ajout est très simple. Il contient 3 TextBox : une pour l'email, une pour le nom et une autre pour le prénom. Un bouton qui va permettre d'enregistrer les informations sur l'utilisateur actuellement connecté :

```
MembershipCreateStatus status;
UserId = (string)Session["UserID"];
MembershipUser user = Membership.CreateUser(UserId, UserId, txtEmail.
Text, "Quel est mon employeur ?", "Wygwam", true, out status);
if (user != null)
{
    Profile.Initialize(UserId, true);
    Profile.firstName = txtPrenom.Text;
    Profile.Name = txtNom.Text;
    Profile.Save();
}

Response.Redirect("Default2.aspx");
```

Au niveau de notre page Default.aspx, nous allons rechercher les différentes informations de l'utilisateur si celui-ci est renseigné dans le MemberShip Provider et nous les affichons dans un label :

```
if (!string.IsNullOrEmpty(token))
{
    WindowsLiveLogin.User user = wll.ProcessToken(token);

    if (user != null)
    {
        UserId = user.Id;
        if (Membership.ValidateUser(user.Id, user.Id))
        {
            MembershipUser memuser = Membership.GetUser(user.Id);
            Profile.Initialize(user.Id, true);
        this.lbWelcome.Text= "Bonjour " + Profile.firstName + " " + Profile.
Name + "(" + memuser.Email + ")";

        }
    }
}
```

Voilà Live ID correctement intégré dans le MemberShip Provider d'ASP.NET. Vous pourrez retrouver plus d'information sur les différentes API de Live ID sur le site de MSDN. Sachez que le SDK Live ID à d'autres utilités dans d'autres contextes. Il est par exemple utilisé afin de pouvoir contacter certaines API comme l'API Live Contact et l'API Live Gallery. Les informations dont vous aurez dès lors besoin ne se trouvent plus dans *WindowsLiveLogin.User* mais dans *WindowsLiveLogin.Consent Token*. Pour plus d'information sur le sujet, je vous laisse soin de consulter l'aide MSDN.

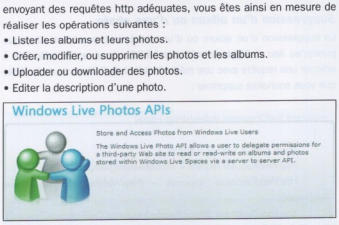
■ **Loic Bar** - *Développeur .NET - Wygwam*

Tout savoir sur Windows Live Photo API

Comme nous avons pu le constater, la plate-forme Windows Live dispose de très nombreux services. Cet article va s'intéresser à l'accès aux photos de votre compte Windows Live via l'API Windows Live Photo.

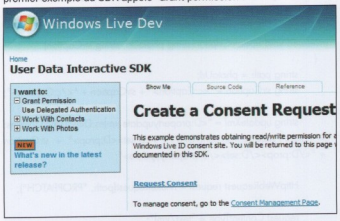
L'API Windows Live Photo vous permet de visualiser et de mettre à jour les albums et les photos associées à votre compte Windows Live en utilisant les protocoles WebDAV ou ATOM. En envoyant des requêtes http adéquates, vous êtes ainsi en mesure de réaliser les opérations suivantes :

- Lister les albums et leurs photos.
- Créer, modifier, ou supprimer les photos et les albums.
- Uploader ou downloader des photos.
- Editer la description d'une photo.

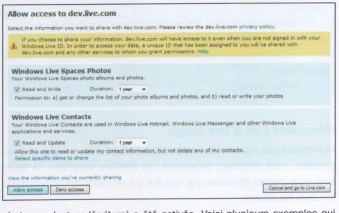


Windows Live Dev

Le site 'User Data Interactive SDK' (http://dev.live.com/livedata/sdk/) présente plusieurs exemples d'utilisation de l'API de stockage de photos. Afin de pouvoir exécuter les exemples, la première étape est d'autoriser explicitement votre compte Windows Live à accéder à vos photos en lecture /écriture. Cela peut être facilement réalisé en passant par le premier exemple du SDK appelé 'Grant permission'.



En cliquant sur le bouton 'Request Content', vous êtes en effet redirigé vers une page du site https://consent.live.com qui va vous demander de confirmer la permission (afin de permettre d'accéder à vos contacts ou à vos photos notamment) et de choisir la durée pendant laquelle vous souhaitez que celle-ci soit effective. Vous pourrez ensuite tester les exemples du SDK puisque vous aurez obtenu un *lid* qui sera utilisé pour construire l'url par laquelle vous devrez passer pour envoyer vos requêtes, ainsi qu'un *delegation token* que vous passerez en paramètre dans l'en-tête de ces requêtes. Vous constaterez également que la permission SpacesPhotos.ReadWrite (qui vous autorise à accéder à vos



photos en lecture/écriture) a été activée. Voici plusieurs exemples qui illustrent les différentes opérations que vous pouvez réaliser en envoyant des requêtes WEBDAV au service de stockage des photos. Dans un premier temps, vous devez initialiser l'url à laquelle vous allez envoyer les requêtes http. Vous devez donc disposer d'un *lid* et d'un *delegation token* valides que vous aurez obtenus après l'étape d'autorisation précédente.

```
string lid = "Votre lid";
string delegationToken = "EwCoARAnAAAUaiat...Zxvaoo4TRgAAA%3D%3D";
string strRoot = "";
NameTable nt = new NameTable();
XmlNamespaceManager xmlnsm = null;

 protected void Page_Load(object sender, EventArgs e)
   {
        if (!WindowsLiveLogin.ConsentToken]Session["ConsentToken"] != null)
        {
             WindowsLiveLogin wll = new WindowsLiveLogin(true);
             WindowsLiveLogin.ConsentToken ct = (WindowsLiveLogin.Consent
Token)Session["ConsentToken"];
             lid = ct.LocationID;
             delegationToken = ct.DelegationToken;
        }

        strRoot = "https://cumulus.services.live.com/@C@" + lid + "/Spaces
Photos/";

        lblRootUrl.Text = strRoot;
        xmlnsm = new XmlNamespaceManager(nt);
        xmlnsm.AddNamespace("D", "DAV:");
        xmlnsm.AddNamespace("c", "http://storage.msn.com/DAV/");
   }
```

Afin de mener à bien ces opérations, il faudra être en mesure d'envoyer des requêtes HTTP et de pouvoir analyser le contenu du XML renvoyé en

Abonnez-vous, voir coupon p. 51 et sur www.programmez.com

réponse. Voici donc une fonction que nous utiliserons souvent puisqu'elle permettra d'appeler une url et de lui passer la commande à transmettre.

```
private HttpWebRequest MakeRequest(string reqPath, string method)
  {
      HttpWebRequest request = (HttpWebRequest)WebRequest.Create
(reqPath);
      request.UserAgent = "Microsoft Windows Live Photo Demo";
      request.Method = method;
      request.Headers.Add("Authorization", "DelegatedToken dt=\"" + dele
gationToken + "\"");
      return request;
  }
```

Récupérer l'url d'une image

Maintenant que vos données permettant d'accéder au service, sont initialisées, vous trouverez ci-dessous, de manière simplifiée, la façon dont vous pouvez effectuer une requête qui récupère des informations sur tous les albums :

```
XmlDocument xmld = new XmlDocument();
  protected void listAlbums()
  {
      string path = strRoot;
      //On envoie une requête PROPFIND.
      HttpWebRequest request = MakeRequest(path, "PROPFIND");
      request.Headers.Add("Depth", "1");

      //On récupère la réponse.
      HttpWebResponse response = (HttpWebResponse)request.Get
Response();
      xmld.LoadXml((new StreamReader(response.GetResponseStream()).
ReadToEnd()).ToString());
      response.Close();

      //On boucle sur les éléments du noeud XML qui contient les informations
des albums.
      XmlNodeList xmlnl = xmld.SelectNodes("/D:multistatus/D:response
/D:propstat/D:prop", xmlnsm);
      foreach (XmlNode xmlnode in xmlnl)
          {
              XmlNode href = xmlnode.SelectSingleNode("D:href", xmlnsm);
              XmlNode name = xmlnode.SelectSingleNode("D:displayname",
xmlnsm);
              //On utilise les informations de l'album.
          }
      }
  }
```

Création d'un album

Pour créer un album il vous suffit d'envoyer une requête avec une méthode MKCOL et de mettre le nom de l'album à la fin de l'url :

```
protected XmlDocument createAlbum(string albumName)
  {
      string path = strRoot + albumName;
      HttpWebRequest request = MakeRequest(path, "MKCOL");
      HttpWebResponse response = (HttpWebResponse)request.GetRes
ponse();
      doc.LoadXml((new StreamReader(response.GetResponseStream()).
ReadToEnd()).ToString());
      response.Close();
      return doc;
  }
```

Suppression d'un album ou d'une photo

La suppression d'un album ou d'une photo est quasiment identique puisqu'au lieu d'utiliser la méthode MKCOL, vous aurez cette fois à envoyer une requête avec une méthode à l'url de l'album ou de la photo que vous souhaitez supprimer :

```
protected XmlDocument delete(string name)
  {
      string path = strRoot + name;
      HttpWebRequest request = MakeRequest(path, "DELETE");
      HttpWebResponse response = (HttpWebResponse)request.Get
Response();
      doc.LoadXml((new StreamReader(response.GetResponseStream()).
ReadToEnd()).ToString());
      response.Close();
      return doc;
  }
```

Modifier la description d'une image

Pour modifier la description d'une image, vous devez envoyer une requête WEBDAV qui contiendra le texte à affecter avec une méthode PROPPATCH et appeler l'url de la photo correspondante :

```
protected void updateCaption(string photoUrl, string strCaption)
  {
      string path = photoUrl;
      string strCaption = "<c:Caption>" + strCaption + "</c:Caption>";

      string updateXml = "<D:propertyupdate xmlns:D=\"DAV:\" xmlns:
c=\"http://storage.msn.com/DAV/\"><D:set><D:prop>" + strCaption
 + "</D:prop></D:set></D:propertyupdate>";

      HttpWebRequest request = MakeRequest(path, "PROPPATCH");

      request.ContentType = "text/xml";
      request.ContentLength = updateXml.Length;
      using (StreamWriter writer = new StreamWriter(request.GetRequest
Stream()))
          writer.Write(updateXml);

      // Analyser la répone pour voir si tout s'est bien passé
  }
```

En réponse, vous obtiendrez un code qui vous permettra de déterminer si tout s'est bien passé ou si une erreur s'est produite.

Upload d'une photo

Enfin, finissons avec un dernier exemple : Pour uploader une photo dans un album, vous devez créer une requête avec une méthode PUT puis copier le flux de l'image à l'intérieur et appeler une url construite avec le nom de l'album suivi du nom de l'image.

```
protected void uploadPhoto(string albumName)
    {
        string path = "https://cumulus.services.live.com/" + "@C@"
+ lid + "/SpacesPhotos/" + albumName + fileUpload.FileName;
        HttpWebRequest request = MakeRequest(path, WebRequest
Methods.Http.Put);
        request.ContentType = fileUpload.PostedFile.ContentType;
        request.ContentLength = fileUpload.PostedFile.ContentLength;

        //On copie le flux de l'image dans la requête.
        using (Stream newStream = request.GetRequestStream())
        {
            CopyStream(fileUpload.PostedFile.InputStream, newStream);
        }

        //On analyse la réponse obtenue.
        try
        {
            HttpWebResponse response = (HttpWebResponse)request.
GetResponse();
            lblResponse.Text = response.StatusCode.ToString();
        }
        catch (WebException ex)
        {
            lblResponse.Text = "Request failed - " + Environment.NewLine
+ ex.Message;
        }
    }

private static void CopyStream(Stream istream, Stream ostream)
    {
        byte[] buffer = new byte[2048];
        int bytes;
        while (0 < (bytes = istream.Read(buffer, 0, buffer.Length)))
            ostream.Write(buffer, 0, bytes);
    }
```

Des API, différents modèles

" Nous avons différents niveaux avec les API Live. Un premier niveau orienté contrôle, pour des utilisateurs non développeurs, en copiant un petit morceau de code, par exemple pour récupérer une vidéo. Au niveau des développeurs, quand on y regarde de plus près, cela manque de cohérence. Il y a webdav, du SOAP, du Rest… Les équipes de développements devraient fournir des API cohérentes. Selon les besoins, les objectifs de l'API, les techniques et protocoles ne sont pas identiques partout. Il faut avoir le bon protocole pour le bon usage. Il existe aussi différents modèles d'intégration, nous avons des API côté client et d'autres côté serveur. Maintenant, ces API visent majoritairement les développeurs Windows (mais pas seulement, ndlr). C'est pour cela que des kits Windows Live, Windows Live Tools for Visual Studio ont été packagés. On dispose aussi de wrapper pour embarquer des API Live dans son application .Net (ex. : LiveNet, projet codeplex, ndlr) ! " nous indique Christophe Lauer, spécialiste plate-forme Web et Windows Live chez Microsoft.

Côté mobile, la stratégie Windows Live n'est pas encore particulièrement claire. C'est un peu au cas par cas, certaines API peuvent fonctionner mais il faudra sans doute attendre encore un peu pour en savoir plus. Sur leur intégration dans Expression Web, c'est la même chose, nous n'avons pas d'indication de la disponibilité des ces API et des services directement dans l'IDE Web de Microsoft. Côté interopérabilité, là aussi c'est un vaste chantier même si aujourd'hui, le fait que certaines API soient indépendantes de la plate-forme par le protocole et le langage de développement offre des possibilités intéressantes. Toutefois, il n' y a pas d'interopérabilité directe, par exemple, entre un Google Maps et un Virtual Earth (la génération des microformats pourrait résoudre bien des problèmes). Cependant, il est possible d'encapsuler les deux API en créant une couche au-dessus et de faire un rendu avec l'une ou l'autre des technologies. Des accords existent aussi, par exemple entre Live Messenger et Yahoo.

" Côté ressources pour le développeur, tout ce qui est lié à la documentation est disponible sur MSDN. Dans le site français, on dispose de nombreux contenus originaux ou traduits. Le point d'entrée général pour le développeur reste dev.live.com ", précise Christophe. Et à quand un événement Windows Live à l'image des journées développeurs Google ? Aucune réponse, mais le sujet est sur la table de Microsoft, pourquoi pas un événement courant 2009 ?

Ressources

Wrapper LiveNet : http://www.codeplex.com/LiveNet
Portail développeur : dev.live.com

■ **François Tonic**

Références

Windows Live Photo API Références : http://msdn2.microsoft.com/en-us/library/bb932323.aspx
Windows Live Photo WebDAV API : http://msdn2.microsoft.com/en-us/library/cc302759.aspx
User Data Interactive SDK : http://dev.live.com/livedata/sdk/

■ **Antoine Griffard** - *Développeur .NET, Wygwam*

Silverlight 2 bêta 1 : les applications RIA sont en marche

L'événement Microsoft MIX 2008, en mars dernier, fut l'occasion pour l'éditeur de Redmond de lancer une nouvelle version de sa plate-forme de développement RIA : Silverlight 2 bêta 1. Cette version, très attendue par la communauté des développeurs, vous révèle ses nouveautés dans cet article.

Silverlight 2.0, Deep Zoom, des contrôles utilisateurs riches, du dataBinding, des styles et des templates, voilà tout ce que Microsoft vient d'annoncer avec l'arrivée de la nouvelle version de sa plate-forme RIA, Silverlight 2 bêta 1. Mais qu'en est-il exactement ? Quelle est la place de cette technologie dans l'environnement du développeur Web d'aujourd'hui ? Faisons un tour dans ce nouvel univers. Nous allons voir au cours de cet article quels sont les outils pour développer avec Silverlight 2 et aussi, après un bref historique de la plate-forme, ce qu'elle peut nous apporter. Nous verrons notamment comment créer des interfaces riches, comment exploiter des données localisées sur un serveur ou encore profiter de la puissance de XAML.

Pré-requis et outils

Les outils pour développer des applications Silverlight sont à destination de deux populations :

- Les développeurs,
- Les graphistes.

Les outils nécessaires aux développeurs :

Outils	URL de téléchargement
Visual Studio 2008	http://msdn2.microsoft.com/fr-fr/vstudio/default.aspx
Les templates de projets pour Visual Studio 2008	http://www.microsoft.com/downloads/details.aspx?FamilyId=E0BAE58E-9C0B-4090-A1DB-F134D9F095FD&displaylang=en
Le SDK de Silverlight 2	http://www.microsoft.com/downloads/details.aspx?FamilyId=4E03409A-77F3-413F-B108-1243C243C4FE&displaylang=en

Les outils nécessaires aux graphistes :

Outils	URL de téléchargement
Expression Blend 2.5 preview	http://www.microsoft.com/expression/products/download.aspx?key=blend2dot5
Deep Zoom Composer	http://www.microsoft.com/downloads/details.aspx?FamilyID=457b17b7-52bf-4bda-87a3-fa8a4673ffbf&DisplayLang=en

Tous ces outils ont pour objectif une meilleure productivité et une meilleure collaboration entre ces deux populations. Ainsi, un projet Visual studio pourra être chargé directement par Expression Blend. Les développeurs se concentrent alors sur les fonctionnalités et les difficultés techniques quand les graphistes s'attachent à l'ergonomie et au design de l'application. Chacun a la visibilité et la possibilité de modifier le travail des autres.

Vue d'ensemble de la plate-forme

Silverlight 2 bêta 1 remplace la version 1.1 Alpha et fait suite à la version 1.0. La v1 était entièrement basée sur deux langages : Javascript et XAML. La nouvelle version, pour sa part, prend en compte les langages managés du Framework .NET : C# et VB.NET ainsi que le XAML et les langages dynamiques IronPython, IronRuby et Jscript (via la DLR).

Silverlight 2 est, comme son prédécesseur, multi navigateur et multi plate-forme. Il supporte donc MAC OS X et Windows (bientôt Linux via le projet Moonlight) ainsi que les navigateurs suivants : IE, Safari, Opera. Le tableau ci-dessous récapitule les différences entre les deux versions majeures de Silverlight.

Fonctionnalités	Silverlight 1.0	Silverlight 2 bêta 1
2D, Graphics	X	X
Audio, Vidéo et HD	X	X
Animations, transformations	X	X
Affichage de texte	X	X
Saisie de texte		X
Contrôles visuels		X
Gestion du Layout		X
Styles et Templates		X
Data Binding		X
Networking Https	X	X
Networking Socket		X
Support VB.NET et C#		X
Support JavaScript	X	X
LINQ To XML		X
Generics		X
Intégration HTML		X
JSon Serializer	X	X
Local Storage		X
Crypto API		X
Threading		X

On constate, à l'étude de ce tableau, que Microsoft a véritablement pris en compte les demandes des développeurs qui étaient déçus de voir si peu de contrôles utilisateurs dans la première version, la qualifiant alors d'orientée graphiste, avec peu d'outils pour le développeur. La version 2 corrige cela, avec le support de code managé, ainsi que tout ce qui est nécessaire au développement d'applications riches, à savoir : le databinding, les styles et templates, mais aussi un panel de contrôles. Silverlight 2 embarque un nouveau framework, le Framework UI qui est un sous-ensemble de WPF.

Premier pas avec Silverlight : Du Windows dans du Web.

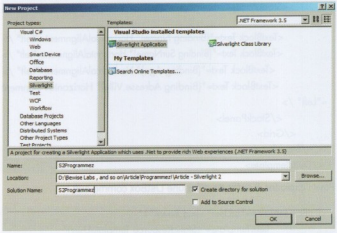
L'objectif, dans la suite de cet article, est de développer une page Silverlight permettant de saisir, visualiser et enregistrer des contacts. Cela nous permettra d'étudier les nouvelles fonctionnalités de cette version. La page définitive est la suivante (Fig.1). Voilà une belle fenêtre de développeur qui ferait hurler le moindre apprenti graphiste (!), mais qui sera notre support.

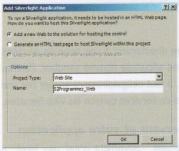
Création d'une application Silverlight 2

Une fois les pré-requis installés, vous pouvez créer une application Silverlight en deux étapes. La première, est la création d'un projet Silver-

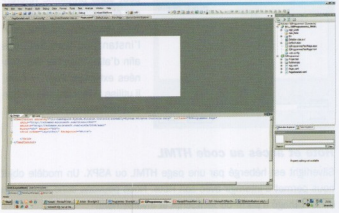
Fig.1

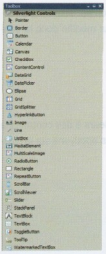
light à partir de Visual Studio 2008. Pour cela, un nouveau type de projet " Silverlight Application " a été installé et vous devez le sélectionner au niveau du nœud Silverlight.





La seconde étape consiste à sélectionner si l'on souhaite créer un hôte Web en même temps que le projet Silverlight ou si l'on veut créer une application Silverlight non hébergée. Une fois ces deux étapes validées, vous voilà devant une page XAML avec, dans le " Solution Explorer ", l'application Silverlight.





Le Layout et les contrôles visuels Silverlight

L'affichage d'une interface Silverlight se fait via le langage déclaratif XAML. Pour nous aider à construire nos interfaces, une palette de contrôles est désormais mise à notre disposition. Celle-ci est riche et contient tout ce qui peut être utilisé pour réaliser une application de gestion comme les boutons, zones de saisies, radio bouton ou encore DataGrid.

Parmi ceux-ci, nous trouvons les contrôles de layout, qui permettent de contenir et positionner d'autres contrôles afin de produire une interface graphique. Ceux à notre disposition sont listés ci-dessous :

Contrôle	Description
StackPanel	Conteneur qui empile des contrôles verticalement et horizontalement
Grid	C'est un tableau qui positionne les éléments en fonction de lignes et de colonnes
Canvas	Conteneur qui permet de positionner des éléments en fonction de ses bords.

Dans notre exemple de gestion de contact, notre interface possède la structure suivante :

```
<Grid x:Name="LayoutRoot" Background="White">
    <Grid.RowDefinitions>
        <RowDefinition Height="10*" />
        <RowDefinition Height="90*" />
    </Grid.RowDefinitions>
    <StackPanel Margin="5" Orientation="Horizontal" Grid.Row="0">
        <TextBox Width="120" Height="20" />
        <TextBox Width="120" Height="20" />
        <TextBox Width="120" Height="20" />
        <TextBox Width="120" Height="20" />
        <Button Width="120" Height="20" Content="Save" Margin=
"2"/>
    </StackPanel>
    <ListBox x:Name="lstUser" Grid.Row="1"
            ItemTemplate="{StaticResource TemplateUser}"
            ItemsSource="{Binding}" />
</Grid>
```

Elle est donc composée d'un tableau avec deux lignes. La partie supérieure affiche les zones de saisie et le bouton " Save " qui nous permettra de saisir un nouveau contact. La partie inférieure contient la liste des contacts. Aujourd'hui, avec la version bêta 1 il n'est malheureusement pas possible de créer une interface " by design " car le designer Cider est inactif. Pour réaliser cela, seul Expression Blend peut vous aider. Une fois notre interface créée, nous voulons l'alimenter de données. Regardons désormais les concepts de DataBinding et de DataTemplate qui sont le cœur de la gestion des données.

Utilisation de données avec Silverlight

Silverlight nous propose un mécanisme de Binding de données très évolué ; nous retrouvons ici l'héritage de WPF. Le databinding est le mécanisme qui consiste à lier des données (via nos objets métiers par

exemple) à un contrôle d'interface. Cette connexion est souvent bidirectionnelle, toute modification du contrôle utilisateur modifie donc la donnée. Il existe deux modes de dataBinding :

- Le databinding simple,
- Le databinding complexe.

DataBinding Simple

Le dataBinding simple consiste à affecter des valeurs à des contrôles à partir d'une classe. Dans le cas suivant nous nous appuyons sur la classe User déclarée comme suit :

```
public class User
{
    public string Name { get; set; }
    public string SurName { get; set; }
    public int Age { get; set; }
    public Adress Adresse { get; set; }
}
```

A partir de cette classe nous pouvons donc réaliser le databinding simple suivant :

```
<Grid x:Name="LayoutRoot" Background="White">
 <Grid.Resources>
   <local:User x:Key="MusaUser" Name="Musashi" SurName="Myamoto"
 Age="31"/>
 </Grid.Resources>

  <StackPanel Orientation="Vertical" DataContext="{StaticResource Musa
 User}">
        <TextBlock Text="{Binding Name}" />
        <TextBlock Text="{Binding SurName}" />
        <TextBlock Text="{Binding Age}" />
    </StackPanel>
 </Grid>
```

Dans cet exemple, nous créons donc un User : <local:User x:Key="MusaUser" Name="Musashi" SurName="Myamoto" Age="31"/> que l'on affecte à la source de donnée nommée " DataContext " de notre conteneur. Ensuite chaque zone de saisie est liée à une propriété de notre user : <TextBlock Text="{Binding Name}" />. La première zone affichera donc le nom, la seconde le nom de famille, etc.

DataBinding Complexe

Le dataBinding complexe est celui qui met en jeu des collections d'objets. C'est le cas de notre liste de contacts. Pour pouvoir afficher chaque élément de la liste, nous nous appuyons sur une autre notion qui est celle du DataTemplate. Le DataTemplate est la représentation visuelle, liaison de données incluse, d'un item d'une collection. Dans notre cas, les informations de la liste apparaissent avec un format : une image et du texte. C'est le DataTemplate de la classe User.

Voici la déclaration de notre DataTemplate :

```
<DataTemplate x:Key="TemplateUser">
  <StackPanel Orientation="Horizontal" Margin="5">
    <Image Source="Inconnu.jpg"/>
    <Grid>
      <Grid.ColumnDefinitions>
        <ColumnDefinition Width="**"/>
        <ColumnDefinition Width="Auto"/>
      </Grid.ColumnDefinitions>
      <StackPanel Orientation="Vertical" Margin="5" Grid.Column
="0">
        <TextBlock Text="Nom :" HorizontalAlignment="Left" Fore
ground=
"White" />
        <TextBlock Text="Prénom : " HorizontalAlignment="Left" Fore
ground="White" />
        <TextBlock Text="Age : " HorizontalAlignment="Left" Fore
ground="White" />
        <TextBlock Text="Adresse : " HorizontalAlignment="Left" Fore
ground="White" />
      </StackPanel>
      <StackPanel Orientation="Vertical" Margin="5" Grid.Column
="1">
        <TextBlock Text="{Binding Name}" HorizontalAlignment="Left" />
        <TextBlock Text="{Binding SurName}" HorizontalAlignment="Left" />
        <TextBlock Text="{Binding Age}" HorizontalAlignment="Left" />
        <TextBlock Text="{Binding Adresse.Ville}" HorizontalAlignment
="Left" />
      </StackPanel>
    </Grid>
  </StackPanel>
</DataTemplate>
```

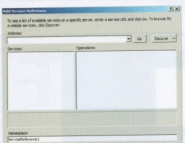
Pour l'exploiter, il suffit de créer une ListBox comme suit :

```
<ListBox x:Name="lstUser" Grid.Row="1"
         ItemTemplate="{StaticResource TemplateUser}"
         ItemsSource="{Binding}" />
```

Accès aux données

Silverlight ne peut pas se connecter directement à une base de données car il est exécuté au sein d'une zone sécurisée nommé sandbox. Pour se connecter à une base de données, il faut obligatoirement passer par une connexion avec un serveur et l'appel de services comme les Web Services.

Communication entre le serveur et votre application



Silverlight n'est pas isolé, il peut sans problème référencer et exploiter des services WCF (basicHttpBinding pour l'instant) ou Web Services afin d'aller chercher des données externes. Pour ce faire, il utilise le mécanisme de Service Reference.

Hôte et accès au code HTML

Silverlight est hébergé par une page HTML ou ASPX. Un modèle objet nous permet d'interagir avec son Hôte. Nous pouvons donc appeler du code managé à partir du JavaScript et inversement.

Le code suivant permet de s'attacher à un évènement JavaScript en code managé :

```
HtmlPage.Document.AttachEvent("onmousewheel",new System.Event
Handler<HtmlEventArgs>(Wheel));
```

On peut alors réagir à l'événement "OnMouseWheel" via la méthode Wheel.

Les styles et les Templates

Voilà deux concepts très enrichissants et qui donnent à Silverlight (et WPF) ce côté applicatif " sexy ", sans aucun doute. Les styles peuvent être comparés au style CSS des applications Web, ce sont des éléments qui permettent de spécifier des valeurs aux propriétés d'un contrôle utilisateur. Ainsi, si nous voulons que toutes les zones de saisies soient espacées de 5 pixels, nous pouvons créer le style suivant :

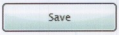
```
<Style TargetType="TextBlock" x:Key="TxtStyle">
    <Setter Property="Margin" Value="2" />
</Style>
```

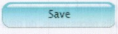
Pour qu'il s'applique, nous devons affecter ce style à chaque zone de saisie comme suit :

```
<TextBox Width="120" Height="25" Style="{StaticResource TxtStyle}"/>
```

Un style par contrôle permet donc de définir des thèmes qui pourront être appliqués à vos contrôles.

Les Templates sont eux plus proches du " Skin ". Ils correspondent au comportement et à la physionomie de vos contrôles. Prenons le cas du bouton de notre application. Si je l'affiche sans modifier son template, voilà le visuel par défaut (ci-contre).





Une fois le Template appliqué, voilà notre magnifique bouton (ci-contre).

Vous m'accorderez le fait que cela est sans conteste bien mieux après qu'avant ! Je vous propose d'analyser un Template en nous appuyant sur le template de la gestion de contact :

```
<ControlTemplate TargetType="ListBoxItem">
    <Grid x:Name="RootElement">
        <Grid.RowDefinitions>
            <RowDefinition Height="20*" />
            <RowDefinition Height="80*" />
        </Grid.RowDefinitions>
    <Rectangle Grid.Row="0" Grid.RowSpan="2" Fill="{StaticResource
BrushSlf}" />
    <Rectangle Grid.Row="0" Fill="{StaticResource BrushWhite}" Radius
X="7" RadiusY="7" />
    <ContentPresenter x:Name="ContentListBoxItem"
            Content="{TemplateBinding Content}"
            ContentTemplate="{TemplateBinding ContentTemplate}"
            FontFamily="{TemplateBinding FontFamily}"
            FontSize="{TemplateBinding FontSize}"
            FontStretch="{TemplateBinding FontStretch}"
            FontStyle="{TemplateBinding FontStyle}"
            FontWeight="{TemplateBinding FontWeight}"
            Foreground="{TemplateBinding Foreground}"
            HorizontalAlignment="Left"
            HorizontalContentAlignment="{TemplateBinding Horizontal
ContentAlignment}"
            Padding="{TemplateBinding Padding}"
            TextAlignment="{TemplateBinding TextAlignment}"
            TextDecorations="{TemplateBinding TextDecorations}"
            TextWrapping="{TemplateBinding TextWrapping}"
            VerticalContentAlignment="{TemplateBinding Vertical
ContentAlignment}"
            Height="{TemplateBinding Height}"
            Grid.Row="0"
            Grid.RowSpan="2" />
    </Grid>
</ControlTemplate>
```

Nous redéfinissions le contenu complet du contrôle et ce nouveau contenu est composé d'une grille, de deux rectangles (un pour le fond orange et un pour le halo blanc) et d'un conteneur de contenu (le ContentPresenter).

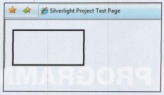
Transformation et animation

Comme dans la version précédente, les transformations et animations sont toujours présentes. Les transformations ont pour objectif de modifier le comportement d'un objet. Il en existe cinq présentées ci-dessous :

Type de transformation	Description	Exemple
TranslateTransform	C'est le déplacement d'un objet d'une position à une autre.	
ScaleTransform	C'est le grossissement, rétrécissement d'un objet.	
RotateTransform	C'est une rotation d'un objet.	
SkewTransform	C'est une transformation qui met en oblique un objet.	
MatrixTransform	Chaque élément 2D peut être représenté par une matrice. Nous pouvons alors créer nous-même des transformations en les appliquant directement sur la matrice de l'objet.	

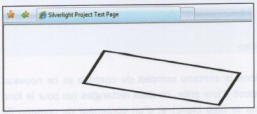
L'exemple de code suivant met en œuvre les quatre premières transformations en s'appuyant sur un rectangle. Avant toute transformation, le rectangle blanc avec un cadre noir d'une longueur de 150 pixels sur 75 est positionné sur le coin gauche de l'écran.

```
<Rectangle x:Name="rectLayout" Width="150" Height="75" Canvas.
Top="5" Canvas.Left="5" Stroke="Black" StrokeThickness="3">
```



Ensuite, nous lui appliquons des transformations : une translation, une rotation pour le faire tourner d'un angle de 20°, un agrandissement de 2 et enfin un Skew de 2. Le résultat est le suivant (p. 28)

```
<Rectangle.RenderTransform>
  <TransformGroup>
    <TranslateTransform X="100"/>
    <RotateTransform Angle="20"/>
    <ScaleTransform ScaleX="2"/>
    <SkewTransform AngleX="5"/>
  </TransformGroup>
</Rectangle.RenderTransform>
```



La gestion des animations :

XAML nous fournit un support d'animation, autrement dit un modèle objet (n'oublions pas qu'écrire du XAML c'est coder !) destiné à animer et interagir dynamiquement avec les objets instanciés. Par exemple, faire tourner en boucle notre rectangle. Le " Storyboard " permet de définir, créer et paramétrer une animation. Pour cela, nous avons besoin d'un déclencheur appelé " trigger " qui lancera l'animation. Dans l'exemple ci-dessous, nous animons la propriété " Opacity " du rectangle créé précédemment.

```
<EventTrigger RoutedEvent="Rectangle.Loaded">
    <BeginStoryboard>
      <Storyboard>
        <DoubleAnimation
          Storyboard.TargetName="rectLayout"
          Storyboard.TargetProperty="Opacity"
          From="1.0" To="0.0" Duration="0:0:5" AutoReverse="True"
RepeatBehavior="Forever" />
      </Storyboard>
    </BeginStoryboard>
  </EventTrigger>
</Rectangle.Triggers>
```

Silverlight Deep Zoom

Silverlight Deep Zoom est une technologie permettant de faire du Multiscale d'image. C'est-à-dire d'afficher la même image à des résolutions différentes afin d'avoir un effet de zoom. Pour réaliser cela il faut composer ses images avec l'outil Deep Zoom composer :

Ensuite, Silverlight fournit un contrôle spécifique pour exploiter le produit de Deep Zoom composer :

```
<MultiScaleImage x:Name="DeepZoomImage"   Source="Images/info
.bin"/>
```

Conclusion

Silverlight 2 bêta 1 est donc bien le produit conforme à nos attentes depuis la version 1.0. Il nous fournit tous les outils nécessaires, du databinding aux contrôles utilisateurs évolués en passant par les styles et les templates, pour créer des applications riches. L'API fournie avec Silverlight 2 est très riche, c'est réellement un nouvel ensemble à appréhender.

■ **Sacha LEROUX**
http://blogs.codes-sources.com/Musashi/
Sacha.leroux@bewise.fr - Bewise

L'INFORMATION PERMANENTE

✓ **Abonnez-vous**
au format NUMERIQUE (PDF)
Tarif unique pour

LE MONDE ENTIER
30 €
(abonnement exclusivement en ligne
www.programmez.com)

Programmez!
LE MAGAZINE DES LOISIRS INFORMATIQUES

✓ **3 NUMEROS GRATUITS**
Faites des économies, abonnez-vous !
Exemple : en France, vous gagnez 20 euros, soit plus de 3 numéros !

✓ **ETUDIANTS**
Tarif spécial 39€
(France métropolitaine exclusivement)

Nouveau PROGRAMMEZ HORS SÉRIE .NET
Abonnement 1 an : 5 numéros : 20€
COUPON D'ABONNEMENT page 51 de ce numéro ou www.programmez.com

Médiation de protocoles REST/SOAP avec Workflow Foundation

Scénario d'interopérabilité en environnement .Net 3.5 et Java SE 6

Les architectures SOA nécessitent la mise en œuvre de médiateurs pour adapter :

1. Les messages échangés entre les points d'exposition de services (endpoints)
2. Les protocoles utilisés par chacun des endpoints.

Un médiateur répond à ces deux natures d'adaptation (Adaptation de messages et Adaptation de protocoles) sans intervenir sur le consommateur ni le producteur.

Figure 1

Pattern

L'adaptation de messages s'apparente à une transformation de données, tandis que l'adaptation de protocoles nécessite la mise en œuvre de connecteurs (ou adaptateurs dans le vocabulaire BizTalk par exemple).

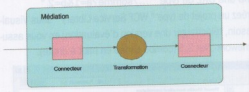


Figure 2

Du pattern à l'implémentation

Il existe plusieurs technologies de médiation en environnement Microsoft. Citons deux implémentations mises en œuvre au sein de l'application consommatrice, c'est-à-dire sans avoir recours à un tiers indépendant chargé de réaliser l'adaptation.

- Utiliser un binding WCF et l'étendre avec des transformations XSLT par exemple
- Utiliser le WCF LOB Adapter SDK pour exposer un service existant au travers de WCF

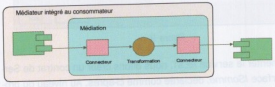


Figure 3

Si l'on préfère isoler la tâche d'adaptation au sein d'un médiateur, il est possible d'utiliser les technologies suivantes avec des capacités croissantes en termes de possibilités d'adaptation et de montée en charge :

- *WorkflowServiceHost* (illustré dans le scénario qui suit) disponible depuis .Net 3.5
- *Managed Services Engine*, technologie de virtualisation de Services disponible en opensource sur CodePlex
- *BizTalk ESB Guidance* pour réaliser les tâches de médiation spécifiques aux ESB (services de transformation & routage), et disponible en opensource sur CodePlex

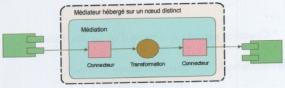


Figure 4

Scénario d'interopérabilité en environnement .Net 3.5 et Java SE 6

Illustrons nos propos au travers du scénario suivant :

- 1 serveur de calcul en Java SE 6
  - Expose un Service Web " Somme " en HTTP/SOAP implémenté JAX-WS 2.0
- 1 consommateur REST
  - Demande le calcul d'une somme à partir d'un browser Web http://serveurdecalcul/somme?a=1&b=2
- 1 médiateur avec les technologies du framework .Net 3.5
  - Windows Communication Foundation (WCF) pour les aspects Connecteurs
  - Windows Workflow Foundation (WF) pour la partie Transformation et conteneur de la médiation.

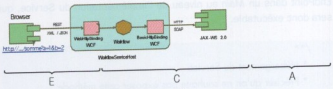


Figure 5

Nous implémenterons la solution en commençant par le service Java " SommeService " puis un service REST avec WCF, et enfin le ServeurDeCalcul avec WF. Le code complet du scénario est fourni sur le blog de l'auteur (http://blog.sfartz.com).

A. Implémentation du Service Somme Java

Depuis Java 6, il est possible de créer un Service Web en Java sans mettre en œuvre d'EJB ni de serveur d'application. A partir d'un simple runtime Java Standard Edition, créer une classe avec l'attribut @WebService proposé par l'API JAX-WS, suffit à être exposé sous la forme d'un Service Web.

```
import javax.jws.*;
import javax.jws.soap.SOAPBinding;
import javax.xml.ws.Endpoint;

/**
 * Service Somme
```

```
 *
 * JAX-WS publie en SOAP Document Literal par défaut
 * Doc : https://jax-ws.dev.java.net/jax-ws-ea3/docs/annotations.html
 *
 */
@WebService(        targetNamespace = "http://jaxwscalcul.org",
                    name="ServiceSomme",
                    serviceName="Somme")
@SOAPBinding(       style=SOAPBinding.Style.DOCUMENT,
                    use=SOAPBinding.Use.LITERAL)
public class ServiceSommeImpl {

        /**
         * Operation exposée sous forme d'un Service Web
         */
        @WebMethod(         operationName="Sommer",
                            action="urn:somme")
        @WebResult( name="result")
        public int getSomme(@WebParam(name="num1") int a,
                            @WebParam(name="num2")int b) {
                return a + b;
        }
}
```

Figure 6

Remarque : Dans le code ci-dessus, nous avons explicité des attributs superflus et à titre d'apprentissage, parce que les valeurs par défaut nous auraient convenu dans le cadre de notre scenario.

Il reste à héberger un point d'exposition. Ceci est réalisé avec la classe *EndPoint* de JAX-WS, qui permet notamment de spécifier l'URL d'écoute du service. Pour faire simple, nous avons placé le code de création du EndPoint dans un *Main* au niveau de l'implémentation du Service, qui sera donc exécutable.

```
    /**
     * Hébergement du Service
     * Préciser qu'on ne souhaite pas exposer cette méthode
     */
    @WebMethod(exclude=true)
    public static void main(String[] args) {

        String myURL = "http://localhost:8383/JAXWS/";
        System.out.println("Publishing JAX-WS ServiceSomme at " + myURL);

        // L'appel est bloquant
        Endpoint.publish(myURL, new ServiceSommeImpl ());
    }
```

Figure 7

Remarque : Il est indispensable de préciser que nous ne souhaitons pas exposer la méthode Main *car JAX-WS considère par défaut que toutes les méthodes de la classe qui possède l'attribut @WebService sont exposées, que l'attribut @WebMethod soit présent ou non. WCF a un fonctionnement inverse, seules les méthodes préfixées avec un attribut* OperationContract *sont exposées.*

A la compilation du code source, succède une phase de génération de la structure d'hébergement du point d'exposition. Cette génération est réalisée par l'outil " wsgen " livré avec le JDK.

```
> set JAVA_HOME="C:\Program Files\Java\jdk1.6.0_03\"
> set PATH=%ATH%;%JAVA_HOME%\bin

> javac ServiceSommeImpl.java
> move ServiceSommeImpl.class DemoMediation

> wsgen -cp . DemoMediation.ServiceSommeImpl

> java DemoMediation.ServiceSommeImpl
```

Figure 8

A l'exécution, le Service Web " Somme " est directement appelable en SOAP et son contrat WSDL est généré par JAX-WS à l'adresse http://localhost:8383/JAXWS/ServiceSomme?wsdl.

B. Implémentation du dialogue REST

La technologie WCF, apparue avec la version 3.0 du framework .Net permet de créer des services indépendamment des protocoles de communication, qui sont précisés au travers de binding. Le paramétrage du ou des binding (il est possible de créer de multiples points d'exposition) peut être externalisé dans un fichier de configuration XML.

Pour les échanges entre le *ServeurDeCalcul* et la partie Java, nous utiliserons dans la section suivante un binding de type SOAP qui implémente le profile WS-I Basic Profile : *basicHttpBinding*.

Nouveauté du framework .Net 3.5, WCF supporte les échanges de type REST au travers du WebBinding. Dans notre scénario, il s'agit d'invoquer une somme via une URL du type " .../somme?a=1&b=2 "

Pour commencer, créez un projet de type " WCF Service Library " sous Visual Studio 2008 ((au besoin, télécharger une version d'évaluation) en vous assurant d'utiliser le framework .Net 3.5 (combo située en haut à droite).

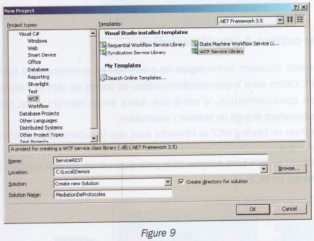


Figure 9

Afin d'implémenter le service REST, nous allons créer un contrat de Service via l'interface *ISomme*, déclarée comme ci-après. Au niveau du projet ServiceREST, il est nécessaire d'ajouter la référence à la librairie REST qui vient avec le framework .Net, à savoir *System.ServiceModel.Web*.

```
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
```

```
namespace ServiceREST
{
   /*
    * Contrat du Service de Calcul
    * Implémentation WCF de type REST
    */
   [ServiceContract(Namespace="http://serveurdecalcul.org")]
   public interface ISomme
   {
      [OperationContract(Name="somme")]
      [WebGet(UriTemplate = "somme?a={valeur1}&b={valeur2}",
            ResponseFormat=WebMessageFormat.Xml,
            BodyStyle = WebMessageBodyStyle.WrappedResponse)]
      [return: MessageParameter(Name = "result")]
      int getSomme(int valeur1, int valeur2);
   }

}
```

Figure 10

A des fins de tests, nous créerons une classe d'implémentation simplissime :

```
using System.ServiceModel.Web;

namespace ServiceREST
{
   public class ServiceSommeImpl : ISomme
   {
      public int getSomme(int valeur1, int valeur2)
      {
         return valeur1 + valeur2;
      }
   }
}
```

Figure 11

Enfin, il nous reste à insérer un *webHttpBinding* et à l'associer au service *ISomme*, au niveau du fichier de configuration *App.config*.

```
<system.serviceModel>
   <services>
    <service behaviorConfiguration="DebugBehavior"
            name="ServiceREST.ServiceSommeImpl">
      <endpoint behaviorConfiguration="WebHttp"
               binding="webHttpBinding"
               name="REST"
               contract="ServiceREST.ISomme" />
      <host>
        <baseAddresses>
          <add baseAddress="http://localhost/" />
        </baseAddresses>
      </host>
    </service>
   </services>
   <behaviors>
    <endpointBehaviors>
      <behavior name="WebHttp" >
        <webHttp />
      </behavior>
    </endpointBehaviors>
    <serviceBehaviors>
      <behavior name="DebugBehavior">
        <serviceMetadata httpGetEnabled="true" />
        <serviceDebug includeExceptionDetailInFaults="true" />
      </behavior>
    </serviceBehaviors
```

Figure 12

Dès lors, la version 2008 de Visual Studio nous permet de tester le service sans avoir besoin de l'héberger au sein d'un *ServiceHost* (il s'agit précisément du WCF Service Host). Il nous suffit donc de faire un clic droit sur le projet et *Debug => Start new instance*, et un hôte local à VisualStudio est lancé. Le WCF Service Host propose une interface qui dresse la liste des services hébergés. Notre service est donc bien démarré.

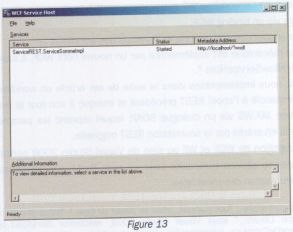


Figure 13

Remarque : ça c'était la bonne nouvelle. La mauvaise c'est que Visual Studio 2008 propose aussi une interface permettant de tester ses Services. Ce " WCF Test Client " n'est par contre pas accessible pour notre scénario dans la mesure où le protocole REST n'impose pas (ni ne propose d'ailleurs dans sa version courante) de décrire les services. Pas de meta-données, pas d'IHM de tests (... par opposition à un service de type SOAP / basicHttpBinding qui expose un contrat WSDL, et permet donc de tester ses services et visualiser les messages d'entrée et de sortie).

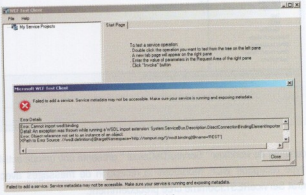


Figure 14

Afin de tester notre service, appelons depuis un navigateur l'URL http://localhost/somme?a=1&b=2, nous obtenons le résultat suivant :

```
<sommeResponse xmlns="http://serveurdecalcul.org">
<result>3</result>
```

```
</sommeResponse>
```

Remarque : Pour obtenir une réponse au format JSON de type {"result":3} plutôt qu' XML, paramétrer l'attribut WebGet *comme suit :*

```
[WebGet(UriTemplate = "somme?a={valeur1}&b={valeur2}",
        ResponseFormat=WebMessageFormat.Json,
        BodyStyle=WebMessageBodyStyle.WrappedResponse)]
```

Occupons nous maintenant de substituer l'implémentation Java à notre implémentation .Net via une composition de services exécutée par le framework Workflow Foundation.

C. Implémentation du médiateur Serveur De Calcul

Si WCF et WF sont apparus avec .Net 3.0, ce n'est qu'à partir de la version 3.5 que nous disposons d'un lien natif entre ces deux frameworks, de façon à pouvoir :

- instancier un workflow WF automatiquement à la réception d'un message via un binding WCF,
- et invoquer des services via WCF depuis des workflows WF

Cette mécanique est implémentée par un nouvel hôte WCF, à savoir le " WorkflowServiceHost ".

Ainsi, nous implémentons dans la suite de cet article un workflow qui est instancié à l'appel REST précédent et invoque à son tour le service Somme JAX-WS via un dialogue SOAP, lequel reprend les paramètres fournis en entrée par la soumission REST originelle.

L'intégration de WCF et WF au sein de Visual Studio 2008 permet de réaliser ce service composite avec un code limité aux opérations de transformation des messages.

Mode opératoire : Tout d'abord, créer un projet de type " Sequential Workflow Service Library " sous Visual Studio 2008, renommer Workflow1.cs en ServeurDeCalcul.cs, *supprimer les variables* InputValue *et* ReturnValue *du code de* ServeurDeCalcul.cs *et ajouter une référence au projet ServiceREST créé précédemment.*

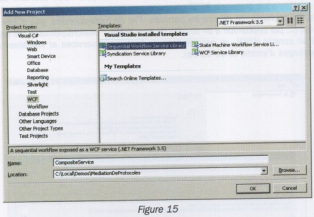


Figure 15

Le designer de workflow " ServeurDeCalcul " présente une unique activité, de type réception WCF. Modifions la réception de messages avec le contrat ServiceREST.ISomme créé précédemment.

Mode opératoire : Cliquer sur l'activité " receiveActivity1 ", la renommer en " receptionREST " en modifiant la propriété " name ", puis sélectionner la propriété ServiceOperationInfo. *Ouvrir l'assistant et choisir " Import... " en haut à droite. Au sein du projet ServiceREST, choisir l'interface " Service REST.ISomme " puis sélectionner l'opération " somme ", et fermer l'assistant. Nous pouvons maintenant supprimer l'interface* IWorkflow1.cs.

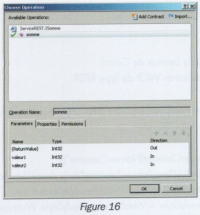


Figure 16

Réutilisons la configuration WCF créée précédemment.

Mode opératoire : Ecraser le fichier App.config par copier/coller depuis le projet ServiceREST, l'ouvrir en mode texte et remplacer le " Service name " par le Workflow en cours de construction, pour obtenir :

```
<service ... name="CompositeService.ServeurDeCalcul"> ...
```

Complétons le workflow *ServeurDeCalcul* par l'invocation du Service Web JAX-WS.

Mode opératoire :

- Exécuter le service " SommeService " créé en première partie et vérifier que le contrat WSDL est bien disponible dans notre exemple à l'adresse http://localhost:8383/JAXWS/ServiceSomme?wsdl. Ajouter une référence au service via l'assistant Visual Studio : Clic droit " Add Service Reference " pour utiliser WCF.

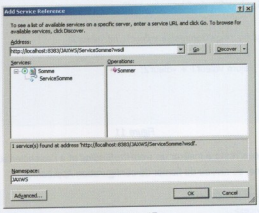


Figure 17

- Le fichier App.config est complété automatiquement avec l'ajout d'un binding basicHttpBinding *correspondant aux échanges SOAP sur HTTP. En ouvrant le fichier App.config avec l'utilitaire* svcConfigEditor, *on constate que le endpoint d'invocation du service JAXWS se nomme " ServiceSommePort " (à conserver dans un coin dans son esprit...)*

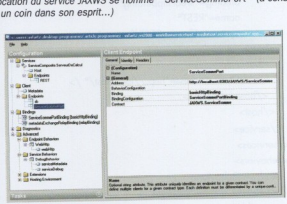


Figure 18

- Puis, ajouter une activité " sendActivity " juste après l'activité de réception REST, à partir de la Toolbox Workflow Foundation v3.5, et renommer l'activité en appelServiceSomme. Bonus : Afin d'illustrer des possibilités complémentaires de la technologie WF, nous avons placé dans l'exemple fourni en ligne une activité de logs en parallèle de l'invocation de la partie JAXWS

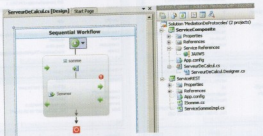


Figure 19

- Cliquer sur l'activité d'envoi " appelSommeService " et sélectionner la propriété ServiceOperationInfo, et via l'assistant, importer le contrat de service JAXWS.ServiceSomme. Remarquer que les paramètres entrants et sortants ne sont pas des entiers mais des messages... Fermer l'assistant. Le message d'erreur " Activity 'appelSommeService' does not have a service operation specified. " disparaît.

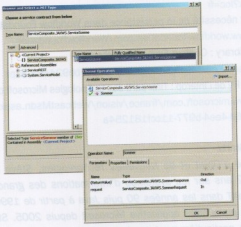


Figure 20

Il nous reste à connecter les paramètres d'entrée de l'appel REST, aux paramètres d'invocation du service JAXWS, et réciproquement, à lier la valeur de retour du service JAXWS à la valeur de retour du *ServeurDeCalcul* attendu par l'appel REST. Il s'agit de la fonction de transformation présentée en introduction.

Mode opératoire

- Cliquer sur l'activité " appelSommeService ", sélectionner la propriété " request ", et invoquer l'assistant situé en bout de ligne [...]. Sélectionner l'onglet " Bind to a new member " afin de créer une variable au sein de l'instance de workflow ServeurDeCalcul.

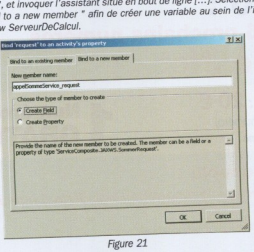


Figure 21

- Répéter la même opération pour la propriété " returnValue ". Si vous accédez au code du Workflow ServeurDeCalcul.cs, vous constatez que 2 variables d'instances ont bien été créées et initialisées avec des valeurs par défaut.

- Sélectionner l'activité " receptionREST ", et dans le panneau des propriétés sélectionner successivement " value1 ", " value2 " puis " returnValue " et au travers des assistants situés en bout de ligne [...], rattacher les champs des variables appelSommeService_request.num1, appelSommeService_request.num2 et appelSommeService__ReturnValue.result

- Sélectionner la propriété ChannelToken de l'activité appelSommeService, lui donner la valeur JAVA, et pour le sous-champ EndpointName la valeur " ServiceSommePort " spécifiée dans le fichier App.config (cf valeur précédente mémorisée).

> Attention : un message d'erreur erroné apparaît de façon répétitive sur l'activité appelSommeService " Property '{ReturnValue}' has invalid value. Field type 'ServiceComposite.JAXWS.SommerResponse' does not match with the expected type 'ServiceComposite.JAXWS.SommerResponse'". Ignorer ce message qui correspond à une anomalie de gestion de cache. La compilation fonctionne parfaitement, même si ce message est plutôt perturbant.

Désormais, le Workflow *ServeurDeCalcul* est connecté en entrée à l'appel REST et invoque à son tour le service JAXWS " ServiceSomme ".

Compilation, Exécution...

Notons que nous n'avons pas besoin d'héberger le Workflow dans un WorkflowServiceHost. De la même façon que dans notre implémentation REST avec WCF, Visual Studio 2008 lance le runtime d'hébergement WCF Service Host.

Ça c'était la bonne nouvelle. La mauvaise nouvelle c'est que nous obtenons l'erreur suivante à l'exécution :

> System.InvalidOperationException: Service behavior WorkflowServiceBehavior requires that the binding associated with endpoint REST listening on http://localhost/ supports the context protocol, because the contract associated with this endpoint may require a session. Currently configured binding for this endpoint does not support the context protocol. Please modify the binding to add support for the context protocol or modify the SessionMode on the contract to NotAllowed.

Plutôt explicite comme message, non ? WorkflowServiceHost ne sait travailler qu'avec des binding qui permettent de gérer un contexte. A cet effet, trois nouveaux binding ont été introduits en .Net 3.5 : *basicHttpContextBinding*, *netTcpContextBinding*, et *wsHttpContextBinding*, mais point de *webHttpContextBinding* ! Qu'à cela ne tienne, voici l'occasion de construire un binding WCF en spécifique " customBinding " qui correspondra à un " webHttpBinding" avec gestion de contexte.

Mode opératoire : Créer un customBinding, comprend un transport http, une gestion de contexte via Cookie et sans sécurité (Il est possible de sécuriser le contexte en mettant en place un transport sécurisé), et un encodage de type web. Modifier le binding du endpoint REST en customBinding avec la configuration MyWebHttpContextBinding.

```
<customBinding>
  <binding name="MyWebHttpContextBinding">
    <webMessageEncoding />
  <context contextExchangeMechanism="HttpCookie"
        protectionLevel="None" />
    <httpTransport manualAddressing="true" />
  </binding>
</customBinding>
```

Lorsque nous lançons à nouveau le *ServeurDeCalcul*, le WCF Service Host charge correctement le Workflow, même s'il ne parvient pas à lire les meta-données associées (toujours pour la non disponibilité d'une description WSDL des contrats de services). En appelant http://localhost/somme?a=1&b=2, nous obtenons le résultat précédent :

```
<sommeResponse xmlns="http://serveurdecalcul.org">
        <result>3</result>
</sommeResponse>
```

Le médiateur est opérationnel.

Remarque : Si nous invoquons l'URL http://localhost/somme?a=1&b=2 une seconde fois, nous obtenons l'erreur suivante, liée au fait que notre hôte du runtime ne propose pas de technologie de persistance pour le runtime Workflow Foundation.

```
The server encountered an error processing the request. The exception
message is 'The workflow hosting environment does not have a persisten-
ce service as required by an operation on the workflow instance
"5da5e205-19e6-45fd-952e-354c35344de5e".'. See server logs for more
details. The exception stack trace is:

at System.Workflow.Runtime.WorkflowRuntime.InitializeExecutor(Guid instan
ceId, CreationContext context, WorkflowExecutor executor,
WorkflowInstance workflowInstance) at System.Workflow.Runtime.Workflow
Runtime.Load(Guid key, CreationContext context, WorkflowInstance
workflowInstance) at System.Workflow.Runtime.WorkflowRuntime.GetWorkflow
(Guid instanceId) at System.ServiceModel.Dispatcher.WorkflowDurable
Instance.GetWorkflowInstance(Boolean canCreateInstance) at
System.ServiceModel.Dispatcher.WorkflowOperationAsyncResult..ctor(Work
flowOperationInvoker workflowOperationInvoker, WorkflowDurableInstance
workflowDurableInstance, Object[] inputs, AsyncCallback callback, Object
state, Int64 time) at System.ServiceModel.Dispatcher.WorkflowOperation
Invoker.InvokeBegin(Object instance, Object[] inputs, AsyncCallback callback,
Object state) at
System.ServiceModel.Dispatcher.DispatchOperationRuntime.InvokeBegin(
MessageRpc& rpc) at ...
```

En effet, le premier appel au Workflow ServeurDeCalcul a positionné un cookie de gestion de contexte " WscContext ". Aussi, au second appel, parce qu'un cookie de contexte " WscContext " est positionné dans le header HTTP, le runtime Workflow utilise le service de persistance pour retrouver l'instance de Workflow associé.

Afin de corriger ce dysfonctionnement, mettre en place un mécanisme de persistance.

1er appel
Affectation du cookie sur la réponse

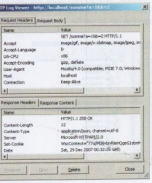


Figure 22

2e appel
Présence du cookie sur la requête

Figure 23

Ressources

- Programmation HTTP avec WCF et .Net 3.5
http://msdn.microsoft.com/msdnmag/issues/08/01/WCFinOrcas/default.aspx?loc=fr
- Utilité et nécessité de la persistance dans Workflow Foundation
http://www.workflow-foundation.com/WorkflowFoundation/
- MSDN Library : Context Exchange Protocol
http://msdn2.microsoft.com/en-us/library/bb924468.aspx
- WebCast RV de l'interop : Tirer parti des technologies Microsoft depuis Java
http://www.microsoft.com/france/vision/WebcastMsdn.aspx?EID=fece86cc-056d-4ee4-b977-11ccf181254a

Stève Sfartz

Passionné d'Objet et de Nouvelles Technologies, Stève intervient depuis 12 ans sur les systèmes d'informations des grandes entreprises. NeXT dans les années 90 puis Java à partir de 1996 et enfin .Net et la plate-forme applicative Microsoft depuis 2005. Stève intervient plus particulièrement sur les problématiques SOA (Standards SOAP et WS-, BizTalk, WCF, WF, MSMQ) et l'interopérabilité des technologies Microsoft .Net avec les plates-formes Java.*

Web : http://blogs.msdn.com/ssfartz et http://blog.sfartz.com

NOUVEAU :
Recevez les 5 prochains numéros
Hors Série .Net

- *5 numéros par an :* **20 €**
(au lieu de 25 € tarif au numéro)
Tarif France métropolitaine

1 Numéro GRATUIT

- **Abonnement couplé :**
Le magazine 11 numéros + le magazine Hors Série 5 numéros
63 € seulement (Tarif France métropolitaine)

Abonnez-vous sur www.programmez.com et coupon page 51

Requêter avec LINQ : tout savoir ou presque !

LINQ (Language-Integrated Query) est une série d'extensions qui ont été ajoutées au Framework .NET 3.5 pour permettre aux développeurs d'effectuer des requêtes sur divers types de collections de données en utilisant une syntaxe unique très proche des langages de requêtes ensemblistes tels que SQL.

Dans sa version de base, LINQ est disponible uniquement pour les types de collections de données suivantes : base de données SQL (LINQ-to-SQL), données XML (LINQ-to-XML), DataSets (LINQ-to-DataSets), liste d'objets (LINQ-to-Objects), Entities (LINQ-to-Entities) mais le modèle d'architecture sur lequel il repose permet de pouvoir l'étendre facilement à d'autres types de collections. Le but de cet article est justement de vous présenter les grandes lignes de la création d'un provider LINQ pour un type quelconque de collection de données. Cette possibilité d'extensions de LINQ, est une opportunité intéressante d'offrir aux développeurs, une syntaxe unique d'interrogation de divers types de collections de données. Le développeur spécifie sa requête en utilisant la syntaxe LINQ et le provider se charge de l'analyser, de la convertir dans le langage associé à la collection de données, de l'exécuter et de restituer le résultat dans une structure typée directement manipulable par le développeur. La technologie et la grammaire des requêtes associées à la collection de données sont ainsi masquées au développeur.

Pour illustrer la démarche, nous nous appuierons sur un exemple basique de provider LINQ-to-LDAP que nous avons implémenté. Même si l'exemple reste très simpliste (nous nous limitons aux opérateurs LINQ *Select* et *Where*), il reste néanmoins utile pour vous donner une idée de la démarche à adopter, des fonctionnalités importantes de LINQ et des patterns utiles qui vous faciliteront l'implémentation du provider.

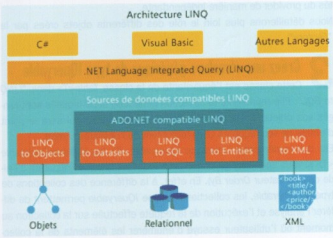
Nous supposons que vous maîtrisez au moins, les principales fonctionnalités du Framework 3.5 sur lesquelles s'appuie LINQ, à savoir : les expressions Lambda, les méthodes d'extension, les types anonymes, les variables locales typées de façon implicite, les initialiseurs d'objets et les expressions de requêtes.

1 Identifiez l'API de communication avec la source de données distante

Il s'agit d'identifier et de choisir l'API que vous allez utiliser pour communiquer avec la source de données que vous souhaitez interroger. Ce sera par exemple *ADO.NET* pour une communication avec des bases de données ou *DirectoryServices* pour une communication avec des annuaires LDAP comme c'est le cas dans notre exemple.

Vous pouvez choisir plusieurs API et implémenter votre provider de sorte qu'il puisse choisir l'API à utiliser suivant différents critères. On peut, par exemple, imaginer un provider qui offre deux modes différents de communication, un mode distribué où on utilise des Web Services pour communiquer avec la source de données distante et un mode local qui utilise une API qui communique directement avec la source de données. C'est le cas du provider " *LINQ-to-Sharepoint* " proposé par Bart de Smet sous CodePlex.

Les critères à prendre en compte dans le choix de l'API sont : les performances de l'API, le type de communication avec la source de données (accès distant ou local), les facilités d'intégration au langage LINQ.

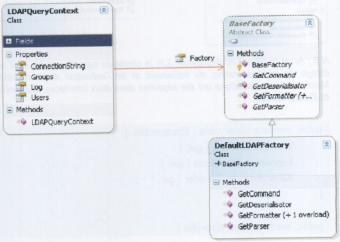


2 Créez une classe centrale pour gérer le contexte d'exécution de vos requêtes

Cette classe est équivalente à la classe *DataContext* dans " *LINQ to SQL* ". C'est le point d'entrée de votre provider. Elle expose les différentes collections de données de la source de données distante et gère le cycle de vie des objets récupérés par le provider. Ainsi si vous souhaitez gérer les mises à jour des données récupérées par votre provider, c'est dans cette classe que vous allez intercepter toutes les différentes actions effectuées sur les objets.

Les paramètres généraux du contexte d'exécution des requêtes tels que l'adresse de connexion à la source de données sous-jacente, la sortie des traces (Log) du fournisseur, etc. sont aussi stockés dans cette classe.

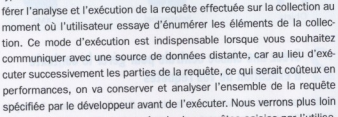
Dans notre exemple, cette classe correspond à *LDAPQueryContext* dont la structure est donnée sur le diagramme ci-dessous.



Comme vous pouvez le remarquer, nous avons aussi implémenté une *"Factory"* d'objets qui est instanciée par la classe *LDAPQueryContext*. En effet, pour pouvoir découpler les différentes opérations réalisées par le provider telles que l'interprétation d'arbre d'expression, la translation de requête, la communication avec la source de données distante, la désérialisation des résultats, nous avons séparé ces opérations dans des classes qui sont instanciées uniquement par la *" Factory "*. Le fait que l'on ne dépende que d'interfaces ou de classes abstraites permettra de réutiliser, maintenir ou faire évoluer les différentes fonctionnalités du provider de manière indépendante.
Nous détaillerons plus loin le rôle des différents objets crées par la *"Factory"*.

3 Créez une structure générique de type IQueryable

Cette classe est l'équivalente de la classe *Table<T>* dans *"LINQ to SQL"*. Elle représente la collection générique dans laquelle sont rangés vos objets. C'est une instance de cette classe qui sera le point d'entrée de vos requêtes LINQ (argument de l'opérateur *From*).
Elle doit implémenter l'interface *IQueryable <T>* (ou *IOrderedQueryable <T>* si vous souhaitez gérer dans vos requêtes le tri des résultats à l'aide de l'opérateur *Order By*). En effet, à la différence des collections de type *IEnumerable*, les collections de type *IQueryable* permettent de différer l'analyse et l'exécution de la requête effectuée sur la collection au moment où l'utilisateur essaye d'énumérer les éléments de la collection. Ce mode d'exécution est indispensable lorsque vous souhaitez communiquer avec une source de données distante, car au lieu d'exécuter successivement les parties de la requête, ce qui serait coûteux en performances, on va conserver et analyser l'ensemble de la requête spécifiée par le développeur avant de l'exécuter. Nous verrons plus loin comment LINQ conserve en mémoire les requêtes saisies par l'utilisateur.
Dans notre exemple, cette structure générique correspond à la classe *DirectoryEntries<T>*. *Cette* classe hérite d'une classe abstraite *BaseQueryableClass<T>* qui offre une implémentation minimale de l'interface IQueryable<T>, réutilisable par d'autres types de providers. La structure de ces classes est présentée dans le diagramme ci-dessous :



N.B : Avant la Bêta 2 du Framework 3.5, la classe de type IQueryable était en charge de la récupération, du traitement et de l'exécution de la requête. Maintenant ces opérations ont été séparées dans deux interfaces IQueryable et IQueryProvider.

```
public interface IQueryable : IEnumerable {
        Type ElementType { get; }
        Expression Expression { get; }
        IQueryProvider Provider { get; }
}

public interface IQueryProvider {
        IQueryable CreateQuery(Expression expression);
        IQueryable<TElement> CreateQuery<TElement>(Expression expression);
        object Execute(Expression expression);
        TResult Execute<TResult>(Expression expression);
}
```

La classe de type IQueryable a pour seul rôle de conserver l'arbre d'expressions correspondant à la requête à exécuter sur la collection, le type des éléments de la collection et une instance sur une classe de type IQueryProvider, qui elle est en charge du traitement et de l'exécution de la requête. A noter que cette exécution n'est effectuée qu'au moment où l'on essaye d'énumérer les éléments de la collection via sa méthode GetEnumerator().

4 Créez vos entités d'objets .NET

Il s'agit de définir des classes qui vont représenter en modèle objet les données de la source de données que vous souhaitez interroger. Ces classes jouent donc un rôle d'interface entre le modèle objet et le modèle de la source de données distante (on parle souvent de " mapping " Objet/Relationnel lorsque la source de données distante est un SGBDR). Ainsi, on pourra ajouter aux entités définies des métadonnées (via des attributs personnalisés par exemple), qui vont être utilisées lors de la construction de la requête et de la transformation des résultats en objets .NET.
Voici un extrait de l'entité User de notre exemple qui correspond à " la classe d'objet " de type user dans le monde LDAP.

```
namespace CustomQueryProvider.LDAP.Entities
{
    [ObjectClass(Name ="user")]
    public class User
    {
        public User()
        { }

        [LDAPAttribute(Name  = "sn")]
        public string Surname {get;set;}

        [LDAPAttribute(Name  = "gn")]
        public string FirstName { get; set; }

        [LDAPAttribute(Name  = "Description")]
        public string Description { get; set; }

        [LDAPAttribute(Name = "LogonCount")]
        public int LogonCount { get; set; }

        [LDAPAttribute(Name = "Dn")]
        public string DistinguishName { get; set; }

        [LDAPAttribute(Name  = "memberOf")]
        public string[] Groups { get; set; }

    }
}
```

Nous nous sommes limités à quelques propriétés utiles qui permettent d'illustrer la structure des entités.
La classe User et ses propriétés sont décorées par des attributs personnalisés qui permettent de faire le " mapping " entre le modèle OO et le modèle LDAP.
L'exemple ci-dessous montre l'utilisation de ces métadonnées dans le processus de conversion d'une requête LINQ en une requête LDAP :

Requête LINQ	Requête LDAP correspondante
var query = from u in ctx.Users where {(u.Surname == "i*" & u.FirstName ="l*")} select new { Nom = u.Surname , Prenom = u.FirstName, Groupes = u.Groups };	(&(ObjectClass=user)(&(sn=i*)(gn=l*))

ctx.Users correspond à une instance de la collection générique *DirectoryEntries<T>* avec comme type T la classe *User*.
En effet, la classe *LDAPQueryContext* va devoir exposer différentes collections de type *DirectoryEntries<type_de_votre_entité>* sur lesquelles l'on va pouvoir effectuer des requêtes LINQ fortement typées.
Il est d'usage de nommer l'entité par un nom singulier et la collection par le nom au pluriel de l'entité. Voici un extrait des collections exposées dans notre exemple :

```
namespace CustomQueryProvider.LDAP
{
    public class LDAPQueryContext
    {

        #region#
        Propriétés

        #region#
        Constructeurs

        #region#
        Fabrique des objets

        #region Collections de données

        private DirectoryEntries<User> users;
        public DirectoryEntries<User> Users{...}

        private DirectoryEntries<Group> groups;
        public DirectoryEntries<Group> Groups{...}

        #endregion

    }
}
```

5 Créez une classe de type IQueryProvider

Comme nous l'avons vu dans l'étape 3, les classes de type IQueryProvider sont en charge du traitement et de l'exécution des requêtes effectuées sur la collection de type IQueryable à laquelle elles sont associées. Il est donc nécessaire de créer au moins une classe de type IQueryProvider. Cette classe doit implémenter au minimum 2 méthodes principales : *CreateQuery()* et *Execute()* (version générique et non-générique)

- La méthode *CreateQuery()* renvoie une nouvelle instance de classe de type *IQueryable* associée à l'arbre d'expression qu'elle reçoit en paramètre. Lorsqu'on fait appel à cette méthode, c'est que l'on désire généralement créer une nouvelle instance de classe de type *IQueryable* qui, lorsqu'elle sera énumérée, va faire appel à la méthode *Execute()* pour traiter et exécuter l'arbre d'expressions associé à cette dernière. Cette méthode est appelée par la plupart des opérateurs LINQ tels que *Where, Select, Join, OrderBy…*

La requête suivante par exemple, va générer deux appels à la méthode *CreateQuery()*.

```
Var query = from u in ctx.Users
        where {(u.Surname == "i*" & u.FirstName ="l*" )}
        elect new { Nom = u.Surname , Prenom = u.FirstName, Groupes
= u.Groups };
```

Le premier appel sera effectué par l'opérateur *Where* avec comme paramètre l'arbre d'expressions contenant les prédicats de filtre et le type de la collection *IQueryable* sur laquelle doit s'effectuer le filtre. Le second appel sera effectué par l'opérateur *Select* avec comme paramètre l'arbre d'expression contenant la projection effectuée dans la clause Select et l'opération précédente.
Dans notre exemple, la classe abstraite *BaseQueryProvider* offre une implémentation minimale de la méthode *CreateQuery()*. A chaque appel cette méthode renvoie une nouvelle instance de la classe *BaseQueryable<T>*, classe mère de toutes nos collections d'objets, en lui passant en paramètre l'arbre d'expressions associé à l'appel. Ainsi la dernière instance de classe créée contient l'arbre d'expressions correspondant à l'ensemble de la requête.
Cette classe peut servir de structure de base à vos classes de type *IQueryProvider* qui devront fournir une implémentation de la méthode *Execute()*, détaillée dans le paragraphe suivant.

- La méthode *Execute()* est le point d'entrée du traitement et de l'exécution effective de la requête spécifiée sur la collection de type IQueryable. Les opérations que doit gérer cette méthode sont les suivantes :
  - L'analyse et l'interprétation de l'arbre d'expressions associé à la requête
  - La construction de la requête dans le langage correspondant à la source de données distante
  - La connexion à la source de données distante et l'exécution de la requête
  - La récupération et la transformation des résultats en entités .NET correspondant au type de retour de la requête

Nous vous recommandons de séparer ces opérations dans des classes différentes, pour les problématiques de réutilisation, de maintenance et d'évolution du code énumérées plus haut. Ainsi, on peut très bien imaginer avoir une classe générique d'interprétation d'arbre d'expressions LINQ qui pourra être réutilisée pour d'autres types de provider.
La méthode *Execute()* aura donc pour rôle de coordonner l'exécution de ces différentes opérations en faisant appel aux méthodes de classes correspondant à chacune des opérations, comme le montre cet extrait de code de notre exemple :

```
public object ExecuteQuery(Expression expression)
    {
        // Parsing de l'expression
        IQueryParser parser = this.context.Factory.GetParser();
        parser.Parse(expression);

        // Construction de la requête
        IQueryFormatter formatter = this.context.Factory.GetFormatter(parser.
From ,parser.Where.Expression ,
                parser.Select == null ? new HashSet<MemberExpression>() :
 parser.Select.SelectColumns);

        // Exécution de la requête
        BaseCommand cmd = this.context.Factory.GetCommand();
```

Donnez votre avis sur ce numéro www.programmez.com/magazine_satisfaction.php

```
        cmd.ConnectionString = context.ConnectionString;
        cmd.Query = formatter.GetQuery();
        cmd.Columns = formatter.GetMappingProperties().Values.ToArray
<string>();
        object results = cmd.ExecuteCommand();

        // Conversion du resultat
        return this.context.Factory.GetDeserialisator(parser.ResultType,
formatter.GetMappingProperties(), parser.Select != null ? parser.Select
.Projection : null, results, parser.OriginalType);
    }
```

6 Implémentez un interpréteur d'arbre d'expressions LINQ

Comme nous l'avons indiqué plus haut, LINQ gère deux modes d'interprétation et d'exécution des requêtes effectuées sur une collection de données. Pour les collections de type *IEnumerable*, les éléments de la requête sont directement convertis en code IL et exécutés en mémoire par appels successifs aux différents opérateurs LINQ trouvés dans la requête. Pour les collections de type *IQueryable*, les opérateurs LINQ ont été surchargés pour prendre en paramètre non pas un délégué mais une arborescence décrivant l'expression réelle tapée par l'utilisateur. Cette arborescence est une instance de la classe *System.LINQ.Expressions.Expression<T>*. L'exemple ci-dessous montre un exemple d'instance de la classe *Expression<T>* et l'arbre d'expression correspondant.

```
Expression<Func<int, bool>> e = (int i) => i > 1 && i % 2 == 1;
```

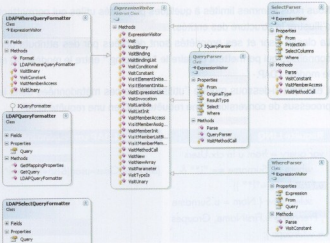
Chaque nœud de l'arbre correspond à un type d'expression contenant des informations spécifiques comme par exemple une constante pour un nœud de type *ConstantExpression* ou une variable pour un noeud de type *ParameterExpression*. Il est donc nécessaire d'implémenter un algorithme efficace et structuré permettant de parcourir récursivement l'arbre d'expressions et appliquer un traitement spécifique à chaque type de nœud rencontré afin d'en extraire les informations nécessaires à la construction et à l'exécution de la requête.

Le design pattern *Visitor* est un pattern recommandé pour ce type de traitement. Il permet d'ajouter simplement sur une classe ou instance des fonctionnalités non prévues au départ sans surcharger la structure de base de l'objet et en gardant si possible une certaine flexibilité et souplesse pour ajouter ou enlever rapidement d'autres fonctionnalités sans alourdir à nouveau l'implémentation définie au départ.

Une structure de ce type est nécessaire compte tenu du nombre infini d'opérateurs et de types d'expressions qui peuvent être crées sur LINQ. Dans notre exemple, l'implémentation de ce pattern s'appuie sur une structure de base, *ExpressionVisitor*, qui est une classe abstraite jouant le rôle de dispatcheur de traitements.

En effet, cette classe possède une méthode *Visit()* qui reçoit en paramètre une expression de l'arbre et délègue le traitement de cette expression à une méthode virtuelle VisitXXX correspondant au type de l'expression. On a ainsi des méthodes *VisitMemberAccess(), VisitConstantExpression(), ...*

Les classes héritant de cette structure de base peuvent spécialiser les méthodes VisitXXX et appliquer un algorithme de traitement spécifique à



leur rôle. C'est le cas des classes *QueryParser, WhereParser, SelectParser, LDAPWhereQueryFormatter* de notre exemple.

Nous avons découpé l'opération d'analyse et d'interprétation de l'arbre d'expressions correspondant à la requête saisie en deux étapes :

- Une première étape de " parsing " dans laquelle on extrait des parties de l'arbre correspondant aux différents opérateurs LINQ (nous nous sommes limités aux opérateurs *Select* et *Where*). Pour le *Select* (classe *SelectParser*), on s'intéresse à une projection éventuelle effectuée dans la clause Select (constructeur de la classe anonyme, liste des propriétés de l'entité participant à la projection). Pour le *Where* (classe *WhereParser*), on récupère tout simplement l'arbre d'expression contenant les prédicats de filtre spécifiés dans la requête, ainsi que le type de la collection sur laquelle doivent s'appliquer les filtres.
- Une deuxième étape dans laquelle on va utiliser les informations extraites par les parseurs pour construire la requête dans le langage associé à la source de données distante (requête LDAP dans notre exemple). La classe *LDAPWhereQueryFormatter* parcourt alors l'arbre d'expression correspondant à la clause Where et traduit les prédicats en langage LDAP. Les attributs LDAP à charger dans la recherche, sont eux récupérés grâce à la liste des propriétés participant à la projection de la clause *Select*. Les classes *QueryParser* et *LDAPQueryFormatter* sont tout simplement les façades de communication avec les classes précédentes.

Voici quelques extraits de code qui vous permettront d'avoir une idée du processus d'interprétation et de construction de la requête.

Classe ExpressionVisitor# : extrait de la méthode Visit

```
protected virtual Expression Visit(Expression exp)
  {

      if (exp == null)

          return exp;

      switch (exp.NodeType)
      {

              case ExpressionType.ExclusiveOr:

                  return this.VisitBinary((BinaryExpression)exp);
```

```
            case ExpressionType.TypeIs:

                return this.VisitTypeIs((TypeBinaryExpression)exp);

            case ExpressionType.Conditional:

                return this.VisitConditional((ConditionalExpression)exp);

            case ExpressionType.Constant:

                return this.VisitConstant((ConstantExpression)exp);

            case ExpressionType.Parameter:

                return this.VisitParameter((ParameterExpression)exp);

            case ExpressionType.MemberAccess:
```

**N.B : l'implémentation de base de cette classe a été fournie par Matt Warren dans son article sur la construction d'un " IQueryable Provider SQL " que nous vous invitons vivement à lire à l'adresse suivante : http://blogs.msdn.com /mattwar/archive/2007/07/31/LINQ-building-an-iqueryable-provider-part-ii.aspx*

Classe SelectParser : méthodes Parse et VisitMemberAccess

```
public void Parse(Expression expression)
  {

      MethodCallExpression m = expression as MethodCallExpression;
      this.Visit(m.Arguments[0]);

      // récupération du délégué de projection
      LambdaExpression lambda = (LambdaExpression)StripQuotes
(m.Arguments[1]);
      Projection = lambda.Compile();

      // Récupération des différentes colonnes de la clause Select
      SelectColumns = new HashSet<MemberExpression>();
      this.Visit(lambda.Body );
  }

protected override Expression VisitMemberAccess(MemberExpression m)
  {
      if (m.Expression != null && m.Expression.NodeType == Expression
Type.Parameter)
      {
          SelectColumns.Add(m);
          return m;
      }
      throw new NotSupportedException(string.Format("The member
'{0}' is not supported", m.Member.Name));
      }
```

Classe WhereParser : méthodes Parse et VisitConstantExpression

```
public void Parse(Expression expression)
  {

      MethodCallExpression m = expression as MethodCallExpression;

      // Récupération du type de la collection
      this.Visit(m.Arguments[0]);

      // Récupération de la l'expression de la clause WHERE
      LambdaExpression lambda = (LambdaExpression)StripQuotes
(m.Arguments[1]);
      this.Expression = lambda.Body;
  }

protected override Expression VisitConstant(ConstantExpression c)
  {
      IQueryable q = c.Value as IQueryable;
      if (q != null)
      {
          From = c;

      }
      return c;
  }
```

Classe LDAPWhereQueryFormatter : extrait de la méthodes VisitBinaryExpression et VisitMemberAccess

```
protected override Expression VisitBinary(BinaryExpression b)
  {

  switch (b.NodeType)
  {

      case ExpressionType.And:

          FormatterHelper.OpenParenthesis(query);
          query.Append("&");
          this.Visit(b.Left);
          this.Visit(b.Right);
          FormatterHelper.CloseParenthesis (query);
          break;


      case ExpressionType.Equal:

          FormatterHelper.OpenParenthesis(query);
          this.Visit(b.Left);
          query.Append("=");
          this.Visit(b.Right );
          FormatterHelper.CloseParenthesis(query);
          break;
      [...]
  }

protected override Expression VisitMemberAccess(MemberExpression m)
  {

      if (m.Expression != null && m.Expression.NodeType == Expression
Type.Parameter)
  {
```

```
            query.Append(FormatterHelper.GetMappingAttributeName(m).
ToLower () );
            return m;
        }

        throw new NotSupportedException(string.Format("The member '{0}'
is not supported", m.Member.Name));

    }
```

7 Créez une façade de communication avec la source de données distante

Il s'agit de créer une classe qui va encapsuler la logique de communication avec la source de données que vous souhaitez interroger. Elle va donc utiliser l'API que vous avez identifiée à l'étape 1, pour se connecter à la source de données distante, exécuter la requête construite dans les étapes précédentes, et récupérer le résultat correspondant.
Cette façade permet donc de découpler le reste de votre application avec la technologie utilisée pour communiquer avec la source de données distante. Dans notre exemple, cette façade est matérialisée par la classe abstraite BaseCommand et la classe concrète *DirectorySearcherCommand* qui hérite de *BaseCommand*. Cette classe concrète correspond à la façade par défaut de notre provider et utilise l'API *DirectoryServices* pour communiquer avec le serveur LDAP.

```
Public abstract class BaseCommand
  {
      public string Query { get; set; }
      public string ConnectionString { get; set; }
      public abstract object ExecuteCommand();
      public string[] Columns { get; set; }
  }

public class DirectorySearcherCommand : BaseCommand
  {

      public override object ExecuteCommand()
      {
          DirectoryEntry root = new DirectoryEntry(ConnectionString);
          DirectorySearcher search = new DirectorySearcher(root, Query
, Columns , SearchScope.Subtree);
          SearchResultCollection res = search.FindAll();
          return res;
      }

  }
```

Avec le principe de " *Factory* " que nous avons expliqué plus haut, on pourrait imaginer avoir plusieurs mécanismes de communication et choisir dynamiquement lors de l'instanciation du contexte, le mécanisme le plus adapté à utiliser.

8 Créez une classe de désérialisation des résultats de la requête

Une fois la requête exécutée, les résultats sont généralement rangés dans une structure fortement couplée à l'API de communication que vous avez choisie (*DataReader* ou *DataSet* si vous utilisez ADO.NET, *SearchResultCollection* si vous utilisez *DirectoryServices* comme dans notre exemple). Il est alors nécessaire d'implémenter un algorithme permettant de convertir les éléments de cette structure de données en une collection d'objets .NET typés correspondant au résultat attendu par l'utilisateur.

Conclusion

Dans cet article, nous avons présenté les principales étapes de création d'un provider LINQ, qui permet d'interroger un type particulier de collections de données en utilisant la syntaxe LINQ. Même si certains aspects n'ont pas été couverts, tels que la gestion de la mise à jour des données sur la source de données distante, la politique de cache pour optimiser les performances du provider, la gestion des opérateurs LINQ autre que Select et Where… nous avons fourni les informations suffisantes et un socle technique réutilisable pour pouvoir démarrer rapidement la création de votre provider.
Plusieurs providers LINQ ont déjà vu le jour sur la communauté Internet : LINQ-to-Sharepoint, LINQ-to-Amazon, Linq-to-Streams, LINQ-to-RDF, et bien d'autres. N'oubliez donc pas de regarder ce qui existe avant de vous lancer dans la création de votre provider.

■ **Ismaël René Nzouetom** - *Consultant - Microsoft Services I France*

Programmez! LE MAGAZINE DU DÉVELOPPEMENT

Prochain numéro Hors Série .Net : 14 juin 2008

• Visual Basic 9 :
Les meilleurs trucs et astuces pour maîtriser votre langage préféré et exploiter au mieux VB dans Visual Studio 2008.

• Silverlight :
Développer et déployer vos applications Silverlight. Les outils, les bonnes pratiques.

Sources sur le site
www.programmez.com

Créez vos documents Open XML avec .NET et Linq

OPENXML

Les utilisateurs adoptent les produits que les développeurs créent. C'est pour cette raison que les outils et les SDK mis à la disposition des entreprises et des équipes de développement sont un critère majeur dans l'adoption d'une technologie ou d'un format. Microsoft propose depuis quelques mois un SDK pour Open XML et Linq.

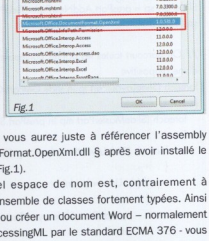


Fig.1

Pour travailler avec ce format, Microsoft fournit un espace de nom dans .NET 3 ainsi qu'un SDK permettant l'utilisation de classes fortement typées pour représenter les différentes parties contenues dans un document.

Attention : le SDK Open XML est toujours en CTP.

Le vénérable espace de nom System.IO.Packaging

Un document Open XML est un document compressé au format Zip (du moins par défaut, la spécification n'exclut pas d'autres types de conteneur). L'espace de nom System.IO.Packaging disponible avec .NET 3 permet de manipuler l'enveloppe OPC (Open Packaging Convention) de stockage des données d'un document. Vous aurez donc à votre disposition des classes telles que Package, PackagePart et PackageRelationship pour manipuler les différents éléments de l'enveloppe. Pour utiliser cet espace de nom, il vous suffira simplement de référencer l'assembly WindowsBase.dll dans votre projet.

Remarque : les méthodes d'extension de .NET 3.5 profitent également à la simplicité d'utilisation de cet espace de nom en embellissant les collections. Cela a pour effet de simplifier quelque peu le code nécessaire :

```
monPackage.GetRelationshipsByType(documentRelationshipType).FirstOrDefault();
```

Malheureusement, la classe PackagePart qui représente une partie ou la classe Package qui spécifie l'enveloppe d'un document sont définies de façon abstraite. Ainsi, il n'y a aucune différence au niveau de la classe entre une partie XML contenant des paragraphes et une partie binaire contenant une image ; et il en est de même entre une enveloppe PowerPoint et celle d'un document XPS (qui utilise aussi la structure OPC). Cette limitation est évidemment intentionnelle, puisque cet espace de nom n'est censé ne manipuler que la partie OPC du format et non les éléments XML des différents formats Open XML.

C'est ici que rentre en jeu le SDK Open XML et son assembly dont l'objectif est justement de régler cette limitation.

Le SDK Open XML

Ce SDK fournit une API permettant de faire abstraction de la structure logique de l'enveloppe OPC pour se concentrer sur les différents éléments constituant les formats Open XML. Il est construit au-dessus de l'espace de nom System.IO.Packaging avec une architecture de classes l'abstrayant totalement. Cette nouvelle DLL apporte un nouvel espace de nom Microsoft.Office.DocumentFormat.OpenXml.Packaging que nous utiliserons dans les exemples qui suivront.

Pour réaliser vos projets, vous aurez juste à référencer l'assembly Microsoft.Office.DocumentFormat.OpenXml.dll § après avoir installé le SDK - dans votre solution (Fig.1).

Ce qu'apporte ce nouvel espace de nom est, contrairement à System.IO.Packaging, un ensemble de classes fortement typées. Ainsi lorsque vous voulez ouvrir ou créer un document Word – normalement appelé document WordprocessingML par le standard ECMA 376 - vous disposerez de la classe WordprocessingDocument pour réaliser toutes vos opérations.

L'exemple suivant permet de créer intégralement un document WordprocessingML que vous pourrez ouvrir dans Office 2007 :

```
using (WordprocessingDocument wordDoc = WordprocessingDocument.
Create("MonDoc.docx", WordprocessingDocumentType.Document))
{
  // On ajoute la partie principale
  MainDocumentPart mainPart = wordDoc.AddMainDocumentPart();

  // Création de la structure XML WordprocessingML
  const string wNamespace = "http://schemas.openxmlformats.org/
wordprocessingml/2006/main";
  XmlDocument xmlDoc = new XmlDocument();
  XmlElement documentElem = xmlDoc.CreateElement("w:document", w
Namespace);
  XmlElement bodyElem = xmlDoc.CreateElement("w:body", wNamespace);
  XmlElement paragraphElem = xmlDoc.CreateElement("w:p", wNames
pace);
  XmlElement runElem = xmlDoc.CreateElement("w:r", wNamespace);
  XmlElement textElem = xmlDoc.CreateElement("w:t", wNamespace);
  XmlText textElemContent = xmlDoc.CreateTextNode("Texte de mon
document ici !");

  xmlDoc.AppendChild(documentElem);
  documentElem.AppendChild(bodyElem);
  bodyElem.AppendChild(paragraphElem);
  paragraphElem.AppendChild(runElem);
  runElem.AppendChild(textElem);
  textElem.AppendChild(textElemContent);

  xmlDoc.Save(mainPart.GetStream());
}
```

Listing 1

Le SDK fournit bien évidemment des classes typées pour les documents SpreadsheetML et PresentationML. L'exemple suivant montre comment le SDK rend très simple la manipulation des différentes parties d'un document quelconque PresentationML :

```
// Création d'une diapositive et injection de la structure XML à partir d'un flux
SlidePart sp = presPart.AddNewPart<SlidePart>();
sp.FeedData(slideXML.GetStream())

// On parcourt toutes les parties contenant de disposition des éléments
d'une diapositive
foreach (SlideLayoutPart lp in slide.GetPartsOfType<SlideLayoutPart>())
{
  foreach (SlideMasterPart mp in presPart.SlideMasterParts)
  {
      foreach (SlideLayoutPart slp in mp.SlideLayoutParts)
      {
          if (slp.Uri == lp.Uri)
          {
              sp.AddPart<SlideLayoutPart>(slp);
              break;
          }
      }

      SlideLayoutPart newSlideLayoutPart = presPart..AddNewPart<Slide
LayoutPart>();
      newSlideLayoutPart.FeedData(
  }
}
```

Listing 2

Le code précédent montre plusieurs aspects intéressants de l'utilisation de cette nouvelle API. Tout d'abord toutes les parties sont typées, ce qui facilite la manipulation des parties en proposant des méthodes adéquates. Ensuite, vous remarquerez l'utilisation des méthodes génériques pour typer une méthode et le résultat retourné. L'utilisation du SDK permet un gain de productivité assez conséquent par rapport à un code n'utilisant que l'espace de nom System.IO.Packaging. Dans certains cas, on pourra lui reprocher de manquer quelquefois de flexibilité.

Utilisation de XLinq dans un document Open XML

Les différentes librairies apportées par l'espace de nom System.IO.Packaging et le SDK gèrent pour vous tout l'aspect inhérent à l'enveloppe du document, c'est-à-dire la structure logique OPC. Vous n'aurez par conséquent qu'à vous concentrer pour l'essentiel sur la structure XML. L'arrivée du framework 3.5 et notamment de LINQ (Language Integrated Query) tombe à point nommé pour réaliser cette tâche. Nous allons donc utiliser XLinq (XML Linq) pour construire et requêter en tout simplicité une structure XML. Reprenons l'exemple du Listing 1 dans lequel nous créons un document Word et remplaçons la construction de la structure XML par un code XLinq :

```
using (WordprocessingDocument wordDoc =
  WordprocessingDocument.Create("MonDoc.docx",  Wordprocessing
DocumentType.Document))
{
  // On ajoute la partie principale
  MainDocumentPart mainPart = wordDoc.AddMainDocumentPart();

  // Création de la structure XML WordprocessingML
  XNamespace wNamespace = "http://schemas.openxmlformats.org/
wordprocessingml/2006/main";
  XDocument xmlDoc = new XDocument(
  new XDeclaration("1.0", "utf-8", "yes"),
      new XElement(wNamespace + "document",
          new XElement(wNamespace + "body",
              new XElement(wNamespace + "p",
                  new XElement(wNamespace + "r",
                      new XElement(wNamespace + "t",
                          new XText("Texte de mon document ici ! ")
                      )
                  )
              )
          )
      )
  );
  xmlDoc.Save(new StreamWriter(mainPart.GetStream()));
}
```

N'oublions pas le principal avantage de XLinq, voici un exemple retournant tous les nœuds et le style des paragraphes d'un document WordprocessingML :

```
var paragraphs =
  from para in xmlDoc
      .Root
      .Element(w + "body")
      .Descendants(w + "p")
      let styleNode = para
          .Elements(w + "pPr")
          .Elements(w + "pStyle")
          .FirstOrDefault()
      select new
      {
          ParagraphNode = para,
          StyleName = styleNode != null ? (string)styleNode.Attribute(w
+ "val") : "normal"
      };
```

XLinq représente un nouveau moyen élégant de créer et de requêter les structures XML des documents Open XML.

Conclusion

Microsoft fournit un jeu d'API intéressant mais néanmoins incomplet – d'où son statut de CTP - pour pouvoir réellement manipuler les documents Open XML de façon totalement orientée objet. Néanmoins, en attendant de prochains ajouts dans les API, XLinq permet de combler ce manque tout en garantissant aux développeurs .NET un code compact et une productivité accrue.

■ **Julien Chable** – *Wygwam France*

LANGAGE : F# | APPLICATION : --
DIFFICULTE : ●●●●● | SOURCE : oui

F#, un langage fonctionnel pour la plate-forme .Net

Sources sur le site
www.programmez.com

Les langages fonctionnels ont toujours peiné à s'imposer dans le monde du développement. Avec F# qui bénéficie de la plate-forme .Net, les choses pourraient évoluer rapidement. Découvrons ce langage qui dispose d'atouts maîtres.

F# n'est pas un langage Microsoft. Ou du moins pas encore. F# a été développé par le laboratoire Microsoft Research, mais l'éditeur l'intégrera officiellement dans la prochaine version de .Net, et déjà la distribution de F# dispose d'un add-in pour Visual Studio qui permet de coder facilement. Il fonctionne aussi parfaitement sous Linux/mono et a été le fruit d'une lente évolution. Au départ, modèle a été pris sur Haskell, langage fonctionnel pur, et F# en conserve quelques traces. Mais finalement les travaux du Microsoft Research sont essentiellement basés sur le langage Objective Caml, pour plusieurs raisons dont la plus évidente, en regard de la plate-forme .Net est l'extension objet de OCaml. Nous avons à cette occasion, nous Français, de quoi être fiers, car OCaml est une création de l'Inria (http://caml.inria.fr/about/index.fr.html). Malheureusement OCaml a toujours peiné à être reconnu, notamment en France. Toujours est il que OCaml entre par la grande porte dans le monde du développement car, grosso modo, F# est un O Caml pour .Net. Si OCaml conserve des fonctionnalités bien à lui comme les modules paramétrés qui n'existent pas en F#, sauf erreur de votre serviteur, F# est presque OCaml et présente avec lui une bonne compatibilité au niveau source. D'un autre côté F# a pour lui de bénéficier du socle de .Net, ce qui représente non seulement tout le jeu de librairies, mais aussi la possibilité d'une utilisation astucieuse. Ainsi on pourra écrire le code délicat d'une application, le travail avec des structures de données complexes telles que graphes ou arbres de recherche, par exemple, en profitant de la puissance d'expression de F#, placer ce code dans un assembly et utiliser celui-ci par le gros de l'application écrite de façon plus classique, avec C#. On voit ainsi une opportunité formidable de profiter du meilleur de deux mondes, fonctionnel et objet. On peut même aller plus loin et émettre une opinion osée: F# étant multi-paradigme, comme OCaml, il fait ce que tous les autres langages .Net savent faire, aussi bien si ce n'est mieux, et il apporte par le paradigme fonctionnel, des fonctionnalités et une expressivité que les autres langages n'ont pas. On peut donc se demander si, une fois officiellement intégré à .Net, F# ne serait pas LE langage de choix, devant tous les autres. Mais au fait, qu'est-ce qu'un langage fonctionnel ? La réponse mériterait à elle seule plusieurs articles. Disons simplement, comme Wikipédia, que "La programmation fonctionnelle est un paradigme de programmation qui traite le calcul comme l'évaluation de fonctions mathématiques et évite l'état et les données mutables". Dans ce contexte les fonctions sont des valeurs du langage, puisqu'elles s'évaluent, et comme telles, peuvent être passées en argument à une fonction ou être une valeur de retour. Bref, en fonctionnel, une fonction est une entité de première classe. Il existe déjà de nombreux didacticiels dédiés à F#, toutefois votre serviteur a remarqué que ceux-ci font l'impasse sur des points pas nécessairement évidents pour qui n'est pas familiarisé avec un langage fonctionnel en général, et avec OCaml en particulier. La suite de cet article sera donc un modeste didacticiel, qui à partir de petits exemples concrets, veut aider à prendre le langage en main en fournissant les informations qu'on ne trouve pas clairement ailleurs. Le but étant que le lecteur puisse apprécier F#, qui, objectivement, le mérite. Le lecteur trouvera tout le code dans un projet Visual Studio, sur le site. L'utilisation de Visual Studio, pratique, n'est toutefois pas obligatoire, pour compiler et exécuter les exemples.

1 Hello World!

Tradition oblige :) Mais en plus de la phrase classique, nous allons voir comment imprimer sur la console un type de données omniprésent en programmation fonctionnelle: la liste. Voici un premier jet.

```
System.Console.WriteLine("Hello World");
do printf "Programmez!\n"
do print_string "Abonnez vous :)\n"

let maliste = [1;2;3;4;5;6;7;8;9]
do print_string (maliste.ToString())
let _ = read_line ()
```

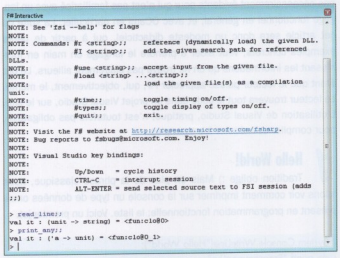
La première ligne de ce code est une ligne à la C# qui montre qu'appeler une méthode statique d'une classe .Net est une simple formalité. La suite du code est plus dans l'esprit F# et utilise des fonctions intégrées. Ces fonctions sont les mêmes qu'en OCaml. F# présente une très bonne compatibilité avec OCaml ainsi que nous l'avons dit. On remarque que les arguments d'une fonction F# ne figurent pas entre parenthèses, à contrario des arguments d'une méthode C#. La quatrième ligne, par l'instruction *let*, associe la liste des entiers de 1 à 9 avec le nom *maliste*. La ligne suivante essaie d'imprimer la liste "à la C#". Mais cela ne donne rien de bon. On se doute bien que, sous le capot, la liste F# est implémentée à partir des collections .Net. Mais on voit ici que cela n'implique pas pour autant l'existence d'une methode ToString redéfinie comme on aurait pu l'espérer. Nous verrons plus loin comment procéder. En revanche le fait que *maliste* dispose de la méthode ToString par défaut indique que nous avons là un objet, même si le programmeur fonctionnel et donc le programmeur F# ne s'en préoccupera pas plus que cela. Si les arguments d'une fonction F# ne doivent pas être placés entre parenthèses, pourquoi trouve-t-on des parenthèses autour de *maliste.ToString()* ? C'est la syntaxe pour demander l'évaluation de *maliste.ToString()* avant son passage en argument à *print_string*. La dernière ligne lit une ligne sur la console, ce qui est un moyen

d'attendre que l'utilisateur appuie sur la touche [return]. L'instruction let _ fait tomber le résultat de la lecture à la trappe, dans ce que l'on appelle un motif universel, une sorte de joker, fourre-tout et poubelle tout à la fois. Au fait pourquoi y –a-t-il des parenthèses derrière read_ligne ?

2 L'interpréteur interactif

Depuis Visual Studio lancez l'interpréteur F# interactif avec la combinaison de touches Alt-return. Ou bien lancez l'exécutable fsi.exe si vous n'avez pas Visual Studio. A l'invite de l'interpréteur saisissez

```
read_line ;;
```



sans oublier les deux points-virgules puis appuyez sur return.

Ce faisant, et puisque nous n'avons pas donné d'arguments nous demandons l'évaluation du terme read_line lui même. Si nous avions donné un argument (en donnant read_line () ;;) nous aurions demandé l'évaluation (ou son exécution si vous préférez) de la fonction. Voici ce que nous répond l'interpréteur à propos de read_line :

```
val it : (unit -> string) = <fun:clo@0>
```

Il nous dit qu'il s'agit d'une fonction (fun) qui reçoit unit en argument et retourne une chaîne, ceci étant noté par unit->string. Qu'est-ce donc que unit ? Unit est en quelque sorte l'alter ego du void de C, et il se note (). Donc quand nous écrivons

```
read_string ()
```

nous appelons la fonction avec l'argument qu'elle s'attend à recevoir. En F# une fonction reçoit toujours au moins un argument et retourne toujours une valeur, celui-ci et celle-là pouvant être () ou unit. Le compilateur F# éprouverait un mal être existentiel devant une fonction déclarée comme ne recevant et/ou ne retournant rien. En F# la dernière chose évaluée au sein d'une fonction constitue ipso facto sa valeur de retour.

3 Hello World! II

Ré-écrivons notre Hello World en ce qui concerne la liste. Nous utilisons cette fois la fonction intégrée print_any dont le rôle est de rendre n'importe quelle entité sous la forme d'une chaîne humainement compréhensible. Par ailleurs, nous construisons notre liste plus élégamment et nous voyons une autre façon de passer à la trappe la valeur de retour de read_string :

```
let maliste = [1..9]

do print_any maliste
do print_newline ()
do ignore (read_line () )
```

La signature de print_any est intéressante. Pour la connaître, tapez

```
print_any ;;
```

dans l'interpréteur interactif. Vous obtenez :

```
val it : ('a -> unit) = <fun:clo@0_1>
```

le **'a** de la signature signifie "n'importe quoi". print_any est donc une fonction générique. Mais pourquoi sa signature n'est-elle pas 'a -> string ? Parce que print_any imprime la chaîne sur la console, mais ne la retourne pas. Ce sont deux choses indépendantes. En interne, il est probable que print_any s'appuie sur la fonction intégrée any_to_string, également générique et dont la signature ne surprend pas cette fois :

```
any_to_string ;;
val it : ('a -> string) = <fun:clo@0_2>
```

4 Créer un module F#

Un programmeur fonctionnel a horreur du code répétitif. A ce titre enchaîner les séquences print_any/print_newline n'est pas acceptable. Nous allons donc écrire une fois pour toutes une fonction qui réalise la séquence. Et pour pouvoir réutiliser cette fonction, nous ne la placerons pas dans le code, mais dans un module. Avec les langages fonctionnels mono-paradigme, le module est l'unité d'encapsulation. Avec F# nous disposons aussi de l'objet.

Toutefois en F# le module reste l'outil de choix. Un module est simplement un fichier contenant des définitions de fonction. Si l'on veut faire les choses bien, on donne également un fichier interface contenant les signatures des fonctions du module.

Ceci présente deux avantages : d'abord il est possible, si on le veut, de remplacer plus tard l'implémentation du module tout en gardant la même interface. Ensuite seules les fonctions déclarées dans l'interface seront publiques. Voici (exemple HelloWorld III) les fichiers d'interfaces et de définition de notre module :

```
(* fichier fred.fsi*)

module Fred
val print: 'a -> unit

(* fichier fred.fs)
module Fred
let print a = print_any a ; print_newline ()
```

Et on utilise notre module ainsi

```
(* fichier helloworld.fs *)

open Fred
```

```
let maliste = [1..9]

do print maliste
do ignore (read_line () )
```

Ou bien, si l'on n'ouvre pas le module avec l'instruction open, on écrira :

```
do Fred.print maliste
```

Pour que le compilateur F# puisse s'y retrouver lorsqu'on travaille sous Visual Studio, ce dernier doit lui présenter les fichiers dans le bon ordre, à savoir le fichier .fsi en premier, puis le fichier de définition de module, puis les fichiers utilisant le module. Les fichiers doivent donc apparaître dans cet ordre dans l'arborescence du projet. Si ce n'est pas le cas, déchargez le projet (Unload Project dans le menu contextuel), éditez le fichier .fsharpp à la main afin de corriger l'ordre, puis rechargez le projet (Reload Project dans le menu contextuel)

5 Les fonctions sont des valeurs

Que les fonctions soient des valeurs rend bien souvent la programmation élégante et concise. Imaginons, à titre d'exemple, un wargame hypothétique. Un soldat sait faire plusieurs actions. Chacune de ces actions peut en outre donner un score.

Une fonction d'intelligence artificielle pourrait déterminer, pour une phase de jeu donnée, quelles séquences d'actions le soldat devra accomplir. Cette séquence déterminée, son exécution et le cumul du score ne demande que deux lignes de code en F#, le reste du code à peine plus! Voyons comment :

```
let actions_potentielles =
[|
   (fun x -> print_string (x ^ " avance\n") ; 0) ;
   (fun x -> print_string (x ^ " recule\n") ; -1) ;
   (fun x -> print_string (x ^ " tire\n") ; 1) ;
   (fun x -> print_string (x ^ " lance une grenade\n") ; 3)
|]

let random = new System.Random()
let actions = [ for i in 1 .. 10 -> actions_potentielles.(random.Next 4) ]

let result_actions = List.map (fun f -> f "Soldat Fred") actions
let score = List.fold_left (+) 0 result_actions
do printf "Score: %d" score
```

On commence par déclarer un tableau de fonctions anonymes aussi facilement qu'on le ferait pour un tableau d'entiers en C#, puisque les fonctions sont des valeurs :) Chaque fonction s'attend à recevoir le nom d'un soldat en argument, imprime l'action exécutée et retourne le score correspondant. Ensuite nous instancions la classe .Net System.Random. La ligne suivante simule l'algorithme d'intelligence artificielle. Nous créons une liste d'actions à exécuter. La constitution de cette liste est effectuée au moyen d'une liste compréhensive.

Son expression, pour puissante et concise qu'elle soit, reste facile à saisir : il s'agit de la constitution d'une liste de dix actions, chacune de ces actions étant prise aléatoirement dans le tableau d'actions potentielles. Décortiquons maintenant cette ligne magique :

```
let result_actions = List.map (fun f -> f "Soldat Fred") actions
```

La fonction *map* du module *List* reçoit deux arguments. Le premier est une fonction, le second une liste. La fonction reçue est ensuite appliquée à chaque élément de la liste, et la valeur de retour de la fonction est à chaque fois collectée, constituant ainsi une liste qui sera la valeur de retour de *map*.

La fonction anonyme que nous passons à *map* est elle aussi magique. Techniquement il s'agit d'une fonction dite d'ordre supérieur. Cela signifie qu'elle s'attend à recevoir une fonction en argument, ici chaque élément de la liste, et cette fonction/élément sera appelée à son tour avec "Soldat Fred" en argument.

Ainsi en une ligne de code chaque action de la séquence est exécutée et le résultat des courses est collecté dans une liste. Pour terminer, nous faisons la somme de tous les éléments de cette liste au moyen de *List.fold_left*. Cette fonction "plie" la liste en une seule valeur en lui appliquant une fonction, ici l'opérateur infixe +, sur la base de la valeur initiale 0. Est-il possible de rêver pareille concision avec C# ou un autre langage impératif/objet ?

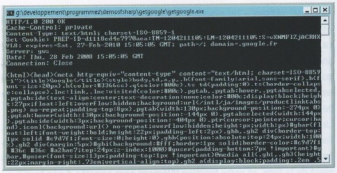
6 Filtrage de motifs, gardes et exception

Les deux premiers sont classiques en programmation fonctionnelle, le troisième un peu moins. Voici un petit jeu de C++/C- qui met en oeuvre ces fonctionnalités.

On notera encore la lecture sécurisée d'un entier depuis la console et la déclaration de deux fonctions récursives.

Bien noter la présence du mot-clé *rec* pour celles-ci. Sans lui le compilateur ne sait pas gérer l'appel récursif.

```
let rec getNumber () =
try
   printf "Entrez un nombre entier entre 1 et 100: " ;
   read_int ()
with
   | Failure x -> print_string "Mauvaise saisie\n";
                  getNumber ()


let rec jouer nombre tentative =
   let saisie = getNumber ()
   in match saisie with
   | 0 -> print_string "Abandon du jeu\n"
   | _ when (saisie = nombre) -> printf "Bravo. ( %d essai(s) )\n" tentative
   | _ when (saisie > nombre) -> print_string "Trop grand\n" ; jouer nombre
(tentative+1)
   | _ -> print_string "Trop petit\n" ; jouer nombre (tentative+1)

let jeu () =
   let random = new System.Random ()
   in   print_string "Vous devez deviner un nombre entre 1 et 100\n" ;
        jouer (random.Next 101) 1 ;
        print_string "Appuyez sur [return]" ;
        ignore ( read_line () )

do jeu ()
```



7 Programmation objet et réseau

Notre dernier exemple obtient la page d'accueil de Google à travers un socket. C'est l'occasion d'illustrer comment les paradigmes fonctionnels et objets peuvent collaborer avec profit dans le cadre de .Net. Le code aurait pu être écrit avec plus de concision mais il aurait été moins parlant. Notamment, nous lisons la page ligne par ligne et nous constituons une liste de lignes pour un affichage ultérieur. En commentaire dans le source vous trouverez cette médiocre fonction pour la lecture des lignes et la constitution de la liste :

```
let rec ReadAllLines (socket :Socket) (sr :StreamReader) lines =
        match socket.Available with
        | 0 -> List.rev lines
        | _ -> ReadAllLines socket sr (sr.ReadLine() :: lines)

let ReadPage socket (sr :StreamReader) =
  let first = sr.ReadLine()
  in   ReadAllLines socket sr [first]
```

Nous lisons d'abord la première ligne dans le socket, étape obligatoire, puis nous déléguons à une autre fonction la tâche de lire les lignes restantes tant que des octets se présentent dans le socket. Ce code s'il est instinctif, est néanmoins maladroit. La présence de ReadAllLignes pollue inutilement. Rappelons-nous toujours qu'en F# une fonction est une valeur. Partant de cela il est plus habile de déclarer une fonction auxiliaire locale à ReadPage, comme ceci :

```
let ReadPage socket (sr : StreamReader) =
  let rec aux (socket : Socket) (sr : StreamReader) lines =
            match socket.Available with
            | 0 -> List.rev lines
            | _ -> aux socket sr (sr.ReadLine() :: lines) in
  let first = sr.ReadLine()
  in   aux socket sr [first]
```

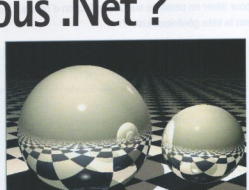
Le reste du code montre comment manipuler les classes .Net, les parenthèses accompagnant les constructeurs et méthodes étant cette fois classiques. Il montre encore comment préciser le type des arguments des fonctions lorsque le compilateur ne peut pas les inférer et encore comment tester le type d'une exception qui est une classe .Net.

```
open System.IO
open System.Text // pour Encoding
open System.Net
open System.Net.Sockets

#nowarn "67" ;;
#nowarn "25" ;;

let server = "google.fr"
let port = 80
let requete =  "GET / HTTP/1.0\r\nHost: www.google.fr\r\n\r\n"


let OpenConnexion () =
  let socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
 ProtocolType.Tcp)
  in   socket.Connect(server, port) ;
       socket


let SendRequest (ns : NetworkStream) request =
  let buffer = Encoding.ASCII.GetBytes(request : string)
  in ns.Write(buffer, 0, buffer.Length)


let ReadPage socket (sr : StreamReader) =
  let rec aux (socket : Socket) (sr : StreamReader) lines =
            match socket.Available with
            | 0 -> List.rev lines
            | _ -> aux socket sr (sr.ReadLine() :: lines) in
  let first = sr.ReadLine()
  in   aux socket sr [first]

let rec PrintPage page =
  match page with
       | head :: tail -> printf "%s\n" head ; PrintPage tail
       | [] -> ()



let go () =
try
  let socket = OpenConnexion () in
  let network_stream = new NetworkStream(socket) in
  let stream_reader = new StreamReader(network_stream) in
  SendRequest network_stream requete ;
  let page = ReadPage socket stream_reader in
  socket.Close() ;
  PrintPage page

with
  | :? Sockets.SocketException ->
       print_string "Exception socket"
  | :? System.Exception ->
       print_string "Exception inattendue"


do     go () ;
  ignore ( read_line () ) )
```

■ **Frédéric Mazué** - *fmazue@programmez.com*

Parallel FX : la parallélisation facile sous .Net ?

Avec l'avènement des multi-cœurs, les ordinateurs de bureau disposent d'une puissance accrue par rapport au mono cœur. Mais à l'heure actuelle, cette puissance est mal maîtrisée. D'une part, nos applications qui n'utilisent qu'un seul cœur, n'en profiteront pas automatiquement, d'autre part les concepts du développement parallèle sont compliqués, et écrire des algorithmes qui profitent des multi-cœurs reste difficile. Mais nous ne pouvons plus nous voiler la face, nous devons désormais concevoir nos applications d'une manière différente et penser parallèle.



Pour nous aider dans cette entreprise, Microsoft met à disposition une extension à .NET qui a pour mission de simplifier et réduire les concepts de la programmation parallèle.

Extension parallèle à .NET (PFX)

Disponible en version CTP (Community Tech Preview) : http://msdn2.microsoft.com/en-us/concurrency/default .aspx l'idée de base est de pouvoir fournir aux développeurs .NET une bibliothèque qui pourra exploiter naturellement tous les processeurs disponibles, sans que cela deviennent une tâche insurmontable

Une fois installée, vous retrouverez la documentation ainsi que des exemples dans le répertoire C:\Program Files\Microsoft Parallel Extensions Dec07 CTP

Note : Pour tous les exemples qui suivent la bibliothèque C:\Program Files (x86)\Microsoft Parallel Extensions Dec07 CTP\System.Threading.dll a été ajoutée en tant que référence.

Nous allons donc dès à présent créer notre premier algorithme parallèle. Imaginons que nous souhaitions paralléliser la boucle suivante, où nous alimentons un tableau d'une chaîne de caractères, constituée du résultat d'une multiplication, ainsi que du numéro de thread qui a traité la demande.

```
//En C#
for (int i = 0; i < MAX; i++)
{
vecteurs[i] = "Carré : " + (i * i).ToString() + " ThreadID :" + Thread.Current
Thread.ManagedThreadId.ToString();
}
//En VB
For i As Integer = 0 To MAX
 vecteurs(i) = "Carré : " + (i * i).ToString() + " ThreadID :" + Thread.Current
Thread.ManagedThreadId.ToString()
Next
```

Chaque itération, étant indépendante, il est possible de tenter de paralléliser cette boucle en employant notre nouvelle bibliothèque.

```
//En C#
string[] vecteurs = new string[MAX];
Parallel.For(0, MAX, (int i)=>
{
vecteurs[i] = "Carré : " + (i * i).ToString() + " ThreadID :" + Thread.Current
Thread.ManagedThreadId.ToString();
});

//En VB
Dim vecteurs(MAX) As String
Parallel.For(0, MAX + 1, Function(i) CalculCarre(i, vecteurs))
Function CalculCarre(ByVal i As Integer, ByVal vecteurs() As String) As
Boolean
vecteurs(i) = "Carré : " + (i * i).ToString() + " ThreadID :" + Thread.Current
Thread.ManagedThreadId.ToString()
        Return True
End Function
```

Dans cet exemple, nous utilisons la méthode statique (shared en vb) Parallel.For pour paralléliser notre algorithme, qui est capturé dans le 3e argument de la méthode, par un delegate (delegate, exprimé ici sous forme de lambda expression, nouveauté disponible avec C#3 et VB 9). Notre algorithme de départ reste inchangé, et la bibliothèque tentera de le paralléliser sur plusieurs cœurs.

A ce stade, plusieurs remarques sont à faire :

1) Si vous ne disposez pas de multi-cœur, cette bibliothèque fonctionnera quand même sur des architectures mono-cœur, vous pouvez donc dès à présent l'utiliser et profiter plus tard de la puissance des multi-coeurs

2) Il est probable que vous ne constatiez pas (tout de suite) d'amélioration significative de performance, car cela dépend beaucoup de l'algorithme et surtout du volume de données que vous utilisez. Dans notre exemple, si MAX est très faible, l'initialisation et la finalisation de la bibliothèque peuvent prendre plus de temps que la boucle elle-même. Un seuil est donc à trouver.

Recevez la Newsletter www.programmez.com/inscription.php

3) Bien qu'il soit simple de paralléliser une boucle, il n'en reste pas moins que des mécanismes de verrouillage de la mémoire sont quand même à prévoir pour certains algorithmes.

Il est également possible d'utiliser la méthode statique *Parallel.ForEach* pour itérer en parallèle sur une collection d'objets. Une bonne candidate est la liste générique List<T>.

```
//en C#
List<long> vecteurs = new List<long>();
ReaderWriterLockSlim lck = new ReaderWriterLockSlim();
long cumule=0;
Parallel.ForEach(vecteurs, delegate(long j)
{
   try
   {
      lck.EnterWriteLock();
      cumule += j;
   }
   finally
   {
      lck.ExitWriteLock();
   }
});
//En VB
Dim lck As New ReaderWriterLockSlim
Parallel.ForEach(vecteurs, Function(z) Somme(z, cumule, lck))

Function Somme(ByVal j As Long, ByRef cumule As Long, ByRef lck As
ReaderWriterLockSlim) As Boolean
      Try
         lck.EnterWriteLock()
         cumule += j

      Finally
         lck.ExitWriteLock()
      End Try
      Return True
   End Function
```

Je vous l'accorde, cet algorithme est assez sot, car l'objet " List " possède déjà une méthode d'instance Sum(). Mais il me permet d'introduire une notion importante, la gestion des états.

Dans cet exemple, nous partageons entre chaque thread, la variable " cumule ", qui est incrémentée de " j ". Si nous ne posons pas un verrou en écriture de type *ReaderWriterLockSlim* par exemple (nouveau avec .NET 3.5), le résultat risque d'être incertain. J'attire ici votre attention sur le fait que la gestion de verrou, concept ô combien important dans un environnement parallèle, reste à la charge du développeur. Cette bibliothèque n'apporte rien de nouveau dans ce sens. Les problèmes de concurrences d'accès, d'étreinte fatale, et autres conflits restent bien évidement une réalité.

Exécution de tâches asynchrones

Avec la plate-forme .NET, lorsqu'on souhaite exécuter une ou plusieurs tâches asynchrones, nous disposons de divers moyens. Le thread brut, le gestionnaire de pool de threads, ou le modèle de programmation asynchrone fourni avec les délégués (delegate).

Dans cet exemple, nous allons utiliser la méthode statique *Parallel.Do*, qui nous permet d'exécuter facilement des tâches indépendantes les unes des autres.

```
//En C#
try
 {
    Parallel.Do( () => AfficherMessage("Bonjour"),
                 () => AfficherMessage("et bienvenue"),
                 () => AfficherMessage("au"),
                 () => AfficherMessage("techdays 2008"));
 }
 catch (AggregateException agrEx)
 {
    foreach (Exception ex in agrEx.InnerExceptions)
    {
       Console.WriteLine(ex.InnerException.Message);
    }
 }

static void AfficherMessage(string message)
      {
       Console.WriteLine("Thread ID :" + Thread.CurrentThread.Managed
ThreadId.ToString());
         Console.WriteLine(message);
        //Simule une activité
         System.Threading.Thread.Sleep(1000);
      //Les erreurs sont encapsulées dans une erreur de niveau supérieur
 AggregateException
      throw new ArithmeticException("Erreur ThreadID" + Thread.Current
Thread.ManagedThreadId.ToString());
      }
```

```
//En VB
Try
         Parallel.Do(Function() AfficherMessage("Bonjour"), _
                     Function() AfficherMessage("et bienvenue"), _
                     Function() AfficherMessage("au"), _
                     Function() AfficherMessage("techdays 2008"))
      Catch exAgr As AggregateException
         For Each ex As Exception In exAgr.InnerExceptions
            Console.WriteLine(ex.InnerException.Message)
         Next
      End Try

Function AfficherMessage(ByVal message As String) As Boolean
      Console.WriteLine("Thread ID :" + Thread.CurrentThread.Managed
ThreadId.ToString())
      Console.WriteLine(message)
      'Simule une activité
      System.Threading.Thread.Sleep(1000)
      'Les erreurs sont encapsulées dans une erreur de niveau supérieur
 AggregateException
      Throw New ArithmeticException("Erreur ThreadID" + Thread.Current
Thread.ManagedThreadId.ToString())
```

```
    Return True
  End Function
```

Encore une fois cet exemple est assez idiot et loin de la réalité du terrain. Mais il met en exergue la facilité avec laquelle il est possible d'exécuter des tâches asynchrones, ainsi que la facilité à gérer les exceptions.
En effet, une des notions difficiles à maîtriser en développement parallèle est celle de la gestion des erreurs. Dans notre exemple, au lieu d'arrêter l'exécution à la 1re exception levée, la bibliothèque les agrège dans l'exception " AggregateException " de plus haut niveau. Puis, lorsque toutes les tâches sont terminées, elle lève alors l'exception. Il suffit ensuite de parcourir la collection " InnerExceptions ", pour retrouver les réelles erreurs qui se sont produites dans les threads.

Notion de Tâches (Task) et de Future

Parfois, vous aurez sans doute besoin de maîtriser plus finement la gestion de vos tâches asynchrones, car vos algorithmes seront plus complexes. Il faudra alors faire appel aux classes " Task " et " Future " (Cette dernière dérivant elle-même de la classe Task).

Note : Toutes les méthodes statiques de la classe " Parallel " que nous venons de voir utilisent d'une manière ou d'une autre l'objet " Task ".

Imaginons par exemple le besoin simple d'exécuter des tâches en parallèle qui soient indépendantes les unes des autres, mais dont on à besoin d'agréger les résultats. Il est donc indispensable de mettre une barrière, pour attendre que tous les threads aient achevé leur travail.

```
//En C#
Task[] tasks = new Task[2];
int z = 0;
int w = 0;
 tasks[0] = Task.Create(delegate
 {
    z = Carre(42);
 });
 tasks[1] = Task.Create(delegate
 {
    w = Carre(42);
 });
 //attendre que toutes les tâches aient fini leur travail
 TaskCoordinator.WaitAll(tasks);
 Console.WriteLine("La somme des carrés {0}", w + z);
static int Carre(int a)
      {
         return a * a;
      }
```

```
//En VB
  Dim Tasks(1) As Task
Dim w As Integer = 0
Dim z As Integer = 0
Tasks(0) = Task.Create(Function() Carre(42, z))
Tasks(1) = Task.Create(Function() Carre(42, w))
''attendre que toutes les tâches aient fini leur travail
TaskCoordinator.WaitAll(Tasks)
Console.WriteLine("La somme des carrés {0}", w + z)
Function Carre(ByVal a As Integer, ByRef ret As Integer) As Boolean
  ret = a * a
  Return True
End Function
```

Dans cet exemple, nous instancions 2 tâches qui vont exécuter en parallèle la méthode " Carre() ". Ensuite, nous utilisons l'objet " TaskCoordinator " qui a pour charge d'attendre que toutes les tâches soient finies. Une fois cette garantie obtenue, nous pouvons en toute tranquillité calculer la somme des deux carrés.
Néanmoins, en tant que développeur, nous souhaiterions exprimer notre somme de manière plus naturelle ; exemple Somme=A+B ; sans nous soucier des mécanismes additionnels de synchronisation. L'objet " Future ", va nous aider dans ce sens.

```
//EN C#
Future<int> z=Future.Create (()=>Carre(42));
Future<int> w = Future.Create(() => Carre(42));
Console.WriteLine("La somme des carrés {0}", w.Value  + z.Value);
```

```
//En VB
Dim z As Future(Of Integer) = Future.Create(Function() Carre(42))
Dim w As Future(Of Integer) = Future.Create(Function() Carre(42))
Console.WriteLine("La somme des carrés {0}", w.Value + z.Value)
```

Plus besoin de créer un tableau de tâches, et de mettre en place un mécanisme d'attente, c'est l'objet " Future " et sa propriété " Value() " qui se met en attente de résultat. En d'autres termes, tant que les deux threads exécutant en parallèle la méthode " Carre() " n'ont pas fini leur travail, l'addition de w+z est en attente. Les Futures sont une manière élégante de programmer en asynchrone, en réduisant considérablement le nombre de lignes de code. Plus on réduit le nombre de lignes en programmation parallèle, plus nous avons de chance de réduire également les erreurs. Voici par exemple, comment nous aurions pu faire le même calcul avec le Thread de base de .NET.

```
static private int[] values;
struct UserState
{
   public WaitHandle waitHandle;
   public int index;
   public int value;
}
static void TestAPM()
{
   WaitHandle[] waitHandles=new WaitHandle[2];
   values = new int[2];
   waitHandles[0] = new AutoResetEvent(false);
   waitHandles[1] = new AutoResetEvent(false);
   UserState param1=new UserState ();
   UserState param2 = new UserState();
   param1.waitHandle = waitHandles[0];
   param1.index = 0;
   param1.value = 42;
   Thread t1 = new Thread(new ParameterizedThreadStart(Carre));
```

```
    t1.Name = "Thread1";
    t1.IsBackground = true;
    param2.waitHandle = waitHandles[1];
    param2.value = 42;
    param2.index = 1;
    Thread t2 = new Thread (new ParameterizedThreadStart (Carre));
    t2.Name = "Thread2";
    t2.IsBackground = true;
    t1.Start(param1);
    t2.Start(param2);
    WaitHandle.WaitAll(waitHandles);
    Console.WriteLine("La somme des carrés {0}", values[0] + values[1]);
}

static void Carre(object state)
{
        UserState parametres = (UserState)state;
        values[parametres.index] = parametres.value * parametres.value;
        AutoResetEvent mre =(AutoResetEvent)parametres.waitHandle;
        mre.Set();

}
```

Je ne vais pas détailler l'algorithme, mais sa longueur parle d'elle-même. Vous noterez qu'à la différence de la méthode Carre() utilisée avec la classe Future, chaque instance de la méthode Carre() doit signifier (par l'intermédiaire de l'objet AutoResetEvent) au thread appelant qu'elle a fini son travail, afin que ce même thread puisse continuer.

PLINQ

Nous venons de voir comment à l'aide de programmation impérative nous pouvons développer des applications concurrentielles. Voyons maintenant, comment le faire de manière plus déclarative à l'aide de PLINQ. Avec le framework .NET 3.5, C# 3 et VB 9 arrive une technologie d'interrogation d'objets nommée LINQ (Language Integrated Query), la bibliothèque d'extension parallèle lui emboîte le pas, et propose sa version parallélisée nommée PLINQ (Parallel Linq). Pour bien comprendre la suite, quelques rudiments de LINQ sont nécessaires.

Imaginons le scénario suivant

J'ai une entité nommée Client, organisée de la manière suivante : Client (ID,Nom,Prénom,Age). Dans notre scénario, nous remplissons dans le désordre une liste générique List<Client> de clients et ce que nous souhaitons, c'est pouvoir, en extraire un sous-ensemble qui réponde à un certain nombre de critères. Avec le langage SQL, cette demande pourrait s'exprimer de la manière suivante :

```
"Select Age, Nom, Prénom from [tableClients] where Age > 50 and Age
< 70 Order By Nom"
```

LINQ va alors nous permettre d'interroger la liste de clients dans une syntaxe qui se rapproche peu ou prou du langage SQL.
Par exemple, si je souhaite dans cette liste ne récupérer que les clients ayant un âge compris entre 50 et 70 ans, voici comment cela se traduit.

```
//En C#
List<Client> Clients = EntiteMetier.Client.CreerClient(1000);
var cs = from c in Clients where c.Age >50 & c.Age < 70   select c;

//En VB
Dim Clients As List(Of Client) = EntiteMetier.Client.CreerClient(1000)
Dim cs = From c In Clients Where c.Age > 50 And c.Age < 70 Select c
```

Si je souhaite trier par le nom, je peux y ajouter une clause OrderBy comme ceci :

```
//EN C#
List<Client> Clients = EntiteMetier.Client.CreerClient(1000);
var cs = from c in Clients where c.Age >50 & c.Age < 70 orderby c.Nom
select c;

//EN VB
Dim Clients As List(Of Client) = EntiteMetier.Client.CreerClient(1000)
Dim cs = From c In Clients Where c.Age > 50 And c.Age < 70 Order By
c.Nom Select c
```

Ensuite il est simple de parcourir ce sous-ensemble

```
//En C#
foreach (Client c in cs)
{
Console.WriteLine("Nom {0}, Prenom {1},  Age : {2}", c.Nom, c.Prenom, c.Age);
}

//En VB
For Each cl As Client In cs
Console.WriteLine("Nom {0}, Prenom {1},  Age : {2}", cl.Nom, cl.Prenom,
 cl.Age)
Next
```

Avec PLINQ il sera possible, de manière déclarative, d'exprimer ces mêmes demandes, mais cette fois-ci en exploitant tous les processeurs disponibles et ceci d'une manière extrêmement simple. Il suffit d'utiliser l'extension AsParallel() comme ceci.

```
//En C#
List<Client> Clients = EntiteMetier.Client.CreerClient(1000);
var cs = from c in Clients.AsParallel () where c.Age >50 & c.Age < 70
  orderby c.Nom  select c;

//En VB
Dim Clients As List(Of Client) = EntiteMetier.Client.CreerClient(1000)
Dim cs = From c In Clients.AsParallel() Where c.Age > 50 And c.Age < 70
 Order By c.Nom Select c
```

Conclusion

Comme je le disais en introduction, les concepts et le développement parallèle restent difficiles à maîtriser. Nous aimerions, nous développeurs, exprimer nos algorithmes de manière traditionnelle et que le système fasse le reste. Aujourd'hui ce n'est pas encore le cas, mais avec cette nouvelle bibliothèque d'extension parallèle à .NET, nous nous en approchons à grand pas.

■ **Eric Vernié** - *Microsoft France - Relation Technique Développeurs*

ABONNEMENT .NET

DÉVELOPPEZ VOTRE SAVOIR-FAIRE

Programmez ! est *le* magazine du développement

Langage et code, développement web, carrières et métier : Programmez, c'est votre outil de veille technologique.

Pour votre développement personnel et professionnel, abonnez-vous à Programmez !

Abonnez-vous :

Les Hors-Série .Net

5 numéros par an : **20 €***

soit **1 Numéro GRATUIT**

**Tarifs France métropolitaine*

PROGRAMMEZ HORS SERIE .NET *Seulement*

- **Abonnement 1 an aux Hors Série : 5 numéros : 20 €** (au lieu de 25 € tarif au numéro) *Tarif France métropolitaine*

PROGRAMMEZ

- **Abonnement 1 an au magazine : 45 €** (au lieu de 65,45 € tarif au numéro) *Tarif France métropolitaine*
- **1 an au magazine + 5 numéros HS .NET : 63 €** *Tarif France métropolitaine*
- **Abonnement Intégral : 1 an au magazine + archives sur Internet et PDF : 57 €** *Tarif France métropolitaine*
- **Abonnement PDF / 1 an : 30 €** - *Tarif unique*
  Inscription et paiement exclusivement en ligne
  www.programmez.com
- **Abonnement Etudiant : 1 an au magazine : 39 €** (au lieu de 65,45 € tarif au numéro) *Offre France métropolitaine*

Abonnez-vous :

Le magazine mensuel

11 numéros par an : **45 €**

soit **3 Numéros GRATUITS**

**Tarifs France métropolitaine*

OUI, *je m'abonne*

Vous pouvez aussi vous abonner en ligne et trouver tous les tarifs www.programmez.com

PROGRAMMEZ HORS SERIE .NET NOUVELLE OFFRE

☐ **Abonnement 1 an aux Hors Série : 5 numéros : 20 €** *(au lieu de 25 € tarif au numéro) Tarif France métropolitaine*

PROGRAMMEZ

☐ **Abonnement 1 an au magazine : 45 €** (au lieu de 65,45 € tarif au numéro) *Tarif France métropolitaine*

☐ **1 an au magazine + 5 numéros HS .NET : 63 €** *Tarif France métropolitaine*

☐ **Abonnement Intégral : 1 an au magazine + archives sur Internet et PDF : 57 €** *Tarif France métropolitaine*

☐ **Abonnement Etudiant : 1 an au magazine : 39 €** (au lieu de 65,45 € tarif au numéro) *Offre France métropolitaine*

☐ 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 - 75472 Paris Cedex 10.
abonnements.programmez@groupe-gli.com

Offre limitée, valable jusqu'au 31 mai 2008

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.

Programmez! LE MAGAZINE DU DÉVELOPPEMENT

Votre potentiel, notre passion™
Microsoft
Microsoft
Visual Studio
SATISFAIRE CHACUN, SANS COMPROMIS.
VISUAL STUDIO 2008.
Votre défi : concevoir de meilleures applications pour la bureautique et le web dans les meilleurs délais et presque sans effort.
Votre arme : Visual Studio 2008. Pour gagner en temps, en expérience utilisateur et en qualité, tout simplement.
Plus d'informations sur www.releveztouslesdefis.com
RELEVEZ TOUS LES DÉFIS
© 2008 Microsoft Corporation. Tous droits réservés. Microsoft, Visual Studio, le logo Visual Studio et « Votre potentiel, notre passion » sont des marques de Microsoft déposées et/ou utilisées aux États-Unis et/ou dans d'autres pays.