PROGRAMMEZ !
Le magazine du développement
PROgrammez !
www.programmez.com
mensuel n°130 - mai 2010

Objectif développeurs

Visual Studio 2010 et .Net 4.0

Toutes les nouveautés pour le développeur

Découvrez la nouvelle CLR

Comment migrer son code vers .Net 4 ?

Pex : l'outil indispensable !

ANDROID

- ✔ Gagner de l'argent avec Android
- ✔ Gérer la téléphonie et la géolocalisation

ADOBE Flex 4.0

Cap sur le Multi-plateforme

CARRIÈRE

Finance active recrute
des développeurs JAVA/JEE expérimentés

HTML 5

Écrasera-t-il Flash et Silverlight ?

Réalité augmentée

Les nouvelles interfaces du web

M 04319 - 130 - F: 5,95 €

Printed in France - Imprimé en France - BELGIQUE 6,45 €
SUISSE 12 FS - LUXEMBOURG 6,45 € - DOM Surf 6,90 €
Canada 8,95 $ CAN - TOM 940 XPF - MAROC 50 DH

DÉVELOPPEZ 10 FOIS PLUS VITE

WINDEV®

ENVIRONNEMENT PROFESSIONNEL DE DÉVELOPPEMENT (AGL)

Créez des applications pour Windows, Internet, Mobile et SaaS
Java, .Net, PHP, J2EE, Webservice, XML, Ajax, Linux, Android, Client riche, Base de Données...

AGL N°1 EN FRANCE

Demandez le dossier + DVD gratuit

Elu «**Langage le plus productif du marché**»

Tout est inclus
Gère le cycle complet de développement.
Support Technique gratuit

555 NOUVEAUTÉS

PC SOFT 25e ANNÉE Au service des développeurs

VERSION EXPRESS GRATUITE
Téléchargez-la !

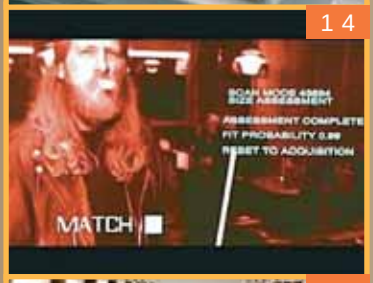
Fournisseur Officiel de la Préparation Olympique

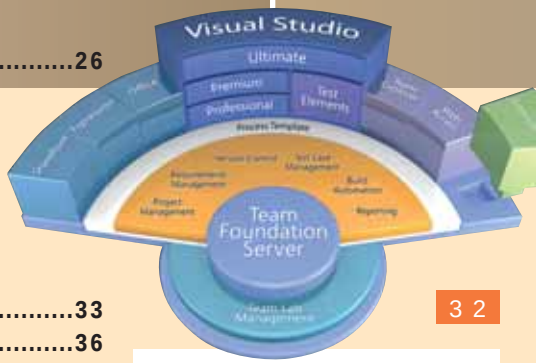
www.pcsoft.fr

Dossier gratuit 200 pages sur simple demande. Tél: **04.67.032.032** info@pcsoft.fr

PROgrammez ! Le magazine du développement
numéro 130 - mai 2010

sommaire





LES CONTRÔLES LES PLUS RAPIDES...

Chez Infragistics, nous nous assurons que nos contrôles pour .Net vous permettent de créer les meilleures interfaces utilisateur possible. C'est pourquoi nous avons testé et retesté nos Data Grids afin de nous assurer qu'elles sont les plus rapides sur le marché et que nos Data Charts surpassent tout ce que vous avez expérimenté. Utilisez nos contrôles et vous obtiendrez non seulement le temps de téléchargement le plus rapide mais aussi des applications qui sont toujours attrayantes. Rapide et belle... C' est une « Killer app ». Essayez les vous-même sur **infragistics.com/wow**.

Infragistics
KILLER APPS. NO EXCUSES.

Infragistics Ventes France N° Vert 0800 667 307
Infragistics Europe Ventes +44 (0) 800 298 9055
twitter.com/infragistics

éditO

Natif tu seras... ou pas

Depuis que je fais de l'informatique, c'est-à-dire depuis maintenant 25 ans (sur 36, cela donne un bon ratio), j'ai vu, entendu, touché de nombreux débats, discussions, révolutions qui tombèrent à plat quelques mois plus tard. L'avantage d'être un gentil journaliste informatique est de pouvoir observer notre univers de silice par le haut. Et ce mois-ci, j'avoue qu'un débat m'intrigue... mais ne me surprend pas.

On parle beaucoup de programmation multicore, de parallélisme, à juste titre. Mais une partie du débat me semble biaisée ou mal posée. Car trop souvent, on oublie une dimension essentielle : quel langage utiliser ? Faut-il passer nativement ou par un langage managé, donc fondamentalement interprété ? Et cela m'a sauté aux yeux lors de l'après-midi « programmation parallèle » organisée par Intel et Microsoft. Une opposition réelle entre les deux mondes. Serait-on revenu au débat : le seul langage « couillu » serait C ou C++, le reste servant à « faire mumuse » ou jouer aux « legos », bref un langage pour bisounours... ? Bête et futile comme digression ? Pas forcément.

Pourtant, le parallélisme remet le débat en selle et de manière cruciale. Pour coder multicore, seul le C++ est-il pertinent ? Reprenons le problème. On peut paralléliser du code managé de type C#, VB, Java... par des librairies, des outils notamment en multithreadant massivement le code et en jouant sur les tasks. Mais l'interprété reste de l'interprété et le socle d'exécution restera la faiblesse du langage et du code. Oui, on peut gagner en performances mais peut-on, sans rire, comparer les performances d'un C, C++ avec un VB, C# ou Java ? Je vais être limpide : NON, NON et NON. Arrêtons l'illusion. Même si j'entends souvent le contraire, mais soyons sérieux durant quelques latences de cycles.

Les langages natifs restent et resteront les langages de référence pour les performances pures, l'optimisation. À la rigueur, un langage fonctionnel peut offrir des performances de haut niveau. Mais franchement, qui voudrait coder un calcul massif HPC en Java ou en PHP ? Et même sur l'informatique embarquée et mobile, le natif est la seule solution pour la performance. D'autre part, les possibilités de parallélisme en interprété sont limitées par rapport à un C, aussi bien au niveau langage qu'outillage.

Cependant, une 3e voie s'offre aux développeurs : le mixage interprété et natif. Il s'agit de paralléliser les deux mais de garder un cœur de code en interprété quand celui-ci l'est déjà, de paralléliser ce qui peut l'être mais pour des fonctions très complexes, très lourdes, passer par un code natif. Oui cette approche complexifie le code mais les gains attendus justifient l'effort. Et si on va plus loin, c'est toute l'architecture logicielle que l'on peut repenser. Car le parallélisme ne prend sa valeur que si on parallélise toutes les couches : données, threads, tasks... En effet, le multicore peut profiter à différentes parties du code et pas uniquement à une seule. Mais cela impose un travail de réflexion. L'architecture parallélisée n-tiers sera-t-elle l'avenir de l'application ? J'avoue que j'en suis désormais convaincu.

Je conclurai par un bref slogan : pensez parallèle !

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

Le magazine du développement
PROgrammez!
www.programmez.com

Rédaction : redaction@programmez.com
Directeur de la Rédaction : Jean Kaminsky.
Rédacteur en Chef : François Tonic - ftonic@programmez.com. **Ont collaboré à ce numéro :** F. Mazué, F. Dewasmes, G. Delamarre, A. Gentet.
Experts : G. Hamrouni, G. Le Fur, M. Chaize, Ph. Daucourt, F. Lavocat, J. de Oliveira, C. Heral, J. Dollon, V. Bellet, J. Carnelos, F. Gerlier, F. Queudret, V. Schiopu, E. Jambou, D. Guignard, O. Penhoat, C. Villeneuve, N. Guilbert, A. Habert.
Illustration couverture : © iStockphoto/Andrew Roche
Publicité : Régie publicitaire, K-Now sarl**.** Pour la publicité uniquement : Tél. : 01 41 77 16 03 - diff@programmez.com. Editeur : Go-02 sarl, 21 rue de Fécamp 75012 Paris - diff@programmez.com. Dépôt légal : à parution - Commission paritaire : 0712K78366 ISSN : 1627-0908. Imprimeur : S.A. Corelio Nevada Printing, 30 allée de la recherche, 1070 Bruxelles Belgique.
Directeur de la publication : J-C Vaudecrane
Ce numéro comporte un encart libre Microsoft sur tous les exemplaires

Abonnement : Programmez 22, rue René Boulanger, 75472 Paris Cedex 10
Tél. : 01 55 56 70 55
abonnements.programmez @groupe-gli.com
Fax : 01 40 03 97 79 - du lundi au jeudi de 9h30 à 12h30 et de 13h30 à 17h00, le vendredi de 9h00 à 12h00 et de 14h00 à 16h30. **Tarifs** abonnement (magazine seul) : 1 an - 11 numéros France métropolitaine : 49 € - Etudiant : 39 € - CEE et Suisse : 55,82 € - Algérie, Maroc, Tunisie : 59,89 € - Canada : 68,36 € - Tom : 83,65 € Dom : 66,82 € - Autres pays : nous consulter. **PDF :** 30 € (Monde Entier) souscription exclusivement sur www.programmez.com

PROCHAIN NUMÉRO

N°131 juin 2010 parution 29 mai

✔ **Webmaster**
La VIDEO sur le web :
choisir le format, la technologie

✔ **Enquête**
PHP est-il mort ?

✔ **Cloud Computing**
Mes premiers pas avec
Windows Azure

■ **RIM** (fabricant du Blackberry) a annoncé le rachat de QNX, un des éditeurs mondiaux sur les systèmes temps réel.
Cette annonce étonne, tout comme l'accent mis lors de la présentation du rachat sur le marché automobile, de là, à dire que RIM veut conquérir la voiture...
QNX possède aussi une autre perle technique : l'interface Photon. A suivre de près !

■ **James Gosling** a quitté Oracle début avril ! C'est un nouveau départ d'importance depuis le rachat de Sun par Oracle. La communauté Java était secouée depuis un an par les retards de version, des modifications à la hâte de clauses dans la licence d'utilisation.
Et ce départ ne va pas calmer les esprits, surtout que James n'explique pas les raisons, mais visiblement le rachat n'a pas dû arranger les choses, bien au contraire. James aimait sa liberté de parole et d'action, ce qui n'est pas forcément très compatible avec l'esprit Oracle. Ce départ-surprise et rapide, renforce le camp des sceptiques sur le rachat de Sun par Oracle et surtout sur la conduite des nombreux projets open source de l'éditeur. Nous ne savons pas non plus si James gardera un rôle dans la communauté Java et de quelle manière. Le nouveau blog de James :
http://nighthacks.com/roller/jag/entry/time_to_move_on

■ **Windows Phone 7** est aussi une plate-forme de jeux (si, si). Et cela passe par le portage du Platformer Starter Kit XNA 3.1 vers XNA 4 pour Windows Phone 7 justement.
Cette annonce est importante pour pousser la nouvelle plate-forme mobile et surtout attirer les développeurs... pour tout savoir :
http://blogs.msdn.com/davrous/archive/2010/03/29/portage-du-platformer-starter-kit-xna-3-1-vers-xna-4-0-pour-windows-phone-7.aspx

iPhone OS 4.0 : Apple veut préserver son avance

Au-delà du lancement en France de l'iPad et donc de son SDK, Apple a dévoilé la 4e version majeure du système iPhone OS qui va animer dès cet été le iPhone et en automne le iPad en personne. Pour rappel : le iPhone c'est aujourd'hui 185 000 applications, 100 millions de terminaux d'ici fin 2010, des milliers de développeurs et plus de 4 milliards d'applications téléchargées !

Une pré-version est disponible depuis mi-avril pour les développeurs iPhone enregistrés. Il nécessite toujours un Mac sous MacOS X pour développer les applications. D'autre part, la licence Apple a été modifiée pour autoriser uniquement les applications en langage Objective-C ou HTML, JavaScript. Cela exclut donc a priori les environnements de développement tiers comme la moulinette Flash pour iPhone ou encore MonoTouch. Un grand débat s'est fait jour sur cette limitation. Apple argue qu'il s'agit d'apporter de la valeur ajoutée aux applications tournant sous iPhone et non pas de créer un contenu uniforme partout sans exploiter le potentiel de la plate-forme. Cette position se défend quoi qu'on puisse en penser. Mais ce sont les développeurs, et les utilisateurs finaux, qui auront le dernier mot.

Enfin du multitâche

iPhone OS 4 inclut plus de 1 500 nouvelles API et API améliorées : calendrier, librairie photo, aperçu rapide, SMS, accéléromètre, mapping, etc. On bénéficiera d'un framework pour accélérer les opérations mathématiques. On pourra utiliser un clavier sans fil bluetooth... Et surtout le très attendu multitâche sera bien présent permettant d'exécuter des opérations, des fonctions d'applications en arrière-plan ! Un des problèmes sera de préserver l'autonomie du téléphone par rapport au multitâche et les développeurs devront faire un travail de finesse dans son implémentation. Reste à voir ce que cela va donner à l'usage. Et le dock a été revu pour intégrer le multitâche et l'accès à différentes applications. Sept services seront capables de fonctionner en multitâche. Les logiciels de voix sur IP pourront aussi en bénéficier. Et la partie géolocalisation pourra profiter de fonctionner en arrière-plan... Encore une fois, attention à la gestion de l'énergie !
Deuxième nouveauté : les dossiers ! On pourra y ranger des applications par type, préférence... mais visiblement pas encore pour les fichiers, documents, dommage... Sur iPad ce serait pratique ! La partie mail a été améliorée. C'était une des critiques (justifiées) de l'iPhone actuel. On pourra par exemple se connecter à deux Exchange Server contre un seul aujourd'hui. Autre faiblesse, la protection des données. Cela passe aussi par une gestion du terminal, un déploiement de l'application par wifi. On bénéficiera aussi du support d'Exchange Server 2010 et du SSL, VPN. Mais plus important encore, Apple se lance dans la bataille de la publicité mobile, avec iAd ! Notons aussi que Apple met en avant, notamment dans les publications, HTML 5 pour la partie animation et vidéo (absence de Flash oblige diront certains). ■

MODEL2CODE

Agile Development in action!

CERTAINS AGL VOUS PROMETTENT DE DÉVELOPPER 10 FOIS PLUS VITE?

ET SI VOUS VOUS LAISSIEZ... DIRIGER PAR LES MODÈLES ?

NE DÉVELOPPEZ PLUS, MODÉLISEZ !

Passez à la vitesse du Model Driven!

Maquettez vos IHMs en HTML ou MXML et modélisez simplement vos processus métier sous forme de diagrammes UML, nos générateurs les transforment instantanément en application Spring, Java EE, Flex ou Improve Foundations, prête à être déployée !

Nouveau ! Plugin CRUD Booster

Créez en quelques minutes vos prototypes et accélérez considérablement vos projets applicatifs. Générez une application CRUD fonctionnelle (y compris les IHMs) à partir d'un simple diagramme de classe !

VERSIONS D'ÉVALUATION GRATUITES & TUTORIAUX :

WWW.MODEL2CODE.COM

Model2Code propose des ateliers agiles et collaboratifs de génération d'applications web et riches, combinant les meilleures technologies Model Driven du marché : l'outil de modélisation UML Magicdraw® associé au moteur de génération par transformation de modèles BLU AGE®.

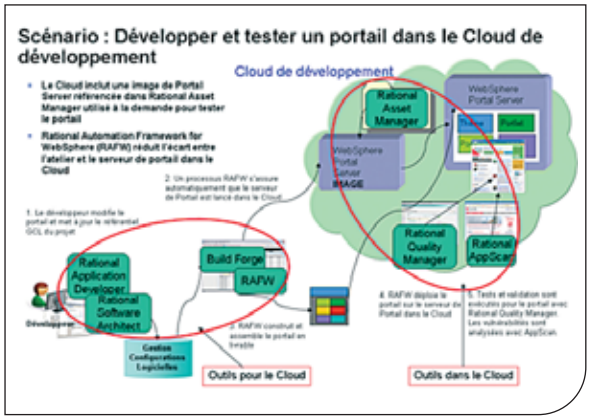
magicdraw™ BLU AGE AGILE MODEL TRANSFORMATION

BLU AGE Software
Immeuble le Gabriel Voisin - 79, rue Jean-Jacques Rousseau 92158 Suresnes cedex FRANCE.
Tel +33 1 56 05 60 91 Fax +33 1 56 05 88 01

www.model2code.com

■ **Obeo et SonarSource** annoncent la sortie d'un plugin Sonar pour VisualBasic 6. Il s'agit d'une solution d'analyse automatique de la qualité du code d'applications VisualBasic 6. La sortie est prévue au début du mois de mai. La première version du plug-in permettra de calculer les métriques de base et disposera de fonctionnalités avancées comme le calcul de la complexité cyclomatique ou bien encore la recherche de code dupliqué. Cette version disposera également d'un moteur de règles de codage spécialisé pour Visual Basic. En parallèle de la sortie de **Sonar 2.1**, ce nouveau plug-in viendra enrichir l'offre de SonarSource. Le modèle de commercialisation du plug-in consistera en un abonnement annuel incluant la maintenance, le support et les mises à jour.

IBM lance Rational Software Delivery Services for Cloud Computing



IBM fournit une plate-forme de gestion du cycle de vie qui s'appuie largement sur le socle technique Jazz. Spécialement conçu pour les équipes globales et géographiquement distribuées, Jazz transforme la façon dont les équipes travaillent pour fabriquer du logiciel et des systèmes, en rendant cette activité plus collaborative, plus productive et surtout plus transparente. On peut voir la plate-forme Jazz comme un framework extensible qui intègre et synchronise dynamiquement les personnes, les processus et les actifs produits sur les projets de développement et de livraison de système et de logiciels. Le but des services Cloud Rational est de fournir des offres de produits virtualisés, provisionnés directement par l'utilisateur. Le processus de déploiement inclut les meilleures pratiques, et ne requiert pas d'intervention humaine. Dans une première étape, IBM Rational met à disposition une collection de services dans le cloud, provisionnés sous forme d'images virtuelles prêtes à l'emploi. Ces services sont au nombre de cinq : Services de Gestion de la Qualité, Services de Développement Agile, Services d'Automatisation des fabrications, Services de Gestion des Actifs et de Gouvernance et Services de Gestion Collaborative des Exigences. Dès la fin de l'année 2010, Rational SDS (Software Delivery Service) fournira un environnement de collaboration complet pour le développement et le test d'application sur le cloud. Il permettra de mettre en œuvre une solution collaborative de gestion du cycle de vie en quelques minutes en réduisant les coûts d'adoption et le niveau d'engagement nécessaire à la mise en place d'une telle plate-forme. Ces services ont été présentés lors de la conférence utilisateur de juin 2009, ils sont disponibles sous forme de Bêta depuis octobre 2009 et accessibles depuis le 1er avril 2010. IBM Rational prépare un cloud public d'ores et déjà accessible sous forme de Bêta.

IBM Rational propose également des outils et des services pour le cloud afin d'aider les entreprises qui souhaitent y déployer des solutions. Les outils de modélisation et de développement permettent de concevoir des applications pour le cloud. Les outils d'automatisation permettent le déploiement d'applications. Les outils de tests aident à s'assurer de la qualité, de la sécurité et de la montée en charge des applications avant leur déploiement dans le cloud.

■ Fabrice **Gerlier**
(Architecte IBM Rational France).

agenda \\

MAI 2010

• Du 05 mai au 06 mai 2010, CNIT, Paris La Défense, Salons **Solutions Data Center Management et Cloud Computing**, dédiés à toutes les préoccupations des responsables des SI. http://www.datacenter-cloud.com

• Jeudi 6 mai 2010, à l'Ecole Supérieure d_Informatique et Applications de Lorraine, 193 av. Paul Muller, 54602 Villers-lès-Nancy **En mai, fais ce qu'il te Play!**. http://jugevents.org/jugevents/event/26009

• jeudi 06 mai 2010, Paris la Défense, Tour Ariane. InterSystems organise un **atelier pratique** sur ses solutions : **Ensemble** et la base de données **Caché**. http://www.intersystems.fr/page/fr/decouverte_technologies_intersystems.html

• jeudi 06 mai 2010, Paris 9e : **Alfresco Meetup 2010,** Alfresco Software organise plusieurs journées d'échange avec l'ensemble de sa communauté en Europe. http://www.alfresco.com/about/events/2010/05/paris_meetup/

• Le 11 mai 2010, Amphithéâtre de l'ESMA - Montpellier Aéroport, **Michael Chaize d'Adobe présente Flex 4 et AIR 2.0**. http://groups.adobe.com/groups/1249cb9946/summary

ETRANGER

• Du 02 au 05 Mai, Valence, Espagne, **3e EMEA Partner Summit de Red Hat et JBoss**. http://www.europe.redhat.com/mktg/partnersummit/

• Du 16 au 20 mai 2010, Mandalay Bay Resort & Casino de Las Vegas, **CA World 2010,** http://www.ca.com/caworld/

Journée française des Tests Logiciels : une réussite !

Si le marché français de l'informatique reste atone malgré une très légère hausse, les tests et la qualité logicielle semblent (enfin) décoller, avec d'après une étude PAC récente : + 9 %. Il faut dire que sur le domaine, on part de très bas ! Pour cette nouvelle journée du test, se déroulant le 30 mars dernier à Paris, plus de 300 inscrits payants ! Joli score après les 150 lors de la journée de juin 2008. La journée fut ponctuée de conférences plus ou moins pointues sur le test et des retours d'expérience. La partie exposition était bien garnie avec des éditeurs comme IBM, Microsoft, Smartesting, Neotys, HP mais aussi de nombreuses SSII ayant un service tests comme Sopra, Steria ou encore EGL pour la formation et la certification. La journée a aussi permis aux personnes présentes de considérer le test non plus comme un coût mais comme un facteur d'amélioration de son IT, de la qualité du logiciel et finalement une amélioration du retour sur investissement et de la rentabilité du développement. Mais il reste encore beaucoup à faire.

FusionCharts | à partir de € 142

InfoSoft Global

Diagrammes interactifs et animés pour les applications Web et les applications de bureau.

- Animez vos applications Web avec les diagrammes Flash animés
- Créez des diagrammes compatibles AJAX pouvant changer côté client sans requêtes serveur
- Exportez les diagrammes en tant qu'images/PDF et les données en CSV pour les rapports
- Créez des jauges, des diagrammes financiers, de Gantt, en entonnoir et plus de 550 mappages
- Utilisé par plus de 15 000 clients et quelques 250 000 utilisateurs dans 110 pays

ActiveReports 6 | à partir de € 498

GrapeCity

Dernière version du générateur de rapports .NET sans droits le plus vendu.

- Moteur de génération de rapports rapide, souple avec licence sans droits pour le Web/Windows
- Visualisation des données et contrôles de mise en forme: Chart, Barcode et Table Cross Section
- Prise en charge de Medium Trust dans l'environnement ASP.Net
- Vaste gamme de formats d'exportation/aperçus: Windows Forms Viewer, Web Viewer, Flash & PDF
- Contrôle utilisateur de conception avec API souple et personnalisation

SQL Developer Bundle | à partir de € 975

redgate

Ensemble d'outils essentiels pour tous les développeurs de BD.

- Dix outils salués par la critique qui éliminent les pensums SQL
- Rapides, précis et testés constamment pour vous faire gagner du temps
- Inclut SQL Compare Pro, SQL Data Compare Pro, SQL Prompt Pro, SQL Data Generator, SQL Packager, SQL Refactor, SQL Dependency Tracker, SQL Doc, SQL Multi Script, SQL Servers et SQL Comparison SDK

ComponentArt Web.UI for Silverlight | à partir de € 570

ComponentArt

Une suite complète de commandes de ligne métier ou de visualisation pour Silverlight.

- Grille de données gérant des millions d'entrées, nombreux styles intégrés
- Contrôles graphiques avec rendu exceptionnel, API puissante côté client
- Contrôles de navigation conciliant fonction et style
- L'éditeur offre une expérience WYSIWYG et inclut les retours d'utilisateur en temps réel
- Arborescence à édition en ligne, glisser-déposer intégré et sélection de nœuds multiples

Siège social en Europe
ComponentSource
30 Greyfriars Road
Reading
Berkshire
RG1 1PE
Royaume-Uni

Siège social aux États-Unis
ComponentSource
650 Claremore Prof Way
Suite 100
Woodstock
GA 30188-5188
États-Unis

Siège social au Japon
ComponentSource
3F Kojimachi Square Bldg
3-3 Kojimachi Chiyoda-ku
Tokyo
Japon
102-0083

Numero vert:
0800 90 92 62
www.componentsource.com

Nous acceptons les bons de commande. Contactez-nous pour demander un compte de crédit.

VISA

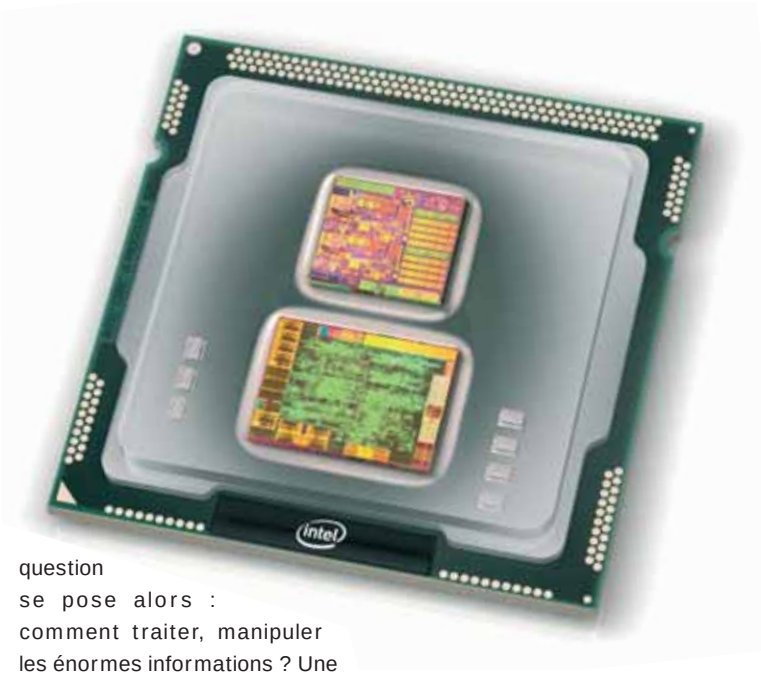
■ La gamme **Visual Studio Express** (gratuite) s'aligne sur le grand frère Visual Studio 2010 et surtout, dévoile une toute nouvelle gamme : Express Web, Express Windows, Express Phone et Express Database. Mais, l'éditeur garde les éditions VB, C# et C++. La version pour Windows Phone est une surprise mais pour Microsoft, il s'agit de favoriser dès maintenant le développement d'applications mobiles Windows Phone 7, même si le produit ne sera pas disponible avant plusieurs mois. C'est une annonce stratégique et vitale pour la partie mobile de Windows, surtout face à Android, iPhone et autre iPad ! Cependant, il s'agit encore d'une pré-version. Elle inclura (autre surprise) XNA Game Studio, atelier dédié au développement de jeux ! Site : http://www.microsoft.com/express/Default.aspx

■ **Linagora** a fêté son 10e anniversaire il y a quelques jours. Pour l'événement, Linagora, spécialisé dans l'open source, a mis à l'honneur l'incubateur dont elle est issue, celui de Télécom Ecole de Management et Télécom Sud-Paris qui depuis sa création, veut donner envie aux étudiants d'entreprendre, de créer ! Alexandre Zapolsky (Linagora) parrainera le challenge 2010.

■ **Oracle** a finalement sorti un patch de sécurité, et bien plus rapidement que prévu, pour Java 6. La faille découverte était particulièrement dangereuse, faille de type 0-day. L'éditeur avait initialement mis en place une politique de patch de sécurité trimestriel, mais l'affaire a pris une telle importance qu'Oracle a finalement réagi rapidement. Ouf !

Intel Software conforte la programmation parallèle

Mi-avril, la division logicielle d'Intel a organisé sa conférence européenne annuelle centrée sur le logiciel, les nouveautés, les tendances. Cette année, comme dans l'édition 2009, l'accent fut le multicore, le multicore et... le multicore !



Cap sur Visual Studio 2010

Pour l'occasion, Microsoft était très présent avec la sortie de Visual Studio 2010 et des couches parallèles disponibles dans l'IDE et .Net 4. Intel a tenté de marteler un message simple et clair : penser parallèle dès maintenant car le multicore est là et le restera. Mais comment inciter le développeur à s'y mettre. Pour cela, les différents intervenants ont présenté les divers modèles de développement, les opportunités du marché mais aussi les difficultés pour réussir un bon parallélisme car ce n'est pas tout de vouloir mettre du parallèle dans son code, il faut bien le coder... L'auditoire fut particulièrement intéressé par une architecture logicielle parallèle sur un modèle n-tiers. Autrement dit, il ne suffit pas de paralléliser uniquement la donnée ou un morceau de code mais il faut paralléliser en même temps. Cela permettrait de mieux exploiter les architectures matérielles et d'optimiser considérablement les traitements. Nous vous conseillons de regarder de plus près cette approche originale.

La donnée concentre un problème crucial à ne pas oublier. Une question se pose alors : comment traiter, manipuler les énormes informations ? Une seule solution : paralléliser le traitement. Intel a une réponse : le projet Ct qui devrait arriver en version finale fin 2010 ou courant 2011. Mais surtout, Intel confirme l'orientation purement C et C++ du parallélisme et de ses outils et tout particulièrement la suite Parallel Studio qui sera mise à jour pour Visual Studio 2010. Mais surtout, Intel a confirmé sa volonté de proposer des outils communs, compilateurs et librairies à Windows, Linux et MacOS X, sans oublier l'embarqué et le mobile. Courant 2010, l'éditeur sortira Parallel Advisor.

Simplifier le parallèle sur C

Autre bonne nouveauté, la disponibilité de Cilk (issu de la recherche) courant 2010. Il s'agit ici de paralléliser facilement du code en C et C++, un peu comme le propose déjà Microsoft avec PLinq ou encore TPL (.Net 4). Pour la partie compilateur parallèle, on aura le projet SIMD. Autre nouveauté bienvenue, un SDK pour la technologie OpenCL. D'autre part, Intel travaillerait à un langage permettant de faire automatiquement la bascule entre GPU et CPU, en attendant les processeurs hybrides. Actuellement, cette bascule se fait au prix de librairies et de codages lourds et complexes...

Aujourd'hui, très peu de développeurs utilisent des instructions, des librairies de parallélisme, d'où un décalage entre le matériel actuel et le logiciel mais d'ici 18 mois, l'ensemble des périphériques (mobiles, netbooks, PC) auront des processeurs multicore ; reste le parc informatique existant qui utilise majoritairement des processeurs monocore...

Java n'est plus le roi des langages !

Mesurer l'usage d'un langage est une tâche difficile avec une bonne partie d'hypothèse. L'index de Tiobe Software publie régulièrement le baromètre des langages. Et pour l'édition d'avril 2010, l'index Tiobe réserve quelques surprises. Il confirme la baisse régulière de Java : celui-ci perd sa première place pour 0,007 % face au C qui, lui, confirme son renouveau. Et malgré une baisse de son usage, C++ conserve, sans surprise la 3e place, devant PHP. Basic subit une nouvelle forte baisse, preuve de son désintérêt progressif. Par contre, on peut noter la bonne tenue de C# qui progresse légèrement et grimpe à la 6e place (7e en 2009). En revanche, Python perd 1,88 % et passe 7e, seulement 7e pourra-t-on dire. Comme le note Tiobe, deux surprises : Objective-C grimpe de 31 places pour arriver 11e (succès iPhone oblige) et l'arrivée en fanfare de Google Go, directement 15e mais qui reste très marginal (0,7 % d'utilisation contre 2,28 % pour Objectif-C ou 2,22 % pour Ruby). Les langages orientés objets sont largement majoritaires avec 54,2 % de l'utilisation globale. À l'opposé, les langages fonctionnels pèsent un tout petit 2,7 %. Notons aussi une autre constante de l'étude, les trois premiers langages pèsent 45 % ! Ce qui est énorme. Mais cela confirme une fois de plus l'extrême fragmentation du marché. Bien entendu cela ne prend pas en compte la diversité des situations dans les pays.

Pour en savoir plus :
http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

VMware prépare le projet Napa

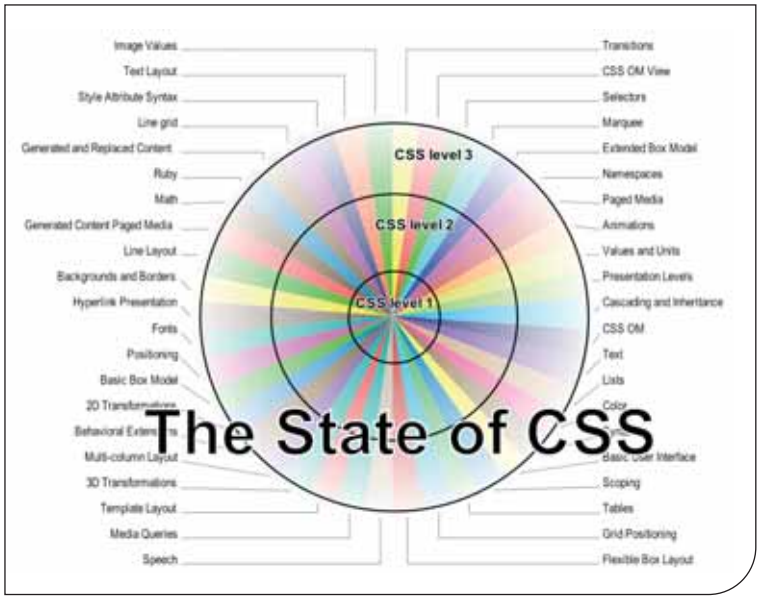
Depuis le rachat de SpringSource par VMware, l'éditeur de Spring travaille beaucoup sur la partie cloud. Un des axes concerne la virtualisation des applications Java (prendre une application java et l'exécuter directement dans le cloud, qu'il soit privé ou public). Il sera possible de combiner les briques techniques de SpringSource côté Java et de CloudFoundry côté service cloud. C'est le projet Napa que l'on aperçoit ici ou là ! À suivre ! Car finalement, la possibilité d'exécuter directement une application dans un cloud est une fonction plus que nécessaire !

Intel teste en ligne

Intel propose Intel Parallel Universe, un outil pour développeur permettant de tester les performances et la montée en charge de son application sur le web. C'est gratuit et assez intuitif ! Il supporte les applications C et C++ en 32 et 64-bit Windows XP et Vista. Mais l'application ne doit pas avoir d'interaction utilisateur... Et des rapports croisés avec Parallel Amplifier peuvent se réaliser. Site : http://paralleluniverse.intel.com

HTML 5 : le tueur de RIA !

Le 7 avril dernier, le W3C organisait une conférence à Paris, une première. Occasion pour le consortium de revenir sur une étape importante du web : HTML 5. Avec ce nouveau HTML, le W3C veut reprendre la main sur les éditeurs qui concentrent beaucoup leurs efforts sur le développement web et les plates-formes riches de type RIA. HTML 5 pourra-t-il proposer une alternative standard aux Flash, Silverlight, JavaFS, Ajax ?



HTML 5 concerne aussi bien l'éditeur d'outils, le développeur, que les fournisseurs de moteurs web comme WebKit ou Mozilla avec Gecko. Lors de la présentation de l'iPhone OS 4, Apple a montré des pubs entièrement réalisées en HTML 5 comme s'il s'agissait de la technologie future pour le web et les animations, à la place de Flash ! HTML 5 doit proposer les fonctions, animations, transitions, présentations que l'on peut avoir avec une plate-forme RIA, tout en prenant en compte l'existant HTML et intégrer SVG, CSS, MathML, la vidéo, la géolocalisation, etc. Et HTML 5 n'est que la face visible des profonds tournants que le W3C veut proposer avec CSS 3 (ultra-complet), WebGL, Web Storage, ECMAScript 5, SVG, etc.

HTML 5 : une grande avancée, mais pour quand ?

Depuis plus de 6 ans, HTML 5 est en gestation, et en discussions. Si les premiers supports furent timides dans les navigateurs, les derniers commencent à passer à la vitesse supérieure, notamment avec les versions en développement. Microsoft avec IE9 propose un support étendu, avec SVG et accélération matérielle. Apple, Google, Mozilla, Opera investissent sur HTML 5. Reste à savoir quand la spécification sera totalement fixée et publiée. Et il faut encore plusieurs années avant un support total des outils de développement, dans les navigateurs et une migration des sites actuels...

De plus, HTML5 est l'héritier de HTML et non XHTML 2 qui est mort en juin 2009, même si HTML5 conjugue la sérialisation HTML et XML, ce qui permettra une compatibilité et le multilingue ainsi que la structure HTML ou XHTML des documents. CSS 3 constitue aussi une sacrée évolution avec une panoplie fonctionnelle sans précédent : polices, transformations 2D et 3D, transitions, Ruby, requête média, parole, etc. Autre point vital : SVG. Ce format est actuellement en version 1.2 depuis 2009. Longtemps applaudi, il fut banni par plusieurs éditeurs avant de revenir à la mode.

CSS 3 va très loin dans les couches textuelles, les sélections, les transitions, les apparences. Les possibilités sont effectivement impressionnantes, reste à posséder les bons outils et une solide expérience. Surtout, CSS 3 veut offrir une expérience, la plus universelle possible, quel que soit le navigateur. Mais l'autre intérêt de HTML 5 est aussi la génération du SVG. Couplé au CSS 3 et aux fonctions vidéo, on possède toute une plate-forme vidéo, de diffusion directement embarquée dans le navigateur sans le moindre plug-in ! Et n'oublions pas le support de MathML qui permettra d'utiliser plus simplement des formules mathématiques sur le web...

Et après ?

Cependant, plusieurs défis restent à relever : interopérabilité, bon support des spécifications par les éditeurs, performances. Et HTML5 se bat aussi contre des concurrents redoutables et plus matures : Flash, Silverlight, voire JavaFX. Le problème de performance est crucial à résoudre mais comment ? Par le moteur de rendu ? Par la GPU ? Par le multicore ? Là, le W3C ne peut travailler seul, les éditeurs doivent fournir des réponses. D'autre part, la mobilité, l'internet mobile, les téléphones nouvelle génération bousculent aussi l'usage du web. HTML5 et le W3C doivent considérer ce marché et ses usages comme la géolocalisation et l'intégration des multiples terminaux et accessoires (micro, photo, Smartphone, PC, téléviseur, etc.).

François **Tonic**

HTML 5 pour fin 2010 ?


P. Le Hégaret


D. Hazaël-Massieux

Interview express avec P. Le Hégaret (Directeur du domaine Interaction, W3C) et D. Hazaël-Massieux (Initiative Web Mobile, W3C)

■ Propos recueillis par F.T.

Programmez : *HTML 5 est maintenant en gestation depuis plus de 6 ans, quand peut-on espérer une version finale des spécifications ?*
W3C : Le W3C a commencé à travailler sur HTML5 depuis fin 2007. Le groupe de travail espère résoudre les problèmes de la spécification d'ici la fin de l'année, avant de passer en phase de tests.

P ! : Et la partie vidéo, qu'en est-il aujourd'hui ? Est-elle plus complète que dans les premières versions ?
W3C : HTML5 est la première version d'HTML qui inclut le support pour la vidéo et cette fonctionnalité est la plus implémentée par les navigateurs.

P ! : Les éditeurs ont-ils fait pression pour retirer cette partie ?
W3C : Non, les éditeurs n'ont pas fait pression.

P ! : SVG est passé par beaucoup de phases : format adoré puis banni. Microsoft va désormais le supporter… Adobe qui n'est pas toujours très clair sur le format, etc. Que représente réellement SVG dans le futur du web ?
W3C : SVG va être implémenté par tous les navigateurs Web majeurs et son déploiement à grande échelle est maintenant quasi certain.

P ! : Un des soucis concerne tout de même les performances. Les versions de développement des navigateurs proposent des résultats assez intéressants. Comment aujourd'hui peut-on améliorer les performances d'affichages et d'interprétation des pages web sur le navigateur ? Cela passe-t-il uniquement par le moteur de rendu, l'usage de GPU, le multicore ?
W3C : Au niveau du rendu graphique, les principaux gains de performance se situent en utilisant les possibilités matérielles de la plate-forme telles que le GPU et les processeurs spécialisés dans le décodage vidéo.

P ! : HTML 5 est parfois vu comme un Flash ou un Silverlight killer, n'est-ce pas plutôt un complément ?
W3C : HTML5 couvre les cas d'utilisation pour lesquels des plugins de Flash ou de Silverlight sont actuellement nécessaires. Donc, même si HTML5 ne va pas remplacer toutes les fonctionnalités qu'ont Flash ou Silverlight, y compris dans le support vidéo, l'intérêt d'utiliser ces plugins va s'amenuiser sur le long terme. Les logiciels de création de contenu ne permettent pas encore de produire du HTML5.

P ! : Concernant les développeurs, cela oblige à apprendre de nouvelles fonctions, comment se fera le passage vers HTML 5 ? Cela va-t-il nécessiter une réécriture ?
W3C : Non, il ne faudra pas tout réécrire. Les nouveautés d'HTML5 s'intègrent dans les technologies développées au W3C depuis 15 ans. Les développeurs vont devoir juste apprendre les nouvelles fonctionnalités existantes dans HTML5, comme la balise vidéo et les nouvelles API.

P ! : L'autre problème est le bon support par les navigateurs. Comment allez-vous veiller à la bonne conformité des navigateurs et moteurs de rendu ?
W3C : Le travail du sous-groupe "testing" du groupe de travail HTML5 vient juste de démarrer. L'interopérabilité du Web entre les différents navigateurs et terminaux dépendra des résultats de ce groupe.

■

actu web

WebKit2 : vive les processus séparés dans le moteur !

La future évolution majeure de WebKit s'appelle WebKit2. Rappelons que le moteur WebKit anime de nombreux navigateurs (desktop et mobile) : Safari, Chrome pour citer les plus importants. Une des grosses nouveautés est l'apparition directement dans le moteur de la séparation des processus en onglets, comme le fait Chrome actuellement. Mais l'équipe WebKit insiste, contrairement à Chrome, sur la séparation des processus par onglet (ainsi on évite un plantage général Javascript si le contenu d'un onglet tombe), ce mécanisme sera automatiquement disponible dans le cœur du moteur, ce qui va permettre (on l'espère) de généraliser le principe.
Pour en savoir plus :
http://trac.webkit.org/wiki/WebKit2

Google, Adobe et Mozilla changent le plug-in

L'équipe de Chromium dit travailler étroitement avec Adobe et Mozilla pour changer radicalement l'approche du plug-in pour navigateur et apporter une nouvelle valeur ajoutée, (quasi inexistante jusqu'à présent, aux dires des développeurs). Cela passe par une toute nouvelle API : PlatformIndependentNPAPI. Il s'agit de disposer d'une API, la plus indépendante possible des navigateurs et des systèmes pour concevoir des plug-ins les plus portables possibles. La modularité de l'API et son ouverture doivent permettre de l'enrichir et de l'adapter, notamment sur la 2D et 3D. Il s'agit aussi d'améliorer la composition des pages et leur rendu. Dans ce cas, cette API doit posséder une sémantique claire quel que soit le navigateur et en séparant les processus de rendus et ceux du navigateur lui-même. Il s'agira de définir des événements et de savoir quelles sont les extensions disponibles dans le navigateur sans recourir à un téléchargement. Le but est d'optimiser la taille des plug-ins et d'éviter tout téléchargement inutile. Il semble aussi que ChromeOS puisse intégrer cette API.

La réalité augmentée sur le web ou « une dérive dans le Cyberpunk »

Le concept de réalité augmentée n'est pas nouveau et ses racines proviennent en grande partie des auteurs de science-fiction et notamment de Philip K. Dick qui a initié, avec son livre « les androïdes rêvent-ils de moutons électriques » (adapté au cinéma en 1982 sous le nom Blade Runner), la vague Cyberpunk. Les piliers du Cyberpunk sont en effet l'idée d'un futur noir et incertain engendré par une société tellement tournée vers l'électronique que la limite entre réalité et virtuel devient floue.

Le site Cyberpunkreview.com décrit même une des caractéristiques du genre comme traitant d'un accès à l'information qui est continu et en constante propagation. Rappelez-vous par exemple en 1984 lorsque le T800 est à la recherche de Sarah Connor dans Terminator : la caméra subjective, au travers des yeux du cyborg, permet de voir des données compilées en surimpression de l'image. D'autres exemples sont tout aussi mémorables comme par exemple RoboCop, iRobot ou encore Matrix. Aujourd'hui nombre des concepts imaginés par les auteurs de science-fiction sont devenus réalité grâce à la démultiplication de l'information électronique. Les sources de données étant toujours plus nombreuses, il est possible de faire des recoupements d'information très utiles, inimaginables il y a encore quelques années et qui font même parfois peur. Derniers champions en date sur le sujet : Microsoft et Google.

Quand les éditeurs s'en mêlent

Le 13 février, Blaise Aguera y Arcas, architecte de Bing Maps chez Microsoft a présenté les possibilités de StreetSide lors du TEDTalk. Microsoft a en effet lancé depuis décembre 2009 une version bêta de la solution de cartographie de Microsoft, concurrente de Google Maps. On aurait pu croire que Google avait déjà tout inventé sur le sujet mais la présentation a démontré que Microsoft avait su reprendre une longueur d'avance en combinant plusieurs technologies : Photosynth, Seadragon et bien sûr le moteur de recherche Bing.

Microsoft Live Labs Seadragon

Bing Maps s'appuie avant tout sur Seadragon, une technologie qui permet un zoom continu et fluide sur une image. L'image de plus fort détail se surimpose avec un calage parfait à l'image de moindre détail au fur et à mesure du zoom, permettant de zoomer presque à l'infini. C'est cette technique qu'utilise Bing Maps pour

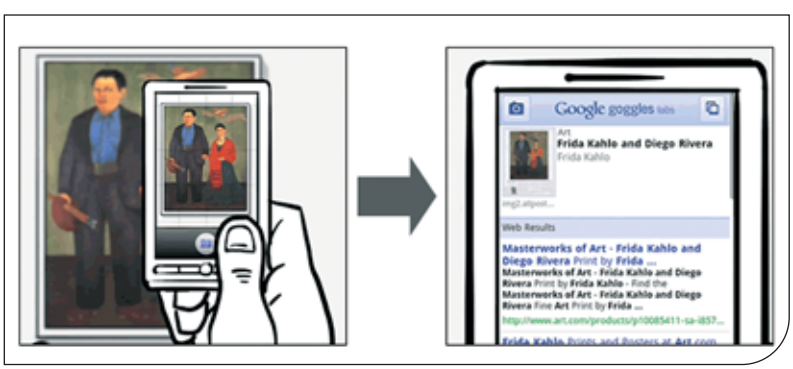
zoomer de plus en plus sur le planisphère. Passé un certain zoom, Bing Maps permet de voir les images satellites et même le relief 3D des bâtiments.

Mais Microsoft est allé un cran plus loin en tirant parti de PhotoSynth, un puissant logiciel de 'stitching' qui permet d'assembler des photos prises de façon complètement disparate d'un même sujet. Microsoft a ainsi utilisé des prises de vues de quelques dizaines de villes aux Etats-Unis pour constituer un équivalent de Google StreetView (c'est-à-dire une vue immersive dans un endroit) à ceci près que les images présentées tiennent compte de la 3D, offrant ainsi une vue panoramique non pas depuis un seul point de vue comme chez Google mais depuis n'importe quel point, permettant de 'voler' dans le paysage. En étant capable de faire l'agencement d'images de façon automatique, Photosynth permet aux sources d'être assemblées rendant dès lors possible l'assemblage non seulement des prises de vues faites par Microsoft lui-même mais aussi d'autres sources publiques comme Flickr, Picasa etc... D'un coup la base d'information et les possibilités s'enrichissent considérablement en permettant de surimposer à la vue actuelle du paysage des photos parfaitement calées pour redécouvrir le paysage à un autre moment de la journée ou de l'année ou même quelques dizaines d'années auparavant.

Et au-delà ?

Avec cela, nous sommes déjà bien avancés mais Blaise Aguera a présenté quelque chose de sans précédent : l'intégration d'une vidéo live directement sur la carte, le flux vidéo étant recalculé pour tenir compte de la 3D et de la rotation de la vidéo puis « collé » sur l'image statique en temps réel. Nous sommes bien cette fois aux confluents de la réalité et de la donnée informatique. Enfin, Blaise Aguera termine en montrant que StreetSide possède également un cousin dénommé skyview qui permet de s'envoler dans le ciel comme les enfants dans E.T. pour découvrir la cartographie des astres en réalité augmentée.

Mais d'autres applications sont également utilisables avec Bing Maps et permettent des recoupements d'informations intéressants comme par exemple avec Twitter Maps qui permet de voir sur la carte ou directement dans StreetSide les derniers tweets, permettant de voir ce que pensent les gens directement dans leur contexte. Il est même possible de regarder l'état du trafic pour voir ainsi si les gens 'tweetent' dans les bouchons et si oui, à quoi ils pensent. Au fur à mesure que de nouvelles sources d'informations créeront de nouvelles applications dans Bing, il sera possible de faire des recoupements toujours plus pointus et obtenir des données parfois inattendues, voire indiscrètes à n'en pas douter. À



quand d'ailleurs une application Facebook ? Mais Google n'est pas en reste sur ce thème avec son Google Goggles qui permet de faire une reconnaissance de caractères et de formes sur une image pour lancer une recherche sur le web et ainsi donner des informations sur l'objet que l'utilisateur a photographié : pour un livre, Google permet de comparer les prix, pour un tableau d'artiste de connaître le peintre et l'histoire du tableau, pour un paysage de connaître les éléments importants, pour une rue commerçante les bons restaurants...

Le service est très impressionnant mais Google le gardera-t-il pour lui ou en fera-t-il profiter les autres ? De fait, pourrons-nous recouper la puissance des deux outils (Bing maps et Google Goggles) afin de recouper toujours plus d'informations ? Il n'est pas sûr qu'il faille l'espérer car si on voit l'intérêt immédiat on peut aussi facilement en imaginer les dérives.

■ Fabrice **Dewasmes**
http://jtruc.dewasmes.net

Migration d'applications pour les systèmes multi-cœurs et 64 bit

Les processeurs à cœurs multiples et hybrides arrivent à la tête du classement établi par Gartner pour les technologies de rupture de 2008 à 2012. Ces technologies sont depuis entrées dans le mainstream. Ainsi, le défi de supporter les processeurs de nouvelles générations est devenu une nécessité. Ceci nous amène à nous interroger sur les possibilités de migration pour les applications existantes. La migration vers les processeurs modernes peut poser plusieurs problèmes aux développeurs. Nous avons donc choisi de mettre l'accent sur les erreurs communes en développement 64 bit et les techniques les moins intrusives pour paralléliser son application.

Processeurs 64 bit

Est-il suffisant de recompiler le code avec un compilateur 64 bits ? La recompilation est suffisante pour plusieurs types de programmes. On vérifiera que l'application utilise seulement des librairies compilées en 64 bit, car un processus 64 bit peut exécuter uniquement du code 64 bit. Dans des applications triviales, cette étape suffit à elle seule. Cependant, en réalité, rien ne garantit que les librairies utilisées aient une version 64 bit, la conversion est d'autant plus compliquée si leurs codes source sont indisponibles.

Dans ce cas, on pourra envisager des scénarios de communication interprocessus entre l'application et la bibliothèque en question. Outre cela, la compilation ne garantit pas le bon fonctionnement de l'application. De nouveaux bugs et anomalies peuvent avoir lieu, allant des erreurs d'affichage jusqu'aux failles de sécurité les plus sévères dues à un dépassement de tampon « buffer overflow » par exemple.

La mauvaise manipulation des pointeurs, la lecture des fichiers binaires ainsi que l'usage du code assembleur sont des sources potentielles de bugs. Prenons un exemple simple de programme qui a un comportement différent selon l'architecture matérielle.

```
int main(int argc, char* argv[])
{
  ///Le type size_t est 64bit pour le
  ///compilateur 64bit.
  size_t size = sizeof(myObject);
  ///L'affectation peut engendrer une
  ///perte de données.
  long a = size;

  ///%x assume que le type est 32 bit
  printf("%x\n",a);
  printf("%x\n",size);

  return 0;
}
```

La procédure main ne compte que 5 lignes, pourtant on peut identifier déjà deux bugs. Le type size_t a une valeur 64 bit pour un compilateur 64 bit. Le type long est un type 32 bit, la conversion : long a = size; Cela peut donc engendrer la perte de données, car la taille du type size_t est supérieure à celle du type long. De plus, %x qui permet de formater l'affichage en hexadécimal ne prend en compte que les valeurs 32 bit. On pourrait corriger la fonction d'affichage pour le programme précédent en remplaçant %x par %I64x pour l'architecture x64. Essayons à présent de corriger l'opération d'affectation, on pourrait songer à remplacer le long par le type *long long*. Le programme fonctionne correctement pour l'architecture x64, mais il ne retourne plus la bonne valeur pour l'architecture x86 ! Ceci était prévisible puisque *long long* est un entier 64 bit. À l'évidence, la solution la plus simple serait de remplacer le long par size_t. Ce petit programme de quelques lignes montre que la correction de ce type d'erreurs est loin d'être aisé.

```
#ifdef _WIN64
  #define xformat "I64"
#else
  #define xformat ""
#endif

int main(int argc, char* argv[])
{
  ///Le type size_t est 64bit pour le
  ///compilateur 64bit.
  size_t size = sizeof(myObject);
  ///L'affectation peut engendrer une
  ///perte de données.
  size_t a = size;

  ///%x assume que le type est 32 bit
  printf("%x" xformat "\n",a);
  printf("%x" xformat "\n",size);

  return 0;
}
```

Les deux erreurs dans le programme précédent sont causées par le changement de taille pour le type size_t. Ci-après une liste de

quelques types prédéfinis en C++ et leurs tailles respectives pour les architectures 32 et 64 bit.

	x86	x64
Int	32	32
Long	32	32
time_t	64	64
ptrdiff_t	32	64
Size_t	32	64

Le tableau ci-dessus montre qu'une classe contenant des attributs de type pointeur par exemple, n'occupe pas la même taille en mémoire pour un programme 64 bit et 32 bit.

Structures de données

Malgré la puissance brute des processeurs modernes, certains programmes compilés en 64 bit peuvent avoir une baisse de performance accompagnée généralement d'une augmentation importante en consommation de mémoire.

L'augmentation en consommation mémoire n'est pas seulement due au changement de taille pour les types de base. En effet, il y a de fortes chances qu'un mauvais alignement des structures soit la cause de cette augmentation. Considérons un cas ou ces effets secondaires se manifestent.

Soit la structure suivante :

```
struct heavyStruct
{
  char b;
  int *a;
  int  u;
};
```

En 32 bit, cette structure occupe une taille de 12 octets en mémoire. Par contre, en 64 bit elle aura une taille de 24 octets. Observons le cas d'une structure qui a exactement les mêmes attributs que « heavyStruct », mais qui ne sont pas dans le même ordre. On pourrait s'attendre que cette dernière ait naturellement une taille identique à « heavyStruct » mais il n'en est rien ! En effet en 64 bit elle aura une taille 16 octets et en 32 bit elle reste égale à 12 octets. Retenons donc ceci : la taille d'une structure n'est pas égale à la somme des tailles de ses attributs.

Dans la plupart des cas, ceci n'a pas un impact perceptible sur la consommation en mémoire. Toutefois, l'ordre des attributs à l'intérieur d'une structure peut limiter la consommation en mémoire de l'application, notamment lorsqu'il s'agit de créer un tableau de grande taille.

```
struct lightStruct
{
  int *a;
  int  u;
  char b;
};
```

La taille d'un struct n'est pas égale à la somme des tailles de ses attributs comme le montre la figure suivante, les cases en rouge sont celles occupées par les données de la structure tandis que celles en gris représentent l'espace gaspillé.

heavyStruct

	Octets							
	1	2	3	4	5	6	7	8
caractère	•	•	•	•	•	•	•	•
pointeur	•	•	•	•	•	•	•	•
entier	•	•	•	•	•	•	•	•

Un agencement meilleur des attributs permet d'optimiser l'usage de mémoire.

lightStruct

	Octets							
	1	2	3	4	5	6	7	8
pointeur	•	•	•	•	•	•	•	•
Entier+caractère	•	•	•	•	•			

Conclusion

Les erreurs dues au changement de taille des types prédéfinis sont très fréquentes. Et comme nous l'avons vu il y a de fortes probabilités que ces erreurs aient des conséquences dramatiques sur le comportement de l'application.

Systèmes multi-cœurs

Nous avons vu quelques pièges à éviter lors de la migration d'applications pour une architecture 64 bit. Malheureusement, cela ne suffit plus à tirer avantage des processeurs à cœurs multiples. En effet, si notre application est monothread, son processus sera exécuté sur un seul cœur.

Mais comment ferait-on pour paralléliser son application existante, sans être amené à la réécriture du code source ?

Bien entendu, on peut utiliser les threads Windows ou ceux de POSIX, mais pour la migration d'application on préfère utiliser la technologie la moins intrusive. La solution qui répond le plus à ce besoin est la technologie OpenMP. Car elle ne nécessite pas de modifier la structure de l'application.

Et c'est une technologie standard multi-plate-forme supportée par plusieurs compilateurs C / C++.

Supposons que nous ayons à notre disposition une technologie qui nous permette de paralléliser assez rapidement des portions de notre code existant. Si par exemple on parvient à paralléliser 60 % de notre code. Quel est le gain de performance maximal que l'on peut obtenir ?

La loi d'Amdahl, très utilisée en informatique, permet d'avoir une idée sur la performance maximale que l'on peut atteindre en parallélisant une portion P du code sur N cœurs :

$$\text{performance maximale} = \frac{1}{1 - P + \frac{P}{N}}$$

Si l'on parvient par exemple à paralléliser 60 % (P = 0,6) du code sur un système à quatre processeurs (N = 4) alors on peut s'attendre à une augmentation des performances allant jusqu'à deux fois la vitesse pour un seul processeur.

Pour un ordinateur ayant 16 processeurs en parallèle, le gain peut atteindre 2.35 fois la performance pour un seul processeur. En revanche si 100 % du code est parallèle, alors sur un système à 16 processeurs le gain est multiplié par 16. En règle générale plus il y a de proportion du code source tournant en parallèle plus le programme est extensible (scalable).

Granularité des threads

Lorsqu'il s'agit de paralléliser son application, la solution la plus simple serait de décomposer le programme en tâches indépendantes. La décomposition est souvent fonctionnelle. Par exemple, dans un jeu vidéo on peut exécuter le système sonore, l'intelligence artificielle et le rendu sur des threads séparés.
Cette solution est assez répandue, néanmoins, lorsque le nombre de cœurs dépasse le nombre de blocs parallèles, les performances ne seront plus améliorées avec l'ajout d'autres cœurs au système. Une granularité grossière des threads ne permet pas toujours d'augmenter l'extensibilité du programme.
Une granularité fine des threads permet une plus grande extensibilité. On doit créer les threads, assigner à chaque thread la tâche qu'il doit exécuter et finalement terminer leurs exécutions. L'implémentation manuelle de cette méthode peut apporter des gains considérables en termes de performances, mais elle est généralement assez difficile à réaliser.
Encore une fois, la technologie OpenMP peut simplifier les choses en proposant d'automatiser la gestion des threads.

Le couteau suisse du développeur

La migration peut être facilitée à l'aide des outils d'analyse et d'inspection automatique. Ils permettent de détecter plusieurs erreurs classiques en programmation parallèle, telles que les *deadlocks* et *race conditions*. En outre, à l'aide de la visualisation graphique, on peut détecter facilement les goulots d'étranglement dans une application. Avec Intel Parallel Studio, on trouve une gamme d'outils incontournables.

Exemple d'application

Afin de mieux présenter les spécificités des processeurs de nouvelles générations, nous avons choisi de réaliser un simple système de particule. Car il présente un cas typique ou une région du code ralentit l'ensemble du programme.
J'aimerais rappeler qu'un système de particule est une simulation d'un grand nombre de points dans l'espace soumis à un ensemble de forces comme la gravité. Il est souvent utilisé en infographie pour simuler l'apparence de nombreux phénomènes naturels et pour créer des effets spéciaux.
Nous n'avons pas la prétention de créer un système complexe de A jusqu'à Z, mais seulement de retenir les détails les plus importants de la migration.
L'idée de base était de trouver un cas d'application ou il y a un goulot d'étranglement au niveau des calculs effectués par le CPU. Les données relatives aux particules sont conservées dans la mémoire à haute performance du GPU.
Les données sont encapsulées dans des "buffer objects" pour minimiser le transfert de données entre processeurs et carte graphique.
À chaque frame, on calcule la nouvelle position, vitesse et accélération de chaque particule.
Ce temps de calcul est plus important que le temps pris par le rendu des particules par la carte graphique. Pour peu qu'on arrive à optimiser cette partie du code source, la vitesse d'exécution sera nettement améliorée. Mais avant d'entamer l'étape d'optimisation, examinons de plus près notre code pour y déceler d'éventuelles erreurs.La boucle permettant de mettre à jour les particules est sous la forme :

```
 for (ptrdiff_t i = 0;i<particleSystem->getParticleNumber();i++)
   particleSystem->Update(i);
```

Le type « ptrdiff _t» a une taille de 32 bit pour les systèmes 32 bit et 64 bit pour les systèmes 64 bit.
Ce qui le rend idéal pour les index des tableaux. Par ailleurs, il permet d'éviter de sacrifier inutilement de la mémoire pour les systèmes 32 bit. Et peut-être utilisé invariablement, quel que soit le système.
La boucle peut être réécrite de la façon suivante, dans l'intention d'expliciter les instructions de la fonction de mise à jour.

```
void particleSystem::Update()
{
 ptrdiff_t i;
 for (i = 0;i<particleNb;i++)
 {
   ForceZ += sinf(ForceZaxis[i]);
   //L'acceleration est la somme des forces divisée par
   //la masse de la particule.
   particle[i].acceleration = ForceZ / particle[i].mass * ez;
   //On met à jour la vitesse de la particule.
   particle[i].velocity += particle[i].acceleration;
   //On met à jour la position de la particule.
   particle[i].position += particle[i].velocity;
 }
}
```

La fonction de mise à jour pour une particule donnée ne nécessite pas la connaissance de l'état des autres particules. Elles agissent indépendamment et par conséquent on peut imaginer que cette tâche peut être répartie sur plusieurs threads.
Chaque thread mettra à jour un certain nombre de particules. Le problème ne nécessite aucune solution de synchronisation avancée. Sans doute, la solution doit être très simple, dans le cas d'OpenMP elle l'est, mais dans bien d'autres cas, elle nécessite la modification du code.
Pour indiquer au compilateur que la boucle peut être parallélisée, il suffit d'ajouter la directive :

```
pragma omp parallel for shared(particle)
```

Notons au passage que dans le cas où le support d'OpenMP est désactivé, le compilateur ignore tout simplement cette ligne et on obtiendra la version sérielle de notre application.

```
 #pragma omp parallel for shared(particle)
for (i = 0;i<particleNb;i++)
 {
   ForceZ += sinf(ForceZaxis[i]);
   //L'acceleration est la somme des forces divisée par
   //la masse de la particule.
   particle[i].acceleration = ForceZ / particle[i].mass * ez;
   //On met à jour la vitesse de la particule.
   particle[i].velocity += particle[i].acceleration;
   //On met à jour la position de la particule.
   particle[i].position += particle[i].velocity;
 }
```

La granularité des threads est un facteur important lors de la parallélisation de l'application. L'ajustement de la granularité en OpenMP se fait par le biais de la clause : schedule (type, nombre)

```
schedule(static, 100)
```

On indique que la capacité du lot est égale à 100. Par conséquent à chaque thread on affecte 100 itérations.

```
#pragma omp parallel for schedule(static, 100) private(i)
\ shared(particle, ForceZ)
for (i = 0;i<particleNb;i++)
 {
   ForceZ += sinf(ForceZaxis[i]);
   //L'acceleration est la somme des forces divisée par
   //la masse de la particule.
   particle[i].acceleration = ForceZ / particle[i].mass * ez;
   //On met à jour la vitesse de la particule.
   particle[i].velocity += particle[i].acceleration;
   //On met à jour la position de la particule.
   particle[i].position += particle[i].velocity;
 }
```

Considérons à présent l'instruction suivante permettant d'effectuer la somme des éléments d'un tableau :

```
 ForceZ += sinf(ForceZaxis[i]);
```

On peut indiquer à OpenMP d'optimiser l'opération de sommation. En spécifiant le nom de la variable et l'instruction. L'opération de réduction permet de se débarrasser des sémaphores de synchronisation. Dans le code suivant la variable ForceZ ne nécessite pas de primitive de synchronisation grâce à l'opération de réduction.

```
#pragma omp parallel for schedule(static, 100) private(i)
\ shared(particle) reduction(+:ForceZ)
for (i = 0;i<particleNb;i++)
 {
   ForceZ += sinf(ForceZaxis[i]);
   //L'acceleration est la somme des forces divisée par
   //la masse de la particule.
   particle[i].acceleration = ForceZ / particle[i].mass * ez;
   //On met à jour la vitesse de la particule.
   particle[i].velocity += particle[i].acceleration;
   //On met à jour la position de la particule.
   particle[i].position += particle[i].velocity;
 }
```

Fig.1

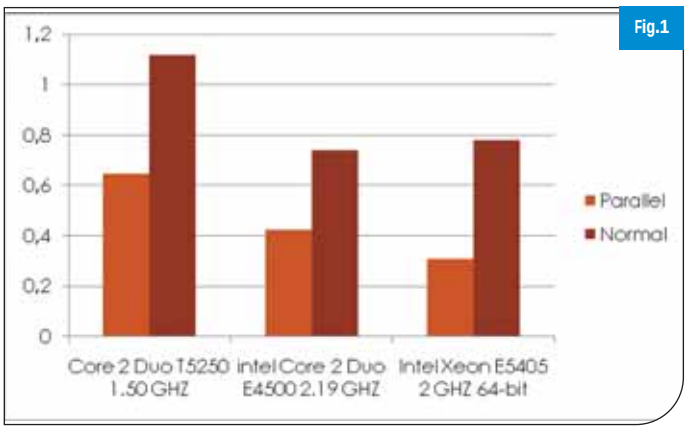
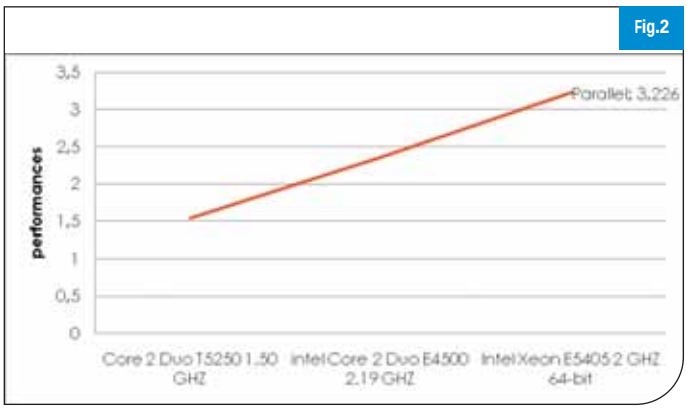


Fig.2



Résultats

Nous avons testé le temps d'exécution de l'application sur différents processeurs.

Nous remarquons une baisse de performance pour la version sérielle de l'application pour le processeur Intel Xeon, malgré qu'il soit le plus puissant des processeurs testés.

La version parallèle de l'application est nettement plus performante, en effet, elle est 2.26 fois plus rapide que la version sérielle pour le processeur Intel Xeon. [Fig.1]

La courbe suivante montre l'extensibilité de notre application. La version parallèle permet en effet d'exploiter la puissance des processeurs testés. [Fig.2]

Conclusion

Nous avons vu dans cet article quelques pièges à éviter lors de la migration d'applications pour les processeurs modernes. Une des technologies clefs pour une migration progressive est la technologie OpenMP. Ceci étant, il reste encore de nombreuses voies à explorer afin d'exploiter au mieux les capacités des ordinateurs modernes et notamment la puissance de calcul du GPU.

Le standard OpenCL comme le fut OpenMP ouvre de nouvelles possibilités aux développeurs pour paralléliser leurs applications.

Références et ressources

http://developer.amd.com/documentation/articles/pages/6220067.aspx

http://software.intel.com/en-us/blogs/2010/02/16/search-of-64-bit-errors-in-array-implementation/

http://software.intel.com/en-us/articles/granularity-and-parallel-performance/

http://msdn.microsoft.com/en-us/magazine/cc300794.aspx

http://msdn.microsoft.com/en-us/library/3b2e7499%28VS.80%29.aspx

http://www.viva64.com/content/articles/64-bit-development/?f=64-bit-migration-7-steps.html&lang=en&content=64-bit-development

Ghassen **Hamrouni**

Expert C++ chez IP-Tech : SSII offshore en Tunisie (www.iptech-offshore.com).

EclipseCon 2010 : la grand-messe d'Eclipse


D.R.

D.R.

Le point de ralliement des spécialistes et des utilisateurs d'Eclipse commence à être connu. C'est à Santa Clara que la conférence internationale Eclipse a pris ses quartiers cette année encore. Malgré les craintes des organisateurs quant à la fréquentation, suite à la crise, la conférence n'a pas été boudée et a gardé tout son intérêt. L'événement a eu lieu du 22 au 25 mars dernier.

Un sondage très intéressant concernant la provenance des participants a été réalisé durant une session. Celui-ci a révélé que dans la salle, la majorité des personnes présentes venait, non pas des États-Unis comme on pourrait s'y attendre, mais d'Europe. Toutefois, une analyse a posteriori des inscriptions réelles sur les 4 jours a démontré une majorité de participants américains, mais cela révèle bien la tendance qui veut que les Européens et plus particulièrement les Français, les Allemands et les Suisses, soient très impliqués dans le monde Eclipse.

Une conférence parfaitement orchestrée

L'organisation était particulièrement réussie, chaque jour se déroulant sur le même rythme : les matinées réservées aux tutoriaux, les après-midi aux différentes présentations et les soirées se concluant par des BoF (session de travail informelle) et des panels sur l'avancement et le devenir des projets.

En terme de contenu, les sessions sur les innovations 2010 étaient très enrichissantes et les sujets variés ; tout le monde pouvait y trouver son compte, spécialistes du runtime, d'applications métiers ou de la conception d'IDE. Les keynotes ont été comme chaque année à la hauteur des attentes avec une mention toute particulière pour celle de Jeff Norris « Rocket Science and the Republic » de la NASA. Jeff a réalisé durant sa présentation une démonstration d'une application basée sur Eclipse qui lui a permis de piloter un robot araignée de plusieurs mètres de haut dans un hangar de la NASA en direct depuis son portable. Une démonstration qui a bien évidement fait sensation auprès de l'assistance, pour preuve la standing ovation ponctuant la session ! La NASA était très présente cette année. Elle organisait également le e4 Rover Mars Challenge, un concours consistant à piloter des robots via e4 (le futur socle d'Eclipse 4). Ce concours se décomposait en deux jeux : développer un programme pour contrôler le robot à l'aide de la plate-forme e4 ou piloter le robot sur un parcours de façon à gagner un maximum de points. Heureux, le vainqueur du meilleur client, Jon Dearden, remporte une visite du laboratoire de la NASA à Los Angeles. À noter le client exceptionnel développé par Peter Friese et Heiko Behrens permettant de piloter le robot depuis un iPhone.

Un bilan positif

La fondation Eclipse est arrivée à la conférence avec de bonnes nouvelles et un bilan positif. La communauté approche les 1 000 commiteurs actifs, presque 100 organismes membres et 250 projets actifs hébergés. Cette activité s'est ressentie dans les différents Eclipse Days et Eclipse Democamp organisés durant l'année qui ont été de francs succès ! Deux ont été organisés en France : à Paris et Nantes.

La conférence était l'occasion parfaite pour lancer le plugin intégré à Eclipse du Marketplace, un espace de téléchargement et commercialisation de produits basés sur la plateforme Eclipse, une sorte d' « AppStore made in Eclipse » mais avec l'ouverture habituelle de la fondation.

Analogue au Google Code, Eclipse compte également ouvrir une forge pour héberger des projets open source basés sur la plate-forme. Répon-

dant au nom de Eclipse Labs, elle devrait apporter un incubateur idéal pour le futur d'Eclipse.

Le futur n'attend pas !

Le sujet incontournable attendu par beaucoup était e4, le projet de refonte du cœur d'Eclipse, incluant notamment un support du Web (déploiement des plugins dans un navigateur et utilisation des standards Internet). Beaucoup de discussions et de présentations ont eu lieu à ce sujet et l'accent a été mis sur le travail effectué pour la compatibilité Eclipse 3.X / e4, la simplification de la programmation des plugins via les mécanismes d'injection et d'annotations, et le support des widgets OpenSocial au sein même d'Eclipse. Ceci associé à l'effort important de développement laisse présager une bascule de la communauté vers e4, assez rapide dans les années à venir.

La fondation a également annoncé son désir de passer à Git pour la gestion des sources d'ici 2012. Les acteurs des projets jGit (implémentation d'un moteur Git en Java) et eGit (client Git pour Eclipse) s'impliquent activement dans la réalisation d'outils qui assureront à la communauté une migration de CVS (ou SVN) vers Git en toute sérénité.

L'Eclipse Modeling Project de plus en plus présent

Comme depuis quelques années, le projet Modeling était très bien représenté avec de nombreuses sessions. Ceci reflète bien l'expansion de la technologie au sein de la communauté. De plus en plus de projets adoptent EMF et commencent à intégrer les composants gravitant autour pour gagner en fonctionnalités. Le « Modeling panel » du lundi a été un véritable succès, en partie grâce à cela.

Le panel a été l'occasion pour de nombreux industriels de faire des retours d'expérience sur leurs cas d'utilisation. Ceci a mis en évidence que des composants comme EMF Compare ou CDO étaient très utilisés dans le cadre industriel. Ces retours d'expérience ont également amené une discussion importante sur la communication dans le monde open source. Il est difficile actuellement pour les développeurs Eclipse d'estimer à quelle hauteur leur produit est utilisé. Or cette information est précieuse et des moyens pourraient être mis en œuvre pour en permettre la collecte. L'essor de la modélisation transparaît également avec le gain en visibilité d'outils comme Acceleo ou Xtext. Dans le cas de la génération de code, les fonctionnalités d'Acceleo 3 se sont multipliées ces derniers mois et l'éditeur est désormais très évolué pour aider à la fabrication d'un template de générateur à partir d'un source existant. Les démonstrations d'Acceleo (sur la fabrication d'applications Android particulièrement) montrent très rapidement l'intérêt de la génération de code et le gain de productivité qu'un outil de ce genre peut apporter. Côté Xtext, le nombre important de projets ayant intégré une syntaxe textuelle à leur outil de modélisation montre à quel point la technologie se répand.

Concernant la modélisation graphique, GMF qui était jusqu'à présent le seul projet proposant de créer des modeleurs graphiques (orientés MDA) a été rejoint par Graphiti. Ce nouveau projet contribué par SAP était de fait très attendu cette année, car il est censé offrir de nouvelles fonctionnalités ergonomiques à GMF. Hélas, les présentations auxquelles nous avons pu assister nous laissent sur notre faim ... Les nombreux slides techniques auraient dû laisser la place à des démonstrations visuelles de l'outil. Heureusement, une autre session, nommée « GMF showcase » mettait en évidence la richesse fonctionnelle du framework à travers des démos pour la modélisation d'applications JavaEE ou SOA, de référentiels Togaf, de contraintes temps réel, la cartographie d'existant et l'introduction d'une nouvelle méthode de conception par « points de vue ».

Eclipse Con a également permis d'introduire l'Extended Editing Framework (EEF), un framework de présentation pour les modèles EMF qui facilite la création de formulaires Eclipse.

En parallèle, le projet PMF (Presentation Modeling Framework) apportait ses premières démonstrations : de la génération SWT pour le projet XWT et de l'interprétation de modèles graphiques pour le projet Wazaabi. En matière de transformation de modèles, cette année a révélé que seul le moteur ATL maintenait un effort constant et se révèle actuellement l'outil industriel le plus avancé.

Finalement, la présentation « Modeling Project Runway 2010 » a introduit dix projets (cinq minutes par projet) témoignant de toute l'étendue fonctionnelle que la modélisation pouvait couvrir. Elle a également amené l'introduction de nouveaux projets comme Amalgamation par Cédric Brun qui vise à épurer la distribution Eclipse Modeling et à simplifier la procédure pour installer de nouveaux composants. Benoit Langlois de Thalès a pour sa part annoncé la création du projet EGF destiné à produire des « Software Factories » : des configurations pour enchaîner des générateurs, des extensions et des paramétrages de l'environnement de conception. Thalès venait également annoncer, par le biais de Daniel Exertier et Benoit Langlois, son intention de s'impliquer de plus en plus au sein de la communauté Eclipse. Ainsi, la libération du projet EGF ou encore le désir d'apporter des contributeurs à la communauté (comme par exemple au projet EEF) sont des signes forts de cette volonté.

Un des stands sur EclipseCon.

Rendez-vous en Californie en 2011...

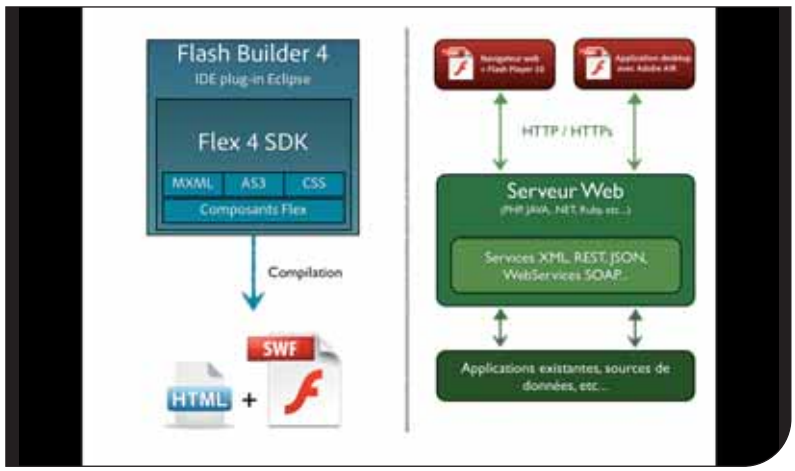
Cette conférence aura permis de montrer qu'Eclipse ne cesse d'évoluer, aussi bien en tant que fondation (Marketplace, Eclipse Labs,...) qu'en tant que plate-forme (e4). La communauté demeure très vivante, les projets comme l'Eclipse Modeling Project sont particulièrement actifs et productifs. Nous ne pourrons que regretter les fontaines de chocolat, grandes absentes de la traditionnelle réception des posters... En espérant qu'elles reviennent l'année prochaine !

■ Goulwen **Le Fur**
Leader du projet EEF
goulwen.lefur@obeo.fr

Flex 4 : cap sur le multi-plateforme et les nouveautés

Disponible depuis mi-mars, la 4e version majeure de Flex, framework open source de développement dédié aux interfaces internet riches (RIA) est très prometteuse : nouvelles fonctions, nouveaux outils, nouvelles approches. Pas de véritable rupture avec la v3 mais de la consolidation et des fonctions bienvenues. Petit tour du propriétaire.

Le développeur Flex connaissant la v3 ne sera pas pénalisé car il retrouvera très rapidement les mêmes réflexes. Et les équipes d'Adobe ont tout fait pour préserver la compatibilité du code vers la v4. Surtout, l'éditeur met l'accent dessus, cette version est issue d'une étroite collaboration entre Adobe et la communauté. Les nombreux retours de développeurs, les suggestions, ont abouti à l'introduction de nombreuses améliorations et nouveautés. Ainsi, on dispose du nouveau modèle de composants : Spark. Relookage de nom aussi, car Flex Builder devient Flash Builder, mais reste basé sur Eclipse.



Les nouveautés du Flex SDK

Premier changement notable, un développement Flex 4 compilé produit un fichier Flash qui ne pourra être exécuté que par un Flash Player 10 au minimum. Avec Flex 3, la majorité des développements étaient à destination de la version Flash Player 9. Bonne nouvelle, Flash Player 10 est déjà très largement diffusé et présent sur plus de 95% des postes (PC, Mac et Linux). Les nouveautés du SDK suivent trois orientations souhaitées par la communauté de développeurs de RIA : une meilleure gestion du design des interfaces, une augmentation de la productivité des développeurs, une évolution du framework pour tenir compte des nouveautés du Flash Player.

La principale innovation en termes de design d'applications riches réside dans la nouvelle architecture des composants Flex, appelée Spark. Construite à partir de l'ancien modèle MX, Spark fournit un mécanisme plus simple et plus expressif pour les développeurs et les designers, qui pourront désormais travailler ensemble sur le design des applications Flex. L'objectif de Spark est de séparer au sein même d'un composant l'apparence dans une classe de skin, du comportement (dans une classe principale). Une autre nouveauté est le support du format FXG, le nouveau format d'échange de graphisme d'Adobe. Le Framework Flex possède désormais des primitives graphiques alignées avec le standard FXG. Vous pouvez ainsi décrire en code des formes complexes, des dégradés, etc. Le compilateur de Flex 4 optimise les graphiques FXG en les transformant en tags SWF interprétés nativement par le Flash Player, ce qui améliore les performances de rendu des applications Flex. Enfin, la nouvelle architecture Spark introduit un modèle de mise en page (layout) plus flexible. En quelques lignes de code, vous pouvez créer un nouveau mode d'agencement de composants, ou d'éléments d'un composant (une liste) en utilisant les capacités 2D ou 3D du Flash Player. Les layout peuvent être modifiés au runtime, et une même liste de photos peut donc s'afficher sous forme d'enchaînement vertical ou de carrousel 3D en modifiant la classe de layout associée au composant liste.

MXML devient aussi déclaratif

Flex 4 revoit aussi la façon de déclarer les états (states) d'une application ou au sein même d'un composant. Le langage de description des interfaces MXML a donc été revu et étendu pour offrir une syntaxe plus lisible. Le langage MXML devient donc MXML 2009, avec un nouveau namespace à déclarer. Par exemple, la déclaration de ce bouton indique que dans l'état 'NewButton', la propriété enabled passera à false :

```
<s:Button id="b1" label="Ok" enabled.
NewButton="false"/>
```

Le développeur Flex 4 sera aussi plus productif grâce aux nouveautés du SDK. Premièrement, les performances du compilateur sont grandement améliorées, ainsi que le support de l'ASDoc dans les documents MXML. Le data-binding, qui consiste à associer une source de données à un composant visuel, est une des

grandes forces de Flex. Ce modèle est étendu avec la possibilité de déclarer du data-binding dans les deux sens en rajoutant un @ devant votre expression. Du coup un changement dans le source de données impactera le composant visuel, et vice-versa. D'autre part, on dispose avec MXML d'une structure contenant des zones déclaratives et plus uniquement d'une structure de description d'interface. On peut ainsi avoir des éléments non visuels dans un code MXML ou encore déclarer les états de l'application.

Le Framework a aussi été amélioré pour tenir compte des dernières nouveautés du Flash Player. Par exemple, le lecteur de vidéo est totalement personnalisable et tient compte des recommandations open-source de l'Open Source Media Framework (OSMF). Le nouveau moteur de texte, Text Layout Framework, introduit dans le Flash Player 10 est aussi pris en charge par Flex 4. Il supportera très prochainement le « layout mirroring », soit l'affichage et l'édition de texte de droite vers la gauche.

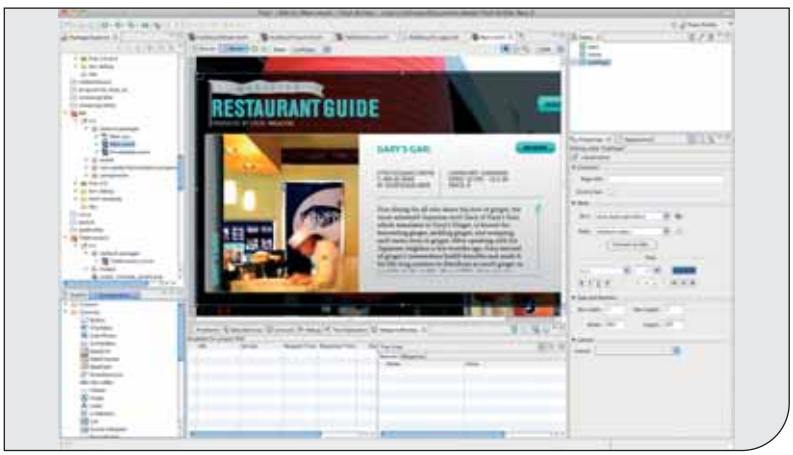
Flash Builder 4 : toujours Eclipse !

Première nouveauté : le changement de nom. En effet les développeurs Flash, qui utilisent uniquement l'API ActionScript 3 du Flash Player, pourront aussi désormais utiliser l'IDE Flash Builder 4 pour coder leurs projets. Même s'ils n'utilisent pas le Flex SDK, ils profiteront de toutes les nouveautés du Builder et d'une expérience de coding professionnelle.

L'expérience de collaboration avec des designers a été améliorée. Un développeur Flex peut facilement accueillir et interagir avec des composants créés par un développeur Flash (avec Flash CS4 par exemple). Ce dernier dispose désormais d'une commande qui permet de sélectionner un MovieClip et de le transformer en composant Flex. Flash Catalyst verra bientôt le jour, et permettra à des designers d'interfaces, habitués à PhotoShop ou Illustrator, de produire des prototypes interactifs, et de facilement skinner sans une ligne de code des composants Flex. Ces nouveaux outils vont faciliter le skinning d'applications Flex. Avancée majeure, le nouvel outil de connexion à une source de données depuis l'environnement de développement : connexion à un service http, REST, introspection avancée d'un WebService, communication avec des services PHP, Java, .NET, etc. Flash Builder 4 introspecte des services distants et génère automatiquement le code Flex pour communiquer avec ces services pour effectuer des lectures, des mises à jour de données, des suppressions d'enregistrements, etc. L'IDE est même capable de générer pour vous des classes PHP côté serveur pour communiquer avec des tables MySQL par exemple ! En quelques minutes, vous avez un backoffice qui fonctionne parfaitement !

L'IDE est aussi très mûr en matière d'expérience de coding : refactoring de code, debugging amélioré (debug conditionnel, saut vers une ligne, évaluation d'expression en temps réel…), un nouveau profiler pour mesurer l'activité en mémoire au runtime, déplacement de classes et de packages avec refactoring… Des assistants permettent aussi la génération de getter / setter pour vos objets, de la génération automatique de gestionnaires d'événements ou le support de l'ASDoc en aide visuelle. Il est aussi désormais possible de créer des modèles (templates) de génération automatique de fichiers MXML, ActionScript ou de CSS à la demande. L'intégration continue avec Flex est aussi officialisée avec Flash Builder 4. Les tests unitaires (projet open source Flex Unit) sont intégrés dans l'environnement, et un nouvel outil de network monitor permet de scruter les objets qui transitent sur le réseau. D'autres projets comme FlexPMD permettent de tester la qualité du code directement dans l'IDE, de générer des rapports pour Hudson ou Sonar par exemple.

Avec cette nouvelle version, Flash Builder 4 rassure les développeurs d'applications d'Entreprise qui retrouvent le même niveau de professionnalisme qu'avec des langages comme Java ou .Net.



Flash Builder 4 peut se résumer par 3 axes

1 le workflow développeur – design qui va enfin se concrétiser réellement, la disponibilité d'un CSS avancé

2 un meilleur cycle de vie des applications avec le protocole d'échange AMF, l'intégration continue et le support de Ant, Maven, Flex Unit, Flex PMD (pour la qualité du code). On bénéficie aussi d'un support de Sonar, éditeur spécialisé dans la qualité du code, la couverture du code.

3 Mobilité : Flex 4 cible Air et le Flash Player sur mobile. On ne dispose pas encore d'un framework Flex sur mobile mais une version optimisée sera disponible. Autre point non pris en charge, le parallélisme du code qui nécessite une refonte en profondeur du Flash Player car le support du multicore passera par le lecteur. La version 10.1 permet aujourd'hui de décharger une partie de la charge CPU vers le GPU mais demain, l'objectif sera de proposer un player massivement multithread avant de le proposer directement dans le code, dans le framework.

■ Michaël **Chaize** et François **Tonic**

Le pôle développement techno de Finance active recrute :

des opportunités pour les développeurs JAVA/JEE expérimentés

Interview de Nicolas FRANK, responsable du pôle.

Finance active est un éditeur français qui fournit des services d'aide à la décision aux directions financières depuis 2000. Il associe conseil et plateformes web pour offrir un service expert complet. L'entreprise, située place de la Bourse, est leader français dans ce domaine et est présente de manière croissante en Europe.

Les équipes du pôle Technologies & Innovation réalisent, pour les 4 000 utilisateurs européens, le développement des plateformes internet.

Programmez ! : Quel est votre parcours ?

Nicolas Frank : A 35 ans, j'ai travaillé plusieurs années en SSII et effectué des missions en milieu financier et bancaire à développer de l'applicatif JEE. J'ai rejoint Finance active en 2003 pour relever le défi de la construction d'un leader technologique des plateformes SaaS.

P !: Pouvez-vous détailler l'offre logicielle ?

N.F. : Finance active s'est d'abord développée autour d'Insito, notre service d'aide à la gestion de la dette et du risque de taux d'intérêt pour les entreprises et les collectivités locales. Cette solution, utilisée par plus de 2 000 clients, est aujourd'hui éprouvée d'un point de vue technologique et nous a permis d'affirmer un leadership fort sur nos marchés. Depuis 5 ans, nous avons élargi notre offre avec la conception d'un nouveau service chaque année : Fx'ent pour la gestion du risque de change, Profolio pour les placements, Inviseo pour la prospective financière... Notre offre de services est devenue globale et cela nous permet d'alimenter une croissance supérieure à 10% chaque année.

P !: Quelles technologies utilisez-vous pour vos services ?

N.F. : Chaque plateforme est l'occasion de mettre en œuvre les dernières technologies arrivant à maturité. Les défis technologiques sont donc notre quotidien. Le passage d'Insito à J2EE en 2003 a été mené autour de JBoss et des EJB2. Le service Fx'ent, lancé il y a 6 mois, utilise JPA / hibernate avec des interfaces « web 2.0 » et met en avant l'interactivité et l'ergonomie. Nous allons prochainement lancer une brique technique de gestion des données de marché basée sur la suite Spring (MVC, Intégration, Batch).

P !: Vous êtes fortement impliqués dans les méthodologies agiles ?

N.F. : Tout au long de ces années nous avons également mûri méthodologiquement. Notre philosophie s'inscrit effectivement dans la pratique des méthodes Agiles : la recherche de l'amélioration continue, de la communication, de l'échange et de la qualité. Tous nos projets sont gérés en SCRUM, avec un project manager fonctionnel qui accompagne au quotidien chaque équipe. Nous portons également beaucoup d'attention à l'innovation. Nous organisons semestriellement un challenge technique où chacun peut expérimenter un sujet technique de son choix. Les meilleurs projets sont ensuite portés par nos activités de R&D tout au long de l'année.

P !: Quels sont les prochains défis ?

N.F. : Nous avons de nombreux projets dans les cartons, nouveaux produits ou évolutions techniques majeures. Nous allons également travailler sur la connectivité entre nos services et ceux de partenaires. Nous continuons à développer de nouvelles briques techniques pour toujours mieux répondre aux problématiques de performance, de calculs mathématiques complexes gourmands en ressources, ...

> ***« Innovation, excellence, challenges sont des valeurs que vous partagez... vous avez du talent... Rejoignez nous et soyez acteur d'une réussite commune ! »***

P !: Et les recrutements ?

N.F. : Pour relever ces défis, nous comptons embaucher de nombreux collaborateurs dans les prochaines années, en privilégiant un recrutement de qualité et de bonnes conditions d'intégration. Nous cherchons des personnes intéressées par la technique mais aussi par les métiers de la finance, de bons communicants capables de travailler en équipe dans un esprit ouvert et critique, et désireuses de créer des logiciels de qualité qui répondent réellement aux besoins des utilisateurs. ■

Des outils considérés comme des références dans le domaine de la gestion financière.

Finance active recrute

- **Ingénieurs en développement Java / JEE confirmés**
- **Architecte Logiciel**
- **Web Designer**
- **Ingénieur support applicatif junior**

Contact : Isabelle **BRASQUET**, *Chargée de recrutement*
rh@financeactive.com

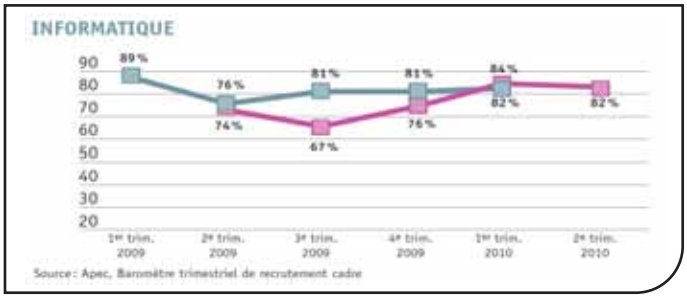
L'emploi est reparti en mars 2010

2e trimestre : 82 % des entreprises de l'informatique prévoient de recruter des cadres

Selon l'APEC, la proportion d'entreprises de plus de cent salariés qui recrutent des cadres est en progression. 50 % des entreprises interrogées ont recruté au moins un cadre au premier trimestre 2010, soit 6 points de plus que l'an passé. De même, 43 % des entreprises prévoient de recruter au deuxième trimestre 2010, soit 7 points de plus qu'au deuxième trimestre 2009. Ces résultats encourageants sont toutefois à tempérer, dans la mesure où les chiffres de 2009 constituent un plancher pour le Baromètre Apec. Ainsi, les résultats de ce trimestre se situent toujours très en deçà en comparaison des mêmes trimestres de 2006 à 2008.

Informatique : le recrutement redémarre timidement

82 % des entreprises du secteur Informatique prévoient de recruter des cadres au deuxième trimestre 2010 contre 74 % au deuxième trimestre 2009. De plus, la moitié des entreprises qui recrutent déclarent que leur volume de recrutement est actuellement plus élevé qu'il y a un an, ce qui pourrait témoigner d'une reprise dynamique dans ce secteur.



Internet + 366 %

Mars 2010

	Nb offres	*Ecart 2010/09*
Direction informatique	151	
Exploitation, maintenance informatique	446	
Informatique de gestion	3 616	
Informatique industrielle	1 037	
Informatique web, sites et portails internet	1 012	
Maîtrise d'ouvrage et fonctionnel	665	
Système, réseaux, données	1 454	
TOTAL INFORMATIQUE	**8 381**	**+ 43%**

en cumul sur 12 mois glissants

	Nb offres	*Ecart 2010/09*
Direction informatique	**1 252**	**+ 178%**
Exploitation, maintenance informatique	3 987	- 24%
Informatique de gestion	27 711	- 26%
Informatique industrielle	9 426	- 27%
Informatique web, sites et portails internet	**4 312**	**+ 366%**
Maîtrise d'ouvrage et fonctionnel	6 996	- 37%
Système, réseaux, données	12 105	- 25%
TOTAL INFORMATIQUE	**65 789**	**- 22%**

Agrégation de données en SQL avec les fonctions d'analyse

Dans de nombreuses situations, l'informaticien de gestion doit effectuer des regroupements sur les données qu'il extrait du SGBD afin de leur appliquer des fonctions d'agrégation (COUNT, SUM, AVG, MAX, ...) pour produire, par exemple, des statistiques.

Bien que cette problématique puisse être traitée par des requêtes SQL classiques en recourant au « GROUP BY » et aux fonctions de groupe, certains SGBD proposent aujourd'hui une nouvelle classe de fonctions appelées « fonctions d'analyse » (« Analytic Functions » en anglais) pour y répondre plus facilement. Ces fonctions sont normalisées (ISO / IEC 9075-1:2003 et ISO / IEC 9075-1:2008) mais encore très rarement implémentées par les SGBD du marché et encore moins connues par les développeurs. Dans le cadre de cet article, nous nous baserons sur le SGBD Oracle qui supporte ces fonctions depuis sa version 8i.

Contexte des exemples

Les exemples que nous allons étudier sont inspirés d'une librairie PL / SQL que nous avons développée pour permettre d'invoquer de façon asynchrone des services REST depuis une transaction dans la base de données Oracle [Fig.1]. Comme on peut le voir dans le diagramme de séquence, une transaction déclenche de façon asynchrone la création d'un ordre. Toutes les heures, le scheduler de la base de données est chargé d'exécuter les ordres qui sont en attente. Un envoi est alors créé et le service REST est appelé. Si l'invocation échoue, autrement dit, si le code de statut de la réponse HTTP est différent de « 200 », nous déterminons alors si l'ordre doit être ré-exécuté ultérieurement. Dans l'affirmative, par exemple avec un code de statut « 404 Not Found » ou « 503 Service Unavailable », l'ordre sera exécuté à nouveau par le scheduler et un nouvel envoi sera créé. Voici un diagramme simplifié du schéma de la base de données que nous avons utilisée pour développer cette librairie. Nous nous baserons sur ce modèle pour illustrer cet article [Fig.2].
D'un point de vue pédagogique, afin de ne pas complexifier d'avantage les requêtes SQL des exemples, nous avons choisi de créer deux vues [Fig.3]. Nous attirons votre attention sur les colonnes commençant par le caractère « / » qui indique des données « calculées ».

Cas d'emploi des fonctions d'analyse

Nous allons démontrer quelques cas d'emploi des fonctions d'analyse au travers de plusieurs exemples concrets.

Pour commencer, nous voulons créer une requête qui retournera, pour chaque envoi, sa durée, mais également celle de l'envoi qui l'a précédée [Fig.4]. Voici un exemple de requête qui permet d'aboutir à ce résultat :

```
-- requête E01
SELECT ordenv1.ord_numero, ordenv1.env_numerodep, ordenv1.env_duree,
       ordenv2.env_duree env_duree_precedent
  FROM v_ordre_envois ordenv1
    LEFT OUTER JOIN v_ordre_envois ordenv2
      ON ordenv2.ord_numero = ordenv1.ord_numero AND
         ordenv2.env_numerodep = ordenv1.env_numerodep - 1
  WHERE ordenv1.ord_numero = 2
  ORDER BY ordenv1.ord_numero ASC, ordenv1.env_numerodep ASC;
```

Comme vous pouvez le constater, la compréhension du code SQL n'est pas vraiment évidente. Pour obtenir le résultat escompté, nous avons effectué une jointure réflexive sur la vue « V_ORDRE_ENVOIS » et indiqué dans la condition qu'il faut décaler la jointure de 1 sur la base de la colonne « NUMERODEP ». Il faut obligatoirement faire une jointure externe afin de prendre en compte le premier envoi qui n'a pas de précédent [Fig.5]. Pour obtenir la durée de l'envoi suivant, il suffit de changer la condition de la jointure.

```
-- requête E02
SELECT ordenv1.ord_numero, ordenv1.env_numerodep, ordenv1.env_duree,
       ordenv2.env_duree env_duree_suivant
  FROM v_ordre_envois ordenv1
    LEFT OUTER JOIN v_ordre_envois ordenv2
      ON ordenv2.ord_numero = ordenv1.ord_numero AND
         ordenv2.env_numerodep = ordenv1.env_numerodep + 1
  WHERE ordenv1.ord_numero = 2
  ORDER BY ordenv1.ord_numero ASC, ordenv1.env_numerodep ASC;
```

Il faut également faire une jointure externe afin de prendre en compte le dernier envoi qui n'a pas de suivant [Fig.6].

Fig.1

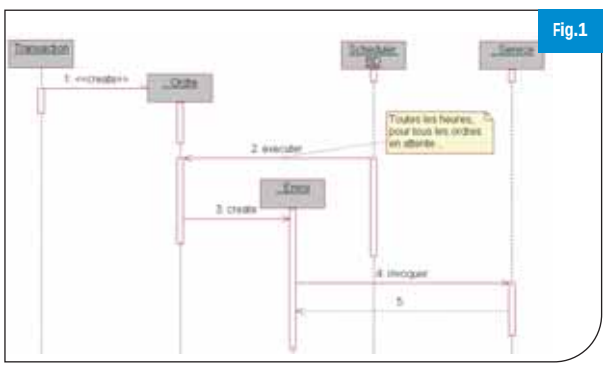


Fig.2

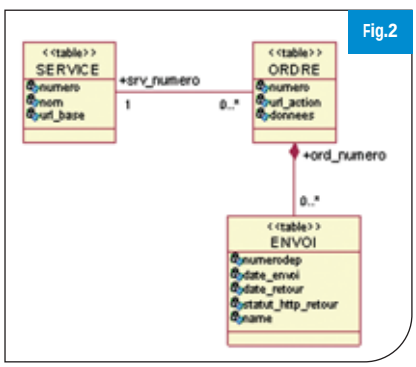
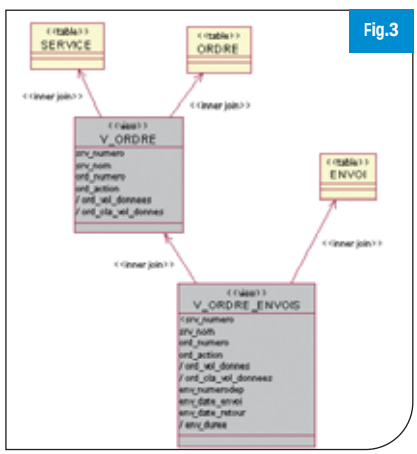


Fig.3



Mais ces requêtes ne fonctionnent que grâce au fait que chaque envoi possède un numéro d'identification (« NUMERODEP ») qui indique la séquence dans laquelle il a été envoyé. Il est cependant possible de contourner cette problématique au moyen de la pseudo-colonne « ROWNUM » qui permet d'attribuer un numéro de séquence unique à chaque enregistrement retourné par une requête. Voici un exemple de requête pour obtenir la durée de l'envoi précédent en utilisant cette fois-ci « ROWNUM ».

```
-- requête E03
SELECT ordenv1.ord_numero, ordenv1.env_numerodep, ordenv1.env_duree,
       ordenv3.env_duree env_duree_precedent
  FROM v_ordre_envois ordenv1
    INNER JOIN (
      SELECT ROWNUM i, ord_numero, env_numerodep
        FROM (
          SELECT ord_numero, env_numerodep
            FROM v_ordre_envois
            ORDER BY ord_numero ASC, env_date_envoi ASC
        )
    ) ordenv2
      ON ordenv1.ord_numero = ordenv2.ord_numero AND
         ordenv1.env_numerodep = ordenv2.env_numerodep
    LEFT OUTER JOIN (
      SELECT ROWNUM i, ord_numero, env_numerodep, env_duree
        FROM (
          SELECT ord_numero, env_numerodep, env_duree
            FROM v_ordre_envois
            ORDER BY ord_numero ASC, env_date_envoi ASC
        )
    ) ordenv3
      ON ordenv3.ord_numero = ordenv1.ord_numero AND
         ordenv3.i = ordenv2.i - 1
  WHERE ordenv1.ord_numero = 2
  ORDER BY ordenv1.ord_numero ASC, ordenv1.env_numerodep ASC;
```

La compréhension de cette requête n'est, à nouveau, pas chose aisée. Nous avons dû créer des vues temporaires (« ORDENV2 » et « ORDENV3 ») retournant le numéro de séquence (« ROWNUM ») des enregistrements pour un tri sur le numéro de l'ordre et la date d'envoi. Ces vues sont ensuite reliées au moyen d'une jointure externe comme dans l'exemple précédent mais cette fois sur la base du numéro de séquence généré. Vous pouvez facilement imaginer la complexité de la requête qu'il faudrait pour obtenir à la fois, pour chaque envoi, la durée de l'envoi qui l'a précédée et celle de celui qui le suit [Fig.7]. Nous allons maintenant voir comment résoudre facilement ce problème avec les fonctions d'analyse du SGBD Oracle.

```
-- requête E04
SELECT ord_numero, env_numerodep, env_duree,
       LAG(env_duree, 1) OVER
         (PARTITION BY ord_numero
           ORDER BY env_date_envoi ASC) env_duree_precedant,
       LEAD(env_duree, 1) OVER
         (PARTITION BY ord_numero
           ORDER BY env_date_envoi ASC) env_duree_suivant
  FROM v_ordre_envois
  WHERE ord_numero = 2
  ORDER BY ord_numero ASC, env_numerodep ASC;
```

Les fonctions « LAG » et « LEAD » sont des fonctions d'analyse qui permettent d'accéder aux enregistrements précédant ou suivant l'enregistrement courant, sans devoir faire de jointure réflexive et sans recourir à « ROWNUM » . Il faut indiquer en paramètre à la fonction, la colonne dont nous souhaitons récupérer la valeur ainsi que le décalage souhaité par rapport à l'enregistrement courant. Vous conviendrez que, malgré une syntaxe un peu bizarre de prime abord, la requête est cette fois bien plus compréhensible. Nous reviendrons en détail un peu plus tard sur la syntaxe des fonctions d'analyse. Prenons un autre exemple, nous aimerions cette fois obtenir la somme des durées de l'envoi courant et de l'envoi précédent [Fig.8]. Voici une requête qui permet d'obtenir ce résultat en nous basant sur la colonne « NUMERODEP » pour définir l'ordre des envois :

```
-- requête E05
SELECT ord_numero, env_numerodep, env_duree,
       ( SELECT SUM(env_duree)
           FROM v_ordre_envois
           WHERE ord_numero = ordenv.ord_numero AND
                 env_numerodep BETWEEN ordenv.env_numerodep - 1 AND
                                       ordenv.env_numerodep
       ) env_duree_2_passe
  FROM v_ordre_envois ordenv
  WHERE ord_numero = 2;
```

Nous avons besoin d'utiliser cette fois-ci une sous-requête corrélée qui récupère l'envoi courant et son prédécesseur et qui effectue la somme de leur durée.
Voici la même requête mais en se basant cette fois sur la date d'envoi, ce qui nécessite, comme nous l'avons vu précédemment, l'utilisation de la pseudo-colonne « ROWNUM » :

```
-- requête E06
SELECT i, ord_numero, env_numerodep, env_duree,
       ( SELECT SUM(env_duree)
           FROM (
           SELECT ROWNUM i, ord_numero, env_numerodep, env_duree
                FROM (
                 SELECT ord_numero, env_numerodep, env_duree
                   FROM v_ordre_envois
                  ORDER BY ord_numero ASC, env_date_envoi ASC
                )
           )
           WHERE ord_numero = ordenv1.ord_numero AND
                 i BETWEEN ordenv1.i - 1 AND ordenv1.i
       ) env_duree_2_passe
```

Fig.4

ORD_NUMERO	ENV_NUMERODEP	ENV_DUREE	ENV_DUREE_PRECEDENT
2	1	56	(null)
2	2	88	56
2	3	112	88
2	4	32	112
2	5	103	32

Fig.5

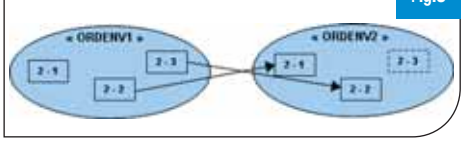
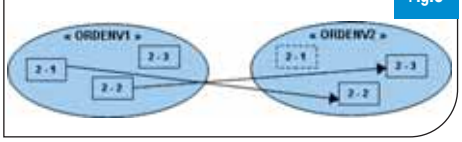


Fig.6



```
  FROM (
    SELECT ROWNUM i, ordenv.*
      FROM (
        SELECT *
          FROM v_ordre_envois
          ORDER BY ord_numero ASC, env_date_envoi ASC
      ) ordenv
  ) ordenv1
  WHERE ord_numero = 2;
```

Comme vous pourrez vous en rendre compte ci-dessous, il est possible d'arriver très simplement au même résultat avec une fonction d'analyse.

```
-- requête E07
SELECT ord_numero, env_numerodep, env_duree,
       SUM(env_duree) OVER
        (PARTITION BY ord_numero ORDER BY env_date_envoi ASC
          ROWS BETWEEN 1 PRECEDING AND CURRENT ROW) env_duree
_2_passe
  FROM v_ordre_envois
  WHERE ord_numero = 2
  ORDER BY ord_numero ASC, env_numerodep ASC;
```

Sur le même principe, nous souhaitons maintenant obtenir, pour chaque envoi, la somme cumulative de la durée écoulée [Fig.9].
Voici un exemple de requête pour obtenir ce résultat :

```
-- requête E08
SELECT ord_numero, env_numerodep, env_duree,
       ( SELECT SUM(env_duree)
           FROM v_ordre_envois
           WHERE ord_numero = ordenv.ord_numero AND
                 env_numerodep <= ordenv.env_numerodep
        ) env_duree_cumulée
  FROM v_ordre_envois ordenv
  WHERE ord_numero = 2;
```

Et voici la requête utilisant une fonction d'analyse :

```
-- requête E09
SELECT ord_numero, env_numerodep, env_duree,
       SUM(env_duree) OVER
         (PARTITION BY ord_numero
           ORDER BY env_date_envoi ASC) env_duree_cumulée
  FROM v_ordre_envois
  WHERE ord_numero = 2
  ORDER BY ord_numero ASC, env_numerodep ASC;
```

Nous allons encore étudier un dernier exemple. Nous aimerions finalement une requête retournant la durée totale par ordre ainsi que la durée globale de tous les ordres [Fig.10].
Voici un exemple de requête permettant d'obtenir ce résultat :

```
-- requête E10
SELECT ordenv1.ord_numero, ordenv1.env_numerodep, ordenv1.env_duree,
       ordenv2.env_duree_totale, ordenv3.env_duree_globale
  FROM v_ordre_envois ordenv1
    INNER JOIN (
      SELECT ord_numero, SUM(env_duree) env_duree_totale
        FROM v_ordre_envois
        GROUP BY ord_numero
    ) ordenv2
      ON ordenv1.ord_numero = ordenv2.ord_numero
    CROSS JOIN (
      SELECT SUM(env_duree) env_duree_globale
        FROM v_ordre_envois
    ) ordenv3
  WHERE ordenv1.ord_numero = 2
  ORDER BY ordenv1.ord_numero ASC, ordenv1.env_numerodep ASC;
```

Et voici la requête utilisant des fonctions d'analyse :

```
-- requête E11
SELECT ord_numero, env_numerodep, env_duree,
       SUM(env_duree) OVER
         (PARTITION BY ord_numero
           ORDER BY env_date_envoi ASC) env_duree_totale,
       SUM(env_duree) OVER
         () env_duree_globale
  FROM v_ordre_envois
  WHERE ord_numero = 2
  ORDER BY ord_numero ASC, env_numerodep ASC;
```

Vous pouvez facilement vous imaginer la complexité de la requête qu'il faudrait écrire pour agréger toutes ces informations au sein d'une seule et même requête [Fig.11].
Mais cela ne pose aucun problème avec les fonctions d'analyse :

```
-- requête E12
SELECT ord_numero, env_numerodep, env_duree,
       LAG(env_duree, 1)
         OVER (PARTITION BY ord_numero
           ORDER BY env_date_envoi ASC) env_duree_precedent,
       LEAD(env_duree, 1)
         OVER (PARTITION BY ord_numero
           ORDER BY env_date_envoi ASC) env_duree_suivant,
       SUM(env_duree) OVER(PARTITION BY ord_numero
         ORDER BY env_date_envoi ASC
         ROWS BETWEEN 1 PRECEDING AND CURRENT ROW) env_duree
_2_passe,
       SUM(env_duree) OVER (PARTITION BY ord_numero
         ORDER BY env_date_envoi ASC) env_duree_cumulee,
       SUM(env_duree) OVER (PARTITION BY ord_numero
         ORDER BY env_date_envoi ASC) env_duree_totale,
       SUM(env_duree) OVER () env_duree_globale
  FROM v_ordre_envois
```

Fig.7

ORD_NUMERO	ENV_NUMERODEP	ENV_DUREE	ENV_DUREE_PRECEDANT	ENV_DUREE_SUIVANT
2	1	56	(null)	85
2	2	85	56	112
2	3	112	85	32
2	4	32	112	103
2	5	103	32	(null)

Fig.8

ORD_NUMERO	ENV_NUMERODEP	ENV_DUREE	ENV_DUREE_2_PASSE
2	1	56	56
2	2	85	141
2	3	112	197
2	4	32	144
2	5	103	

Fig.9

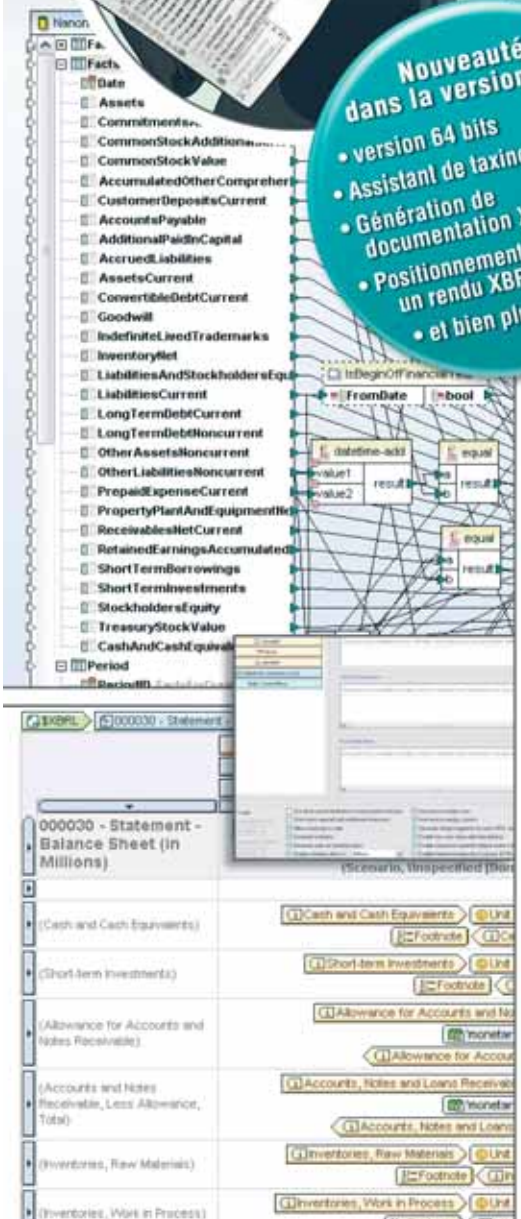
ORD_NUMERO	ENV_NUMERODEP	ENV_DUREE	ENV_DUREE_CUMULEE
2	1	56	56
2	2	85	141
2	3	112	253
2	4	32	285
2	5	103	388

Passez à la norme XBRL grâce aux différents outils Altova®

ALTOVA® missionkit®

Découvrez comment Altova MissionKit® 2010, la suite intégrée comprenant des logiciels de langage XML, de bases de données et d'intégration de données, vous offre tous les outils dont vous avez besoin pour travailler en XBRL, et tout cela sans vous ruiner.

Nouveautés dans la version 2010:
- version 64 bits
- Assistant de taxinomie XBRL
- Génération de documentation XBRL
- Positionnement absolu pour un rendu XBRL précis
- et bien plus encore...



Altova MissionKit 2010 inclue des outils intelligents pour éditer, mapper et publier du XBRL:

XMLSpy® – l'éditeur XML leader avec prise en charge du XBRL
- Validation de XBRL et Dimensions
- Edition graphique de la taxonomie XBRL

MapForce® – l'outil graphique de mappage et de conversion des données
- Mappage "any-to-any" de XBRL, bases de données, XML & données Excel 2007+
- Génération de rapports d'archivage XBRL par glisser-déposer-

StyleVision® – l'outil de conception de feuilles de style et de publication de rapports
- Publication simultanée de rapports financiers XBRL en HTML, PDF et Word
- Assistant intelligent pour présenter facilement le rendudes données XBRL

Téléchargement gratuit!

Testez le produit avec une version d'essai totalement fonctionnelle, à télécharger sur **www.altova.com**.

```
WHERE ord_numero = 2
ORDER BY ord_numero ASC, env_numerodep ASC;
```

Nous allons maintenant étudier l'impact sur les performances que peuvent avoir ces fonctions d'analyse en comparant plusieurs requêtes sur un jeu de test d'environ 210 000 enregistrements.

Quelles performances ?

Si nous reprenons notre requête numéro « E06 » vue précédemment qui nous rapporte la somme des durées de l'envoi courant et de l'envoi précédent, l'examen de sa trace nous indique un temps total de 18,75 secondes, comme nous le montre la capture d'écran provenant du fichier de trace de la base de données, ouvert avec Oracle SQL Developer : [Fig.12]. Si nous regardons maintenant la requête numéro « E07 » arrivant au même résultat en utilisant des fonctions d'analyse, nous pouvons constater que le temps total est descendu à 1 621 microsecondes, soit 0,001621 secondes [Fig.13]. Dans ce cas-là, nous constatons que le temps d'exécution de la requête « E07 » est 99,99% plus petit que le temps d'exécution de la requête « E06 ». Reprenons un autre exemple avec la requête numéro « E03 » également vue précédemment qui nous rapporte la durée de l'envoi précédent, nous voyons que le temps d'exécution s'élève à 3,57 secondes [Fig.14].

En revanche, le temps d'exécution de la requête « E04 » rapportant également les mêmes résultats mais en utilisant des fonctions d'analyse s'élève à 1629 microsecondes, soit 0,001629 secondes [Fig.15]. Nous constatons à nouveau que le temps d'exécution de la requête « E04 » est 99,95% plus petit que le temps d'exécution de la requête « E03 ».

Maintenant que nous avons vu les performances de ces fameuses fonctions d'analyse, il ne nous reste plus qu'à étudier leur syntaxe et leur comportement.

Syntaxe et comportement des fonctions d'analyse

Comme vous avez pu le voir au travers des différents exemples précédents, les fonctions d'analyse permettent d'effectuer des jointures réflexives sans qu'il soit nécessaire de décrire celles-ci explicitement. Cela à pour avantage, d'une part, de ne pas complexifier la requête et, d'autre part, d'offrir des performances améliorées.

Il faut savoir que les fonctions d'analyse sont invoquées après que toutes les jointures et les clauses WHERE, GROUP BY et HAVING ont été réalisées mais avant le ORDER BY de la requête.

Les fonctions d'analyse ressemblent fortement aux fonctions de groupe dans le sens ou elles s'appliquent toujours à un ensemble d'enregistrements.

```
FONCTION(param1, param2, ...) OVER (PARTITION BY colonne
  ORDER BY colonne ROWS BETWEEN r1 AND r2)
```

Le mot-clé « OVER » indique alors qu'il s'agit d'une fonction d'analyse et non pas d'une fonction de groupe. Le mot-clé « PARTITION » permet d'exprimer les critères de groupement comme on le ferait avec un « GROUP BY ». Le mot-clé « ORDER BY » permet d'indiquer l'ordre dans lequel les enregistrements du groupe doivent être triés. Si un critère de tri est précisé, alors la fonction s'applique à l'enregistrement courant, mais également aux enregistrements précédents ce qui permet par exemple d'effectuer facilement des sommes cumulatives. Cet ensemble d'enregistrements est appelé une fenêtre et celle-ci peut être affinée au moyen du mot-clé « ROWS ». Hormis « OVER » toutes les autres clauses sont facultatives. Sans partitionnement spécifique, la fonction s'applique à l'ensemble de la table ou de la vue en question.

Voici la liste des fonctions d'analyse supportées par le SGBD Oracle :

- AVG
- CORR
- COVAR_POP
- COVAR_SAMP
- COUNT
- CUME_DIST
- DENSE_RANK
- FIRST
- FIRST_VALUE
- LAG
- LAST
- LAST_VALUE
- LEAD
- MAX
- MIN
- NTILE
- PERCENT_RANK
- PERCENTILE_CONT
- PERCENTILE_DISC
- RANK
- RATIO_TO_REPORT
- REGR_SLOPE
- REGR_INTERCEPT
- REGR_COUNT
- REGR_R2
- REGR_AVGX
- REGR_AVGY
- REGR_SXX
- REGR_SYY
- REGR_SXY
- ROW_NUMBER
- STDDEV
- STDDEV_POP
- STDDEV_SAMP
- SUM
- VAR_POP
- VAR_SAMP
- VARIANCE

Conclusion

Cet article se veut une introduction aux fonctions d'analyse qui sont encore très peu connues des développeurs et n'aborde qu'une infime partie des possibilités offertes par cette technique. Nous encourageons fortement les lecteurs intéressés à consulter les quelques articles sur le Web et la documentation Oracle qui présentent plus en détail le mécanisme de ces différentes fonctions.

Références

- Normes SQL
- Fonctions SQL & normes SQL
- http://www.orafaq.com/node/55
- http://lalystar.developpez.com/fonctionsAnalytiques/

Professeur Ph. **Daucourt**
Avec la collaboration d'A. Gentet
Haute école de gestion Arc, HES-SO
philippe.daucourt@he-arc.ch

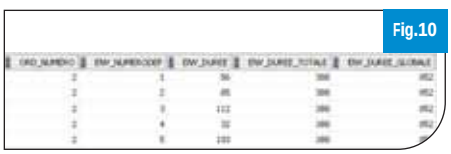


Fig.10

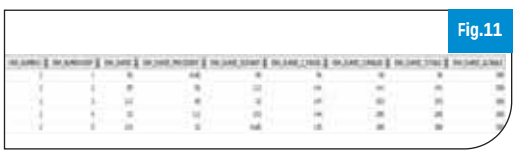


Fig.11

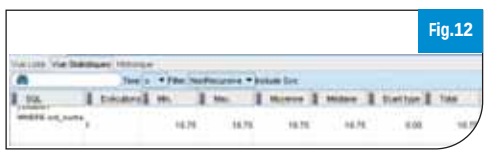


Fig.12

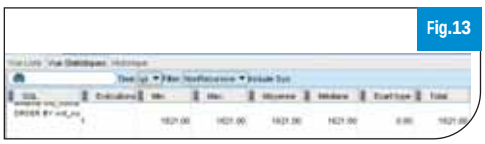


Fig.13

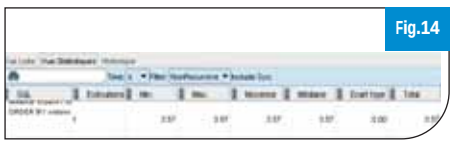


Fig.14

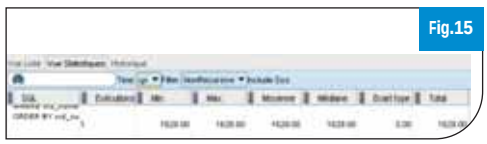
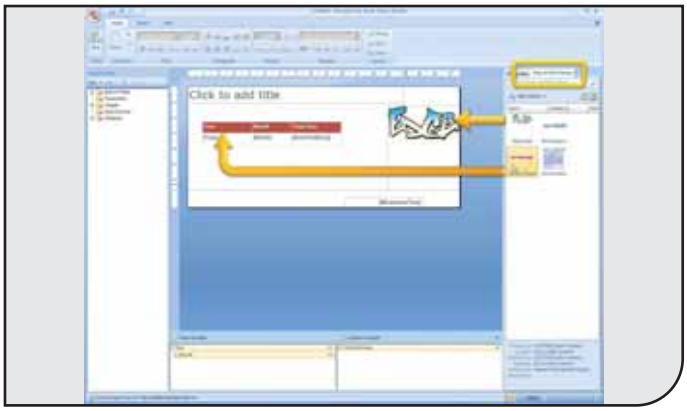


Fig.15

SQL Server 2008 r2 :
disponibilité courant mai



Très attendue par les développeurs et DBA, la R2 de SQL Server 2008 arrive. Elle est passée en version RTM vers le 20 avril. Cette version s'organise autour de trois axes principaux : Business Intelligence (en libre-service maîtrisé comme le dit si bien l'éditeur), très forte capacité de montée en charge et enfin une disponibilité du SGBD du Datacenter jusqu'au cloud.

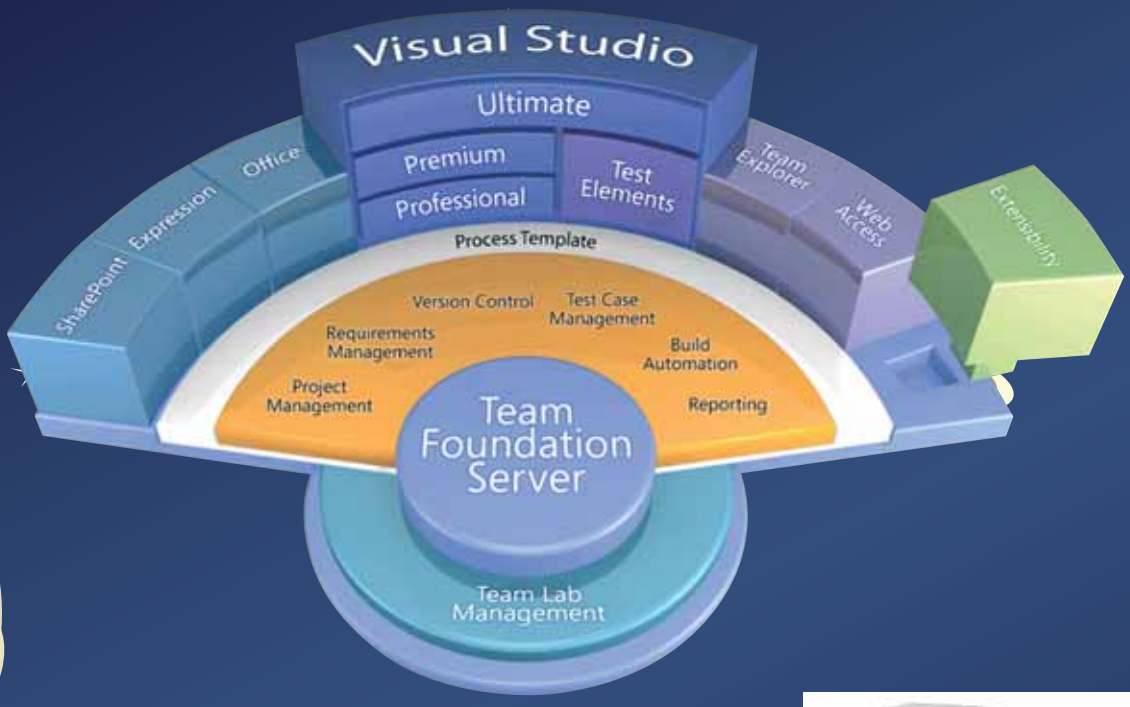
La BI est un des accès importants de SQL Server et la R2 apporte plusieurs nouveautés et améliorations. Tout d'abord, on peut mettre en œuvre l'outil PowerPivot qui permet facilement dans Excel de créer des rapports, d'agréger des données depuis d'énormes bases de données tout en laissant une maîtrise, un contrôle technique au service informatique. On dispose aussi du tout nouveau moteur de reporting Report Builder 3 et enfin de la composition de créer des mashup de reporting à partir de morceaux d'autres rapports, ce qui est particulièrement pratique !

Côté montée en charge, la R2 supporte jusqu'à 256 processeurs logiques, avec la possibilité d'être dans un Datacenter virtualisé avec Hyper-V. Et cette version supporte la migration dynamique (Live Migration) en contexte virtualisé. Dans cette version, d'autres améliorations incluent deux nouveaux composants de la plateforme d'information Microsoft : SQL Server 2008 R2 Master Data Services, qui aide les clients à assurer une meilleure cohérence des données entre des systèmes hétérogènes, et SQL Server 2008 R2 StreamInsight qui prend en charge des traitements de flux d'événements complexes à grande échelle (CEP – Complex Event Processing). Le CEP est un élément vital dans des contextes très complexes et à fortes contraintes de traitement, mais les outils CEP sont souvent très chers et d'une approche compliquée.

SQL Server 2008 R2 représente aussi l'engagement de Microsoft dans le cloud. La puissance de SQL Server s'étend au cloud via SQL Azure qui fournit un modèle de programmation cohérent, des outils et des fonctionnalités communs, tout en proposant de nouvelles opportunités métier, une haute disponibilité, une auto-administration et une capacité massive à monter en charge via des services de données distribuées dans le cloud. La récente version en avant-première du service Microsoft « Dallas » permet aux clients d'accéder à de riches ensembles de données fournis par des entreprises. Concernant SQL Azure, l'objectif à terme est d'avoir un SGBD le plus complet possible dans le cloud et non plus uniquement le moteur de données.

À noter que la version Express continuera à évoluer elle aussi. L'édition 2010 est prévue pour fin mai ou courant juin. ■

Visual Studio 2010
Cap sur le développeur !



Après plusieurs pré-versions, et plusieurs mois d'attente, Visual Studio 2010 est enfin disponible depuis quelques jours quand vous lirez ces lignes. Couplé au framework .Net 4.0, Visual Studio 2010 est plus que jamais le navire amiral du développeur Windows pour desktop, serveur, mobile, web et bien entendu le cloud.

Qui dit sortie événement, dit dossier événement de 20 pages dans votre magazine préféré ! Difficile de faire moins. Nous aborderons de nombreux sujets : les nouveautés de l'éditeur, l'intégration de l'outil Pex, la CLR 4, comment créer des protocoles de tests, migrer vos projets vers .Net 4 et VS 2010 ou encore découvrir le mot-clé dynamic dans .Net 4 ! Bref, ca va « poutrer » !

Pour Microsoft, et le développeur, Visual Studio 2010 est une version majeure comparée à l'édition 2005 ! Elle intègre toutes les nouvelles plates-formes web, mobile, serveurs de l'éditeur : Sharepoint 2010, Azure, Windows Phone,

Office 2010. Et Microsoft a enfin eu la bonne idée de simplifier la gamme en réduisant le choix entre 4 éditions, cinq si on rajoute la gamme Express qui sera, elle aussi, mise à jour. À vous de jouer !

■ François Tonic

Les nouveautés Visual Studio 2010

2010 est encore une année de nouveautés pour les produits Microsoft. Cette toute dernière version de Visual Studio arrive avec une gamme d'outils destinés à un large type de profils utilisateurs : Visual Studio 2010 (et ses déclinaisons Test, Developer…), Team Foundation Server, Expression Studio 4. Nous nous attarderons ici sur les changements de l'éditeur par rapport à la version 2008.

Depuis quelques mois, Microsoft s'efforce de simplifier au maximum le licensing de ses produits, c'est pour cela que les 7 éditions de Visual Studio 2008 (hors abonnement MSDN) se transforment en 3 versions 2010.

Si vous avez un abonnement MSDN dont la date d'expiration est postérieure au 12 avril 2010 (date de sortie de Visual Studio 2010), vous serez automatiquement migré vers les nouvelles versions (Professional, Premium ou Ultimate suivant la catégorie d'abonnement) et vous aurez également accès aux versions finales de Visual Studio 2010 qui correspondent à votre abonnement.

Par exemple, si vous avez une licence Visual Studio Team System 2008 Team Suite avec un abonnement MSDN, vous obtiendrez une licence Visual Studio 2010 Ultimate dans votre abonnement MSDN.

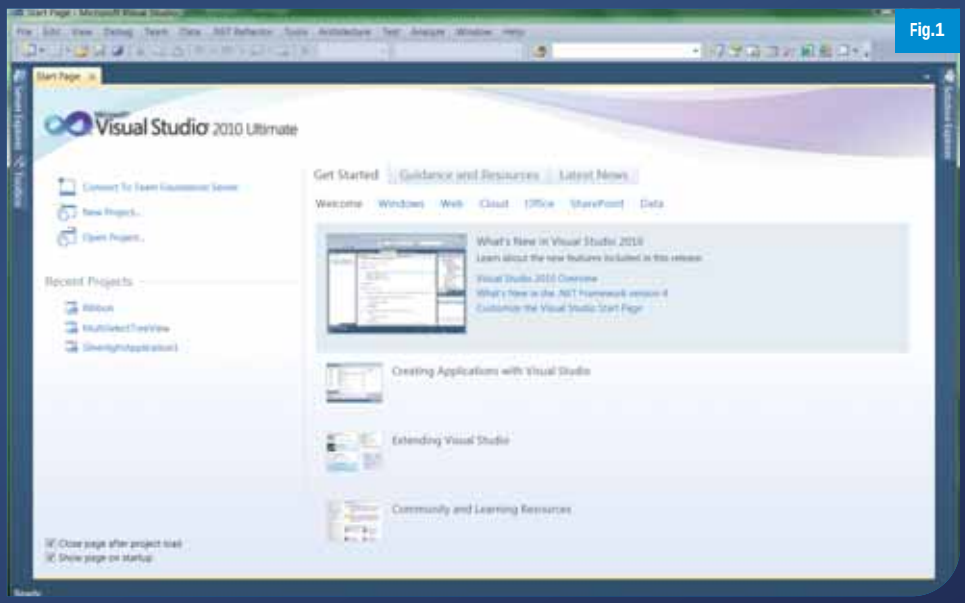


Fig.1

La page d'accueil [Fig.1].

Voici la toute nouvelle page d'accueil de Visual Studio 2010. Celle-ci utilise la technologie WPF (Windows Presentation Foundation) et peut être personnalisée très facilement. Pour cela, le SDK de Visual Studio 2010 et Expression Blend seront vos meilleurs amis. Vous pourrez non seulement personnaliser votre page d'accueil, mais également créer des compléments, de nouveaux menus… Bref de quoi créer un éditeur totalement adapté à vos équipes, à vos projets, vos clients. Sachez également que le SDK de la version 2010 pèse à peine plus de 10Mo, contre presque 100Mo pour la version 2008. La durée d'installation est dont logiquement revue à la baisse, car il faut moins d'une minute pour son installation.

La gestion des projets récents a été repensée et intègre la possibilité de pouvoir « épingler » ses projets favoris pour qu'ils ne disparaissent pas de la liste et pouvoir ainsi les rouvrir bien plus rapidement. [Fig.2]

Fig.2

Autre toute petite fonctionnalité, vraiment très simple, mais qui fait gagner quelques secondes au démarrage d'un projet, c'est la possibilité de fermer automatiquement la page d'accueil lorsque l'on charge un projet. Pour cela, il suffit de cocher la

Visual Studio 2008	→	Visual Studio 2010
Visual Studio 2008 Standard Edition Visual Studio 2008 Professional Edition		Visual Studio 2010 Professional
Visual Studio 2008 Professional Edition avec MSDN Professional Visual Studio 2008 Professional Edition avec MSDN Premium		Visual Studio 2010 Professional avec MSDN
Visual Studio Team System 2008 Architecture Edition avec MSDN Premium Visual Studio Team System 2008 Database Edition avec MSDN Premium Visual Studio Team System 2008 Development Edition avec MSDN Premium Visual Studio Team System 2008 Test Edition avec MSDN Premium		Visual Studio 2010 Premium avec MSDN
Visual Studio Team System 2008 Team Suite avec MSDN		Visual Studio 2010 Ultimate avec MSDN

case « Close page after project load » ou « Fermer la page après le chargement d'un projet » pour les versions françaises. [Fig.3]

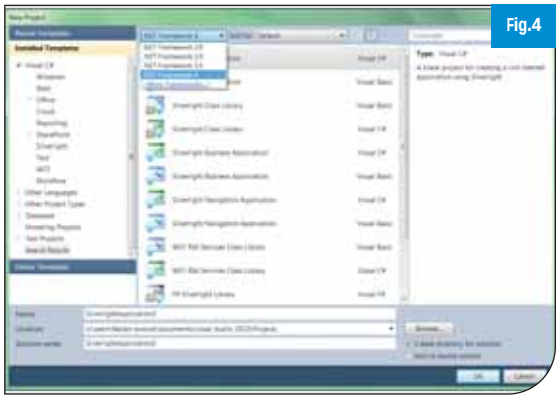
Création de projets

[Fig.4]. Comme dans l'édition précédente, lors de la création d'un nouveau projet, vous pouvez choisir la version du .NET Framework (notez que les versions 1.0 et 1.1 ne sont pas disponibles), avec une possibilité d'intégrer de futures éditions du .NET Framework ou, pourquoi pas, d'autres frameworks.

La zone de saisie en haut à droite de la fenêtre vous permet de rechercher rapidement des modèles de projets disponibles. Cette fonctionnalité a été intégrée à plusieurs reprises dans les fenêtres de l'éditeur, par exemple lorsque vous souhaitez ajouter un nouvel élément à un projet en cours. Remarquez qu'il est également possible de récupérer automatiquement des modèles mis à disposition sur le site Visual Studio Gallery (http://visualstudiogallery.msdn. microsoft.com).

Fig.3

Fig.4



L'éditeur

L'éditeur de code a également été entièrement réécrit en WPF pour profiter de la fluidité et de la qualité graphique ainsi que de toutes les possibilités qu'apporte cette technologie.

L'inconvénient d'une interface WPF comme celle-ci est peut-être que l'on peut constater quelques lenteurs à l'utilisation pour des machines peu puissantes, bien que de nets progrès sont constatés entre la bêta 2 et la version finale.

Enfin un vrai designer pour Silverlight

Il était temps ! Visual Studio 2010 intègre désormais un designer pour les projets Silverlight, dans sa version 2 et 3 pour le moment, et en version 4 à la sortie de ce dernier, prévue après la sortie de Visual Studio.

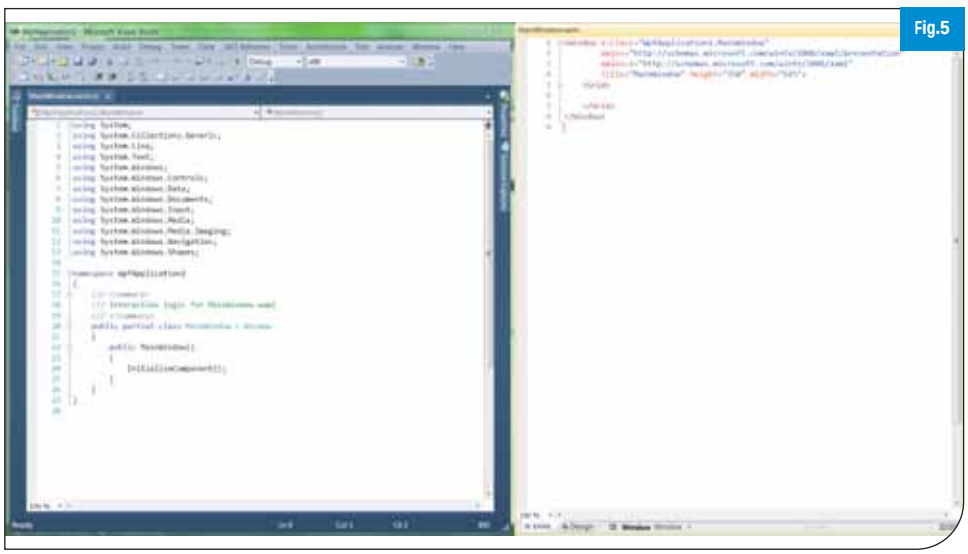
Concrètement qu'est ce que cela signifie ? Pour de petites modifications de votre code XAML, il n'est plus nécessaire d'ouvrir Expression Blend qui peut, pour certains, être assez complexe à utiliser lorsque l'on n'est pas designer ou ergonome.

Disposition des fenêtres

Dans les éditions précédentes, il était possible de déplacer les fenêtres telles que la boîte à outils ou les propriétés, en dehors de la fenêtre principale. Il était donc assez fastidieux de travailler sur la partie graphique d'une fenêtre par exemple et de son code behind, car il fallait à chaque fois basculer d'un onglet à l'autre. [Fig.5]

Comme vous pouvez le voir sur l'image, il est maintenant possible de déplacer une page de code ou une fenêtre de designer en dehors de l'éditeur. Les possesseurs d'un double écran vont être ravis de pouvoir travailler à la fois sur plusieurs fichiers en même temps sur chaque écran. De plus, si vous travaillez sous Windows 7, vous pourrez profiter des nouvelles fonctionnalités de déplacement et d'ancrage des fenêtres sur le bureau.On peut toutefois regretter de ne pouvoir séparer le code XAML d'une page de sa représentation graphique dans le designer.

Fig.5



Zoom dans l'éditeur de code

Cette fonctionnalité va ravir tous les développeurs lors des présentations sur vidéoprojecteurs ou pour des enregistrements de webcast. En effet, qui n'a jamais perdu de longues minutes pour trouver la bonne taille de police de caractères qui offrira le meilleur rendu ?

C'est terminé, WPF offre un rendu vectoriel des polices et objets graphiques, ce qui permet un zoom quasi illimité. En appuyant sur la touche Ctrl et en utilisant la molette de la souris, vous pourrez zoomer et dézoomer sur le code, comme sur l'image [Fig.6] où l'on passe d'une taille normale à un zoom à 146% en l'espace de moins d'une seconde. [Fig.7]

Bien sûr, ceci fonctionne parfaitement avec du code XAML comme du code C# ou VB.NET... Remarquez également que lorsque l'on passe la souris sur l'image , la région concernée se grise.

Ajouter une référence à un projet

La fenêtre de chargement des références a été revue et corrigée. Jusqu'à la version 2008, lorsque vous souhaitiez ajouter une référence à un projet, la fenêtre s'ouvre et tant que

toutes les références de la GAC (Global Assembly Cache) ne sont pas chargées, toute l'interface reste bloquée, il n'était plus possible d'agir pour choisir une référence de projet. C'est pourquoi les développeurs ont trouvé une parade, pas forcement la meilleure. Les références se chargent de manière asynchrone en quelques secondes.

L'avantage de cette solution, est que la fenêtre n'est pas bloquée et donc si vous êtes sur l'onglet des références .NET, vous pouvez instantanément naviguer vers l'onglet Projet et ajouter votre référence de projet.

L'inconvénient est que lorsque vous cherchez une référence particulière, vous vous positionnez aux environs de celle-ci dans la liste et des dizaines d'autres références apparaissent, ainsi vous vous retrouvez rapidement noyé au milieu des références. La solution est donc d'attendre que toutes les références aient été chargées.

IntelliSense

L'IntelliSense a grandement été améliorée pour nous faciliter encore plus le travail de développement.

Déjà, lorsque vous entrez un mot, l'IntelliSense filtrera automatiquement toutes les méthodes, objets... qui contiennent ce mot, peut importe la position de celui-ci.

Comme vous pouvez le constater sur l'image, le mot « action » est contenu dans trois méthodes et un objet, l'IntelliSense ne vous offre que ces quatre propositions, alors que dans les éditions précédentes, vous n'auriez eu aucun filtrage, toutes les entrées accessibles vous seraient proposées. [Fig.8]

Ensuite, très souvent, par convention, lorsque l'on crée une méthode qui contient plusieurs mots, chacun d'entre eux commencera par une majuscule. Prenons l'exemple de la méthode « DoItNow ».

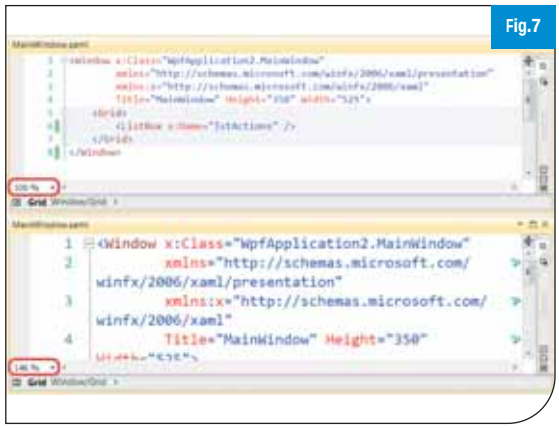
Si vous entrez uniquement les trois lettres en majuscules, à savoir « DIN », Visual Studio vous proposera automatiquement la méthode « DoItNow ». Dans les éditions de Visual Studio précédentes, comme pour la fonctionnalité précédente, vous n'auriez eu aucun résultat. [Fig.9] Bien sûr, je vous propose ces exemples en C# mais tout fonctionne avec les autres langages.

Fig.6

```
<Window x:Class="WpfApplication2.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/
presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
    <Grid>

    </Grid>
</Window>
```

Fig.7



Mise en valeur des références

Il peut être parfois assez difficile de rechercher toutes les références à un élément dans une page de code. Cliquez simplement sur la référence que vous souhaitez rechercher, attendez une petite seconde et toutes les références de votre objet ou méthode seront surlignées. [Fig.10]

Conclusion

Pour conclure, cette nouvelle édition de Visual Studio 2010 est riche en nouveautés, tant par ses nombreuses fonctionnalités qui facilitent la vie du développeur, que par les supports de développement. En effet Visual Studio est l'outil indispensable, tout en un, pour pouvoir développer des applications avec les technologies « cloud » de Windows Azure, Windows Phone 7, mais également les nouvelles versions des technologies WPF 4.0, Silverlight 4.0, .NET Compact Framework 4.0, .NET Micro Framework 4.0, ASP.NET 4, VSTO pour Microsoft Office 2010.

■ Fabien **Lavocat**

Fig.8

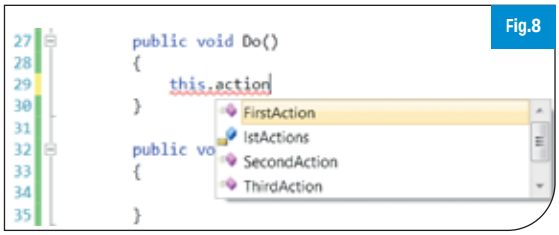


Fig.9

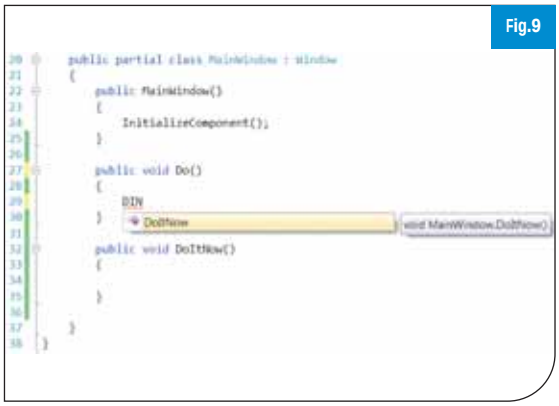


Fig.10

```
        public void Init()
        {
            this.lstActions.ItemsSource = this._dataSource;
            this.lstActions.SelectionChanged += new SelectionChangedEventHandler(lstActions_SelectionChanged);
        }

        private void lstActions_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            String selectedItem = this.lstActions.SelectedItem as String;
            MessageBox.Show(selectedItem);
        }
```

Migration du code existant vers .Net 4 et Visual Studio

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

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

Compatibilité des versions

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

La migration simple...

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

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

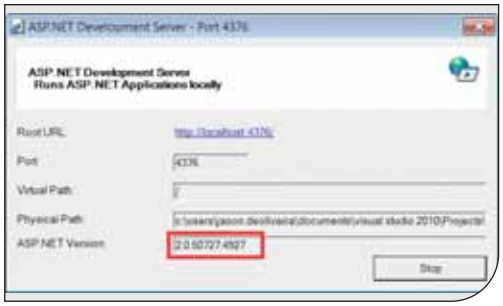
Version du Framework .NET	Version
4	v4.0
3.5	v2.0.50727
2.0	v2.0.50727
1.1	v1.1.4322
1.0	v1.0.3705

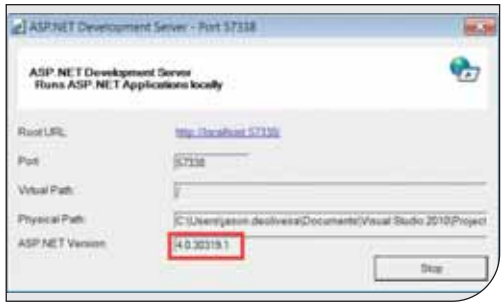
Et voici un exemple de fichier dans le cas de migration vers .NET 4 :

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

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

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





egilia®

LEARNING

LE SPÉCIALISTE DE LA
FORMATION CERTIFIANTE
EN INFORMATIQUE
ET MANAGEMENT

Faire de vos succès
notre réussite

www.egilia.com

CONTACTEZ NOS CONSEILLERS FORMATION

N°National 0 800 800 900

APPEL GRATUIT DEPUIS UN POSTE FIXE

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

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

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

L'assistant de conversion

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

Ce qu'il faut savoir…

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

Core

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

ASP.NET

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

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

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

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

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

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

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

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

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

Windows Presentation Foundation (WPF)

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

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

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

L'attribut suivant n'est plus valable :

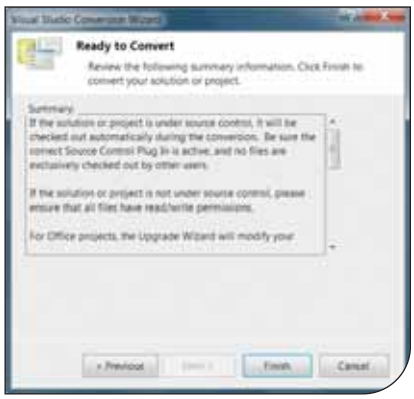
```
<button Control.Button.Background="Red"/>
```

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

Les fonctionnalités obsolètes

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

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



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

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

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

Types obsolètes

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

Membres obsolètes

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

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

Stratégies de tests

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

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

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

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

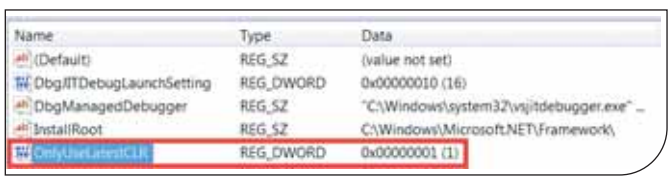
Ces tests sont à effectuer avant l'installation du Framework 4, l'application tournant alors sous l'ancienne version de .NET, ainsi qu'après la migration vers le Framework 4 via le fichier de configuration ou la modification de la clé. Par exemple, pour les applications Web, les scénarios de tests comprennent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

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

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

Conclusion

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



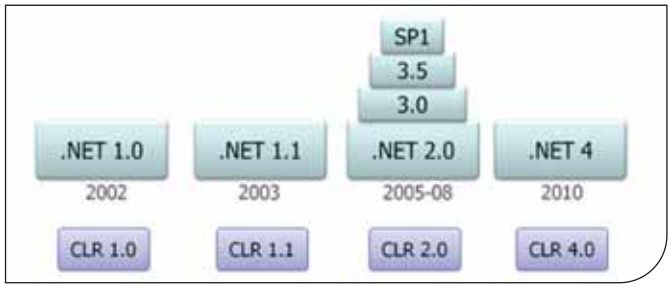
■ Jason **De Oliveira**
Solutions Architect chez Winwise, 13 ans d'expérience dans le développement logiciel.
Son Blog: http:// jasondeoliveira.com

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

CLR 4.0 : la première version majeure depuis la CLR 2.0

Intéressons-nous à la toute nouvelle CLR 4.0 qui accompagne la sortie de .Net 4. Il ne s'agit pas d'une simple mise à jour avec des rajouts de fonctionnalités comme pour les frameworks 3 et 3.5, mais bien d'une toute nouvelle version !

La CLR 4 permet de développer avec de nouveaux langages comme IronPython, F#... Il est aussi possible de faire tourner la CLR 2.0 dans le même processus que la CLR 4.0 contrairement à l'ancienne CLR. On appelle cela le "In Process Side by Side" SxS, ce qui est particulièrement intéressant. L'interopérabilité est améliorée grâce à un générateur P / Invoke (P / Invoke Interop Assistant) disponible sur codeplex. Il en va de même pour le garbage collector qui a lui aussi droit à une petite rénovation (GC server, GC client en background pour éviter de "freezer" le thread en cours...).



Nouveaux types

La CLR apporte de nouveaux types :

- BigInteger (System.Numeric.BigInteger) vous permet d'utiliser de gros numériques suivant l'espace mémoire disponible.
- Tuple vous permet d'avoir plusieurs valeurs de retour :

```
1. private void Form1_Load(object sender, EventArgs e)
2. {
3.     MessageBox.Show(GetResults(12, "juliend", 12.0).Item1.
ToString());
4. }
5.
6. public Tuple<int, string, double> GetResults(int arg1, string
arg2, double arg3)
7. {
8.     return new Tuple<int, string, double>(arg1, arg2, arg3);
9. }
```

Code Contract

Il est désormais possible de développer avec du "Code Contract". Cette nouvelle fonction vous permet de poser des conditions avant (Requires), pendant (Invariant) et après (Ensures) une méthode. Si la condition n'est pas respectée, une exception de type ContractException est levée. Afin de profiter du Code Contract, il vous faut installer le Managed Contract Tools.

Puis, direction les propriétés de votre projet pour activer cette fonctionnalité à l'exécution et à la compilation : [Fig.1]

Prenons un exemple. Contract.Requires (condition booléenne) permet de poser une pré-condition à notre méthode. Ce qui déclenchera un warning lors de la compilation si elle n'est pas respectée, et une erreur à l'exécution.

```
1. public string Echo(string s)
2. {
3.     Contract.Requires(s != null);
4.     MessageBox.Show(s);
5.     return s;
6. }
```

[Fig.2]

De même pour Contract.Ensures (condition); pour les posts conditions et Contract. Invariant (condition); pour vérifier une variable tout au long du déroulement du traitement.

Managed Extensibility Framework

Nous allons voir un cas pratique, conception d'une architecture logicielle d'un projet WPF composite avec un pattern MVVM pour chaque module. Le but de MEF est de créer une application dite

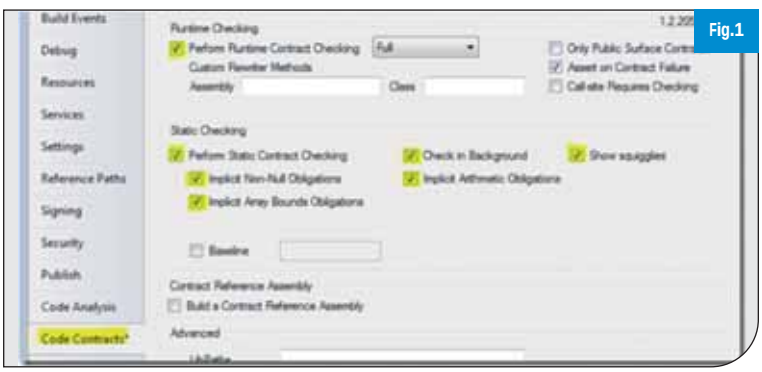


Fig.1

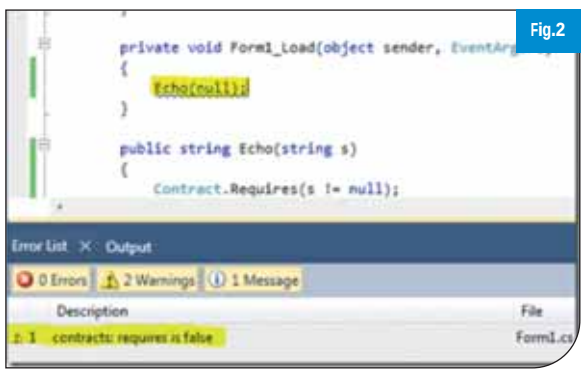


Fig.2

"composite" (un peu comme Prism v2). On va pouvoir ajouter des fonctionnalités à notre logiciel avec le simple ajout de dll dans un dossier spécifique. Chaque module est indépendant et peut être architecturé de la façon que vous souhaitez (n-tiers, MVVM, MVC...). Dans notre cas nous allons utiliser MVVM, puisque c'est le pattern le plus en vogue pour WPF en ce moment.

Création de l'interface commune à tous les modules dans une classlibrary

```
1. public interface IModule
2. {
3.      //Nom du module
4.      string Name { get; set; }
5. }
```

Développement de notre premier module

Ajoutez à votre solution un projet WPF, supprimez le fichier app.xaml et changez les options de compilation : [Fig.3]
Puis architecturez votre module grâce à MVVM : [Fig.4]
Puis appliquez certaines modifications à votre projet pour qu'il devienne un module MEF :

1 • ViewProduct doit implémenter IModule

```
1. public partial class ViewProduct : UserControl, IModule
2. {
3.      public ViewProduct()
4.      {
5.          InitializeComponent();
6.          DataContext = new ViewModelProduct();
7.      }
8.
9.      string IModule.Name
10.      {
11.          get;
12.          set;
13.      }
14. }
```

2 • Ajout de la référence System.ComponentModel.Composition à votre projet [Fig.5]

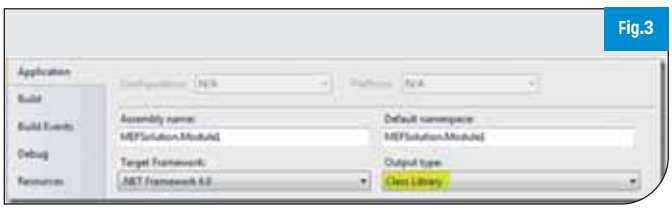


Fig.3

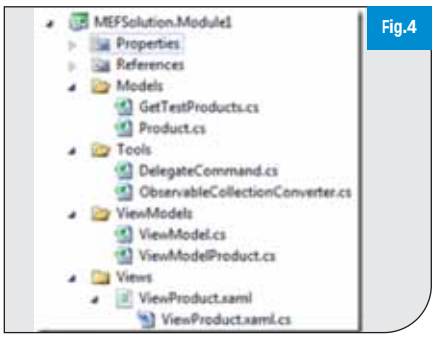


Fig.4

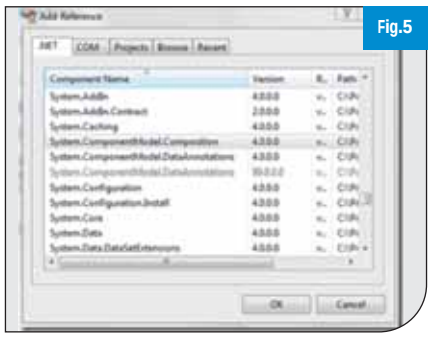


Fig.5

3 • Ajout de l'attribut Export et ExportMetaData au-dessus de votre View :

```
1.  [Export(typeof(IModule))]
2. [ExportMetadata("Name", "Products")]
```

À ce stade, votre class library est configurée pour MEF.

Création du projet principal qui chargera le module

Créez un projet WPF et ajoutez la référence au projet contenant IModule ainsi que System.ComponentModel.Composition.
Xaml :

```
1. <StackPanel x:Name="stack"/>
```

Code Behind :

```
1. public Window1()
2. {
3.      InitializeComponent();
4.      AggregateCatalog catalog = new AggregateCatalog();
5.      catalog.Catalogs.Add(new DirectoryCatalog("."));
6.      this.modulesContainer = new CompositionContainer
(catalog);
7.      this.modulesContainer.ComposeParts(this);
8.
9.      foreach (var module in this.Modules)
10.      {
11.          this.stack.Children.Add(new TextBlock(){ Text =
(String)module.Metadata["Name"] });
12.          this.stack.Children.Add(module.GetExportedObject
() as UserControl);
13.      }
14. }
15.
16. [Import(typeof(IModule))]
17. public ExportCollection<IModule> Modules { get; set; }
18.
19. public CompositionContainer modulesContainer { get; set; }
```

DLR et C# 4

La DLR a été introduite afin de supporter les nouveaux langages dynamiques dans .net (IronPython, IronRuby et maintenant C# et VB.NET). Il s'agit d'une surcouche à la CLR qui permet, entre autres, de remplacer la « reflexion ». La DLR apporte d'intéressantes nouveautés, tout comme C# 4. Sur la DLR, reportez-vous à l'article qui lui est consacré dans le présent dossier. C# qui est

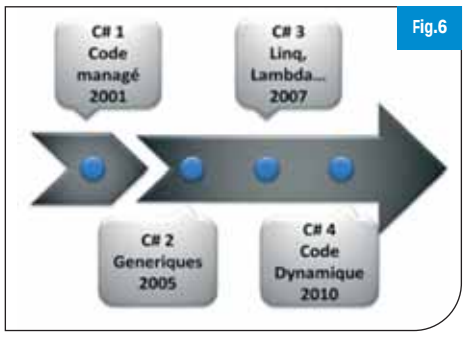


Fig.6

un langage impératif / statique permet d'être performant, robuste et surtout analysable afin d'avoir des outils tels que l'intelliSense. Les langages dynamiques tels que PHP ou JavaScript ont la puissance de pouvoir embarquer du code non compilé, par exemple en JavaScript :

```
1. var1 = "juliend"; //Type string
2. alert(var1);
3. var1 = 2; //On passe en int
4. alert(var1);
```

Vous remarquez que l'on peut typer à la volée la variable et que le compilateur ne peut déterminer son type. Le JavaScript perd donc en sécurité mais on gagne en souplesse.

Les nouveautés du C#4

Deux belles nouveautés à noter : les paramètres et les aspects dynamiques [Fig.6].
C#4 permet enfin d'avoir des paramètres par défaut comme en C++ mais aussi l'utilisation de paramètres nommés :

```
1. public int Test(int i, int b = 2)
2. {
3.     return i + b;
4. }
5.
6. private void Form1_Load(object sender, EventArgs e)
7. {
8.     MessageBox.Show(Test(2).ToString());
9. }
```

```
1. public int Test(int i, int b = 2)
2. {
3.     return i + b;
4. }
5.
6. private void Form1_Load(object sender, EventArgs e)
7. {
8.     MessageBox.Show(Test(b:3, i:7).ToString());
9. }
```

Avant C# 4, l'utilisation de la réflexion était fréquente. Par exemple si nous avions une variable dont on ne connaissait pas le type mais dont nous savions qu'il existait la méthode HelloWord, alors nous l'utilisions comme ceci :

```
1. class Class1
2. {
3.     public void HelloWord(string t)
4.     {
5.         MessageBox.Show("Hello " + t);
6.     }
7. }
```

```
1. public object GetObject()
2. {
3.     return new Class1();
4. }
5.
6. private void Form1_Load(object sender, EventArgs e)
7. {
8.     object monobjet = this.GetObject();
9.     Type montype = monobjet.GetType();
10.     montype.InvokeMember("HelloWord",
11.     BindingFlags.Default |
12.     BindingFlags.Public | BindingFlags.Public |
13.     BindingFlags.Instance | BindingFlags.InvokeMethod,
14.     null, monobjet,
15.  object[] { "Test" });
16. }
```

Il est désormais possible d'utiliser la DLR pour remplacer la réflexion, voici comment faire (toujours avec Class1) :

```
1. dynamic monobjet = this.GetObject();
2. monobjet.HelloWord("test");
```

La méthode HelloWord ne possède pas d'intelliSense, car c'est à l'exécution qu'il va générer de la réflexion (compilation à la volée). Attention à ne pas abuser de ce type (comme en PHP par exemple) pour ne pas se retrouver à faire :

```
1. dynamic monobjet = "lol";
2. monobjet = 2;
3. MessageBox.Show(monobjet.ToString());
```

Parallel Extensions : vive le parallélisme

La programmation parallèle s'impose de plus en plus, puisque nos CPU comportent désormais au moins 2 cœurs. Mais la programmation parallèle, de quoi s'agit-il ? C'est la possibilité d'exploiter ces cœurs dans le code de tous les jours. Cependant, cela implique d'utiliser des deadlocks, des mutex etc... et cela devient vite compliqué. Parallel Extensions, anciennement appelé ParallelFX, va nous permettre de faciliter tous ces mécanismes. Visual Studio 2010 et .Net 4 apportent de belles nouveautés sur le sujet que Programmez ! a abordé ces derniers mois. Revoyons-les rapidement ici.

Task Parallel Library (TPL)

Prenons comme exemple une boucle qui parcourt des objets (de façon ridicule, le but étant que le temps d'exécution soit lent) :

```
1. Stopwatch watch = Stopwatch.StartNew();
2. foreach (Utilisateur item in mesUsers)
3. {
4.     foreach (Role r in mesRoles)
5.         if (r.ID.Equals(item.Role.ID))
6.         {
7.             for (i = 0; i < 1000; i++)
8.             {
9.                 b++;
10.             }
11.         }
12. }
13. watch.Stop();
14. MessageBox.Show(watch.ElapsedMilliseconds.ToString());
```

[Fig.7]

Nous allons maintenant utiliser la programmation parallèle afin d'accélérer notre code. Pour cela, le framework nous met à disposition la classe Parallel contenant la méthode statique ForEach. L'un des problèmes de ce code est que la variable b risque de ne pas contenir la bonne valeur (plusieurs threads incrémentant en même temps par exemple). Pour cela, on contourne le problème avec la classe InterLocked qui incrémentera notre variable de façon atomique.

```
1. watch = Stopwatch.StartNew();
2. //Syntaxe: Parallel.ForEach<Type à parcourir>
3. //Puis n=> est une expression lambda
4. Parallel.ForEach<Utilisateur>(mesUsers, item =>
5. {
6.     Parallel.ForEach<Role>(mesRoles, r =>
7.     {
8.         if (r.ID.Equals(item.Role.ID))
9.         {
10.             Parallel.For(0, 1000, z =>
11.             {
12.                 Interlocked.Increment(ref b);
13.             });
14.         }
15.     });
16. });
17. watch.Stop();
```

[Fig.8]

Notre boucle est désormais réalisée en programmation parallèle. Vous remarquez cependant que l'on perd du temps plutôt que d'en gagner ! En effet, les parallèles extensions mettent en place beaucoup d'éléments pour réaliser votre boucle, ce qui n'est pas intéressant pour aussi peu de données. Le gain de temps aurait pu s'effectuer si j'avais eu plus de traitement à faire dans mes boucles.

Parallel Linq (PLINQ)

Il existe la possibilité de paralléliser les requêtes Linq-To-Xml et Linq-To-Object. Prenons par exemple :

```
1. Stopwatch watch = Stopwatch.StartNew();
2. List<Utilisateur> mesUsers2 = (from u in mesUsers
3.                                orderby u.Login ascending
4.                                  select u).ToList();
5. watch.Stop();
6. MessageBox.Show(watch.ElapsedMilliseconds.ToString());
7. watch = Stopwatch.StartNew();
8. mesUsers2 = (from u in mesUsers.AsParallel()
9.              orderby u.Login ascending
10.              select u).ToList();
11. watch.Stop();
12. MessageBox.Show(watch.ElapsedMilliseconds.ToString());
```

Le code effectue un tri sur une liste, l'un en Linq et l'autre en PLinq grâce à .AsParallel(). Vous pouvez constater que dans les deux cas, tous les processeurs sont utilisés sauf que PLinq utilisera de façon optimale vos processeurs. Plus vos listes sont importantes, plus l'utilisation de PLinq s'impose :

[Fig.9] Avec PLinq

[Fig.10] Sans PLinq

■ Julien **Dollon**

MVP | Consultant Trainer l'FORM | Dotnet-France Lead Manager | SUPINFO Full Professor .NET Technologies

Visual Studio 2010, un environnement simple pour exécuter des tests de charge

Visual Studio 2010 sort avec le .NET framework 4.0 courant avril. Et c'est avec son lot de nouveautés que cette version va venir frapper une nouvelle fois le monde du développement. Connu et reconnu comme l'un des meilleurs environnements de développement, Visual Studio 2010 évolue encore pour faciliter le développement et le déploiement des solutions .NET avec pour même objectif depuis 10 ans, d'augmenter la productivité des développeurs, et de faciliter la mise en place des tests…

Un des domaines qui a évolué notamment sur la configuration et la mise en place d'une architecture est celui des tests. En effet, au travers de cet article, nous allons voir que la mise en place d'un environnement de tests, en particulier pour les tests de charge, est grandement facilitée, de son installation à son exploitation, en passant par sa configuration.

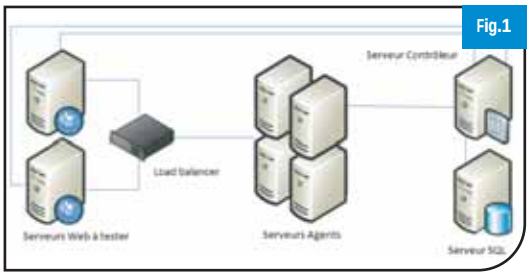


Fig.1

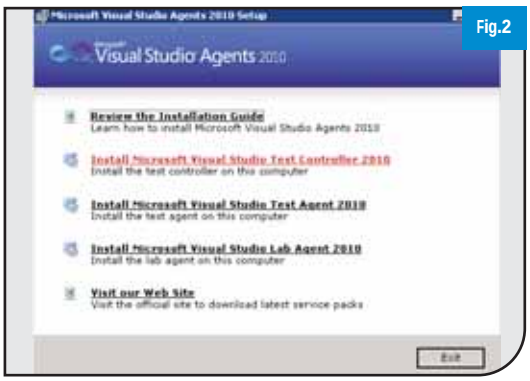


Fig.2

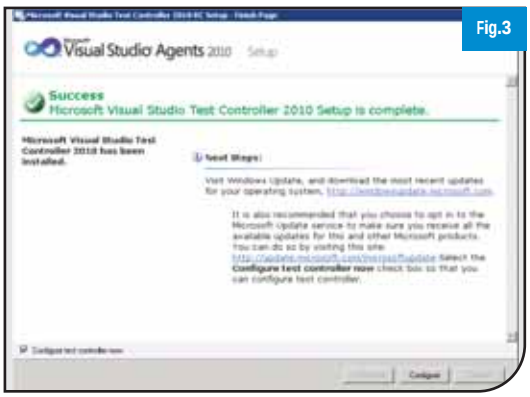


Fig.3

Architecture

Voici un exemple d'architecture [Fig.1] que l'on pourrait mettre en place pour des tests de charge. Les machines agents simulent des utilisateurs virtuels et donc des requêtes vers les machines que l'on souhaite tester, en l'occurrence des frontaux Web. Notre serveur contrôleur lit les compteurs de performance sur les frontaux Web que l'on teste dans cet environnement. Il stocke les résultats au sein d'une instance SQL Server. On peut également remarquer la présence d'un load balancer dont le rôle sera de répartir les requêtes entre les frontaux. Dans les tests de charge sous Visual Studio, on a pris en compte ce type d'architecture pour simuler des plages d'adresses IP sur nos agents afin qu'un load balancer ait l'impression qu'à chaque requête c'est une nouvelle machine. Cela nous permettra par exemple de nous affranchir de problèmes comme le « stick IP » de certains boîtiers et de fausser ainsi les résultats. Pour rappel, le « stick IP » permet d'associer un client avec un serveur, typiquement ce que l'on veut éviter dans nos tests de charge.

Installation et configuration de l'environnement

L'installation de l'environnement de tests passe forcément par un travail en amont à définir les besoins réels pour les tests. Typiquement, on devra définir la capacité physique des machines, le nombre de machines nécessaires pour simuler nos charges, les types de réseau, de navigateurs Internet dans le cas d'applications Web... En termes de capacité physique, il faudra porter une attention particulière à la mémoire vive à la fois pour les agents comme le contrôleur. Une fois ces points précisés, on peut alors commencer à installer l'environnement. La première étape consiste à installer le service contrôleur. Son rôle est de gérer les différentes actions d'un test et on peut citer notamment les points suivants :

- Lancer, arrêter, gérer la configuration des tests
- Lire les compteurs de performance des machines testées ou des agents, stocker ces informations dans une base de données dédiée au produit appelée LoadTest
- Gérer la répartition des requêtes entre les machines agents

En plus de la machine dite contrôleur, on trouvera N machines dites agents qui servent à la simulation en masse d'utilisateurs virtuels et donc de requêtes.

Pourquoi commencer par l'installation du contrôleur ? On verra que dans la phase de configuration des agents, il faut forcément associer un agent à un contrôleur. Il faut souligner qu'il y a une installation spécifique du contrôleur et de l'agent, ce n'est pas présent avec l'installation de Visual Studio 2010.

L'installation du contrôleur est très simple, elle se compose de cinq écrans successifs qui ne présentent aucune difficulté particulière [Fig.2].

Notez toutefois que pendant l'installation du contrôleur, le framework .NET 4.0 sera déployé s'il n'est pas déjà présent sur la machine. Cette installation néces-

site un redémarrage de la machine pendant l'installation [Fig.3]. Une fois l'installation faite, une nouveauté par rapport à Visual Studio 2008 est d'avoir un assistant de configuration. Cet assistant a pour objectif de définir différents paramètres :

- Le compte de service qui sera utilisé pour exécuter le service Windows associé « Visual Studio Test Controller ». Ce compte peut être Network Service ou un compte de domaine. De manière générale, dans un environnement de tests complet d'entreprise, on utilisera bien entendu un compte de domaine.
- Nous avons dorénavant la possibilité d'intégrer notre environnement de tests avec une collection sous Team Foundation Server (TFS).
- Un élément déterminant pour le stockage des données des tests est bien entendu de disposer d'une instance SQL Server. En indiquant le nom du serveur et l'instance, l'assistant créera pour vous automatiquement la base de données *LoadTest2010*. C'est une réelle différence par rapport aux versions précédentes où il fallait exécuter manuellement un script SQL pour créer la base *LoadTest* [Fig.4].

Une fois ces paramètres renseignés, l'assistant va exécuter un certain nombre d'étapes :

- Démarrage du service Windows
- Création de la base de données
- Création d'une exception dans le pare-feu Windows sur le port TCP 6901
- Gestion de la sécurité

A noter également que durant cette phase, un avertissement sera probablement levé en vous indiquant que le compte du contrôleur doit avoir des droits spécifiques sur les machines dont on souhaite lire les compteurs de performance [Fig.5].

En cas de problème inattendu, on peut vérifier les logs à l'emplacement suivant (chemin par défaut d'une plate-forme 64bits) : C:\Program Files (x86)\Microsoft Visual Studio 10.0\Microsoft Visual Studio Test Controller 2010 ENU\Logs

Une fois le contrôleur déployé, il nous reste à déployer les agents. Nous allons voir que leur installation et encore plus leur configuration a largement été simplifiée. Sur le même setup que le contrôleur, il suffit de lancer l'installation du service agent sur chacune des machines portant ce rôle dans notre environnement de tests [Fig.6]. Tout comme l'installation du service contrôleur, celle du service Agent inclut le déploiement du framework .NET 4.0 qui nécessite un redémarrage de la machine [Fig.7]. Une fois l'installation finie, on a la possibilité de lancer l'assistant de configuration. La première étape consiste à définir le mode de l'agent. C'est une nouveauté, on peut ainsi définir l'agent comme simple service Windows ou comme un processus interactif qui permet d'interagir avec l'interface graphique.

La seconde étape permet de configurer deux points fondamentaux :

- Le compte utilisé pour exécuter le service Windows. Deux possibilités s'offrent à vous : utiliser le compte network service ou un compte de domaine. Vous noterez que si vous définissez un compte de domaine, ce compte doit avoir les privilèges d'administrateur local sur la machine agent.
- Ce deuxième paramètre est lui aussi une vraie nouveauté dans cette mouture 2010 puisqu'en 2008 nous ajoutions les agents depuis une instance Visual Studio connectée à service contrôleur. Dorénavant, vous devez spécifier un contrôleur pour compléter la configuration d'un agent.

  Ce contrôleur sera de la forme *nom_machine :6901* [Fig.8].

Une erreur typique que l'on peut rencontrer lors de ce processus de configuration est de ne pas arriver à contacter le service contrôleur. Si tel est le cas, la première chose à vérifier est de voir si vous arrivez à pinger la machine du service contrôleur et ensuite de vérifier qu'une exception sur le port TCP 6901 est bien présente sur la machine contrôleur. Tout comme la configuration du service contrôleur, un ensemble d'actions est effectuée comme la gestion de la sécurité, le démarrage du service, l'ajout de l'agent au contrôleur et la création d'une exception au pare-feu pour le service Windows « Microsoft Visual Studio Test Agent 2010 » [Fig.9].

Si vous souhaitez avoir plus d'informations et l'historique des étapes, vous trouverez les logs à l'emplacement suivant (chemin par défaut d'une plate-forme 64bits) : C:\Program Files (x86)\Microsoft Visual Studio 10.0\Microsoft Visual Studio Test Agent

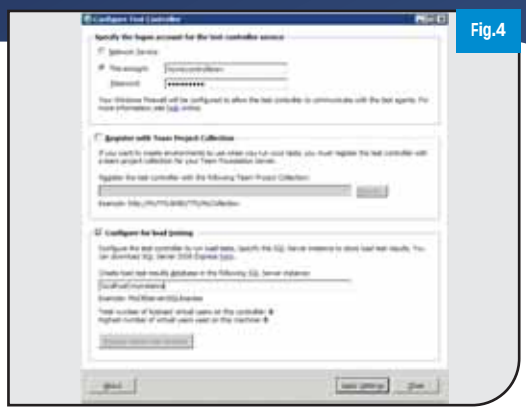


Fig.4

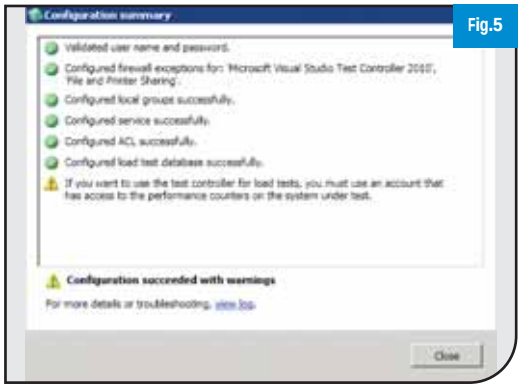


Fig.5


Fig.6

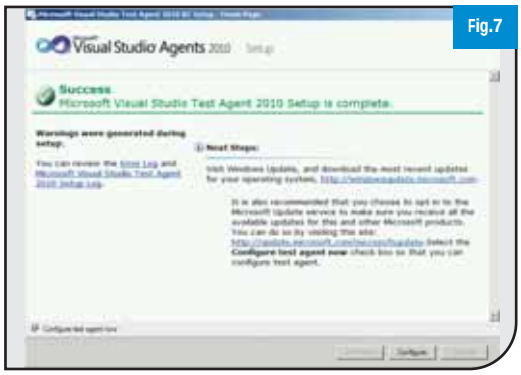


Fig.7

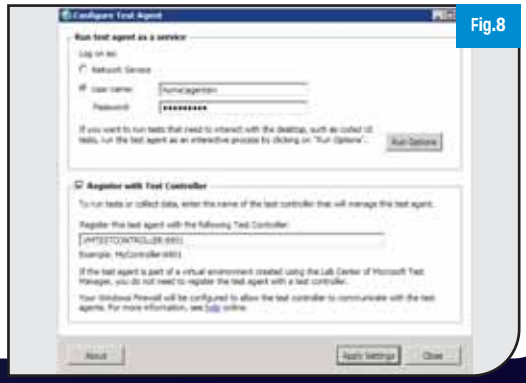


Fig.8

2010 - ENU\Logs. Une fois l'installation du contrôleur et des agents dans votre environnement effectuée, il vous reste à vérifier que cet environnement est prêt pour exécuter les tests. Pour cela, il faut démarrer une instance Visual Studio 2010 en tant qu'administrateur, cliquez sur **Test** puis **Manage Test Controllers**... [Fig.10]. Une nouvelle fenêtre va apparaître. La première étape est d'indiquer le nom de la machine où a été installé un service contrôleur. Une fois renseigné, Visual Studio va tenter de se connecter au service et récupérer les informations de configuration. Ainsi, il affichera par exemple la chaîne de connexion vers le serveur de base de données et le couple nom de machine / statut du service Windows pour chacun des agents que l'on a associé à ce contrôleur. Depuis cette fenêtre on a également la possibilité de :

- Redémarrer une machine agent en particulier
- Enlever une machine agent de l'environnement de tests nommé « rig »
- Redémarrer l'ensemble des machines contrôleur / agents de l'environnement

A noter que si vous avez des problèmes avec des services agents, pensez à vous connecter sur les machines et à vérifier l'*event viewer* de Windows pour voir pourquoi le service est stoppé, pourquoi il ne redémarre pas... [Fig.11]

Initialiser un test de charge

Afin de créer votre test de charge, vous devez commencer par créer un projet de type test dans Visual Studio 2010 [Fig.12]. Lorsque l'on crée un projet de type test, des fichiers pour la solution Visual Studio sont automatiquement ajoutés. Le fichier qui va particulièrement nous intéresser est le fichier de configuration d'un test de charge nommé par défaut Local.testsettings. En ouvrant ce fichier, vous pourrez définir la configuration générale d'un test de charge :

- Déploiement des fichiers de la solution
- Exécution en 32 ou 64 bits
- Définir les limites de timeout
- Le type navigateur pour un test de charge sur un test Web
- Exécution locale ou à distance
- ...

Avant de créer un test de charge, il faut définir un test individuel. Vous pouvez par exemple définir un test Web et on associera ensuite ce test à notre scénario de notre test de charge [Fig.13].

Lorsqu'on crée un test de type Web, nous allons avoir le navigateur par défaut (Internet Explorer ;)) qui se lance et qui nous donne la possibilité d'enregistrer les différentes actions que l'on fait, par exemple une navigation dans un site, en somme notre scénario de test. Une fois fini, Visual Studio va récupérer l'ensemble des informations de notre navigation comme les paramètres en GET / Post, d'autres informations sur le protocole HTTP.

Sur l'exemple [Fig.14], nous voyons que notre test a été d'interroger la page d'accueil de Bing (www.bing.com). En interrogeant cette page, une autre page est appelée et on voit que Visual Studio a récupéré les paramètres en GET passés au navigateur, par exemple format avec la valeur *xml* [Fig.14].

Une fois ce scénario en place, nous pouvons créer notre test de charge. L'objectif va être de démultiplier le nombre d'utilisateurs virtuels qui interrogent www.bing.com [Fig.15].

Lors de la création d'un test de charge, un assistant se lance pour le configurer. La première étape consiste à définir un nom pour le scénario et de préciser le think time. Cette notion représente en réalité le fait de simuler un temps entre les actions d'un utilisateur virtuel. Un exemple typique serait par exemple de faire une recherche sur Bing puis de cliquer sur un lien dans les résultats. Si aucun think time n'est spécifié, les actions virtuelles s'enchaîneront alors instantanément alors qu'un utilisateur réel aurait peut-être regardé 3- 4 secondes les résultats avant de cliquer sur l'un d'entre eux. Ce point est important pour avoir une simulation la plus proche possible de la réalité.

La seconde étape permet de définir le

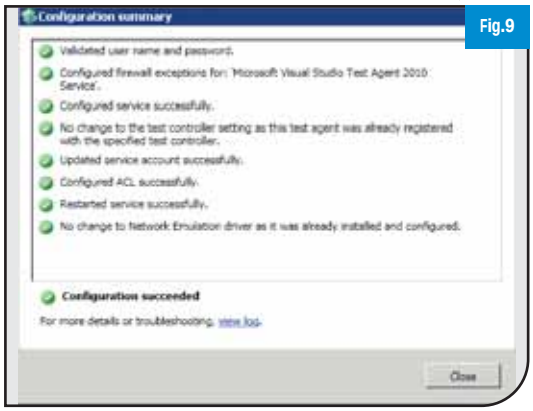


Fig.9

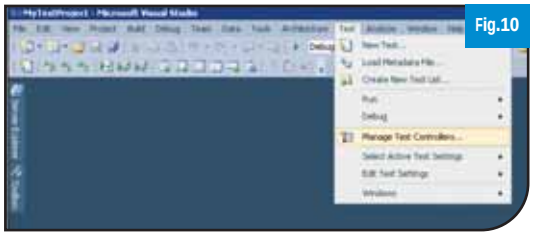


Fig.10

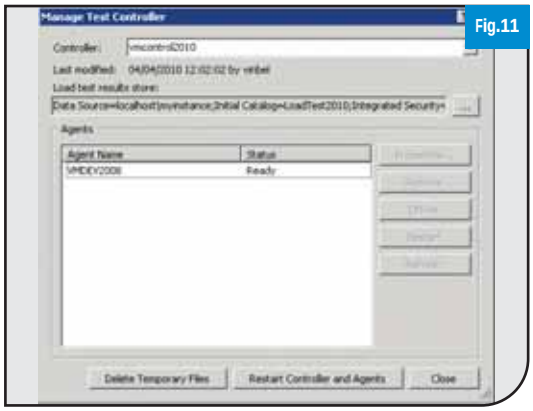


Fig.11


Fig.12

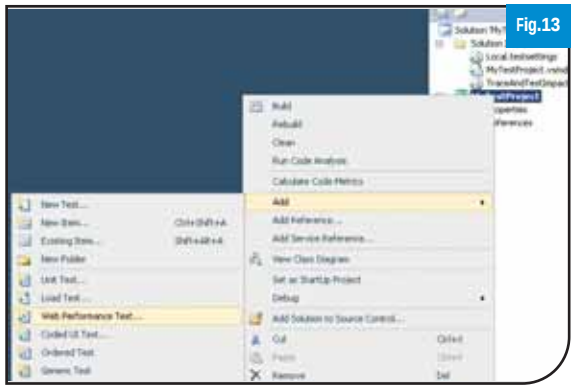


Fig.13

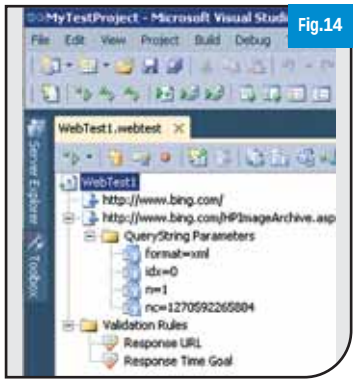


Fig.14

Les outils des Décideurs Informatiques

Vous avez besoin d'info sur des sujets d'administration, de sécurité, de progiciel, de projets ? Accédez directement à l'information ciblée.

Le numéro 13-JUIN Paraît le 5 Mai

Dossiers : Cloud Computing
Office 2010, etc…
www.solutions-logiciels.com

L'INFORMATION EN CONTINU

www.solutions-logiciels.com

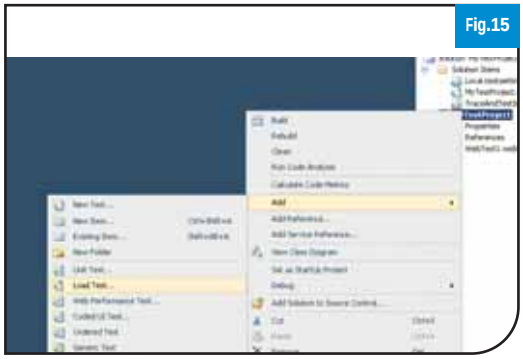


Fig.15

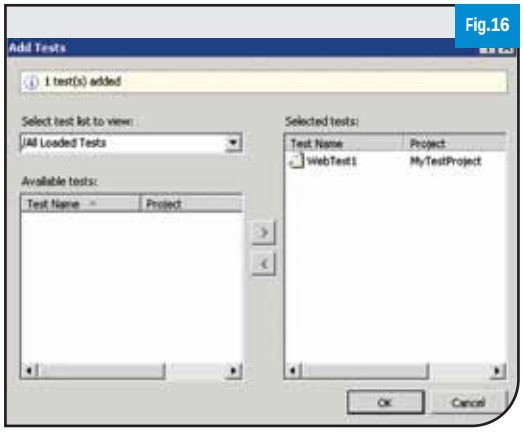


Fig.16


Fig.17

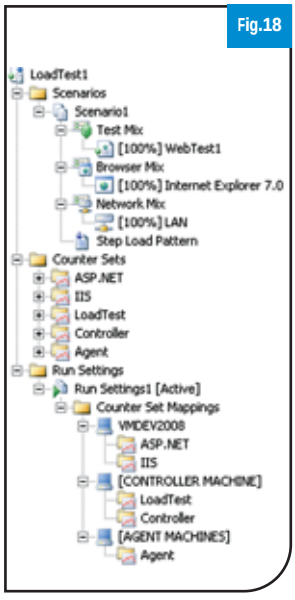


Fig.18

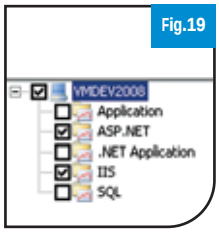


Fig.19

pattern du test. Il en existe deux :

- Une charge constante : on définit alors un nombre d'utilisateurs virtuels constant qui sera utilisé tout au long de la durée du test de charge
- Une charge par étapes : la charge va être graduée selon des étapes où le nombre d'utilisateurs virtuels va augmenter progressivement. L'intérêt de ce pattern est de pouvoir évaluer un seuil au-delà duquel une application ne tient plus la charge avec une saturation de la mémoire, du processeur ou de toute autre ressource.

Le test de charge peut être constitué de plusieurs tests que ce soit des tests unitaires, web…vous aurez ainsi la possibilité de définir quels tests sont intégrés au test de charge et dans quel ordre [Fig.16]. Les dernières étapes permettent de définir quelle est la proportion de tel ou tel navigateur Internet, quelle est la proportion de tel ou tel type de réseau (LAN, T1…). L'étape la plus importante de la configuration d'un test de charge est bien celle où on définit les machines que l'on souhaite monitorer et sur quelles métriques, c'est-à-dire quels compteurs de performance. On pourra ainsi spécifier des compteurs orientés IIS pour un serveur Web ou plus ADO.NET, SQL pour un serveur backend [Fig.17].

Par défaut, Visual Studio ajoute le contrôleur et les agents, il vous faudra ajouter manuellement chacune des machines que vous souhaitez tester. Pour chaque machine, on définira le groupe de compteurs de performance qui nous intéresse [Fig.18].

Enfin, il restera à définir la durée du test ainsi qu'une période dite de chauffe où le contrôleur simule des actions progressives afin que les machines testées soient prêtes au moment où le test débutera concrètement. Une fois la configuration terminée, vous avez une visualisation arborescente des différents paramètres que l'on peut bien entendu éditer afin d'affiner ces paramètres si nécessaire [Fig.19].


Fig.20

Une fois le test de charge lancé, nous allons pouvoir suivre en temps réel les métriques des machines de l'environnement. À notre disposition un ensemble de graphiques dynamiques sur lesquels on pourra ajouter / supprimer tel ou tel compteur de performance mais aussi des données tabulaires comme des agrégations MIN / MAX ou encore des moyennes [Fig.20]. Les informations étant stockées en base, elles sont exploitables à tout moment. On peut donc réafficher les graphiques, récupérer les données plus tard. La durée de rétention de ces données est à configurer, par défaut il conserve les vingt derniers tests. À noter également que l'on peut exporter les données d'un test dans un fichier de type archive afin de pouvoir les relire dans un autre environnement. Fonctionnalité très intéressante lorsque l'on fait un audit et que l'on n'a pas accès librement et indéfiniment à l'environnement de tests ! Enfin, Visual Studio collecte également les données par rapport à des seuils qui sont soit ceux fixés par défaut, soit des seuils que l'on a fixés nous-mêmes. Ainsi, si l'utilisation CPU dépasse par exemple 92% sur telle ou telle machine on lève un avertissement ou une erreur. Ces informations seront visibles directement sur les graphiques, ce qui nous permettra de rechercher la cause avec d'autres indicateurs, comme le nombre d'utilisateurs virtuels.

Ce n'est pas l'objet de cet article mais il faut également penser que nous avons la possibilité d'alimenter dynamiquement des données de test depuis une base. Typiquement, si on souhaite se loguer à une application avec différents utilisateurs enregistrés en base, il suffira d'indiquer où chercher cette information et le contrôleur générera les actions en prenant de manière aléatoire ces données. De la même manière, nous pouvons aller plus loin dans la mise en place des tests en programmant nous-mêmes les actions, le comportement du test.

■ Vincent **Bellet**
Spécialiste Solutions .NET / SQL Server
Microsoft France
vincent.bellet@microsoft.com

.NET 4.0 : le mot clé Dynamic

Les langages .NET, tels que le C#, VB.NET ou le C++ managé font partie de la famille des langages de type statique ou « fortement typé », tout comme l'est le langage Java. Cela signifie que le langage utilise des types qui sont identifiés à la compilation par opposition aux types variables. La conséquence de ce fonctionnement est une rigidité dans l'utilisation des types.

Par exemple, pour une fonction « void Traitement(ClassMetier m) », il est impossible de passer en paramètre autre chose qu'une classe de type « ClassMetier » ou dérivant de cette classe. Cette vérification de type s'effectue dès l'étape de compilation (ou directement dans un éditeur évolué tel que Visual Studio). À l'exécution, la CLR (Runtime d'exécution) nous protège également en vérifiant les types. Ceci est la raison des fameuses « InvalidCastException » que le programme génère lors de ces erreurs.

Pourtant, d'autres langages permettent de programmer avec des types faiblement typés. Visual Basic possède le mot-clé « variant » pour définir une variable dont le type est mutable. D'autres langages modernes comme le Ruby utilisent le principe de types dynamiques dont le contrat porté par le type de classe est évalué directement à l'exécution (nous verrons plus loin le détail de ce fonctionnement). Microsoft a compris l'intérêt de ce genre de programmation puisqu'il nous offre depuis la 3.5 une « DLR » (Dynamic Language Runtime), c'est-à-dire un moteur d'exécution pour des langages dynamiques. Cette DLR est, par exemple, embarquée dans Silverlight. Et c'est justement à travers cette DLR que nous trouvons des langages « .NET » dynamiques tels que RubyCLR et IronPython qui sont des implémentations de Ruby et de Python s'exécutant sur la runtime de Microsoft.

Le mot-clé « var » de .NET 3.5

Faisons dès à présent la distinction entre le « dynamic » de .NET 4.0 et le « var » de .NET 3.5. Pour rappel, l'utilisation du mot-clé « var » de .NET 3.5 permet de déclarer une variable sans connaître son type. À la différence d'une définition dynamique, ce n'est pas à l'exécution mais à la compilation que le compilateur va devoir déterminer le type « masqué » par cette définition. Prenons cet exemple :

```
var chaine = "Test";
string chaine2 = chaine.Trim() ;
```

Ici, le compilateur va déterminer que puisque l'on affecte une chaîne de caractères à la variable chaîne alors le type de chaîne est string. Il agit donc en remplaçant le « var chaine = … » en « string chaine = … ».

Les conséquences du mot-clé « var » font, que même si le type exact n'est pas précisé à la déclaration, l'utilisation de la variable déclarée en type var doit respecter strictement le contrat porté par le type qui sera trouvé à la compilation. Pour notre exemple précédent, appeler sur notre objet chaîne une méthode n'existant pas dans la classe string provoquerait une erreur de compilation (et non une erreur d'exécution) puisque le compilateur, déterminant le type de chaîne en string constaterait ensuite l'erreur d'appel sur une méthode inexistante. Voilà pourquoi le mot-clé « var » n'est pas de la programmation dynamique mais une déclaration tardive de type. Pour rappel, l'intérêt du var se trouve ailleurs. Ainsi les requêtes Linq offrent la possibilité de retourner des types déclarés sur l'instant (grâce au new { …}). Ces types ne possédant pas de nom propre, le mot-clé var vient ici pour éviter de les nommer explicitement.

Introspection

Pour appeler une méthode dynamiquement sur une classe, une technique classique est d'utiliser l'introspection. Dans l'exemple suivant, nous allons écrire une méthode prenant un type générique en entrée. Dans le corps de la fonction, nous allons appeler une méthode sur cet objet. Ne connaissant pas le type exact, nous ne pouvons pas écrire le code suivant :

Exemple qui ne compile pas :

```
Public void MethodeDeTest( T obj )
{   obj.MethodeMetier() ; }
```

En effet, T n'est pas connu, donc le compilateur n'accepte pas cet appel. Par l'introspection nous allons pouvoir à l'exécution aller chercher une méthode d'obj et l'appeler. Le code est technique et le voici :

```
Public void MethodeDeTest( T obj )
{
   var methodeDeObj = typeof(T).GetMethod("MethodeMetier");
   if ( methodeDeObj != null )
     methodeDeObj.Invoke(obj, new object[0]);
}
```

Ici, nous voyons que nous pouvons réaliser du « vrai » code dynamique dont le traitement est réalisé à l'exécution et non à la compilation. Malheureusement, le premier inconvénient de cette technique est la quantité de code à écrire pour réaliser l'appel. L'autre inconvénient, moins visible, est que chaque appel aux méthodes d'introspection telles que GetMethod interroge l'objet pour connaître ses caractéristiques. Ces appels ont un coût non négligeable qui se répète à chaque appel. Maintenant que nous avons posé les bases et vu les limites de ce qu'offrait la version 3.5 de .NET, intéressons-nous aux nouveautés du Framework .NET 4.0.

Dynamic

Comme nous pouvons nous en douter au vu des deux chapitres précédents, l'objectif du nouveau mot-clé dynamic est d'une part de simplifier l'écriture de code pour définir un type inconnu et d'autre part d'améliorer les performances d'appels de méthodes

en dynamique de façon à ne pas obliger le moteur d'exécution à interroger l'objet et le type à chaque appel de méthode.
Dans le cadre de notre problématique d'appel introspectif, voici le code en version .NET 4.0 :

```
public void MethodeDeTest2(T obj)
{
    dynamic o = obj;
    o.MethodeMetier();
}
```

Il est intéressant de noter qu'au moment où Visual Studio interprète notre code, il est incapable de proposer les méthodes de « o » si l'on appelle l'IntelliSense (Control-Espace ou affiché automatiquement après le point). Cela s'entend puisque o sera évalué uniquement à l'exécution (voir Fig 1).
A l'exécution, si nous positionnons un point d'arrêt, alors nous pouvons constater que le type de o est correctement affiché en « ClasseMetier » (voir Fig 2).
Au final, chaque type « vivant » en exécution dans le framework correspond à un type réel et un seul. Ici c'est uniquement le type déclaré qui est en **dynamic**.

DynamicObject

Une habitude courante en langage dynamique est d'utiliser la possibilité d'interpréter les appels de méthode à la volée. L'idée est de pouvoir intercepter les appels qui se font à une classe pour réaliser un comportement sur mesure. Pour montrer la puissance de cette fonctionnalité, nous allons réaliser une classe « tableau » qui va permettre de stocker et récupérer des valeurs sous forme d'attributs d'une classe dont la liste n'est pas connue à l'avance. Pour créer cette classe « TableauMagique », nous allons étendre une nouvelle classe du Framework .NET 4.0 « DynamicObject ». DynamicObject est une classe qui propose deux méthodes à surcharger :

- TryGetMember : Cette méthode est appelée lorsqu'un attribut est récupéré depuis la classe (monObjet.AttributExemple = xxx ;)
- TrySetMember : Cette méthode est appelée lorsqu'un attribut est mis à jour dans la classe (xxx = monObject.AttributExemple ;)

Afin de stocker les items du tableau dynamique, nous allons instancier un dictionnaire interne à notre classe. Voici donc le code de cette classe :

```
public class TableauMagique : DynamicObject
{
    // dictionnaire interne
     Dictionary<string, object> dictionary = new Dictionary
<string, object>();

    // Nb elements
    public int Count
    { get { return dictionary.Count; } }

    // Intercepteur des getters
    public override bool TryGetMember( GetMemberBinder binder,
 out object result)
    {
        string name = binder.Name.ToLower();
        return dictionary.TryGetValue(name, out result);
    }

    // Intercepteur des setters
    public override bool TrySetMember( SetMemberBinder binder,
 object value)
    {
        dictionary[binder.Name.ToLower()] = value;
        return true;
    }
}
```

Dans cette classe, nous allons pour chaque appel « TrySet » stocker la valeur dans notre dictionnaire et dans chaque appel TryGet retourner cette valeur.
Nous n'avons plus qu'à tester notre tableau avec le code suivant :

```
static void Main(string[] args)
{
  dynamic person = new TableauMagique();

  person.Prenom = "Julien";
  person.Nom = "Carnelos";

  Console.WriteLine(person.prenom + " " + person.nom);
}
```

ExpandoObject

L' « ExpandoObject » est une autre nouvelle classe du framework .NET 4 autorisant l'ajout et la suppression de comportement à la volée. Son utilisation est simple mais ses possibilités sont grandes. Pour le créer, nous instancions un ExpandoObject dans un dynamic :

```
dynamic test = new ExpandoObject();
```

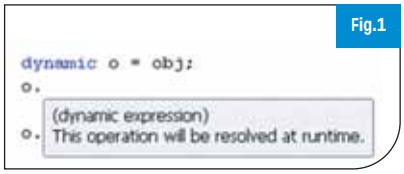
Très simplement nous pouvons lui créer un attribut puis récupérer la valeur stockée :

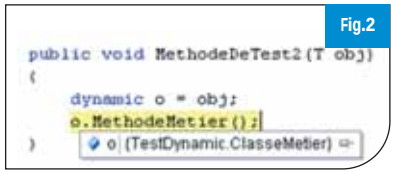
```
test.stockage = "Julien";
Console.WriteLine(test.stockage);
```

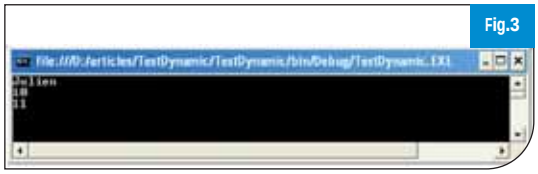
Pour aller plus loin, il est possible d'associer à notre objet test du comportement. Voici l'injection d'une méthode :

```
test.number = 10;
test.Increment = (Action)(() => { test.number++; });

Console.WriteLine(test.number);
test.Increment();
```







```
Console.WriteLine(test.number);
```

Dans le code précédent, nous avons créé un attribut « number » valant 10, puis nous avons créé une méthode « Increment » qui incrémente son attribut compteur. À l'exécution, La console affichera le comportement attendu (voir Fig 3).

COM Interop

Un scénario dans lequel la DLR / dynamic va nous apporter aussi un gain de productivité est l'interopérabilité avec les objets COM. Jusqu'à présent, l'appel à des méthodes encapsulées COM nécessite de transtyper (caster) chaque valeur à des types C#. Avec le Framework .NET 4.0, le code devient plus simple puisque nous pouvons appeler directement les méthodes de ces objets COM qui seront perçus comme des dynamic. Voici un exemple pris sur la MSDN :

```
// Avant le mot-clé dynamic.
((Excel.Range)excelApp.Cells[1, 1]).Value2 = "Name";
Excel.Range range2008 = (Excel.Range)excelApp.Cells[1, 1];

// avec dynamic, l'accès a Value et la conversion en Excel.Rang
```

Fig.4

```
 s'effectue
// à l'exécution par le COM binder version 4.0.
excelApp.Cells[1, 1].Value = "Name";
Excel.Range range2010 = excelApp.Cells[1, 1];
```

Mise en cache

Pour terminer, un point intéressant dans l'utilisation du dynamic est la mise en cache par le Framework des appels « introspectés ». Cela signifie que l'appel d'une méthode sur un objet dynamique sera évalué puis mis en cache. Ainsi lors du rappel de cette méthode, le Framework connaîtra son existence et pourra directement l'appeler avec un « overhead » (temps supplémentaire) minimal d'indirection vers l'appel réel de la méthode.

Pour notre premier exemple d'appel dynamic, si nous regardons l'IL généré grâce à Reflector (http://www.red-gate.com/products/reflector/) sur la fig 4, nous constatons que le compilateur met en cache l'appel dans le « if ».

Conclusion

Avec ce tour d'horizon, nous voyons que l'ajout de dynamic ouvre de nouvelles perspectives innovantes dans notre approche du code. Les possibilités offertes sont nombreuses et vont permettre de bâtir de nouvelles API et applications plus flexibles, plus productives… et plus dynamiques !

■ Julien **Carnelos**
SQLi Consulting Bordeaux

Pex : générez vos tests unitaires !

Les tests unitaires sont considérés comme une bonne pratique et permettent de valider la non-régression tout au long du cycle de développement ou de maintenance. Utilisés avec la couverture de code, les tests unitaires sont un outil extrêmement puissant pour la mesure de la qualité du code produit. Malgré tous les avantages qu'ils offrent et la littérature à leur sujet, peu de développeurs les implémentent : trop coûteux en temps et pas suffisamment exhaustifs. Avec Pex, nous vous proposons d'améliorer rapidement et facilement la qualité de vos développements et de générer vos tests unitaires sans effort !

Les tests unitaires, une vérification par exécution d'une partie déterminée d'un logiciel (appelée "unité" ou "module"), sont à présent des procédés obligatoires dans le cadre d'un projet de développement informatique. Ce sont les développeurs qui mettent en œuvre ces tests afin de vérifier si un module particulier répond aux spécifications et fonctionne correctement, indépendamment du reste du code. Il est possible d'utiliser le framework de test MSTest fourni avec la suite Visual Studio, ou d'autres comme NUnit ou MbUnit. Ces outils structurent l'écriture et l'exécution de tests unitaires. Mais des problèmes restent à résoudre, notamment en termes de création des tests. Deux problèmes majeurs des tests unitaires sont l'explosion combinatoire des valeurs d'appels des méthodes et la gestion des multiples chemins que peut prendre une exécution. Les tests unitaires doivent pouvoir emprunter un maximum de chemins possibles, et idéalement tous les chemins. C'est ici que nous parlons de la couverture du code, un ratio de chemins couverts sur l'ensemble des chemins possibles. Pour résoudre ces problèmes, Microsoft Research propose Pex (Program EXploration), un outil de génération des tests unitaires assurant une bonne qualité et une juste quantité de ces tests. Pex est un assistant qui s'intègre à Visual Studio. En explorant le code source .Net des méthodes, cet outil découvre les cas de vérification les plus critiques : c'est une approche de type « boîte blanche », contrairement à l'approche « boîte noire » qui se base sur les valeurs des appels de méthodes et les résultats de sortie. À partir d'un test unitaire paramétré, le moteur Pex génère automatiquement une suite de tests unitaires avec un taux de couverture de code élevé.

Découverte de Pex

Pour installer Pex, il vous sera nécessaire de télécharger la dernière version en ligne sur http://research. microsoft.com/en-us/projects/Pex/. Suivant la version de Visual Studio que vous utilisez, vous pourrez télécharger une version commerciale accessible aux abonnés MSDN (pour VS Team Suite ou Team Dev ou Team Test) ou version académique qui pourra s'installer sur Visual Studio à partir de la version Pro ou être simplement utilisé en ligne de commandes. Une fois installé, exécutons Visual Studio et découvrons Pex. Pour les besoins de cette découverte nous utiliserons un petit programme Console permettant d'afficher un message. Le code est volontairement simple pour les besoins de l'article afin de montrer Pex dans son usage. Vous pourrez complexifier le code à volonté afin de tester les différentes capacités de Pex.

Pour utiliser Pex depuis le code et tester une méthode en particulier, il est possible de cliquer avec le bouton droit de la souris sur la définition d'une méthode, puis « Run Pex » pour tester cette méthode. [Fig.1]

L'exécution du test entraîne au préalable la compilation du projet puis l'exécution des tests générés en mémoire et basés sur l'exploration du code.

L'explorateur Pex

L'explorateur Pex affiche la liste des tests exécutés par Pex ainsi que leur statut. [Fig.2]. Dans la barre d'icône de l'explorateur et par ordre d'affichage de gauche à droite, les commandes sont :

- Affichage des tests exécutés,
- Affichage par erreur,
- Contracts : Les contrats sur les valeurs en entrée et en sortie des méthodes,
- Events : Informations sur l'exploration ou éventuellement les limitations rencontrées,
- Afficher tout ou seulement les erreurs.

L'affichage par erreur permet d'obtenir des commandes supplémentaires : [Fig.3]. En sélectionnant l'erreur surlignée en bleu, les commandes qui apparaissent permettent de :

- Se déplacer directement sur la ligne de code qui produit l'erreur (le bouton vert avec la flèche) ;
- Générer les tests unitaires paramétrés (voir[Fig.4]) ;
- Appliquer les correctifs à votre code ;
- Traiter l'erreur comme une exception autorisée.


Fig.1

Un code exemple et le menu contextuel de Pex.

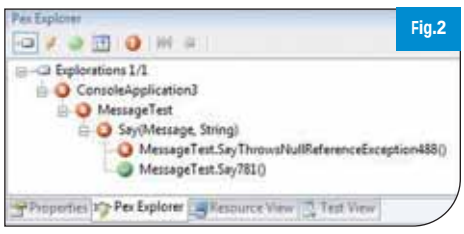


Fig.2

L'explorateur Pex.

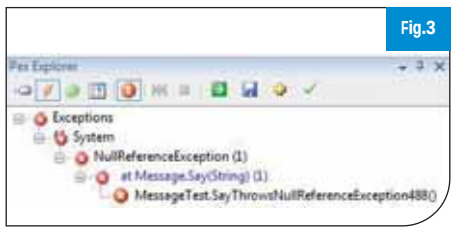


Fig.3

Affichage des erreurs.

La fenêtre des résultats

La fenêtre des résultats affiche tous les tests exécutés par Pex, les méthodes, les paramètres utilisés, les erreurs, la sortie de l'exécution… [Fig.5]. La partie « Details » permet de connaître le test qui a généré l'erreur ainsi que la pile d'appels. [Fig.6]

Nous retrouvons les raccourcis aux commandes de l'explorateur Pex mais aussi d'autres fonctions, comme l'envoi de l'erreur par mail à un collègue ou son assignation via la création d'un work item TFS, l'exécution de tests sur la classe ou l'assembly, ou encore l'affichage de la couverture de code.

La génération d'un projet de tests unitaires

La première sauvegarde des tests effectués permet de générer un projet de tests unitaires. [Fig.7]

Les tests ainsi générés sont réutilisables, modifiables et intégrables à la couverture de code de Visual Studio ou encore au processus de compilation automatisé couplé à l'intégration continue gérée par Team Foundation Server. Par ailleurs, les tests générés sont dits tests unitaires paramétrés. Si le développeur souhaite modifier les tests unitaires, il lui est recommandé de modifier le fichier .cs qui couvre les tests unitaires de bas niveau (fichier .g.cs). Les tests sont ainsi visibles depuis la fenêtre d'exploration des tests unitaires Visual Studio. [Fig.8]

Fonctionnement de Pex

Pour chaque expression du code, Pex essaye de proposer des valeurs d'entrée pour atteindre l'expression en question, en tenant compte des branchements conditionnels et des opérations qui peuvent lever une exception. Cet outil fonctionne en boucle avec retour de l'information et le code est exécuté plusieurs fois avec une analyse qui influence les itérations suivantes et qui éventuellement peut découvrir de nouveaux cas de test.

Par définition les tests unitaires représentent des fonctions sans paramètres. Afin de variabiliser ces tests, nous pouvons ajouter des paramètres, ce qui nous donne des tests unitaires paramétrés (PUT). Un test de ce type peut être décomposé en 4 parties :

- *Paramètres* : valeurs d'entrée qui sont passées à une fonction à tester.
- *Hypothèses* : des suppositions sur les paramètres qui aident à construire des entrées de test valides.
- *Séquence de méthodes* : caractérise le scénario de test.
- *Assertions* : la notion clé d'un test unitaire.

Le fonctionnement de PEX est basé sur ce type de tests, ce qui facilite la génération et l'évaluation des valeurs d'entrée. Un PUT permet la séparation des responsabilités. Premièrement, nous définissons le comportement externe de notre fonction par un test unitaire paramétré, ce travail n'étant réalisable que par un humain. Ensuite, un outil comme Pex peut automatiquement construire un ensemble de tests avec une bonne couverture du code. Ci-dessous, un exemple de test paramétré qui fait ressortir les quatre parties mentionnées.

```
public void AddSpec2(
    //paramètres
    ArrayList list, object element){
    //hypothèses
    PexAssume.IsTrue(list != null) ;
    //séquence de méthodes
    int len = list.Count ;
    list.Add(element) ;
    //assertions
    PexAssert.IsTrue(list[len] == element) ;
}
```

Ce PUT est plus général qu'un test unitaire traditionnel et il reste à construire l'ensemble des entrées adéquates qui va assurer une couverture élevée du code.

Afin de construire l'ensemble mentionné, il faut faire un bon choix des valeurs d'entrées. Dans les tests traditionnels ce choix est manuel et il est difficile de garantir la pertinence de ces choix. Avec Pex, cette étape est automatisée. Pex utilise une technique de test par exploration qui est connue sous le nom d'exécution par symbole dynamique (dynamic symbol execution). L'outil exécute le code plusieurs fois et apprend le comportement du programme en analysant le flux de contrôle et de données. Suite à chaque exécution, l'outil essaye d'emprunter un nouveau chemin d'exécution qu'il juge important en générant des nouvelles valeurs d'entrée. De cette façon, ce test par exploration automatisé permet d'aller de plus en plus loin dans l'implémentation.

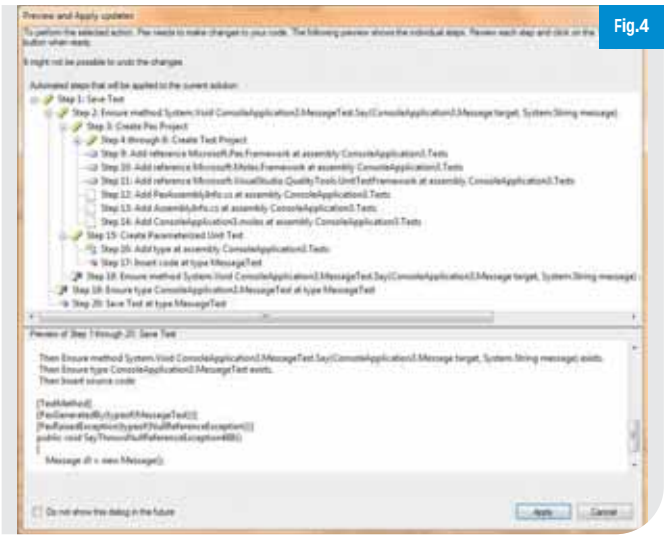


Fig.4

Génération des tests unitaires paramétrés.


Fig.5

Fenêtre des résultats.

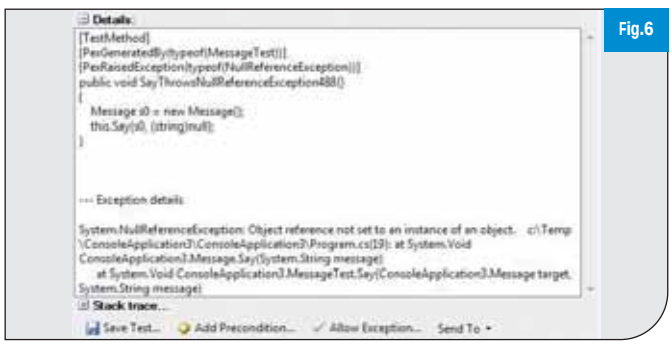


Fig.6

Détails par résultat.

Génération de rapports de tests

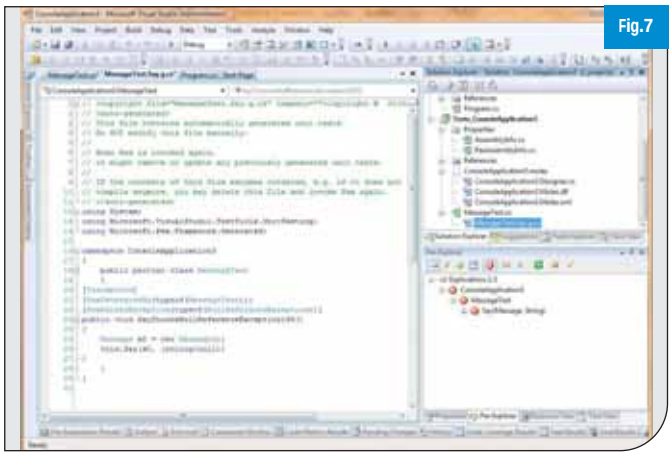
Pour générer des rapports de tests, il faut au préalable modifier la configuration de Pex (Visual Studio > Menu Tools > Options, puis dans l'onglet Pex, choisir General, Reports et mettre la valeur à True pour Reports). Il est alors possible d'obtenir le rapport des tests via la fenêtre des résultats (Views > Open Report). Pour chaque test, il est alors possible d'obtenir la couverture de code, les paramètres et les détails de l'exécution [Fig.9]. Par ailleurs, l'ensemble des données de rapport sont stockées à chaque exécution des tests dans le répertoire de sortie de la compilation du projet dans un dossier .PEX horodaté. Ce dossier contient le fichier .PER qu'il est possible d'importer depuis un autre Visual Studio pour avoir les résultats générés depuis une autre machine.

Limites

Il existe des situations où Pex ne peut pas analyser proprement le code. Il s'agit notamment de :

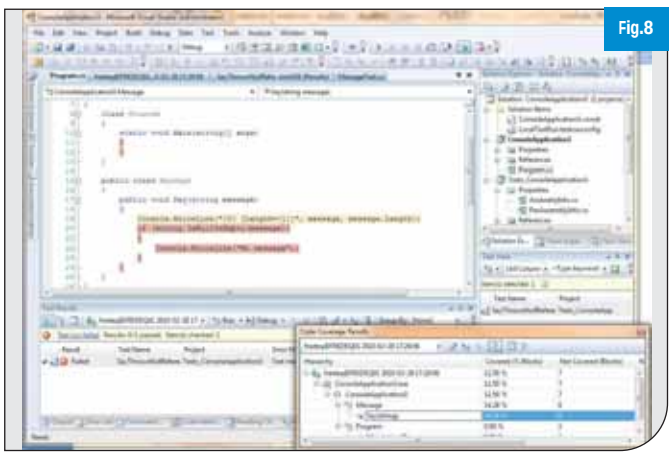
- **Non-déterminisme**. Pex suppose que le programme à analyser est déterministe, sinon l'outil va boucler et va s'arrêter après avoir atteint la limite de l'exploration.
- **Parallélisme**. L'exigence d'un programme déterministe rend le Pex non compatible avec les programmes multithreads.
- **Code natif**. On ne peut pas lancer l'exploration Pex sur du code natif ou celui qui utilise le mécanisme P / Invoke.
- **Langage**. En principe, il est possible d'analyser des programmes écrits dans n'importe quel langage .Net, néanmoins l'extension pour Visual Studio ne génère que du code C#.

Fig.7



Projet de tests unitaires générés.

Fig.8



Les tests de Visual Studio .NET.

Conclusion

Pex et Moles sont désormais des Powertools pour Visual Studio 2010 qui étendent les tests unitaires en .Net. Pex génère automatiquement des ensembles de tests avec une couverture du code élevée. Directement à partir de Visual Studio, Pex trouve les valeurs d'entrée les plus intéressantes qui peuvent être sauvegardées en tant que tests unitaires. Le générateur de tests unitaires Pex est activement développé par l'équipe de Microsoft Research et a déjà été adopté par certaines sociétés dans le cadre de leurs projets de développement.

Glossaire

• **Code Contract** : framework permettant d'implémenter des hypothèses de code de différentes formes : pré-conditions, post-conditions et objets invariants. Ce framework offre la capacité dans tous langages .NET de bénéficier de l'approche « conception par contrat ». Les hypothèses sont utilisées par Pex pour guider l'exploration de code et peuvent être utilisées pour générer la documentation.

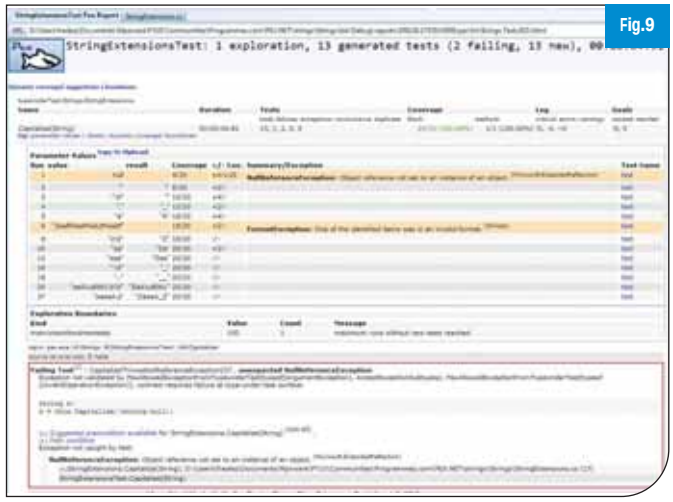
• **Moles** : Moles est une boîte à outils, fournie avec Pex, permettant de détourner le comportement par défaut d'une méthode. Moles est connu sous le nom de « mocking framework » et avec lequel il est possible de remplacer toute méthode .Net par un delegate.

• **PEX Extensions** : Pex est fourni avec le support natif du framework de tests de Visual Studio. Il est possible d'éteindre Pex en lui ajoutant par exemple le support d'un autre framework de tests ou d'un générateur de rapport personnalisé.

■ Frédéric **Queudret**
MVP Client Application & CTO de la société Mpoware, accélérateur d'innovation. Société française d'édition de logiciels et de prestations de services sur les technologies .NET.
http://www.mpoware.com/

■ Vitalie **Schiopu**
Microsoft Student Partner
Ecole d'Ingénieur Polytech'Tours - Informatique
www.dotnetclub.info
www.vitalieschiopu.info

Fig.9



Un rapport PEX.

Création d'une application mobile à succès

Avant lancement de votre application, demandez-vous dans quelle mesure votre création donnera envie à des utilisateurs de donner une chance à votre application et dans quelle mesure votre application touchera un public assez large.

En octobre 2009, après plusieurs mois de développements, Magma Mobile (dont je suis le co-fondateur) a lancé Pinball, un jeu permettant à partir des touches de son téléphone ou de son écran tactile de jouer au flipper. En développant un jeu universellement connu et reconnu, nous avons réussi à toucher une population très large. Aujourd'hui, le nombre de téléchargements de Pinball dépasse les 2 millions.

Pré-Lancement : Tester les fonctionnalités et les modèles

Le marché Android met en valeur les nouvelles applications lors de leur sortie. Il est donc primordial de réussir sa sortie en obtenant à la fois un nombre de désinstallation faible mais aussi de bonnes premières notes. Vous devrez donc avant le lancement partager votre .apk avec vos amis et connaissances du monde Android pour qu'ils vous donnent un vrai feedback sur les fonctionnalités de votre application. Notez également qu'Android est un marché dans lequel coexistent plusieurs téléphones dont les résolutions et aspects ergonomiques changent. Assurez-vous donc que vos applications tournent bien à la fois dans les différents émulateurs mais aussi sur les téléphones physiques. C'est une approche que nous avons notamment mise en place avec l'application d'intelligence collective Market Suggest. Ainsi, afin de s'assurer de son adoption future par de nombreux utilisateurs, nous avons envoyé un .apk à un groupe de bêta testeurs afin d'accumuler assez de feedback pour s'assurer une bonne expérience utilisateur dès la sortie de l'application.

Lancement : Early Adopters et Marketing

Avant même de vous engager dans une campagne marketing de votre application, essayez de bien évaluer l'impact de celle-ci auprès de vos premiers utilisateurs. Non seulement le feedback de ces premiers utilisateurs est primordial pour effectuer certains ajustements nécessaires, leurs notes et commentaires auront un impact fort sur votre classement dans le market Android. Si cette première phase est bien gérée, vous verrez rapidement votre classement augmenter et les téléchargements suivront. En moyenne, le ratio notes / nombre de téléchargements oscille entre 1% et 10%. Nos études ont d'ailleurs tendance à démontrer que le taux tend vers 1% quand les téléchargements augmentent. Dans les premiers jours du lancement, il est primordial d'avoir le taux le plus fort possible afin de bien remonter dans les classements. N'hésitez donc pas à encourager vos utilisateurs à ajouter un commentaire et une note à votre application. Cela peut se faire de manière directe dans l'application avec une Message Box, ou de manière indirecte en offrant aux utilisateurs la possibilité de vous envoyer un mail à partir de l'application.

Nous avons récemment lancé l'application PodKast. Cette application permet de retrouver tous les podcasts audio et vidéo sur le marché français. En ajoutant la possibilité aux early adopters de nous contacter directement, nous avons pu ajouter l'ensemble des podcasts ou fonctionnalités souhaitées. Notre réactivité a été récompensée par des utilisateurs heureux par la suite de nous mettre une bonne note dans le market Android.

Lancement : Promotion Organique et Payante

L'application fonctionne bien sur tous les terminaux. Votre taux de désinstallation est faible. La valeur ajoutée est donc au rendez-vous. Vous vous êtes assurés de prendre soin de vos premiers utilisateurs. Vous souhaitez donc désormais aller à une vitesse supérieure et voir vos téléchargements décupler. Il est temps de promouvoir votre application. Comme sur le web, il existe la promotion organique (gratuite) et payante. Les deux sont complémentaires. Au niveau de la promotion gratuite, il est nécessaire de faire connaître vos applications auprès des différents sites et blogs dédiés à Android. L'échange de bons procédés (vous leur donnez de la matière) devrait vous permettre d'obtenir un article si votre application offre une vraie valeur ajoutée. Essayez dans cette promotion organique d'utiliser au maximum les divers outils de promotion tels que Twitter, Facebook ou YouTube.

Toutes les nouvelles applications que nous créons s'accompagnent d'une vidéo YouTube pour montrer la valeur ajoutée de notre application sans forcer vos interlocuteurs à télécharger l'application pour se faire une idée de vos talents.

Concernant la mise en valeur organique sur le mobile, nous vous encourageons à vous assurer que l'ensemble des applications et jeux que vous développez font référence les unes aux autres. Ainsi vous offrez à vos utilisateurs actuels de découvrir votre travail. Au niveau de la promo-

tion payante, il faut aussi ici distinguer la promotion mobile et web. De nombreux réseaux d'annonceurs mobiles (surtout aux Etats-Unis) se développent. Ils vous permettront de toucher directement les utilisateurs de la plate-forme mobile que vous souhaitez toucher. Au niveau du web, vous pouvez utiliser des moyens publicitaires traditionnels au CPC / CPM sur sites spécialisés. Vous pouvez à ce titre essayer de contacter les divers annuaires iPhone ou Android sur lesquels vous pourrez mettre en valeur votre application auprès d'une population très ciblée.

Pérennisation de votre travail : Mise à jour de vos applications

Maintenant que vous avez plusieurs milliers d'utilisateurs, il va vous falloir pérenniser votre succès et vous assurer que votre application continue à être utilisée. La mise à jour de votre application est donc fondamentale. Chaque mise à jour doit s'accompagner d'une valeur ajoutée notable. Une mise à jour dont le seul objectif serait de remonter dans les classements aura tendance à irriter vos utilisateurs. Régler les bugs (oui c'est inévitable) est donc primordial mais aussi s'assurer d'apporter des fonctionnalités additionnelles pour augmenter le taux d'utilisation. À titre d'exemple, nous avons créé un jeu en 3D : Mahjong 3D dans lequel nous essayons d'apporter des nouvelles tables de jeu une fois par semaine afin d'attirer de nouveaux utilisateurs et d'offrir de nouvelles fonctionnalités à nos utilisateurs actuels.

Monétisation de vos Applications

Le choix de créer une application payante peut être motivé par différents facteurs. L'un de ces facteurs est intimement lié à la valeur ajoutée et l'exclusivité du contenu de votre application. Ainsi, la création d'une application de fonds d'écran a peu de chance de voir un grand nombre de téléchargements, de nombreuses applications gratuites existant déjà. À l'inverse, si vous créez une application médicale à partir de données propriétaires, la valeur ajoutée de votre contenu est telle que vous pouvez rendre votre application payante. De plus dans le cadre de cette application médicale, vous aurez tendance à toucher une population en mesure de payer un prix assez élevé pour pouvoir avoir accès à une telle valeur ajoutée. Non seulement la population au niveau démographique est importante à considérer, mais il est aussi important d'évaluer le type de distribution géographique que vous souhaitez avoir. En effet, une minorité de pays ont accès aux marchés payants, ce qui restreint énormément le nombre de personnes que vous pourriez toucher. Enfin, il me paraît important d'adapter le choix du payant en fonction de la plate-forme sur laquelle vous développez. À ce jour, le parcours d'achat sur Android est moins simple que sur l'App Store d'Apple, demandant notamment à l'utilisateur de créer un compte checkout au moment du paiement de la première application payante quand l'App Store facilite ce process dès la création d'un compte iTunes. Ce facteur évoluera dans le temps avec la création du Billing Operator permettant ainsi un achat rapide et simple en passant par l'opérateur, qui sera d'ailleurs très heureux de récupérer une commission sur chaque vente d'application.

Concernant le gratuit, vous serez amenés à trouver diverses solutions publicitaires.

Pour vos premières applications, il est bon d'intégrer un réseau d'annonces tel qu'AdMob dont la couverture devient de plus en plus forte. Si votre trafic s'étend à l'international, vous pourrez commencer à essayer des réseaux Image de type Quattro ou Millenial. Google a également une offre mais celle-ci est actuellement en Bêta parmi des développeurs ayant déjà fait leurs preuves. Alors que ces réseaux d'annonces vous offriront une variété et un volume d'annonceurs, les régies mobiles professionnelles peuvent vous offrir d'intégrer des annonceurs à CPM élevés. Un tel partenariat avec une régie professionnelle ne pourra se faire cependant que lorsque vous aurez démontré un volume de plusieurs dizaines voir centaines de milliers d'impressions par jour.

À terme, vous aurez tendance à vouloir mettre l'ensemble de ces réseaux en concurrence en utilisant un serveur d'annonces. Une nouvelle start-up française s'illustre d'ailleurs dans le domaine avec son offre : Swelen Mobile Manager. Enfin, pour les applications et jeux à très forts volumes, vous pouvez envisager du co-branding avec un annonceur souhaitant associer de manière durable sa marque au contenu créé.

Concernant la monétisation, il vous faudra tout d'abord prendre une décision sur le prix. Une fois cette décision effectuée, nous vous encourageons à l'optimiser que ce soit avec une étude de l'élasticité du côté payant ou du côté gratuit avec une optimisation publicitaire.

En conclusion, une application à succès implique une vraie réflexion des besoins des utilisateurs mobiles. Une fois vos développements terminés, assurez-vous d'effectuer tous les tests nécessaires pour que votre application ou jeu fonctionne sur l'ensemble des téléphones de la plate-forme mobile sur laquelle vous travaillez. Une fois le lancement effectué, assurez-vous de bien prendre en compte les premiers feedbacks de vos premiers utilisateurs. Ils vous aideront à convaincre l'industrie de parler de votre application et permettront de convertir un nombre d'utilisateurs additionnels dans vos campagnes marketing payantes. Vous entrerez ensuite dans une phase de pérennisation dans laquelle les diverses mises à jour vous permettront de poursuivre votre croissance.

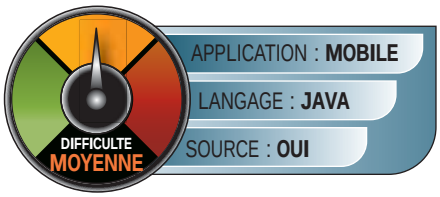
Enfin, si vous souhaitez monétiser le fruit de votre travail, vous serez amenés à choisir de rendre vos applications payantes ou de les rendre gratuites, vous rémunérant par l'intermédiaire de divers produits publicitaires.

■ Etienne **Jambou**
Co-fondateur de Magma Mobile

Android : Création d'interfaces graphiques

2e partie

Voici donc le second volet de la série d'articles consacrés à Android. Rappelons que l'objectif est de couvrir les plus grands thèmes de la programmation Android depuis l'installation du kit de développement (article du mois dernier) jusqu'à la réalisation d'applications multimédia, géolocalisées…



La création d'interfaces graphiques est un passage incontournable tout simplement parce que c'est au premier coup d'œil que l'utilisateur va juger votre application, en quelques minutes. Votre interface doit être à la fois adaptée à l'appareil, simple d'utilisation mais également agréable visuellement ! Nous avons un programme chargé : présentation du concept central des vues, introduction aux mises en pages (layout), gestion des actions utilisateur, navigation entre écrans grâce aux intentions, création de menus et enfin internationalisation.

CONCEPTS GÉNÉRAUX

La classe centrale pour la réalisation d'interfaces graphiques est la classe View. Chaque partie de votre interface est une vue, ou un groupe de vues, héritiers de cette classe.

Les vues

Quand on regarde plus précisément la javadoc concernant cette classe View, on peut lire qu'elle est la classe d'interface utilisateur basique qui gère à la fois la disposition écran et l'interaction utilisateur. En effet, une activité Android est composée de vues (et de groupes de vues). Une vue est en somme un widget qui a une apparence à l'écran : label, boîtes de textes, boutons, …

Une ou plusieurs vues peuvent être rassemblées dans un groupe de vues (ViewGroup) ce qui permet de les ordonner : [Fig.1]

Ces groupes de vues (dont le LinearLayout est l'exemple le plus classique) sont utilisés pour gérer la mise en page des composants graphiques… ce que nous n'allons pas tarder à voir !

Un terminal android

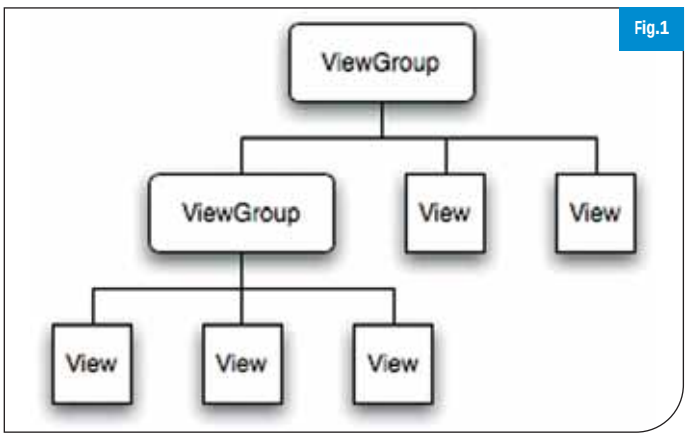


Fig.1

Hiérarchie des vues extraite de la documentation Android.

Présentation de quelques vues

Voici quelques unes des vues qui vous permettront de réaliser des interfaces Android :

- Button : un bouton;
- TextView : du texte;
- EditText : une zone de texte éditable;
- CheckBox : une case à cocher;
- RadioButton : les boutons radio;
- DatePicker : selection de dates;
- …

Les mises en pages

La notion de mise en page (ou de layout) correspond à l'agencement d'éléments d'interface graphique (en fait, de vues qui peuvent représenter des

boutons, des champs texte, des images, ...). Comme nous l'avons vu plus haut, toutes ces mises en page héritent de ViewGroup. En voici quelques unes avec une courte description de leur utilisation :

- LinearLayout : ce layout arrange les différentes vues sur une colonne ou sur une ligne;
- TableLayout : groupe les vues en lignes / colonnes découpant ainsi l'écran en cellules, comme une table HTML par exemple.
- RelativeLayout : positionne les vues les unes par rapport aux autres;
- AbsoluteLayout : permet de donner une position exacte à tous les éléments. Attention à son utilisation sur des variété d'écrans différents en termes de taille et de résolution.

La déclaration des mises en pages peut - comme la plupart des ressources - se faire de deux façons : grâce à du code ou via des éléments XML. Dans cet article, nous nous concentrons sur la déclaration XML qui est recommandée pour séparer clairement la partie présentation de votre application, du code qui lui donne son comportement. Il faut tout d'abord écrire un fichier XML qui contiendra la disposition de l'écran souhaité. Voici un squelette à titre d'exemple :

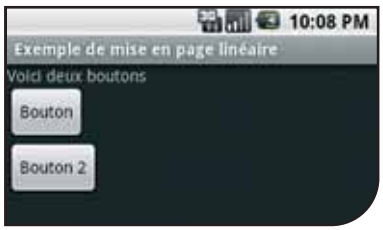
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout  xmlns:android="http://schemas.android.com/apk/res
/android">
   <!-- Ici seront placés les éléments mis en page par le Linear
Layout-->
</LinearLayout>
```

Ce fichier se place dans les ressources, plus précisément dans le répertoire "res / layout".
Appelons-le "ecran_principal.xml". Ensuite, il faut affecter cette source à l'activité, dans sa méthode onCreate(). Cette opération est réalisée grâce à la méthode setContentView() qui prend en paramètre un identifiant de ressource obtenu grâce à la classe R que nous avons découvert lors du premier article :

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.ecran_principal);
}
```

Voici un exemple de mise en page linéaire avec un texte et deux boutons :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:layout_width="fill_parent" android:layout_height=
"fill_parent"
   xmlns:android="http://schemas.android.com/apk/res/android"
    >
    <TextView   android:text="Voici deux boutons "
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
    <Button  android:text="Bouton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
    <Button  android:text="Bouton 2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
</LinearLayout>
```



Rendu de l'exemple de mise en page.

RÉAGIR AUX ACTIONS UTILISATEUR

Gestion des événements

Une des premières choses attendues est de réagir à des actions sur les différents contrôles proposés. Par exemple, comment déclencher une action sur un clic bouton ? La classe View dispose d'une méthode setOnClickListener qui permet de transmettre une instance de type OnClickListener.
Tout objet qui implémente cette interface - et donc donne corps aux méthodes décrites - peut s'enregistrer auprès d'une vue pour être notifié lors d'une action de type clic sur le composant concerné. De façon analogue, il existe les interfaces OnKeyListener, OnTouchListener, OnLongClickListener, ... selon votre besoin sélectionnez celle qui convient. Voici un exemple avec l'interface OnClickListener présentée plus haut :

```
protected void onCreate(Bundle savedValues) {

    // Création d'une implémentation d'un écouteur personnalisé
    OnClickListener ecouteurClique = new OnClickListener() {
        public void onClick(View v) {
            // Réaliser une action
        }
    };

    // Récupération du bouton déclaré dans la mise en page XML
    Button bouton = (Button)findViewById(R.id.boutonAPresser);

    // Enregistrement de l'écouteur auprès du bouton
    bouton.setOnClickListener(ecouteurClique);
}
```

Pour les traitements longs, pensez qu'il faut rendre le contrôle à l'utilisateur ou au minimum l'informer si un traitement se déroule et nécessite qu'il patiente. Pensez à utiliser les capacités de multi-threading pour rendre vos applications fluides et agréables.

Naviguer vers d'autres écrans

Dans l'article du mois dernier, nous avons introduit la notion d'intention. C'est cette notion qui va nous permettre de naviguer à la fois entre écrans de l'application et écrans système comme l'exemple de numéroteur que nous avions construit.

```
Button boutonChangerActivite = (Button)findViewById(R.id.bChanger
Activite);
boutonChangerActivite.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        Intent intent = new Intent(ActivitePrincipale.this,
 ActiviteSuivante.class);
```

```
        startActivity(intent);
      }
});
```

Attention, pour que tout ceci fonctionne, n'oubliez pas de déclarer vos activités dans le fichier de configuration "AndroidManifest.xml " :

```
<activity android:name=".ActiviteSuivante" />
```

LES MENUS

Il existe trois types de menus disponibles grâce au framework Android :

- les menus accessibles via l'appui sur la touche "menu";
- les menu contextuels, obtenus par un appui long sur une vue;
- les sous-menus proposent des items hiérarchiquement inférieurs, accessibles à partir des deux autres types de menus.

Dans cet article, nous allons nous focaliser sur le premier type de menu. La logique inhérente à l'ensemble des ressources Android est respectée : vous pouvez créer des menus de manière programmatique (au cours de l'exécution) ou alors les définir dans des ressources XML. C'est encore cette dernière approche que nous allons aborder. Voici un exemple que nous appellerons " exemple_menu. xml". Ce fichier est à déposer dans les répertoires" res / menu".

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/ajouter"  android:title="Ajouter une
 photo" />
   <item android:id="@+id/supprimer" android:title="Supprimer
une photo" />
    <item android:id="@+id/quitter" android:title="Quitter" />
</menu>
```

Ensuite, il faut "décompresser" la ressource dans le corps de la méthode onContentChanged() redéfinie dans votre activité (on voit le terme "inflater" apparaître car les ressources sont compressées grâce à l'algorithme deflate) :

```
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater menuInflater = getMenuInflater();
    menuInflater.inflate(R.menu.exemple_menu, menu);
    return true;
}
```

Pour réagir à la sélection d'un de vos menus il ne vous reste qu'à redéfinir la méthode onOptionsItemSelected().

PERSONNALISATION EN FONCTION DE LA LANGUE OU DE L'APPAREIL

Regroupées sous le nom de ressources alternatives dans la documentation Android, les ressources sont là encore au cœur des mécanismes de différenciation des interfaces en fonction des langues ou du matériel. Android s'occupera de trouver les fichiers les plus adaptés en fonction de la langue courante du système et du matériel disponible.

Internationalisation

L'exemple des chaînes de caractères est typique : c'est la première chose à traduire pour rendre son application accessible à des utilisateurs parlant différentes langues. Mais tout d'abord, revenons rapidement sur les ressources.

Dans notre exemple, nous avons figé le texte des boutons dans le fichier de mise en page.

C'est une méthode rapide utilisée pour écourter l'exemple, cependant pour construire correctement une application, vous devez utiliser des chaînes regroupées dans des fichiers de ressource.

Ces chaînes sont placées dans un fichier, dans le répertoire "res / values". Par convention, le fichier est le plus souvent appelé "strings.xml". Voici un exemple de contenu :

```
<resources>
    <string name="message1">Voici un exemple</string>
    <string name="message2">Et encore un autre !</string>
</resources>
```

Et bien la manipulation est assez simple. Il suffit de créer des répertoires "res / values" personnalisés. Par exemple, pour une traduction en français, utilisez" res / values-fr" avec le fichier contenant les traductions françaises !

Le même mécanisme peut être appliqué à d'autres ressources comme les images, ce qui permet de contextualiser encore plus finement vos interfaces.

Différenciation en fonction du matériel

De façon similaire, il est possible de personnaliser vos ressources en fonction du matériel sur lequel votre application s'exécutera.

Plusieurs critères sont utilisables comme la densité pixel de l'écran (ldpi pour low, mdpi pour medium ...), les dimensions de l'écran, le type d'interactivité utilisable avec l'écran (stylet, doigt, ...) etc.

En sachant cela, on comprend mieux l'utilité des répertoires créés par défaut lors de l'initialisation d'un nouveau projet Android (avec les dernières versions de SDK) : drawable-ldpi, drawable-hdpi, etc.

CONCLUSION

Voici donc quelques bases pour la création d'interfaces Android. La plate-forme permet beaucoup plus, notamment en termes de composants personnalisés ou même de gadgets intégrables sur le bureau.

Avec un peu de pratique et l'aide des concepteurs d'interface, la mise en place d'interfaces graphiques devient rapidement intuitive. Les mécanismes disponibles - basés notamment sur XML - sont à la fois simples et efficaces ... les anciens développeurs J2ME ne me contrediront pas ! Enfin ayez à l'esprit qu'à la différence de l'iPhone par exemple, la plate-forme Android est disponible sur une variété d'appareils très importante.

Vos développements devront (donc) en tenir compte au risque de limiter les cibles de votre application à un sous-ensemble d'appareils.

Bon design !

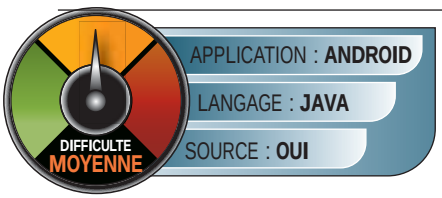
Références

- Android par Google : http://developer.android.com
- Best practices Google pour la prise en compte d'écrans hétérogènes : http://developer.android.com/guide/practices/screens_support.html

■ Damien **Guignard**

Détecteur d'événements sous Android : l'application BigBrother

Le SDK d'Android propose un modèle de composants et des API pour gérer différents dispositifs qui font la particularité des plates-formes mobiles : connectivité, capteurs, téléphonie, multimédia … Dans cet article nous allons nous intéresser à la détection d'événements liés à la téléphonie et la géolocalisation.



L'activité est le premier composant essentiel permettant de gérer le cycle de vie d'une application et l'interactivité avec l'utilisateur ; mais qu'en est-il lorsqu'on souhaite exécuter un traitement en tâche de fond, qui démarre automatiquement, et qui doit réagir à des événements externes comme un appel téléphonique ? Le framework propose pour cela des composants de type service et receiver qui utilisent des intentions (Intent) pour collaborer [Fig.1].

LES SERVICES : POUR DES TRAITEMENTS EN TÂCHE DE FOND

Le service peut être vu comme une activité à longue durée de vie (potentiellement infinie), en tâche de fond, et privée d'IHM ; il est implémenté par une classe qui doit étendre android.app.Service.

Démarrage du service : à la différence d'une activité, l'utilisateur ne dispose pas de raccourci dans son bureau, il faudra donc démarrer le service explicitement de manière programmatique (souvent depuis une activité). Les services d'une application doivent être déclarés dans son manifeste :

```
 <application  android:icon="@drawable/icon"  android:label="
@string/app_name">
    <activity .../>
    <service android:name=".Service" />
 </application>
```

Un service peut, comme une activité, enregistrer des écouteurs spécialisés (listeners) pour obtenir des informations sur un capteur particulier, il est alors responsable du désenregistrement des listeners.

RECEIVER : UN DÉCLENCHEUR LÉGER

Le receiver est un composant susceptible de recevoir des intentions exprimées par le système Android ou d'autres applications. Les intentions symbolisent des requêtes (ou souhaits) et sont orchestrées par le framework ; cela offre un cadre simple et générique qui fait penser au style d'architecture REST (ou au Web), et confère un niveau d'abstraction intéressant pour faciliter l'intégration de composant. L'intérêt de ce composant réside en son exécution automatique dès qu'un événement correspondant survient. Il agit donc souvent comme un déclencheur : par exemple, pour lancer un service. Son implémentation nécessite d'étendre la classe android.content.BroadcastReceiver et de déclarer les filtres d'intentions dans le manifeste :

```
 <receiver android:name=".EventReceiver">
   <intent-filter>
     <category android:name="android.intent.category.DEFAULT" />
     <action android:name="android.intent.action.PHONE_STATE" />
   </intent-filter>
 </receiver>
```

Lors d'un broadcast d'événement du système, de nombreux récepteurs peuvent être prévenus. Pour éviter les embouteillages, le traitement de chaque récepteur doit être court, quitte à sous-traiter à un autre composant (le service par exemple).

DÉTECTION D'ÉVÉNEMENT : MISE EN ŒUVRE

Il existe deux approches pour détecter des événements sous Android : utiliser un broadcast receiver générique, ou enregistrer autant de listeners spécifiques que nécessaire.

Caractéristiques d'un broadcast receiver :

- pas besoin d'enregistrer le composant, il suffit de le déclarer
- réagit à tous les événements, pourvu que les bons filtres d'intentions soient déclarés
- simplicité de mise en oeuvre : une seule méthode à redéfinir (éviter tout de même l'inflation)
- pas besoin de démarrage : l'instanciation et l'invocation de la méthode sont automatiques
- les informations fournies sont souvent insuffisantes pour certains événements

Caractéristiques d'un listener :

- informations spécialisées et détaillées
- pas besoin de déclaration dans le manifeste
- nécessité d'enregistrer chaque listener
- peu de factorisation possible (particularités des listeners)

Pour ces raisons, il n'est pas rare d'utiliser les deux approches de manière complémentaire. Dans les deux cas, les permissions nécessaires devront bien évidemment être déclarées dans le manifeste. L'architecture générale de l'application est illustrée par les schémas : [Fig.2+3]

DÉTECTER UN APPEL TÉLÉPHONIQUE ENTRANT

Maintenant que les principes sont posés, voyons un exemple de mise en œuvre avec une application « BigBrother » capable de détecter un appel téléphonique entrant. Nous avons besoin d'un objet métier pour encapsuler la notion d'événement :

```
package com.programmez.android.bigbrother.domain;
```

```
public class Event {
  public long eventId; // Identifiant unique de l'événement
  public String deviceId; // Identifiant unique du dispositif Android
  public String eventType; // Type de l'événement
  public long timestamp; // Timestamp de l'événement sur le dispositif
  public String params; // Eventuelles données supplémentaires
}
```

À noter : utilisation de propriétés publiques plutôt qu'un bean anémique.

Réalisons le récepteur d'événements :

```
package com.programmez.android.bigbrother;

import static com.programmez.android.bigbrother.Constants.*;
import android.content.Context;
import android.content.Intent;
import android.telephony.TelephonyManager;
import android.util.Log;
import com.programmez.android.bigbrother.domain.Event;

public class EventReceiver extends android.content.Broadcast
Receiver {
  private Event event;

  private void onReceivePhoneStateChanged(final Context context,
      final Intent intent) {
    final String phoneState = intent
      .getStringExtra(TelephonyManager.EXTRA_STATE);
    final String phoneNumber = intent
      .getStringExtra(TelephonyManager.EXTRA_INCOMING_NUMBER);
    event.eventType = EVENT_TYPE_PHONE_STATE_CHANGED;
    final StringBuilder sb = new StringBuilder();
    if (TelephonyManager.EXTRA_STATE_IDLE.equals(phoneState)) {
      sb.append("idle");
    } else if (TelephonyManager.EXTRA_STATE_OFFHOOK.equals(phoneState)) {
      sb.append("offhook");
    else if (TelephonyManager.EXTRA_STATE_RINGING.equals(phoneState)) {
      sb.append("ringing");
    } else {
      sb.append(phoneState);
    }
    if (phoneNumber != null) {
      sb.append(',');
      sb.append("number:" + phoneNumber);
    }
    event.params = sb.toString();
    Log.v(LOG_TAG, "event : " + event);
  }

  @Override
  public void onReceive(final Context context, final Intent intent) {
    event = null;
    final String action = intent.getAction();
    event = new Event();
    if (TelephonyManager.ACTION_PHONE_STATE_CHANGED.equals(action)) {
      onReceivePhoneStateChanged(context, intent);
    } else {// Default event code
      event.eventType = action;
      final String data = intent.getDataString();
      Log.v(LOG_TAG, "broadcast : action=" + action + ", data=" + data);
      Log.v(LOG_TAG, "event : " + event);
    }
  }
}
```

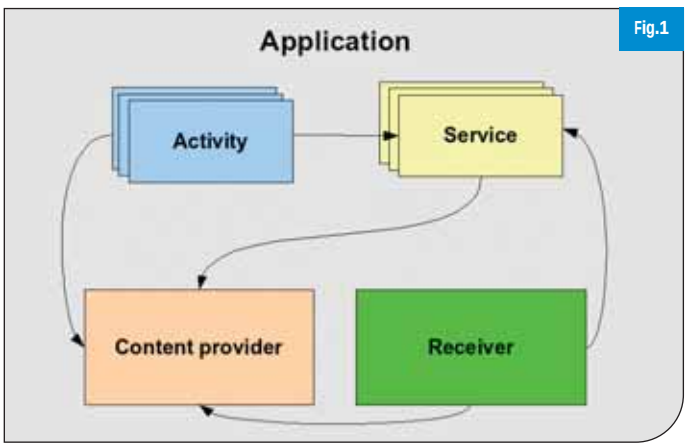


Fig.1

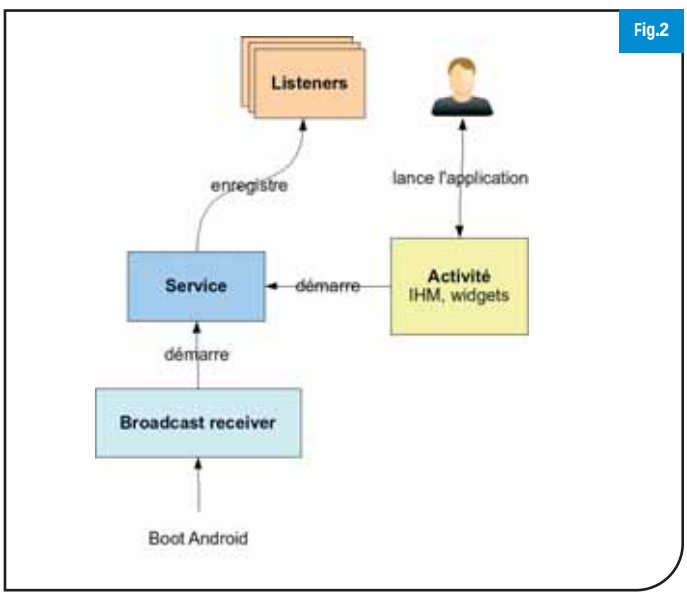


Fig.2

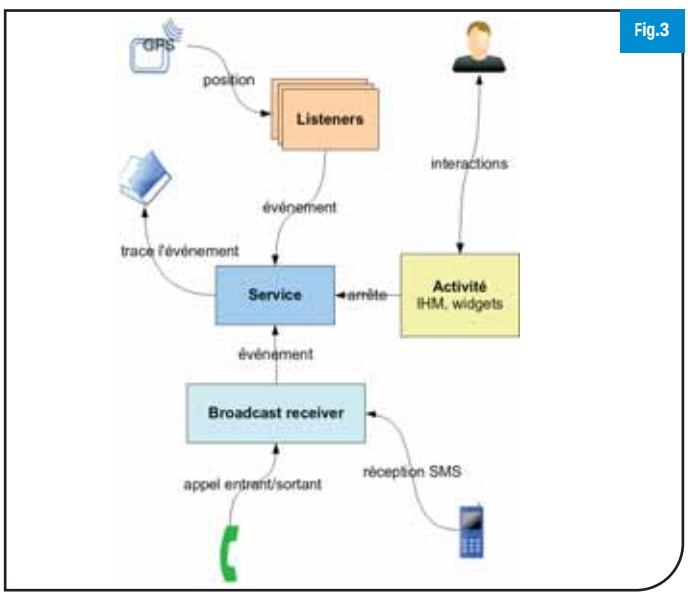


Fig.3

Constantes de l'application :

```
package com.programmez.android.bigbrother;

public interface Constants {
 String LOG_TAG = BigBrother.class.getSimpleName();
 String EVENT_TYPE_PHONE_STATE_CHANGED = "phone state changed";
}
```

Notre récepteur est censé réagir à l'intention ACTION_PHONE_STATE_CHANGED ; il faut donc adapter le manifeste :

```
<receiver android:name=".EventReceiver">
  <intent-filter>
    <category android:name="android.intent.category.DEFAULT" />
    <action android:name="android.intent.action.PHONE_STATE" />
  </intent-filter>
</receiver>
```

Et ne pas oublier les permissions :

```
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
```

Fonctionnement du récepteur :
La seule méthode à redéfinir est *onReceive* ; elle fournit un contexte et l'intention qui a été exprimée. Dans notre cas, la méthode vérifie l'intention, délègue le traitement à une autre méthode s'il s'agit de l'intention qui nous intéresse, sinon applique un comportement par défaut. On se limite ici à tracer l'événement. L'intention fournit quelques informations supplémentaires (extra), indispensables pour bien qualifier l'événement : les états RINGING, OFFHOOK, IDLE correspondant respectivement à un appel qui arrive (sonnerie), la prise d'appel, et la fin d'appel. On dispose en plus du numéro appelant via la clé EXTRA_INCOMING_NUMBER ; on en profite pour l'ajouter à notre événement.
Pour cet exemple, le broadcast receiver répond bien à notre besoin en fournissant les informations pertinentes via le paramètre Intent. Ce n'est malheureusement pas toujours le cas, et pour certains événements les informations à disposition sont pauvres ; dans ce cas les listeners viennent en renfort pour fournir des informations spécialisées. En fait, le broadcast receiver correspond à un besoin de détecter un changement d'état d'un périphérique, pas beaucoup plus…

GÉOLOCALISATION : DÉTECTER UNE NOUVELLE POSITION

La géolocalisation est un bon exemple d'utilisation d'un listener. On souhaite obtenir la position géographique à chaque changement suivant des critères de distance et de période. L'acquisition des informations qui nous intéressent (latitude et longitude) nécessite la mise en œuvre d'un LocationListener.
Pour rendre le système industrialisable, une classe abstraite pourra figer le contrat des implémentations de listeners :

```
package com.programmez.android.bigbrother.listener;

import android.content.Context;

public abstract class Listener {
 private boolean registered;

 public boolean isRegistered() {
   return registered;
 }

 public boolean register(final Context context) {
   registered = false;
   if (registerImpl(context)) {
     registered = true;
     return true;
   }
   return false;
 }

   protected abstract boolean registerImpl(final Context context);

   public boolean unregister(final Context context) {
     if (unregisterImpl(context)) {
       registered = false;
       return true;
     }
     return false;
 }

   protected abstract boolean unregisterImpl(final Context context);
}
```

Implémentation du LocationListener :

```
package com.programmez.android.bigbrother.listener;

import static com.programmez.android.bigbrother.Constants.*;
import java.util.List;
import android.content.Context;
import android.location.Location;
import android.location.LocationManager;
import android.os.Bundle;
import android.util.Log;
import com.programmez.android.bigbrother.Service;
import com.programmez.android.bigbrother.domain.Event;

public class LocationListener extends Listener implements
   android.location.LocationListener {
 private static LocationManager manager;
 private static List<String> locationProviders;

 public void onLocationChanged(final Location location) {
   final Service service = Service.getInstance();
   final Event event = new Event();
   event.eventType = EVENT_TYPE_LOCATION;
   final StringBuilder sb = new StringBuilder();
   sb.append("latitude:");
   sb.append(location.getLatitude());
   sb.append(",longitude:");
   sb.append(location.getLongitude());
   event.params = sb.toString();
   service.acceptIncomingEvent(event);
 }
```

```
  public void onProviderDisabled(final String provider) {
    // Do nothing
  }

  public void onProviderEnabled(final String provider) {
    // Do nothing
  }

  public void onStatusChanged(final String provider, final int status,
      final Bundle extras) {
    // Do nothing
  }

  @Override
  protected boolean registerImpl(final Context context) {
    if (!Service.checkPermissions(context, new String[] {
        android.Manifest.permission.ACCESS_COARSE_LOCATION,
        android.Manifest.permission.ACCESS_FINE_LOCATION }))
      return false;
    manager = (LocationManager) context
        .getSystemService(Context.LOCATION_SERVICE);
    locationProviders = manager.getAllProviders();
    for (final String provider : locationProviders) {
      Log.v(LOG_TAG, "register location provider : " + provider);
      manager.requestLocationUpdates(provider,   LOCATION_MIN_
UPDATE_TIME,LOCATION_MIN_UPDATE_DISTANCE, this);
    }
    return true;
  }

  @Override
  protected boolean unregisterImpl(final Context context) {
    if (manager != null) {
      manager.removeUpdates(this);
      return true;
    }
    return false;
  }
}
```

L'enregistrement du listener est déclenché par un composant de type service dont vous trouverez le code complet sur notre site www.programmez.com.

Lorsque le service démarre, il enregistre les différents listeners (ici seulement un LocationListener) ; et les désactive en s'arrêtant.

Le service fournit aussi la méthode *acceptIncomingEvent* pour traiter tous les événements arrivant de manière centralisée, raffiner l'événement avec l'identifiant du terminal (subscriberId) et le timestamp puis notifier l'événement (méthode notifyEvent qui se limite à tracer). Le service respectant le pattern singleton, un autre composant pourra facilement interagir avec lui grâce à la méthode *getInstance* et vérifier qu'il est bien en exécution (isRunning).

Fig.4

À noter l'utilisation des *SharedPreferences* pour mémoriser l'état du service (démarré ou pas). Le démarrage du service doit être déclenché par un autre composant, classiquement une activité.

Ajoutons donc un bouton au layout qui permettra à l'utilisateur de démarrer ou arrêter le service : [Fig.4]

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout   xmlns:android="http://schemas.android.com/apk
/res/android"
  android:id="@+id/mainLayout" android:layout_height="fill_parent"
  android:layout_width="fill_parent" android:background="#FFFFFF"
  android:orientation="vertical">
  <TextView android:id="@+id/TextView01" android:layout_height=
"wrap_content"
    android:layout_width="fill_parent" android:gravity="center_
horizontal"
    android:paddingTop="20sp" android:paddingBottom="20sp"
    android:textStyle="bold" android:textColor="#000066" android:
text="@string/app_name"
    android:textSize="30sp"></TextView>
  <ToggleButton android:textOff="@string/start_service"
    android:textOn="@string/stop_service" android:layout_width=
"fill_parent"
    android:layout_height="wrap_content"   android:id="@+id/
ToogleServiceButton" />
</LinearLayout>
```

Pour faire en sorte que le service démarre automatiquement, utilisons le broadcast receiver qui est capable de détecter un événement de fin de boot. Modifications de la classe EventReceiver :

```
public class EventReceiver extends android.content.Broadcast
Receiver {
  private Service service;
  private Event event;

  @Override
  public void onReceive(final Context context, final Intent intent) {
    service = null;
    event = null;
    final String action = intent.getAction();
    if (Intent.ACTION_BOOT_COMPLETED.equals(action)) {
      onReceiveBoot(context, intent);
      return;
    }
    service = Service.getInstance();
    if (service == null || !Service.isEnabled(context) || !Service.
isRunning())
      return;
    event = new Event();
    if (TelephonyManager.ACTION_PHONE_STATE_CHANGED.equals(action))
      onReceivePhoneStateChanged(context, intent);
    else {// Default event code
      event.eventType = action;
      final String data = intent.getDataString();
      Log.v(LOG_TAG, "broadcast : action=" + action + ", data=" + data);
      service.acceptIncomingEvent(event);
    }
  }
```

```
private void onReceiveBoot(final Context context, final Intent intent) {
  Log.v(LOG_TAG, "boot completed");
  if (Service.isEnabled(context) && !Service.isRunning())
    Service.start(context, false);
}
```

Remplacer la dernière ligne de la méthode *onReceivePhoneStateChanged* pour confier le traitement de l'événement au service :

```
private void onReceivePhoneStateChanged(,,,) {
  ...
  service.acceptIncomingEvent(event);
}
```

Il reste à modifier le manifeste pour ajouter le nouveau filtre d'intention et les permissions :

```
    <receiver android:name=".EventReceiver">
      <intent-filter>
        <category android:name="android.intent.category.DEFAULT" />
        <action android:name="android.intent.action.BOOT_COMPLETED" />
        <action android:name="android.intent.action.PHONE_STATE" />
      </intent-filter>
    </receiver>
  </application>
  <uses-sdk android:minSdkVersion="3" />
  <uses-permission   android:name="android.permission.RECEIVE_
BOOT_COMPLETED" />
  <uses-permission android:name="android.permission.READ_PHONE
_STATE" />
  <uses-permission android:name="android.permission.ACCESS_COARSE
_LOCATION" />
  <uses-permission   android:name="android.permission.ACCESS_
FINE_LOCATION" />
```

Lorsque l'utilisateur clique sur le widget ToggleButton, le service démarre. L'utilisateur peut ensuite sortir de l'activité (touche back), cela n'aura pas d'effet sur le service qui continuera de tourner.
Si l'utilisateur éteint et rallume son téléphone Android, le service sera lancé automatiquement juste après le boot par le broadcast receiver. L'état du ToggleButton est sauvegardé dans les préférences partagées et utilisé partout pour savoir si le service doit démarrer ou pas, et si les événements doivent être traités.
Ainsi, l'activité se résume à donner à l'utilisateur un moyen de contrôle sur le lancement du service.

POUR ALLER PLUS LOIN...

Cet embryon d'application peut facilement évoluer et gérer de nombreux événements en complétant le receiver, ou en implémentant d'autres listeners ; pour y intégrer par exemple la réception de SMS. Par ailleurs, le traitement de l'événement pourrait aussi évoluer pour publier les informations sur un serveur. Une application RESTful serait alors une solution de premier choix, d'autant que l'implémentation du client sous Android est très simple.
Cela implique, qu'au moment du traitement, le téléphone ait accès à Internet, sous peine de générer une erreur et potentiellement perdre l'événement. Là encore, une solution simple consiste à, tout d'abord, persister les informations localement dans une base SQLite ; le service se contentant alors d'alimenter la base, tandis qu'un nouveau composant (un thread poster) aurait en charge de consulter la base, publier les événements si une connexion 3G ou Wifi est disponible (pour ça aussi il y a un événement !), et purger la base.
Ce mode de fonctionnement est illustré dans les schémas : [Fig.5 et 6]

BigBrother is watching you !

Sur cette base, il est assez aisé d'imaginer une application «BigBrother» déployable, avec sur le serveur une interface Web qui permet de consulter les informations, faire des statistiques, et plein d'autres choses... Bien évidemment, outre les aspects techniques traités ici et la créativité que peut susciter ce type de développement, une telle application n'est pas sans poser un certain nombre de questions d'ordre légal, moral et social : respect de la vie privée, conformité CNIL ...

■ Olivier **Penhoat**
Consultant & Formateur
Valtech Training

Fig.5

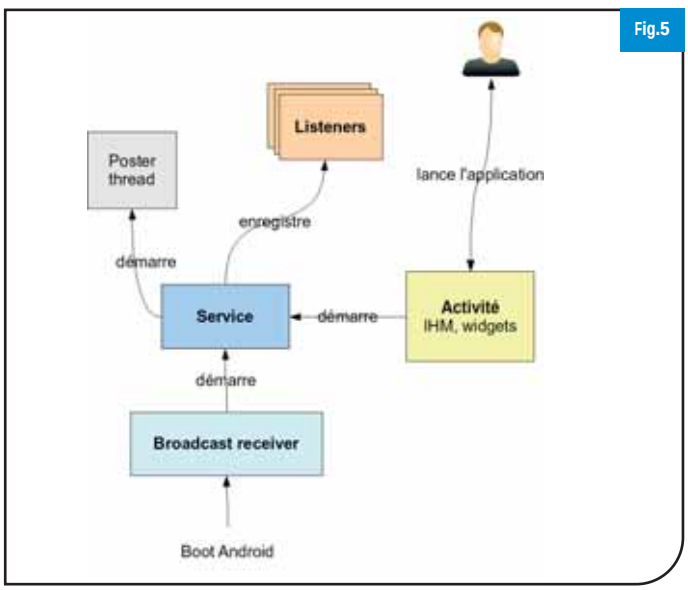
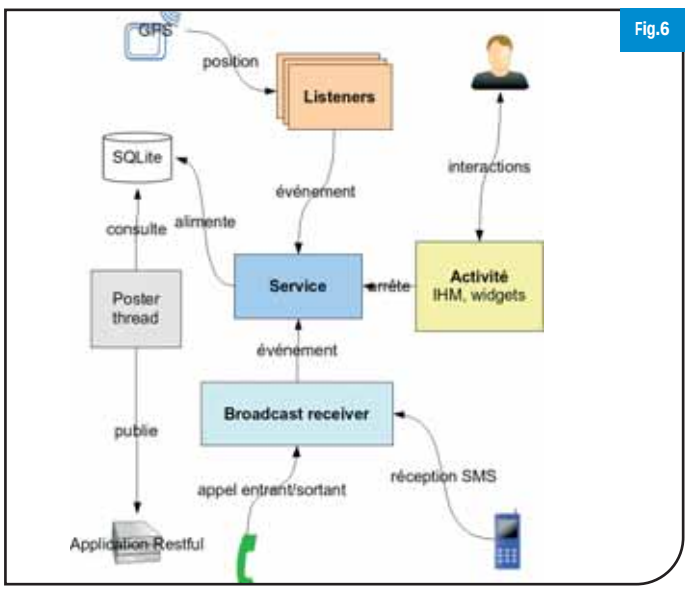


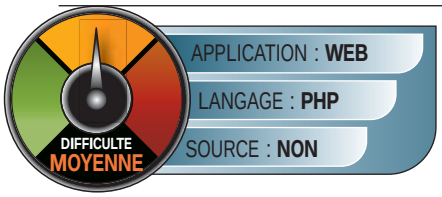
Fig.6



Le nouveau Symfony est arrivé !

2e partie

Le Framework Symfony 2 affiche une nouvelle orientation du framework par rapport aux versions précédentes. Cette nouvelle version permet aux développeurs de mieux construire, et plus facilement, des sites web avec PHP.



LE CONTRÔLEUR

Dans une page web, le contenu ne peut être 100 % statique, il est nécessaire de le rendre dynamique. Pour ajouter du contenu dynamique, vous devez passer par un contrôleur, qui appelle une classe PHP, et l'afficher au format HTML. Pour illustrer ces quelques lignes, vous pouvez consulter le fichier *HelloController.php* qui se trouve à l'emplacement suivant :

```
src/Application/HelloBundle/Controller
```

```
<?php

namespace Application\HelloBundle\Controller;

use Symfony\Framework\WebBundle\Controller;

class HelloController extends Controller
{
  public function indexAction($name)
  {
    return $this->render('HelloBundle:Hello:index', array('name'
 => $name));
  }
}
```

Avec Symfony 2, l'utilisation des « namespaces » est devenue standard, car cette fonction est apparue avec l'arrivée de PHP 5.3. Elle va vous permettre d'utiliser un contrôleur spécifique. Comme cela, la classe HelloController va étendre la classe Controller. Bien entendu, chaque contrôleur possède différentes actions avec différents arguments. Les contrôleurs retournent obligatoirement une réponse : la méthode render() utilisée ici permet de retourner le résultat d'un template donné.

RÉCUPÉRATIONS DE DONNÉES

Lorsque vous choisissez de réaliser un site dynamique, de nombreuses informations arrivent lors du chargement d'une page HTML. Ces données peuvent provenir d'un formulaire de saisie, d'une requête ou de configurations. Ces différentes méthodes possibles se présentent de la façon suivante :

```
<?php
$request = $this->getRequest();
$request->isXmlHttpRequest();
$request->getPreferredLanguage(array('en', 'fr'));
$request->getQueryParameter('page');
$request->getRequestParameter('page');
?>
```

La première ligne récupère des données provenant d'une requête HTTP, la seconde vérifie s'il s'agit d'une requête Ajax, la troisième la langue de l'utilisateur, et les quatrième et cinquième lignes récupèrent des informations venant d'un formulaire GET ou POST.

LE BUNDLE

Le Bundle correspond à un paquet lorsque vous utilisez Symfony (appelé plugin dans d'autres API). Il est une des briques principales dans Symfony, et permet d'avoir une certaine flexibilité pour l'utilisation des paquets déjà existants. Ainsi, vous pouvez choisir les éléments dont vous avez besoin dans votre application et de les optimiser. Les paquets proposés vont vous permettre d'implémenter les fonctionnalités requises par votre application, d'utiliser un mécanisme d'authentification et de connexion des utilisateurs, ou encore la gestion des connexions à votre base de données à partir de PDO ou d'un ORM tel que Doctrine. Aujourd'hui, quelques bundles existent tels qu'un livre d'or ou une mini-application de gestion de panier.

L'ARCHITECTURE

L'arborescence de base est très souple, dans le sens où rien n'est imposé, et où le développeur est libre de l'adapter à sa convenance. Toutefois, il est conseillé de suivre une architecture standard. Si vous décidez d'utiliser celle proposée par Symfony, vous devez retrouver une architecture comme le montre l'image (1). Le dossier « hello » est le nom de votre application. Ici, il s'agit de l'application « Hello World ». Vous pouvez donc changer ce nom pour qu'il corresponde au nom de votre application. Le dossier « web » est destiné à contenir le répertoire racine web, c'est-à-dire, les fichiers Javascripts, css, images. Par ailleurs, les appels du contrôleur s'effectuent dans ce dossier, via le fichier index.php par exemple. Le dossier « src » doit contenir l'ensemble de vos fichiers PHP, ainsi que le code du framework. Le fichier autoload.php permet de surcharger le mécanisme d'autoloading afin de pouvoir ajouter plus facilement des librairies externes.

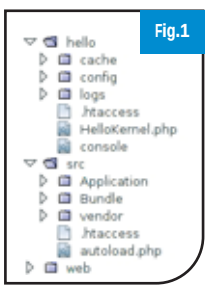


Fig.1

RÉALISER UN BUNDLE

Tout doit être encapsulé dans un bundle, plusieurs outils sont donc fournis avec le framework pour faciliter leur développement. Plusieurs types de bundles coexistent dans un projet Symfony :

- Les bundles du framework
- Les bundles « publics », potentiellement partageables entre différents projets, placés dans le répertoire *src / Bundle*
- Les bundles privés, qui sont propres au projet, placés dans le répertoire *src / Application*

Nous allons réaliser un client Twitter basique, ce qui permettra d'apprendre à générer un bundle, créer une extension à l'injecteur de dépendance, configurer le routing et utiliser les templates.

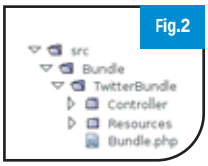
Initialiser un bundle

La première étape est de créer l'arborescence minimale d'un bundle. Pour cela, le framework propose la tâche suivante :

```
$ php app/console init:bundle [namespace]
```

Le paramètre "namespace" indique l'endroit où le bundle doit être installé. Pour initialiser le bundle «TwitterBundle » dans le répertoire Bundle, il faut lancer la commande suivante :

```
$ php console init:bundle "Bundle\\TwitterBundle"
```



Cette commande (image 2) va générer l'arborescence

Anatomie du bundle

La structure d'un bundle est composée des éléments suivants :

- Bundle.php : il s'agit de la classe d'initialisation du bundle
- Controller / : le répertoire dédié aux contrôleurs
- Resources / : les ressources du bundle, à savoir les vues, la configuration ou encore les images, css et Javascripts

Activer le bundle

Pour utiliser le bundle généré, il est nécessaire de modifier le kernel de l'application. Cette classe est en charge de la configuration et de l'initialisation des différentes ressources du projet. Il faut donc modifier la méthode *registerBundles()* de l'application :

```
<?php
# /hello/HelloKernel.php
/* ... */

class HelloKernel extends Kernel
{
  /* ... */

  public function registerBundles()
  {
    return array(
      /* ... */
      new Bundle\TwitterBundle\Bundle(),
    );
  }
}
```

Créer un service twitter

Pour accéder à l'API de twitter, il est nécessaire de créer un service. Un service est un élément de l'injecteur de dépendance. Pour faciliter cette étape, l'exemple utilisera la librairie php-twitter, disponible ici : http://github.com/noelg/php-twitter, et elle doit être téléchargée dans le répertoire src / vendor / php-twitter. Il faut ensuite modifier le fichier src / autoload.php pour pouvoir charger correctement la classe Twitter (surligné en gris) :

```
// for Zend Framework & SwiftMailer
set_include_path(__DIR__.'/vendor/php-twitter/lib'. PATH_SEPARATOR.
__DIR__.'/vendor/zend/library'.   PATH_SEPARATOR.__DIR__.'/ven
dor/swiftmailer/lib'. PATH_SEPARATOR.get_include_path());
```

CONFIGURER L'INJECTEUR DE DÉPENDANCE

Il faut ensuite ajouter la configuration nécessaire pour intégrer cette classe à l'injecteur de dépendance. Pour cela, créez un fichier twitter.yml dans le répertoire Resources du bundle, et ajoutez la configuration suivante :

```
parameters:
  twitter.class:    Twitter
  twitter.username: ~
  twitter.password: ~
  twitter.init_file: twitter_init.php

services:
  twitter:
    class: %twitter.class%
    file:  %twitter.init_file%
    arguments:
      username: %twitter.username%
      password: %twitter.password%
```

L'ensemble de cette configuration est détaillé ici : http://components.symfony-project.org/dependency-injection/documentation.

CRÉER UNE EXTENSION À L'INJECTEUR DE DÉPENDANCE

L'inconvénient d'un conteneur d'injection de dépendance est que celui-ci nécessite un nombre de paramètres relativement important. Les extensions du conteneur d'injection de dépendance permettent de faciliter cette configuration en créant des « raccourcis » de configuration dans votre application. Pour cela, créez un répertoire *DependencyInjection* à la racine du bundle, et créez un fichier TwitterExtension.php, et ajoutez alors le code suivant :

```
<?php

namespace Application\TwitterBundle\DependencyInjection;

use Symfony\Components\DependencyInjection\Loader\LoaderExtension;
use Symfony\Components\DependencyInjection\Loader\YamlFileLoader;
use Symfony\Components\DependencyInjection\Builder;
use Symfony\Components\DependencyInjection\BuilderConfiguration;

class TwitterExtension extends LoaderExtension
{
  public function twitterLoad($config)
  {
    $configuration = new BuilderConfiguration();

    $loader = new YamlFileLoader(__DIR__.'/../Resources/config');
    $configuration->merge($loader->load('twitter.yml'));

    if (isset($config['username']))
    {
      $configuration->setParameter('twitter.username', $config
['username']);
    }

    if (isset($config['password']))
    {
```

```
     $configuration->setParameter('twitter.password', $config
['password']);
    }

    return $configuration;
  }

  public function getAlias()
  {
    return 'twitter';
  }
}
```

Le service *twitter* devient alors configurable depuis le fichier config.yml de l'application, avec ces trois paramètres de configuration :

```
twitter.twitter:
  username: VOTRELOGIN
  password: MOTDEPASSE
```

Routing

Pour accéder aux différents contrôleurs d'un bundle, Symfony a besoin de règles de routage. Les règles sont définies dans le fichier routing.yml du projet, dans *app / config / routing.yml*. La syntaxe de ce fichier permet d'importer des règles de routing depuis d'autres fichiers de configuration. Vous allez donc ajouter une règle de routing dans le fichier routing.yml du bundle :

```
timeline:
  url: /timeline
   defaults: { _bundle: TwitterBundle, _controller: Twitter,
_action: index }
```

La configuration du routing nécessite les paramètres suivants :

- Un identifiant unique, qui pourra servir à générer les urls
- Un pattern, c'est-à-dire le format de l'URL. Ici l'url sera */ timeline*
- Les paramètres par défaut : il faut au minimum les trois définis ci-dessus :
  - _bundle : le nom du bundle à utiliser
  - _controller : le nom du contrôleur
  - _action : le nom de l'action

Il faut ensuite importer ce fichier depuis la configuration du projet :

```
twitter:
  resource: TwitterBundle/Resources/config/routing.yml
  prefix: /twitter
```

Contrôleur

Ensuite, modifiez le code du contrôleur afin de récupérer les derniers tweets de vos amis. Pour cela, nous allons récupérer le service twitter créé précédemment.

```
<?php


namespace Application\TwitterBundle\Controller;

use Symfony\Framework\WebBundle\Controller;

class TwitterController extends Controller
{
  public function indexAction()
  {
    $twitter = $this->container->getTwitterService();
    $entries = $twitter->statuses->friends_timeline();

    return $this->render('TwitterBundle:Twitter:index', array
('entries' => $entries));
  }
}
```

Le code de l'action est relativement simple à comprendre : il retourne le résultat de la méthode *render()*. Celle-ci prend un paramètre qui correspond au nom du template que l'on souhaite utiliser : *'TwitterBundle:Twitter:index'*. Cette valeur se décompose en trois informations : le nom du bundle, le nom du contrôleur et le nom du template. Dans ce cas, on souhaite utiliser le template « index », du contrôleur « Twitter » du bundle « TwitterBundle ».

Template

Enfin, créez le template index.php dans le répertoire Resources / views du bundle, avec le code suivant :

```
<h1>Twitter Timeline</h1>

<div>
  <?php foreach ($entries as $entry): ?>
  <div class="entry">
    <img src="<?php echo $entry->user->profile_image_url ?>" />
    <p>
       <a href="http://twitter.com/<?php echo $entry->user->
screen_name ?>">
        <?php echo $entry->user->screen_name ?>
      </a>
      <span>
       <?php echo $entry->user->screen_name.' '.$entry->text ?>
      </span>
    </p>
    <p>
      <?php echo $entry->created_at ?> From <?php echo $entry->source ?>
    </p>
  </div>
  <?php endforeach ?>
</div>
```

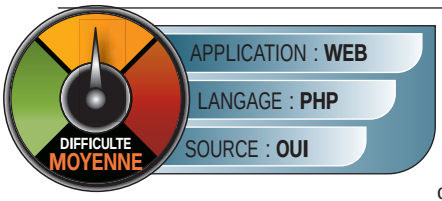
Développer un bundle est une tâche indispensable dans un projet Symfony 2. Le fait qu'il soit à la base de tout développement constitue un avantage car il permet de structurer le développement en brique logicielle, et ainsi de mieux organiser le développement des projets.

■ Christophe **Villeneuve**
Consultant pour Alter Way Solutions et auteur du livre « PHP & MySQL-MySQLi-PDO, Construisez votre application » 2e édition, aux Éditions ENI. Rédacteur pour nexen.net, Trésorier AFUP et membre du LeMug.fr, PHPTV, PHPteam...

■ Noël **Guilbert**
Consultant et formateur chez Sensio Labs, société créatrice du framework Symfony. Passionné par le développement et Internet, il encourage l'utilisation du PHP en milieu professionnel en s'investissant dans la communauté Symfony.

Un événement en PHP ? Non, mais des événements !

La notion d'événements est généralement absente de la conception d'application PHP, au moins au sens classique du terme, pour la simple raison que le langage n'offre pas de support natif de cette approche de la programmation. Si certains frameworks ou CMS, comme Drupal et ses hooks, en proposent leur propre implémentation, nous allons voir ici une ingénieuse librairie permettant d'étendre le principe à toute application.



Initialement pensée pour matérialiser ce qui constitue l'essence du framework Symfony 2, à savoir le découplage maximum des composants, la librairie sfEventDispatcher est librement inspirée d'un système équivalent au coeur des systèmes OS X d'Apple, le Cocoa Notification Center. En effet, l'échange de messages entre le système d'exploitation et les applications comme entre les applications elles-mêmes est le concept central autour duquel est bâtie l'architecture de MacOS X.

L'idée simple à l'origine de sfEventDispatcher, c'est qu'à l'instar du système d'Apple, une application PHP est un microcosme dans lequel évoluent différents composants, qui sans nécessairement avoir conscience les uns des autres, ont besoin de communiquer entre eux pour constituer un workflow. La majorité des composants des applications modernes, même en PHP, sont représentés par des classes, dont les méthodes, lors de leur exécution, peuvent être assimilées à des « moments » du flux d'exécution. L'enchaînement des événements est orchestré en général par un composant central, ou frontal (vocable particulièrement usité dans un contexte MVC), qui décide qui fait quoi, et dans quel ordre, le tout en fonction de la requête de l'utilisateur. La gestion d'événements dans l'application peut, dans une certaine mesure, être mise en parallèle de ce processus, mais à l'échelle du composant. Fondamentalement, on permet aux méthodes d'informer les autres composants de ce qu'elles sont en train de faire, en leur donnant la possibilité de prendre la main le temps d'ajouter leurs propres traitements à ceux de la méthode appelante. La différence principale résidant bien sûr en ce que chaque méthode peut déclencher un ou plusieurs traitements, faisant ainsi prendre au workflow des chemins de traverse que ne saurait gérer un composant central, sous peine de la confusion la plus totale. Cette manière de fonctionner a de nombreuses qualités, dont la souplesse évolutive n'est pas la moindre. En effet, une fois un traitement implémenté, en fonction d'un cahier des charges précis, il est possible d'en modifier certains aspects, ou de les étendre, sans pour autant toucher au code initial, si ce n'est en ajoutant les deux lignes de code nécessaires à l'émission d'un événement. Autrement dit, on va pouvoir ajouter des fonctionnalités sans remettre en cause un code testé et validé. Le risque de régression est donc réduit drastiquement.

En outre, en émettant un seul événement, une méthode peut déclencher de nombreux traitements, sans avoir besoin de connaître jusqu'à leur existence ! Cette caractéristique, tout à fait formidable pour développer un système de plugins fiable et simple (c.f. Les hooks de Drupal), constitue également ce qui est la plus grosse faiblesse, si ce n'est la seule, du concept : la multiplication des plugins peut entraîner une sérieuse problématique concernant l'ordonnancement de leur exécution. Nous avons consacré un encadré à ce problème pour vous proposer les stratégies les plus sûres pour éviter les écueils.

INSTALLATION

Rien que de très classique à ce chapitre… deux classes seulement le composent, sfEventDispatcher et sfEvent. Assurez-vous simplement d'inclure les deux fichiers qui les contiennent de manière systématique dans tous les scripts de votre application. Concernant l'installation, le plus simple consiste à utiliser le canal PEAR de Symfony, à l'aide des deux commandes suivantes :

```
pear channel-discover pear.symfony-project.com
pear install symfony/EventDispatcher
```

Avec une installation en bonne est due forme de PHP, les classes seront accessibles dans votre include_path, que vous pourrez charger comme suit :

```
// sfEventDispatcher.php s'occupe de charger également sfEvent.php
require 'SymfonyComponents/EventDispatcher/sfEventDispatcher.php';
```

Vous pouvez bien sûr également télécharger une archive tgz ou zip et placer ces sources où vous le souhaitez.

PRISE EN MAIN

La prise en main de sfEventDispatcher est vraiment très intuitive. En témoigne le fragment de code suivant, parfaitement fonctionnel et faisant appel aux deux méthodes principales de sfEventDispatcher : notify() et connect() :

```
// chargement des librairies
require 'SymfonyComponents/EventDispatcher/sfEventDispatcher.php';

$dispatcher = new sfEventDispatcher();

class User {

    private $_dispatcher;

    public function  __construct(sfEventDispatcher $dispatcher) {
        $this->_dispatcher = $dispatcher;
    }


    public function login() {
```

```
        try {
            // logique d'identification
           throw new Exception("Méthode non-implémentée !");
        } catch (Exception $e) {
            $event = new sfEvent($this, 'log', array('summary'
 => 'login failed', 'exception' => $e));
            $this->_dispatcher->notify($event);
        }
    }
}

class ErrorLog {

    private $_dispatcher;

    public function __construct(sfEventDispatcher $dispatcher) {
        $this->_dispatcher = $dispatcher;

        // connexion de la classe aux composants
        $this->_dispatcher->connect('log', array($this, 'addEntry'));
    }

    public function addEntry(sfEvent $event) {

        $parameters = $event->getParameters();
        $summary = $parameters['summary'];
        $message = $parameters['exception']->getMessage();
        $class = get_class($event->getSubject());

        echo "[" . date('d/m/Y - H:i') . '] - ' . $summary .
 ' - '. $message . ' (happened on a ' . $class . ' object)';
    }
}

$log = new ErrorLog($dispatcher);
$user = new User($dispatcher);

$user->login();
// produit l'affichage de :
// [DATE - HEURE] - login failed - Méthode non-implémentée !
 (happened on a User object)
```

Voici quels sont les points les plus importants, et pouvant nécessiter quelques explications :

```
public function __construct(sfEventDispatcher $dispatcher)
```

Nous passons aux constructeurs des classes concernées une même instance de sfEventDispatcher. Ceci est bien sûr essentiel, car c'est lui qui enregistrera les connexions entre les notifications et les « listeners » (méthodes à exécuter lorsqu'une notification est émise).

```
$event = new sfEvent($this, 'log', array('summary' => 'login
failed', 'exception' => $e));
```

La création d'un objet sfEvent, qui sera ensuite transmis aux « listeners » permet de faire communiquer les composants. Le premier paramètre ($this) est le sujet de l'événement, c'est-à-dire l'objet à partir duquel l'événement est émis. Le sujet étant un objet, il est passé par référence, et pourra de fait être modifié par les « listeners ». Vient ensuite le nom de l'événement, qui servira de clé pour la connexion de « listeners ». Enfin, un tableau associatif libre peut être également transmis. Notez que sfEvent dispose également d'une méthode setReturnValue() qui permet de préciser dans un événement quelle valeur va retourner la méthode qui l'a émis.

```
$this->_dispatcher->notify($event);
```

émet à proprement parler un événement. Tous les « listeners » associés vont être exécutés.

```
public function addEntry(sfEvent $event)
```

Le « listener » de notre exemple, qui attend donc explicitement un objet sfEvent. Bien entendu, vous pouvez étendre la classe sfEvent, et émettre un événement étendu plutôt que celui d'origine si vous souhaitez ajouter des fonctionnalités particulières à vos événements.

CONCLUSION

La place me manque pour parler en détail de certaines subtilités de la librairie, comme la méthode d'émission alternative notifyUntil(), permettant de stopper l'exécution des « listeners » si l'un d'entre eux retourne un booléen « true » par exemple, ou encore la méthode filter(), prévue pour filtrer des chaînes de caractères au travers d'une séquence de filtres. Vous trouverez cependant toutes les ressources nécessaires sur le site officiel. J'ajouterai juste un dernier mot pour signaler les plus de l'usage de PHP 5.3 avec sfEventDispatcher : la possibilité de connecter des closures (grâce aux fonctions anonymes) plutôt que des « callables » (méthodes ou fonctions), ainsi qu'un gain considérable en performances lors de l'exécution de ces derniers. Bref, encore de bonnes raisons de considérer la mise à jour !
Site officiel : http://components.symfony-project.org/event-dispatcher/

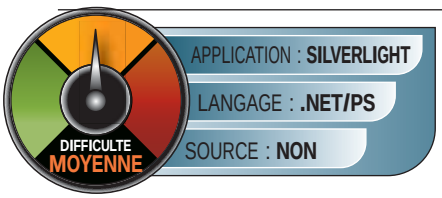
Gauthier **Delamarre**

Comment ne pas détruire la souplesse du mécanisme !

La connexion d'actions à divers événements d'une application offre tant de possibilités que l'on pourrait être tenté d'en faire un peu trop. Or, dès lors que l'on connecte plusieurs traitements à un même événement se pose la question de l'ordre dans lequel seront exécutés ces divers traitements.
Par défaut, les « listeners » seront exécutés les uns après les autres, dans le même ordre que celui de leur connexion à l'événement. Ce qui a pour funeste conséquence de corrompre le workflow si par malheur deux « listeners » supposés s'exécuter l'un après l'autre ne sont pas enregistrés dans le bon ordre.
Cette situation est en contradiction totale avec l'objectif de découplage maximal que vise la librairie. Nous avons débattu de cette problématique avec Fabien Potencier, concepteur et auteur de sfEventDispatcher, pour qui deux stratégies sont à privilégier pour se prémunir contre les risques liés à l'ordonnancement de l'exécution des « listeners » :
– Cela peut paraître trivial, mais tâcher de concevoir son application et ses plugins de sorte qu'aucune dépendance n'existe entre eux reste la solution la plus efficace.
Dans l'hypothèse où vous vous retrouvez contraint de créer ce type de dépendances, choisissez alors de chaîner vos événements plutôt que de compter sur l'ordre de connexion. En résumé, plutôt que de connecter le plugin A et le plugin B sur l'événement X, connectez le plugin A sur l'événement X et le plugin B sur le plugin A. Et pour les situations les plus délicates, dans lesquelles par exemple B doit s'exécuter après A quand celui-ci est présent, mais qu'il ne l'est pas toujours, créez un « méta-plugin », connectez-le sur le composant, pour connecter vos plugins dessus, dans l'ordre qui vous convient.

PoshBoard : piloter Silverlight avec PowerShell

PoshBoard est un portail IT modulaire open source basé sur PowerShell et Silverlight. L'idée de ce projet est de permettre à des administrateurs et ingénieurs système de piloter et monitorer tout type d'infrastructure en générant des interfaces Silverlight.



PoshBoard intègre la plupart des contrôles « standards » Silverlight : TextBox, TextBLock, ComboBox, ListBox..., ainsi que d'autres contrôles open source tels que Visifire (génération de graphiques) AgDatagrid (DataGrid avec des fonctionnalités avancées).

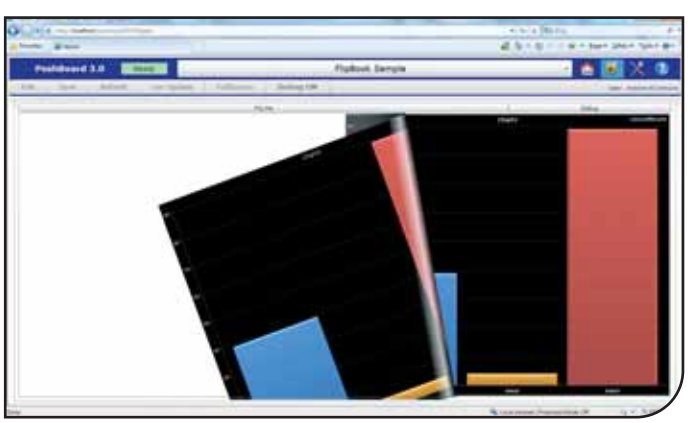
Nous utilisons PoshBoard chez Nelite en tant que solution de complément à nos offres d'intégration de solutions infrastructure, notamment autour de la virtualisation où PoshBoard permet de mettre en place rapidement un système de gestion de machines virtuelles ou d'analyses de charges des serveurs.

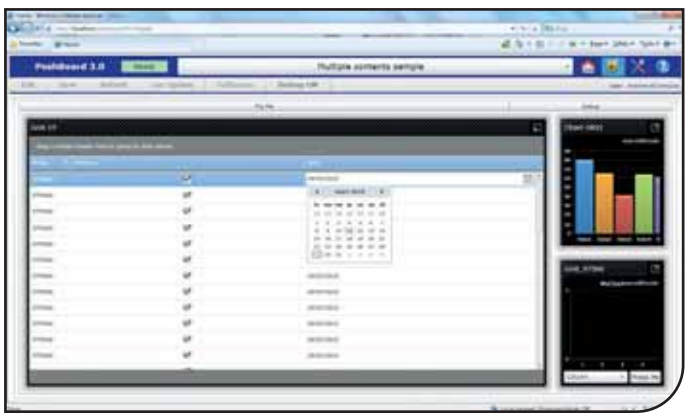
Tous les scripts PowerShell existants peuvent être adaptés à PoshBoard, ceci permet de profiter d'un éventail très large de scripts, disponible sur internet pour la personnalisation de portails de nos clients, avec des modifications minimes pour les intégrer au portail.

PoshBoard est disponible gratuitement sur CodePlex : http://poshboard.codeplex.com. Il existe aussi un site dédié avec vidéos, tutoriaux et forums : http://www.poshboard.com

L'idée de départ est de permettre aux IT de créer des interfaces riches en Silverlight sans avoir à connaître autre chose qu'un langage de scripting. Le choix s'est naturellement porté sur PowerShell, car PowerShell est le nouveau langage de scripting de référence pour les infrastructures Microsoft et est basé sur .NET, ce qui permet de créer un pont avec Silverlight. Mais si PowerShell comme Silverlight savent tous deux manipuler des objets .NET, développer un portail permettant l'interaction des deux a soulevé trois contraintes principales :

- Comment exposer PowerShell à Silverlight, Silverlight tournant dans un environnement sandboxé et n'ayant pas un accès direct aux composants du système hôte ou d'un système distant ?
- Comment transformer le résultat d'un script en contrôle Silverlight, sans pour autant devoir apprendre le fonctionnement et les spécificités de Silverlight pour un administrateur système ?





- Comment gérer la sécurité et l'authentification des utilisateurs ayant accès au portail, et donc éviter les utilisations frauduleuses (et potentiellement dangereuses) des scripts hébergés ?

CONNECTIVITÉ SILVERLIGHT ET POWERSHELL

Si PowerShell peut manipuler tout type d'objet et API .NET, il ne peut directement générer des contrôles Silverlight, qui ne font pas partie à proprement parler du Framework .NET mais en sont un dérivé. Silverlight de son côté ne peut invoquer directement les API PowerShell, et l'hébergement du langage n'est pas possible directement au sein d'une application Silverlight. Pour résoudre cette problématique, PoshBoard s'appuie sur l'utilisation d'un service web WCF et d'un ensemble de classes partagées pour établir la communication entre Silverlight et PowerShell.[Fig.1]

Ces classes partagées servent de contrat de communication via WCF entre les deux langages. Quand l'utilisateur affiche une page ou invoque un événement d'un contrôle, Silverlight appelle le service WCF en fournissant le script à exécuter. WCF instancie un runspace PowerShell et récupère le ou les objets renvoyés par le script et les retourne à Silverlight. L'hébergement de PowerShell au sein d'une application .NET est très simple. Voici l'extrait de code qui permet d'exécuter un script dans une application .NET :

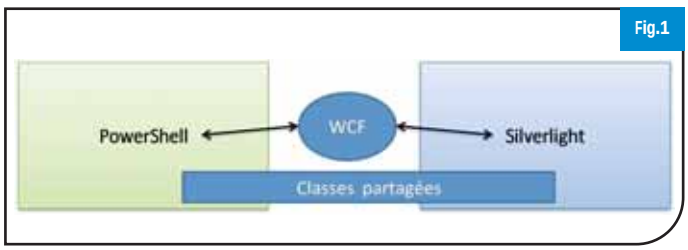


Fig.1

```
Runspace runspace = RunspaceFactory.CreateRunspace();
runspace.Open();
Pipeline pipeline = runspace.CreatePipeline();
pipeline.Commands.AddScript(code);
Collection<PSObject> results = pipeline.Invoke();
pipeline.Stop();
```

On voit ici que PowerShell retourne une collection d'objet de type PSObject. Il s'agit de l'objet .NET décoré d'attributs spécifiques à l'encapsulation PowerShell. Pour récupérer l'objet encapsulé, il suffit de faire appel à la propriété BaseObject de l'objet PSObject pour retrouver l'objet original.

NOUVEAUTÉS DES API POWERSHELL 2.0

PowerShell 2.0 est disponible sur la majorité des systèmes clients et serveurs et apporte de nouvelles fonctionnalités de développement .NET autour de la solution. Les plus notables sont :

Remote Runspaces : cette fonctionnalité permet de créer des runspaces et d'exécuter du code PowerShell sur des machines distantes. En effet, l'un des principaux apports de PowerShell 2.0 est de pouvoir exécuter des commandes sur des machines distantes, ceci n'était pas directement possible avec PowerShell 1.0. Le SDK de PowerShell 2.0 propose aussi la gestion des éléments liés à l'exécution de code distant, comme la notion de « Jobs » (tâches envoyées à une session distante).

Constrained Runspaces : cette fonctionnalité permet de créer des runspaces « contraints » dans une application. Par défaut, toutes les commandes sont mises à disposition d'un utilisateur lorsque nous développons une application qui s'appuie sur PowerShell. Ceci peut poser des problèmes en termes de sécurité, certaines commandes pouvant avoir un impact fort sur les systèmes visés. Les runspaces contraints permettent de définir une liste de commandes pouvant être utilisées, ceci permettant donc de limiter l'hébergement de PowerShell au sein d'une application .NET aux seules commandes nécessaires.

Runspace Pool : à l'instar du ThreadPool disponible pour .NET, Le runspace Pool permet de créer à la demande des runspaces pour une exécution simultanée de commandes, l'API se chargeant de l'allocation et la destruction des runspaces.

TRANSFORMATION D'UN RÉSULTAT DE SCRIPT POWERSHELL EN RENDU SILVERLIGHT

Afin de simplifier et fluidifier la génération de contrôle Silverlight à partir de PowerShell, nous avons créé un modèle objet simple et polyvalent, permettant à partir d'un type unique d'objet de déclarer l'ensemble des propriétés et méthodes nécessaires à la création des différents contrôles Silverlight.

L'objet principal utilisé (PBElement), regroupe à la fois des propriétés communes à l'ensemble des contrôles Silverlight (Colonne, ligne, hauteur, largeur, type, etc....) ainsi qu'une propriété spéciale « Properties » de type Dictionnaire permettant de stocker des propriétés spécifiques à un contrôle Silverlight (Exemple : la taille de la fonte pour une boîte de texte, le titre d'un graphique, etc.) [Fig.2]

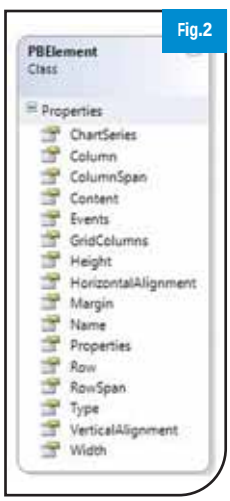


Fig.2

L'utilisation d'un seul type d'objet dans la transaction permet de simplifier la communication WCF et la modélisation côté PowerShell, par l'utilisation de types basiques (String, int, double, Boolean...)

Côté PowerShell, nous avons développé un Snapin en C# (ensemble de commandes) qui permet de modéliser tous les types de contrôles Silverlight prévus dans PoshBoard.

Exemple : New-PBButton pour la création d'un bouton, New-PBChart pour la création d'un graphique.

Chacune de ces commandes renvoie un objet défini dans notre classe partagée qui peut donc être transmis directement à Silverlight pour la génération du rendu. [Fig.3]

Silverlight interprète ensuite les objets retournés par le service pour construire le rendu et enregistrer les évènements. Dans PoshBoard, l'intégralité des contrôles Silverlight sont générés dynamiquement par le code-behind.

Deux classes sont utilisées pour la création de contrôles Silverlight : SilverlightElementBuilder.cs et SilverlightElementUpdater.cs

La première permet de générer les contrôles, la seconde, de les mettre à jour. PoshBoard propose un mode « live update » qui permet de rafraîchir en continu le résultat des scripts en les invoquant. Au lieu de recréer le contrôle, la classe SilverlightElementUpbdater permet de modifier uniquement les données, quel que soit le type de contrôle : Graphique, DataGrid, TextBox, etc.

En fonction du type d'objet retourné par le script, PoshBoard fait appel à la classe correspondante et définit ce qui doit être mis à jour dans le contrôle.

La prochaine version intégrera un système de Plug-in, permettant aux utilisateurs d'intégrer de nouveaux contrôles Silverlight sans avoir à modifier le code source.

SÉCURISATION DES TRANSACTIONS ENTRE SILVERLIGHT ET POWERSHELL

Méthode d'authentification Silverlight – ASP.NET

La sécurité est un point particulièrement important pour une solution d'administration des infrastructures systèmes. Il est en effet important de savoir qui exécute une action, et d'autoriser ou non des tâches en s'appuyant sur les droits définis dans l'annuaire d'entreprise, Active Directory dans notre cas.

C'est pourquoi il est important que PoshBoard s'appuie sur l'authentification Kerberos, d'autant que PoshBoard à pour vocation première d'être un portail intranet pour administrateurs et ingénieurs systèmes. Afin de répondre à ce besoin, nous avons utilisé 2 modes

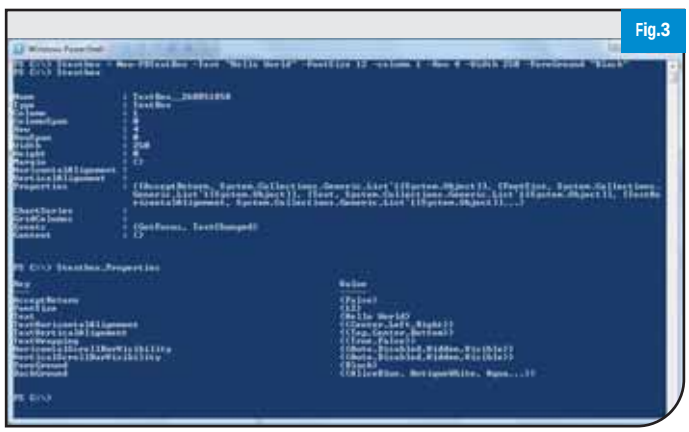


Fig.3

d'authentification dans PoshBoard :

• Authentification intégrée Windows pour la partie ASP.NET

La partie ASP.NET est configurée en authentification Windows, avec un Deny explicite sur les utilisateurs anonymes :

```
<authentication mode="Windows"/>
<authorization>
  <deny users="?"/>
</authorization>
```

Nous profitons ainsi des mécaniques d'authentification mises à disposition par IIS et le navigateur.

• Authentification anonyme pour la partie Silverlight

L'authentification pour la couche de transport http côté Silverlight est, quant à elle, définie en anonyme, car nous n'utilisons pas directement le composant Silverlight pour exécuter les scripts PowerShell.

```
<httpTransport proxyAuthenticationScheme="Anonymous" >
```

Nous utilisons le mode de compatibilité ASP.NET qui nous permet d'utiliser l'identité ASP.NET lors de l'appel au service :

```
<serviceHostingEnvironment aspNetCompatibilityEnabled="true"/>
```

C'est ensuite au niveau du service WCF que l'identité de l'utilisateur est vérifiée à chaque appel, en fonction de l'identité retournée par le navigateur :

```
System.Web.HttpContext.Current.User.Identity
```

Impersonification

Nous avons deux grands cas d'usage de l'identité dans l'utilisation de PoshBoard. Soit l'administrateur du portail souhaite utiliser un compte de service, soit l'identité de l'utilisateur pour l'exécution des scripts (délégation).

L'utilisation d'un compte de service pour l'exécution des scripts permet de donner l'accès à des utilisateurs n'ayant pas directement les droits sur les systèmes. Par exemple, ceci permet à un DSI d'avoir accès à des informations sur la santé d'un composant de l'infrastructure sans pour autant l'inscrire en tant qu'administrateur du composant.

Le second cas correspond à l'utilisation du portail par les administrateurs systèmes : utiliser les droits effectifs d'un administrateur permet d'utiliser un seul script pour tout le monde, les droits et la visibilité des objets dépendant alors directement des droits du compte. PoshBoard permet pour chaque page du portail de définir les accès, mais aussi le mode d'impersonificiation pour chaque groupe ou utilisateur inscrit. Ceci permet, au choix, d'utiliser le compte de service ou l'identité de l'utilisateur pour l'exécution d'un script. Un switch est présent dans le code, permettant, si la page est en mode impersonification, d'utiliser le contexte de l'identité de l'utilisateur :

```
WindowsImpersonationContext ctx = null;
WindowsIdentity currentUser =  (WindowsIdentity)HttpContext.
Current.User.Identity;
if (impersonation == true)
{
ctx = currentUser.Impersonate();
}
```

Une nuance ici : ceci permet en l'état uniquement d'impersonifier l'exécution de script. Si l'on souhaite effectivement déléguer l'exécution de code sur une machine distante, il faut pour cela, comme pour toute application web, approuver l'ordinateur pour la délégation dans Active Directory :

http://technet.microsoft.com/en-us/library/cc738491(WS.10).aspx

Fig.4

Développement de PoshBoard 3.0

La version stable disponible est la version 2.5. La version 3.0 est disponible en Bêta et apporte les fonctionnalités suivantes :

- Prise en charge d'un mode « page » : ce mode permet d'intégrer plusieurs composants (appelé widgets) sur une même page, en choisissant des modèles de rendus prédéfinis : DashBoard, liste de contrôles, « FlipBook », etc.
- MultiTouch : un des modes Page introduit la gestion du MultiTouch proposée par Silverlight 3.0
- Debugger : un débuggeur permet de voir en temps réel quel script est exécuté, et quelles modifications sont apportées au script de base par l'interface [Fig.4]

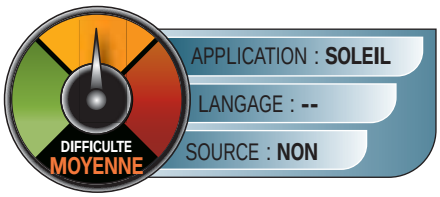
La version 3.0 va être pour sa version finale portée sur Silverlight 4.0 et C# 4.0, afin de profiter notamment de la gestion des objets dynamiques. Cette nouvelle fonctionnalité est très importante pour PoshBoard où la majorité des contrôles et des événements sont dynamiques. Cette nouveauté de C# 4 va grandement simplifier le modèle de génération et de mise à jour des contrôles.

Les mises à jour de Silverlight 4 concernant le mode « Out Of Browser » vont permettre un portage plus facile de PoshBoard sur ce mode, qui offre plusieurs avantages, notamment en termes de déploiement. Le mode Multitouch va être grandement étendu pour prendre en compte plus de gestes, et proposer aussi une nouvelle librairie de reconnaissance gestuelle pour essayer d'innover sur ce terrain. Ceci fera l'objet prochainement d'une démo technique en vidéo sur le site web de PoshBoard. La gestion des événements va aussi être revue pour rendre le développement d'événements plus simple pour les utilisateurs, avec la mise en place d'un designer plus intuitif. Enfin, nous travaillons actuellement sur un système communautaire autour de PoshBoard pour permettre l'échange de composants et de scripts entre les utilisateurs, avec une bibliothèque en ligne accessible depuis PoshBoard et un système de classification et de notation.

■ Antoine **Habert**

A la découverte d'OpenSolaris

Pour le geek, le monde de l'Open Source c'est la caverne d'Ali Baba. Bien souvent, nous y avons ouvert le coffre au trésor Linux. Aujourd'hui nous découvrons un autre joyau : le système d'exploitation OpenSolaris.



Il est bien loin le temps du système d'exploitation unique et médiocre pour nos PC. Non seulement le système "par défaut" a fait des progrès considérables, mais les systèmes alternatifs sont désormais nombreux et de qualité : Linux, FreeBSD, OpenBSD, OpenSolaris, etc. Nous avons souvent parlé de Linux dans les pages de Programmez! Nous souhaitons vous emmener aujourd'hui à la découverte d'OpenSolaris, un système dont la réputation n'est pour l'instant pas à la hauteur de ses capacités.

1 OPENSOLARIS EN QUELQUES MOTS

OpenSolaris est un projet lancé par Sun Microsystems. L'idée est de rassembler une communauté pour créer et tester les outils destinés à faire partie de Solaris, le système UNIX de Sun dont la réputation de solidité n'est plus à faire. OpenSolaris étant, et devant rester, un logiciel libre. OpenSolaris est donc très similaire à Solaris. Il en conserve le noyau, le support réseau, les bibliothèques et le jeu de commandes. OpenSolaris est donc, comme Solaris, un système de la famille des UNIX BSD (Berkeley Software Distribution), comme le sont aussi FreeBSD, OpenBSD, NetBSD, Dragonfly BSD, etc. Nous nous intéresserons d'ailleurs à certains de ces systèmes prochainement. En attendant nous nous concentrons sur OpenSolaris, dont la version 2009.06, qui sert de support à cet article, est un système formidable, vraiment abouti, et innovant. OpenSolaris présente de nombreux atouts qui le démarquent de Linux. Ainsi, on reproche parfois à Linux, ou plutôt aux diverses distributions Linux, un manque d'homogénéité, ou de cohérence entre elles, en raison de l'absence d'une ligne directrice unique dans l'univers Linux.

Avec OpenSolaris, ce problème ne se pose pas, et OpenSolaris propose ce qu'Ubuntu tente de faire (fort bien d'ailleurs), à savoir fournir un système bien léché, très facile à installer, configurer, et utiliser. Au-delà de cela, OpenSolaris offre des particularités vraiment intéressantes. D'abord un système évolué de gestion des droits des utilisateurs, sur lequel nous reviendrons plus loin, ZFS, un système de fichiers ébouriffant, DTrace, un mécanisme de débogage en temps réel très supérieur au PTrace de Linux (cf. Programmez! 123), Zones, une sorte de mécanisme de bacs à sable, qui, à partir d'une seule instance du noyau, permet de fournir à des applications des environnements d'exécution isolés et des droits particuliers. C'est encore SMF, un système de gestion de services (ou démons) qui remplace avantageusement l'ancien System V qui continue toutefois d'être supporté. Bien sûr, les applications classiques de l'Open Source comme Firefox ou Thunderbird tournent sous OpenSolaris, et la mouture 2009.6 est très riche en drivers.

2 MATÉRIEL ET PRÉ-REQUIS

OpenSolaris étant un système UNIX, nous supposons que le lecteur en est familier car il n'est pas question de présenter UNIX ici. Plus concrètement, nous considérons que le lecteur est familier de Linux que nous mettons en perspective avec OpenSolaris. Pour installer OpenSolaris, il est nécessaire de télécharger l'image ISO du CD Live, puis démarrer sur ce CD. Une fois dans le bureau, il suffit de cliquer sur l'icône de l'utilitaire d'installation pour lancer la procédure. Le plus simple est d'avoir un disque dédié au système, ou encore de virtualiser. Cet article à pour base un OpenSolaris virtualisé sous VirtualBox (cf.Programmez 124). VirtualBox est gratuit et peut être téléchargé à www.virtualbox.org. L'utiliser fait économiser un disque et éviter tous dégâts éventuels sur la machine hôte. Cela fait même économiser la gravure d'un CD car VirtualBox sait booter un système depuis une image ISO.

3 INSTALLATION

La procédure est toute simple. On commence par booter sur le CD Live de Solaris. Vous allez alors arriver dans le bureau qui n'est autre que Gnome. Là, cliquez sur l'icône de l'outil d'installation. Celui-ci vous demandera sur quel disque installer, et si vous souhaitez par-

Fig.1

Premiers pas pour l'installation d'Open Solaris, avec le partitionnement du disque.

Fig.2

Création des premiers utilisateurs, lors de la procédure d'installation

titionner le disque ou l'utiliser en entier [Fig.1]. Suivront ensuite des dialogues vous demandant de spécifier le fuseau horaire et la langue, ce qui est une formalité. Vient ensuite le dialogue de création des premiers utilisateurs [Fig.2]. Ce dialogue mérite toute notre attention car il demande à la fois le mot de passe de root et un nom d'utilisateur "ordinaire", par exemple "fred" ainsi que son mot de passe. On se croirait dans une procédure d'installation de Linux. Mais des différences sont pour l'instant cachées que nous découvrirons un peu plus loin. Ainsi, en l'état actuel des choses, root n'est pas vraiment le super-utilisateur que l'on croit, et "fred" est plus qu'un utilisateur ordinaire. Poursuivez la procédure jusqu'à son terme, ce qui ne devrait poser aucune difficulté, et redémarrez sur votre OpenSolaris tout neuf.

4 LA GESTION DES UTILISATEURS ET DES DROITS

Une fois le système démarré, vous arrivez à l'interface de connexion. [Fig.3]. Et là si vous essayez de vous connecter en tant que root, une surprise vous attend: la connexion est purement et simplement rejetée. [Fig.4]. Et on vous informe que les rôles ne peuvent être assumés que par les utilisateurs autorisés. Que root ne soit pas autorisé à faire quelque chose désarçonne, et on se demande comment il va être possible d'effectuer la moindre configuration du système. La réponse est : en se connectant en tant que fred (ou l'utilisateur normal que vous aurez créé sous un nom vous convenant mieux) et en comprenant les principes d'une des spécificités d'OpenSolaris: RBAC. Role-Based Access Control, ou RBAC, est une alternative évoluée et plus sécurisée par rapport au simple super-utilisateur qui a tous les droits, notamment celui de tout casser :-) RBAC permet d'attribuer aux utilisateurs des rôles dont les pouvoirs sont définis à travers des profils.

Par exemple un utilisateur classique peut se voir attribuer, en plus de ses droits standards, et par exemple, le droit d'administrer le réseau et / ou de définir des jobs cron, et cela seulement. Sous OpenSolaris, les utilisateurs sont de deux types, soit le type normal, soit le type rôle qui en tant que tel n'est pas un utilisateur particulier. Par défaut, la procédure d'installation a créé root comme un rôle et non comme un utilisateur, et c'est ce qui explique le message d'erreur à la connexion : root n'est qu'un rôle, pas un utilisateur autorisé à endosser le rôle de root.

5 ACQUÉRIR LES DROITS DU SUPER-UTILISATEUR

Tout cela est fort bien, mais alors comment fait-on pour configurer quelque chose ? On commence par se connecter à l'utilisateur fred qui peut endosser le rôle de root, donc ses droits. Par exemple si dans un terminal on donne :

```
su root
```

Le système demandera le mot de passe de root spécifié lors de l'installation. Alors fred devient un super-utilisateur. Voilà qui nous rassure, mais sous OpenSolaris, on ne procède généralement pas ainsi. On donnera plutôt, toujours dans le terminal, cette commande :

```
pfexec bash
```

Cette commande, qui est un peu l'alter ego du sudo d'Ubuntu, lance le shell bash sous le premier profil trouvé dans une petite base de données de gestion des utilisateurs.

Il se trouve que fred s'est vu attribuer, lors de l'installation, le profil Primary Administrator, ce qui lui donne accès à toutes les tâches administratives.

Donc, depuis le Shell ainsi lancé nous pouvons tout faire. Il existe un outil graphique permettant de gérer les utilisateurs depuis Gnome. Cet outil permet de lister les profils existants et de les attribuer ou retirer à un utilisateur.

De même pour les rôles. L'illustration [Fig.5] montre que le profil Primary Administrator est bien attribué à fred. Ne décochez pas la case par erreur, ce serait … assez ennuyeux :-) Si cette approche ne vous convient pas, il est possible de modifier root pour en faire un super-utilisateur traditionnel. Les données présentées par l'outil graphique de gestion des utilisateurs résident dans le fichier / etc / user_attr. Regardons :

```
cat /etc/user_attr

adm::::profiles=Log Management
dladm::::auths=solaris.smf.manage.wpa,solaris.smf.modify
lp::::profiles=Printer Management
postgres::::type=role;profiles=Postgres Administration,All
root::::type=role;auths=solaris.*,solaris.grant;profiles=All;
lock_after_retries=no;clearance=admin_high;min_label=admin_low
zfssnap::::type=role;auths=solaris.smf.manage.zfs-auto-snapshot
;profiles=ZFS File System Management
fred::::profiles=Primary Administrator;roles=root
```

Nous voyons clairement que root est un rôle. Pour faire de root un utilisateur classique, nous procédons ainsi :

```
pfexec bash
usermod -K type=normal root
```

Fig.3

L'interface de connexion d'OpenSolaris

Fig.4

Aussi surprenant qu'il y paraisse, impossible de se connecter en tant que root après l'installation.

Après quoi, dans /etc/user_attr nous trouvons

```
root::::type=normal;auths=solaris.*,solaris.grant;profiles=All
;lock_after_retries=no;clearance=admin_high;min_label=admin_low
```

et vous pourrez désormais vous connecter directement en tant que super-utilisateur depuis l'interface de connexion. Nous n'avons fait qu'effleurer RBAC qui mériterait plus qu'un article entier à lui seul. Nous invitons le lecteur à approfondir ses connaissances avec la documentation.

6 LE TOUR DU PROPRIÉTAIRE

Où trouver cette documentation ? Comme tout bon système UNIX propre sur lui, OpenSolaris propose les classiques pages de man, qui sont en principe stockées sous /usr/share/man. Cependant, à l'heure d'Internet on préférera sans doute se tourner vers une documentation en ligne très complète (et qui contient elle aussi les pages de man :). Cette documentation réside à http://dlc.sun.com/osol/docs/content/2009.06/. Si nous jetons un coup d'œil à l'arborescence du système de fichier, nous verrons que celle-ci ressemble à celles que l'on trouve sous Linux, avec de petites différences toutefois :

Linux	OpenSolaris
/home	/export/home
/tmp	/var/tmp
	/var/log /var/log
	/var/adm
	/usr/adm
/sys et /dev	/devices et /dev

Sans ces petites différences et bizarreries, nous ne serions pas dans le monde d'UNIX :-) Ainsi vous pourrez constater que /home existe aussi, mais néanmoins c'est bien dans /export/home que vous trouverez les répertoires des utilisateurs. Les fichiers du super-utilisateur résident normalement dans /root. Même remarque pour /tmp. Il existe bel et bien, est utilisé, mais OpenSolaris lui prèfère nettement /var/tmp/. Dans /var/log, vous trouverez syslog, le journal système. En revanche, messages, le fichier contenant les messages émis par le noyau au démarrage se situe dans /var/adm. Quand à /usr/adm, ce n'est qu'un lien symbolique vers le précédent. Hormis les répertoires, OpenSolaris est riche en fichiers de configuration. Là encore, il y a de petites différences avec Linux, notamment :

Linux	OpenSolaris
/etc/fstab	/etc/vfstab
/etc/exports	/etc/dfs/dfstab
/etc/ntp.conf	/etc/inet/ntp.conf
/etc/inetd.conf	/etc/inet/inetd.conf
/etc/networks	/etc/inet/networks

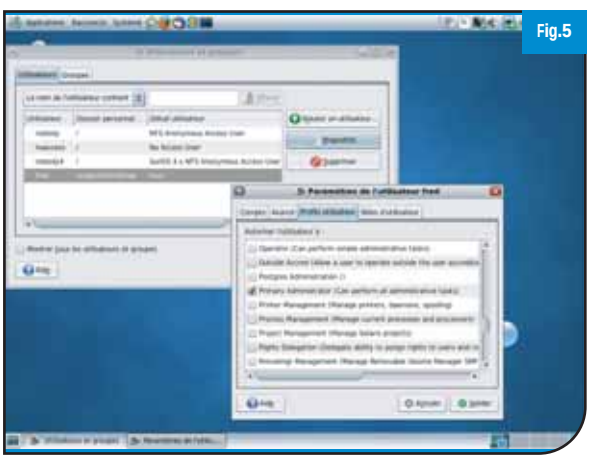
Lors de la prise de contact avec le système, on prendra le temps de visiter le menu Système|Administration du bureau Gnome. Deux outils retiendront particulièrement l'attention. D'abord l'outil de gestion du réseau, qui permet de configurer manuellement celui-ci. En principe cette opération n'est pas nécessaire, car le service Network Auto Magic a déjà trouvé et configuré votre interface réseau. Si ce n'est pas le cas, où si sous VirtualBox avec un hôte Windows, le système à du mal à trouver le DNS, l'outil de gestion vous permettra de le configurer manuellement [Fig.6].

7 ZFS

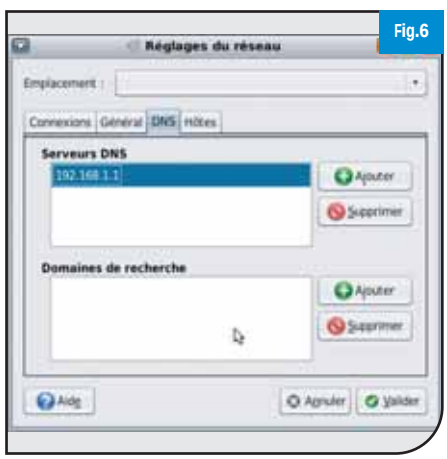
Le deuxième outil à remarquer est dénommé curseur temporel en bon français, ou time slider. Cet outil est une killer app qui fonctionne de concert avec ZFS, le système fichier d'OpenSolaris. ZFS c'est du costaud, avec une taille maximale de fichier de 16 exbioctets (soit 16 fois 1,152,921,504,606,846,976 octets) ! Ce qui devrait couvrir assez largement nos besoins à venir :-) Mais ZFS c'est aussi la possibilité de générer des instantanés du système de fichier à intervalles de temps réguliers, ceci aussi bien pour une seule unité de stockage que pour un RAID. Avec cette fonctionnalité nous avons quelque chose de similaire à la restauration système Windows introduite avec Windows XP, et qui manque à Linux. Vous l'avez compris, c'est ici que l'outil "Curseur Temporel" intervient [Fig.7]. Les options avancées permettent de détailler quels répertoires seront inclus dans les sauvegardes périodiques. ZFS est un univers et nous invitons le lecteur à consulter sa page de man (man zfs) pour en apprendre davantage.

8 INSTALLER DES OUTILS DE DÉVELOPPEMENT

Le lecteur de Programmez! n'appréciera OpenSolaris que s'il peut y installer et utiliser des outils de développement. Par défaut, Oracle oblige, Java est présent :-) Sont encore présents des langages de scripts comme Python, mais pas d'outils tels que Netbeans côté Java ou gcc côté compilateurs C/C++. À l'instar des distributions



Le premier utilisateur du système est doté du profil Primary Administrator.



Réglage manuel du DNS pour l'interface réseau.

Définition d'instantanés des données, avec le Curseur Temporel

Linux, OpenSolaris vient avec un gestionnaire de paquetages, et offre une interface graphique au-dessus de celui-ci, accessible depuis le menu Système|Administration de Gnome. Avec lui, c'est un jeu d'enfant d'installer par exemple Netbeans, ou, comme illustré [Fig.8], sunstudio, qui contient les outils de développement pour C / C++. Pour en apprendre plus au sujet des paquetages on consultera les pages de man des outils pkg, pkgadd, pkgadm, pkgchk et paginfo principalement.

9 GÉRER LES SERVICES

Historiquement, OpenSolaris lançait les services au démarrage du système avec des scripts System V exécutés en fonction de runlevels ou niveaux de démarrage. Linux fonctionne sous ce principe. Les scripts devant être exécutés selon le niveau sont rangés dans des répertoires /etc/rc0.d, /etc/rc1.d, etc. OpenSolaris 2009.6 reste compatible avec System V et vous pouvez continuer de déposer vos scripts dans les répertoires dédiés. Cependant il offre un nouvel outil, SMF, pour Service Management Facility, que nous avons tout intérêt à utiliser. Là encore il existe une interface graphique accessible depuis le menu Système|Administration de Gnome. [Fig.9] Toutefois, cette interface n'offre pas suffisamment d'informations pour une gestion fine et il est ici intéressant de prendre le temps de se familiariser avec les deux outils en ligne de commandes qui permettent de travailler avec SMF: svcs et svcadm. La première, svcs, est un outil de diagnostic. Ainsi avec :

```
svcs -a
```

Nous pouvons tout connaître de l'état des services de notre système. Voici un extrait de la sortie émise par cette commande sur le système de votre serviteur :

```
STATE          STIME    FMRI
legacy_run     14:14:42 lrc:/etc/rc2_d/S20sysetup
legacy_run     14:14:43 lrc:/etc/rc2_d/S47pppd
disabled       14:13:39 svc:/system/device/mpxio-upgrade:default
disabled       14:13:40 svc:/system/svc/global:default
online         14:13:38 svc:/system/svc/restarter:default
online         14:13:40 svc:/network/loopback:default
offline        14:13:46 svc:/system/console-login:vt2
offline        14:13:46 svc:/system/console-login:vt3
```

L'état *legacy_run* signale un service démarré à l'ancienne par un script System V. Un état *disabled* signale un service présent sur le système mais non activé. Un état *online* signale un service fonctionnant normalement, et un état *offline* signale un service qui n'a pas démarré ou s'est arrêté en raison d'un problème. Vous l'avez compris l'état offline est anormal et mérite toute votre attention. Manipulons un peu ces deux commandes. Par exemple ssh est-il présent et activé sur notre système ?

```
svcs -a | grep -i ssh

online         14:14:42 svc:/network/ssh:default
```

La réponse est donc oui. Très bien, nous voulons le désactiver:

```
svcadm disable svc:/network/ssh:default
```

Maintenant la commande svcs -a | grep -i ssh émet :

```
disabled       14:14:42 svc:/network/ssh:default
```

Le service est bien désactivé, et cette désactivation est persistante d'un démarrage du système à l'autre. Nous devons donc réactiver le service si nous souhaitons l'utiliser :

```
svcadm enable svc:/network/ssh:default
```

10 EN GUISE DE CONCLUSION

Nous avons fait un petit peu connaissance avec OpenSolaris. Les choses deviendront passionnantes quand nous programmerons sous ce système, prochainement dans les pages de Programmez! :-) En attendant stoppons le système proprement, en donnant dans une console :

```
init 5
```

Les runlevels n'ont pas perdu tous leurs droits :-) A bientôt.

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

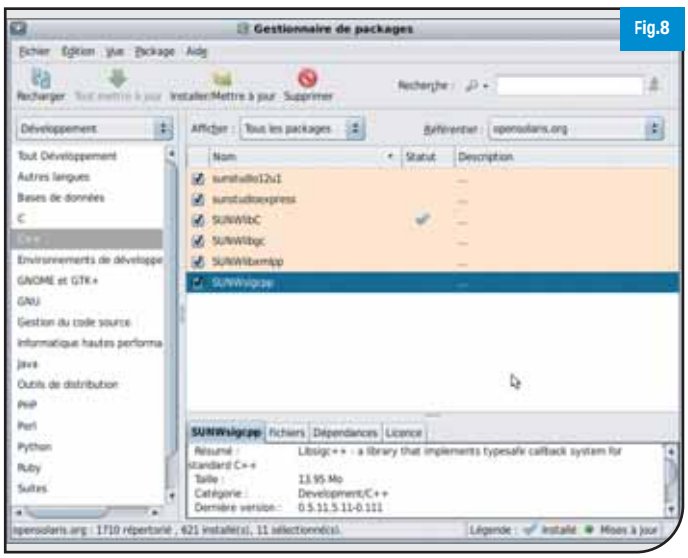


Fig.8

Installation des outils de développement C/C++ au moyen du gestionnaire de paquetage.

Fig.9

L'interface graphique au-dessus de SMF.

DÉVELOPPEZ VOTRE SAVOIR-FAIRE

*11 numéros par an : 49 €**

Economisez 16,45 €*

**Tarif France métropolitaine*

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 !

Choisissez votre formule

- **Abonnement 1 an au magazine : 49 €** (au lieu de 65,45 € tarif au numéro) *Tarif France métropolitaine*
- **Abonnement Intégral : 1 an au magazine + archives sur Internet et PDF : 59 €** *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*

+ Abonnement INTÉGRAL

ACCÈS ILLIMITÉ aux ARCHIVES du MAGAZINE pour 0,84€ par mois !

Cette option est réservée aux abonnés pour 1 an au magazine, quel que soit le type d'abonnement (Standard, Numérique, Etudiant). Le prix de leur abonnement normal est majoré de 10 € (prix identique pour toutes zones géographiques). Pendant la durée de leur abonnement, ils ont ainsi accès, en supplément, à tous les anciens numéros et articles/dossiers parus.

OUI, *je m'abonne* ***Vous pouvez vous abonner en ligne et trouver tous les tarifs*** www.programmez.com

PROGRAMMEZ

☐ **Abonnement 1 an au magazine : 49 €** (au lieu de 65,45 € tarif au numéro) *Tarif France métropolitaine*

☐ **Abonnement Intégral : 1 an au magazine + archives sur Internet et PDF : 59 €** *Tarif France métropolitaine*

☐ **Abonnement Etudiant : 1 an au magazine : 39 €** (joindre copie carte étudiant) *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

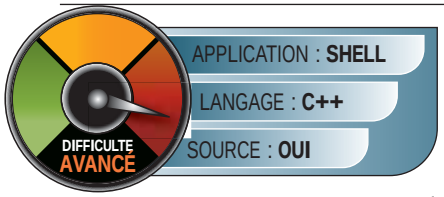
Offre limitée, valable jusqu'au 31 mai 2010

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.

PROG 130

Manipuler les bibliothèques du Shell de Windows 7

Windows 7 voit son Shell graphique enrichi d'un nouveau mécanisme de représentation de données de l'utilisateur : les bibliothèques. Voyons ensemble comment travailler avec ces nouvelles API



Avec Windows 7, toute l'interface utilisateur a été repensée. Nouvelle barre des tâches, nouvelles possibilités graphiques, apparition des rubans, etc. Nous nous intéressons aujourd'hui à une nouveauté qui concerne le Shell graphique de Windows 7: les bibliothèques. Comme son nom de Shell l'indique, le Shell de Windows permet de lancer des applications et de représenter les données présentes sur le disque. C'est ce que font tous les Shell, celui de Windows étant, on le sait, entièrement graphique.

Jusqu'ici Windows suggérait aux utilisateurs de ranger leurs données (fichier texte, images, vidéos, etc.) dans les répertoires prédéfinis bien connus "Mes Documents", "Mes Images", etc. Suggestion que les utilisateurs suivaient ou pas, mais qui, dans tous les cas, montrait vite ses limitations. Un seul répertoire prédéfini fourre-tout c'est trop peu, et lorsqu'on utilise de nombreux répertoires personnels différents, on en vient à vite à oublier dans quel répertoire est rangé tel ou tel fichier particulier. Windows 7 propose une nouvelle approche voulant à la fois éviter de mélanger les torchons et les serviettes, tout en donnant une vue complète de la lingerie. C'est le mécanisme des bibliothèques. Une bibliothèque Shell regroupe des répertoires différents de manière transparente (ou quasi transparente) pour l'utilisateur tout en lui présentant les données d'un bloc. Derrière les bibliothèques se trouve une nouvelle API que nous apprenons à manipuler.

1 NOTIONS SUCCINCTES DE PROGRAMMATION COM

De nombreuses parties constitutives de Windows sont écrites avec la technologie Microsoft COM. C'est le cas du Shell. Nous avons manipulé cette technologie de nombreuses fois au fil des pages de Programmez! Nous rappelons ici quelques notions de base minimales pour le lecteur qui découvrirait COM. COM est une technologie de composants logiciels qui implémentent des interfaces COM, interfaces qui sont assez similaires aux interfaces Java. Les composants COM sont instanciés au moyen d'API Windows, ou par le biais d'autres composants COM. Une instanciation de composant COM retourne un pointeur qui permet d'invoquer les méthodes du composant. En ce qui concerne la gestion des ressources, les composants détiennent un compteur de référence. Quand le compteur tombe à zéro, le système détruit le composant et libère ses ressources associées. C'est du devoir du programmeur d'incrémenter et décrémenter le compteur. Tâche fastidieuse s'il en est. Il est par ailleurs nécessaire de vérifier les codes de retour des méthodes COM. Tout l'ensemble conduit rapidement à un code imbuvable si écrit dans un style C. La librairie ATL de Visual Studio, l'IDE Microsoft, offre CComPtr, un pointeur intelligent, pour simplifier la gestion des ressources. Nous utiliserons nous aussi la puissance de C++ en proposant notre propre pointeur intelligent, et quelques fonctions auxiliaires. Notre pointeur intelligent, SmartCOMPtr, pourra être utilisé avec n'importe quel compilateur C++, contrairement à CComPtr. Un dernier mot : la programmation COM ressemble à la programmation objet, mais avec une différence notable. Les composants COM n'ont pas de constructeur. Les composants doivent donc être liés avec des données soit par le biais d'une API, soit par l'invocation d'une méthode a posteriori. Garder ce point à l'esprit permet d'éviter des erreurs agaçantes.

2 POUR SE FAIRE LA MAIN

Commençons par un exemple pour nous familiariser avec le Shell, COM et leur philosophie. Microsoft fournit à www.microsoft.com un SDK Windows 7 contenant fichiers en-têtes, librairies et documentation. Bien entendu vous devez installer ce SDK et vous devez configurer votre compilateur ou votre environnement de développement afin que ces en-têtes et librairies soient trouvés. Dans la documentation vous trouverez listées les nouvelles API du Shell Windows, autrement dit les nouvelles interfaces COM. Parmi elles, IFileDialog2, qui contient quelques méthodes supplémentaires par rapport à son prédécesseur IFileDialog. La documentation ne précise pas clairement qu'il est impossible d'instancier cette interface en tant que telle. (Pour les habitués de COM cela signifie que la constante CLSID_FileDialog2 n'existe pas). En fouillant la documentation, on finit par comprendre que l'on doit instancier soit IFileOpenDialog soit IFileSaveDialog, et qu'une instance de IFileDialog2 est obtenue à l'issue de l'opération. Voici un exemple, DemoIFileDialog, qui procède avec IFileOpenDialog (tous nos exemples sont disponibles sur notre site) :

```
#include <iostream>
using namespace std;

#include <COMUtils.h>
#include <shobjidl.h>


int main(int argc, char* argv[])
{
 try
 {
   COMInitializer ci;
   HRESULT hr;
   SmartCOMPtr<IFileDialog2> fd;
```

```
    // Créer indirectement une instance COM de IFileDialog2
    hr = ::CoCreateInstance(CLSID_FileOpenDialog,
                            NULL,
                            CLSCTX_INPROC_SERVER,
                            IID_IFileDialog2,
                            (void **)&fd);
    COMHelper::TestOk(hr);
    fd->Show(0);
  }
  catch(COMException ce)
  {
    std::cout << ce.raison() << std::endl;
    return EXIT_FAILURE;
  }
  return EXIT_SUCCESS;
}
```

Ce code ne pose aucune difficulté. On ne doit jamais oublier d'initialiser la plomberie COM avant toute utilisation. C'est cela que se charge d'effectuer notre classe auxiliaire COMInitializer. Une fois notre instance de IFileDialog2 obtenue, nous invoquons sa méthode Show, et nous voyons avec satisfaction la boîte de dialogue de fichiers du Shell apparaître sur notre écran [Fig.1]. Cette boîte de dialogue nous présente les librairies par défaut : Documents, Images, Musique, Vidéos. L'illustration nous montre encore que la librairie Documents est composée de deux répertoires, ou emplacements : Mes Documents et Documents publics.

3 LES RÉPERTOIRES PRÉDÉFINIS DU SHELL

Dans l'exemple précédent, si les librairies par défaut sont toutes apparentes, le dialogue se positionne systématiquement sur la librairie Documents. Une application de dessin, par exemple, préférera naturellement ouvrir le dialogue sur la librairie Images. Ceci s'obtient facilement. Dans l'en-tête Knownfolders.h se trouve une ribambelle de constantes identifiant des répertoires et librairies Shell prédéfinis. Ces constantes sont des GUID, des valeurs 128 bits, dont l'unicité est garantie. Une nouvelle API Windows 7, SHGetKnown FolderItem, permet d'instancier une interface IShellItem à partir de ces valeurs. IShellItem est l'interface COM qui décrit les entités constitutives du Shell (fichiers, répertoires, etc.). Une fois l'instance obtenue, nous pouvons nous en servir pour positionner le répertoire

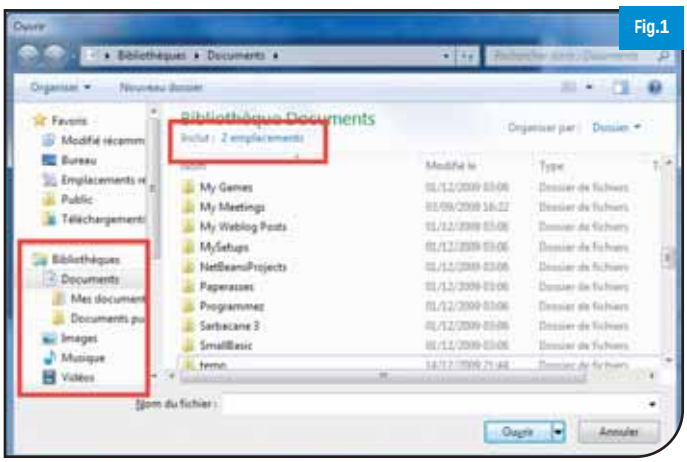


Fig.1

La boîte de dialogue du fichier de Windows 7 fait apparaître les librairies.

d'affichage du dialogue. C'est ce que fait notre second exemple (DemoIFileDialog2-1 sur notre site), pour 5 répertoires prédéfinis à la suite. Nous en donnons seulement l'extrait intéressant ici :

```
// pour KF_FLAG_DEFAULT_PATH
#include <shlobj.h>
// Pour les répertoires de librairie prédéfinis
#include <Knownfolders.h>


void SetFolders(SmartCOMPtr<IFileDialog2> fd)
{
 GUID folders[] =
 {
   FOLDERID_DocumentsLibrary,
   FOLDERID_PicturesLibrary,
   FOLDERID_MusicLibrary,
   FOLDERID_RecordedTVLibrary,
   FOLDERID_VideosLibrary
 };


 int nb = sizeof(folders)/sizeof(GUID);
 for(int i=0; i<nb; i++)
 {
   HRESULT hr;
   WCHAR *nom_fichier;
   SmartCOMPtr<IShellItem> item;
   SmartCOMPtr<IShellItem> result;
   // SHGetKnownFolderItem est une nouvelle fonction Windows 7
   hr = ::SHGetKnownFolderItem(folders[i],
     KF_FLAG_DEFAULT_PATH,
     0,
     IID_IShellItem,
     reinterpret_cast<void **>(&item));
   COMHelper::TestOk(hr);

   hr = fd->SetFolder(&(*item));
   COMHelper::TestOk(hr);
   hr = fd->Show(0);
   if(hr == S_OK)
   {
     hr = fd->GetResult(&result);
     COMHelper::TestOk(hr);
     hr = result->GetDisplayName(SIGDN_NORMALDISPLAY , &nom_fichier);
     COMHelper::TestOk(hr);
     wcout << "Selection: " << nom_fichier << endl;
     ::CoTaskMemFree(nom_fichier);
   }
 }
}
```

En outre, nous avons rendu le dialogue fonctionnel. Ainsi lorsqu'on clique sur le bouton OK après avoir sélectionné un fichier, nous récupérons et affichons le nom de ce fichier. Une occasion de rappeler une caractéristique de COM. Dans un tel cas nous obtenons un pointeur WCHAR* sur le nom de fichier, et il est de la responsabilité du programmeur de libérer la mémoire ainsi pointée au moyen de l'API CoTaskMemFree.

4 LIBRAIRIES ET RÉPERTOIRES ASSOCIÉS

Lorsque l'explorateur Windows ou une boîte de dialogue affichent une librairie, ceux-ci présentent un lien "x emplacements". Cliquer sur ce lien fait apparaître une nouvelle boîte de dialogue qui permet de gérer une librairie et ses répertoires associés [Fig.2]. Bien sûr, cette boîte de dialogue peut être ouverte par programmation. Le code, sans intérêt, n'est pas donné ici. Le lecteur le trouvera sur notre site, dans l'exemple DemoLibraryManagementDialogue. Comme on le voit, il est même possible d'inclure des répertoires situés sur une machine distante dans une librairie. Les données constitutives d'une librairie sont rangées dans un fichier XML maintenu par le système. La documentation recommande très chaudement de ne jamais toucher manuellement à un tel fichier. En revanche, il est possible de le faire par programmation.

5 LISTER LES RÉPERTOIRES D'UNE LIBRAIRIE

Pour commencer il est possible de lire ces données, c'est-à-dire de lister les répertoires constitutifs d'une librairie. Voici un extrait de l'exemple DemoGetFolders qui réalise ce travail. Nous voulons lister la librairie *Images* correspondant à l'illustration précédente :

```
void ListFolders(GUID libid)
{
 HRESULT hr;
 SmartCOMPtr<IShellLibrary> library;
 SmartCOMPtr<IObjectArray> ioarr;

 hr = ::SHLoadLibraryFromKnownFolder(
   libid,
   STGM_READ,
   IID_IShellLibrary,
   reinterpret_cast<void**>(&library));
 COMHelper::TestOk(hr);

 hr = library->GetFolders(LFF_ALLITEMS,
   IID_IObjectArray,
   reinterpret_cast<void**>(&ioarr));

 if(hr == S_OK)
   cout << "Tous les repertoires sont obtenus" << endl;
 if(hr == S_FALSE)
   cout << "Des repertoires ne seront pas listes" << endl;
 if(hr != S_OK && hr != S_FALSE)
   COMHelper::TestOk(hr);

 UINT count;
 hr = ioarr->GetCount(&count);
 COMHelper::TestOk(hr);
 for(UINT i=0; i<count; i++)
 {
   SmartCOMPtr<IShellItem> item;
   WCHAR *nom;
   hr = ioarr->GetAt(i, IID_IShellItem,
     reinterpret_cast<void**>(&item));
   COMHelper::TestOk(hr);
   hr = item->GetDisplayName(SIGDN_NORMALDISPLAY , &nom);
   COMHelper::TestOk(hr);
   wcout << nom << endl;
   ::CoTaskMemFree(nom);
 }
}
```

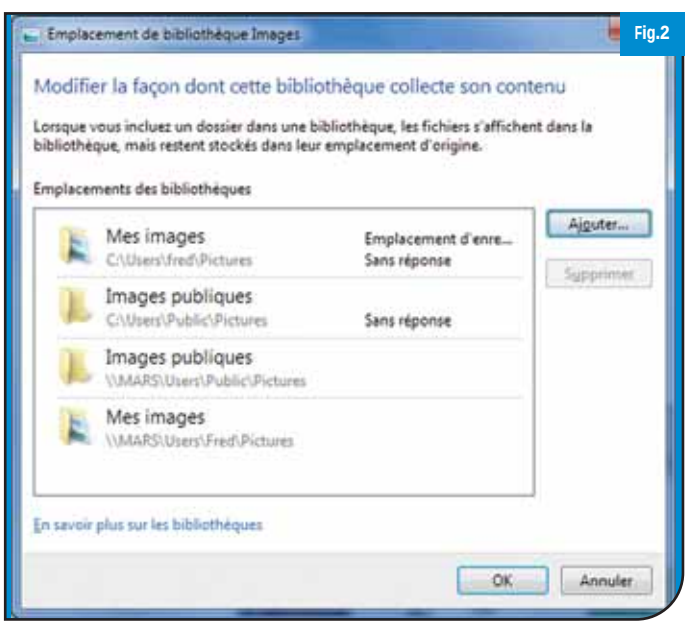


Fig.2

La boîte de dialogue de gestion des répertoires d'une librairie.

Quelques points méritent notre attention. Tout d'abord l'instance de IShellLibrary, l'interface COM pour travailler avec les librairies Shell, n'est pas instanciée classiquement par l'API COM CoCreateInstance. On pourrait le faire, mais alors la librairie ne contiendrait aucune donnée et donc aucun répertoire à lister. Rappelez-vous l'absence de constructeur mentionnée plus haut et donc l'impossibilité de passer des paramètres lors de la création d'objet. Par contre, cette approche serait parfaite si le but était de créer une nouvelle librairie à partir de zéro. Pour nous qui voulons charger une librairie prédéfinie, dont le fichier XML descriptif est soigneusement caché par le système, nous devons passer par l'API *SHLoadLibraryFromKnownFolder*. Une fois l'instance de IShellLibrary obtenue, il est très simple d'obtenir les répertoires rangés dans un IObjectArray, une autre nouvelle interface Windows 7 exposant des fonctionnalités de tableau. Lister ensuite le contenu est sans difficulté. Comme précédemment, on n'oubliera pas de libérer les ressources par CoTaskMemFree. Toutefois, le résultat des opérations nous réserve une petite surprise :

```
Mes images
Images publiques
Mes images
Images publiques
```

Nous ne voyons aucune différence entre les répertoires locaux et distants, et aucune indication quant aux répertoires réels associés.

6 RÉSOLUTION DE RÉPERTOIRES

Pour obtenir les renseignements qui nous manquent, nous devons forcer la résolution des répertoires. Nous obtenons alors les bonnes données, même si les répertoires ont été déplacés ou

renommés depuis leur intégration à la librairie. Le lecteur trouvera le code complet DemoGetFoldeResolve sur notre site. Par rapport à l'exemple précédent, la seule modification se situe dans la boucle d'affichage :

```
UINT count;
hr = ioarr->GetCount(&count);
COMHelper::TestOk(hr);
for(UINT i=0; i<count; i++)
{
 SmartCOMPtr<IShellItem> item;
 SmartCOMPtr<IShellItem> resolved;
 WCHAR *nom;
 hr = ioarr->GetAt(i, IID_IShellItem,
   reinterpret_cast<void**>(&item));
 COMHelper::TestOk(hr);
 hr = item->GetDisplayName(SIGDN_NORMALDISPLAY , &nom);
 COMHelper::TestOk(hr);
 wcout << nom << " -> ";
 ::CoTaskMemFree(nom);

 hr = library->ResolveFolder(&(*item),
   1000,
   IID_IShellItem,
   reinterpret_cast<void**>(&resolved));
 if(hr != S_OK && hr != S_FALSE)
 {
   cout << endl;
   continue;
 }
 hr = resolved->GetDisplayName(SIGDN_FILESYSPATH, &nom);
 COMHelper::TestOk(hr);
 wcout << nom << endl;
 ::CoTaskMemFree(nom);
}
```

Le résultat est alors pleinement satisfaisant. Et les noms de répertoires distants incluent le nom de la machine distante. [Fig.3]

7 AJOUTER UN RÉPERTOIRE ET LE DÉFINIR PAR DÉFAUT

Qu'une librairie soit composée de plusieurs répertoires, c'est très bien, mais lorsqu'on souhaite écrire un fichier dans cette librairie on en vient à se demander où le fichier va se trouver physiquement ? Il sera dans le répertoire qui est défini par défaut. Notre dernier exemple DemoAddFolder, dont vous trouverez un extrait ci-dessous, fait ce travail. [Fig.4]

```
/*
* AddFolderAndSetDefault
*
* ATTENTION, si le répertoire est déjà présent dans la librairie
* cette fonction échoue
*/

SmartCOMPtr<IShellItem>
AddFolderAndSetDefault(SmartCOMPtr<IShellLibrary> library, WCHAR* folder)
{
 HRESULT hr;
 SmartCOMPtr<IShellItem> item;

 hr = ::SHCreateItemFromParsingName(
   folder,
   NULL,
   IID_IShellItem,
   reinterpret_cast<void**>(&item));
 COMHelper::TestOk(hr);

 hr = library->AddFolder(&(*item));
 COMHelper::TestOk(hr);

 hr = library->SetDefaultSaveFolder(DSFT_DETECT, &(*item));
 COMHelper::TestOk(hr);

 hr = library->Commit();
 COMHelper::TestOk(hr);

 return item;
}
```

Il est important d'ouvrir la librairie en écriture pour ces opérations lors de l'instanciation de IShellLibrary avec l'API SHLoadLibraryFromKnownFolder (Voir la fonction GetLibrary dans l'exemple complet sur notre site). Le répertoire à ajouter doit être décrit par une instance de IShellItem. Cette instance doit être obtenue via l'API SHCreateItemFromParsingName qui reçoit le chemin absolu du répertoire en argument. Lorsque tout est terminé, il faut valider les opérations au moyen de la méthode Commit de IShellLibrary.
À bientôt pour d'autres aventures dans le cœur de Windows 7.

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

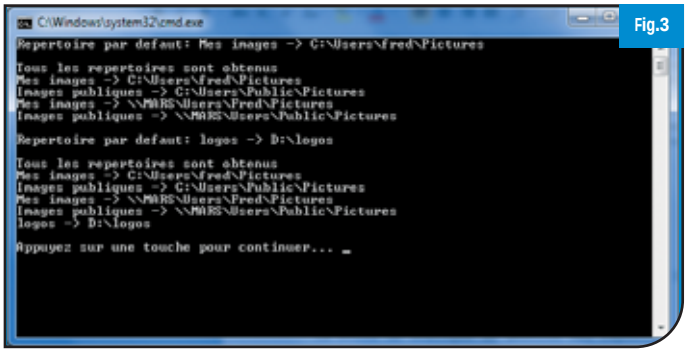


Fig.3
Ajout d'un répertoire à notre librairie Images

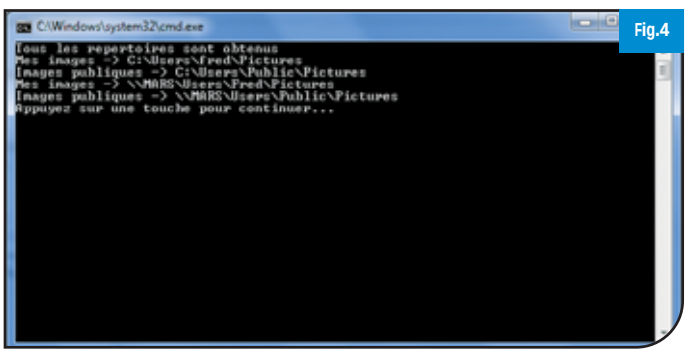


Fig.4
Liste des répertoires constitutifs d'une librairie, y compris les répertoires distants.

MAP

Mon GPS en action

*Difficulté : ***
Editeur : Eyrolles
Auteur : Paul Correria
Prix : 19,90 €

Aujourd'hui il existe de multiples manières de se guider : GPS, iPhone, Google Maps et Earth, Bing, etc. Ce livre se propose de maîtriser les concepts de parcours GPS et de savoir comment partager vos découvertes. Vous apprendrez à capturer vos traces GPS, à afficher vos déplacements, à vous géolocaliser, à présenter vos voyages, ou tout simplement à créer vos propres cartes ! Farfelu ? Non, car aujourd'hui de nombreux outils gratuits, ou non, sont disponibles sur le web. Vous saurez comment préparer vos cartes, vos parcours, comment géolocaliser vos cartes ou encore tout simplement comprendre le fonctionnement des GPS. Idéal pour préparer l'été !

SOCIÉTÉ

Web social

*Difficulté : ***
Editeur : Presse de l'université du Québec
Auteur : collectif
Prix : 27 €

Internet a pris aujourd'hui la forme du Web social : en mobilisant les technologies 2.0, Internet devient un lieu participatif où l'usager est appelé à créer des contenus, à les échanger, à les remixer. Il est invité à se mettre en valeur à travers des sites de réseaux sociaux et à s'exprimer dans des blogs personnels ou politiques.
L'usager est ainsi placé au centre du Réseau des réseaux. Les recherches sur ces nouveaux usages interpellent plusieurs communautés scientifiques intéressées par les rapports entre technique, communication et société. Cet ouvrage propose une cartographie inter-disciplinaire de ces travaux récents. En réunissant des analyses dans des sphères variées (culture, jeu, travail, journalisme, démocratie participative, éducation, santé) et des essais critiques sur l'utopie du Web social, les auteurs de cet ouvrage interrogent les figures de l'Internet contemporain : le Web social annonce-t-il une mutation de la communication ?

WEB

PHP et MySQL – MySQLi – PDO 2e édition

*Difficulté : ****
Editeur : édition Eni
Auteur : Christophe Villeneuve
Prix : 39 €

Dans un premier temps, l'auteur choisit de décrire les principales fonctions de PHP en prenant des exemples facilement compréhensibles. Il passe ensuite en revue les différentes étapes du développement en s'aidant des exemples de la première partie (accès sécurisés, gestion du carnet d'adresses, gestion des mots de passe, gestion des administrateurs, affichage et exportation des données...). Un chapitre complémentaire détaille des notions plus avancées comme les contrôles de sécurité, le suivi de la navigation des visiteurs...Cette nouvelle édition de l'ouvrage met l'accent sur deux points : le premier concerne internet (sécurité captcha, transmissions des données entre sites...), l'autre point concerne la programmation objet, de sa découverte à l'utilisation des espaces de noms (namespace). Une bonne référence pour le développeur PHP.

ADMINISTRATION

Windows Server 2008 R2

*Difficulté : ****
Editeur : édition Eni
Auteur : collectif
Prix : 39 €

Avec la release 2 de Windows Server 2008, Microsoft remet à l'heure son système serveur, notamment en l'alignant sur certaines fonctions de Windows 7 et en particulier sur son noyau. Le duo exploite aussi plusieurs technologies réseaux très intéressantes telles que DirectAccess, évitant de créer un VPN.
Cet ouvrage se destine aux administrateurs et utilisateurs avancés. On y parlera Active Directory, DFS, Hyper-V, répartition de charge et des nombreuses améliorations du système en lui-même.
Ce livre se veut un compagnon à votre R2 depuis son déploiement à son utilisation au quotidien, dans un contexte purement réseau ou lié à une virtualisation de serveur. On peut regretter l'absence de perspectives liées au cloud computing ou encore le peu d'éléments sur la migration.

LIVRE DU MOIS

Windows 7 pour les développeurs

*Difficulté : ****
Editeur : Microsoft Press
Auteur : collectif
Prix : N.C. €

Windows 7 introduit de nombreux changements et des améliorations significatives par rapport à Vista, notamment sur l'interface. Le développeur doit apprendre de nouveaux réflexes et connaître les nouveaux paradigmes d'interface pour réussir une bonne intégration avec Windows 7. Les auteurs couvrent aussi la nouvelle barre des tâches, le Ribbon, la création des applications tactiles, l'intégration du tactile dans Silverlight, la géolocalisation et toutes les nouvelles fonctions d'instrumentation, de performance et de diagnostic. Clair et bien agencé, l'ouvrage se lit assez rapidement et les nombreux exemples permettent d'appréhender plus vite les nouveautés, les mécanismes. Sans être un guide de référence, il s'agit néanmoins d'une très bonne introduction à la programmation Windows 7.
Si vous vous lancez dedans, ce livre est tout indiqué.

Amen.fr, un lieu unique pour faire le plein de services.

7 services intégrés dans une offre web globale et unique.

- **Une gamme de services complète et réellement novatrice qui répond à 100% des besoins Internet :** plateforme de création de site simple ou marchand, solutions multiples d'hébergement (mutualisé, privé et dédié), outils de promotion et de protection de noms de domaine.
- **Un nouveau site web** à la navigation simple et intuitive : www.amen.fr
- **Un partenaire solide** grâce à son appartenance au Groupe Dada, l'un des leaders internationaux sur le marché de l'hébergement web et de la publicité en ligne.

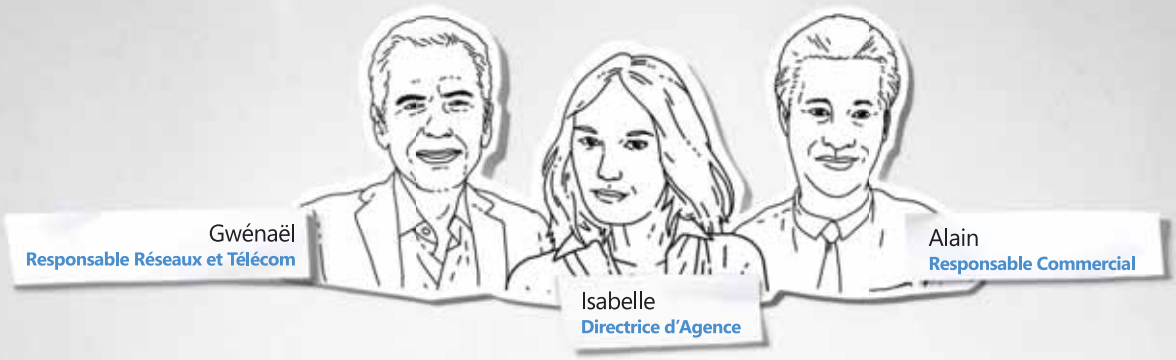
domaine | email | hébergerment | serveur | publicité | protection de marque | e-commerce

Amen
A DADA COMPANY

0 892 55 66 77
(0,34 €/mn)

www.amen.fr

Microsoft



Communiquer et partager les idées en toute sécurité

Parfois, l'information que l'on recherche est sur un disque dur, parfois elle se trouve dans la tête de quelqu'un. En déployant Microsoft Exchange Server, avec ses services de Communications Unifiées, SharePoint, et ForeFront, vous ferez des économies en permettant à vos collaborateurs d'échanger des idées où qu'ils soient et d'accéder aux informations dont ils ont besoin en toute sécurité.

Pour une communication et une collaboration plus efficace, plus sûre et plus facile à gérer, rendez-vous sur : **www.nouvelle-efficacite.fr**

Parce que c'est l'affaire de tous