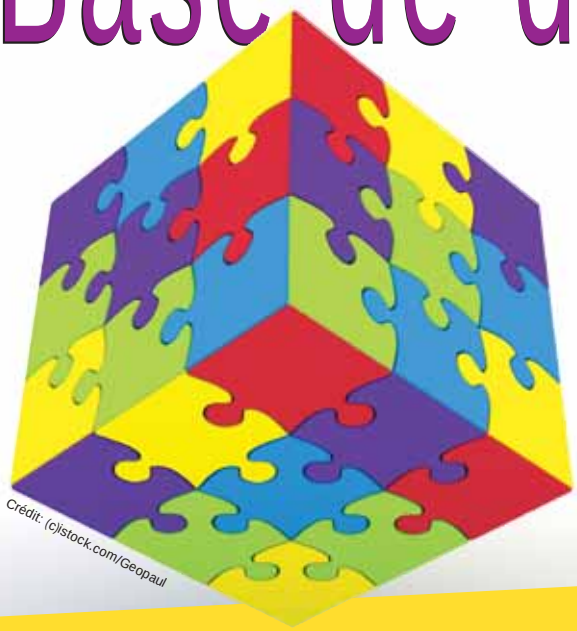
PROGRAMMEZ!
Le magazine du développement
PROgrammez!
www.programmez.com
mensuel n°126 - janvier 2010

Base de données


Crédit: (c)istock.com/Geopaul

MySQL
menaçé par les forks ?

PostgreSQL *vs MySQL : comment choisir ?*

SQL Server 2008 :
Maîtriser les index et adopter une gestion par règles

Objet vs Relationnel
ami ou ennemi ?

Open Source

Mike Milinkovich
Fondation Eclipse

Anthony Wasserman
Université Carnegie Mellon

◀ ***Jim Zemlin***
Fondation Linux

Trois gourous se dévoilent

Tout savoir sur Visual Studio 2010

Webmaster

Flex 4.0 débarque

Le nouveau langage FXG

L'interface devient dynamique

Pas de répit chez Google

GO
Chrome OS
Closure Tools

T4, Axum :

Les incroyables langages de Microsoft

Choisir son
ÉCOLE INFORMATIQUE

Cloud
Utiliser Cloud Foundry

Shell
Maîtriser PowerShell 2

Java
Scala : le nouveau Java ?

Technique
Mettre en oeuvre Hudson

M 04319 - 126 - 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

Microsoft

Touchez du doigt une nouvelle efficacité qui n'a rien de virtuel.

Mettre en œuvre une infrastructure virtualisée, du Data Center au poste de travail est désormais à portée de main. Choisissez Windows Server®2008 R2 avec Hyper-V™, et vous pourrez vous passer de logiciels tiers aussi coûteux que superflus. Ajoutez-y SQL Server®2008 Entreprise et vous voici libéré de vos racks de serveurs sous-utilisés. Quant à System Center, il vous donnera la touche finale pour une gestion de votre système d'information homogène et ce jusqu'au niveau applicatif. Résultat ? Une architecture virtualisée et réactive qui vous garantit un retour sur investissement optimal et une fluidité sans égale de vos processus métiers. Vous ne pouvez plus laisser passer une telle efficacité…

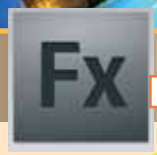
Pour découvrir ce que la virtualisation peut vous faire gagner en efficacité, rendez-vous sur : **www.nouvelle-efficacite.fr**

Parce que c'est l'affaire de tous

PROgrammez! Le magazine du développement

numéro 126 - janvier 2010

sommaire



PROCHAIN NUMÉRO

N°127 Février 2010, parution 30 janvier

✔ Modélisation & Model Driven

Comment les nouvelles générations de modélisation aident-elles le développeur au quotidien ? Découvrez les nouvelles tendances du Model Driven

✔ Microsoft en 2010

A l'occasion des TechDays 2010, le panorama et l'analyse des nouveautés Microsoft pour les développeurs, le web, les bases de données, la bureautique, le cloud computing...

PROgrammez! Le magazine du développement

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, C. Remy, L. Guillois, P. Davy. **Experts :** G.Renard, G. Belmas, M. Chaize, C.P. de Geyer, S. Combaudon, V. Bellet, L. Ait Ali, O. Caudron, L. Baumann, G. Rouchon, V. Labatut, M. Oppetit, G. Boissineau, N. François, B. Talmard, S. Saurel. **Illustration couverture :** ©istock.com/Geopaul. **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 : ETC - 76198 Yvetot. **Directeur de la publication :** J-C Vaudecrane

Abonnement : Programmez 22, rue René Boulanger, 75472 Paris Cedex 10 - Tél. : 01 55 56 70 55 mail : 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

BLU AGE & magicdraw™
AGILE MODEL TRANSFORMATION

présentent

A PARTIR DE
1490 €
PAR UTILISATEUR !

M2Flex & M2Spring

L'atelier agile de génération d'applications dirigé par les modèles !

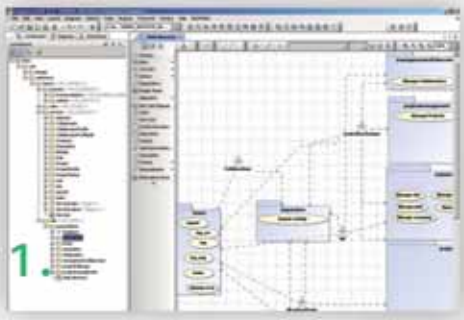
BOOSTEZ VOS DÉVELOPPEMENTS D'APPLICATIONS FLEX & SPRING !

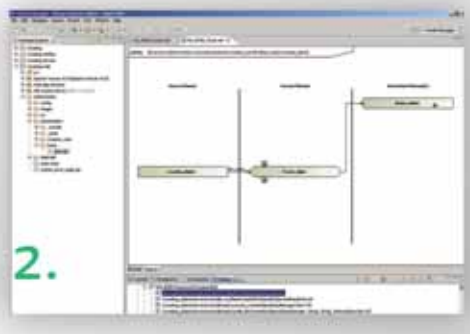
Egalement disponible :

M2Code FOR JAVA™

Pour la génération automatique de vos applications JEE !

1. DESSINEZ FACILEMENT VOS MODÈLES UML,
2. VALIDEZ ET DÉBUGGEZ VOS MODÈLES,
3. GÉNÉREZ EN UN CLIN D'OEIL 100% DE VOS APPLICATIONS !





Votre revendeur
Comsoft-SOS Developers
Tel : 08 25 07 06 07
infos@comsoft.fr
www.sosdevelopers.com

COMSOFT direct | SOS DEVELOPERS
Bechtle's Software Specialist

Informations et licences d'évaluation :
www.model2code.com ou **01 56 05 60 91**

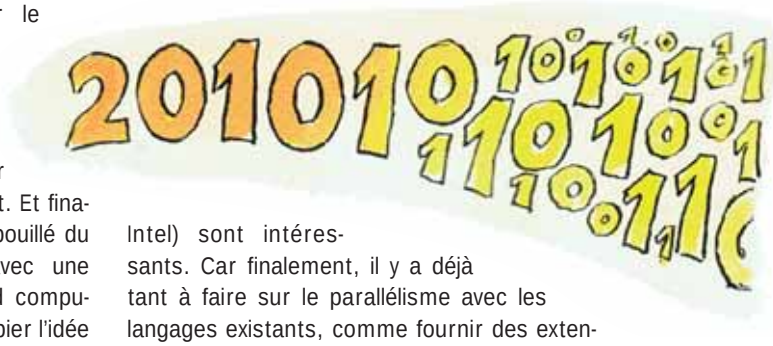
éditoll

Plus fort que Nostradamus

Il est bien connu que la nature a horreur du vide et le monde de l'incertitude. Or 2009 n'a été qu'une longue succession de questions. L'affaire la plus emblématique concerne le rachat de Sun par Oracle, toujours suspendu à la décision européenne qui devrait intervenir fin janvier. Le sort de MySQL exacerbe les tensions, les scénarios hypothétiques. Cette turbulence n'aide pas MySQL, même si la communauté se veut rassurante. Par contre, à terme, cela peut aider les forks à sortir de l'ombre ou encore aux concurrents ouverts à reprendre des utilisateurs.

Google a joué au chat et à la souris une bonne partie de l'année 2009, mais depuis septembre dernier, c'est une cascade d'annonces : nouveaux langages et outils, nouveau système, futur téléphone... A croire que Google veut appliquer la vision de Microsoft : S + S, logiciels et services. Clairement, avec ChromeOS, Google veut concurrencer Windows, principalement sur le Netbook, et avec le futur « Google Phone », Blackberry et iPhone. Mais il est décevant, pour ne pas dire surprenant, que ChromeOS ne soit pas compatible avec Android, le système mobile de Google. Cela oblige le développeur à créer une nouvelle branche de développement. Et finalement, ChromeOS n'est qu'un autre Linux, dépouillé du superflu et avec une dose de cloud computing. Sur le papier l'idée est assez séduisante mais il est bien trop tôt pour prédire quoi que ce soit. Notre souhait ? Un modèle de développement unifié entre Android et ChromeOS, un support matériel bien plus large. Mais la grosse interrogation reste la connexion à son cloud (donc à internet). Là, l'utilisateur doit composer avec la qualité de son abonnement Internet et la disponibilité des services Google... Et de tout cela, les responsables évitent soigneusement d'en discuter. Ne parlons pas des sujets qui fâchent.

Autre constat, impactant directement le développeur, l'évolution des langages, voire l'apparition de nouveaux dialectes plus ou moins « régionaux ». Nous avons déjà eu l'occasion d'en discuter ici même. Et l'éternel débat revient : faut-il enrichir un langage existant ou en créer un autre ? Nous étions, particulièrement pour la programmation parallèle, partisan de la création mais dorénavant, avouons que la prudence est de mise. Et les arguments de James Reinders (spécialiste parallèle chez Intel) sont intéressants. Car finalement, il y a déjà tant à faire sur le parallélisme avec les langages existants, comme fournir des extensions communes afin de proposer un modèle de programmation suffisamment indépendant des langages et des plates-formes. Si l'industrie réussissait cette étape, le parallélisme aurait fait un grand pas vers le développeur. Voilà notre vœu pour 2010.



Toute l'équipe de Programmez ! vous souhaite une bonne année 2010, avec tout plein de 0 et de 1.

François Tonic

C'est Noël ! 4 milliards d'euros pour le numérique

A l'heure où ce magazine part sous presse, les chiffres du Grand Emprunt viennent d'être annoncés. Nous ne connaissons pas tous les détails, mais nous retenons ce cadeau de Noël pour l'informatique et pour le logiciel. Reprenant la préconisation du rapport Juppé-Rocard, 4 milliards d'euros sont attribués à l'IT : deux pour le développement du très haut débit et deux pour le développement d'usages et de contenus numériques innovants.

Pour « Investir dans la société numérique » ces deux milliards devront permettre de *« financer des projets partenariaux publics-privés de recherche et de démonstration visant la conception de logiciels, d'usages et de contenus numériques innovants dans tous les domaines, en premier lieu dans les réseaux intelligents (électriques et de transport), mais aussi dans le télétravail, la télémédecine, l'e-santé, l'e-administration, l'e-éducation, l'e-justice, la numérisation des contenus culturels... au besoin en s'appuyant sur le développement d'infrastructures partagées (cloud computing, super-calculateurs...). Un fonds dédié à cet objectif, pourrait être créé au sein de la nouvelle agence pour le numérique ».*

D'autres fonds concerneront partiellement l'informatique, au travers de la création de pôles de formation et de recherche de taille mondiale, et du financement des PME innovantes. Nous avons perdu la bataille mondiale de l'industrie, nous commençons à nous donner les moyens pour gagner celle de l'intelligence. Et pour développer l'industrie de l'intelligence. 2 milliards d'euros, cela n'équivaut qu'à 2 semaines et demi du CA annuel de Microsoft (60 milliards de dollars en 2008). A vous de le faire fructifier !

Jean Kaminsky

Lire les propositions du rapport Juppé Rocard :
http://www.elysee.fr/documents/index.php?mode=view&lang=fr&cat_id=8&press_id=3109

■ **Zend** dévoile Zend Studio 7.1. Cette version supporte désormais le développement orienté tâche, une synchronisation simple de serveur distant, le support des archives PHP. Que du bonheur pour les développeurs PHP !

■ **JetBrains**, éditeur d'outils de développement, a sorti la version finale de IntelliJ IDEA 9. Cette mouture améliore les performances, compatible Java EE 6, support de PHP, de PHPUnit, de Adobe AIR et de Flex. Bref, une excellente version !

■ **Alfresco** et **SpringSource** annoncent Spring Surf Extension. Il s'agit d'un framework de développement pour créer du contenu riche basé sur Java. Projet conçu par Alfresco, il passe désormais en licence Apache. Le framewok permet d'utiliser plusieurs librairies comme Web Scripts. Il utilise aussi une approche MVC (via Spring MVC). Inclut aussi des outils de développements (Web Studio).

■ **4D** dévoile 4D v11 SQL Release 5. Cette version est désormais compatible (officiellement) avec Windows 7 et MacOS X 10.6. Elle inclut aussi des modifications dans le langage avec de nouvelles commandes. L'exécution des applications 4D se voit améliorée avec de nouvelles fonctions d'optimisation (ex. : gestion de la mémoire cache).

■ **Google** prépare son propre Smartphone. Rumeur depuis des mois, l'information a été confirmée mi-décembre. HTC en sera le constructeur. Pour le moment, rien de précis sur le modèle, son business model ou encore sa date de commercialisation. Il fonctionnera naturellement sous Android. Les constructeurs de mobiles ne sont sans doute pas pressés de voir ce futur concurrent arriver…

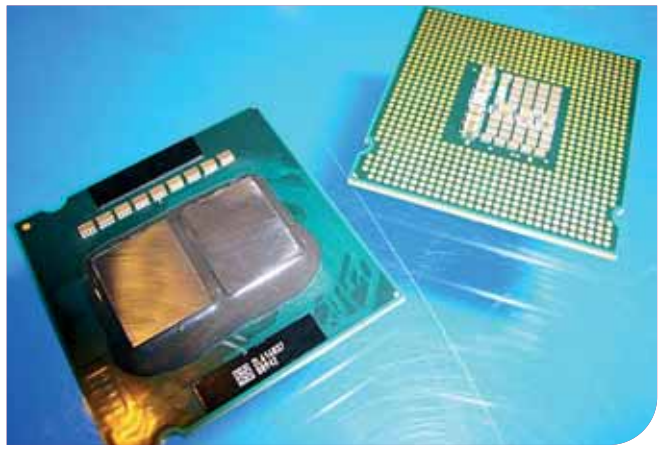
Multicore

Intel veut renforcer le développement parallèle

Intel continue de miser plus que jamais sur la programmation parallèle, le multicore et les outils permettant de simplifier le développement parallèle. Aujourd'hui, il y a très peu de code réellement optimisé pour le multicore, c'est-à-dire les applications massivement parallèles capables d'utiliser au mieux plusieurs processeurs et plusieurs cœurs simultanément dans le même ordinateur. Pour cela, il faut éduquer, convaincre les développeurs de devenir de bons experts en programmation parallèle, ce qui est loin d'être le cas !

Pour Intel, la première étape fut franchie au printemps dernier avec la disponibilité de la gamme Parallel Studio, dédiée à la programmation parallèle, à la traque des bugs et à l'optimisation du code. Disponible pour les développeurs C++ sous Visual Studio, la gamme a eu droit à un premier service pack en début octobre. La principale nouveauté étant la prise en charge de Windows 7 ainsi que la possibilité de piloter Parallel Inspector en ligne de commande et quelques corrections de bugs.

James Reinders, expert en parallélisme et responsable des outils parallèles d'Intel, de passage à Paris début décembre, nous a dévoilé une partie de l'agenda 2010 des outils Intel. Sur la gamme Parallel Studio, le fondeur sortira une nouvelle version pour supporter Visual Studio 2010 et les nouveautés qui iront avec (ex. : CCR, TPL, piles parallèles de Microsoft). James se réjouit de l'investissement de Microsoft sur le parallélisme avec Visual Studio. Cela ne peut qu'aider ce type de développement à s'imposer. Mais au-delà de Windows, Intel va proposer de plus en plus de fonctions Windows de Parallel Studio aux outils Linux. L'objectif est d'avoir le même niveau, ou quasi, entre Windows et Linux. Cependant, James nous a précisé qu'aucune version des plug-in Parallel Studio n'est prévue pour Eclipse ou NetBeans.



Le projet Ct

L'autre gros chantier est le projet Ct, dédié au parallélisme des données et sur la manière d'adapter les données, leurs traitements dans un contexte parallèle. Projet excitant et primordial dixit James, Ct tente de répondre à la problématique des données, de leur traitement et manipulation dans un contexte parallèle. Ct se présente comme une extension au C++. Il doit permettre de traquer les bugs, les erreurs et de découvrir les éventuels goulets d'étranglements. Car traiter des données en parallèle s'avère délicat pour garder une synchronisation entre les cores et surtout l'intégration des informations. Une première version bêta est disponible depuis peu (annoncée par James pour fin décembre). Par contre aucune date précise de disponibilité finale n'est encore annoncée. Intel souhaite tout d'abord stabiliser la technologie et avoir le retour des développeurs.

A la question : « *allez-vous créer un langage parallèle spécialisé* », James Reinders répond prudemment. Car pour notre expert, le plus important est de disposer des (mêmes) extensions parallèles dans les langages existants et de savoir les utiliser. Car qui dit nouveau langage, dit apprentissage et migration sur ce dit langage… Par contre, les langages fonctionnels (F#, Erlang), constituent des approches intéressantes même s'ils ne sont pas très populaires auprès des développeurs. James s'interroge sur un usage plus « intensif » de ces langages. Ils possèdent une bonne abstraction et leurs qualités ne sont plus à démontrer. Enfin si la première génération de multicoeur graphique, le projet Larabee, est purement et simplement annulée pour passer directement à l'étape suivante, Intel continue cependant de fournir les outils aux développeurs afin qu'ils se préparent dès maintenant à cette prochaine évolution majeure des PC et des applications… ■

Microsoft TechDays 10
L'innovation avance avec vous

8, 9, 10 février 2010 - Palais des Congrès de Paris

3 journées incontournables pour découvrir toutes
les dernières innovations informatiques

EVENEMENT GRATUIT*

Inscrivez-vous dès maintenant

www.mstechdays.fr

300 conférences techniques

5 500 m² d'exposition

Près de 130 exposants

Des experts et plus encore...

Edition spéciale lancements

Microsoft Office 2010 · Microsoft SharePoint Server 2010 · Microsoft Exchange Server 2010 · Microsoft Visio 2010 · Microsoft Project 2010

Windows 7 · Windows Server 2008 R2 · 01net. · Microsoft System Center

Microsoft Visual Studio 2010 · Windows Azure · Microsoft SQL Server 2008 R2 · et bien d'autres...

En partenariat avec : Le magazine du développement PROgrammez! www.programmez.com · SOLUTIONS & LOGICIELS

* Sur inscription en ligne

PLATEFORME PROFESSIONNELLE DE DÉVELOPPEMENT (AGL)

Windows, .Net, Java
Windows 7, 2000, NT, 2003, XP, Vista, 2008

W

La **version 15** de **WINDEV** vous apporte des nouveautés irremplaçables dans le domaine de la sécurité et des performances.

Vos applications sont plus sûres, plus rapides, plus compactes.

Vos utilisateurs et clients apprécient immédiatement ces évolutions.

■ **Sun** propose désormais pour son Java Application Store la possibilité de payer les applications avec le système sécurisé paypal. Cela concerne les applications allant de 1,99 à 200 dollars. Dans le même temps, Sun en profite pour modifier l'interface de l'actuelle bêta de la boutique en ligne Java. Egalement accessible pour les non-américain.

■ Le nouveau noyau **Linux** est disponible. Il s'agit de la version 2.6.33. Elle propose un support natif d'accélération 2D et un support 3D encore en développement. Elle intègre aussi le système de fichier DRBD dédié à la haute disponibilité, avec notamment la possibilité de répliquer le contenu d'un disque vers un autre à travers le réseau !

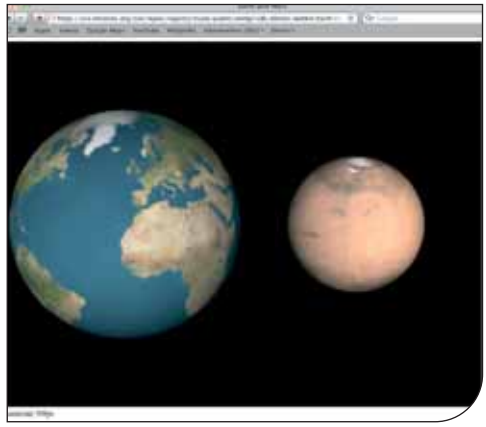
■ Le projet **Mono** renforce sa compatibilité .Net en sortant la version 2.4.3. Cette mouture corrige de nombreux bugs. Il implémente une version open source de l'outil MSBuild de Microsoft, xbuild. Autre bonne nouvelle, le compilateur C# de Mono est compatible avec C# 4, même si ce dernier n'est pas encore disponible par défaut dans la pile Mono !

■ **Blackberry** renforce sa boutique App World en y proposant depuis mi-décembre des thèmes d'interface pour Blackberry. Ils sont conçus depuis Blackberry Theme Studio 5.0, outil de création de thème téléchargeable.

■ **Weelya** a dévoilé une nouvelle version de son Ajax Push Engine. La version 1.0 propose un modèle technique respectueux des standards web (sans addition client), qui permet au serveur d'envoyer les informations à tous les utilisateurs dès qu'elles sont disponibles. Ainsi, ceux-ci ne sont plus ignorants des agissements du serveur, et peuvent attendre que les informations leur soient envoyées en temps réel.

Projet WebGL
va-t-il bousculer l'animation sur le web ?

Khronos Group a dévoilé courant décembre les premières spécifications autour de WebGL. Pour rappel, WebGL est une technologie web permettant, dans un navigateur web, sans rajout d'extension, de jouer des animations 3D, d'afficher des objets 3D ou encore d'interagir avec. WebGL supporte l'accélération graphique, indispensable pour avoir une bonne fluidité des textures et animations. Pour pouvoir afficher ces objets graphiques, WebGL utilise le canvas apparu avec HTML 5. Nous n'en sommes qu'au début du travail et des fonctionnalités. Il est difficile de savoir si cette technologie sera rapidement utilisée par les développeurs. Khronos espère une première version aboutie pour la première moitié de 2010.



La première spécification est intégrée dans les *nightly build* de Firefox, Safari, Chrome et Opera, même si ces versions restent instables et à l'implémentation plus ou moins complète. D'autre part, WebGL n'est pas activé par défaut. Notons que la sécurité constitue une partie non négligeable du travail de Khronos, notamment sur les restrictions d'accès aux ressources de la machine hôte et qu'il doit disposer de mécanismes empêchant les dénis de services. Autre point intéressant, WebGL supporte le langage OpenGL ES Shading Language.
Site officiel : http://khronos.org/webgl/wiki/Main_Page

Java EE 6 : certifié et disponible !

Certains diront enfin ! Car on attendait Java EE 6 depuis plusieurs mois. Mais au moins, contrairement aux versions Java précédentes, Java EE 6 a reçu immédiatement un soutien marqué d'IBM, Oracle, Red Hat. Cette version introduit de nombreuses nouveautés, consolidations et améliorations. On peut tout d'abord citer l'introduction des profils. Ils permettent de cibler des scénarios d'applications pour mieux coller aux marchés, aux secteurs métiers. On dispose aussi d'un profil très léger ciblant spécifiquement les applications web. L'extensibilité de Java EE est un autre élément présenté avec JEE 6. Cela permettra de créer et d'ajouter des plug-in. Tout cela doit rendre plus souple l'architecture de Java EE et permettre une meilleure adaptation. La partie web services et technologies web a subi une nouvelle fois une grosse mise à jour : Java API for RESTful Web Services, Contests and Dependency Injection (CDI) et enfin, la validation bean. CDI est un ensemble de composants JEE (session EJB, JSF...) permettant d'injecter et d'interagir avec des événements EJB ou JSF. Ainsi, il est possible de remplacer des beans d'entreprise par des beans JSF managés dans une application JSF. CDI tisse un pont entre une couche web d'une plateforme JEE et une couche entreprise. Reste à voir comment cela va être concrètement mis en œuvre. Autre point intéressant, le Bean Validation (JSR 303). Il permet de valider, par exemple, des caractères saisis dans un champ avant que cette saisie ne soit traitée dans l'application. Ainsi le développeur peut plus facilement vérifier et valider les saisies dans l'ensemble des couches de son application.

Dans la foulée, Sun a annoncé la disponibilité de NetBeans 6.8, bien entendu compatible JEE 6 et de la version finale de Glassfish, en le proposant dans un bundle Eclipse.
Pour en savoir plus : http://java.sun.com/javaee/

WinDev 15 : PC SOFT réussit son tour de France 2009

Comme chaque année, PC SOFT parcourt la France pour montrer la nouvelle version de sa solution intégrée : WinDev. La version 15 devrait être disponible quand vous lirez ces lignes. Une pré-version était disponible depuis la mi-décembre. Devant plusieurs centaines de personnes à Paris, les responsables de PC SOFT ont démontré durant plus de quatre heures les avantages et la simplicité de cette 15e version et comment WinDev réduit le temps de conception d'une application desktop, mobile ou web. Pour clore la session, une démo d'une interface proche de Microsoft Natal a été réalisée...

La mobilité est un des axes stratégiques pour l'éditeur, les entreprises ayant besoin d'intégrer la mobilité dans leurs applications. Ce n'est donc pas un hasard si Android, Windows Mobile et les écrans tactiles sont supportés. Il ne manque plus que le iPhone (déjà présent pour la partie site web).

L'autre élément à retenir de cette v15 concerne WebDev, la partie développement web qui intègre des animations « à la Flash » sans être du Flash. On dispose de belles animations de pages, d'onglets, de champs tiroirs, etc. On peut bien entendu utiliser du code Ajax pour disposer de sites dynamiques. On dispose aussi des url rewriting, ce qui simplifie bien le travail du développeur. PC SOFT suit aussi les dernières tendances en incluant la possibilité de créer des applications SaaS en cochant une simple option dans l'environnement de travail... Pour l'administration, on dispose d'une console centralisée d'où l'on peut gérer les utilisateurs, les créer, les supprimer. On dispose d'une API d'authentification SaaS assurant la connexion, l'identification (entreprise et utilisateur), la vérification de l'IP, la durée de connexion maximale, etc. Toujours dans le monde web, le support de PHP a été renforcé avec 45 nouvelles fonctions. Autre nouveauté importante, le support des nouveaux navigateurs (avec tests des sites) : Chrome, Opera et Safari. Encore une nouveauté bienvenue : la possibilité de déployer simplement des sites vers Free et OVH ! Sur une partie plus système, WinDev 15 permet aujourd'hui de générer des services Windows et Linux (sans interface). ■

Livre évènement

Richard Stallman et la révolution des logiciels libres

Le 21 janvier prochain sortira en librairie un livre événement sur Richard Stallman, figure emblématique du Libre depuis les origines.
Projet à l'initiative de Framasoft, cette biographie autorisée, revue et enrichie par lui-même, donne un éclairage nouveau sur ce personnage intègre aux allures de prophète. Une référence indispensable pour qui veut comprendre l'origine et les évolutions de cette lame de fond qu'est le logiciel libre.

agenda \\

JANVIER

Du 18 au 19 janvier 2010, Paris 17, Palais des Congrès
iLearning Forum Ltd, nouvelle édition du salon de référence européen le plus important en France dans le domaine des technologies au service de l'apprentissage. http://www.ilearningforum.org/fr/index.php

FEVRIER

Du 8 au 10 février, Paris 17, Palais des Congrès, **TechDays 2010.** Le rendez-vous incontournable des développeurs, décideurs et professionnels de l'informatique. Trois jours pour se former sur toutes les nouveautés Microsoft et découvrir les tendances du marché http://www.microsoft.com/france/mstechDays/

MARS 2010

Du 16 mars 2010 au 18 mars 2010, Paris Expo, Porte de Versailles
Solutions Linux / Open Source 2010
Organisé par Tarsus France, ce salon demeure le carrefour d'échanges incontournables des acteurs du logiciel libre.
http://www.solutionslinux.fr

■ **W4** dévoile sa nouvelle plate-forme **LEONARDI.** Il s'agit d'une solution complète d'automatisation de la production des interfaces homme-machine qui améliore le cycle

de développement. Cette version 4.4 intègre finement le projet de reporting BIRT, expose les services web et autorise de nouveaux composants. La partie outillage a elle aussi été améliorée et renforcée pour tirer parti des nouveautés.

■ **Novell** veut conquérir de nouveaux marchés en 2010. L'un d'entre eux concerne les solutions de gestion intelligente des environnements système. Novell y intègre la gestion des identités et des ressources. Cette approche répond aux problématiques des entreprises qui souhaitent réduire les risques liés à l'hétérogénéité des systèmes, tout en garantissant à leurs utilisateurs un accès sécurisé et conforme aux services dont ils ont besoin.

■ **Sodifrance** lance FASTtoFlex. Il s'agit d'une solution de transformation des applications client-serveur en application Flex /

Air. Pour la modernisation d'architectures, F.A.S.T. permet d'automatiser l'ensemble des processus, depuis la rétro-modélisation des applications existantes jusqu'à la génération massive du code final des applications dans la technologie cible. F.A.S.T. s'appuie sur une méthodologie et une plateforme logicielle outillée pour réaliser la transformation dans des délais plus rapides et avec un niveau de sécurité important.

Trois *gourous* du Libre se dévoilent !

Dans le cadre de l'Open World Forum, qui s'est déroulé à Paris début octobre, nous avons rencontré plusieurs figures marquantes du logiciel libre et du développement logiciel : Mike Milinkovich (Fondation Eclipse), Jim Zemlin (Linux Foundation) et Anthony Wasserman, qui est à la tête du centre d'investigation sur l'Open Source de l'Université Carnegie Mellon.

Réalisé par Philippe Davy

D.R.

Anthony Wasserman

Directeur exécutif du Centre de recherche sur l'Open Source, Université Carnegie Mellon

Tour à tour professeur, éditeur et chercheur, Anthony Wasserman est l'un des pionniers des outils de développement graphiques et le concepteur de StP. Il dirige désormais le COSI (Center for Open Source Investigation) de Carnegie Mellon University. C'est également l'un des fondateurs du consortium Open Source for America, qui vise à développer l'usage des logiciels libres au sein de l'administration américaine.

P ! : A quel moment avez-vous découvert les logiciels libres ?

Anthony Wasserman : J'étais à San Francisco à l'époque du développement de Berkeley Unix. Nous avions conçu des systèmes interactifs, assez nouveaux pour l'époque, qui n'étaient pas destinés à des informaticiens mais à des personnes du secteur de la santé. C'était en 1980. Ces outils ont été distribués sous licence BSD, et ont commencé à rencontrer un certain succès auprès des utilisateurs. Le problème était de trouver la bonne structure pour les supporter. Alors j'ai fondé IDE (Interactive Development Environments) en 1984 pour éditer Software Through Pictures (StP) qui était le premier outil graphique de modélisation multi-utilisateur. Nous avons utilisé du code sous licence BSD au sein d'un produit propriétaire. Nous étions parmi les tous premiers à le faire, car à part Sun je ne vois personne d'autre. IDE s'est développé, et la compagnie a été rachetée. Une partie du code de StP se trouve à présent au sein d'OpenAmeos, le projet libre d'outil de modélisation UML issu d'Aonix, qui avait racheté IDE au début des années 90.

P ! : Quelles sont les activités du centre de recherche ?

A. W. : La recherche du COSI est concentrée sur l'évaluation, l'adoption et l'usage des logiciels libres, avec pour objectif secondaire leur commercialisation. Les projets récents incluent le Business Readiness Rating, (un cadre pour l'évaluation des projets et des logiciels libres), qui traitera des notions de tests et de fiabilité nécessaires pour les entreprises. Il y a également le Software Project Governance Framework, qui permet d'évaluer l'ouverture des projets open source. Le COSI mène actuellement une enquête sur les développeurs d'applications mobiles, ainsi qu'une étude des activités de réseau social au sein des projets libres. Par ailleurs, je dirige un séminaire de formation sur le management de projets logiciels. Le but est de permettre a des développeurs déjà actifs de passer à la prochaine étape dans leur carrière, en prenant en compte la dimension business aussi bien que la partie technique, de façon à leur donner les moyens de lancer leur affaire. On étudie en détail la manière dont le logiciel est distribué, qu'il soit propriétaire, hébergé ou open source, ou encore une combinaison des trois. Je dirige également un séminaire sur l'Open Source.

> « Dans le monde académique, ceux qui développent du logiciel s'attendent à ce qu'il soit librement accessible. »

P ! : Comment en êtes-vous arrivé à diriger le COSI ?

A. W. : Après le rachat de IDE, je me suis investi dans le boom "dot com" à San Francisco, une très bonne façon d'apprendre JEE. Puis j'ai rejoint Bluestone Software, qui développait un serveur d'applications Java. Ma mission consistait à ouvrir leur centre de développement sur la Côte Ouest. Là nous avons conçu un toolkit open source, Total e-Mobile, destiné à permettre aux applications fonctionnant sur les téléphones portables d'échanger avec le serveur d'applications. C'était en 2000, et un peu en avance sur son temps. Ensuite HP a racheté Bluestone, et ils ne savaient pas vraiment quoi faire de nous. Alors, je suis parti, et j'ai été contacté par Carnegie Mellon, qui voulait ouvrir un centre de recherche sur la Côte Ouest. C'est très gratifiant. La notion de liberté est très importante dans le monde académique. Les développeurs dans ce domaine s'attendent à ce que leurs productions soient librement accessibles et utilisables. Cela tranche avec les habitudes de la publication scientifique, où toutes les revues sont propriétaires. C'est la prochaine bataille, la publication libre !

P ! : A quel moment avez vous été convaincu de la viabilité du modèle Open Source ?

A. W. : Tout a démarré en 1998, avec l'Open Source Definition. Bien sûr, d'autres avaient déjà prouvé la viabilité économique du modèle avant cela, mais le modèle a véritablement permis la relance après l'éclatement de la bulle Internet. Pendant la bulle, il y avait beaucoup d'argent disponible, on achetait des applications Oracle et des serveurs Sun, c'était facile. Mais quand l'argent a fait défaut, ce sont les PC et les logiciels libres qui on connu un vrai succès.

Jim Zemlin

Directeur exécutif de la Fondation Linux

Issue de la fusion de l'Open Source Development Labs et du Free Standards Group, la Fondation Linux s'est donnée pour tâche la promotion de l'OS libre, sa protection et sa standardisation. Elle est dirigée par Jim Zemlin depuis la fusion en 2007.

Programmez ! : Comment s'est constituée la Fondation Linux et quel est son rôle ?

Jim Zemlin : La Fondation est le résultat de la fusion du Free Standards Group et de l'OSDL. Séparément, les deux organisations sans but lucratif partageaient le même objectif, accélérer l'adoption de Linux, et nous avions déjà travaillé ensemble. Il y a eu une période où nous nous sommes demandés comment travailler en commun plus efficacement, et en 2007, compte tenu du rythme d'adoption de Linux, il a été décidé de les fusionner. Au départ, il s'agit de fournir une base de développement neutre pour Linux, à l'écart des contraintes à court terme imposées par le marché, à la fois pour la communauté des utilisateurs et pour les entreprises qui l'utilisent commercialement. Nous employons les principaux développeurs du noyau Linux, dont Linus Torvalds. Par ailleurs, nous fournissons une base légale importante pour protéger Linux sur le plan juridique, et nous coordonnons les travaux sur sa standardisation et sur les améliorations à lui apporter, avec des projets comme Desktop Linux pour le poste de travail ou Carrier Grade Linux, à très haute disponibilité pour les opérateurs réseaux. La liste des membres comprend des géants tels qu'HP, IBM, Intel, Oracle ou encore Google, mais aussi de nombreuses sociétés plus petites, et des universités affiliées.

P ! : Comment êtes-vous venu au logiciel libre ?

J. Z. : J'ai grandi au milieu d'ordinateurs. Mon grand-père était programmeur et l'un des fondateurs de Cray Research, le pionnier des supercalculateurs. Mon père était programmeur, mes frères le sont aussi. Quand nous étions petits, il y avait une règle simple: nous avions le droit de jouer sur l'ordinateur pendant dix minutes, mais il fallait programmer pendant une heure. Ma première machine fut un Tandy, nous avons eu des Atari, des Apple. Je me suis investi dans le logiciel libre il y a environ 9 ans. Je travaillais dans une société, Corio, dont l'activité consistait à héberger des progiciels tels que PeopleSoft. Lorsque nous avons décidé de nous introduire en Bourse, notre grande inquiétude était que nous n'étions pas propriétaires des logiciels que nous hébergions. Avec des logiciels libres, cela aurait été bien plus facile. Après l'introduction en Bourse, j'ai quitté la société et travaillé dans une start-up qui proposait des services autour du serveur Apache. Et puis j'ai eu la possibilité de participer à la définition de la plate-forme LSB (Linux Standard Base) au sein du Free Standards Group, un projet novateur.

> « Aujourd'hui, quiconque produit du logiciel ou du matériel dépend du logiciel libre »

P ! : Le logiciel libre se répand, et en même temps, de nombreux projets sont absorbés par des acteurs propriétaires. Comment cela va-t-il évoluer selon vous ?

J. Z. : La question est plutôt "qui absorbe qui ?" Je ne crois pas que l'on puisse réduire l'équation à un camp contre l'autre. Quiconque produit du logiciel, du matériel informatique, des solutions techniques, dépend du logiciel libre. Si vous regardez ce smartphone d'un constructeur de renom, il est propriétaire, mais rempli de logiciels open source. Il n'est plus possible de réaliser un système un tant soit peu sophistiqué sans recourir au logiciel libre. Le secteur entier dépend de l'open source. Les apports du logiciel libre sont l'efficacité, la stabilité, l'échange, le fait que quelqu'un va penser à un détail ou une fonction à laquelle vous n'auriez pas songé vous-même. Linux est un bon exemple : en général, les entreprises ne prennent pas Linux pour en faire leur propre version qu'elles devront supporter ad vitam aeternam. On n'a aucun des avantages du libre de cette façon. Manifestement des entreprises gagnent de l'argent avec le libre, en vendant des services associés. IBM gagne beaucoup d'argent avec, Intel également, de même que Google, même si c'est de manière indirecte. Je vis dans la Silicon Valley et tout le monde est d'accord pour dire que le modèle avec de fortes marges sur du logiciel propriétaire, comme Microsoft, c'est fini. MySQL, SpringSource, Xen, Jboss, SleepyCat ont été évalués bien au-delà de leur valeur comptable, et les financiers sont constamment à la recherche de nouveaux investissements. Je le sais parce qu'ils me demandent en permanence sur quoi investir.


D.R.

Mike Milinkovich

Directeur exécutif de la Fondation Eclipse

A l'origine en 2001, il s'agissait pour IBM de placer ses outils de développement Java sous la bannière Open Source. Mais depuis 2004, le projet Eclipse est devenu une fondation regroupant 160 sociétés membres et plus de 230 projets, l'une des plus importantes communautés sur la planète. Et, au delà de Java, ces projets concernent désormais C++ ou encore PHP, dans un large éventail de domaines. Mike Milinkovich dirige la Fondation Eclipse depuis sa création.

Programmez ! : parmi toutes les grandes communautés libres, comment caractériseriez-vous Eclipse ?

Mike Milinkovich : L'une des particularités d'Eclipse est que le processus est centré sur l'adoption commerciale. Pour améliorer cela, nous organisons chaque année un 'train de versions", c'est-à-dire un grand nombre de projets publiés le même jour, testés pour garantir que ces versions fonctionnent bien ensemble. En juin dernier, nous avons publié 37 projets, soit environ 20 millions de lignes de code. Cela fait six ans que nous sortons nos versions à l'heure dite, au jour près. Cette publication a lieu le dernier jeudi de juin, chaque année. Ce côté prédictible est ce qui caractérise Eclipse, parce que des entreprises comptent sur ces produits pour fonctionner, savent qu'elles peuvent tabler sur des délais respectés. Et cela renforce leur confiance. Si l'on se place dans le contexte d'origine, initié par IBM, cela a entraîné d'importants changements culturels. Garantir l'indépendance vis à vis d'un acteur est le principal objectif de la Fondation, et la partie la plus intéressante de notre travail.

> « Depuis six ans, nous publions nos versions à l'heure dite, au jour près. »

P ! : Personnellement, comment en êtes vous venu au logiciel libre ?

M. M. : La première fois que j'ai entendu parler du logiciel libre, je n'y ai rien compris du tout. Comment gagne-t-on sa vie avec du libre ? Pourquoi mettre un logiciel sous licence libre ? C'était dans les années 1994, 1995. Par la suite dans une autre activité, je me suis trouvé en compétition avec l'Open Source, puisque je travaillais pour Toplink, le logiciel de mapping objet / relationnel, en concurrence avec le projet Hibernate. J'ai observé leurs parts de marché s'éroder dans cette compétition, et il faut avouer que j'étais un peu frustré face au côté inexorable de cette érosion.

Dans mon nouveau travail, j'ai en quelque sorte rendu les armes et je me suis associé au mouvement. Je suis devenu complètement convaincu que l'open source représentait le futur de la production de logiciels. Quand les chercheurs expliquent qu'un tiers de l'argent dépensé pour le logiciel est dilapidé, je pense que ces statistiques sont vraies, c'est un tel gâchis de talent, d'avoir toutes ces nombreuses sociétés réinventant la roue en permanence.

P ! : Mais la réutilisation n'était-elle pas déjà un sujet de réflexion central avant l'émergence du logiciel libre ?

M. M. : Bien sûr, mais à l'époque tout le monde, moi y compris, croyait qu'il s'agissait d'un problème technique. Nous étions des ingénieurs sans perspectives, car il ne s'agit pas d'un problème technique, mais bien d'une question de licence de gouvernance, de processus de développement. Toutes ces questions doivent être traitées également, et l'open source fait un excellent travail là-dessus. Il ne s'agit pas de technologie, mais de coordonner de multiples sociétés de façon à ce qu'elles construisent quelque chose en commun, de gérer l'ensemble des questions liées à la propriété intellectuelle, aux licences, etc. Comme Eclipse, les autres communautés comme la fondation Apache, ou Linux ont toutes les mêmes problèmes à gérer. Chacun à sa manière, mais il s'agit toujours du même problème.

P ! : Comment voyez-vous l'avenir du logiciel libre ?

M. M. : On ne peut qu'être optimiste quand Microsoft crée une fondation Open Source. Mais, plus sérieusement, la tendance est irréversible. Les utilisateurs ne sont plus enclins à payer des licences hors de prix.

L'ouverture des communautés doit toutefois impérativement être préservée. Des communautés comme Apache, Eclipse, Mozilla ou encore OW2 sont véritablement le cœur du mouvement Open Source.

Il faut se méfier des communautés fermées, qui dépendent d'un seul acteur possèdant tous les droits. Et cet acteur est la propriété d'investisseurs dont le but dans la vie n'est pas d'être gentil avec la communauté. A l'inverse, nous sommes dans une structure sans but lucratif, et nous pensons être dans une bien meilleure position pour conserver l'équilibre entre les besoins de la communauté et ceux de l'écosystème commercial. ■

GAGNEZ LE POUVOIR DE CRÉER UNE KILLER APP

Lors d'une réaction électrochimique qui a animé les cellules dormantes d' un œuf très puissant, Gort a éclos. Doté de pouvoirs spéciaux et de la capacité d'insuffler aux applications ordinaires des interfaces utilisateurs extrêmement fonctionnelles, d'une grande facilité d'utilisation et possédant le "facteur Wow!", Gort conçoit des Killer Apps. Allez sur infragistics.com/killerapps pour découvrir comment commencer à créer vos propres Killer Apps.

Infragistics
KILLER APPS. NO EXCUSES.

Infragistics Ventes France N° Vert 800 231 8588
Infragistics Europe Ventes +44 (0) 800 298 9055
Infragistics India +91-80-6785-1111

Retour sur la PDC 2009 : *le futur est pour demain*

Il y a maintenant un mois, s'est tenue à Los Angeles la Microsoft Professional Developer Conference (PDC). Au travers de ces 3 jours et des centaines de sessions techniques, Microsoft a dévoilé sa stratégie technologique mais aussi un contenu permettant à ceux qui y ont assisté de repartir avec un véritable bagage technologique. Les keynotes (sessions plénières) sont données à l'occasion des grandes annonces mais les sessions techniques et les activités annexes (stands, rencontres, etc.) sont souvent une mine d'informations qu'il ne faut pas négliger. Morceaux choisis…

Windows Azure : les petits détails passés inaperçus mais vitaux !

Windows Azure **XDrive**, ça vous parle ? Il s'agit de la possibilité de « simuler » un volume NTFS à partir d'un Blob Azure sous-jacent. Une annonce qui peut se révéler cruciale pour décider ou non de migrer des applications existantes utilisant une persistance des données « classiques » vers Windows Azure et qui est pourtant passée quasiment inaperçue ! Toujours au niveau du storage d'Azure, connaissez-vous la différence entre un **block blob** et un **page blob** ? Le premier est le blob que l'on connait depuis la PDC 2008 tandis que le second est un nouveau type de blob introduit avec la nouvelle CTP de Windows Azure. Une des principales différences réside dans la taille maximale de stockage (200 Go pour le block blob contre 1 To pour le page blob). Le **snapshot** d'un blob, qui permet d'en conserver différentes versions, et le Content Delivery Network, sorte de **geo-caching** de blob, sont aussi des annonces « annexes » de la PDC. Si vous voulez encore plus de nouveautés sur l'Azure Storage, regardez donc la session Windows Azure Blob and Drive Deep Dive (http://microsoftpdc.com/Sessions/SVC14) disponible en ligne.

Windows Azure : du .NET et bien au-delà

Java dans Azure ? Oui, et cela marche ! Nous l'avons vu fonctionner ! La session *Developing Java Application With Windows Azure* (http://microsoftpdc.com/Sessions/SVC50) nous a offert plusieurs démonstrations sur l'hébergement d'une application tournant à la fois sur le couple Apache-Tomcat et sur la machine virtuelle Java. Même si l'intégration de Java n'est pas aussi poussée et native que celle de .NET, il y a tout de même un effort notable d'interopérabilité de la part de Microsoft et de ses partenaires : mise à disposition d'une API Java pour manipuler l'Azure Storage, un « Solution Accelerator » pour faciliter le déploiement de Tomcat, etc. De véritables scénarios de production sont maintenant envisageables ! Une session à voir d'urgence si le sujet vous intéresse.

Visual Studio 2010 Ultimate et TFS 2010

Ils arrivent bientôt (vers mars 2010) et n'occupent donc pas le devant de la scène : le couple VS 2010 Ultimate et TFS 2010 offrent de belles perspectives pour les équipes de développement, surtout quand la chaîne complète de développement est intégrée et fonctionne automatiquement. Les speakers de la session Automating « Done Done » in Team Workflows with VS Ultimate 2010 and TFS 2010 (http://microsoftpdc.com/ Sessions/P09-02) proposent de monter une chaîne complète d'outillage basée sur la suite Visual Studio ALM en moins d'une heure :

- Build automatique avec le nouveau Team Build reposant sur Workflow Foundation
- Intégration continue avec les Gated Checkins (publication automatique du code seulement après réussite de la compilation et des tests unitaires)
- Tests d'interface utilisateur (Coded UI Tests) pour assurer les tests fonctionnels mais aussi la non-régression lors du développement
- Mise en place des environnements de test à la demande grâce au module Team Lab Management. Ce

dernier composant permet de construire un ou plusieurs environnements virtuels afin d'y déployer et exécuter les différents tests.

Les usines logicielles deviennent de plus en plus concrètes et complètes avec cette nouvelle version de l'IDE de Microsoft. La RTM nous offrira sans doute encore plus de possibilités que ce que l'on peut déjà voir ici dans cette session.

Silverlight 4 : disponibilité de la bêta 1

L'une des annonces phares de cette PDC 2009 fut sans aucun doute celle de Silverlight 4.0 Bêta qui apporte un lot de nouvelles fonctionnalités non négligeables tendant à rompre de plus en plus la frontière entre le RIA et RDA ! Les nouveautés qui seront ainsi apportées à Silverligth 4 ont été travaillées sur 3 axes majeurs : « Media », « Business Applications » et « Beyond the Browser » !

Constatez par vous-même au travers d'une liste non exhaustive de ces nouvelles fonctionnalités :

- Le designer XAML Silverlight qui fait sa réapparition au sein de Visual Studio 2010
- La mise à disposition d'une API d'impression permettant d'imprimer des écrans Silverlight
- La gestion du clic droit ainsi que du menu contextuel
- L'accès non négligeable aux Webcams et Micros permettant d'étendre de manière radicale les usages de la technologie Silverlight 4 dans le domaine des réseaux sociaux.
- La gestion du scroll de la souris (MouseWheel)
- Le contrôle de saisie de texte riche (RichTextArea) qui était plus qu'attendu
- L'arrivée de l'ICommand sur ButtonBase et l'Hyperlink
- Le gestionnaire de copier / coller (Clipboard)
- Le WebBrowserControl permettant de charger du HTML au sein de Silverlight... imaginez charger vos applications flash à terme au sein de SL4.
- La mise en œuvre d'une nouvelle fonctionnalité permettant à Silverlight Out Of Browser d'avoir des droits plus élevés lui permettant d'accéder par exemple à des répertoires et fichiers locaux
- L'interop COM : Gestion de fichiers Excel par exemple
- La barre de notification pour Silverlight Out of Browser (Toast) à la AIR
- Le TextTrimming permettant enfin d'afficher du texte lisible dans Silverlight
- La gestion du text RTL permettant maintenant d'afficher votre texte de droite à gauche
- Le ViewBox control déjà connu des utilisateurs des Toolkits
- La gestion du H.264 PlayReady dans le cadre des DRM
- Le DependencyObject Binding
- Ou encore le support de Google Chrome

Retrouvez l'ensemble des nouveautés de Silverlight 4 depuis l'adresse suivante : http://www.silverlight.net/getstarted/silverlight-4-beta

Après avoir parcouru les nouveautés mentionnées ci-dessus, force sera de constater la stratégie mise en œuvre par Microsoft mettant à disposition de tout un chacun l'ensemble des outils nécessaires à créer la nouvelle génération d'applications sur la plateforme Software + Services (avec Windows Azure) tant annoncée et présentée par Microsoft au-cours de ces dernières années !!

Ces nouveautés de Silverlight 4 apportent aux développeurs la possibilité de créer des applications capables de fonctionner aussi bien en mode online que offline sans changement de technologies RIA / RDA.

Souvenez-vous en définitive de la présentation des offices web applications lors de la PDC 2008, le temps est-il maintenant venu pour les éditeurs de logiciels de s'intéresser plus fortement à cette nouvelle génération d'applications ? La question mérite d'être plus que soulevée et prise en compte ! En attendant la conférence web du Microsoft Mix 2010 qui devrait dévoiler la version mobile de Silverlight...

L'évolution des conteneurs cloud Microsoft

Microsoft a également dévoilé la prochaine génération des conteneurs Windows Azure, et force est de constater que ceux-ci comportent de nombreuses évolutions majeures ! Effectivement, l'une des avancées étant que ce nouveau type de bloc d'hébergement de nos futurs data center, dit de 4e génération, possèdera une capacité modulaire et d'évolution permanente de par la construction progressive par ajout de conteneurs.

Cette nouvelle stratégie de constitution des centres d'hébergement de type nuages (Cloud) porte le nom de PAC (Pre-Assembled Components), cette stratégie devra permettre ainsi à Microsoft d'optimiser les coûts de construction de ceux-ci et surtout l'évolution pragmatique et en cohérence avec les besoins du marché. Chaque conteneur contenant 455 unités et le tout est alimenté par du 420 Volts (en mode plug & play) !

Ces blocs d'hébergement (conteneurs) de 4e génération semblent être clairement optimisés pour une utilisation en extérieur avec une conception basée sur l'air frais, « free cooling », plutôt que sur une climatisation non conforme avec les orientations Green-IT. Le tout étant basé sur une exploitation de l'eau déversée par petites quantités et gestion de l'air ambiante permettant ainsi la stabilisation de la température entre 20 et 25 degrés. Ces conteneurs devraient être déployés dans les installations Windows Azure de Chicago, San Antonio, Dublin, Amsterdam, Singapour et Hong Kong. Chicago devant accueillir à terme 112 conteneurs d'une contenance totale de 224 000 serveurs. Microsoft annonce ainsi avec cette 4e génération vouloir réduire ses investissements de 20 à 40 % et ses délais de déploiement de centre de données de 18 mois à moins de 6 mois.

Vivement 2010 !

2009 est l'année de la confirmation pour Microsoft : cette PDC n'aura pas vu de grosses annonces comme en 2008 mais plein de nouveautés cohérentes et en phase avec la stratégie amorcée il y a un an. En conclusion : vivement 2010 !

■ *Par nos envoyés spéciaux Grégory Renard (alias Rédo) et Guillaume Belmas, Wygwam.*

Flex 4 et le skinning de composants d'interfaces riches

Flex 4 est en bêta depuis plusieurs mois. Les principaux changements par rapport à la version 3 sont nés des échanges entre Adobe et la communauté. Ils portent principalement sur une nouvelle architecture des composants et leur skinning. L'objectif de Flex 4 est de maintenir la compatibilité avec les développements Flex 3, cependant la nouvelle architecture de composants de Flex 4 représente clairement le futur du framework.

Première différence de taille, les applications Flex 4 sont compilées pour être exécutées par la version 10 du Flash Player. En Septembre 2009, plus de 93% des ordinateurs ont déjà installé la version 10. On peut donc penser que lorsque le nouveau framework sortira au printemps 2010, la quasi-totalité des ordinateurs seront prêts à exécuter des RIA Flex 4.

Quelques nouveaux tags permettent de mieux structurer son code MXML. Le tag <fx :Declarations> permet de définir une zone de déclaration d'éléments non visuels de l'application. C'est dans celui-ci que l'on déclare désormais les RemoteObjets, ou les Web Services utilisés par l'interface.

Le tag <fx :Private> stocke de l'information qui sera ignorée par le compilateur comme les informations sur le développeur ou sur la version de l'application. Le nouvel outil de Design interactif Flash catalyst utilise cette zone pour stocker de l'information utile pour optimiser le workflow entre le designer et le développeur. Le tag <fx :Library> est utilisé pour stocker des éléments graphiques qui ne seront instanciés qu'à leur utilisation.

Layout

Dans Flex 3, le développeur agence des composants en fonction du container. Le Canvas permet de placer des éléments en coordonnées (x,y), le container VBox pour enchaîner verticalement les composants et HBox pour une mise en forme horizontale. Le nouveau tag layout de Flex 3 propose ces trois agencements sans tenir compte du composant père. Cela donne un contrôle total de l'agencement des composants enfants d'une application. Autre avantage, un layout peut être défini, modifié ou supprimé au runtime pour un composant. Désormais, pour facilement afficher verticalement deux boutons, on utilisera cette syntaxe :

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
 xmlns:s="library://ns.adobe.com/flex/spark"
 xmlns:mx="library://ns.adobe.com/flex/halo">

 <s:layout>
   <s:VerticalLayout/>
 </s:layout>

 <s:Button label="button one"/>
 <s:Button label="button two"/>

</s:Application>
```

Vous pouvez créer vos propres classes de layout. Cet exemple montre comment créer un Layout circulaire. Les composants s'afficheront en suivant le contour d'un cercle.

```
package
{
 import mx.core.ILayoutElement;

 import spark.layouts.supportClasses.LayoutBase;

 public class CircularLayout extends LayoutBase
 {
   override public function updateDisplayList(w:Number, h:Number):void
   {
     super.updateDisplayList(w, h);

     if (!target)
       return;

     var layoutElement:ILayoutElement;
     var count:uint = target.numElements;

     var angle : Number = 360/count;
     var radius : Number = Math.min( target.width/2, target
.height/2 ) - 25;

     var w2 : Number = target.width/2;
     var h2 : Number = target.height/2;

     for (var i:int = 0; i < count; i++)
     {
       layoutElement = target.getElementAt(i);

       if (!layoutElement || !layoutElement.includeInLayout)
         continue;

       var radAngle : Number = (angle * i) * (Math.PI / 180) ;

       var _x : Number = Math.sin( radAngle );
       var _y : Number = - Math.cos( radAngle );

       layoutElement.setLayoutBoundsPosition( w2 + (_x * radius)
- 25, h2 + (_y * radius) - 10 );
     }
```

```
    }
  }
}
```

Pour l'utiliser dans une application, il suffit de l'instancier avec le tag layout :

```
<s:layout>
      <local:CircularLayout />
</s:layout>
```

La gestion des états

La gestion des états, ou states, a grandement été revue et améliorée avec Flex 4. Dans Flex 3, les « states » décrivent comment des éléments visuels doivent être ajoutés ou supprimés de la couche visuelle, en modifiant des styles, des propriétés et des événements. Ces états sont très simples à gérer avec Flex Builder, car visuellement, il suffit d'agencer les vues et le développeur peut voir les différences en temps réel. Par contre le code généré est assez illisible et difficile à maintenir.

Avec Flex 4, la définition des states ne se fait plus dans une zone spécifique du code, avec ajout ou suppression par programmation des composants. Les composants existent dans le code quoi qu'il arrive, et ils hébergent des propriétés qui les rattachent à certains states.

Comme en Flex 3, chaque état est défini avec un tag State qui contient des nœuds states. Les méthodes *AddChild* et *RemoveChild* dans les états de Flex 3 ont été remplacées par des attributs *includeIn* et *excludeFrom* directement dans les tags MXML. Une nouvelle syntaxe « propertyName.stateName » remplace les méthodes *SetProperty* et *SetStyle* qui rendaient le code trop verbeux. Je reviendrai dans l'article sur cette nouvelle syntaxe lors du skinning d'un bouton.

L'objectif principal de Flex 4 est d'améliorer l'expérience de skinning, et plus globalement les échanges entre designers et développeurs de RIA. Le framework fournit du coup une séparation claire entre les vues et le comportement d'un même composant. Avec Flex 3, le même code pour un composant inclut sa logique comportementale et visuelle. Les composants Flex 4 sont factorisés en plusieurs classes qui contiennent des éléments de comportement et la gestion de l'apparence.

Une classe principale, qui représente aussi le nom du composant (utilisé comme nom du tag MXML) encapsule le comportement de base : les événements émis par le composant, le modèle de données, les connexions aux sous-composants, et la gestion des états internes du composant. Cette classe principale est couplée avec une classe de skin qui gère l'apparence visuelle du composant, les éléments graphiques, la mise en forme, la représentation de la donnée, les transitions entre les états.

Skinning avec Flex 4

Le FXG est un nouveau langage de déclaration de vecteurs graphiques en XML. Il est nativement interprété par le Flash Player et par de nombreux outils de la gamme Adobe (dont Illustrator). Il est désormais possible de décrire en code XML un bouton avec une forme rectangulaire.

J'ai dessiné dans Illustrator un bouton « Pro » aux couleurs du magazine Programmez. Après un export en FXG, je peux copier-coller directement le résultat dans mon projet Flex 4.

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application>

 <s:Group verticalCenter="0" horizontalCenter="0">
   <s:Path data="M 69.054 37.607 L 0.5 37.607 L 21.884 0.5 L
90.437 0.5 L 69.054 37.607 Z"   winding="nonZero" x="0" y="0">
     <s:fill>
       <s:LinearGradient rotation="90" scaleX="17.9275" x="45.
834" y="14.022">
         <s:GradientEntry color="0xFF2400" ratio="0"/>
         <s:GradientEntry color="0xD01E00" ratio="0.993865"/>
       </s:LinearGradient>
     </s:fill>
     <s:stroke>
       <s:SolidColorStroke caps="none" color="0xB50000" joints
="miter" miterLimit="4" scaleMode="normal" weight="1"/>
     </s:stroke>
   </s:Path>
   <s:RichText color="0xFFFFFF" fontFamily="Arial Black" font
Size="21" kerning="on" lineHeight="120%"  whiteSpaceCollapse=
"preserve" x="22" y="11" >
     <s:content><s:p  whiteSpaceCollapse="collapse"><s:span >PRO<
/s:span></s:p></s:content>
   </s:RichText>
 </s:Group>
</s:Application>
```

Résultat :

Bien que la syntaxe reste très proche de celle de Flex 3, quelques nouveaux composants apparaissent. L'espace de nom « s » correspondant au nouveau modèle de composant baptisé Spark. Le container « Group » est l'élément le plus simple pour encapsuler des éléments graphiques. Des primitives graphiques comme *Rect*, *Circle* ou ici *Path* sont accessibles pour dessiner des formes en FXG (les spécifications sont en ligne sur opensource.adobe.com).

Pour convertir cette forme en composant (un bouton dans l'exemple), il suffit de transformer cette description graphique en classe de skin, puis d'associer cette skin à un composant bouton. Cette nouvelle architecture autorise la séparation de la vue et des comportements au sein même de la définition d'un composant. Cette approche était impossible en Flex 3 et fait toute la richesse de la nouvelle version du framework. Pour transformer cette description en classe de skin, il suffit de rajouter une déclaration de quatre états propres au composant Button (up, over, down et disabled).

La classe de skin buttonSkin.mxml :

```
<?xml version="1.0" encoding="utf-8"?>
<s:Skin xmlns:fx="http://ns.adobe.com/mxml/2009"
    xmlns:s="library://ns.adobe.com/flex/spark"
     xmlns:mx="library://ns.adobe.com/flex/halo"  width="400"
 height="300">
<s:states>
 <s:State name="up" />
 <s:State name="over" />
 <s:State name="down" />
 <s:State name="disabled" />
</s:states>
```

```
 <s:Path data="M 69.054 37.607 L 0.5 37.607 L 21.884 0.5 L 90.
437 0.5 L 69.054 37.607 Z"   winding="nonZero" x="0" y="0">
   <s:fill>
     <s:LinearGradient rotation="90" scaleX="17.9275" x="45.
834" y="14.022">
       <s:GradientEntry color="0xFF2400" ratio="0"/>
       <s:GradientEntry color="0xD01E00" ratio="0.993865"/>
     </s:LinearGradient>
   </s:fill>
   <s:stroke>
     <s:SolidColorStroke caps="none" color="0xB50000" joints=
"miter" miterLimit="4" scaleMode="normal" weight="1"/>
   </s:stroke>
 </s:Path>
 <s:RichText color="0xFFFFFF" fontFamily="Arial Black" fontSize
="21" kerning="on" lineHeight="120%"  whiteSpaceCollapse="preserve"
x="22" y="11" >
   <s:content><s:p  whiteSpaceCollapse="collapse"><s:span
>PRO</s:span></s:p></s:content>
 </s:RichText>
</s:Skin>
```

L'application Main.mxml du coup n'implémente que l'ajout d'un composant Bouton associé à notre classe de skin :

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
         xmlns:s="library://ns.adobe.com/flex/spark"
         xmlns:mx="library://ns.adobe.com/flex/halo" minWidth
="1024" minHeight="768">

 <s:Button x="14" y="15" label="Button" skinClass="buttonSkin"/>
</s:Application>
```

Création de skins dynamiques

Cette classe de skin présente peu d'intérêt dans son état actuel. Il faut l'enrichir pour gérer les interactions avec le composant. Un « contrat » est donc passé entre le composant et sa skin. Le contrat est divisé en trois sections : les états (states), les données (data), et les parties (parts). Si dans le composant je définis une variable « titre », alors la skin peut y accéder par la propriété hostComponent.titre grâce au contrat.

Pour enrichir les states d'un composant, je vais juste rajouter une réaction quand la souris passe sur notre bouton. Dans la classe de skin, je vais juste modifier le tracé rouge en fond du bouton pour faire apparaître une ombre au passage de la souris, et le rendre un peu transparent.

```
<s:Path data="M 69.054 37.607 L 0.5 37.607 L 21.884 0.5 L 90.
437 0.5 L 69.054 37.607 Z"  winding="nonZero" x="0" y="0" alpha.
over="0.7">
   <s:filters>
     <s:DropShadowFilter includeIn="over" knockout="false" blurX=
"5" blurY="5"
                alpha="0.32" distance="2" />
   </s:filters>
   <s:fill>
     <s:LinearGradient rotation="90" scaleX="17.9275" x="45.
834" y="14.022">
       <s:GradientEntry color="0xFF2400" ratio="0"/>
       <s:GradientEntry color="0xD01E00" ratio="0.993865"/>
     </s:LinearGradient>
   </s:fill>

   <s:stroke>
     <s:SolidColorStroke caps="none" color="0xB50000" joints=
"miter" miterLimit="4" scaleMode="normal" weight="1"/>
   </s:stroke>
</s:Path>
```

Etat « over »:

Deux astuces sont ici employées pour définir des changements visuels sur le composant au passage de la souris. L'emploi du point après une propriété permet de l'assigner à un état : alpha.over, ne sera interprétée que lors du survol de la souris. L'autre astuce consiste à employer les propriétés *includeIn* ou *excludeFrom* dans un composant. Pour rendre désormais le label dynamique, je recommande l'utilisation de la métadonnée *HostComponent* dans la classe de skin. Rajoutez ces lignes dans la classe de skin :

```
<fx:Metadata>
 [HostComponent("spark.components.Button")]
</fx:Metadata>
…
<s:RichText color="0xFFFFFF" fontFamily="Arial Black" fontSize=
"21" kerning="on" lineHeight="120%"  whiteSpaceCollapse="preserve"
x="22" y="11" >
<s:content><s:p  whiteSpaceCollapse="collapse"><s:span>{host
Component.label} </s:span></s:p></s:content></s:RichText>
```

C'est la méthode la plus élégante pour associer de la donnée depuis un composant. Je rajoute dans mon application main.mxml deux boutons avec deux labels différents qui seront pris en charge par la skin.

```
<s:Button  x="14"  y="15"   label="PRO"  skinClass="buttonSkin"
width="145" height="56"/>
<s:Button  x="14"  y="79"   label="AMF"  skinClass="buttonSkin"
width="145" height="56"/>
```

Une autre approche consiste à renseigner les parties attendues par le composant (parts). Un bouton par exemple attend une partie « labelDisplay » : le composant texte qui va accueillir l'affichage du label. Si vous remplacez le bloc <s :RichText> de la classe de skin par un label à qui vous donnez l'id labelDisplay, le composant va automatiquement l'associer à la partie qui affiche la propriété label.

```
<s:Label id="labelDisplay" x="22" y="10" color="0xFFFFFF" font
Family="Arial Black" fontSize="21"/>
```

Le bon usage des parts simplifie le skinning de composants, spécialement celui de composants plus complexes comme un slider ou une barre de défilement. Un composant slider attend par exemple deux parties « thumb » et « track » (la piste).

Michael **Chaize**
Adobe France, Consultant

Base de données

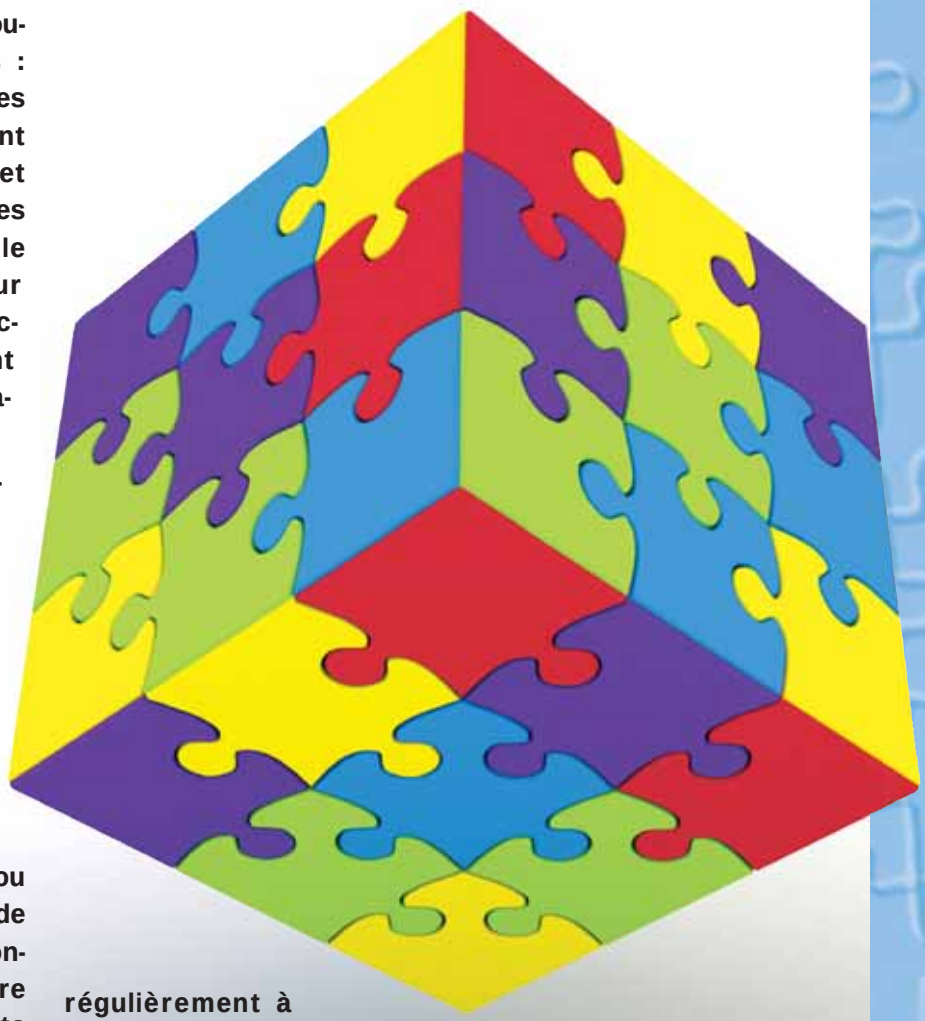
Choisir et optimiser

Le marché des bases de données est toujours dominé par trois gros éditeurs : Oracle, IBM et Microsoft. Mais les bases de données open source / libres se taillent une belle place auprès des entreprises et développeurs, notamment pour les sites web où MySQL a su s'imposer avec le fameux modèle LAMP. Par contre, pour les volumétries de données ou les secteurs critiques, les entreprises restent fidèles aux valeurs traditionnelles des leaders historiques.

Si le marché semble stable, ponctué par les nouvelles versions, les nouvelles fonctions, le monde open source est secoué par l'offensive de certaines bases (Ingres, PostgreSQL) ou encore par les incertitudes concernant l'avenir, le positionnement de MySQL depuis la volonté de rachat de Sun par Oracle et la multiplication des forks MySQL, dont le plus ambitieux est MariaDB.

Mais au-delà de ces querelles, motivées ou non, une des questions récurrentes est de savoir comment choisir sa base de données. L'un des dilemmes se situant entre MySQL et PostgreSQL. Les deux projets possèdent leurs atouts propres et finalement, tout va dépendre du contexte mais effectivement, on a tendance à trop choisir MySQL par défaut alors qu'il est loin d'être le seul !

Les autres bases ne seront pas oubliées pour autant. Nous reviendrons sur un débat technique et fonctionnel que l'on croyait enterré, mais qui revient régulièrement à la surface : faut-il choisir une base objet ou une base relationnelle (et faussement objet) ? Pour des utilisations critiques où les exigences de performances sont primordiales, l'approche objet reste la meilleure. Enfin, nous reviendrons sur les tendances actuelles et futures dans les bases de données.



François Tonic

MySQL et ses forks, comment choisir

MySQL au côté d'Oracle ? Cette hypothèse aurait pu faire sourire il y a quelques années. Depuis le rachat de MySQL par SUN puis de SUN par Oracle c'est en passe de devenir une réalité. Quelles conséquences pour les utilisateurs, quelles alternatives ? Nous allons au cours de cet article essayer de présenter les différentes options.

MySQL est un système de gestion de bases de données (SGBD) dont les principaux atouts sont la rapidité, la robustesse et la facilité d'utilisation. Son moteur est basé sur la norme ANSI SQL 92, avec cependant quelques fonctions spécifiques. Il est disponible sous deux licences, la licence GPL (General Public License) des projets GNU et FSF (Free Software Foundation) et une licence propriétaire moins contraignante mais payante. Pour permettre à la société MySQL AB de vendre des licences il lui faut détenir toutes les sources du logiciel, c'est l'une des raisons qui font que certains patchs externes ne peuvent pas être repris.

La société MySQL AB a été rachetée le 16 janvier 2008 par Sun Microsystems pour un milliard de dollars. Le 20 avril 2009, Oracle Corporation annonce racheter Sun Microsystems pour 7,4 milliards de dollars, créant la crainte de voir MySQL disparaître - ou du moins ne plus être développé - au profit d'Oracle, le SGBD d'Oracle Database. MySQL est cependant loin d'avoir toutes les fonctionnalités d'Oracle, et pourrait donc être vu au contraire par la société comme un produit gratuit d'appel, banalisant l'usage de SQL et préparant donc ses futures ventes.

Considérations géopolitiques

Oracle, propriétaire de MySQL ? Si la commission Européenne valide le rachat de SUN par Oracle le 27 Janvier prochain ce sera le cas. La question que l'on peut légitimement se poser est : *« quel intérêt pour Oracle de maintenir des concurrents Open Source à ses solutions ? »*. A priori quasiment aucun si l'on en croit Michael Widenius qui lors du Forum PHP martelait : *« MySQL fait perdre un milliard par an à Oracle et rapporte au mieux 100 millions »*. Effectivement, Oracle a beau être une très belle base de données, sur de nombreux cas d'utilisation une base plus légère comme MySQL peut très bien convenir. En suivant ce raisonnement on pourrait même envisager que MySQL aurait pu continuer son développement et à terme concurrencer encore plus Oracle.

Pour les clients c'était une aubaine, nombre d'entre eux utilisaient la menace de migration vers MySQL pour obtenir des rabais substantiels sur les tarifs d'Oracle. Au mieux, on peut penser qu'Oracle continuera le développement de MySQL en le limitant à une utilisation Web. Cette stratégie permettrait alors de couper les ailes de MySQL sans le tuer.

La réponse de la communauté ne s'est pas fait attendre, Michael Widenius, l'un des créateurs de MySQL a initié un projet OpenSource nommé MariaDB dont il sera question plus loin. Ce projet a attiré plusieurs des principaux Core Développeurs de MySQL.

La meilleure solution pour MySQL serait de ne pas appartenir à Oracle mais à une entité indépendante qui pourrait assurer sa survie et son développement.

Les versions officielles portées par SUN et la communauté MySQL

Autant le dire tout de suite, à ce jour il n'y a pas d'offre aussi fournie que celle liée à la version officielle de MySQL. Nous verrons dans cet article que des alternatives se construisent et ont des avantages ciblés.

Avantages

La base de données MySQL dispose d'une communauté d'utilisateurs extrêmement large ; c'est ce qui fait dire à la société MySQL AB que sa base de données est « la plus populaire au monde ». Effectivement, rien qu'en comptabilisant tous les sites Internet tournant sur le fameux couple LAMP (Linux Apache MySQL PHP) les chiffres s'envolent. Le rachat récent par SUN a également permis d'améliorer les interactions entre Java et MySQL.

Cette communauté permet de trouver facilement de la documentation et de l'aide qui vont au-delà de ce qu'on peut trouver sur le site officiel. On ne compte pas les forums, blogs ou autres wikis animés par des utilisateurs de MySQL, et ce, dans de nombreuses langues. Toutes sortes d'interrogations peuvent trouver une réponse avec la communauté, des plus simples (comment se connecter au serveur ?) aux plus complexes (mes performances vont-elles s'améliorer si je recompile le serveur avec des pages InnoDB de 8ko au lieu de 16ko ?)

La disponibilité d'installeurs pour les plates-formes les plus courantes rend simple la plupart des déploiements. Pour les plates-formes un peu plus exotiques, des binaires sont souvent disponibles. Enfin, en dernier recours ou par choix délibéré, le code source de la version communautaire reste téléchargeable et peut être compilé.

Pour ceux qui ne pourraient pas adop-

ter la version communautaire, Sun / MySQL propose une version « Enterprise » avec une licence commerciale. Cette version fournit également du support et des outils d'administration, par exemple le MySQL Query Analyzer dont le but est d'aider à trouver et à corriger les requêtes lentes ou l'Enterprise Monitor qui permet de faire du monitoring des instances installées.

Historiquement, MySQL a été conçu comme étant un SGBD pour le Web, qui vise d'abord la facilité d'installation et d'administration avant de viser le respect des normes et l'étendue des fonctionnalités. C'est cette logique qui a permis de conquérir des parts de marché. Puis, petit à petit au fil des versions, la liste des fonctionnalités s'est enrichie pour se rapprocher des SGBD traditionnels comme SQL Server, PostgreSQL ou Oracle. Mais l'essence même de MySQL, à savoir cette simplicité dans l'utilisation, a toujours été préservée.

Limites

L'une des limites de MySQL est liée à sa politique de licence. Effectivement pour être en mesure de vendre des versions non Open Source il faut que la société détienne toutes les sources du logiciel. Cela implique que la version officielle de MySQL Entreprise ne peut inclure certaines améliorations apportées par Google (les fameux Google patchs) ou d'autres comme ceux proposés par Percona.

De plus, la politique de maintenance consiste à geler au bout de 2 ans les évolutions pour les versions décrétées stables (GA pour suivre le jargon de MySQL). Seuls des correctifs de sécurité sont publiés régulièrement. C'est ce qui explique que des fonctionnalités développées initialement par des membres de la communauté sous forme de patchs sont disponibles sur la version 5.1 mais pas sur la version 5.0 encore largement utilisée. C'est le cas de la commande SHOW PROFILES qui permet de collecter des informations très précieuses sur les requêtes qu'on cherche à optimiser.

On note par ailleurs une lenteur chronique pour intégrer les patchs provenant de la communauté. La version 5.4, qui embarque nombre de patchs mais reste en bêta à l'heure actuelle, tente de corriger le tir sur ce point.

InnoDB est le moteur à la pointe de MySQL, donnant au SGBD la plupart des fonctionnalités que tout développeur ou administrateur attend d'un SGBD : clés étrangères, transactions, respect des conditions ACID… Oui, mais InnoDB se fait vieillissant et de plus en plus d'applications sont confrontées à des problèmes de scalabilité avec ce moteur. InnoDB a en effet été conçu à une époque où les machines avaient des disques lents, peu de mémoire et un seul processeur, et éprouve des difficultés à tirer parti du matériel le plus moderne.

MySQL s'est voulu dès son origine facile à utiliser et à administrer. C'était effectivement le cas sur la première génération d'applications ayant popularisé MySQL (les sites personnels, les applications métier légères). Aujourd'hui, il n'est pas rare de voir MySQL traiter des téra-octets de données pour des applications critiques. Avec une telle volumétrie, il est impératif que le serveur expose le plus de paramètres possible pour que les administrateurs puissent maintenir les bases de données en condition opérationnelle.

Face à toutes ces limitations, les membres les plus actifs de la communauté MySQL ont démarré plusieurs projets visant à rendre encore meilleur leur SGBD préféré ou à corriger des anomalies qui tardent à être résolues. Dressons-en un panorama.

Les versions MySQL communautaires améliorées

OurDelta

OurDelta est un projet qui a pour but de mettre à disposition sous licence GPL des distributions optimisées de MySQL 5.0 et MariaDB 5.1, dont nous parlons un peu plus loin. Leur travail consiste essentiellement à développer et maintenir des patchs (une vingtaine environ en fonction des versions) qui ont leur branche de code indépendante (disponible sur Launchpad). Ces produits sont disponibles prêts à l'usage sur les systèmes Linux grâce aux outils yum et apt-get. Parmi les contributeurs on compte Google et Percona. Chacun de ces patchs résout ou optimise une fonctionnalité bien particulière et s'adresse donc aux administrateurs de base de données chevronnés. On retrouve notamment un patch qui permet de tuer une connexion uniquement si celle-ci est inutilisée (KILL IF_IDLE), ou encore un patch qui propose une version étendue des statistiques concernant les requêtes lentes (slow.log).

Toutes les informations concernant les versions produites par OurDelta sont disponibles à l'adresse : http://ourdelta.org/

Percona

Percona fournit des versions de MySQL 5.0 incluant là-encore de nombreux patchs par rapport à la version officielle. Ces patchs sont soit développés directement par Percona, soit par OpenQuery ou encore Google (et sans aucun doute très prochainement Facebook). Ces versions proposent également des paramétrages non disponibles dans les versions officielles et susceptibles de fournir des gains de performance appréciables pour les applications à très forte charge sur la base de données.

La liste des versions disponibles se trouve à l'adresse : http://www.percona.com/docs/wiki/release:start

Malgré tout, certains développeurs ne se satisfont pas de la direction que prend MySQL et ont donc décidé de coder des modifications profondes, créant ainsi de véritables forks et plus seulement des versions améliorées.

Les futures alternatives

MariaDB

Initié par les créateurs de MySQL, MariaDB est une branche de développement de la version officielle de MySQL maintenue par la communauté. MariaDB correspond à la distribution de MySQL utilisant le moteur Maria. L'objectif de MariaDB est de fournir une alternative libre et indé-

pendante de MySQL tout en assurant une compatibilité importante avec celle-ci. Pour cette raison, dans la plupart des cas, ce SGBD fonctionnera de manière analogue à MySQL. L'intégralité des commandes, interfaces et bibliothèques sont disponibles.
L'intérêt de Maria réside donc essentiellement au niveau du gain de performances. Les requêtes complexes s'exécutent plus rapidement et on constate une meilleure réactivité. Il est également reconnu que Maria réduit le nombre d'anomalies bloquantes.

Quelques fonctionnalités particulières font également leur apparition comme au niveau des statistiques où l'on peut apprécier l'affichage de la liste des processus en microsecondes. Depuis fin octobre, l'équipe travaille sur la Release Candidate, jalon ultime avant la version stable. Des packages seront bientôt disponibles pour Linux.
Nous avons rencontré le créateur de MariaDB à l'occasion du forum PHP et celui-ci nous à indiqué que beaucoup de développeurs du projet originel MySQL avaient rejoint sa société dans le but de promouvoir MariaDB.
Site : http://www.mariadb.org

Drizzle

Le projet Drizzle est défini par ses concepteurs comme un SGBD léger dérivé de MySQL dont le but est de revenir aux idées originales de MySQL : performances, fiabilité et facilité d'utilisation et d'administration. Drizzle est très orienté applications Web et cloud computing, c'est-à-dire des cas où les accès concurrents sont massifs et où réplication et *sharding* sont courants. Les développeurs ont voulu dès le départ que le SGBD soit capable de gérer au mieux la mémoire disponible et capable d'exploiter les architectures multi-core qui sont aujourd'hui courantes.
Pour espérer parvenir à un logiciel le plus léger possible, le code repris de MySQL a été réécrit en adoptant une architecture de type micro-noyau et surtout la volonté de supprimer le code correspondant à tout ce qui était jugé non indispensable. En conséquence, la plupart des fonctionnalités introduites par MySQL 4 et 5 ont été enlevées du noyau, telles que les vues, les triggers, les procédures stockées ou encore le cache de requêtes ! Cela ne signifie pas que ces fonctionnalités n'existeront jamais avec Drizzle, mais plutôt que si elles sont implémentées, elles le seront sous la forme de plugins. Il existe d'ailleurs déjà des ébauches de tels plugins pour les vues ou les triggers.
La gestion des droits est organisée de façon originale puisque les tables de droits ont été supprimées pour laisser la place à une authentification LDAP / PAM. D'autres simplifications sont également notables : InnoDB est le seul moteur de stockage disponible (bien que MyISAM soit également encore compilé), les types de données sont nettement moins nombreux que sous MySQL et seul UTF-8 est disponible pour stocker les données textuelles.

Le projet a démarré courant 2008 et reste à l'état alpha, aucune date de sortie de version stable n'étant envisagée à présent. D'après certaines rumeurs le rapprochement Oracle / SUN pourrait avoir mis à mal la réalisation de ce projet. On peut effectivement s'interroger sur l'intérêt qu'aurait Oracle à développer ce type de super base de données en version OpenSource.

Pour installer et tester Drizzle, il faut d'abord compiler les sources. Drizzle peut être installé sur Linux, Mac OS X et Solaris. Aucune version n'est disponible pour Windows et un tel portage n'est pas prévu à l'heure actuelle.
L'installation en elle-même ne pose pas de problème particulier dès que toutes les dépendances sont satisfaites (la liste est publiée sur le site du projet). Il faut simplement penser à compiler libdrizzle puis drizzle.

Dans l'utilisation courante, à l'exception des simplifications mentionnées plus haut, les commandes SQL utilisables avec MySQL sont également utilisables avec Drizzle.

Conclusion / perspective

Pour tous ceux qui débutent avec MySQL, il apparaît clairement que les versions officielles sont recommandées. Le choix entre la version communautaire et la version Enterprise sera le plus souvent dicté par des critères économiques ou par la sensibilité à l'open source.
Les utilisateurs avancés aimeront certainement essayer MariaDB ou Drizzle afin de mieux comprendre ce que peut faire ou ce que ne peut pas faire la version officielle ou encore de se projeter dans le futur. Néanmoins, ces versions / forks de MySQL ne sont pas aujourd'hui dans un état permettant d'envisager un passage en production à court terme. On attend beaucoup de MariaDB qui devrait annoncer sous peu ses premières versions de production.
Quant aux versions améliorées, si elles sont, elles aussi, réservées aux utilisateurs avisés capables de comprendre en quoi l'inclusion de patch peut leur procurer de meilleures performances, ce sont déjà des versions utilisées en production et sur des applications de grande envergure. Pourquoi alors ne pas tenter de franchir le pas si des tests de performance montrent que le jeu peut en valoir la chandelle ?

Avec l'amicale participation d'Olivier Dasini

Loïc Guillois est développeur Web indépendant et contributeur actif du framework de développement Play!

Cyril Pierre de Geyer est un architecte OpenSource, co-auteur du livre "PHP 5 avancé", et cofondateur des groupes d'utilisateurs MySQL (LeMug.fr) et PHP (AFUP).

Stéphane Combaudon est DBA MySQL et membre du MySQL User Group France(LeMug.fr).

Gestion par règles et le *resource governor* de SQL Server 2008

Dans ce dossier nous allons traiter de deux nouvelles fonctionnalités qui ont fait leur apparition avec la version 2008 de SQL Server. Ces fonctionnalités sont fortement liées à l'administration de SQL Server et notamment pour les entreprises qui doivent faire face chaque jour à des problématiques toujours plus nombreuses et toujours plus complexes.

Nous vous proposons ainsi une fonctionnalité qui a pour but de gérer de manière intelligente et définie l'allocation des ressources de vos serveurs à telle ou telle application ou tel ou tel groupe / utilisateur. Cela permet de répondre aux problématiques d'accès multiples aux sources de données, en priorisant les accès et traitements. Cette fonctionnalité se nomme « gouverneur de ressources » ou encore « resource governor ». La seconde fonctionnalité permet quant à elle de définir des règles de gestion que l'on peut régir unitairement ou de manière centralisée. Ces règles peuvent ainsi augmenter l'intégrité et la sécurité des données manipulées par exemple par les développeurs, comme définir des règles pour industrialiser toujours plus l'usage de vos instances SQL Server 2008. Le nom de cette fonctionnalité parle de lui-même on parle de « gestion par règle » ou « policy-based management ».

Pour information, ces fonctionnalités fortement appréciées de nos clients sont bien évidemment présentes dans la nouvelle mouture SQL Server 2008 R2.

Le gouverneur de ressources

L'essence même du gouverneur de ressources provient des problématiques bien connues des accès parallèles à des sources de données. Il devient en effet compliqué dans certaines modélisations de cloisonner toutes les sources de données. En effet, lorsque l'on a un datamart de finance à la fois interrogé par un département pour du reporting mais aussi pour des traitements de type insertion / mise à jour qui se font en même temps, il n'est pas possible de les prioriser. On peut également citer en exemple les accès par un DSI à des rapports et des traitements divers qui se font sur une base commune, il peut s'avérer nécessaire de gérer les ressources de façon à donner priorité à l'accès au rapport, quitte à ralentir les autres traitements. Le gouverneur de ressources ne se contente pas de diviser l'allocation des ressources (CPU et mémoire) entre les utilisateurs, il est capable de donner plus de ressources que le quota alloué en cas de ressources non utilisées durant une période T. Cette fonctionnalité introduit trois nouveaux concepts.

Resource pools

Les pools servent à déterminer les restrictions de minimum et maximum que l'on peut allouer par rapport aux ressources physiques du serveur. Ces pools peuvent être considérés comme des instances virtuelles avec leurs propres ressources CPU / mémoire au sein d'une instance physique. La limite est de 18 pools personnalisés en plus des 2 pools système créés à l'installation de SQL Server 2008.

- INTERNAL : ce pool système est utilisé pour les processus internes à SQL Server 2008
- DEFAULT : pool système utilisé pour les charges qui ne seraient pas affectées à un groupe de charge (workload group)

A noter que les pools système ont pour caractéristique d'avoir une plus grande priorité sur les pools personnalisés. Pour déterminer les minimums et maximums quelques règles à suivre :

- Minimum : la somme des minimums entre les pools ne doit pas excéder 100%
- Maximum : entre le minimum du pool spécifié et 100%

Exemple de calcul

Pools	Min %	Max %	Max réel %
Pool A	10	70	70
Pool B	5	100	90
Pool C	0	100	85

Le maximum réel qui peut être utilisé par un pool déterminé est calculé en prenant son maximum alloué moins la somme des minimums des autres pools.

Groupes de charge – Workload groups

C'est le composant indispensable pour lier une charge à un pool de ressources. Un administrateur peut d'ailleurs décider de déplacer un groupe de charge vers un nouveau pool de ressources en cas de changement de priorité ou tout simplement pour augmenter / diminuer les ressources allouées.

Tout comme les pools de ressources, il existe des groupes de charge internes (DEFAULT & INTERNAL).

Fonction de classification

Fonction déterminée par l'administrateur pour aiguiller les requêtes vers les bons groupes de charge. A noter qu'il faut être prudent lors de l'implémentation de cette fonction car elle va être utilisée de manière importante par le mécanisme du gouverneur de ressources.

Modélisation voir [Fig.1]

Implémentation

Avant toute chose, il est fortement recommandé d'autoriser les connexions DAC (Dedicated Admin Connection) pour toujours permettre l'accès à vos instances SQL Server 2008. Pourquoi ? La connexion DAC n'est pas sujette à la fonction de classification qui pour rappel est appelée à chaque initialisation de session, cela s'avère donc extrêmement pratique pour débuguer.

SQL Server Management Studio

La première chose à faire est d'activer le gouverneur de res-

sources. Cela se fait très simplement, l'explorateur d'objet une fois connecté à une instance relationnelle. Dans la partie gestion, vous trouvez l'objet gouverneur de ressources, un clic droit puis activer permettra d'autoriser son fonctionnement.
La seconde étape correspond à la création d'un pool de ressource, clic droit, ajouter un nouveau pool de ressources. [Fig.2]
Dans cette fenêtre, on peut définir les différents pools de ressources dont on a besoin et faire les affectations de ressources CPU / mémoire correspondantes. Pour chacun des pools, on peut définir les groupes de charge avec notamment le niveau de priorité, le nombre maximum de requêtes... Initialement, dans la liste déroulante de fonctions de classification, aucune fonction n'est définie. Pour cela, il suffit de définir une fonction de type scalar et de l'affecter ensuite au gouverneur de ressources dans cette même fenêtre.

Scripts

Une autre manière est bien entendu de scripter ces traitements par des requêtes SQL. Ainsi pour modifier des propriétés d'un pool ou d'un groupe de charge, on procédera de la manière suivante (exemple fait sur le pool default) :

```
USE [master];
GO

ALTER RESOURCE POOL [default] WITH
(
 MAX_MEMORY_PERCENT = 15,
 MAX_CPU_PERCENT = 45
);

ALTER WORKLOAD GROUP [default] WITH
(
 MAX_DOP = 1,
 REQUEST_MAX_CPU_TIME_SEC = 400
);
```

Pour créer un pool de ressource et un groupe de charge associé :

```
CREATE RESOURCE POOL [MonPoolA];

CREATE WORKLOAD GROUP [MonGroupeA] USING [MonPoolA];
GO
```

La dernière étape concerne la création de la fonction de classification et la mise à jour du gouverneur de ressources pour appliquer les changements. Dans cet exemple, on spécifie que si le nom de l'application ou que tel utilisateur est reconnu, alors on adresse la requête vers le groupe de charge MonGroupeA, sinon vers le pool interne *default*.

```
CREATE FUNCTION dbo.MaFonction()
RETURNS SYSNAME
WITH SCHEMABINDING
AS
BEGIN
 RETURN
(
   SELECT CASE
     WHEN APP_NAME() = N'NomApplicationLambda'
     OR SUSER_SNAME() IN ('utilisateur1','utilisateur2',
'utilisateur3')
   THEN N'MonGroupeA'
   ELSE N'default'
   END
);
END
GO

ALTER RESOURCE GOVERNOR WITH
(CLASSIFIER_FUNCTION = dbo.classifier_PreventRunaway);
GO

ALTER RESOURCE GOVERNOR RECONFIGURE;
GO
```

Bonnes pratiques

- Vérifier bien que votre connexion DAC fonctionne correctement
- Bien optimiser le traitement de la fonction de classification
- Utiliser les évènements PreConnect :Starting et PreConnect :Completed dans le Profiler pour identifier les traces générées par la fonction de classification
- Eviter les opérations répétées sur les chaînes de caractères dans la fonction de classification
- Déterminer judicieusement l'allocation des ressources, ne pas hésiter à vérifier au préalable à l'aide de la DMV sys.dm_os_performance_counters l'utilisation des ressources.
- Ne pas hésiter à utiliser les DMV dédiées au gouverneur de ressources comme sys.resource_governor_workload_groups

Pourquoi utiliser le gouverneur de ressources ?

Cette fonctionnalité vous permettra de définir finement l'utilisation des ressources par vos utilisateurs et bases sur votre serveur. Cela peut éviter notamment des mauvaises surprises en cas de charge importante, que des utilisateurs importants ne puissent plus avoir accès aux données, une situation de saturation du serveur... Comme on le sait, ces problèmes sont de plus en plus courants notamment avec la consolidation des parcs que l'on voit de

Fig.1

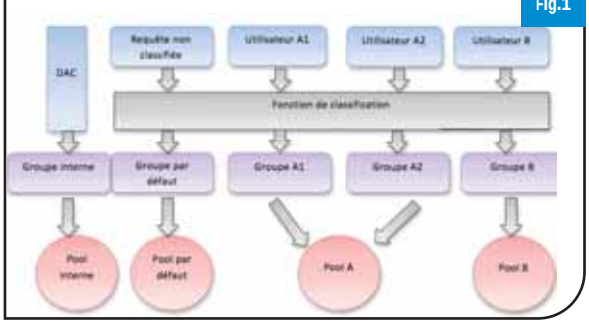


Fig.2

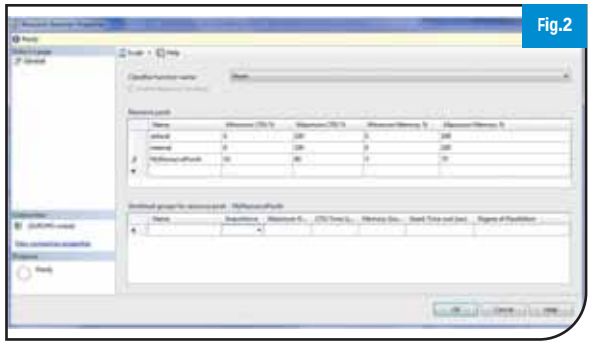
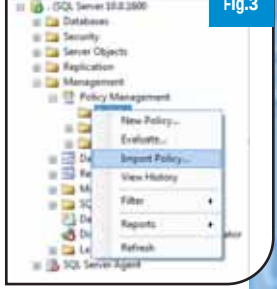


Fig.3



Explorateur d'objets SSMS

plus en plus. On cherche à optimiser logiquement l'utilisation des serveurs et cela nous amène forcément à des situations où les ressources sont mieux utilisées au point de venir frôler la ligne rouge en termes de saturation en cas de charge générale. Il devient donc primordial de bien gérer les allocations des ressources. Les administrateurs peuvent également mieux prévoir leur travail d'optimisation en ayant spécifié des valeurs minimales pour tel ou tel pool utilisé par telle ou telle base. Enfin l'usage du gouverneur de ressources permet aussi de mieux monitorer et prendre les bonnes décisions quant aux usages des ressources.

La gestion par règle

La seconde problématique que tend à résoudre SQL Server 2008 est de pouvoir standardiser son parc de serveurs SQL Server 2008 via une gestion par règle. Cette gestion permet par exemple de définir de manière centralisée un ensemble de règles et conventions défini par l'entreprise pour administrer ses objets SQL Server. En effet, dans la plupart des entreprises, il devient difficile de gérer des dizaines de serveurs avec N instances et déployés dans différents endroits géographiques. Il s'avère ainsi souvent difficile de définir des règles globales et de les voir appliquées à un niveau local, le scénario typique est avec des filiales.

Tant bien que mal on peut définir des règles de base, les mettre en place mais comment vérifier qu'elles sont bien respectées, comment gérer la mise à jour d'une règle et son application ? Ce sont ces problèmes que la gestion par règle résout.

Quelques cas d'utilisation :

- Gérer la sécurité en spécifiant par exemple que tel groupe de serveurs doit utiliser tel mode d'authentification et pas un autre
- Définir des conventions de nommage pour les objets SQL Server et les faire appliquer globalement ou sur un groupe de serveurs / instances
- Gestion des serveurs / instance par exemple en vérifiant quelles instances ont tel service pack déployé

Sémantique

Tout d'abord, cette fonctionnalité intègre de nouvelles notions qu'il est bon de définir avant d'aller plus loin sur l'implémentation.

Managed Targets

Ce sont les objets qui sont gérés par la gestion par règle, typiquement on retrouve :

- Instances SQL Server
- Bases de données
- Tables
- Index
- …

On peut bien entendu créer des groupes pour ces types d'objets comme un groupe formé par les procédures stockées dont le nom commence par « usp_ ».

Facets

Cela représente l'ensemble d'un groupe de propriétés qui s'appliquent à des objets en particulier (managed targets). Par exemple la facet « Server Information » s'applique aux objets de type Server et contient des propriétés comme « Collation », « IsClustered », « IsSingleUser »…

Conditions

Les conditions représentent l'état attendu pour une target ou un groupe de targets. L'expression d'une condition inclut généralement un opérateur de comparaison, un champ et une valeur. On peut bien entendu combiner plusieurs expressions pour représenter une condition. Par exemple, pour le respect d'une norme de nommage, on définit que les vues doivent avoir le champ « @Name » qui commence par « vw_ » et que ces vues ne doivent pas être des vues système :

```
@Name LIKE 'vm%'
AND @IsSystemObject = false
```

Règles ou policies

Les règles sont des entités que l'on active ou désactive. Une règle ne représente qu'une seule condition. Pour faciliter l'usage, on peut grouper ces règles par catégorie. Il suffit ensuite de s'abonner à ces règles afin de les appliquer.

Les modes d'évaluation

Une règle â pour but d'être évaluée par le moteur de règles et cette évaluation peut se faire de 4 manières différentes :

- On demand : mode où on demande explicitement l'évaluation d'une règle
- On change :prevent : utilisation des triggers DDL, on fait l'évaluation avant de valider la modification
- On change :log only : utilisation des triggers DDL, on autorise les changements mais en enregistrant l'événement des logs
- On schedule : on planifie l'évaluation via le SQL Server Agent

Utiliser les règles prédéfinies

Au lieu de réinventer la roue, problème bien connu en informatique, vous pouvez commencer votre implémentation des règles en utilisant des règles déjà définies et fournies avec SQL Server. Depuis l'explorateur d'objets dans SQL Server Management Studio 2008, dans le répertoire Management de votre instance relationnelle, déplier l'item « Policy Management », clic droit puis « Import Policy… ». [Fig.3] Depuis le chemin suivant nous avons la possibilité d'importer les règles correspondant aux bonnes pratiques de :

- SQL Server
- Analysis Services
- Reporting Services

```
C:\Program Files (x86)\Microsoft SQL Server\100\Tools\Policies
```

Depuis la fenêtre d'import il suffit de se rendre dans le répertoire des règles, mentionné précédemment. Dans cet exemple, nous avons choisi d'importer l'ensemble des bonnes pratiques SQL Server. [Fig.4 et 5]. Une fois importées, les règles apparaissent ainsi dans l'explorateur d'objets. Vous pouvez également déplier les facets et conditions et, bien entendu, les personnaliser. [Fig.6]

En double-cliquant sur l'une de ces règles, par exemple « SQL Server Login Mode », une boîte de dialogue s'affiche pour présenter les options suivantes pour pouvoir appliquer la règle :

- La target ou groupe de targets
- Le type d'évaluation de la règle
- La possibilité d'exclure des targets répondant à une condition particulière, par exemple exclure les serveurs dont le nom comporte tel ou tel ID.

On remarque au passage la hiérarchie des règles par groupe, ici « Server Security » [Fig.7]

En ouvrant la condition, on voit quelle expression a été utilisée pour définir cette condition : [Fig.8]

Créer ses propres règles

Maintenant que nous avons vu comment importer des règles, nous allons voir comment créer les nôtres. Vous noterez au passage que les règles définies peuvent évidemment s'exporter et ainsi être importées ailleurs. Particulièrement intéressant pour gérer ses règles entre différents environnements.

Notre règle va permettre de mettre en place une convention de nommage sur la création de bases de données sur nos serveurs. Ces noms de base devront commencer par « myDB_ ».

La première étape consiste à créer notre condition. On la nomme « Nommage Bases » et l'expression est la suivante : @Name LIKE 'myDB_%'. Sélectionnez ensuite « bases de données » comme facet pour l'appliquer aux objets bases. [Fig.9]

Il nous faut maintenant créer notre règle qui se base sur la condition qui vient d'être définie. Outre définir un nom pertinent pour la règle, on va aller chercher notre condition qui doit être classée parmi les conditions de la facet « bases de données ». [Fig.10]

Une fois la condition sélectionnée, la target se positionne automatiquement sur la facet définie dans la condition, c'est-à-dire « bases de données ». On définit maintenant que cette règle sera évaluée à la demande et pas d'exclusion particulière. [Fig.11]

L'objectif est maintenant d'évaluer cette règle sur les serveurs souhaités, nous allons commencer localement et nous verrons ensuite pour gérer les règles sur le parc.

Pour évaluer une règle, vous pouvez lancer l'évaluation des règles de votre serveur ou sélectionner une règle particulière, clic-droit puis évaluer. Une fenêtre de résultats va alors s'afficher. Dans mon exemple, on voit deux erreurs sur les bases « ReportServer » et « ReportServerTempDB » qui ne respectent effectivement pas la règle définie.

Déployer et gérer mes règles pour N serveurs

La version 2008 de SQL Server introduit une nouvelle fonctionnalité nommée «Central Management Servers ». A noter que le serveur défini comme « Central Management Servers » ne fera pas partie des évaluations des règles. Il faut donc utiliser un serveur dédié pour mettre en place les règles et les évaluer sur l'ensemble du parc auquel on souhaite les appliquer. Dans l'onglet « Registered servers » dans Management Studio, vous commencez par définir le serveur qui sera le « Central Management Server ». Dans la fenêtre il suffit d'indiquer le nom du serveur et quelques informations de connexion.

La seconde étape consiste à ajouter les serveurs que l'on souhaite évaluer. A noter que l'on peut ajouter les serveurs un par un et créer des groupes de serveurs. [Fig.12]. Une fois les serveurs ajoutés, on peut lancer les évaluations de règles. A ce moment précis, il faut préciser où se trouve la source des règles qui peut être à la fois des fichiers ou une connexion à une instance SQL Server. L'évaluation peut alors commencer. [Fig.13]. Grâce à cette méthode il devient alors simple de gérer tout ou partie de son parc avec des règles homogènes et distribuées. Cet outil devrait s'avérer être très utile pour l'industrialisation des développements SQL Server mais aussi pour Analysis Services ou encore Reporting Services.

■ Vincent **Bellet**
Consultant - Application Platform Optimization
Microsoft France - Enterprise Services


Fig.4
Parcourir les fichiers de type "règle" format XML


Fig.5
Sélectionner les fichiers de type "règle" format XML

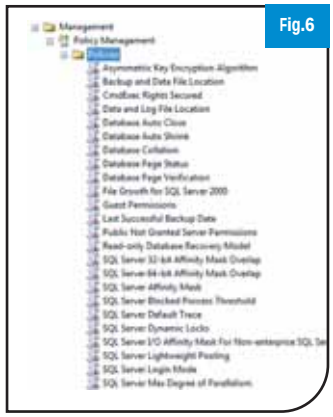


Fig.6
Liste des règles dans SSMS

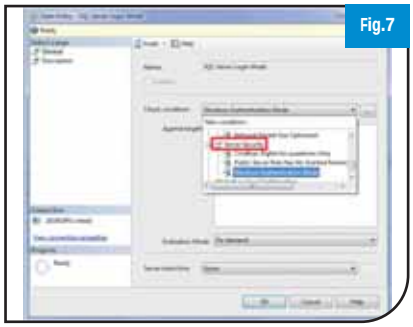


Fig.7


Fig.8


Fig.9

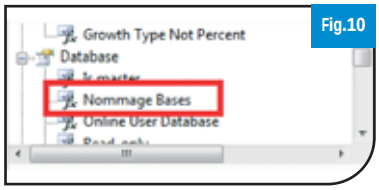


Fig.10

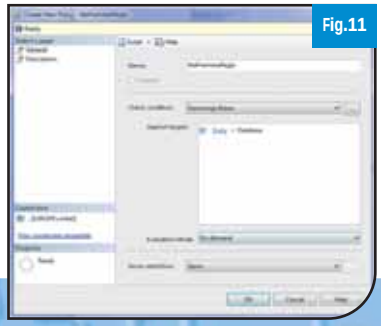


Fig.11

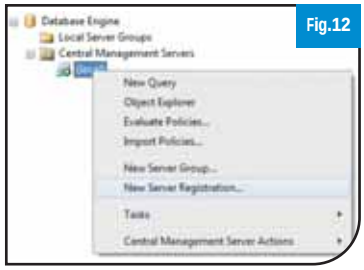


Fig.12


Fig.13

Fonctions avancées de PostgreSQL et MySQL

Dans cet article nous allons comparer les bases de données MySQL et PostgreSQL sous trois angles : le partitionnement des données, l'interopérabilité et les outils d'administration.

Le partitionnement est une technique utilisée avec des tables volumineuses pour organiser les données stockées sur le disque. La base de données peut ensuite exploiter cette organisation pour minimiser les accès au disque, ce qui contribue à améliorer les performances. Dans cette partie nous détaillerons les moyens à mettre en œuvre pour utiliser le partitionnement avec MySQL et PostgreSQL. L'interopérabilité détermine la faculté d'une base de données à être insérée dans une architecture existante et à opérer avec d'autres bases. Nous regarderons donc quelles facilités MySQL et PostgreSQL proposent pour s'intégrer dans un existant. Enfin, nous finirons notre comparaison avec un rappel des outils d'administration disponibles pour ces deux bases.

Partitionnement : optimiser vos tables

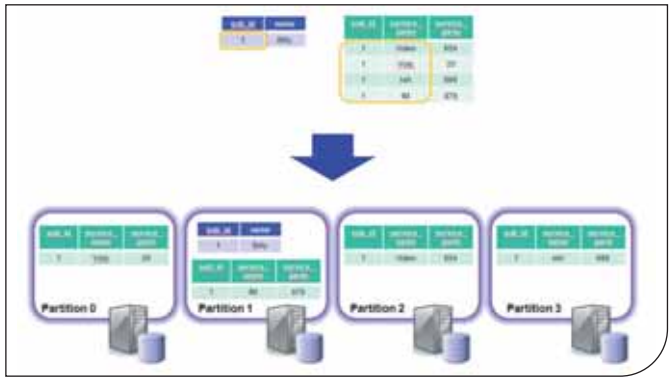
Pour une base de données, le partitionnement est la possibilité de découper une table en plusieurs petites tables. Ce découpage peut être fait selon des règles que l'utilisateur définit en fonction de la nature des données à stocker. Cette fonctionnalité est apparue sous PostgreSQL depuis la version 8.1 (2005). Elle n'a été supportée par MySQL qu'à partir de la version 5.1 (2008). Le but de cette section est d'étudier et comparer la mise en place du partitionnement de tables pour ces deux bases de données.

Il est important de noter que le partitionnement ne peut pas se substituer au travail d'optimisation des requêtes. L'optimisation doit être faite tout le temps alors que le partitionnement permet d'augmenter les performances... pour des requêtes déjà optimisées ! MySQL offre la possibilité de partitionner les tables selon plusieurs critères. Il doit être conçu en fonction des requêtes longues et complexes afin de réduire le nombre d'enregistrements sur lesquels elles portent. MySQL permet de partitionner une table horizontalement ou verticalement.

Le partitionnement horizontal consiste à découper une table en lignes et à définir un critère de groupage afin de stocker chaque groupe sur une partition différente.

Le partitionnement vertical consiste à découper une table en colonnes. Les clés doivent être présentes dans toutes les partitions pour permettre de retrouver les données originales à l'aide des jointures entre les différentes partitions. Dans la suite, nous nous intéresserons au partitionnement horizontal.

Le partitionnement sous MySQL est facile à mettre en place : seules quelques commandes suffisent pour scinder une table en partitions. Nous avons choisi un exemple simple pour illustrer la



mise en place du partitionnement sous MySQL et le même exemple sera utilisé sous PostgreSQL.

Nous avons choisi une table simple qui gère les ventes de glaces par date et par ville. Nous souhaitons stocker ces données avec la date des ventes comme critère car la plupart des requêtes consistent à calculer la somme des ventes sur chaque année.

Une table classique

Voici la structure de la table *mesure* sans partition qui nous sert d'exemple :

```
CREATE TABLE mesure (
    id_ville         int not null,
    date_trace       date not null,
    temperature      int,
    ventes           int
) ;
create index idx_date_trace on mesure(date_trace);
```

Pour alimenter la table *mesure* nous avons créé une procédure stockée proc_mesure.sql qui insère des données aléatoires dans cette table. Nous insérons un million d'enregistrements à l'aide de la ligne de commande suivante :

```
mysql>call insert_mesure(1000000) ;
```

Pour avoir des informations sur le plan d'exécution de la requête, nous utilisons le mot clé explain de cette manière :

```
mysql> explain partitions SELECT sum(ventes) from mesure where
date_trace >= '2004-01-01' and date_trace < '2005-01-01'\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: mesure
   partitions: NULL
         type: range
possible_keys: idx_date_trace
          key: idx_date_trace
```

```
       key_len: 3
           ref: NULL
          rows: 77882
         Extra: Using where
1 row in set (0,00 sec)
```

Comme la table *mesure* n'est pas partitionnée, nous constatons que le mot clé *partitions* est égal à NULL. Pour montrer l'intérêt du partitionnement, nous feront des tests sur des tables avec et sans partitions. Pour cela nous utiliserons deux requêtes. La première calcule la somme des ventes sur l'année 2004 c'est-à-dire dans le cas le plus favorable où la requête accède à une seule partition.

```
mysql> SELECT sum(ventes) from mesure where  date_trace >= '2004
-01-01' and date_trace < '2005-01-01';
+-------------+
| sum(ventes) |
+-------------+
|    41478016 |
+-------------+
1 row in set (1,05 sec)
```

La seconde requête calcule la somme des ventes sur plusieurs années pour tester un cas moins favorable.

```
mysql> SELECT sum(ventes) from mesure where date_trace < '2009
-01-01' and date_trace > '2004-01-01';
+-------------+
| sum(ventes) |
+-------------+
|  1038423038 |
+-------------+
1 row in set (1,27 sec)
```

Mise en œuvre du partitionnement MySQL

Nous reprenons la structure de la table précédente et nous lui ajoutons des partitions. Voici la nouvelle structure de la table avec sept partitions.

```
CREATE TABLE mesure (
    id_ville         int not null,
    date_trace       date not null,
    temperature      int,
    ventes           int
) Partition BY RANGE(YEAR(date_trace))
(
PARTITION p2004 VALUES LESS THAN(2005),
PARTITION p2005 VALUES LESS THAN(2006),
PARTITION p2006 VALUES LESS THAN(2007),
PARTITION p2007 VALUES LESS THAN(2008),
PARTITION p2008 VALUES LESS THAN(2009),
PARTITION p2009 VALUES LESS THAN(2010),
PARTITION p2010 VALUES LESS MAXVALUE
);
create index idx_date_trace on mesure(date_trace);
```

Une fois cette nouvelle table remplie, voyons le nouveau plan d'exécution proposé par MySQL :

```
mysql> explain partitions SELECT sum(ventes) from mesure where
date_trace >= '2004-01-01' and date_trace < '2005-01-01'\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: mesure
   partitions: p2004
         type: ALL
possible_keys: idx_date_trace
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 1000000
        Extra: Using where
1 row in set (0,00 sec)
```

On constate que l'optimiseur de MySQL a choisi de ne parcourir que la partition p2004 susceptible de contenir les données.

```
mysql> SELECT sum(ventes) from mesure where  date_trace >= '2004
-01-01' and date_trace < '2005-01-01';
+-------------+
| sum(ventes) |
+-------------+
|    41478016 |
+-------------+
1 row in set (0,18 sec)
```

Le temps de réponse de cette requête est 5,8 fois plus petit que celui de la même requête exécutée sur la table non partitionnée. Regardons maintenant le cas moins favorable de notre requête pluriannuelle.

```
mysql> explain partitions SELECT sum(ventes) from mesure where
date_trace < '2009-01-01' and date_trace > '2004-01-01'\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: mesure
   partitions: p2004,p2005,p2006,p2007,p2008
         type: ALL
possible_keys: idx_date_trace
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 456222
        Extra: Using where
1 row in set (0,00 sec)
```

Le plan d'exécution indique toutes les partitions qui seront lues.

```
mysql> SELECT sum(ventes) from mesure where date_trace < '2009
-01-01' and date_trace > '2004-01-01';
+-------------+
| sum(ventes) |
+-------------+
|  1038423038 |
+-------------+
1 row in set (0,82 sec)
```

Nous constatons que le partitionnement ci-dessus convient aux requêtes qui portent sur les ventes d'une année et permet également d'améliorer les performances des requêtes qui parcourent plusieurs partitions.

Le partitionnement avec PostgreSQL

Le partitionnement sous PostgreSQL est connu sous le nom de l'héritage et c'est le même concept que l'héritage dans les langages orientés objet. Il consiste à scinder une table volumineuse en plusieurs petites tables. La table mère contient les champs communs à toutes les tables. Les tables filles contiennent les données en fonctions des critères des requêtes et héritent de toutes les colonnes de la table parente.
Reprenons notre exemple précédent pour illustrer le partitionnement sous PostgreSQL.

Création de la table mère et des tables filles

Sous PostgreSQL, les partitions sont des tables à part entière et ne font pas partie de la structure de la table d'origine.

```
CREATE TABLE mesure (
    id_ville         int not null,
    date_trace       date not null,
    temperature      int,
    ventes           int
);
CREATE TABLE mesure_a2004 (
 CHECK ( date_trace >= DATE '2004-01-01' AND date_trace < DATE
'2005-01-01' )
) INHERITS (mesure);
CREATE TABLE mesure_a2005 (
 CHECK ( date_trace >= DATE '2005-01-01' AND date_trace < DATE
'2006-01-01' )
) INHERITS (mesure);
CREATE TABLE mesure_a2006 (
 CHECK ( date_trace >= DATE '2006-01-01' AND date_trace < DATE
'2007-01-01' )
) INHERITS (mesure);
CREATE TABLE mesure_a2007 (
 CHECK ( date_trace >= DATE '2007-01-01' AND date_trace < DATE
'2008-01-01' )
) INHERITS (mesure);
CREATE TABLE mesure_a2008 (
 CHECK ( date_trace >= DATE '2008-01-01' AND date_trace < DATE
'2009-01-01' )
) INHERITS (mesure);
CREATE TABLE mesure_a2009 (
 CHECK ( date_trace >= DATE '2009-01-01' AND date_trace < DATE
'2010-01-01' )
) INHERITS (mesure);
CREATE TABLE mesure_a2010 (
 CHECK ( date_trace >= DATE '2010-01-01')
) INHERITS (mesure);

--Création des indexes
CREATE INDEX mesure_a2004_date_trace ON mesure_a2004 (date_trace);
CREATE INDEX mesure_a2005_date_trace ON mesure_a2005 (date_trace);
CREATE INDEX mesure_a2006_date_trace ON mesure_a2006 (date_trace);
CREATE INDEX mesure_a2007_date_trace ON mesure_a2007 (date_trace);
CREATE INDEX mesure_a2008_date_trace ON mesure_a2008 (date_trace);
CREATE INDEX mesure_a2009_date_trace ON mesure_a2009 (date_trace);
CREATE INDEX mesure_a2010_date_trace ON mesure_a2010 (date_trace);
```

Les tables filles sont créées avec le mot clé INHERITS(mesure) qui signifie que ces tables héritent de la table *mesure* et récupèrent ainsi toutes ses colonnes. Des index sont également créés sur le champ date_trace pour chaque partition. Une requête peut être limitée à la table mère en lui ajoutant le mot clé ONLY comme dans l'exemple ci-dessous :

```
SELECT  *  FROM ONLY mesure ;
```

On peut supprimer une table fille du partage en la gardant comme une table autonome en utilisant la commande SQL suivante :

```
ALTER TABLE table_fille NO INHERIT mesure;
```

Ou en la supprimant définitivement à l'aide de cette commande :

```
DROP TABLE table_fille ;
```

Les données insérées dans la table mère ne sont pas redirigées automatiquement vers la table fille concernée. Pour cela, il est nécessaire de mettre en place une fonction et un trigger.

Insertion des enregistrements dans les tables filles

La procédure stockée utilisée pour alimenter le table *mesure* sous MySQL a été adaptée pour alimenter la table *mesure* sous PostgreSQL. Exécutons là avant la mise en place du trigger.

```
partition#call insert_mesure(1000000) ;

partition=# select count(*) from mesure;
  count
---------
1000000
(1 row)

partition=# select * from mesure_a2007;
 id_ville | date_trace | temperature | ventes
----------+------------+-------------+--------
(0 rows)
```

Nous constatons que toutes les données sont insérées dans la table mère et que les tables filles sont vides. Déclarons maintenant la fonction d'insertion…

```
CREATE OR REPLACE FUNCTION mesure_insert_trigger()
RETURNS TRIGGER AS $$
BEGIN
    IF ( NEW.date_trace >= DATE '2004-01-01' AND
         NEW.date_trace < DATE '2005-01-01' ) THEN
        INSERT INTO mesure_a2004 VALUES (NEW.*);
    ELSIF ( NEW.date_trace >= DATE '2005-01-01' AND
            NEW.date_trace < DATE '2006-01-01' ) THEN
        INSERT INTO mesure_a2005 VALUES (NEW.*);
    ELSIF ( NEW.date_trace >= DATE '2006-01-01' AND
            NEW.date_trace < DATE '2007-01-01' ) THEN
        INSERT INTO mesure_a2006 VALUES (NEW.*);
```

```
    ELSIF ( NEW.date_trace >= DATE '2007-01-01' AND
             NEW.date_trace < DATE '2008-01-01' ) THEN
        INSERT INTO mesure_a2007 VALUES (NEW.*);
    ELSIF ( NEW.date_trace >= DATE '2008-01-01' AND
             NEW.date_trace < DATE '2009-01-01' ) THEN
        INSERT INTO mesure_a2008 VALUES (NEW.*);
    ELSIF ( NEW.date_trace >= DATE '2009-01-01' AND
             NEW.date_trace < DATE '2010-01-01' ) THEN
        INSERT INTO mesure_a2009 VALUES (NEW.*);
    ELSIF ( NEW.date_trace >= DATE '2010-01-01' THEN
        INSERT INTO mesure_a2010 VALUES (NEW.*);
    ELSE
        RAISE EXCEPTION 'Date en dehors !';
    END IF;
    RETURN NULL;
END;
$$
LANGUAGE plpgsql;
```

... et le trigger :

```
CREATE TRIGGER insert_mesure_trigger
    BEFORE INSERT ON mesure
    FOR EACH ROW EXECUTE PROCEDURE mesure_insert_trigger();
```

Avant chaque insertion dans la table *mesure*, le trigger insert_ mesure_trigger se déclenche et appelle la fonction mesure_ insert_trigger() qui permet d'insérer les données dans la table fille correspondant au bon intervalle de date. L'exemple ci-dessous montre que quand le trigger est en place, les données insérées dans une table partitionnée sont redirigées directement dans les tables filles.

```
partition=# select count(*) from only mesure;
 count
-------
     0
(1 row)
partition=# select count(*) from only mesure_a2004;
 count
-------
 91246
(1 row)
partition=# select count(*) from only mesure_a2009;
 count
-------
 90706
(1 row)
```

Pour activer l'optimisation par partitionnement sous PostgreSQL, il est nécessaire de vérifier que le paramètre constraint_exclusion est à « on » dans le fichier de configuration PostgreSQL.conf. Ce paramètre permet au planificateur de prouver que les partitions qui ne satisfont pas les critères de la section « where » de la requête ne sont pas concernées par celle-ci. Seules les partitions concernées seront donc parcourues.

Tests de performances

Reprenons nos deux requêtes de somme des ventes pour illustrer le gain avec les partitions PostgreSQL. Somme des ventes pour l'année 2004 sur une table non partitionnée :

```
Partition=# SET constraint_exclusion=on ;
partition=# explain analyze SELECT sum(ventes) from mesure where
date_trace >= '2004-01-01' and date_trace < '2005-01-01';
                        QUERY PLAN
-----------------------------------------------------------
 Aggregate  (cost=20637.96..20637.97 rows=1 width=4) (actual time
=1066.222..1066.223 rows=1 loops=1)
   ->  Seq Scan on mesure  (cost=0.00..20406.00 rows=92782 width
=4) (actual time=0.080..904.743 rows=91246 loops=1)
         Filter: ((date_trace >= '2004-01-01'::date) AND (date
_trace < '2005-01-01'::date))
 Total runtime: 1066.308 ms
(4 rows)

partition=# SELECT sum(ventes) from mesure where  date_trace >
= '2004-01-01' and date_trace < '2005-01-01';
```

temps d'exécution : 995 ms

Somme des ventes sur plusieurs années sur une table non partitionnée :

```
Partition=# explain analyze SELECT sum(ventes) from mesure
where date_trace < '2009-01-01' and date_trace > '2004-01-01';
                        QUERY PLAN
-----------------------------------------------------------
 Aggregate  (cost=21544.02..21544.03 rows=1 width=4) (actual time
=2049.607..2049.608 rows=1 loops=1)
   ->  Seq Scan on mesure  (cost=0.00..20406.00 rows=455206
width=4) (actual time=0.173..1276.384 rows=455646 loops=1)
         Filter: ((date_trace < '2009-01-01'::date) AND (date
_trace > '2004-01-01'::date))
 Total runtime: 2049.710 ms
(4 rows)

Partition=# SELECT sum(ventes) from mesure where date_trace
< '2009-01-01' and date_trace > '2004-01-01';
```

temps d'exécution : 1103 ms

Exécutons les mêmes requêtes sur une table partitionnée.

Somme des ventes sur une année :

```
partition=# explain analyze SELECT sum(ventes) from mesure where
date_trace >= '2004-01-01' and date_trace < '2005-01-01';
                        QUERY PLAN
-----------------------------------------------------------
 Aggregate  (cost=2086.81..2086.82 rows=1 width=4) (actual time
=757.246..757.249 rows=1 loops=1)
   ->  Append  (cost=0.00..1864.50 rows=88922 width=4) (actual
 time=0.040..584.616 rows=91246 loops=1)
         ->  Seq Scan on mesure  (cost=0.00..36.55 rows=9 width
=4) (actual time=0.004..0.004 rows=0 loops=1)
               Filter: ((date_trace >= '2004-01-01'::date) AND
 (date_trace < '2005-01-01'::date))
          ->  Seq Scan on mesure_a2004 mesure  (cost=0.00..
1827.95 rows=88913 width=4) (actual time=0.029..270.641 rows
=91246 loops=1)
               Filter: ((date_trace >= '2004-01-01'::date) AND
 (date_trace < '2005-01-01'::date))
 Total runtime: 757.361 ms
(7 rows)
```

```
partition=# SELECT sum(ventes) from mesure where  date_trace >
= '2004-01-01' and date_trace < '2005-01-01';
```

temps d'exécution : 162 ms

Le constat est sans appel, seules la table mère et la table fille mesure_a2004 concernées par cette requête sont scannées. Le temps d'exécution de cette requête a été divisé par 6 !
Somme des ventes sur plusieurs années :

```
partition=# explain analyze SELECT sum(ventes) from mesure
where date_trace < '2009-01-01' and date_trace > '2004-01-01';
                      QUERY PLAN
-------------------------------------------------------------
 Aggregate  (cost=10347.03..10347.04 rows=1 width=4) (actual time
=3087.072..3087.074 rows=1 loops=1)
   ->  Append  (cost=0.00..9226.70 rows=448130 width=4) (actual
time=0.122..2371.086 rows=455646 loops=1)
         ->  Seq Scan on mesure  (cost=0.00..36.55 rows=9 width
=4) (actual time=0.005..0.005 rows=0 loops=1)
               Filter: ((date_trace < '2009-01-01'::date) AND
 (date_trace > '2004-01-01'::date))
          ->  Seq Scan on mesure_a2004 mesure  (cost=0.00..
1827.95 rows=88913 width=4) (actual time=0.111..284.506 rows
=90988 loops=1)
               Filter: ((date_trace < '2009-01-01'::date) AND
 (date_trace > '2004-01-01'::date))
          ->  Seq Scan on mesure_a2005 mesure  (cost=0.00..
1813.87 rows=87973 width=4) (actual time=0.063..187.284 rows
=91297 loops=1)
               Filter: ((date_trace < '2009-01-01'::date) AND
 (date_trace > '2004-01-01'::date))
          ->  Seq Scan on mesure_a2006 mesure  (cost=0.00..
1861.92 rows=91177 width=4) (actual time=0.059..187.622 rows
=91261 loops=1)
               Filter: ((date_trace < '2009-01-01'::date) AND
 (date_trace > '2004-01-01'::date))
          ->  Seq Scan on mesure_a2007 mesure  (cost=0.00..
1852.40 rows=90609 width=4) (actual time=0.053..186.164 rows
=91110 loops=1)
               Filter: ((date_trace < '2009-01-01'::date) AND
 (date_trace > '2004-01-01'::date))
          ->  Seq Scan on mesure_a2008 mesure  (cost=0.00..
1834.01 rows=89449 width=4) (actual time=0.055..188.422 rows
=90990 loops=1)
               Filter: ((date_trace < '2009-01-01'::date) AND
 (date_trace > '2004-01-01'::date))
 Total runtime: 3087.331 ms
(15 rows)
partition=# SELECT sum(ventes) from mesure where date_trace
< '2009-01-01' and date_trace > '2004-01-01';
temps d'exécution: 745 ms
```

Grâce au partitionnement le temps d'exécution de cette requête est réduit de 33%.

Interopérabilité entre MySQL et PostgreSQL

L'interopérabilité d'un système de bases de données est la faculté à opérer avec d'autres systèmes. En effet, l'insertion d'une nouvelle architecture dans une infrastructure existante doit permettre d'exploiter les données stockées dans d'autres systèmes ou même cohabiter avec ceux-ci pour permettre une migration de données en douceur.

Avec MySQL

Dans la plupart des cas, la mise en place d'une nouvelle architecture sous un nouveau SGBD nécessite la migration des données de l'ancien système vers le nouveau. Cette étape est souvent délicate et coûteuse.
MySQL dispose des moteurs de stockage iStorage et IBMDB2I afin de faciliter la communication entre une base de données DB2 de IBM et une base de données MySQL. Les données sont gérées par un serveur de bases de données DB2 / 400 et sont accessibles à la fois par les applications PHP ou .NET de MySQL et par les programmes natifs sur AS / 400.
Pour les autres SGBD, MySQL dispose de son kit de migration pour assister les utilisateurs à migrer d'une base de données existante vers une base MySQL. Ce kit est open source et prend en charge la plupart des bases de données du marché.

Avec PostgreSQL

Pour passer d'un SGBD à PostgreSQL en version open source, il est nécessaire de passer par l'étape de la migration. Seul un script Perl existe pour aider à migrer une base de données Oracle vers une base de données PostgreSQL. Toutes les procédures stockées, les triggers et les fonctions doivent être réécrites car le langage pl / sql d'Oracle n'est pas supporté par PostgreSQL. Ils peuvent être réécrits soit en pl / pgsql ou avec d'autres langages compatibles avec PostgreSQL.
PostgreSQL en version entreprise (PostgreSQL plus advanced server) est compatible avec le serveur Oracle, il peut cohabiter avec lui dans la même architecture. On peut même répliquer une base de données Oracle vers une base PostgreSQL.
Cette fonctionnalité permet à PostgreSQL de s'insérer dans une architecture Oracle existante afin de planifier la migration des données, et de remplacer des serveurs Oracle par PostgreSQL. Cette version supporte le langage de développement pl / sql d'Oracle et la migration ne nécessite aucune réécriture du code.

Les outils d'administration

Les deux systèmes de bases de données disposent des interfaces graphiques pour gérer et configurer les serveurs de bases de données. Décrivons brièvement le fonctionnement de certains outils et leurs limitations.

Administrer MySQL

MySQL dispose de l'interface **MySQL Administrator** pour administrer les bases de données d'un serveur et superviser la réplication entre un serveur maître et des serveurs esclaves. Cette interface a ses limites. Elle ne permet pas, par exemple, de mettre à jour une base de données en utilisant les commandes SQL update ou insert.
MySQL Query Browser est un client graphique qui dispose d'un éditeur des ordres sql pour écrire et exécuter des requêtes sur la base de données, mais ne permet pas d'effectuer certaines tâches d'administration.
PHPMyAdmin est un des outils d'administration les plus complets

et les plus célèbres pour MySQL. Il se présente sous la forme d'une interface web écrite en PHP.

Administrer PostgreSQL

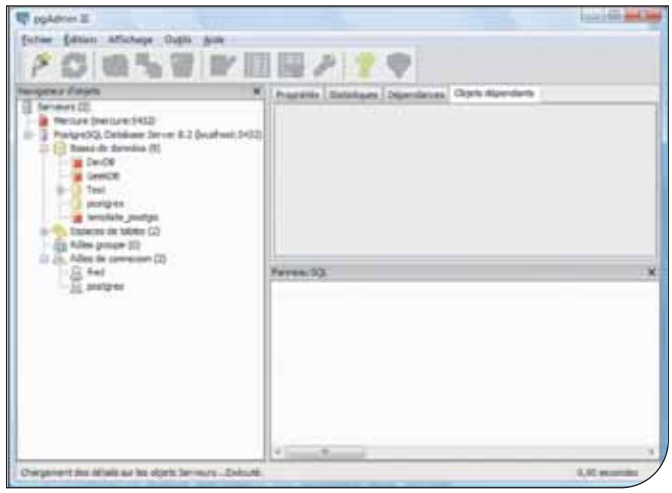
PgAdmin-III est une interface d'administration des serveurs PostgreSQL. Elle comprend une interface graphique d'administration, un outil de requêtes SQL et un éditeur de code procédural. Elle est conçue pour répondre à la plupart des besoins d'un DBA PostgreSQL, depuis l'écriture de simples requêtes SQL jusqu'au développement de bases de données complexes. Cette interface supporte les fonctionnalités de PostgreSQL les plus récentes et rend l'administration PostgreSQL très simple. Pgadmin-III permet également d'administrer plusieurs serveurs de bases de données sur la même interface.

PhpPgAdmin est une interface web d'administration PostgreSQL écrit en PHP et permet d'effectuer toutes les actions nécessaires à l'administration des serveurs de bases de données PostgreSQL.

Conclusion

Le partitionnement sous MySQL est efficace et simple à mettre en place. La redirection des données vers les partitions correspondantes est automatique et ne nécessite aucun développement supplémentaire. Sous PostgreSQL, le partitionnement est implémenté à l'aide de l'héritage au sens des langages objets. Cette architecture est plus complexe à mettre en place et nécessite du développement supplémentaire.

Au travers de ses moteurs iStorage et IBMDB2I, MySQL offre des possibilités d'interopérabilité avancées avec la base DB2. La version entreprise de PostgreSQL de son côté est tournée vers Oracle avec la possibilité de reprendre du code pl / sql. Du côté de la compression de données, MySQL propose des solutions adaptées à plusieurs cas. PostgreSQL n'en propose pas.



Concernant les outils d'administration, on peut noter que PostgreSQL dispose d'une seule interface simple qui permet de faire toutes les actions nécessaires à l'administration et au développement des bases de données PostgreSQL. MySQL dispose de deux interfaces : l'une orientée administration et l'autre destinée aux développeurs.

Lahcen **Ait Ali** - *Devoteam*

Technique d'indexation avancée : les index couvrants

Les performances d'une application, et de la base de données sous-jacente, sont liées en grande partie aux index. Une stratégie d'indexation parfaitement maîtrisée permet d'obtenir de très bonnes performances. De mauvais index ou l'absence d'index engendrent une lenteur perceptible d'un point de vue utilisateur, même si les composantes matérielles (processeur, disque et mémoire) sont largement dimensionnées. Après un bref rappel sur la notion d'index, ordonné et non ordonné, nous aborderons la notion d'index couvrant.

Un index est une structure permettant l'accès rapide à une information contenue dans une table. Tout comme, à la fin d'un livre, l'index permet de savoir quelles pages font référence à un mot bien précis, un index de base de données permet de localiser très rapidement un enregistrement ou un groupe d'enregistrements, sans avoir à parcourir l'ensemble des lignes d'une table. Un index va donc permettre de réduire drastiquement les entrées / sorties disque et les lectures logiques, avec comme résultat un gain de performances.

Un index est constitué d'une ou plusieurs colonnes issues d'une table. Il est représenté sous forme d'un arbre balancé composé d'une page racine, d'une ou plusieurs pages de niveau intermédiaire et de pages de niveau feuille. L'index cluster, ou index ordonné, permet un accès extrêmement rapide aux enregistrements car les pages de niveau feuille sont les pages de données. Cet index va trier les enregistrements de la table suivant cette clé. En conséquence, il ne peut y avoir qu'un seul index cluster par table. [Fig.1] L'index cluster n'est pas obligatoire, si celui-ci n'est pas implémenté sur une table, on parlera alors de segment mémoire, ou de tas. [Fig.2]. L'index non cluster, ou index non ordonné, possède une structure identique avec 16 colonnes et 900 octets au maximum. Les pages de niveau feuille vont soit contenir la clé cluster si un index ordonné existe, soit l'adresse de l'enregistrement dans le cas d'un segment mémoire. Il peut y avoir jusqu'à 249 index non cluster par table. [Fig.3]

Index couvrant

Un index couvrant est un index non ordonné qui va satisfaire en totalité une requête sans avoir à faire référence à la table sous-jacente.

En effet, les requêtes conduisent généralement à deux lectures : une lecture de l'index (accès par identifiant) et une lecture dans la table de données (pour remonter les données supplémentaires).

Un index, reprenant à la fois l'identifiant et les données supplémentaires, est dit couvrant parce que lui seul suffit à récupérer toutes les informations nécessaires au traitement de la requête. Considérons la table *Person* de la base exemple AdventureWorks2008 [Fig.4].

La table contient 19972 enregistrements dont le regroupement par PersonType nous donne : [Fig.5]

Afin de rechercher les 17 personnes de type 'SP', la requête génère 3816 lectures logiques. [Fig.6]

Le plan d'exécution utilise un parcours complet de l'index cluster (Clus-

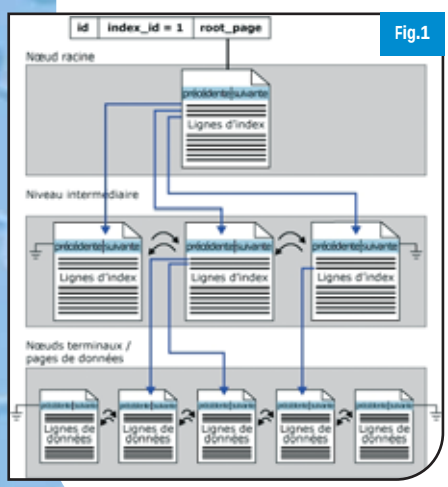


Fig.1

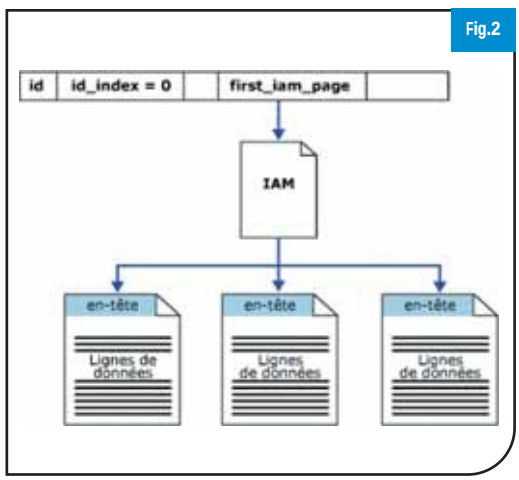


Fig.2

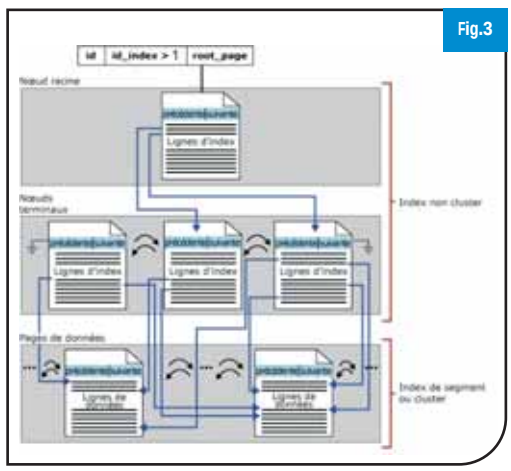


Fig.3

tered Index Scan) pour rechercher les données. [Fig.7]

Un index non cluster sur la colonne PersonType permet de faire baisser le nombre de lectures à 53 et modifie le plan d'exécution : [Fig.8].

On constate l'apparition de l'opérateur Key Lookup (car il existe un index cluster sur la table), exécuté 17 fois (nombre de lignes remontées par l'opérateur Index Seek), qui va permettre de sélectionner les champs FirstName et LastName. [Fig.9]

Bien que performante, cette requête peut être optimisée grâce à un index couvrant.

SQL Server 2005 a introduit les index include, des index couvrants plus légers que dans les versions précédentes de SQL Server. Il ne va stocker les colonnes incluses que dans les pages feuilles de l'index, les niveaux intermédiaires et racines ne comprenant eux que les données de la clé. Le parcours de cet index, léger et couvrant, sera rapide. [Fig.10]

Après seulement 2 lectures disque, la requête est entièrement satisfaite, l'index seul a suffit pour présenter les données à l'utilisateur. [Fig.11 et 12]

L'index couvrant a permis de réduire les IO disques de 3816 à seulement 2. Ce gain de performance important va, dans un environnement multi utilisateur, permettre la diminution du verrouillage sur la table et, de fait, augmenter la concurrence d'accès. De tels index peuvent être encore plus performants en utilisant la notion de filtre apparue avec SQL Server 2008. Les index filtrés contiennent une clause Where permettant de n'indexer qu'une partie des données, en excluant les valeurs NULL par exemple. Ils deviennent encore plus sélectifs et plus performants.

■ Christophe **Laporte**
MCT - MVP SQL Server

Fig.4

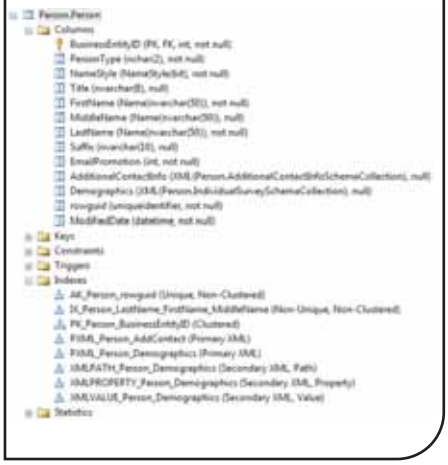


Fig.5

```
SELECT [PersonType], COUNT(*)
FROM [AdventureWorks2008].[Person].[Person]
GROUP BY [PersonType]
```

Results | Messages | Execution plan

	PersonType	(No column name)
1	EM	273
2	IN	18484
3	SP	17
4	SC	753
5	VC	156
6	GC	289

Fig.6

```
SELECT [PersonType]
      ,[FirstName]
      ,[LastName]
FROM [AdventureWorks2008].[Person].[Person]
WHERE [PersonType] = 'SP'
```

Results | Messages

	PersonType	FirstName	LastName
1	SP	Stephen	Jiang
2	SP	Michael	Blythe
3	SP	Linda	Mitchell
4	SP	Jillian	Carson
5	SP	Garrett	Vargas
6	SP	Tsvi	Reiter
7	SP	Pamela	Ansman-Wolfe
8	SP	Shu	Ito
9	SP	José	Saraiva
10	SP	David	Campbell
11	SP	Tete	Mensa-Annan
12	SP	Syed	Abbas
13	SP	Lynn	Tsoflias
14	SP	Amy	Alberts
15	SP	Rachel	Valdes
16	SP	Jae	Pak
17	SP	Ranjit	Varkey Chudukatil

Fig.7

Fig.8

Fig.10

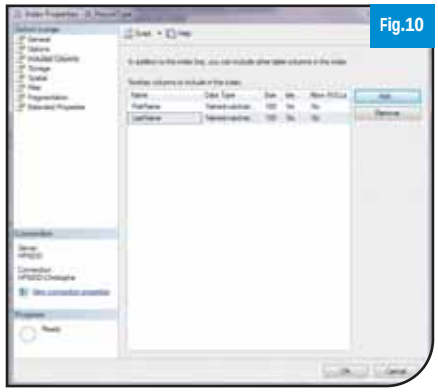


Fig.11

Fig.9

Key Lookup (Clustered)

Uses a supplied clustering key to lookup on a table that has a clustered index.

Physical Operation	Key Lookup
Logical Operation	Key Lookup
Actual Number of Rows	17
Estimated I/O Cost	0,003125
Estimated CPU Cost	0,0001581
Number of Executions	17
Estimated Number of Executions	17
Estimated Operator Cost	0,0525893 (94%)
Estimated Subtree Cost	0,0525893
Estimated Number of Rows	1
Estimated Row Size	113 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	True
Node ID	3

Object
[AdventureWorks2008].[Person].[Person].[PK_Person_BusinessEntityID]
Output List
[AdventureWorks2008].[Person].[Person].FirstName; [AdventureWorks2008].[Person].[Person].LastName
Seek Predicates
Seek Keys[1]: Prefix: [AdventureWorks2008].[Person].[Person].BusinessEntityID = Scalar Operator ([AdventureWorks2008].[Person].[Person].[BusinessEntityID])

Index Seek (NonClustered)

Scan a particular range of rows from a nonclustered index.

Physical Operation	Index Seek
Logical Operation	Index Seek
Actual Number of Rows	17
Estimated I/O Cost	0,003125
Estimated CPU Cost	0,0001757
Estimated Number of Executions	1
Number of Executions	1
Estimated Operator Cost	0,0033007 (6%)
Estimated Subtree Cost	0,0033007
Estimated Number of Rows	17
Estimated Row Size	15 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	True
Node ID	1

Object
[AdventureWorks2008].[Person].[Person].[IX_PersonType]
Output List
[AdventureWorks2008].[Person].[Person].BusinessEntityID; [AdventureWorks2008].[Person].[Person].PersonType
Seek Predicates
Seek Keys[1]: Prefix: [AdventureWorks2008].[Person].[Person].PersonType = Scalar Operator(N'SP')

Fig.12

Index Seek (NonClustered)

Scan a particular range of rows from a nonclustered index.

Physical Operation	Index Seek
Logical Operation	Index Seek
Actual Number of Rows	17
Estimated I/O Cost	0,003125
Estimated CPU Cost	0,0001757
Estimated Number of Executions	1
Number of Executions	1
Estimated Operator Cost	0,0033007 (100%)
Estimated Subtree Cost	0,0033007
Estimated Number of Rows	17
Estimated Row Size	117 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	True
Node ID	0

Object
[AdventureWorks2008].[Person].[Person].[IX_PersonType]
Output List
[AdventureWorks2008].[Person].[Person].PersonType; [AdventureWorks2008].[Person].[Person].FirstName; [AdventureWorks2008].[Person].[Person].LastName
Seek Predicates
Seek Keys[1]: Prefix: [AdventureWorks2008].[Person].[Person].PersonType = Scalar Operator (CONVERT_IMPLICIT(nvarchar(4000),[@1],0))

Avis d'expert

L'opposition relationnel – objet est-elle toujours d'actualité ?

Les bases de données ont à relever deux défis majeurs qui touchent à leurs limites: la complexité (des données et des besoins d'exploitation) et les volumes: par exemple, le stockage de données d'imagerie, notamment l'imagerie médicale DICOM qui représente un défi très pointu – et très actuel.

Les technologies de bases de données « traditionnelles », relationnelles et orientées objet, peinent à répondre à la fois à ces deux défis.

Les SGBD relationnels répondent mieux au défi des volumes – même si les besoins actuels commencent à toucher aux limites même des SGBDR – mais continuent de se montrer inadaptés au traitement de données complexes (voir la floraison de technologies de mapping objet-relationnel comme Hibernate, TopLink, etc.), pour tenter de gérer efficacement des graphes d'objets complexes, et les approches spécifiques requises par l'« intelligence métier ».

Dans les deux cas, on essaie de rapprocher au mieux un besoin métier avec une approche « stockage » en réalité inadaptée à ce besoin. Ces solutions – pour la plupart de grande qualité – ont leurs limites et ajoutent une couche de complexité qui n'a pas toujours un effet positif sur la mise en place des besoins. Notamment, les outils de mapping objet-relationnel sont tous relativement limités quant au niveau de complexité et de volume de données (en stockage et en requête) qu'ils sont capables de gérer efficacement.

De leur côté, les SGBD purement objet répondent de manière adéquate au défi de la complexité mais n'ont jamais réussi à convaincre sur leur gestion des volumes et leur efficacité sur des requêtes simples. De plus, malgré diverses tentatives de standardisation, ils nécessitent souvent l'utilisation d'un langage de programmation afin d'accéder aux données, et sont d'ailleurs souvent fortement liés à un langage de programmation spécifique au détriment des autres.

Rapprocher les deux mondes n'est pas un défi facile à relever. Diverses grandes sociétés informatiques s'y sont frottées en incluant dans leurs SGBD relationnels des capacités orientées objet (ou XML, ce qui revient en pratique au même) plus ou moins « natives » (Sybase, Oracle, Microsoft…) A ce jour, ces tentatives restent relativement anecdotiques et certainement peu convaincantes, l'intégration au SGBDR étant généralement assez pauvre et la performance très insuffisante.

On rêve donc d'une solution mêlant efficacement les modèles relationnel et orienté objet. Mais cette solution devra très certainement remettre en cause les fondements même des modèles de stockage des SGBD actuels, qu'ils soient relationnels ou orientés objet. C'est là sans doute une difficulté majeure pour les grands acteurs du marché: remettre en question la base même de leur offre commerciale.

Peut-être la réponse viendra-t-elle d'un acteur indépendant : InterSystems, par exemple, propose depuis des années un SGBD, « Caché », capable de stocker de grands volumes de données complexes dans une structure efficace pour une exploitation dans chacun des deux modes, relationnel et orienté objet, et accessibles par de nombreux langages orientés objets aussi bien que par du SQL standard (via ODBC ou JDBC).

Par ailleurs, on se prend à repenser à certains modèles passés de mode pour diverses raisons mais qui peuvent peut-être répondre efficacement à ces défis. Modèles (nativement) hiérarchiques ou multidimensionnels (voir IMS DB), en réseau (voir IDMS), etc. Certainement, le système « BigTable » de Google, utilisé notamment pour l'indexation du contenu web et divers produits de la société comme Google Earth et Google Finance, en est une parfaite illustration. Afin de répondre à des besoins extrêmes en termes de volumes (de l'ordre du pétaoctet!) et de transactions, Google a développé un système basé sur des « vecteurs creux multidimensionnels persistants, classés et distribués » somme toute aux antipodes des systèmes relationnels, et très différents des systèmes de stockage orientés objet – en fait, une approche qui rappelle fortement dans les principes directeurs le système de stockage MUMPS, qui existe depuis plus de 40 ans!

Doit-on pour autant abandonner le bateau du relationnel ? Le stockage orienté objet est-il pour autant une impasse ? Il ne faudrait pas dramatiser. Ces systèmes répondent toujours parfaitement à l'immense majorité des besoins en stockage et requêtes. Les modèles alternatifs répondent à des besoins souvent extrêmes où très spécifiques. Il est indéniable cependant qu'à terme, ces approches devront être consolidées en un nouveau modèle répondant mieux à l'ensemble des besoins du domaine.

■ Olivier **Caudron**
Senior Sales Engineer InterSystems

Inside Visual Studio 2010

Dans quelques mois, le développeur Windows / .NET va pouvoir utiliser une des versions de son environnement de développement préféré les plus attendues depuis 10 ans : Visual Studio 2010. Autant les versions 2005 et 2008 étaient dans la continuité de l'édition 2003, autant le millésime 2010 redéfinit l'IDE tel qu'on le connaît depuis une décennie.

PRÊT POUR OFFICE 2010, WINDOWS 7

Avec Visual Studio 2010 (VS 2010), le développeur Windows et / ou .Net va s'ouvrir de nouveaux horizons, de nouveaux développements. Tout d'abord, c'est l'IDE idéal pour créer les meilleures applications pour Windows 7 !

VS 2010 permet de développer des applications natives (C++) Windows 7, ou en code managé (.Net) tout en tirant parti de toutes les nouveautés du système : multitouch, Direct Access, le nouveau SDK Windows, le ruban, etc. C'est aussi l'occasion de passer des applications WinForms aux applications WPF.

L'autre nouveauté est une intégration toujours plus fine entre Visual Studio et l'ensemble des logiciels Office 2010. Microsoft fournit un effort considérable pour simplifier le développement Sharepoint dans VS 2010. Ainsi, concevoir un projet Sharepoint n'est plus une corvée. Visual Studio dispose désormais des outils spécifiques Sharepoint, les templates projets, un explorateur serveur pour les sites et contenus Sharepoint, etc. Le développeur peut ainsi créer rapidement des applications Sharepoint, agréger des données métiers, intégrer des données de différentes sources (SQL Server, SAP, Siebel...). On peut aussi packager et déployer un projet Sharepoint 2010 depuis VS 2010.

Au-delà de Sharepoint 2010, VS 2010 continue à améliorer le développement des applications Office. Avec la prochaine sortie d'Office 2010, VS 2010 dispose d'une toute nouvelle version des outils Visual Studio Tools for Office. Il devient possible de développer en 32 et 64-bit, créer des packages de déploiement, bénéficier des nouvelles interfaces d'Office (ex. : Ruban, Linq).

BOOSTEZ VOS PROJETS AVEC LA PROGRAMMATION PARALLÈLE

Depuis plusieurs années, Microsoft, et l'ensemble de l'industrie informatique, sensibilise les développeurs aux développements multicore, à la programmation parallèle, permettant d'exploiter au mieux les processeurs à cœur multiples. Malheureusement, à peine 3 % du code produit est réellement parallélisé (chiffres Microsoft, 2008). Le défi est colossal. Car un logiciel non parallélisé ne tirera aucun avantage d'un processeur à 4 cœurs et l'utilisateur sera frustré de ne pas exploiter les performances de son ordinateur. Avec des logiciels toujours plus gros, puissants, la programmation parallèle devient un leitmotiv.

C'est pour cette raison que VS 2010 met en exergue le parallélisme aussi bien dans l'environnement que dans les librairies de développement. Tout est fait pour aider le développeur en limitant les modifications de code. Côté librairies, on dispose de PLinq, de Parallel Fx ou encore du Task Parallel Library. Mais pour exploiter au mieux la parallélisation, Microsoft a incorporé un tout nouveau runtime interne : le Concurrency Runtime qui doit gérer les tâches et les ressources. Au-delà de ces librairies et API, pour C++ et .Net, le développeur dispose aussi dans VS 2010 d'outils de débogage parallèle et d'analyse.
Avec VS 2010, c'est le moment ou jamais de sérieusement se mettre à la programmation parallèle !

DEVENEZ UN PRO DU CLOUD COMPUTING !

Cela fait presque 2 ans que l'on entend parler de services en ligne (Saas) ou encore de cloud computing. Microsoft propose sa propre vision du cloud avec Windows Azure. Et durant la conférence développeur (PDC) de novembre dernier, Microsoft a dévoilé la version finale de Windows Azure, des librairies, des outils. VS 2010 devient ainsi le centre de développement par excellence d'une application Azure.

On dispose de template Azure et des Windows Azure Tools for Visual Studio. Ils permettent de créer, déboguer et de déployer des services et applications cloud dans Azure. Là encore, tout est fait pour intégrer le cloud computing dans ses développements tout en facilitant la construction et le déploiement. Le développeur a accès à l'ensemble des services Azure : projet Dallas pour l'agrégation des données, SQL Azure (base de données dans le cloud), Azure Storage, .Net Services.

Et surtout, le développeur bénéficiera avec l'achat d'un VS 2010 de nombreuses heures d'utilisation Windows Azure. Idéal pour tester et comprendre le cloud computing !

Aujourd'hui, le développement ne se limite pas à l'intégration de nouvelles technologies. Il est important pour chaque développeur (et en particulier ceux qui sont en relation directe avec le client) d'avoir une vision de haut niveau de son projet, des relations avec ses clients, de l'avancement, de la qualité. C'est ce qu'on est en droit d'attendre d'un outil de développement moderne. Avec Visual Studio 2010, le référentiel commun à la gestion des bugs, des tests, des exigences et autres éléments de travail donne une transparence nouvelle aux projets de développement. Partager ce référentiel entre développeurs et testeurs garantit une meilleure collaboration entre les deux métiers. Cette collaboration est renforcée par des fiches de bugs, riches de contexte, permettant d'éviter de nombreux aller-retour entre la découverte d'un bug par le testeur et sa reproduction sur le poste du développeur.

Une gamme 2010 plus simple, plus complète !

Avec VS 2010, Microsoft a entièrement revu sa gamme. C'en est fini de Team System et des nombreuses éditions. L'éditeur fusionne l'ensemble de ses outils de développement et d'ALM en une seule offre : « Visual Studio 2010 ». On disposera désormais de 4 éditions, au lieu de 7 :

- Visual Studio 2010 Professionnel
- Visual Studio 2010 Professionnel avec MSDN
- Visual Studio 2010 Premium avec MSDN
- Visual Studio 2010 Ultimate avec MSDN

Chacune d'elles possède ses spécificités fonctionnelles. La version « intégrale » possède l'ensemble des fonctionnalités, la modélisation et toutes les fonctions de tests, d'architecture, etc. **(Fig. A)**

La gamme Visual Studio 2010 contient également trois nouveaux produits satellites. Le premier est un outil dédié aux testeurs fonctionnels, Visual Studio Test Elements 2010. Ce dernier propose une interface épurée par rapport à l'IDE de Visual Studio et permet, de gérer les campagnes de tests manuels, de les exécuter et même de les automatiser. **(Fig. B)**

Le deuxième produit est Visual Studio Team Lab Management 2010 qui permet de gérer des environnements virtuels via System Center Virtual Machine Manager pour les campagnes de tests. Enfin on retrouve TFS qui a été renommé pour l'occasion Visual Studio Team Foundation Server 2010. Visual Studio Team Foundation Server 2010 propose de nouveaux modes d'installation. Il n'y a plus d'édition Workgroup ni Active Directory, Microsoft propose désormais deux modes d'installation :

- Le mode Basic, destiné à remplacer Visual Source Safe, et qui peut être installé soit sur un environnement serveur, soit sur un environnement client (Vista et Windows 7). La couche de données pourra utiliser une version Express de SQL Server et le nombre d'utilisateurs ne sera pas limité. Les restrictions de cette version sont sur les composants installés, seul le gestionnaire de code source, la gestion des WorkItems et le serveur de build (qui, lui, pourra être installé sous XP) seront disponibles.
- Le mode Standard, lui, contient l'ensemble des composants.

Il est bien entendu possible de migrer d'une version basique vers une version Standard si le besoin des composants manquants se faisait sentir (Sharepoint et Reporting). Concernant les modes de licences, Microsoft a indiqué qu'ils allaient être entièrement revus pour la gamme Visual Studio 2010.

■ Loïc Baumann - *Architecte Logiciel et expert en ALM chez Winwise ainsi que MVP Team System.*

■ Guillaume Rouchon - *Consultant / Formateur .NET et expert ALM chez Winwise.*

■ Vincent Labatut - *Consultant / Formateur .NET et expert ALM chez Winwise.*

Fig.A

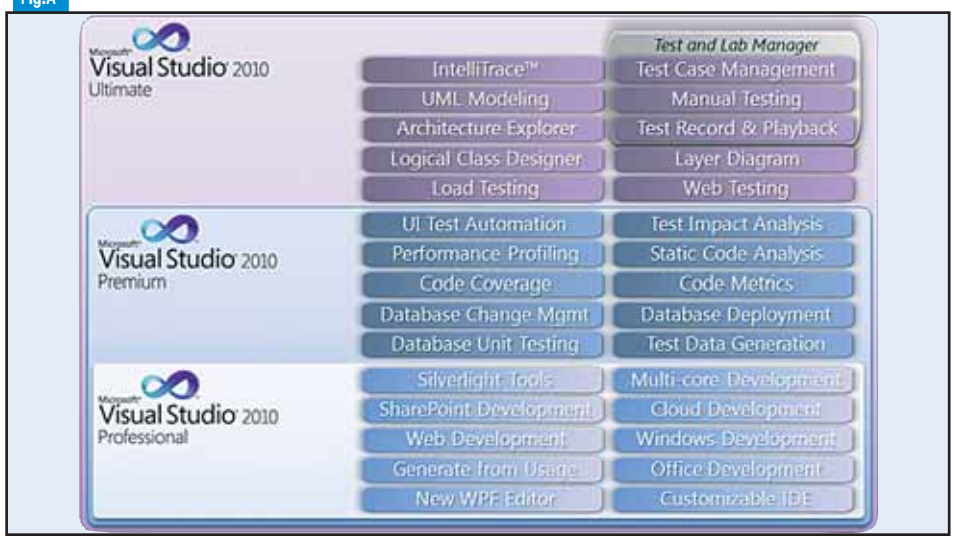


Fig.B

Découverte de Visual Studio 2010

1re partie

Comme vous le savez tous, Microsoft s'apprête à sortir une nouvelle version de son environnement de développement phare : Visual Studio. Autant le dire tout de suite, cette version 2010 est majeure : tant au niveau de l'évolution du contenu qu'au niveau des changements qui ont été apportés sur l'ensemble de la gamme.

Deux articles ne seront pas de trop pour vous présenter la gamme cliente puis la gamme serveur. Nous verrons au passage que Visual Studio est plus que jamais la marque de Microsoft pour les développeurs suite à la disparition du nom Team System. C'est parti pour ce tour d'horizon, attachez vos ceintures !

DU NOUVEAU POUR LES DÉVELOPPEURS

Intellitrace

S'il ne fallait retenir qu'une nouveauté dans Visual Studio 2010 pour les développeurs, il s'agirait de l'IntelliTrace, précédemment connue sous le nom d'Historical Debugger. Cette fonctionnalité révolutionne la manière dont un développeur va pouvoir déboguer une application via :

- L'enregistrement des évènements d'une application.
- La journalisation automatique des appels de méthode (trace) et la navigation à postériori dans ces traces.

Dans les deux cas le mécanisme utilisé est l'injection de code IL à l'exécution. Ce procédé d'instrumentation peut dégrader les performances de l'application. Cependant, dans le cas où l'on cherche à analyser une application en profondeur, on ne sera pas vraiment regardant là-dessus à la vue de l'énorme gain de temps qu'il procure.

Traçage des évènements

L'enregistrement des évènements est l'option la moins intrusive et permet de suivre l'ensemble des évènements émis depuis le lancement d'une application jusqu'à ce qu'elle se termine (Fig. 1).

Voici quelques exemples des évènements disponibles :

- Action sur un contrôle graphique.
- Accès aux fichiers.
- Accès à la base de registre.
- Exceptions.
- ADO.Net

Si vous ne trouvez pas votre bonheur dans les évènements fournis par défaut ou si vous souhaitez enregistrer des appels à vos propres librairies, il est tout à fait possible d'en ajouter de nouveaux via l'édition du fichier « CollectionPlan.xml ». Comme on peut le voir sur la figure 1, Visual Studio affiche dans la trace :

- La catégorie (par exemple « File »).
- La description courte de l'évènement (« Access C:\test.txt »).
- Une description longue (« Initialized a FileStream to the path C:\test.txt »).
- Le thread courant (« Thead: Main Thread [2116] »).
- Des liens vers des différentes vues (“Locals, Call Stack »).

La trace peut également contenir la valeur de certaines variables (dans notre exemple le chemin du fichier). Il est en effet possible lors, de la déclaration d'un évènement, de spécifier un ensemble de variables à tracer lorsque cet évènement survient. La valeur de ces variables est alors disponible dans la vue « Locals ». En plus de ces informations, Visual Studio est capable de faire le lien entre un évènement et la ligne de code l'ayant généré : un double-clic sur l'entrée vous transporte à la ligne correspondante dans le code !

Toutes ces informations sont accessibles aussi bien pendant une session de débogage qu'a postériori. En effet, toutes les données sont stockées dans un fichier « .tdlog » dont l'emplacement est paramétrable dans les options de Visual Studio. Une simple ouverture de ce fichier avec Visual Studio permet de naviguer dans la trace. Plus besoin d'exécuter des dizaines de fois l'application en espérant retomber sur le bug, ayant mis des points d'arrêt au préalable pour connaître l'état de l'application avant le problème : l'application plante, on ouvre la trace et l'on voit l'ensemble des évènements qui ont eu lieu !

Traçage de tous les appels de méthode

Si cette liste d'évènements n'est pas suffisante, il est possible d'activer en plus la trace de l'ensemble des appels des méthodes avec la valeur de leurs paramètres. Cette trace est plus lourde mais va permettre de stocker un arbre exhaustif des appels comme on peut le voir sur la figure 2.

On retrouve donc tous les appels de méthode du projet ainsi que les appels au Framework .Net (en gris). Pour chaque entrée et sortie de méthode, on aura aussi accès aux valeurs des paramètres ainsi

Fig.1

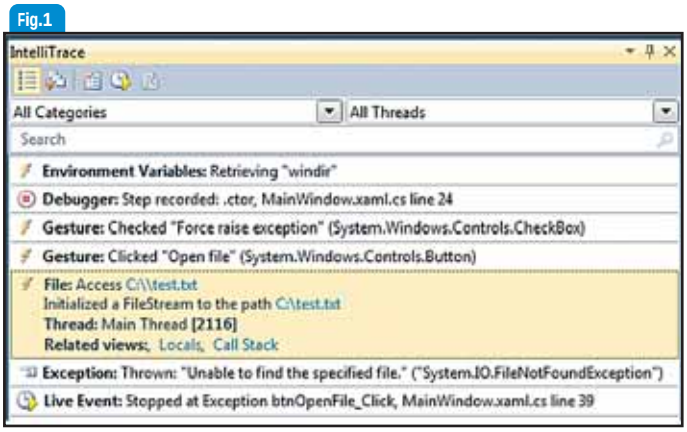
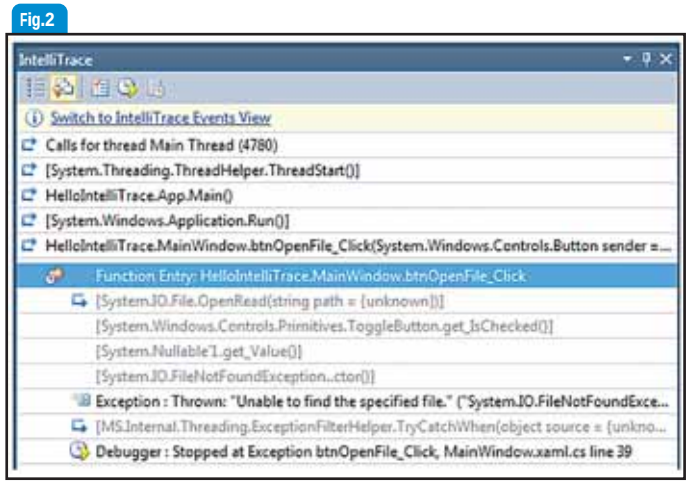


Fig.2



qu'à la valeur de retour de la fonction ! Pour parfaire le tout, il est possible de se déplacer librement dans l'arbre des appels et de remonter dans l'historique afin de suivre exactement le parcours qu'a effectué l'application. Ce voyage dans le temps se reflète dans la trace des évènements qui se restreint dynamiquement aux événements effectivement réalisés. Cette navigation peut se faire soit via les mêmes raccourcis clavier que ceux utilisés pendant le débogage, soit par un jeu de boutons contextuels affichés dans la marge du code **(Fig. 3)**.

Fig.3

Une autre option très utile est la possibilité de se déplacer entre les différents appels d'une même méthode. En sélectionnant « Search For This Method In IntelliTrace » dans le menu contextuel, on peut alors visualiser les différents appels puis reprendre notre investigation en naviguant dans l'arbre des appels.

Exemple d'utilisation de l'IntelliTrace

Afin de montrer l'avantage de l'IntelliTrace, voici un petit exemple. Nous avons une classe Calc exposant quatre méthodes permettant d'ajouter, de supprimer, de diviser et de multiplier un opérande fourni en paramètres à la valeur stockée dans la propriété Resultat. La propriété Resultat est mise à jour avec le résultat de l'opération :

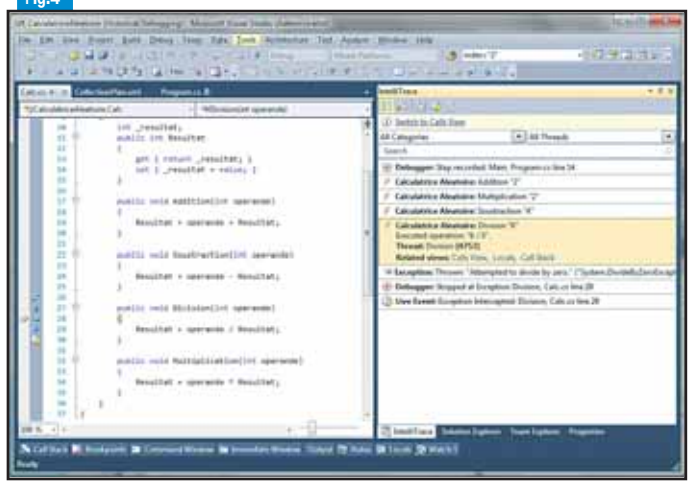
```
public class Calc
{
    int _resultat;
    public int Resultat
    {
        get { return _resultat; }
        set { _resultat = value; }
    }

    public void Addition(int operande)
    {
        Resultat = operande + Resultat;
    }

    public void Soustraction(int operande)
    {
        Resultat = operande - Resultat;
    }

    public void Division(int operande)
    {
        Resultat = operande / Resultat;
    }

    public void Multiplication(int operande)
    {
        Resultat = operande * Resultat;
    }
}
```

Nous allons maintenant appeler ces quatre opérations en parallèle afin de simuler une application multi-thread :

```
class Program
{
    static object SyncLock = new object();

    static void Main(string[] args)
    {
        var calc = new Calc();
        var tasks = new Thread[] {
            new Thread(o => { lock (SyncLock) { calc.
Addition((int)o); }}) {
                Name = "Addition"
                },
            new Thread(o => { lock (SyncLock) { calc.
Soustraction((int)o); }}) {
                Name = "Soustraction"
                },
            new Thread(o => { lock (SyncLock) { calc.
Division((int)o); }}) {
                Name = "Division"
                },
            new Thread(o => { lock (SyncLock) { calc.
Multiplication((int)o); }}) {
                Name = "Multiplication"
                },
            };

        tasks[0].Start(2);
        tasks[1].Start(4);
        tasks[2].Start(6);
        tasks[3].Start(2);

        Array.ForEach(tasks, t => t.Join());

        Console.WriteLine("Resultat: " + calc.Resultat);
        Console.ReadKey();
    }
}
```

Comme on peut s'en douter, le résultat est aléatoire et va dépendre de l'ordre d'exécution des threads. Si vous exécutez ce code plusieurs fois, il peut arriver que vous ayez une exception de type *Divi-*

Fig.4



deByZeroException. Sans IntelliTrace, il faudrait ajouter un mécanisme de journalisation afin de savoir l'ordre d'exécution des appels qui a généré cette erreur et il faudrait relancer l'application en espérant retomber sur le problème. Avec IntelliTrace, plus besoin de tout cela ! Afin de ne pas ajouter de code, nous avons modifié le fichier CollectionPlan.xml afin de définir des évènements sur les méthodes de la classe Calc. Lorsque l'erreur est arrivée, il a suffi de regarder la vue des évènements pour voir ce qui s'est passé depuis le lancement de l'application **(Fig. 4)** :

On s'aperçoit ainsi que si les opérations sont exécutées dans l'ordre suivant :

- Ajout : 2 + 0 = 2
- Multiplication : 2 * 2 = 4
- Soustraction : 4 – 4 = 0
- Division : 6 / 0 => erreur

Nous obtenons alors notre exception. Cet exemple est assez simple et certains d'entre vous auront vu les séquences qui ne fonctionnent pas juste en regardant le code mais dans un vrai programme multi-thread, cela peut faire gagner énormément de temps !

De plus, comme vous pouvez le voir sur l'image, lors de la déclaration des évènements personnalisés nous avons capturé la valeur du paramètre de la fonction et la valeur de la propriété Resultat afin d'enrichir la trace. Il est à noter que la capture de valeur ne peut se faire que sur des variables, d'où le fait que la classe Calc n'utilise pas de propriété automatique.

LES BRANCHES

Penchons-nous maintenant sur une des fonctionnalités de base de la gestion de configuration : les branches. Celles de TFS et VS 2010 ont gagné en importance : elles sont devenues des composantes de premier ordre, nous allons donc passer en revue les dernières améliorations autour des branches et des ChangeSets.

Premier point, les branches sont maintenant directement identifiables au niveau du contrôleur de source **(Fig.5)**.

Attention, si vous migrez depuis TFS 2008 il faudra indiquer où sont les branches à l'aide de l'action « Convert to Branch » du menu contextuel.

Question ergonomie, certaines boîtes de dialogues sont devenues des fenêtres documents à part entière comme l'historique des ChangeSets. Ce dernier permet également d'afficher les labels (enfin !) **(Fig. 6)**.

Lors de développements complexes, on a besoin d'avoir du recul vis-à-vis de la structure des branches en place, on veut pouvoir visualiser l'ensemble ou partie des branches de notre projet. C'est ce que propose la vue hiérarchique des branches, accessible depuis n'importe quelle branche dans son menu contextuel **(Fig.7)**.

Depuis ce document, il est alors très facile de fusionner deux branches. En effet, un simple glisser-déposer d'une branche vers une autre suffit pour effectuer l'opération ! La résolution des conflits a également été sensiblement améliorée. Au lieu de devoir régler tous les conflits dans une fenêtre modale, les conflits sont affichés dans la fenêtre des changements en attente (Pending Changes), ce qui est finalement beaucoup plus naturel. On dispose de plus d'informations, comme les utilisateurs impliqués, ainsi que les liens vers les ChangeSets à l'origine du conflit **(Fig.8)**.

Enfin, il est également possible de suivre l'évolution d'un ChangeSet au fil des branches. En effet, en tant que développeur, comment savoir si nos modifications on bien été répercutées dans la structure de branches existante et surtout de quelle manière ? La réponse est dans la fonction « Track Changeset » accessible directement depuis chaque ChangeSet. Cette dernière nous propose un affichage de type hiérarchique ou chronologique.

Vous pouvez voir comment le correctif a été rétro-intégré dans la branche Main puis partiellement porté dans la branche WPF_Poc (les fusions partielles apparaissent en jaune) **(Fig. 9)**.

Là encore, un simple glisser-déposer permettra d'opérer une fusion entre deux branches, cette fois-ci, seul le ChangeSet sera répercuté.

Fig.5

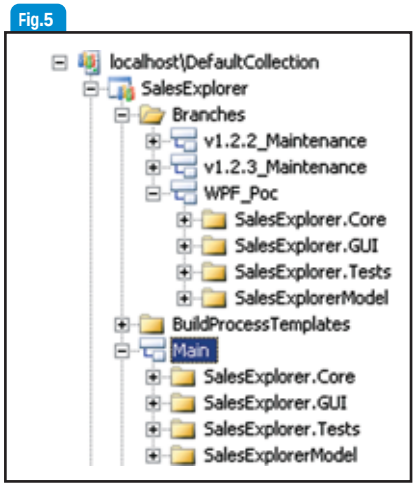


Fig.6

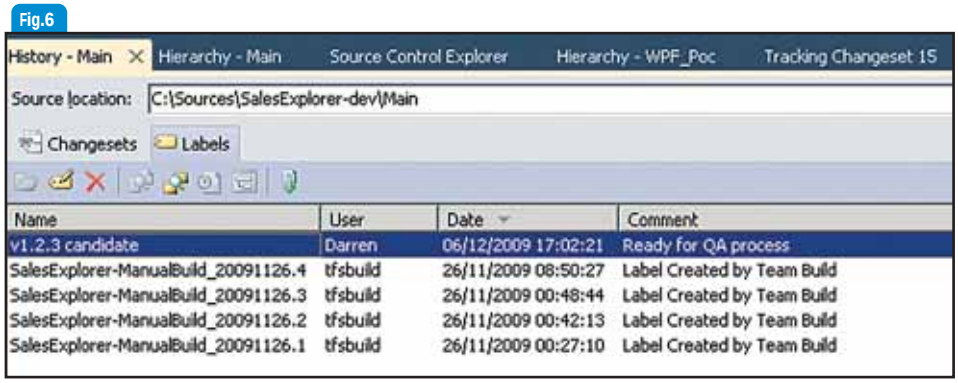


Fig.7

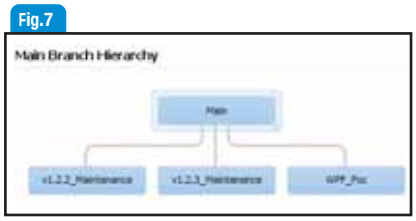
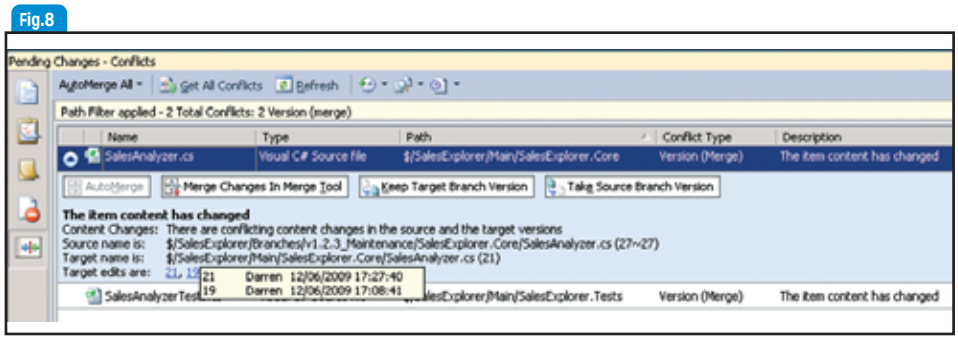


Fig.8



LES NOUVEAUTÉS POUR LES DBA

A partir de la version Premium, Visual Studio 2010 contient des fonctionnalités spécifiques pour le développement de bases de données dont on avait déjà eu un avant-goût avec la release out-of-band de GDR2 pour Visual Studio 2008. Ces outils sont malheureusement encore peu connus du fait de leurs sorties tardives, ce qui va sans doute changer avec la version 2010 qui les inclut nativement. On distingue les outils de comparaison et ceux dédiés au développement.

Les outils de comparaison

Les outils de comparaison marchent aussi bien sur les schémas que sur les données. La comparaison de schémas va, après avoir sélectionné les schémas source et cible ainsi que les objets à comparer, afficher leurs différences sous forme de liste **(Fig. 10)**.

Il est facile de faire ressortir les similitudes ou de se concentrer uniquement sur les différences. Visual Studio permet alors de générer un script TSQL de mise à jour du schéma cible ou de le mettre directement à jour. Il est bien entendu possible de comparer un schéma existant à celui d'un projet de type base de données, les fonctionnalités proposées seront les mêmes. L'outil est particulièrement pratique pour repérer les différences entre les bases de différents environnements (développement et pré-production par exemple) et pour générer les scripts de mise à jour le cas échéant.

Avec l'outil de comparaison de données, on va pouvoir sélectionner les tables dont on veut comparer les données depuis une base source et une base cible. La liste de résultats contiendra **(Fig.11)** :

- Les lignes différentes (même clé primaire mais données différentes).
- Les lignes uniquement présentes dans la table source.
- Les lignes uniquement présentes dans la table cible.
- Les lignes identiques.

Comme pour la comparaison de schémas, Visual Studio 2010 proposera de mettre à jour les tables cibles directement ou de générer un script de mise à jour contenant l'ensemble des requêtes SQL. Il est bien entendu possible de sélectionner les mises à jour que l'on veut effectuer.

QU'EN EST-IL DES TESTEURS ?

Avec la vague 2010, une nouvelle SKU vient compléter Visual Studio, il s'agit de Microsoft Test & Lab Manager ou MTLM (pour éviter un claquage de la mâchoire). Le constat de Microsoft fut simple : le développement d'applications ne demandait principalement l'intervention, il y a vingt ou dix ans, que de développeurs. Aujourd'hui d'autres rôles ont fait surface et dans un souci de productivité, il est important d'offrir un outil qui est adressé aux testeurs fonctionnels. En effet, l'intégration entre les deux mondes est essentielle si l'on veut pouvoir développer efficacement et sereinement son application. MTLM était donc destiné initialement aux testeurs mais il suscite néanmoins de l'intérêt chez les développeurs, d'où sa présence dans la version Ultimate de Visual Studio.

Nous ne pouvons malheureusement pas nous attarder sur l'ensemble des fonctionnalités que nous offre ce nouveau logiciel mais il est intéressant de parler de certains concepts et certaines fonctionnalités qui vont changer la vie d'une équipe de développement.

Test Suite, Test Case et Requirement

Vous avez la possibilité de déclarer des campagnes de test (Test Suite) qui vont enfin formaliser cette tâche dans le développement de votre logiciel. Ces campagnes sont composées d'une série de Test Case ou de Requirement à tester **(Fig. 12)**.

Un Test Case est tout simplement un test manuel (idéal pour tester les IHM) tandis qu'un Requirement est la formalisation d'une exigence à remplir. L'intérêt du Requirement est qu'il peut se lier à n'importe quel type de Work Item qui va détailler son implémentation (typiquement les types Task ou Test Case). La gestion des Requirements et des campagnes de test permettent donc d'assurer la traçabilité complète entre une exigence et la validité de son implémentation !

Nous ne nous étendrons pas plus sur le sujet mais sachez que Microsoft a bien fait les choses, MTLM gère le déroulement d'une campagne, l'analyse des résultats d'une campagne (avec les statistiques de régression par rapport au déroulement précédent, etc.**(Fig.13)**).

Fig.9

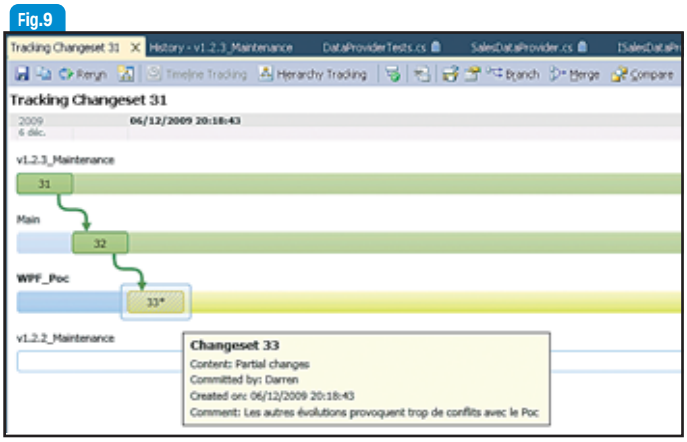


Fig.10

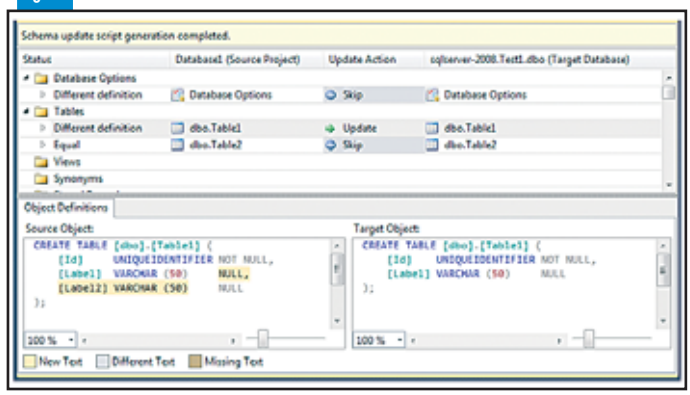
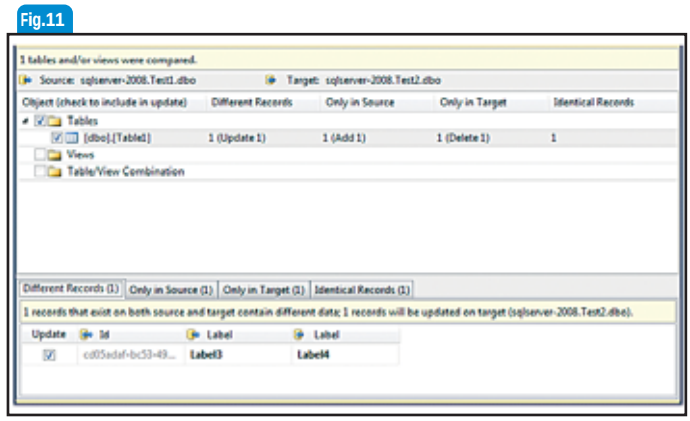


Fig.11



Maîtriser son environnement de test

Une campagne de tests va toujours s'inscrire dans un contexte d'exécution ou Test Plan. Celui-ci permet de définir sur quelles machines (Test Environment) la campagne va être exécutée pour les tests manuels et automatiques et en fonction de quels paramétrages spécifiques (Test Settings). La notion de Test Environment est très importante, nous aurons l'occasion de revenir dessus dans la prochaine partie avec Team Lab. Le Test Plan permettra en outre de maîtriser la campagne dans son exécution (date de début, date de fin, responsable, etc. (Fig. 14)).

A LA RESCOUSSE DES ARCHITECTES

Le rôle d'Architecte d'Entreprise a été traité avec peu de réussite dans les éditions précédentes de Team System. Ayant conscience de son échec, Microsoft a décidé de faire table rase et de repartir sur une nouvelle brique qui serait orientée cette fois-ci vers l'aspect Architecte Logiciel.

Virage à 180° particulièrement réussi! Si l'édition précédente n'avait quasiment aucun intérêt, Microsoft a su retenir toutes les critiques et s'orienter vers une nouvelle direction qui remplit une bonne partie du contrat : venir en aide, assister et fournir les briques de base pour les Architectes Logiciels.

Les différentes fonctionnalités sont réparties dans les éditions Premium et Ultimate de Visual Studio 2010.

Pour l'édition Premium, nous avons l'accès en lecture-seule des différents diagrammes (UML, Couche Logique, et DGML). L'édition Ultimate, quant à elle, offre la totalité des fonctionnalités à travers : la modélisation UML, les diagrammes de couche logique (avec validation de l'architecture applicative) et l'Architecture Explorer.

Fig.12

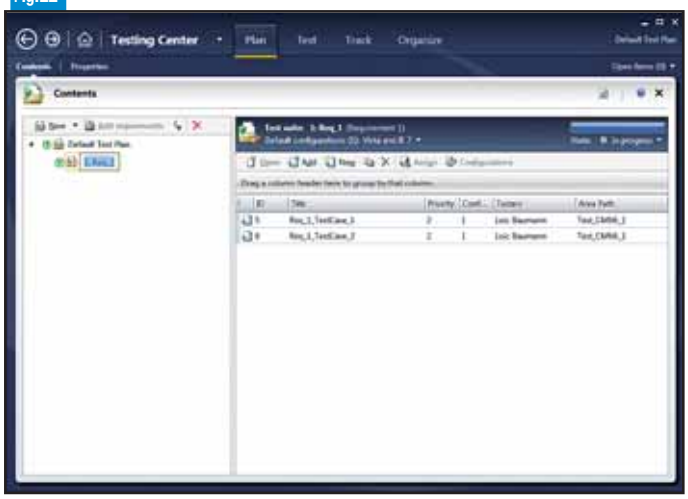
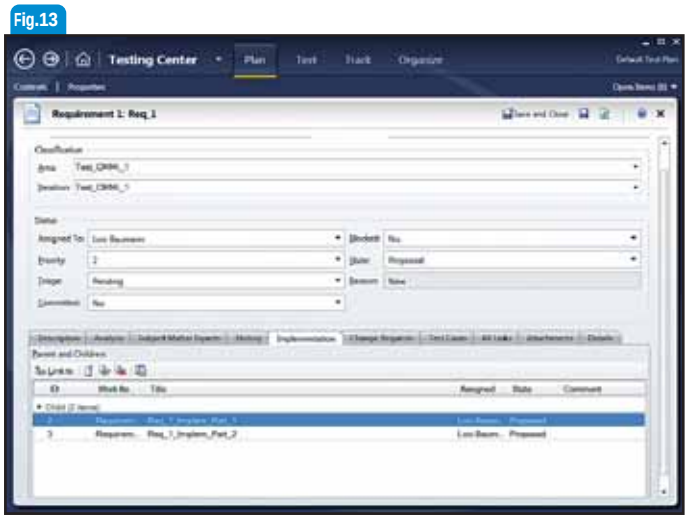


Fig.13



De l'UML dans Visual Studio

Une bonne surprise est le revirement de Microsoft concernant l'utilisation de l'UML. Dans le passé, l'éditeur de Redmond était parti dans la direction des DSL, prétextant une ouverture plus large par opposition à un standard clairement défini et délimité comme l'UML. Cependant, l'un n'empêche pas l'autre et Visual Studio 2010 le prouve par le biais des briques architectes et en délivrant l'excellent DSL Toolkit. L'UML est un standard adopté par beaucoup d'entreprises, l'avoir au sein de Visual Studio renforce la position de l'éditeur et conforte son ambition d'avoir un environnement avec le maximum de fonctionnalités clés intégrées.

Les fonctionnalités Architecte

Les fonctionnalités architecte s'articulent autour de deux concepts principaux : les diagrammes et l'Architecture Explorer.

Fig.14

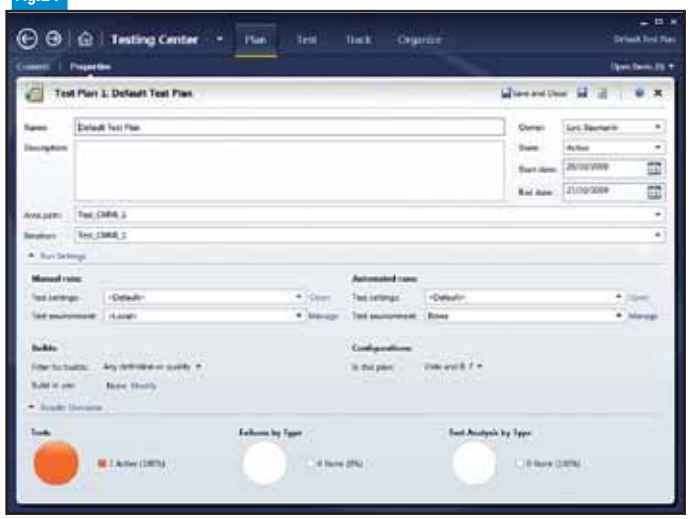


Fig.15

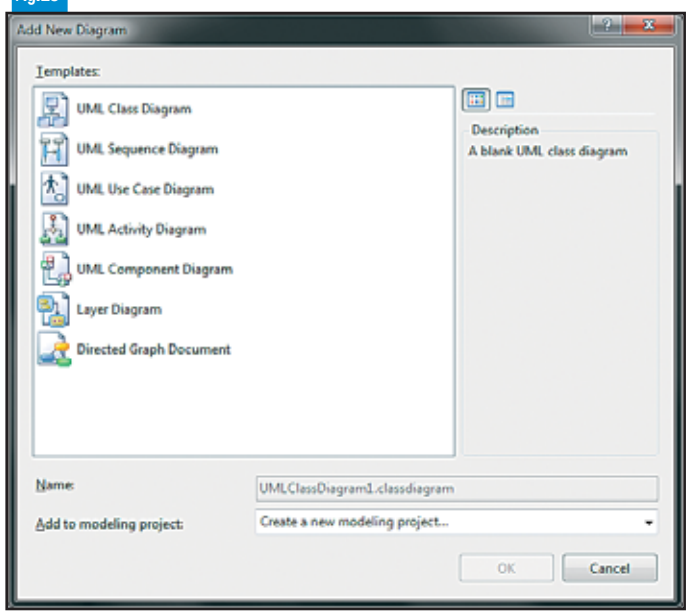


Fig.16

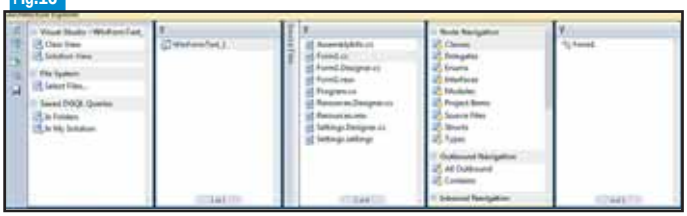


Fig.17

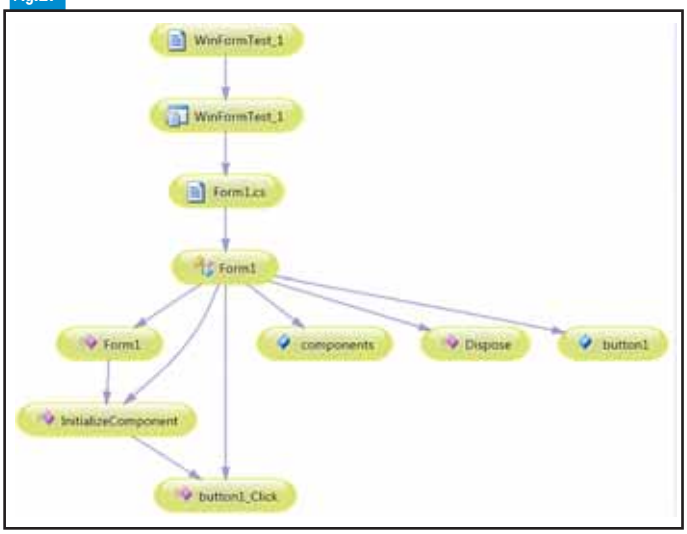


Fig.18

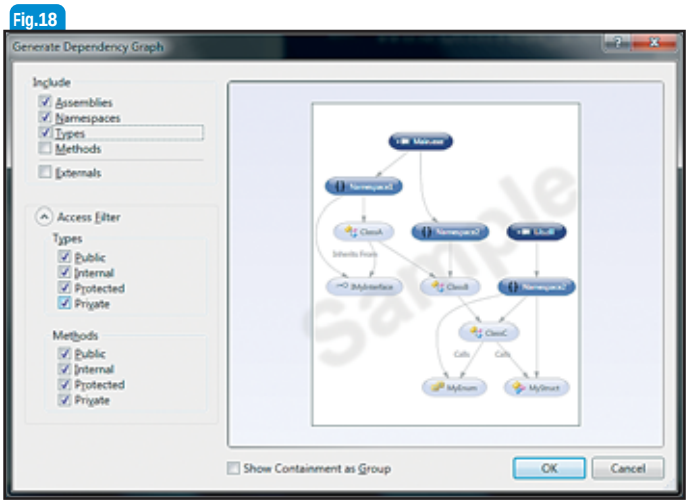
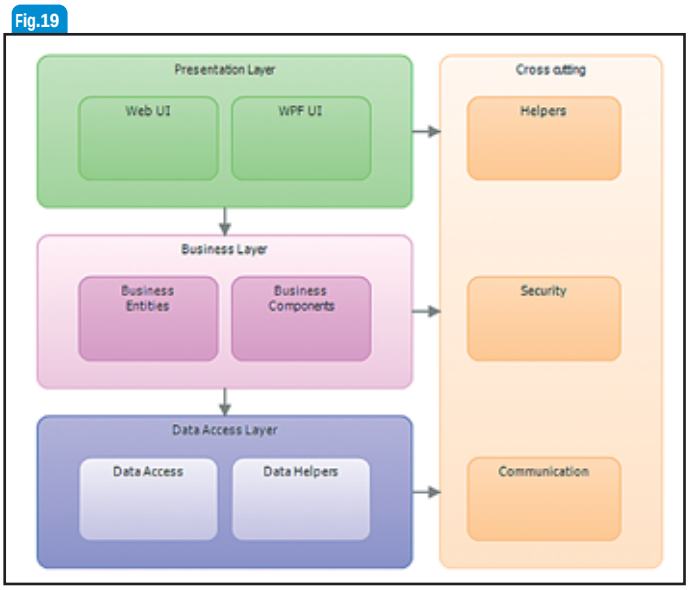


Fig.19



Les différents types de diagrammes sont au nombre de sept, cinq pour l'UML (Class, Sequence, Use Case, Activity et Component), le diagramme de couche logique et enfin le diagramme générique DGML (pour Directed Graph Markup Language) : **(Fig. 15)**.
Architecture Explorer est un concept assez nouveau, un peu déroutant au premier abord, mais qui deviendra vite indispensable lorsque l'on travaille sur des gros projets ou lorsque l'on reprend un projet existant. Celui-ci vous permet de parcourir un Project, une Solution ou des fichiers exécutables afin d'y découvrir les différents éléments tels que les espaces de noms, classes, méthodes, propriétés, etc. **(Fig. 16)**.
Lorsque vous avez ciblé l'élément qui vous intéresse, vous avez la possibilité de générer un diagramme relationnel **(Fig.17)**.
Vous avez aussi la possibilité de générer un diagramme relationnel à partir d'exécutables et en choisissant la granularité de restitution selon les types : assemblage, espace de nom, classe, méthodes et leurs expositions : public, internal, protected et private **(Fig.18)**.

Validation de la couche logique

Microsoft prône souvent le pragmatisme lorsqu'il s'agit de sa suite de développement ALM, une fonctionnalité démontre particulièrement cet état d'esprit : la validation du diagramme de couche logique à travers la compilation.
Une des responsabilités majeures de l'architecte logiciel est de définir pour une application tout un ensemble de couches logiques représentant les différentes parties de celle-ci, puis de définir les interactions possibles entre elles et veiller à ce qu'elles soient respectées dans le code.
Visual Studio 2010 prend ce scénario en charge intégralement.
La création d'un diagramme de couche logique nous permet de modéliser l'architecture logique **(Fig.19)**
Ensuite, nous pouvons attacher des éléments du projet (assemblage, espace de nom, classe, etc.) à des couches spécifiques. Et enfin, nous spécifierons pour chaque diagramme s'il valide ou non l'architecture, et la compilation du projet se chargera de vérifier l'intégrité de celui-ci. Lorsqu'un développeur utilise un objet sans que la liaison entre les couches soit modélisée, une erreur sera générée. De plus, il est possible d'automatiser cette vérification pour qu'elle génère une erreur pendant un Build !

■ Loïc **Baumann**
14 ans d'expérience dans le monde du développement logiciel. Il est actuellement Architecte Logiciel et expert en ALM chez Winwise ainsi que MVP Team System.
http://loicbaumann.org

■ Guillaume **Rouchon**
8 ans d'expérience dans le monde du développement logiciel. Il est actuellement Consultant / Formateur .NET et expert ALM chez Winwise.
http://blog.qetza.net

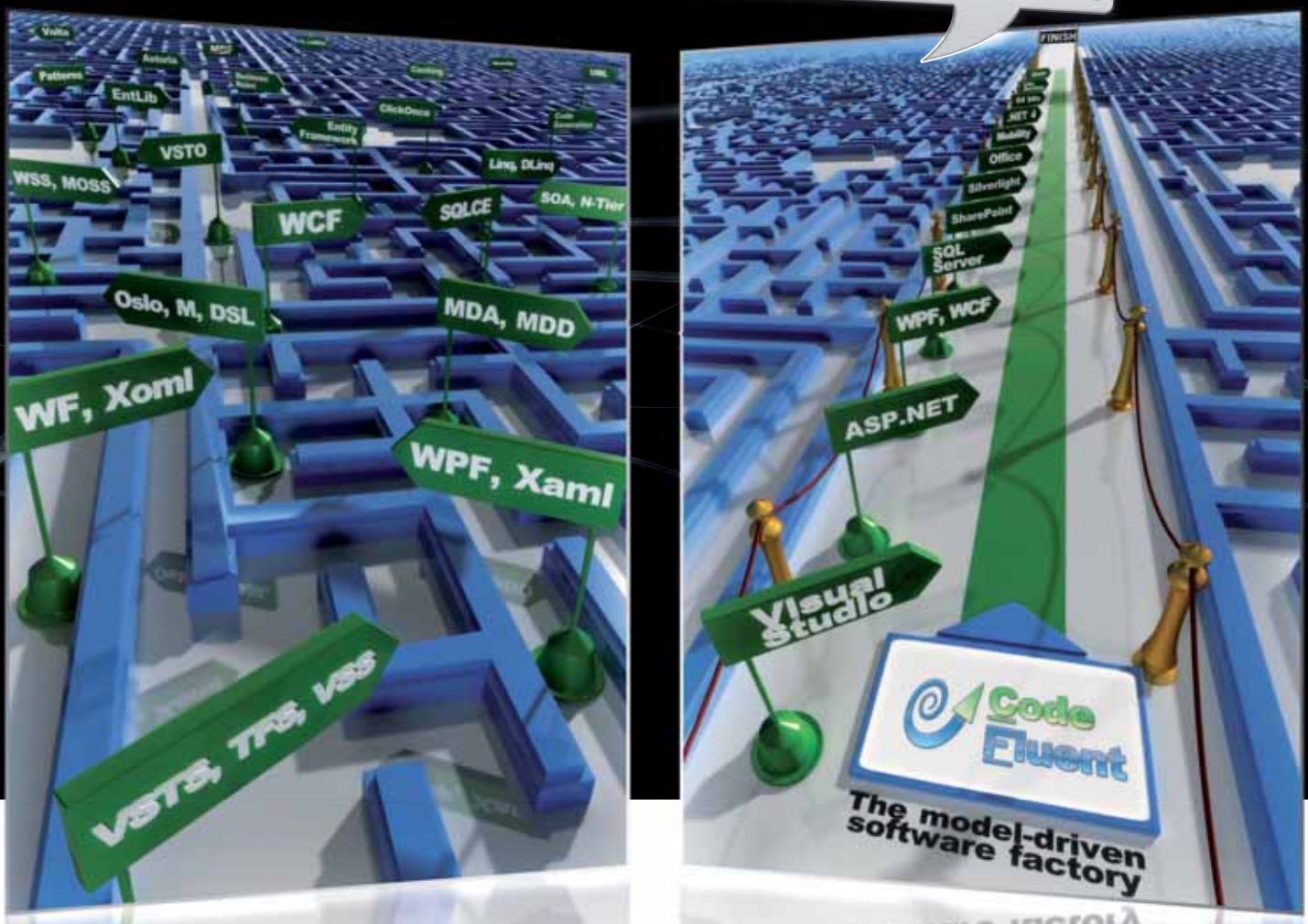
■ Vincent **Labatut**
10 ans d'expérience dans le monde du développement logiciel. Il est Consultant / Formateur .NET et expert ALM chez Winwise.
http://blogs.codes-sources.com/vlabz

CODEFLUENT

La première fabrique logicielle .NET entièrement pilotée par les modèles

Téléchargez la dernière version **gratuite** sur **www.codefluent.com**

sans **avec**



Le plus court chemin vers les technologies .NET

CodeFluent est un produit de génie logiciel qui permet d'industrialiser la fabrication d'applications professionnelles manipulant des données sur la plate-forme .NET en **automatisant la création des composants** à partir d'une modélisation de votre métier. L'utilisation de CodeFluent vous assure **évolutivité**, **productivité**, **qualité** et **facilité de maintenance**.

SoftFluent
3 rue de la Renaissance - 92160 Antony
01 75 60 04 45 - sales@softfluent.com

idEES
L'innovation commence ici

« Formations en soirée » : une expérience originale et motivante.

Depuis 2 ans, la SSII Objet Direct, spécialisée dans les nouvelles technologies, a mis en place un dispositif original de formations internes, éligibles au DIF, et donc rémunéré à hauteur de 50% du salaire.

Objectifs de la SSII : favoriser la montée en compétences de ses collaborateurs, donner un cadre à la capitalisation technique, encourager la formation continue et l'ouvrir au plus grand nombre sans pénaliser l'entreprise par un déficit de facturation, proposer des stages adaptés et de bonne qualité, exploiter le bénéfice d'abriter au sein de l'entreprise un centre de formation aux nouvelles technologies, permettre aux salariés de bénéficier du DIF (droit individuel à la formation),...

Sur la base du volontariat

Chez Objet Direct on appelle le dispositif : « Formation en Soirée » : F.E.S. Il est ouvert à tous les salariés, sur la base du volontariat. Le but de ces formations est de permettre aux consultants de découvrir et d'utiliser concrètement des technologies ou méthodologies qu'ils n'ont pas encore eu l'occasion de mettre en œuvre dans leurs missions. Ces formations se déroulent sur un format plus court que les stages classiques. Organisées dans les agences Objet Direct à Paris, Lyon et Grenoble, les sessions de formation rassemblent 8 à 12 collaborateurs sur une tranche horaire 18h30 - 22h30. Un repas est servi pendant la pause. Une formation complète se compose en général de 2 à 3 sessions.

Les consultants forment les consultants

Les sujets sont proposés par les consultants-formateurs eux-mêmes, et sélectionnés en fonction des thèmes qui rassemblent le plus d'adhésion. Quelques exemples : Formation à la méthodologie agile Scrum, Développement .NET avec Visual Studio 2008, Architectures JEE et EJB3, Intégration continue avec Maven2, Développer des RIA avec le Google Web Toolkit (GWT), Applications Web : XHTML / CSS – Bonnes pratiques, etc. *« Ces formations sur mesure développent un thème technique ou méthodologique en se basant principalement sur du vécu (exprimé par le formateur) et en allant à l'essentiel »,* estime Vianney Grassaud, consultant Objet Direct et formateur. *« C'est très enrichissant. Après une F.E.S, on est vraiment capable d'appréhender au mieux la techno ou la méthodologie présentée. La qualité du matériel pédagogique fourni par Objet Direct (présentation PowerPoint, supports de cours et TP) est aussi un atout. »*

Se former en dehors du temps de travail et bénéficier du DIF

Les formations se déroulent en soirée, donc en dehors temps de travail et sont éligibles au DIF (Droit individuel à la formation). C'est-à-dire que le temps passé en formation est rémunéré à hauteur de 50% du salaire. Ainsi, la participation à une F.E.S. de 4 heures rapporte l'équivalent de 2 heures de salaire net. Ce dispositif mis en place depuis 2 ans a permis à Objet Direct de réaliser 25 % de son plan de formation 2009. *« Au-delà des statistiques, l'impact sur la motivation des salariés et le climat social est très positif. Ces formations en soirée permettent en effet aussi de se former au métier de formateur ; Une formation en Soirée est parfois l'ultime étape de mise en situation avant le passage de la certification qui permettra au consultant d'animer des formations « classiques » face à des clients »,* explique Franck Priore, Directeur des Opérations. C'est d'ailleurs pour le formateur que l'effort est intense, la préparation d'une formation demandant un gros investissement temps, difficilement conciliable avec les horaires habituels de mission. Le travail de préparation empiète souvent sur le temps personnel, sans réelle contrepartie financière puisque le DIF ne bénéficie pas au formateur. Ces formations sont également appréciées pour leur aspect convivial : dîner et ambiance détendue. *« Les F.E.S. ont pour vocation de mettre le pied à l'étrier d'une technologie ou d'une méthodologie. Elles sont très orientées pratique, et même "bonnes pratiques" et intègrent une grande proportion de TP ce qui les rend particulièrement efficaces, souligne Rémi Patriarche, consultant Objet Direct. Elles sont aussi très actuelles, axées sur les dernières tendances technologiques et sont ainsi le reflet d'une réelle veille technologique participative. »*

■ Jean Kaminsky

Comment choisir son école

Il y a encore peu de temps, les entreprises à la recherche d'informaticiens s'arrachaient les jeunes diplômés à la sortie des écoles d'ingénieurs.

D.R.

Aujourd'hui, les temps sont devenus plus durs pour tous, y compris les jeunes diplômés. Raison de plus pour bien choisir son école, d'autant que certaines d'entre elles annoncent que tous leurs élèves sont recrutés dans un délai très bref, et souvent en CDI.

Le premier réflexe est de s'intéresser au taux de recrutement des jeunes diplômés à la sortie des écoles. Celui-ci est souvent lié à la notoriété de l'école. Ainsi, c'est grâce à elle qu'en France et à l'international, les Insa annoncent *« presque 100% des diplômés embauchés en moins de 3 mois. »* Selon une enquête premier emploi 2010 portant sur l'insertion des ingénieurs de la promotion Efrei 2009 (300 ingénieurs formés par an), *« les jeunes ingénieurs Efrei ne connaissent pas la crise. »* En effet, en sortant de cette école, le délai moyen d'insertion est inférieur à 1,5 mois, avec 83% de propositions de pré-embauche. Des taux assez proches de ceux de l'Epita, dont 80 à 95% des étudiants ont trouvé du travail moins de deux mois après la fin des études. Quant à l'Eisti : *« Jusqu'à présent, 100% de nos élèves signaient un contrat d'embauche avant même l'obtention de leur diplôme d'ingénieur »,* assure Marie-Pierre Piguet, responsable administrative de l'Eisti.

Quels débouchés pour quelles écoles ?

A y regarder de plus près, que cherchent les recruteurs ? et quels sont les points forts des écoles ? C'est ce que nous allons essayer de mettre au jour à travers les présentations d'un certain nombre de ces écoles formant à bac+5. Le premier critère, à savoir la reconnaissance du diplôme, n'est pas suffisante en situation de pénurie de postes. La notoriété – et la cooptation par les informaticiens et développeurs déjà en situation – est certes importante. *« Nous choisissons les jeunes diplômés en fonction de la connaissance que nous avons de l'école (Ensimag, pour la finance, par exemple) »,* reconnaît François Salaun, PDG de Softeam. *« Le choix d'un candidat est un équilibre entre deux facteurs : prestige de l'école et personnalité du candidat. »*

La société de conseil et ingénierie en informatique Devoteam recrute 15 à 20% de jeunes diplômés sur environ 500 recrutements par an. Soit près de 100 jeunes diplômés et environ 70 stagiaires par an. *« Nous recrutons des diplômés d'écoles comme l'Epita, Isep, Efrei, etc., pour la solidité de la formation scientifique et technique, d'une part, et les qualités relationnelles qu'elles développent, d'autre part »,* indique **Benoît Castel**, responsable de l'équipe recrutement France de Devoteam. *« Nous attachons aussi de l'importance à la personnalité du candidat, ainsi qu'aux expériences professionnelles (stages) et à l'étranger »,* ajoute François Salaun.


Benoît Castel

Technologie versus personnalité

Les entreprises peuvent apprécier d'embaucher des jeunes rapidement opérationnels, grâce à une formation pratique étayée par des stages en entreprise tout au long de la formation. Le candidat idéal pour les entreprises serait un professionnel de l'informatique immédiatement opérationnel, tout en ayant un profil ouvert et adaptable tout au long de la vie professionnelle. Il en résulte un compromis entre deux types de formations : certaines écoles mettent l'accent sur les technologies et les applications, d'autres sur la formation scientifique, plus théorique, avec la capacité d'adaptation. Ainsi l'Insia, école d'informatique appliquée, met l'accent sur les cours techniques, par exemple C++, JEE, SGBD, PHP,... technologies enseignées au travers de TP en salles machines, ou de TD, plutôt que sur la théorie. Les enseignements dans cette école passent souvent par des projets comme le déploiement et la configuration de services systèmes et réseau sur Linux, Windows, BSD, ou la réalisation de robots compétitifs autonomes. Plutôt que des compétences dans des technologies particulières, comme Java, JEE, etc., Devoteam s'attache plutôt à l'agilité intellectuelle et à l'adaptabilité des candidats : *« Les technologies changent sur 40 ans de carrière. Pour faire un parcours dans l'informatique, il est plus précieux d'avoir l'agilité intellectuelle »,* affirme Benoît Castel.

Quant aux qualités relationnelles, elles sont essentielles pour la gestion de projet, le travail en équipe. *« Nous apprécions les écoles où l'accent est mis sur des travaux collectifs, avec mise en situation. Les élèves apprennent à travailler en réseau, à tenir un calendrier, gérer un projet, faire du reporting »,* souligne Benoît Castel. *« Outre la motivation du jeune, nous*

apprécions la capacité à présenter un projet cohérent, l'aisance, l'esprit de synthèse. » Dans certaines écoles, ces qualités sont développées au travers de « piscines », où l'étudiant est immergé à la fois dans la technologie et dans le travail en équipe. A l'Insia, une « piscine » est une période intense de TP qui s'étale sur 2 à 3 semaines au début de l'année, où l'élève suit des cours sur des matières phares, dans une ambiance de collaboration et d'émulation, visant à la fois à créer un esprit de corps et à acquérir rapidement des connaissances techniques de pointe. A In'Tech Info, chaque semestre, les élèves en équipe de 4 à 6 réalisent deux projets concrets d'envergure, en situation de travail semblables à celles des entreprises.

Diverses démarches pédagogiques

Entre les approches traditionnelles, fondées sur les cours magistraux, et la pédagogie de projets, où l'accent est mis sur les stages et la quasi-totalité du travail se fait en petites équipes, sur des problématiques proches de la réalité professionnelle, il y a tout un éventail de types de pédagogies dans les écoles d'informatique.

Dans la première catégorie, citons l'Epsi *« où tout le cursus est fait par des professeurs selon des méthodes d'enseignement traditionnel »*, comme l'explique son directeur général, Albert Cohen. Les élèves y sont très encadrés : suivis tout au long de leur scolarité, leur présence aux cours et le travail personnel sont obligatoires. A l'autre extrême, In'Tech Info a choisi de calquer son environnement sur celui des entreprises : la pédagogie est fondée sur le travail en projets, l'école étant conçue comme une « communauté de travail », avec des équipes pédagogiques jeunes et disponibles, la relation entre enseignants et élèves préfigurant le mode de fonctionnement des entreprises, avec un accompagnement individuel de type coach ; dès l'entrée à l'école, l'élève se voit attribuer un tuteur, souvent un ingénieur en activité, qui a pour rôle de l'épauler et l'aider à définir son projet professionnel. L'Exia.cesi prône également la pédagogie par projets. *« L'essentiel du contenu de ma formation me sert tous les jours, en particulier tous les aspects liés au management de projets »,* affirme un ancien élève de cette école, aujourd'hui ingénieur de projet dans une SSII. De même pour l'Epitech, où la pédagogie se résume en une phrase : *« Former autour des projets, où l'étudiant est actif dès le début »,* selon Nicolas Sadirac, directeur de cette école. Le tutorat est aussi l'un des points forts de l'Eisti, qui insiste sur la rigueur et l'ouverture d'esprit.

« Dans le domaine pédagogique, nous sommes dans une période d'évolution majeure », souligne **Fabrice Bardèche,** vice-président du groupe Ionis, auquel appartiennent l'Epita et l'Epitech. *« L'Epita forme des scientifiques. C'est l'école de l'intelligence informatique : on y apprend la culture scientifique globale, à restituer ses compétences techniques, ainsi que les connaissances managériales. La spécialisation se fait en dernière année. Elle comporte une réflexion sociétale, économique. »*

Toutes les écoles mettent l'accent sur l'importance des stages, même si certaines écoles les privilégient plus que d'autres. Ainsi, sur les 3 ans de formation dispensée à l'Ensea, plus de 7 mois sont consacrés aux stages en entreprise. L'Efrei associe, dès le cycle préparatoire intégré, la formation scientifique et technique à la formation générale et professionnelle de l'ingénieur, avec quatre stages obligatoires et un stage facultatif, sans faire l'impasse sur les cours magistraux en amphi. *« Cette alliance généraliste est perçue très positivement par les entreprises, en ce sens qu'elle permet une grande adaptabilité des élèves dès leurs premiers stages »,* affirme la direction de l'Efrei.

L'Ece associe les « majeures » couvrant la maîtrise des technologies, aux « mineures » pour l'acquisition des savoir-faire professionnels : s'initier au marketing des technologies, créer une start-up, s'engager dans la vie associative, mener un projet de recherche scientifique… Coachés par des responsables professionnels, les élèves suivent le domaine qu'ils ont choisi.

A la pédagogie, il faut associer étroitement la vie associative et autres activités de l'école. Ces activités, qu'elles soient culturelles ou sportives, voire festives, sont non seulement un facteur d'attraction pour les futurs élèves, mais peuvent également forger une personnalité et développer des qualités appréciées par les entreprises, comme l'épanouissement personnel, la créativité, l'esprit d'équipe. *« Une vie associative qui fait partie de la formation : prises de responsabilité, expérience entrepreneuriale, sens de l'action collective… »,* indique-t-on à Supélec.

Fabrice Bardèche

Multilinguisme et ouverture à l'international

L'international est devenu obligatoire dans de nombreuses écoles. Les cursus internationaux, qu'il s'agisse d'une année ou d'un stage dans un autre pays, de projets au sein d'universités à l'étranger, sont appréciés par toutes les entreprises et considérés comme un atout par les élèves. *« Les deux axes majeurs actuels de développement de l'école sont l'international et l'entrepreneuriat »,* assure Michel Ciazynski, directeur général de l'Isep, une école qui compte parmi ses partenaires des universités prestigieuses du monde entier, et accueille également des étudiants étrangers dans son cursus ingénieur. *« C'est aussi la chance pour les étudiants français de bénéficier d'un environnement international à l'intérieur même de l'école »,* affirme-t-on à l'Isep. Les élèves de l'Exia.cesi peuvent, par le biais de passerelles, suivre la filière recherche au sein de l'UQAM (Universi-

D.R.

té du Québec à Montréal). Les élèves de l'Epitech passent la 4e année à l'étranger, de préférence dans un pays anglophone (Etats-Unis, Canada, Australie, Inde...), mais aussi Chine ou Russie. *« L'objectif est d'acquérir une expérience du travail avec des étrangers, la compréhension de l'international et l'interculturalité »,* précise Nicolas Sadirac. Supinfo ouvre progressivement des écoles hors de France : au Canada, en Chine, au Maroc, à Londres, avec un rythme d'ouverture d'environ 7 écoles par an, et une nouvelle identité : « Supinfo International Universities ». Cet essaimage donne aux étudiants la possibilité de mobilité, ce qui est hautement apprécié par nombre d'entre eux.

La mobilité dans l'espace européen et international est une priorité majeure pour l'Esiea : après la Finlande en 2007 et les Pays-Bas en 2008, l'école a signé un nouvel accord avec Anglia

D.R.

Ruskin University, Royaume-Uni : les élèves de 3e année peuvent effectuer un semestre d'études de niveau Bachelor of Science sur l'un des campus d'ARU, Chelmsford ou Cambridge. Dans le même temps, si cela est nécessaire, ils peuvent bénéficier de cours de renforcement d'anglais afin d'améliorer leur score au TOEIC. L'Esiea a aussi un accord avec King Mongkut's University of Technology Tonbury en Thaïlande, permettant des échanges d'étudiants, des projets inter-universitaires et des rencontres entre établissements. L'Efrei multiplie les partenariats avec les universités et les entreprises à l'étranger : 66 partenariats actifs répartis dans 29 pays, qu'il s'agisse d'échanges académiques (Erasmus, Crepuq, échanges bilatéraux avec des pays d'Asie, Russie, Etats-Unis, Mexique...), de séjours d'immersion ou de stages. Elle propose 30 doubles diplômes (Master of Science). L'Epitech, pour sa part, totalise plus de 100 universités et instituts partenaires dans 42 pays.

L'apprentissage des langues est aussi important. L'anglais est obligatoire pour pratiquement tous les étudiants en informatique. A l'Ensimag, 95% pratiquent une seconde langue. L'Efrei accorde une importance particulière à l'apprentissage de l'anglais et d'une seconde langue étrangère. A l'Ensea également, l'apprentissage de deux langues est obligatoire. 25% de la promotion suit une partie de son cursus à l'étranger. Dans certains cas, les étudiants ont la possibilité d'obtenir un double diplôme. A l'Esigetel, où 20% des cours sont dédiés à l'international, un centre de langues réunit des professeurs d'anglais, allemand, espagnol, japonais, chinois, russe, italien et arabe, et l'école forme, dès la 3e année, au « Business English » sur différents thèmes professionnels. Certaines écoles exigent un score TOEIC pour décerner leur diplôme. *« Maîtriser une langue étrangère est un élément essentiel, surtout l'anglais. Par ailleurs, cela donne une ouverture d'esprit et permet de développer des qualités relationnelles, ainsi que la capacité d'adaptation à un autre environnement, à une autre langue »,* indique Benoît Castel.

L'Ece a depuis longtemps développé des partenariats avec de grandes universités étrangères : séminaires obligatoires, semestres d'études à l'étranger, offre de masters en dernière année (double cursus), summer schools, stages à l'étranger, cursus spécifiques (sections internationales, mineure « international », voies d'approfondissement en anglais...). Ainsi, les étudiants du cycle prépa bénéficient d'un semestre entier de formation à l'université anglophone de Concordia à Montréal pour commencer leur cycle d'ingénieur. La dernière année du cycle ingénieur débute par un séminaire « global management » d'un mois à l'University of California, Irvine, près de Los Angeles. De plus, ceux qui le souhaitent peuvent effectuer la dernière année ou un stage à l'étranger, grâce à des conventions avec différents pays (Australie, Angleterre, Espagne, Danemark, Norvège, Suède, Finlande, Québec, Pologne, Pérou, Norvège, Afrique du sud, Mexique, Brésil, Chine, Islande). La mineure « international » a pour objectif de préparer les étudiants à partir travailler à l'étranger et à étudier la culture du pays de destination. Depuis la rentrée 2005, l'Ece propose des sections internationales à ses élèves de 2e et 3e années de cycle ingénieur, avec un cursus en langue anglais pour la quasi-totalité des cours, TP, TD et trois majeures : systèmes d'information et réseaux, systèmes embarqués, télécoms et réseaux.

Connaissance des entreprises et partenariats

Souvent implantées dans une zone constituant un bassin d'emploi informatique, les écoles facilitent de fait le contact entre les élèves et les entreprises. Ainsi, chaque campus de Supélec (Gif-sur-Yvette, Rennes, Metz) est implanté dans un environnement technologique de haut niveau. Fondée par le GPNI, chambre syndicale des SSII, l'Epsi est par nature proche d'un vivier d'emplois potentiels. Ses différentes implantations répondent à un besoin local de personnel. Les partenariats avec les entreprises sont un atout pour les élèves. A l'Insia, un « cercle des parrains », regroupant les entreprises partenaires, vise à devenir la référence en termes de partenariat entre les univers de la formation et de l'entreprise. Des personnes en activité dans les entreprises peuvent être chargées d'enseignement. Par exemple, des consultants de Devoteam interviennent dans les enseignements de certaines écoles : *« Il en résulte une interpénétration des milieux assez forte »,* indique Benoît Castel. Par ailleurs, les stages en entreprise, qui sont un passage obligé dans la plupart des écoles, permettent aux élèves de mieux connaître le milieu de l'entreprise. C'est d'ailleurs souvent le mode de recrutement préféré des entreprises. Ainsi Devoteam reçoit près d'une soixantaine de stagiaires par an. *« Pour les stages, nous privilégions les étudiants qui viennent à nos conférences et se présentent à nos consultants enseignants »,* souligne

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 .

Benoît Castel. *« C'est l'expression d'une motivation plus forte que celle qui peut ressortir d'une candidature spontanée : cela prouve que l'élève n'arrive pas par hasard. »* La relation entre école et entreprise rassure souvent cette dernière sur la capacité du jeune diplômé à tenir son poste, à s'exprimer… Quant à l'élève, par ses contacts avec des personnes issues du milieu professionnel, il a déjà une première idée de ce qu'il aimerait faire. *« Cela fait partie de la motivation »,* ajoute Benoît Castel.

Supélec met en avant le haut niveau de recherche et la qualité de l'enseignement, grâce à une étroite liaison avec l'industrie, notamment via les contrats de recherche. *« Les formations conçues pour et avec les entreprises permettent de réaliser un véritable système de transfert des connaissances entre l'école et les entreprises. »* L'Exia.cesi fait appel à des experts du secteur informatique, du management ou du droit pour intervenir au cours de conférences ou suivre des projets. Le partenariat entre In'Tech Info et les grandes entreprises couvre un large spectre d'actions : élaboration de projet pédagogique, définition des compétences et des métiers, délégation de tuteurs, de chefs de projets, offres de stages, contrats d'alternance, cofinancement de la formation. Les entreprises partenaires animent la moitié des modules, évaluent les projets présentés à l'occasion des forums et peuvent confier aux élèves des projets informatiques à développer.

Des partenariats avec des éditeurs ont une grande importance dans certaines écoles, comme l'Exia.cesi, mais surtout Supinfo, qui grâce à ces partenariats a mis en place des « laboratoires pédagogiques » consacrés aux différentes technologies (Microsoft, Cisco, Apple, Oracle, IBM…). De tels partenariats permettent aux élèves de bénéficier en avant-première des évolutions technologiques et de l'utilisation privilégiée des logiciels. *« Nous considérons les partenariats avec des éditeurs de logiciels comme très importants pour la formation à ces produits »,* souligne Mauricio Diaz Orlich, directeur des laboratoires de Supinfo.

Comment financer ses études

Enfin, les aspects de financement ne sont pas le moindre critère à considérer en période de difficulté économique. Si beaucoup de formations bac+2 sont publiques (université, math sup et spé dans les lycées d'Etat), les étudiants qui souhaitent acquérir un niveau bac+5 sont confrontés au choix entre deux filières : celles de l'Etat, gratuites mais difficiles d'accès car très sélectives, et les écoles privées, relativement coûteuses : quelques milliers d'euros par an, soit plusieurs dizaines de milliers de k€ pour la totalité de la formation. Toutefois, le prix des études peut être réduit grâce au cofinancement avec des entreprises (Insia ou In'Tech Info, par exemple), par des bourses pour les écoles reconnues par l'Etat, ou des prêts. Prêts bancaires étudiants à tarif préférentiel, rémunération des stages et de l'alternance, missions rémunérées… Les banques partenaires de l'Eisti proposent aux élèves-ingénieurs un financement intégral de leurs études, sans caution parentale, à un taux raisonnable, remboursable une année après l'entrée dans la vie professionnelle sur 36 ou 72 mois. *« Désormais, nous accompagnons ce financement des études par une mesure forte : la prise en charge par l'Eisti des intérêts du prêt bancaire tant que le jeune diplômé n'a pas acquis un contrat d'embauche »,* assure Marie-Pierre Piguet. Quant aux séjours à l'étranger, *« qu'il s'agisse de stages ou d'études, c'est une question de curiosité et de volonté plutôt qu'une stricte question de finances »,* ajoute Marie-Josée Lamerre, responsable relations internationales de l'Eisti. *« De nombreux dispositifs ont été mis en place par la Région Île-de-France et le Conseil Général du Val-d'Oise pour soutenir matériellement les étudiants. »* Quoi qu'il en soit, la difficulté concerne surtout les premières années, puisque, plus l'élève avance dans le cursus, plus les stages lui permettent de financer ses études.

■ Claire **Rémy**

Ecoles informatiques (niveau bac+5) : où les trouver ?

(liste non exhaustive)

Ecole	Situation géographique	Site web
ECE (école centrale d'électronique)	Paris	www.ece.fr
EFREI (école française d'électronique et d'informatique)	Villejuif	www.efrei.fr
EISTI (école internationale des sciences du traitement de l'information)	Cergy, Pau	www.eisti.fr
ENSEA (école nationale supérieure de l'électronique et de ses applications)	Cergy	www.ensea.fr
ENSIIE (école nationale supérieure d'informatique pour l'industrie et l'entreprise)	Evry	www.ensiie.fr
ENSIMAG - INP	Grenoble	www.ensimag.fr
EPITA	Paris	www.epita.fr
EPITECH	Paris, Bordeaux, Lille, Lyon, Marseille, Nancy, Montpellier, Nantes, Nice, Rennes, Strasbourg, Toulouse	www.epitech.fr
EPSI (Ecole Privée des Sciences de l'Informatique)	Paris, Bordeaux, Montpellier, Arras, Nantes, Lyon	www.epsi.fr
ESGI (école supérieure de génie informatique)	Paris	www.esgi.fr
ESIEA (école supérieure d'informatique, électronique, automatique)	Paris, Laval	www.esiea.fr
ESIGETEL (école supérieure en informatique et génie des télécommunications)	Fontainebleau-Avon	www.esigetel.fr
EXIA.CESI (école supérieure d'informatique du groupe CESI)	Arcueil, Pau, Rouen, Toulouse (et autres régions)	www.exia.cesi.fr
IGM (institut d'électronique et d'informatique Gaspard Monge)	Marne-la-Vallée	www-igm.univ-mlv.fr
INSA – informatique	Lyon, Rennes, Toulouse	www.insa.fr
INSIA (institut supérieur d'informatique appliquée)	Paris	www.insia.org
IN'TECH INFO	Paris	www.intechinfo.fr
ISEP (institut supérieur d'électronique de Paris)	Paris	www.isep.fr
SUPELEC (école supérieure d'électricité)	Gif-sur-Yvette, Rennes, Metz	www.supelec.fr
SUPINFO	Paris, Strasbourg, Bordeaux, Rennes, Montpellier, Lille, Marseille,Metz, Reims, Martinique,Réunion, Guadeloupe	www.supinfo.com

Go, Chrome OS et Closure Tools : pas de répit chez Google

Entre la révélation du projet Chrome OS début juillet 2009 et l'annonce le 19 novembre dernier de l'ouverture à tout le monde de son code source, Google poursuit son agenda technologique en annonçant non seulement la sortie de ses Closure Tools, une suite d'outils pour JavaScript, mais également son propre langage de programmation, Go.

Go : "New, Experimental, Concurrent, Garbage-Collected, Systems Language"

Le langage Go est l'aboutissement d'un projet initié en 2007 par Ken Thompson, (créateur du langage B, précurseur du C, et employé Google depuis 2006), Robert Griesemer et Rob Pike, auteurs du format d'encodage très répandu : l'UTF-8.

Hello, World !

Comment présenter un nouveau langage sans montrer son « Hello, World » ? Voici donc à quoi ressemble le fameux Hello World en Go :

```
package main

import "fmt"  / /  Package implementing formatted I / O

func main() {
  fmt.Printf("Hello, World !\n")
}
```

Chaque fichier source Go déclare, en utilisant le mot-clé package, à quel package il appartient (ici, le package « main »). Il peut également importer d'autres packages afin d'en utiliser ses fonctions. Ici, nous importons le package « fmt » afin d'avoir accès à la méthode fmt.Printf. Les fonctions sont déclarées avec le mot-clé func. La fonction main est le point d'entrée du programme, tout comme en Java et en C / C++. Les chaînes de caractères peuvent contenir des caractères Unicode. D'ailleurs, les fichiers source Go sont, vous l'aurez deviné, encodés en UTF-8 par défaut. La convention pour les commentaires est la même qu'en Java, C et C++ :

```
/* … */
// …
```

Les points-virgules sont utilisés comme en C, en C++ et en Java, mais dans beaucoup de cas, on peut s'en passer. Cela s'inscrit dans l'optique de Google de rendre la programmation plus rapide, et de ce fait, plus agréable. Les points-virgules séparent les instructions (et les déclarations) plutôt que de les terminer, ils sont donc facultatifs après la dernière déclaration d'un bloc.
Illustrons ceci avec un exemple de déclaration de constantes.

```
const (
  Space = " ";
  Newline = "\n";
)
```

On peut regrouper les déclarations de même type dans une liste parenthésée, séparée par des points-virgules (à l'exception, comme on vient de le voir, du dernier élément de la liste).

```
const (
  Space = " ";
  Newline = "\n" // On peut omettre le dernier point-virgule
)
```

On pourrait également écrire :

```
const Space = " "
const Newline = "\n"
```

Jetons maintenant un œil à la déclaration et l'affectation de variables avec Go. On utilise le mot-clé *var*, suivi du nom de la variable, suivi de son type, suivi du signe égal, et d'une valeur initiale :

```
var s string = "" ;
```

Mais cette déclaration peut être raccourcie : puisque la valeur "" est de type string, nous n'avons pas besoin de le signaler au compilateur. On pourrait ainsi écrire :

```
var s = "" ;
```

Mais on ne s'arrêtera pas là ! On peut davantage raccourcir et écrire :

```
s := "" ;
```

L'opérateur := est beaucoup utilisé dans Go pour représenter une déclaration et une affectation simultanée.
On remarque donc qu'avec cette afférence de type, Go se veut être un langage dynamique. C'est le principe du *Duck Typing* : *« Si je vois un animal qui vole comme un canard, crie comme un canard, et nage comme un canard, alors j'appelle cet oiseau un canard ».*
Les boucles *for* ont également changé dans Go, les parenthèses de la clause ont disparu, accélérant ainsi un peu plus le codage :

```
for i := 0; i < 10; i++ { }
```

Notons que dans le cas de clauses for, les points-virgules sont indispensables.

La compilation :

A ce jour, il existe deux compilateurs : gccgo est un compilateur Go qui repose sur le compilateur GCC. Il existe aussi une suite de compilateurs avec des noms qui varient en fonction de l'architecture avec, entre autres : 6g pour les 64-bit x86, et 8g pour les 32-bit x86. Ces derniers compilent plus rapidement que le gccgo mais génèrent du code un peu moins efficace. Voyons comment compiler notre helloworld.go avec ces différents compilateurs.

Avec 6g / 8g :

```
$ 8g helloworld.go    # compile;   object goes into helloworld.8
$ 8l helloworld.8     # link;    output goes into 8.out
$ 8.out
Hello, world !
$
```

Avec gccgo :

```
$ gccgo helloworld.go
$ a.out
Hello, world !
$
```

La nomenclature des compilateurs provient du projet Plan 9 de Bell Labs créé, entre autres, par Ken Thompson et Rob Pike, dont les différentes versions de compilateurs de fichiers C sont les suivantes :

`0c spim`	little–endian MIPS 3000 family
`1c 68000`	Motorola MC68000
`2c 68020`	Motorola MC68020
`5c arm`	little–endian ARM
`6c amd64`	**AMD64 and compatibles (e.g., Intel EM64T)**
`7c alpha`	Digital Alpha APX
`8c 386`	**Intel i386, i486, Pentium, etc.**
`kc sparc`	Sun SPARC
`qc power`	Power PC
`vc mips`	big–endian MIPS 3000 family

Langage dynamique… Et compilé ?

En fait, Google a essayé de garder le meilleur des deux mondes. Fortement inspiré du C++ et du Python, Google explique que "Go combine la vitesse de développement quand on programme avec un langage dynamique comme Python, avec l'efficacité et la sécurité des langages compilés comme C ou C++". C'est un langage compilé et multitâche, et Google s'est efforcé de respecter le multithreading (avec des « goroutines », qui remplacent les threads) tout en gardant une performance optimale, proche de celle du C : les performances de Go seraient de l'ordre de 10 à 20% inférieures à celles du C.

« Compiler rapidement du code rapide »

Le fer de lance de Go est sa rapidité de compilation. En effet, lors du Tech Talk, Rob Pike nous démontre que le package tree entier de Go, qui comprend environ 120 000 lignes de code (regroupant toutes les librairies du langage Go, les flags, expressions régulières, et bien plus) se compile en moins de 10 secondes sur un ordinateur portable ! Go fournit également un modèle pour la construction de logiciels qui rend l'analyse de dépendances facile et qui évite ainsi le chargement de la plupart des bibliothèques et des fichiers d'inclusion de type C. Google a aussi voulu améliorer le typage : les types n'ayant pas de hiérarchie sous Go, on pourra ainsi limiter le temps passé à définir leurs relations. Et le langage Go tente de rendre les types plus légers que dans les langages orientés objet classiques.
Donc bien que Go repose sur les notions de types et de méthodes et qu'il permette un style de programmation orientée objet, certains considéreront que l'absence de hiérarchie rend justement ces objets trop légers par rapport à ceux de langages comme Java ou C++. Go est-il donc un langage orienté objet ? Oui et non. Go gère aussi complètement le ramasse-miettes et propose un support fondamental pour l'exécution concurrente, il serait donc particulièrement adapté au développement de logiciels exécutés sur des machines à plusieurs processeurs, ou un seul processeur et plusieurs cœurs d'exécution.

Chrome OS : un système d'exploitation orienté cloud

Chrome OS est le nouveau système d'exploitation de Google. Basé sur une version ultra-allégée de la distribution Debian, il est initialement destiné aux netbooks, et se veut entièrement tourné vers le Web et les services en ligne. Il devrait être disponible au second semestre 2010, mais le code source (projet Chromium OS) est ouvert à tous depuis le 19 novembre. Comme son nom l'indique, il est basé sur le navigateur Chrome, conçu pour répondre à la prolifération d'applications Web de plus en plus puissantes. Chrome OS garde ce cap, et le navigateur devient un système d'exploitation à part entière.

« Rapidité, simplicité, sécurité »

Chez Google, on voulait un système d'exploitation rapide, ils se sont donc débarrassé de tous les processus inutiles, ont optimisé énormément d'opérations et exécutent le plus de traitements en parallèle. Et cela semble payant : Chrome OS bat des records de démarrage : en 5 secondes seulement, l'ordinateur est prêt à être utilisé et connecté à internet.
L'interface est simpliste, elle reprend celle du navigateur Chrome, toutes les applications sont organisées en onglets, et le bureau aussi n'est autre qu'un onglet [Fig.1 et 2].
Toutes les applications sont des applications Web. On y trouvera entre autres les incontournables de la maison : Google Docs pour la bureautique, Gmail pour la messagerie, et Google Talk pour la messagerie instantanée. On ne stocke plus aucune donnée sur l'ordinateur, tout est entièrement en ligne dans le « Cloud ». De ce fait, il n'y a plus besoin d'installer ou de mettre à jour quoi que ce soit. Par contre, il est impossible d'installer des applications classiques

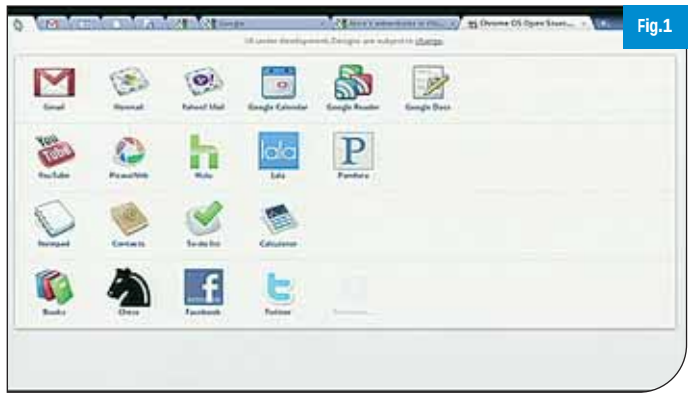


Fig.1

Le bureau de Chrome OS

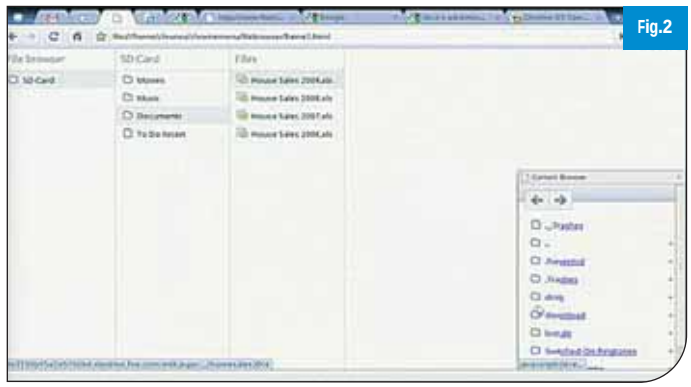


Fig.2

L'explorateur de fichiers de Chrome OS

comme sous Windows ou Linux. Chrome OS est vraiment dédié à l'Internet, et c'est à se demander si Google n'a pas fait un pari un peu osé en limitant cet OS à la navigation sur Internet. Il est vrai que les netbooks sont d'abord utilisés à cet effet, mais certaines personnes qui achètent un ordinateur pourraient certainement avoir envie de faire autre chose que de surfer sur le Web. Vous, par exemple, êtes-vous prêt à laisser tomber vos applications desktop ? La sécurité était aussi une des préoccupations majeures de Google en développant ce système d'exploitation afin que « les utilisateurs n'aient pas à se préoccuper de virus, de malwares, ou de mises à jour de sécurité ». Contrairement aux autres systèmes d'exploitation, Chrome OS ne fait confiance à aucune des applications que vous exécutez. Chacune d'entre elles est lancée dans son propre sandbox (bac à sable), rendant ainsi une infection par un virus ou un malware beaucoup plus difficile.

Chrome OS ne se fait pas non plus confiance : à chaque démarrage de l'ordinateur, l'OS vérifie l'intégrité de son code. Si le système a été compromis, il s'auto répare : une version propre est automatiquement réinstallée. Cela est rendu possible par le fait que toutes les données sont stockées dans le « Cloud » sur un serveur et non sur l'ordinateur. On devrait voir apparaître les premiers netbooks commercialisés avec Chrome OS vers fin 2010, mais si vous souhaitez déjà l'essayer, il existe une version compilée à partir des sources, téléchargeable en bittorrent. L'image est au format VMWare (image VMDK), donc vous aurez besoin de VMWare Player (http://www.vmware.com/products/player/), VMWare Fusion (http://www.vmware.com/products/fusion/) pour ceux d'entre vous qui sont sur Mac, ou encore VirtualBox de Sun (http://www.virtualbox.org/). Site Officiel : http://www.chromium.org/chromium-os

Closure Tools : une suite d'outils JavaScript

Le 5 novembre a vu sortir une collection d'outils JavaScript ayant contribué au développement des applications Web phares de Google comme Google Maps, Gmail ou Google Docs. Google a toujours évangélisé la puissance de JavaScript, et depuis ces applications qui avaient jeté un pavé dans la mare en montrant ses possibilités, il a également sorti, pour le navigateur Chrome, un moteur d'interprétation open-source ECMAScript « V8 », un des plus performants à ce jour. Closure Tools est une suite de 3 outils dont le code est disponible sous licence Apache :

- Closure Library
- Closure Compiler
- Closure Templates

Closure Library : une bibliothèque JavaScript à l'instar de jQuery, MooTools et Dojo

Closure Library est une bibliothèque JavaScript standardisée mettant à disposition des toolkits qui permettent de déployer une application Web et ce, quel que soit le serveur ou le navigateur. On y retrouve, entre autres, la manipulation du DOM (Document Object Model), l'édition WYSIWYG (What You See Is What You Get), les widgets, la sérialisation JSON, la gestion d'évènements et l'animation. Elle étend les fonctionnalités du navigateur et permet d'écrire des scripts plus rapidement avec une API dédiée et une syntaxe spécifique. Regardons un exemple de son utilisation. Vous l'aurez peut-être deviné, un petit Hello World. Il faut tout d'abord télécharger la Closure Library, vous aurez besoin d'un client Subversion afin de la récupérer :

```
svn checkout http://closure-library.googlecode.com/svn/trunk/
closure-library-read-only
```

Une fois téléchargée, on obtient un dossier "closure-library-read-only" contenant tout le code de la Closure Library.

Créons ensuite un script que l'on nommera 'hello.js' :

```
goog.require('goog.dom');

function sayHi() {
  var newHeader = goog.dom.createDom('h1', {'style': 'background-color:#EEE'},
    'Hello world!');
  goog.dom.appendChild(document.body, newHeader);
}
```

Et un fichier HTML qui utilisera ce script :

```
<html>
  <head>
    <script src="closure-library-read-only/closure/goog/base.js"></script>
    <script src="hello.js"></script>
  </head>
  <body onload="sayHi()">
  </body>
</html>
```

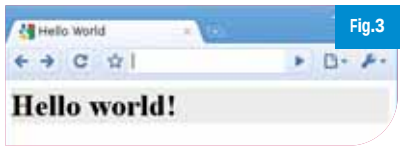


Fig.3

En ouvrant le hello.html, on obtient notre fameux Hello World.

Le script hello.js utilise deux fonctions (*goog.dom.createDom()* et *goog.dom.appendChild()*) qui sont définies dans le fichier *closure-library-read-only / dom / dom.js*. Afin de pouvoir les utiliser, on y fait appel dans le fichier hello.js au moyen de l'expression *goog.require('goog.dom')* ainsi que par l'appel du fichier base.js dans le hello.html. C'est dans le fichier base.js qu'est définie la fonction *goog.require()*. L'appel *goog.require('goog.dom')* charge le fichier JavaScript qui définit toutes les fonctions du namespace goog.dom ainsi que tous les autres fichiers de la Closure Library dont ces fonctions ont besoin. La Closure Library charge ces fichiers en ajoutant dynamiquement une balise de script au fichier HTML. Par exemple, l'expression goog.require('goog.dom') de hello.js entraîne l'ajout d'une balise script dans hello.html :

```
<script src="path-to-closure/dom/dom.js"></script>
```

où *path-to-closure* est le chemin de l'emplacement de hello.html vers le répertoire où se trouve *base.js*. Cependant, bien qu'elle paraisse très utile, cette librairie est encore jeune et il y a déjà quelques retours de la part de développeurs faisant part de soucis de performances, il serait donc imprudent d'abandonner complètement vos librairies éprouvées comme jQuery dès aujourd'hui.

Closure Compiler : optimisez et compressez votre JavaScript

La vocation de ce module est l'efficacité : d'une part, il réduit la taille des fichiers JavaScript (donc le temps d'interprétation de ces fichiers), d'autre part, il vérifie le code, et affiche des erreurs détaillées si le code a été mal rédigé ou s'il pourrait être mal inter-

Fig.4

prété (erreurs de syntaxe, de référencement de variables, ou de types). Il génère donc un script plus propre et plus conforme. Grâce au plugin Firefox Closure Inspector, on peut désormais utiliser le débogueur JavaScript FireBug avec les scripts compilés par Closure Compiler. Closure Compiler a également été intégré au plugin Firefox Page Speed afin d'évaluer le gain de temps qui découlerait de son utilisation. Closure Compiler peut être utilisé de 3 manières différentes :

- Une application Web avec interface graphique
- Une application Java exécutable en ligne de commande
- Une API RESTful

L'application Web

Pour utiliser l'application Web, il suffit de se rendre sur http://closure-compiler.appspot.com et d'indiquer l'URL de votre fichier .js que vous voulez optimiser, ou alors tout simplement faire un copier-coller de votre code JavaScript dans le champ prévu à cet effet, puis d'appuyer sur le bouton 'Compile' : [Fig.4 et 5]

L'application Java en ligne de commande

Tout d'abord, il faut créer un répertoire nommé « closure-compiler ». Ensuite nous devons télécharger le fichier compiler.jar du Closure Compiler, et le placer dans ce répertoire.

Ecrivons ensuite un fichier JavaScript simple: hello.js, et enregistrons-le dans notre répertoire « closure-compiler ».

```
// A simple function.
function hello(longName) {
  alert('Hello, ' + longName);
}
hello('New User');
```

Pour compiler ce fichier, on lance la commande suivante dans la console (à partir du répertoire « closure-compiler ») :

```
java -jar compiler.jar --js hello.js --js_output_file hello-compiled.js
```

Cette commande va créer un nouveau fichier hello-compiled.js qui contient le code suivant :

```
function hello(a){alert("Hello, "+a)}hello("New User");
```

Comme vous pouvez le voir, Closure Compiler enlève les commentaires, les espaces, et les points-virgules inutiles. Il a également remplacé le paramètre « longName » par un nom beaucoup plus court : « a ».

L'API RESTful

L'API RESTful permet d'automatiser le processus d'optimisation, ou alors de l'intégrer dans un système plus important (comme une extension d'IDE, par exemple). Pour faire appel à cette API, il suffit de créer une page HTML qu'on appellera « closure_compiler_test.html » :

Fig.5

```
<html>
  <body>
    <form action="http://closure-compiler.appspot.com/compile" method="POST">
    <p>Type JavaScript code to optimize here:</p>
    <textarea name="js_code" cols="50" rows="5">
    function hello(name) {
      // Greets the user
      alert('Hello, ' + name);
    }
    hello('New user');
    </textarea>
    <input type="hidden" name="compilation_level" value="WHITESPACE_ONLY">
    <input type="hidden" name="output_format" value="text">
    <input type="hidden" name="output_info" value="compiled_code">
    <br><br>
    <input type="submit" value="Optimize">
    </form>
  </body>
</html>
```

Le paramètre « compilation_level » indique au compilateur quel niveau d'optimisation il doit utiliser. Ici, 'WHITESPACE_ONLY' indique qu'il ne doit effectuer que des optimisations les plus basiques (il retire les commentaires et les espaces inutiles).

Il existe d'autres niveaux plus sophistiqués :

- « SIMPLE_OPTIMIZATIONS », le mode par défaut, il n'interfère pas avec les autres scripts et librairies qui pourraient être présents sur la page ;
- « ADVANCED_OPTIMIZATIONS », qui prend des libertés avec les noms de fonctions et de variables afin de les optimiser ; mais attention, ce mode peut dramatiquement modifier votre code qui risque de ne plus être correctement interprété, notamment s'il fait appel à d'autres frameworks comme jQuery.

Si on ouvre « closure_compiler_test.html » dans un navigateur, on obtient une page comme ceci :

Il suffit ensuite de cliquer sur le bouton 'Optimize' et on obtient le code optimisé correspondant que l'on peut copier-coller :

```
function hello(name){alert("Hello, "+name)}hello("New user");
```

Closure Templates : Génération simplifiée de code HTML

Closure Templates est un moteur de 'templating', génération dynamique de pages Web, qui simplifie le codage de pages HTML en séparant l'apparence et le contenu. Closure Templates est mis en œuvre dans Gmail et Google Docs, et son fer de lance est que les templates sont indépendants de la technologie sous-jacente.

En effet, les templates ont une syntaxe spécifique (et une extension .soy), et peuvent être implémentés tant du côté client avec du JavaScript précompilé, que du côté serveur avec Java. La syntaxe est plutôt triviale, et on y retrouve des éléments classiques comme, entre autres, des boucles, des contrôles et des opérateurs.

Aujourd'hui, on peut difficilement imaginer que Closure Library puisse concurrencer des librairies performantes et éprouvées comme jQuery. Closure Templates aussi risque d'avoir du mal à s'affirmer face aux technologies existantes. En revanche, Closure Compiler, grâce au plugin FireBug, fait déjà partie des outils indispensables du développeur JavaScript. Il va donc falloir attendre d'avoir un peu plus de recul afin que les développeurs puissent faire le bon choix technologique.

■ Martin **Oppetit**
Ingénieur de développement chez Aeon Consulting
martin.oppetit@aeon-consulting.fr

Hudson, le serveur d'intégration adopté par les grands groupes industriels

Hudson est un serveur d'intégration continue, open source. Il est hébergé sur java.net et bénéficie d'une des plus fortes activités open source. De grands acteurs industriels et de l'open source comme eBay, Yahoo, SwingLabs, GlassFish, JBoss, Lucene, MySQL, Apache ou Eclipse l'utilisent.

Schéma du dashboard Hudson
http://hudson.zones.apache.org/hudson/

Que fait Hudson?

Concevoir et développer un composant logiciel est une tâche à part entière mais vérifier son intégration avec d'autres composants dans un environnement proche de l'environnement final est une autre tâche beaucoup plus consommatrice. Des processus d'intégration continue permettent d'automatiser les points intégrations manuels, de faire jouer automatiquement les tests de non régression évitant ainsi les phases de recherches d'anomalies longues et périlleuses. Ces processus d'intégration sont orchestrés dans une chaîne de build qui sera exécutée en permanence par l'outil d'intégration continue. Et tout comme ses concurrents, Hudson est capable de déclencher une chaîne de build de manière répétée.

La chaîne de build d'un projet est constituée de scripts automatisables, et composée de différentes étapes allant de la génération de code au déploiement, en passant par la compilation des sources du projet et l'exécution des tests de non régression. Un utilisateur peut déclencher manuellement cette chaîne de build. Néanmoins, dans un scénario classique, Hudson est en écoute des changements du gestionnaire de configuration comme Subversion et déclenche la chaîne de build à chaque changement de code source des développeurs. Les tests et les différents rapports de métriques, dont la couverture de code, la documentation technique (ex:Javadoc), sont reportés par Hudson et accessibles aux membres de l'équipe du projet. Hudson va donc automatiser les étapes manuelles de la construction d'une application afin de permettre aux développeurs d'une équipe projet de passer plus de temps à créer et développer de nouvelles fonctionnalités. Il faut considérer Hudson comme un autre membre de l'équipe, extrêmement méticuleux et organisé et qui n'est jamais malade, ce qui contribue à l'augmentation de la productivité.

La naissance de Hudson

Hudson a été créé par Kohsuke Kawaguchi, ingénieur chez Sun au Japon. Son objectif était de fournir un serveur d'intégration continue multi plate-forme, dont le mot d'ordre serait la simplicité en termes de déploiement, d'utilisation et d'extension, comparé aux outils qui étaient sur le marché au moment de sa création, comme CruiseControl ou Continuum. Hudson est donc né de l'expertise acquise sur les serveurs d'intégration en reprenant les meilleures fonctionnalités des produits existants sans les inconvénients. Hudson est distribué sous licence MIT et Creative Commons, ce qui lui confère sa liberté d'utilisation et d'exploitation.

Les critères de choix d'un serveur d'intégration continue

Le choix d'un outil par rapport à un autre est défini avant tout par l'usage que l'on en fait. Il doit répondre à un large éventail de besoins utilisateurs, comme posséder une interface de visualisation permettant de contrôler le statut des projets, d'administrer et d'accéder aux rapports statiques. Il doit supporter plusieurs outils SCM, pouvoir construire les projets de différents langages et fournir un ensemble d'outils de notification afin d'avertir l'équipe projet dans les plus brefs délais du succès ou de l'échec du résultat de l'exécution de la chaîne de build.

Hudson est doté d'une interface Web graphique fournissant des indicateurs de différents types avec un mécanisme de couleurs pour le résultat de build (rouge en cas d'erreur comme dans le cas d'une erreur de compilation, jaune en cas de build instable comme dans le cas où il y a des tests en échec et bleu dans le cas du succès d'un build). Notons que dans la culture japonaise, la couleur bleue est synonyme d'espoir; il vous est possible de la changer par l'ajout d'une extension à Hudson. De plus, Hudson propose la notion de santé des projets ou météo du projet. Cette santé peut être calculée de différentes manières en fonction des extensions installées et activées sur le projet. Par défaut, la santé du projet correspond au pourcentage des builds récents ayant réussi: plus ce pourcentage est élevé, meilleure est la santé du projet. Cette notion de santé permet d'identifier les projets qui sont souvent en échec ou qui ne satisfont pas un ensemble de règles, par exemple une couverture de code supérieure à 80%; alors que le dernier build n'est pas en échec car le projet compile et que l'ensemble des tests joués sur le projet sont satisfaits.

Les avantages de Hudson

Hudson se distingue tout d'abord de ses concurrents par la très grande simplicité d'installation et d'utilisation.

Installation

Hudson est livré sous forme d'une archive Web. Il vous suffit de la déployer dans le container Web de votre choix comme le serveur Tomcat ou SpringSource TC Serveur. Ce dernier est très pratique puisqu'il intègre un serveur Tomcat corrigeant de nombreux problèmes (plus particulièrement sur le classloader) et fournit des scripts d'administration. Mais l'ajout d'un container de servlet est optionnel. En effet, l'archive Hudson elle-même, inclut le très léger serveur Web Winstome qui permet de disposer d'une instance Hudson sans aucun autre outil supplémentaire. C'est très pratique dans le cadre de la réalisation d'une démonstration de Hudson.

Configuration

Hudson se distingue par la fourniture d'une interface graphique très riche et très intuitive pour configurer les projets qu'il exécutera et les outils (comme Ant, Maven ou autres), dont la chaîne de build d'un projet dépend. Tous les éléments de configuration de l'interface graphique sont constitués par un ensemble de formulaires. L'aisance de la configuration graphique est fournie entre autres, par une zone d'aide pour chaque entrée des formulaires Cette zone est dans certaines situations traduite en 8 langues. La convivialité en termes de configuration permet d'éviter de livrer l'intégrateur à lui même, ce qui n'est souvent pas le cas pour les autres outils du marché. Par ailleurs, la configuration Hudson est convertie en fichier XML simple après la création des projets, ce qui permet une configuration manuelle si besoin. En effet, il est possible de changer un élément de configuration depuis l'interface graphique, ou directement depuis son fichier XML correspondant, pour que ce changement soit effectif. [Fig.1]

Fig.1

De plus, Hudson expose l'ensemble des ses opérations de configuration et d'utilisation par API. Dans un style REST, il est ainsi très facile de pouvoir réaliser des opérations de type CRUD (création, mise à jour, suppression de projets), déclencher un build ou même récupérer le résultat des différents builds.
Exemple de création d'un job Hudson par API via une implémentation Java avec HttpClient :

```
private void put(HttpClient client, String hudsonBaseURL,String
 jobName, File configFile) throws IOException, HttpException {
   PostMethod postMethod = new PostMethod(hudsonBaseURL+ "/create
Item?name=" + jobName);
   postMethod.setRequestHeader("Content-type","application/xml;
 charset=ISO-8859-1");
      postMethod.setRequestBody(new FileInputStream(configFile));
      postMethod.setDoAuthentication(true);
     try {
            int status = client.executeMethod(postMethod);
            System.out.println(status + "\n"+ postMethod.get
ResponseBodyAsString());
         } finally {
              postMethod.releaseConnection();
   }
}
```

Exemple de récupération du statut de l'ensemble des jobs d'un dashboard Hudson :

```
private void read(String hudsonBaseURL) throws IOException, Http
Exception,DocumentException {
 URL url = new URL(hudsonBaseURL + "/api/xml");
 Document dom = new SAXReader().read(url);
 for (Element job : (List<Element>) dom.getRootElement().elements
("job")) {
           System.out.println(String.format("Name:%s\tStatus:
%s", job.elementText("name"), job.elementText("color")));
     }
}
```

En outre, toujours au sein de l'archive Web, un module Hudson CLI est livré. Ce dernier permet de piloter l'outil Hudson entièrement en ligne de commande. Ces possibilités d'administration sont indispensables au sein de grands groupes industriels où les instances Hudson sont souvent pilotées par des équipes d'administration dédiées. En effet, tous ces services permettent de réduire le temps de prise en main de l'outil pour les intégrateurs et les administrateurs.
Exemple de la désactivation d'un job sur une instance Hudson existante par Hudson CLI :

```
java -jar hudson-cli.jar -s http://hudsonserver:8090 disable-job
 my-job
```

Exemple de récupération de la date du dernier build d'un projet Hudson :

```
curl http://code.taglab.com/hudson/job/Gradle/lastBuild/build
Timestamp?format=dd/MM/yyyy
```

Les autres fonctionnalités

Hudson se distingue aussi sur le marché des serveurs d'intégration continue par l'ajout de fonctionnalités supplémentaires comme les dépendances de builds entre les projets, les builds distribués, les matrices de configuration, son extrême extensibilité et même son ouverture sur le cloud computing.

Dépendances de builds

Hudson peut mettre en place un mécanisme de dépendances entre les projets de build avec la notion de projet en aval et projets en amont. Un scénario classique permet par exemple d'exécuter un build léger suivi d'un build d'intégration. Le premier est chargé uniquement de lancer les étapes de compilation et les tests unitaires. En revanche, le second build est chargé de l'exécution des tests d'intégration (plus longs en temps d'exécution) et de l'exécution d'éventuels métriques. Hudson permet d'invoquer ce second build automatiquement à la fin du premier si ce dernier est en succès. Hudson peut agréger les résultats, par exemple le résultat des tests

à la fin de la chaîne de dépendances. Ce découpage permet d'avoir un retour utilisateur plus rapide, très utile dans le cas de systèmes complexes où une chaîne de build complète prend plusieurs heures. Un autre scénario est d'avoir la possibilité de lancer après la construction d'une librairie transverse, la construction simultanée de plusieurs composants indépendants les uns des autres, mais où chaque composant dépend de cette librairie transverse. Cette parallélisation des constructions des composants permet de réduire le temps de construction global d'un projet et permet ainsi d'informer du succès ou de l'échec de la chaîne de build plus rapidement.

Builds distribués

Nombreux sont les logiciels ayant besoin d'être exécutés sur plusieurs plates-formes. Votre outil d'intégration doit donc vous fournir la capacité de tester sur plusieurs environnements. Hudson permet de pouvoir déléguer à d'autres serveurs que le serveur hébergeant Hudson, le lancement de la chaîne de build de certains projets. C'est la notion de serveur maître et de serveurs esclaves. Par exemple, un scénario classique est d'avoir une instance Hudson installée sur un serveur Windows exécutant la chaine de build d'un projet sur une distribution Linux. De ce fait, votre build s'exécute dans un environnement complètement isolé de la plate-forme Hudson. [Fig.2]

Ce service est également souvent très utilisé lorsque la chaîne de build met en œuvre certains outils installés sur des machines spécifiques et qui ne sont pas présents sur le serveur où est installé Hudson. C'est utilisé aussi dans le cas où l'outil requis par la chaîne de build est présent sur le serveur Hudson mais pas dans la bonne version. Un cas d'exemple est de disposer d'une machine de build où serait installé l'outil propriétaire Visual Studio pour disposer des routines de build des projets .NET. Cet outil est installé sur uniquement une machine dédiée avec une licence serveur. Dans ce contexte, l'instance Hudson installée et configurée sur une autre machine va déléguer tous les build .NET au serveur en question.

Matrices de configuration

Une autre fonctionnalité très intéressante de Hudson est la possibilité de mettre en place des matrices de configuration.

Elle est particulièrement adaptée aux projets qui nécessitent un grand nombre de configurations différentes, comme des environnements de tests multiples, des binaires spécifiques à une plate-forme, etc. Par exemple, vous pouvez builder un même projet avec différentes versions du JDK, différents types de base de données; et agréger les résultats. Grâce aux matrices de configuration, la mise en place d'un tel environnement d'intégration continue est simplifiée grandement. [Fig.3 et 4] Couplées à la possibilité de distribuer les builds, les matrices de configuration permettent d'augmenter le nombre de projets pouvant se construire simultanément et donc de réduire le temps global d'exécution des chaînes de builds.

L'extensibilité

Hudson peut être étendu afin de lui permettre de supporter certains outils ou processus que vos équipes utilisent. Cette extensibilité est permise car son architecture expose un ensemble de points d'extensions. Ces derniers permettant de développer des plugins contribuant aux différentes parties de Hudson comme le système de gestion de configuration, les étapes de build, les outils de publication des résultats. Hudson peut ainsi évoluer au-delà d'un simple outil de gestion des constructions en proposant un cycle de développement très avancé comme la promotion de tags construits après le build, le suivi des modifications dans les dépendances de produit, la fourniture des graphiques des résultats de test au cours du temps, le suivi de la couverture de code, et bien d'autres possibilités.

Mais on peut également développer des plugins afin d'étendre les capacités de configuration (cas du support d'un nouvel outil) et d'administration, en contribuant au module Hudson CLI qui pilote Hudson en ligne de commande.

Par exemple, un plugin de configuration logicielle peut rajouter des commandes Hudson CLI pilotant la création de branches; de tags ou de changement de statuts.

Exemple de la promotion d'une baseline composite par le plugin clearcase-release du build 17 pour le job 'job-test'

```
java -jar hudson-cli.jar -s http://server/hudson release-composite
 job-test 17
```

Exemple de la promotion des latest baselines par le plugin clearcase-release du job 'job-test'

```
java -jar hudson-cli.jar -s http://server/hudson release-latest
-baselines job-test
```

Conclusion

A l'instar de ses principaux concurrents que sont CruiseControl, Bamboo ou TeamCity, Hudson se distingue par sa simplicité d'installation, de configuration, d'utilisation, ses fortes possibilités d'administration et son vaste écosystème de plugins qui lui confère un avantage certain. Tous ces avantages font de Hudson, aujourd'hui, le serveur d'intégration continue adopté par la majorité des acteurs de l'industrie pour tous types de projets et de n'importe quelle taille.

■ Grégory **Boissineau**
Expert intégration continue (Zenika) et committeur Hudson

Fig.2

Fig.3

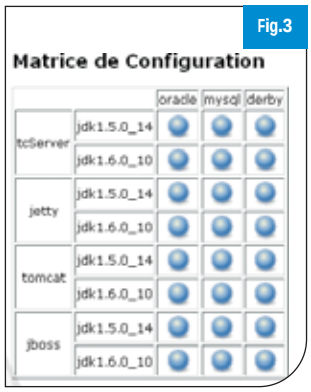
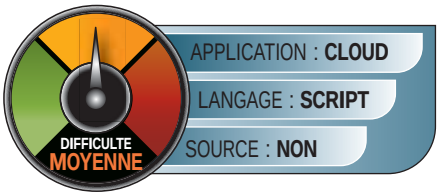


Fig.4

Cloud Foundry : aux fondations du cloud computing

Au cours du mois d'août dernier, après s'être fait racheter par WMware, SpringSource annonce le rachat de Cloud Foundry. La voie est ainsi ouverte pour mettre ses applications *« dans les nuages »* en s'appuyant complètement sur la pile technologique proposée par SpringSource. Quels en sont les enjeux ? Cloud Foundry est actuellement en bêta. Dans cet article, nous découvrirons sur quoi se bâtit l'offre, comment se créer un compte, comment déployer son application, de son premier lancement jusqu'à sa maintenance et enfin nous nous intéresserons aux possibilités qu'offre Cloud Tools.



Cloud Foundry est avant tout conçu pour déployer des applications Spring et Grails. Sa particularité est de ne posséder aucune infrastructure : le service s'appuie sur Amazon EC2. À terme, le déploiement sera possible sur l'environnement virtualisé de VMware : vSphere. Nous avons ainsi une certaine abstraction : que le cloud soit public ou privé l'expérience développeur est voulue pour être la même. La simplification du cloud est la ligne directrice de Cloud Foundry : pouvoir déployer et maintenir en quelques clics seulement une application grâce à une interface d'administration riche.

Fidèle au principe d'open source de SpringSource, l'offre s'appuie sur une infrastructure technique composée de Apache, Tomcat et MySQL. Ces serveurs peuvent être sur une seule instance ou répartis sur plusieurs, de la manière suivante : [Fig.1].

Cloud Foundry déploie toute son infrastructure sur les Amazon Machine Image (AMI) et en garde l'administration. Cette solution est conçue comme *« Infrastructure as a Service »* (IaaS) : l'infrastructure complète du système d'exploitation jusqu'aux serveurs est fournie en tant que service. Contrairement aux offres PaaS comme Google App Engine, les contraintes d'API sont ainsi énormément réduites, voire nulles. Le développement se fait ainsi comme un développement d'application web classique puisque l'unité de livraison à fournir est un war et que la base de données est relationnelle.

Ce qui différencie cette offre d'un simple hébergement web est la caractéristique principale et commune des différentes formes de cloud computing : la gestion élastique en temps réel des serveurs permettant de résister face à la montée en charge. C'est ici qu'intervient une autre brique applicative de SpringSource : tc Server. Ce dernier se définit comme une version entreprise de Tomcat qui fournit aux développeurs un serveur léger qui sera couplé à la gestion opérationnelle des diagnostics de pointe, et le support de charge critique. Cette supervision applicative se base sur Hyperic HQ, produit que SpringSource avait racheté en mai dernier.

Au déploiement, vous choisissez la topologie à adopter et par la suite, le nombre d'instances de serveur se redimensionnera automatiquement. Vous pouvez éventuellement spécifier une fourchette afin de mieux maîtriser leur nombre. Il reste tout de même possible de modifier manuellement le nombre d'instances via la console d'administration. Par contre, si vous choisissez un déploiement mono-instance, il ne sera pas possible de modifier la topologie. Seul le nombre de tc Server se redimensionne, les instances de MySQL esclaves échappent à ce contrôle. Cependant, tc Serveur n'est pas le seul serveur d'application supporté : il est aussi possible d'utiliser un simple Tomcat, Exo ECM ou encore Liferay.

Quant à la base MySQL, celle-ci peut utiliser la technologie EBS de Amazon, un service de stockage persistant sur disque dur réseau haute capacité. Les données sont ainsi protégées en cas de sinistres. Sans ce système, si vous stoppez vos instances ou qu'elles se crashent, vous perdez vos données.

Contrairement à Google App Engine, Cloud Foundry n'est pas un service complètement gratuit : les frais sont ceux d'Amazon EC2. S'agissant d'instances virtualisées de serveur, celles-ci sont facturées non pas à la charge d'utilisation du CPU mais en fonction de la durée de vie des instances [Fig.2]. Il faut en plus compter des coûts supplémentaires pour le stockage, bande passante et les transac-

Fig.1

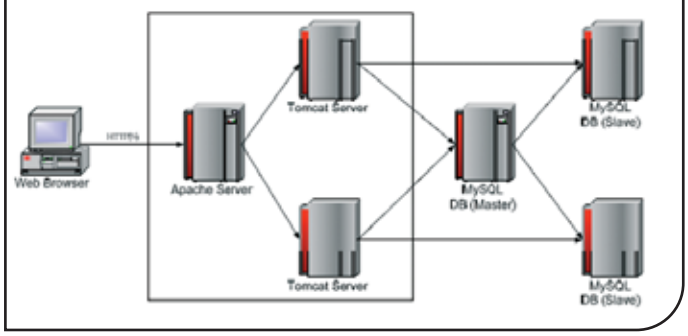


Fig.2

	Virtual Cores	Compute Units /core*	32/ 64 Bit	Memory	Storage	$/hr
Small	1	1	32 bit	1.7G	160G	0.10
High-CPU Medium	2	2.5	32 bit	1.7G	350G	0.20
Large	2	2	64 bit	7.5G	850G	0.40
Extra Large	4	2	64 bit	15G	1690G	0.80
High-CPU XL	8	2.5	64 bit	7G	1690G	0.80

tions. À noter que la facturation étant effectuée aux USA, vous aurez une taxe à payer, en l'occurrence celle du Texas au cours des tests que nous avions effectués pour l'écriture de cet article.

CRÉATION DE COMPTE

Étant donné que Cloud Foundry se base sur les services Amazon, vous devez au préalable disposer d'un compte Amazon EC2 (http://aws.amazon.com/ec2). Vous aurez besoin de :

- Votre « Access Key ID » et votre « Secret Access Key »
- Le ou les « Pair key »de région dont vous disposez.

N'oubliez pas non plus d'ouvrir les ports SSH et HTTP. Et c'est tout ce que vous aurez à faire sur votre compte Amazon, Cloud Foundry se chargera du reste.

Connectez vous sur http://www.cloudfoundry.com et cliquez sur « SIGN UP ». Vous n'avez qu'à mentionner votre mail et vous recevrez une invitation dans laquelle vous aurez un lien d'activation de votre compte. Vous aurez à indiquer quelques informations nécessaires à la création de votre compte. L'écran suivant permet de configurer la connexion à votre compte Amazon EC2. Si vous ne souhaitez pas les mentionner tout de suite, il sera possible de le faire plus tard via le menu « SETTINGS » [Fig.3].

Et c'est tout…

L'interface se décompose en 5 menus :

- « HOME » : Synthèse de différentes informations telles que les applications tournant actuellement, les derniers événements, la météo des services ou les dernières news.
- « APPLICATIONS » : Liste des applications qui ont été envoyées.
- « DEPLOYMENTS » : Liste de déploiements qu'ils soient en cours ou qu'ils aient été arrêtés.
- « SETTINGS » : Options de configurations telles que votre mot de passe et vos identifiants Amazon.
- « HELP » : FAQ et liens vers le forum.

Voilà, votre compte est prêt pour pouvoir lancer vos applications dans les nuages !

DÉPLOIEMENT D'UNE APPLICATION

Le premier déploiement d'une application se fait en deux étapes : *création* d'une application déployable et son *déploiement*. Il est d'abord important de comprendre la notion d'application au sein de Cloud Foundry. Celle-ci est composée d'au moins un war et d'une base de données (même si elle n'est pas utilisée). Ensuite viennent se rajouter de façon facultative : script(s) SQL, d'autres war, script d'installation, jar partagé, ... Pour notre test, nous utiliserons une application très simple qui affiche un message contenu dans une table. Nous déploierons cette application sur des instances multiples. Étant donné que la base de données sera sur une instance indépendante, il est donc impossible de prévoir l'IP qui lui sera attribuée. Nous devons configurer l'URL de connexion de la sorte :

```
jdbc:mysql://${dbHostName}:3306/maBase
```

Ce paramètre sera ensuite fixé comme option de la JVM.

Dans le menu « APPLICATIONS », cliquez sur « Upload Application ». Dans la zone « APPLICATION NAME », renseignez le nom de l'application tel qu'il apparaîtra dans la liste des applications. Dans le panel, « WEB APPLICATION », nous allons sélectionner notre war et lui donner un nom de contexte. Afin que l'application puisse connaître l'hôte de la base de données, c'est dans « JVM OPTIONS » que nous fixerons le paramètre suivant :

```
-DdbHostName=${databasePrivateDnsName}
```

Le panel « DATABASE » nous permet de configurer l'accès à la base de données et de fournir un script d'initialisation de la base. Voilà, nous avons renseigné les informations nécessaires. Il suffit de cliquer sur « Upload » pour envoyer votre application sur Cloud Foundry. Une fois l'envoi terminé, le menu « DEPLOYMENTS » s'affichera avec notre application dans sa liste.

Il suffit de cliquer sur le triangle vert pour accéder au paramétrage du déploiement. Nous donnons un nom au déploiement, sélectionnons « Multiple instances ». À remarquer, qu'un indicateur affiche en haut à droite, le coût horaire et mensuel de l'infrastructure. Nous laisserons les autres paramètres par défaut et il ne reste plus qu'à cliquer sur « Launch » pour que le déploiement commence. Cette action prend un temps variable en fonction du nombre d'instances choisies (de 5 à 8 min).

Le déploiement terminé, celui-ci devrait apparaître dans le menu « DEPLOYMENTS ». Un clic sur la ligne nous donne accès aux détails du déploiement qui permettent d'administrer l'application. Un clic sur « Go to the Home Page » nous permet de nous rendre sur la page de notre application.

Et nous voyons … une page blanche car nous n'avons pas fourni de script de données. Nous allons donc grâce à l'interface d'administration mettre à jour la base avec un simple script SQL qui fera un INSERT. Pour cela, il nous suffit de cliquer sur l'action « Execute SQL » dans la partie administration de MySQL. Une fenêtre de dialogue nous demande le script SQL. Celui-ci sera exécuté directement. Si nous retournons à la page de notre application et que nous rafraîchissons, nous voyons bien notre message apparaître [Fig.4].

Au cours du cycle de vie de votre application, vous serez sûrement amenés à devoir la mettre à jour par un nouveau war. Pour cela, dans les détails de l'application, il suffit de modifier le war de l'application en supprimant l'ancien et en envoyant le nouveau. Il ne reste

Fig.3

Fig.4

plus qu'à redéployer en cliquant sur « Redeploy » dans les détails de déploiement.
A l'heure actuelle, l'interface d'administration ne fournit que des informations de base sur les serveurs : consultation de logs, reboot / stop / redeploy, exécution SQL, modification de topologie, charge CPU, …. Nous aurions aimé pouvoir consulter des éléments comme le trafic réseau ou un graphe historique de l'activité mais malheureusement ce n'est pas encore possible. Il est néanmoins prévu de les intégrer. Si vous avez besoin de ces informations, vous devrez vous rabattre sur CloudWatch, service payant de votre compte Amazon EC2.
Attention, n'oubliez pas que vous disposez d'instances personnalisées et donc que si le déploiement de votre application échoue, vous êtes tout de même facturé. Il en sera de même si vous stoppez une application dont vous ne vous servez plus.

ALLER PLUS LOIN AVEC CLOUD TOOLS

Fondateur de Cloud Foundry, Chris Richardson était déjà l'instigateur de Cloud Tools, projet open source développé en Groovy dont le but est de faciliter le déploiement sur Amazon EC2.
C'est justement sur cette plate-forme que s'appuie Cloud Foundry. Il est possible d'utiliser Cloud Tools en tant que plug-in Groovy ou Maven. Intéressons nous à son utilisation avec Maven pour que le cloud fasse partie du cycle de vie de votre application.
Ce plug-in n'étant pas dans les dépôts officiels, il faut en ajouter à votre fichier POM :

```
<repositories>
   <repository>
      <id>pia-repository</id>
      <url>http://www.pojosinaction.com/repository</url>
   </repository>
</repositories>

<pluginRepositories>
   <pluginRepository>
      <id>pia-repository</id>
      <url>http://www.pojosinaction.com/repository</url>
      <snapshots>
      <enabled>true</enabled>
      </snapshots>
   </pluginRepository>
</pluginRepositories>
```

L'intégration du plug-in dans votre build se fait par exemple de la sorte :

```
<build>
    <plugins>
        <plugin>
            <groupId>net.chrisrichardson</groupId>
            <artifactId>cloudtools-maven-plugin</artifactId>
            <configuration>
                <awsPropertiesFile>./aws.properties</awsPro
pertiesFile>
                <schemaName>maBase</schemaName>
                <schemaUsers>
                    <param>user:pass</param>
                </schemaUsers>
                <sqlScripts>
                    <param>./database/dll.sql</param>
                </sqlScripts>
                <catalinaOptsBuilder>
                    {builder, databasePrivateDnsName, slaves ->
                    builder.arg("-server")
                    builder.arg("-Xmx1000m")
                    builder.prop("com.sun.management.jmxremote")
                    builder.prop("com.sun.management.jmxremote
.port", 8091)
                    builder.prop("com.sun.management.jmxremote
.authenticate", false)
                    builder.prop("com.sun.management.jmxremote
.ssl", false)
                    builder.prop("ptrack.application.environment",
 "ec2")
                    builder.prop("jdbc.db.server", database
PrivateDnsName)}
            </catalinaOptsBuilder>
            </configuration>
        </plugin>
    </plugins>
</build>
et le fichier aws.properties :
imageId.m1.small=ami-6f2cc906
accessKey=...
secretKey=...
keyName=maCle
keyPairFile=./cle_aws_europe.pem
sshDir=/usr/bin
```

CONCLUSION

Cloud Foundry s'adresse principalement au business à court terme comme des campagnes marketing ou publicitaires. Cette stratégie se justifie par le fait que la durée de vie sera suffisamment courte pour qu'un retour sur investissements en achetant et mettant en place de nouveaux serveurs n'ait pas le temps de se produire. D'autre part, l'environnement étant classique, le développement pourra être confié aux équipes habituelles puisqu'il ne nécessite aucune connaissance particulière. Un autre cas d'utilisation proposé par Cloud Foundry est la charge temporaire : je sais qu'un mois par an, j'aurai une forte charge, j'héberge mon site pendant ce temps chez Cloud Foundry et le reste de l'année chez moi.

L'acquisition de Cloud Foundry par SpringSource est encore récente et la roadmap n'est pas encore exactement fixée. La bêta est prévue sur une durée de 6 à 8 mois et devrait donc se terminer vers mars-avril.
D'ici là, un certain nombre de nouvelles fonctionnalités, telles que le déploiement en cloud privé sur vSphere, devrait voir le jour.

Cloud Foundry : http://www.cloudfoundry.com
Cloud Tools : http://www.cloudtools.org
Amazon EC2 : http://aws.amazon.com/ec2

■ Nicolas **François**
Ingénieur Développement chez Sfeir

PowerShell 2.0 : Windows au bout du Shell !

Les sorties de Windows 7, Windows Server 2008 R2 et Exchange Server 2010 nous donnent l'occasion de revenir sur les bases de Windows PowerShell, technologie de scripting, à mi-chemin entre IT et développement.

Trois années se sont écoulées depuis la sortie officielle de Windows PowerShell 1.0. Depuis, le potentiel de cet outil n'a cessé d'évoluer. Entre les fonctionnalités de base à destination des administrateurs systèmes, les outils et produits tirant parti de cette technologie et les nouveautés de la version 2.0, il est aujourd'hui difficile de passer à côté de Windows PowerShell. Il n'est plus nécessaire de démontrer les enjeux et les avantages d'user et d'abuser des scripts en entreprise, les atouts de l'automatisation et de la mise à l'échelle ont fait leurs preuves depuis longtemps. Windows PowerShell est une technologie regroupant deux concepts qu'on avait l'habitude de voir séparés dans les environnements Windows : un invite de commande puissant et un environnement de scripts intégré. Intéressons-nous dans un premier temps à la notion de commande.

PRENEZ LES COMMANDES !

Dans un premier temps, la plupart des opérations que vous allez réaliser le seront grâce aux commandes de base de PowerShell, appelées *CmdLets* (prononcer « commandelettes »). Leur nombre varie en fonction de la version de Windows PowerShell, de la version du système d'exploitation et des différents logiciels installés. J'en ai par exemple 236 sur le Windows 7 que j'utilise actuellement pour la rédaction de cet article.

Ces commandes ont des champs d'applications très divers, allant de la récupération d'informations systèmes (Get-Process, Get-Host, Get-Service, Get-Eventlog, ...) à la manipulation de fichiers (Get-Content, New-Item, Set-Location, ...) en passant par le formatage et l'affichage d'informations (Out-File, ConvertTo-Html, Format-Table, ...). L'un des avantages des CmdLets est la syntaxe Verbe-Nom homogène, vous permettant d'avoir rapidement une idée sur l'action effectuée et sur quoi celle-ci est exécutée. La première CmdLet que je vous invite à tester est celle vous fournissant la liste des CmdLets disponibles sur votre système :

```
Get-Command -CommandType CmdLet
```

En plus du nom de la CmdLet, nous avons utilisé le paramètre CommandType afin de lui fournir l'argument « CmdLet ». Cela indique que nous souhaitons obtenir seulement les commandes de type CmdLet. Il existe en effet d'autres types de commandes, comme les alias (permettant d'appeler une CmdLet via un nom raccourci), les *fonctions* (regroupant plusieurs instructions), les *scripts* ou encore les *applications Windows* disponibles sur la session en cours (.exe par exemple). La syntaxe utilisée pour les paramètres est également cohérente entre les différentes CmdLets. Le tiret « - » précède le nom du paramètre et l'argument (lorsqu'un argument est nécessaire) est renseigné à la suite.

Pour connaître les différents paramètres pouvant être utilisés avec telle ou telle CmdLet ou pour obtenir plus de détails sur les actions exécutées, vous pouvez utiliser la commande *Get-Help*. Je vous conseille de l'utiliser avec le paramètre *-full*, sans argument, afin d'obtenir le plus de détails possible. Par exemple, nous pouvons obtenir l'aide détaillée de la CmdLet Get-Command en exécutant l'instruction suivante :

```
Get-Help -Name Get-Command -full
```

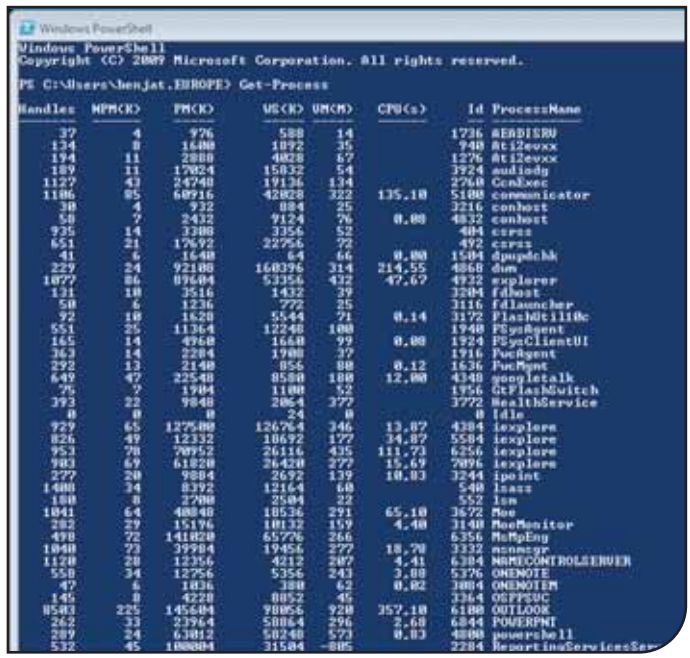
Dans certain cas, il est important de noter que le nom du paramètre peut être omis, et que celui-ci peut être implicite grâce aux différents arguments que vous fournissez. Ainsi, la dernière instruction peut être écrite de cette façon :

```
Get-Help Get-Command -full
```

L'argument Get-Command est en effet automatiquement reconnu comme argument du paramètre Name. Une autre façon de simplifier une instruction est d'utiliser les alias de CmdLets, afin de réduire leurs appels à quelques caractères. Ainsi, Get-Command peut également s'écrire gcm, Get-Process peut s'écrire gps ou ps, ls est l'équivalent de *Get-Childitem*, ...

La commande suivante vous permettra d'obtenir la liste des sujets d'aide disponibles, CmdLets, alias et sujets divers compris :

Get-Help *



C'EST DANS LE PIPE !

L'utilisation du pipe « | », la barre verticale que vous trouverez sur votre clavier grâce aux touches Alt Gr + 6 est bien connu dans le monde Unix. Elle permet de passer le résultat d'une instruction comme argument d'une autre instruction. Cette notion de pipeline prend tout son sens dans Windows PowerShell grâce à une intégration très poussée. Vous pourrez ainsi passer vos résultats de CmdLets en CmdLets, sans vous soucier des conversions complexes mises en œuvre pour vous. Nous verrons dans la prochaine partie qu'elle va encore plus loin grâce aux concepts orientés objets.

La première illustration que nous pouvons faire est l'utilisation d'une CmdLet de sortie, que nous allons utiliser pour écrire dans un fichier le résultat de la commande Get-Process, nous permettant de lister les processus de la machine :

```
Get-Process | Out-File c:\temp\process.txt
```

Une fois cette commande exécutée, vous pourrez vérifier que le fichier *process.txt* contient bien la liste des processus. Cette liste a en effet été passée du résultat de la commande Get-Process à l'entrée de la *CmdLet Out-File.*

Dans le cas où votre deuxième commande vous renvoie un nouveau résultat (ce qui n'est pas le cas d'*Out-File*), vous pouvez une nouvelle fois utiliser un pipe pour le passer à une troisième commande, et ainsi de suite. Voici un exemple un peu plus poussé utilisant le mécanisme de pipeline sur plusieurs CmdLets :

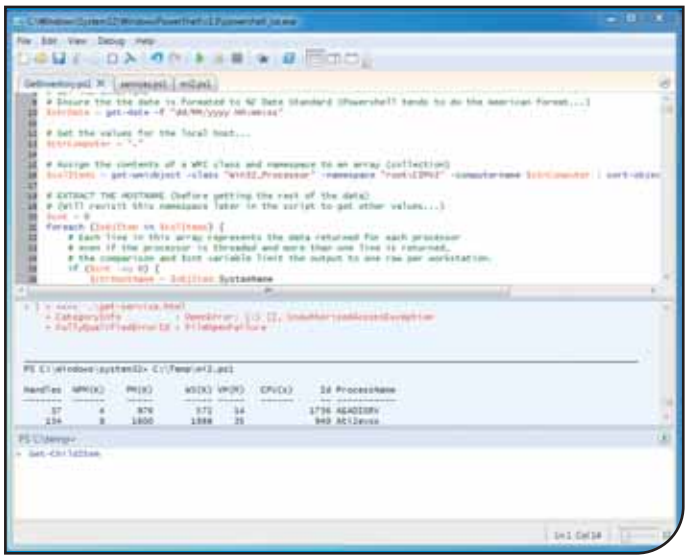
```
Get-Process | Sort-Object -Name CPU | Select-Object -first 10
| Out-File c:\temp\process.txt
```

Même sans connaître toutes ces CmdLets, vous devinez certainement ce que fait cette instruction... non ?

TOUT EST OBJET !

Depuis quelques années, une grande tendance de l'informatique est à l'utilisation de concepts orientés objets. C'est notamment vrai du côté du développement mais le devient de plus en plus du côté infrastructure et administration.

En PowerShell, tout est objet, ou plus précisément, tout est représenté sous forme d'objet. Ceux-ci vous permettent de manipuler plus simplement les informations avec lesquelles vous souhaitez travailler. Une chaîne de caractères est un objet, une date est un objet,



les processus ou les services Windows sont eux aussi représentés sous cette forme, etc.

Si vous n'êtes pas encore familiers à la notion d'objet, considérez l'analogie suivante :

- Voiture est un type d'objet,
- une voiture parmi d'autres est un objet,
- cette voiture a des propriétés, comme la couleur, la cylindré, le nombre de portes,
- enfin, cette voiture peut effectuer des actions comme « accélérer », « freiner » ou même « tomber en panne ». Ces actions sont des méthodes.

Pour résumer, un objet appartient à un certain type, possède des propriétés que vous pouvez lire ou écrire ainsi que des méthodes que vous pouvez exécuter. Les différentes CmdLets de PowerShell ne dérogent pas à la règle : elles manipulent des objets. Par exemple, en utilisant la CmdLet Get-Date, vous obtiendrez la représentation objet de la date en cours. Cette représentation n'est en effet pas une simple chaîne de caractères affichée à l'écran, mais une liste de propriétés (jour, mois, ...) et de méthodes (ajouter jour, comparer à une autre date, ...). Vous pouvez par exemple accéder à la valeur du mois en cours avec la syntaxe suivante :

```
(Get-Date).Month
```

Les parenthèses sont ici utilisées pour indiquer à Windows PowerShell que nous voulons travailler sur l'objet obtenu grâce à la CmdLet Get-Date et le point indique que nous souhaitons utiliser une propriété ou une méthode de cet objet (ici, la propriété Month).

Pour utiliser une méthode, la syntaxe est la suivante :

```
(Get-Date).AddDays(2)
```

Dans le cas de l'appel d'une méthode, les parenthèses après le nom de la méthode AddDays nous permettent d'indiquer les arguments nécessaires à l'exécution, séparés par des virgules. Dans cet exemple, nous ajoutons 2 jours à la date courante. Pour connaître les propriétés et les méthodes d'un objet, vous pouvez le passer (via un pipe « | ») à la *CmdLet Get-Member* de la façon suivante :

```
Get-Date | Get-Member
```

Les objets peuvent donc être manipulés (via leurs propriétés et méthodes), transmis à des commandes (via le pipeline qui montre ici toute sa puissance) mais peuvent également être stockés. C'est le but des variables. Pour stocker la date courante dans la variable nommée *$dateCourante* (une variable commence toujours par le caractère $), vous pouvez utiliser la syntaxe suivante :

```
$dateCourante = Get-Date
```

Votre objet étant stocké dans la variable $dateCourante, vous pourrez utiliser cette dernière plus tard, comme argument d'une autre CmdLet par exemple. Vous pouvez par exemple afficher son contenu à l'écran en utilisant la syntaxe suivante :

```
Write-Host -Object $dateCourante
```

VOTRE PREMIER SCRIPT !

Nos premiers exemples sont utiles pour communiquer avec le système via l'invite de commande de Windows PowerShell. Mais cette technologie prendra tout son sens dès le moment où vous allez commencer à écrire des scripts afin d'automatiser des actions plus ou

moins complexes. Les scripts PowerShell sont de simples fichiers texte avec l'extension « .ps1 » (même pour les scripts utilisant des fonctionnalités de PowerShell 2.0). Ils peuvent être édités grâce au bloc-notes, à Windows PowerShell Integrated Scripting Environment (l'éditeur intégré à Windows PowerShell 2.0) ou avec tout autre éditeur de fichiers textes.
Dans ces fichiers, vous pouvez enchaîner les instructions ligne après ligne, qui peuvent être des appels de CmdLet, des assignations de variables, des instructions utilisant le pipeline, des commentaires (lignes précédées de #), des boucles, etc…
Je vous propose d'écrire dès maintenant votre premier script, en utilisant des exemples utilisés plus haut dans cet article :

```
#Récupération de la date courante
$dateCourante = Get-Date
#Récupération de la liste des processus
$p = Get-Process
#Ecriture de la date courante dans un fichier
$dateCourante | Out-File -FilePath c:\temp\process.txt
#Ecriture des 10 premiers processus consommateurs de CPU à la suite du même fichier
$p | Sort-Object -Property CPU -Descending | Select-Object -First 10 | Out-File -FilePath c:\temp\process.txt -Append
```

Une fois ce fichier sauvegardé, n'essayez pas de double-cliquer dessus, il ne se lancera pas. L'extension de fichier « .ps1 » est en effet associée au bloc-notes, pour éviter d'être exécutée par mégarde. Le deuxième mécanisme de sécurité vous empêchera tout simplement d'exécuter un script sur votre machine. C'est l'*ExecutionPolicy,* réglé par défaut sur « Restricted ». Vous pouvez vérifier ce réglage en exécutant la *CmdLet Get-ExecutionPolicy* via votre invite Windows PowerShell.
Différents réglages existent pour cet ExecutionPolicy. Vous pouvez autoriser tous les scripts (Unrestricted), simplement les scripts locaux et les scripts distants s'ils sont signés (RemoteSigned) ou seulement les scripts signés (AllSigned). Pour définir le réglage souhaité, vous devez être administrateur de la machine, et, en fonction de votre système d'exploitation, lancer PowerShell en mode administrateur via un « Exécuter en tant qu'administrateur » par exemple. Vous pourrez ensuite utiliser la CmdLet Set-ExecutionPolicy en lui passant la valeur souhaitée via le paramètre -ExecutionPolicy. Je vous conseille à des fins de test d'utiliser la valeur RemoteSigned vous permettant d'exécuter vos propres scripts tout en vous protégeant de ceux venant de l'extérieur :

```
Set-ExecutionPolicy RemoteSigned
```

Vous pouvez maintenant lancer votre script en passant son chemin à l'exécutable powershell.exe ou plus simplement en utilisant le point dans un invite de commande Windows PowerShell déjà lancé :

```
. chemin_complet_vers_votre_script.ps1
```

Le script s'exécutera et vous pourrez vérifier le résultat.

A CONDITION DE LA BOUCLER !

Les conditions et les boucles sont des concepts essentiels que vous devez connaître lors de la rédaction de vos premiers scripts. Ceux-ci sont très proches des concepts utilisés en programmation.
Selon le résultat d'une condition (vrai ou faux), vous pourrez exécuter des instructions différentes. C'est le mécanisme if…then…else bien connu. Voici un exemple de sa syntaxe :

```
if((Get-Date).Month -gt 9 -or (Get-Date).Month -lt 4) {Write-Host "Pensez à votre écharpe"} else {Write-Host "Tout va bien"}
```

Au-delà de la façon d'utiliser les opérateurs (-gt, -eq, -and, -match, …) que vous pourrez découvrir plus en détail via l'aide *Get-Help about_logical_operators et Get-Help about_comparison_operators*, il n'y a pas de difficulté particulières. Il est cependant bon de noter que vous n'êtes pas obligé de tout écrire sur une ligne lorsque vous développez un script. Les boucles sont très utiles pour effectuer des actions sur tous les objets d'un tableau ou d'une liste d'objets. C'est le cas quand vous récupérez la liste des services avec la *CmdLet Get-Service* par exemple. Il est en effet très facile d'utiliser le pipeline pour effectuer une action sur toute la liste, comme l'écrire dans un fichier par exemple. Mais pour effectuer une action sur chacun des fichiers individuellement, vous devez utiliser une boucle, grâce à la *CmdLet Foreach-Object* par exemple :

```
Get-Service | Foreach-Object {Write-Host $_.Name}
```

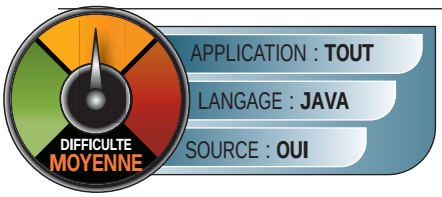
Ici, la spécificité se trouve dans la variable « $_ », qui contient, à chaque itération de la boucle, la valeur de l'objet en cours. C'est grâce à ce mécanisme que vous avez la possibilité d'effectuer des actions sur chacun des objets d'une collection. La CmdLet Foreach-Object n'est pas la seule à utiliser ce mécanisme, vous le retrouverez par exemple avec la commande de filtre *Where-Object*.
D'autres syntaxes existent pour effectuer des boucles for, while ou do…while. Vous les découvrirez encore une fois grâce à la commande *Get-Help *loop** par exemple.

■ Benjamin **Talmard**
Technical Account Manager ADC, Microsoft France
http://benjamin.talmard.com

Scala : le Java nouveau est arrivé !

1re partie

Dans l'univers en pleine expansion des langages tournant sur la plate-forme Java, il en est un qui fait de plus en plus de bruit depuis plusieurs mois. Créé en 2001, le langage Scala commence à atteindre une certaine maturité, ce que semblerait confirmer l'intérêt de plus en plus grand de la communauté Java à son égard. Découverte d'un langage prometteur aux approches innovantes que les plus audacieux n'hésitent pas à présenter comme le futur remplaçant du langage Java ! Rien que ça …



Depuis son apparition il y a de cela une quinzaine d'années, le langage Java n'a cessé d'évoluer et surtout de s'enrichir au point de devenir aujourd'hui incontournable dans le monde industriel. A tel point que désormais on ne désigne plus seulement un simple langage en parlant de Java mais plutôt une plate-forme d'exécution complète et performante. Toutes ces évolutions ont eu un coût cependant, puisque désormais, les spécifications du langage Java atteignent facilement les 600 pages, ce qui a pour effet premier de ralentir les avancées futures du langage comme on a pu le voir ces 2 dernières années avec Java 7. De ce fait, il commence à devenir évident que le langage Java n'est plus vraiment en adéquation avec les contraintes informatiques actuelles et qu'il faut chercher d'autres voies d'évolution.

Ce constat, certains l'avaient déjà fait bien avant tout le monde et ils ont ainsi commencé à préparer un successeur au langage Java s'appuyant sur sa plate-forme d'exécution, formidable outil qui n'a cessé d'être amélioré et optimisé depuis toutes ces années, mais apportant un nouveau langage à la pointe des avancées techniques. Parmi eux, Martin Odersky, un professeur universitaire bien connu de la communauté Java pour avoir travaillé sur *javac* le compilateur Java et pour avoir dirigé les travaux sur les Generics de Java 5.

Dès 2001, il se met à plancher sur un langage qui serait extensible et qui permettrait de résoudre des problèmes de toute taille à partir des mêmes concepts. Comme un clin d'œil à cette volonté de rendre ce langage *scalable,* il lui donne pour nom Scala (SCAlable LAnguage). Afin d'atteindre cette scalabilité, il travaille sur une approche innovante, mixant à la fois le paradigme objet mais également le paradigme fonctionnel. Désormais en version 2.7.7, le langage semble arriver à maturité et l'intérêt de plus en plus grand qu'il suscite conforte son créateur dans son idée que Scala est le futur langage de programmation star pour la plate-forme Java.

UN LANGAGE 100% OBJET

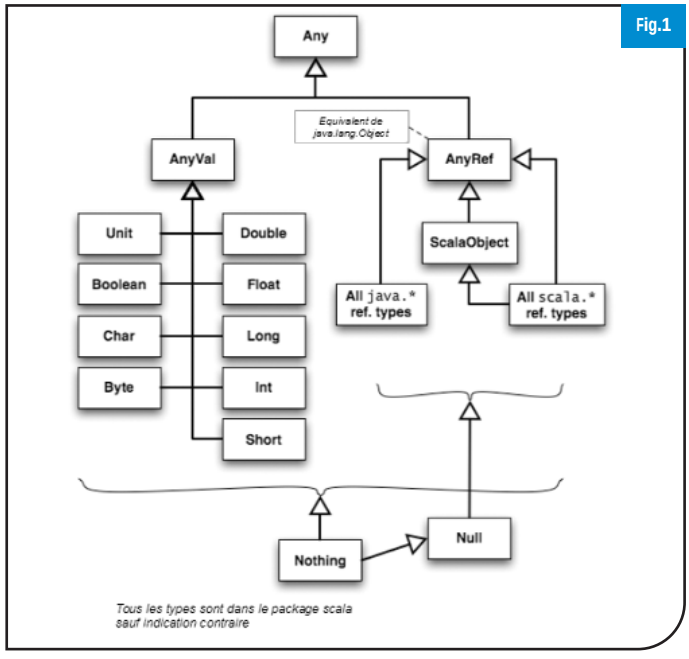
Scala se distingue tout d'abord du langage Java par son modèle orienté objet pur [Fig.1]. Contrairement à Java qui avait amené quelques déviations avec notamment des types primitifs, des données membres statiques et une syntaxe spéciale pour la gestion des tableaux, tout est objet en Scala ! Ainsi, les tableaux sont gérés via une classe spécifique nommée Array, de même que l'emploi de singletons natifs est utilisé pour la gestion de données devant être uniques. Ainsi, le classique exemple "Hello World" peut s'écrire de la manière suivante en Scala :

```
object HelloWorld {
  def main(args : Array[String]) {
    val msg = "Hello World !"
    println(msg)
  }
}
```

Ici, on utilise un objet Scala pour définir un *singleton* dont la gestion est transparente pour le développeur. Au sein de ce dernier, on définit une méthode *main* à l'aide du mot-clé *def*. Cette méthode prend en entrée un tableau de *String*. On remarque qu'en Scala la définition du type d'une variable suit son nommage. Ensuite, on utilise la méthode *println* pour afficher à l'écran notre célèbre message. Cette dernière est utilisable car Scala importe implicitement tout le contenu du package *java.lang* de Java.

Le lecteur attentif aura remarqué que la méthode *main* ne déclare pas un *void* par exemple comme en Java pour préciser qu'elle ne retourne rien. En effet, en Scala le retour vide est de type *Unit* et seule la valeur () est de type Unit. Son utilisation ici n'est pas requise. La dernière instruction d'une fonction est automatiquement renvoyée en sortie. Ici, la dernière instruction étant un *println*, on renvoie bien un type *Unit*. En outre, il est intéressant de noter que l'utilisation du **;** en fin d'instruction est également optionnelle dans la

Fig.1



Hiérarchie de type du modèle objet de Scala.

mesure où l'on respecte la règle d'utiliser une instruction par ligne. Enfin, on notera que la définition d'une valeur se fait à l'aide du mot-clé *val* et que, par définition, une valeur est immuable. Pour pouvoir utiliser des variables, il faut passer par le mot-clé *var*. Cette immuabilité comme principe de base du langage est un apport primordial dans le cadre de la programmation concurrente mais nous y reviendrons par la suite.

La déclaration de notre variable *msg* n'est pas suivie de la définition d'un type. Bien que Scala soit un langage statiquement typé, dont les types des variables sont donc connus à la compilation, cela est rendu possible par le moteur d'inférence de type du langage qui dans bien des cas nous permet de nous abstraire de cette déclaration de type. Ceci soulageant le développeur d'une tâche que le compilateur peut réaliser. Ainsi, Scala permet la concision et la légèreté de langages de scripts tels que Python ou Ruby tout en gardant la sécurité qu'apportent les langages statiquement typés.

DES CLASSES ALLÉGÉES

En Scala, le concept de classe est toujours utilisable via le mot-clé class. Et si l'ensemble de la syntaxe du langage Java reste bien entendu utilisable, Scala va beaucoup plus loin en proposant des facilités permettant de réduire grandement la taille de vos classes. Prenons l'exemple d'une classe modélisant une personne :

```
// classe Person paramétrée par le type T qui est publique par défaut
class Person[T](var lastName: String, var firstName: String,
var age: Int, var objects:List[T]) {
  // variable d'instance non définie à la construction
  var nickname : String = _
// vérification de valeurs à la construction
if(lastName == null){
    error("Last Name must be not null !")
  }
  if(age < 0 ){
    error("Age must be greater or equal to 0")
  }
  // définition d'un constructeur auxiliaire
 def this(lastName: String, age: Int) = this(lastName, "Undefined",
 age, Nil)
  // surcharge de la méthode toString
 override def toString() = "Last Name = " + lastName + " / First
 Name = " + firstName + " / Age = " + age
  // définition d'une méthode simple
  def printAll() {
    println(toString)
    // utilisation d'une closure avant l'heure ;)
    objects.foreach(obj => println("Objet : " + obj))
  }
}
```

Le premier détail qui peut étonner concerne le constructeur de la classe *Person*. En effet, celui-ci est placé directement au niveau de la déclaration de la classe, contrairement à ce que l'on fait en Java ou dans la plupart des langages orientés objets. Par défaut, les arguments passés en entrée de ce constructeur sont considérés comme immuables et ne sont pas accessibles à l'extérieur de cette classe. Dans notre exemple, ils sont précédés du mot-clé *var,* ce qui permet de les déclarer comme des variables muables et les rend automatiquement accessibles depuis l'extérieur de la classe *Person*. Lors de la création d'un objet, il arrive fréquemment qu'on ne souhaite pas initialiser certaines données. En Scala, les données membres d'une classe ne sont pas initialisées implicitement avec la valeur par défaut du type correspondant, comme cela est le cas en Java. Ainsi, il est nécessaire d'initialiser la variable *nickname* dans la classe *Person*. Puisqu'on ne souhaite pas la renseigner à sa création, on utilise le caractère magique Scala _ qui va se charger de l'initialiser correctement avec la valeur par défaut de son type déclaré. Scala étendant le concept de type abstrait aux données membres des classes, la non initialisation de la variable *nickname* aurait rendu la classe Person abstraite et aurait contraint ses classes filles à définir sa valeur d'initialisation.

Scala permet en outre de définir des contraintes sur les données membres de la classe qui seront testées lors de son instanciation. La définition de constructeurs auxiliaires se faisant quant à elle à l'aide du mot-clé *this* en faisant appel au constructeur principal défini à la déclaration de la classe. Père de l'API Generics de Java, Martin Odersky a, bien entendu, implémenté ce concept dans son bébé mais en y ajoutant des possibilités encore plus grandes.

L'utilisation de classes abstraites ne présente pas énormément de spécificités à cela près qu'il suffit de ne pas définir le corps d'une méthode ou de ne pas initialiser une donnée membre pour qu'elles soient considérées comme abstraites par le compilateur. Dans ce cas-là, il est nécessaire de préfixer la déclaration de la classe par le mot-clé abstract. Enfin, les classes filles doivent faire appel au constructeur principal de leurs classes parentes directement dans la définition de l'héritage via le mot-clé *extends*. L'exemple suivant vient détailler ces quelques points :

```
abstract class Person (var firstName : String, var lastName :
String, var age : Int) {
  // partie abstraite de la classe Person
  var nickname : String
  def doAction(value:String)

  override def toString = "[Person : firstName = " + firstName
+ ", lastName = " + lastName + ", age = " + age + "]"
}
// héritage simple avec appel au constructeur de la classe mère
class Student(firstName : String, lastName : String, age : Int)
                        extends Person(firstName, lastName, age) {
  // Initialisation de la donnée membre abstraite
  var nickname : String = _

  def doAction(value:String) {
    println("I'm student " + value + " / "  + toString)
  }
}
```

LES TRAITS : DES INTERFACES SURVITAMINÉES ...

Dernier point central des langages orientés objets, les interfaces qui définissent des services tout en permettant un découplage de leur implémentation. Là encore Scala innove en proposant le concept de traits qui tirent parti de la puissance des interfaces et des classes abstraites Java tout en les alliant aux possibilités de *mixin* offertes par Ruby. Ainsi, non seulement les traits définissent des services

mais ils offrent en outre la possibilité de spécifier des comportements en leur sein. Enfin, ces traits peuvent être composés via ce que le créateur de Scala appelle la *mixin-composition*. Tout ceci étant très théorique, prenons un premier exemple montrant la puissance des traits Scala :

```
// trait simple spécifiant qu'on doit pouvoir avoir accès au
poids d'une personne
trait Person {
  def weight() : Int
}
// trait générique de comparaison d'éléments tirant parti de
la simulation
// de surcharge d'opérateurs Scala ...
trait Comparable[T]{
  def < (that: T) : Boolean
  def <= (that: T) = (this < that) || (this == that)
  def > (that: T) = !(this <= that)
  def >= (that: T) = !(this < that)
}
// implémentation d'une personne possédant un poids et pouvant
être comparée
class Student(var lastName : String, var firstName : String,
var weightc : Int)
                      extends Person with Comparable[Person] {
  def weight() = weightc
  // définition nécessaire d'une seule méthode
  def < (that : Person) = weight() < that.weight()
}

// exemple simple d'utilisation
object Main {
  def main(args: Array[String]) {
    val student = new Student("lastName", "firstName", 19)
    val student2 = new Student("lastName2", "firstName2", 20)
    println(student > student2)
  }
}
```

Ici, on considère tout d'abord un trait générique *Comparable* qui permet de comparer des éléments à l'aide des symboles de comparaison mathématiques classiques. Mathématiquement parlant, ces fonctions de comparaisons peuvent toutes être définies à partir d'une seule à savoir la fonction strictement inférieure. Avec une interface Java, il serait impossible de tirer parti de ces propriétés mathématiques. Avec les traits Scala, cela devient possible et même très facile ... En effet, les traits offrent la possibilité de définir des comportements en leur sein. Ainsi, il est tout à fait possible de définir la fonction < comme étant abstraite puisque non définie et de définir les autres fonctions de comparaison à partir de celle-ci. Ainsi, les classes implémentant ce trait de comparaison n'auront qu'à définir une seule fonction par la suite pour avoir accès à l'ensemble des opérateurs de comparaison mathématique !

Pour mettre en exergue cette caractéristique, on définit un trait *Person* implémenté par une classe *Student* qui doit donc définir la méthode *weight* exposée par le trait *Person*. En outre, on souhaite rendre les instances de *Student* comparables via des opérateurs mathématiques. Pour cela, *Student* étend logiquement le trait *Comparable* en le paramétrant avec le type d'éléments sur lesquels porte la comparaison. On constate que l'héritage multiple de traits est toléré et se fait par l'utilisation du mot-clé *with*.

Finalement, la classe Student se contente de définir la méthode < et permet l'accès à l'ensemble des opérateurs de comparaison définis par le trait *Comparable.* Au passage, on notera que l'utilisation de caractères alphanumériques comme identificateur de fonctions donne à Scala la possibilité de simuler la surcharge d'opérateurs de manière élégante.

... AUX POSSIBILITÉS STUPÉFIANTES

Autre point fort des traits Scala, le concept de *mixin-composition* inspiré du langage Ruby. Cela permet de bénéficier d'une approche orientée module au sein même du code où les modules sont représentés par des classes ou des traits. Ainsi, on peut facilement réutiliser des modules définis et les utiliser avec différentes classes. Ceci ouvre un grand nombre de possibilités et s'avère très puissant. On peut par exemple imaginer utiliser la *mixin-composition* pour réaliser des décorateurs permettant notamment de rajouter ou de supprimer des fonctionnalités. Prenons l'exemple suivant :

```
// trait Service avec une méthode execute
trait Service {
  def execute(value: Any)
}

// trait pour un service de log
trait LoggerService extends Service {
  abstract override def execute(value: Any) {
    println("Before execute with param : " + value)
    super.execute(value)
    println("After execute")
  }
}
// trait pour un service d'audit
trait AuditService extends Service {
  abstract override def execute(value: Any) {
    println("Start audit : " + value)
    super.execute(value)
    println("After audit : " + value)
  }
}
// Implémentation du service
class ServiceImpl extends Service{
  def execute(value: Any) {
    println("Execute with " + value)
  }
}

object Main {
  def main(args: Array[String]) {
    // ici on décore l'instanciation du service avec un logger
 et un audit
     def service = new ServiceImpl with LoggerService with
AuditService
    service.execute("Test")
  }
}
```

On définit tout d'abord le comportement attendu d'un service basique avec une simple méthode *execute*. Puis, on crée 2 traits spécifiant un service de log et un service d'audit. Tous deux implémentent le trait Service et *overrident* sa méthode *execute* tout en la redéclarant *abstract,* ce qui est essentiel dans le cadre de la composition par *mixin*. A l'instanciation d'une classe *ServiceImpl,* on va la composer avec les traits *LoggerService* et *AuditService* à l'aide du mot-clé *with*. Ainsi décorée, lors de l'appel à sa méthode *execute,* on passera d'abord dans la méthode *execute* de ses décorateurs. L'ordre de passage étant l'inverse de l'ordre de composition par *mixin*. Dans notre cas, cela signifie qu'on passera d'abord dans la méthode execute du trait *AuditService* puis dans celle du trait *LoggerService* et enfin dans celle de la classe *ServiceImpl,* comme on peut le constater sur la sortie d'écran de la [Fig.2].

La composition par *mixin* laisse entrevoir de nombreuses possibilités comme le démontre cet exemple. On pourrait même considérer qu'avec ce concept, Scala offre de la programmation orientée aspect directement au sein de son langage ...

PACKAGES ET ENCAPSULATION

Scala reprend le principe des packages Java pour rassembler de manière logique un ensemble d'objets. Le mot-clé package est conservé et la déclaration de plusieurs packages au sein d'un même fichier source Scala est permise. L'import du contenu de ces packages est réalisé à l'aide du même mot-clé qu'en Java mais il offre une plus grande souplesse et une plus grande concision. En effet, il autorise l'import de plusieurs classes d'un même package en procédant comme suit : *import myPackage.{x, y, z}* . D'autre part, il est possible de renommer une classe importée et de l'utiliser via son alias dans le fichier source concerné. La construction est la suivante : import myPackage.{x => aliasX}. Enfin, outre l'import implicite du contenu de *java.lang,* le compilateur Scala importe également le package scala et l'objet scala.*Predef*.

L'encapsulation des données proposée par Scala amène 3 niveaux d'accès là où Java en propose 4 avec le fameux niveau package-level pour lequel aucun mot-clé n'existe et qui cause bien des tracas à bon nombre de développeurs. Par défaut, le niveau d'accès *public* est appliqué. Le niveau d'accès *private* restreint la visibilité d'une donnée à la classe à laquelle il appartient. En plus de cela, il permet de préciser si ce niveau d'accès *private* peut être étendu au package englobant via la construction suivante : *private[myPackage] def myFunction()* . Ici, le niveau d'accès *private* de *myFunction* est étendu au package englobant en plus de la classe ou l'objet dans lequel elle est déclarée. Quant au niveau d'accès protected Java, il a la même signification en Scala.

La première partie de cet article prend fin et nous aura permis de découvrir les caractéristiques principales du paradigme objet de Scala. Ces dernières, différentes en de nombreux points de ce qu'on peut connaître en Java, offrent de nouvelles perspectives aux développeurs Java. La seconde partie permettra, quant à elle, de mettre en avant ses aspects fonctionnels qui viennent lui apporter toute sa souplesse et font de lui un langage extensible.

Fig.2

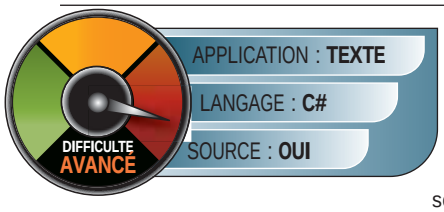
```
Console  Problems  Tasks
<terminated> Main (3) [Scala Application] C:\Program Files\Java\jre6\bin\javaw.exe
Start audit : Test
Before execute with param : Test
Execute with Test
After execute
After audit : Test
```

Exemple de composition par mixin

■ Sylvain **Saurel** – *Ingénieur d'Etudes Java / JEE*
ACP – www.acp-qualife.fr - *sylvain.saurel@gmail.com*

Générez des documents sous Visual Studio avec le langage T4

Comme Eclipse ou NetBeans, Visual Studio est un vaste univers, avec des recoins peu explorés. Nous découvrons aujourd'hui le langage T4, intégré à tous les Visual Studio depuis l'édition 2005.



Eclipse ou NetBeans sont des IDE puissants et extensibles. Cela est bien connu, parce que leur code est ouvert. Visual Studio, l'IDE de Microsoft est lui aussi très puissant et extensible, ce que l'on sait moins. Il existe pourtant un Visual studio SDK qui permet d'étendre l'IDE. Lire la documentation de ce SDK fait découvrir un monde, d'ailleurs très complexe. Parmi d'autres possibilités, le SDK nous apprend que l'on peut, avec Visual Studio, créer ses propres langages, pour répondre à des besoins particuliers. De tels langages sont appelés des DSL, pour Domain Specific Langage, et en bon français, des langages dédiés. Un langage dédié est le contraire d'un langage généraliste comme le sont des Java, Python, C++, C#, etc. Un langage dédié est un "petit" langage permettant de résoudre des problèmes particuliers dans un contexte précis. A partir de son SDK, Microsoft a créé un de ces langages. Ce langage est surnommé T4, appellation plus pratique que " Text Template Transformation Toolkit". Comme le nom complet le suggère, T4 permet de générer, à partir de modèles, des documents texte pour divers usages. Si le SDK de Visual Studio n'est pas supporté par les éditions Express, T4 est bel et bien présent dans toutes les éditions et chacun pourra expérimenter avec. T4 est documenté en ligne sur le site http://msdn.microsoft.com au chapitre "Generating Artifacts Using Text Templates". Quand on aborde T4, on se demande, bien à quoi il peut servir... mais à l'usage on découvre que ses applications sont nombreuses... Génération de code, ou de documentation, par exemple. A travers cet article nous allons voir comment mettre en place un générateur de documentation capable de générer un fichier de documentation pour chaque classe contenue dans un assembly.

1 HELLO WORLD !

Langage dédié ou pas, la tradition est la tradition :-) Créez un projet, le plus rudimentaire possible, avec votre Visual Studio. Un projet de type console en C# est parfait. VB irait aussi comme nous le verrons plus loin. Il n'est pas possible de créer un projet "T4". Votre projet créé, il contiendra un fichier *program.cs*. Transformez-le en un fichier *MaClasse.cs* contenant ce code.

```
using System;
using System.Text;
using System.Windows.Forms;


namespace basic {
  public class MaClasse {
    public MaClasse() {}
    public void Hello(String s) {
      MessageBox.Show(s);
    }
    static void Main(string[] args) {
      MaClasse p = new MaClasse();
      p.Hello("Programmez!");
    }
  }
}
```

Tant que nous y sommes, je vous propose de créer un fichier *MonAutreClasse.cs,* contenant exactement le même code, à l'exception d'une seule méthode dont le nom passe de *Hello* à *AutreHello*. (Vous trouverez tous ces exemples sur notre site http://www.programmez.com). Au point où nous en sommes, nous avons un assembly de classes C# tout à fait ordinaire. C'est le contenu de cet assembly que nous documenterons automatiquement avec T4. Il est maintenant temps de manipuler celui-ci; Ajoutez à votre projet un fichier texte et baptisez le *hello.tt*. L'extension est significative. Tellement significative que dès que vous sauvegarderez votre fichier, Visual Studio l'exécutera. A ce moment, un message d'avertissement sera affiché, que vous pourrez ignorer par la suite. Voici le contenu de *hello.tt*

```
<#@ output extension=".txt" #>
Programmez!
<# Write("Abonnez-vous! :-)"); #>
```

Trois lignes de code qui à elles seules appellent beaucoup de commentaires. Notre fichier est un modèle de document écrit en T4. Comme tel, il contient un mélange de directives, de données et de code C#! Tout ce qui n'est pas encadré par les signes <# et #> constitue des données. Ainsi dans notre exemple le mot Programmez! est une donnée. La première ligne est une directive qui force T4 a générer un fichier d'extension *.txt*. Un fichier de nom hello.txt sera donc généré. Sans cette directive c'est un fichier *.cs* qui sera généré, ici *hello.cs*. La dernière ligne est une ligne de code. Une méthode de nom *Write,* appartenant à une classe pour l'instant inconnue, écrit du texte dans le document généré. Ainsi, après exécution de notre modèle en T4, nous trouvons, dans notre projet, un fichier *hello.txt* dont le contenu est le suivant :

```
Programmez!
Abonnez-vous! :-)
```

Pour exécuter un template T4 il suffit de le sauvegarder, ou bien de cliquer, dans l'explorateur du projet, avec le bouton droit de la souris puis de sélectionner "Run Custom Tool", comme illustré [Fig.1]. La dernière possibilté est de cliquer sur le bouton "Transform All Template" dans la barre du boutons de l'explorateur du projet, comme illustré [Fig.2]. Dans ce cas tous les templates du projet, ou de la solution, seront exécutés.

2 FONCTIONNEMENT

La génération d'un document à partir d'un template T4 se déroule en trois étapes. A la première étape, le moteur de T4 analyse le template et à partir de son contenu génère une classe C# dérivant de *Microsoft.VisualStudio.TextTemplating.TextTransformation*. Le code généré pour notre exemple est probablement très proche du code donné ci-dessous

```
public class GeneratedtextTransformation :
 Microsoft.VisualStudio.TextTemplating.TextTransformation
{
 public override String TransformText()
 {
   this.Write("Programmez!\n");
   this.Write("Abonnez-vous! :-)\n");
   return this.GenerationEnvironment.ToString();

 }
}
```

Nous voyons maintenant où se situe concrètement la méthode *Write* invoquée par <# Write("Abonnez-vous! :-)"); #>. A la seconde étape, le moteur instancie la classe C# générée et exécute sa méthode *TransformText* qui retourne le fichier généré à partir du modèle dans une chaîne. La troisième et dernière étape consiste simplement à écrire le contenu de cette chaîne dans le fichier cible. La compréhension de ce mécanisme ouvre des perspectives.

3 UN HELLO WORLD FENÊTRÉ

En soi, avoir un template T4 qui ouvre une fenêtre n'est pas très intéressant. Mais c'est pour nous une occasion de faire le tour des principales directives T4. En outre, ouvrir une *MessageBox* peut s'avérer intéressant au cours du débogage. C'est en tout cas moins lourd que d'utiliser la fonctionnalité de débogage prévue. Voici donc un template qui ouvre une *MessageBox* avant d'écrire quelques lignes dans le fichier généré :

```
<#@ template language="C#" #>
<#@ assembly name="System.Windows.Forms.dll" #>
<#@ import namespace="System.Windows.Forms" #>
<#@ output extension=".txt" #>
<#@ include file="data.tt" #>

<# MessageBox.Show(texte); #>
<# Write(texte); #>

<#@ include file="data.txt" #>
```

La première directive affine la définition du template lui-même. Ici nous demandons, avec le paramètre *language*, à travailler avec C# 2.0, ce qui est le comportement par défaut. Nous pourrions demander C# 3.5 en donnant language="C#v3.5" ou Visual Basic en donnant language="VB". Nous incitons vivement le lecteur à consulter la documentation en ce qui concerne le paramètre *inherits* qui permet de spécifier n'importe quelle classe dérivée de *Microsoft.VisualStudio.TextTemplating.TextTransformation*, ce qui ouvre de très riches possibilités. Ensuite vient la directive *assembly* qui permet de référencer un assembly puis la directive *import* qui est l'équivalent de l'instruction *using* de C#. Plus loin, nous découvrons la directive *include* qui permet d'embarquer un autre template. Ici ce template contient :

```
<# String texte = "Programmez! T4"; #>
```

Ensuite, nous ouvrons la *MessageBox* qui affiche le texte défini dans le template importé. Pour terminer, nous voyons que nous pouvons aussi importer un fichier de données sous la forme d'un fichier texte. Celui-ci contient :

```
Programmez!
Abonnez vous!
```

Et au final le fichier généré par notre template contient

```
Programmez! T4
Programmez!
Abonnez vous!
```

Voici maintenant une variante de l'exemple précédent. Dans cette variable nous générons le fichier en fonction de ce que répond l'utilisateur à l'ouverture de la *MessageBox*. Remarquez bien comment une variable C# devient une chaîne dans le fichier généré, via la directive <#= #>

```
<#@ template language="C#" debug="true" #>
<#@ import namespace="System.Diagnostics" #>
<#@ assembly name="System.Windows.Forms.dll" #>
<#@ import namespace="System.Windows.Forms" #>
<#@ output extension=".txt" #>
<#@ include file="data1.tt" #>

<#
 DialogResult dr =
 MessageBox.Show(texte, "Programmez!",
      MessageBoxButtons.YesNo);
#>
<#= dr #>
<#

if( dr == DialogResult.Yes )
 Write("C'est une très bonne chose");
if( dr == DialogResult.No )
 Write("Abonnez-vous sans tarder");
#>
```

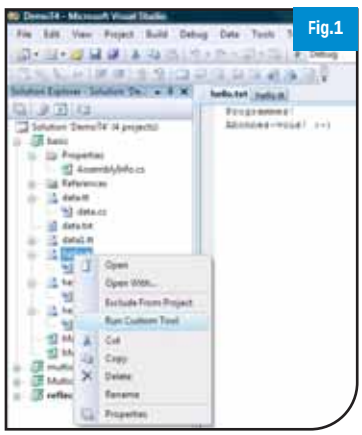


Fig.1

Lancement de l'exécution du template T4 depuis l'explorateur de Visual Studio

Fig.2

Tous les templates T4 peuvent être exécutés par un clic sur un bouton de la barre d'outils de l'explorateur de solution.

Enfin cet exemple montre comment déboguer un template. D'abord on donne le paramètre debug="true" à la directive template, puis on importe le namespace *System.Diagnostics*. A la suite de quoi on peut insérer n'importe où cette directive :

```
<# Debugger.Break(); #>
```

qui, lorsqu'elle est rencontrée, charge le débogueur. Vous pouvez alors continuer l'exécution en pas à pas avec celui-ci.

4 UN PEU DE RÉFLEXION

Tout ceci est très bien, mais jusqu'à présent, on ne peut pas dire que ce soit d'une utilité folle.
Ce que nous voudrions, c'est par exemple écrire un template capable de documenter toutes les classes d'un assembly et surtout que ce template soit suffisamment générique pour pouvoir être réutilisé au sein d'un autre projet.
Ceci est possible grâce à l'utilisation conjointe de T4 et des capacités d'introspection (ou *reflection*) des langages .Net.
Cet article n'ayant pas pour but l'étude de l'introspection en .Net, nous donnons simplement un programme C# qui passe en revue l'assembly 'basic' de notre projet, en extrait les noms des classes le constituant, ainsi que les noms des méthodes de chaque classe.
La capture montre l'affichage produit par ce code : [Fig.3]

```
using System;
using System.Collections.Generic;
using System.Text;
using basic;
using System.Reflection;

namespace reflection {
 class Program {
   void Reflechir(String assembly_name) {
     System.Reflection.Assembly assembly =
       System.Reflection.Assembly.LoadFrom(assembly_name);
     System.Console.WriteLine("Assembly: "
       + assembly.GetName());
     Console.WriteLine("");

     foreach (Type type in assembly.GetTypes()) {
       Console.WriteLine("Type : " + type.Name);
       foreach (MemberInfo member in type.GetMembers()) {
         Console.WriteLine("Membre :" + member.Name);
       }
       Console.WriteLine("");
     }
   }
   static void Main(string[] args) {
     Program p = new Program();
     p.Reflechir(
       "..\\..\\..\\basic\\bin\\debug\\basic.exe");
   }
 }
}
```

Ce code sert de base pour écrire un template qui documente tout l'assembly dans un fichier HTML :

```
<#@ template language="C#" #>
<#@ output extension=".html" #>
<#@ import namespace="System.Reflection" #>
<# String assembly_name = "C:\\Developpement\\Programmez\\Demo
T4\\basic\\bin\\Debug\\basic.exe"; #>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
     "http://www.w3.org/TR/html4/strict.dtd">

<html>
<#

System.Reflection.Assembly assembly =
 System.Reflection.Assembly.LoadFrom(assembly_name);
 String le_titre = assembly.GetName().ToString();

#>

<h1><#= le_titre #></h1>
<ul>
<#
foreach (Type type in assembly.GetTypes())
{
#>
<p />
Type : <#= type.Name #>
 <#
 foreach (MemberInfo member in type.GetMembers())
 {
 #>
 <li />Membre : <#= member.Name #>
 <#
 }
 #>
<#
}
#>
</ul>
</html>
```

Si tout ce qui précède est bien assimilé, le présent code ne mérite que peu de commentaires. On notera que si, dans le code C#, on donne la localisation de l'assembly à traiter relativement à l'arborescence du projet, ceci n'est pas possible avec T4. En effet, le moteur de T4 n'est pas situé dans cette arborescence et lorsque Visual Studio le lance, le répertoire courant est positionné sur sa localisation. Voilà pourquoi

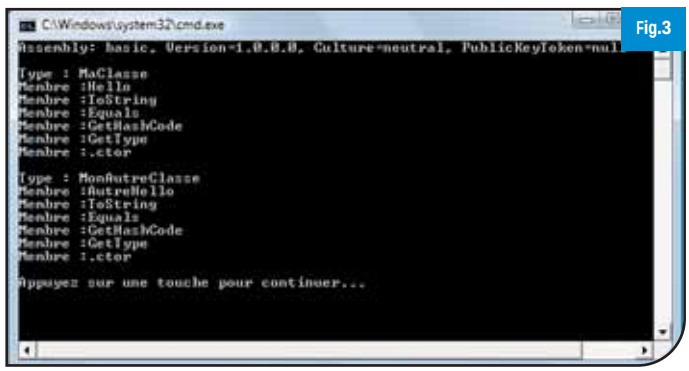


Fig.3

Mise en oeuvre des capacités d'introspection de C#.

nous donnons le répertoire absolu pour l'assembly, répertoire que le lecteur devra adapter à sa configuration. En lui-même, ce template est suffisament générique puisqu'il suffit de spécifier l'assembly à traiter. En outre nous pourrions l'enrichir. En effet nous savons que Visual Studio sait générer un fichier XML de documentation à partir de commentaires écrits à cet effet dans le code C#. Par exemple :

```
namespace basic
{
    /// <summary>
    ///  Cette classe est sans équivalent dans le monde
    /// </summary>
    public class MonAutreClasse
    {
// etc. etc.
```

Si comme votre serviteur, vous n'avez pas d'atomes crochus avec XSLT, vous préférerez utiliser conjointement T4, l'introspection de C# et les classes .Net pour travailler avec XML et générer une documentation aux petits oignons.

5 GÉNÉRER PLUSIEURS FICHIERS À PARTIR D'UN SEUL TEMPLATE

Générer un seul document HTML pour un assembly est sans doute très bien, mais maintenant nous nous posons le problème de générer un fichier par classe, et ceci à partir d'un seul template. Nous savons que normalement le moteur de T4 ne génère qu'un document par template traité.
La solution consiste tout simplement à instancier le moteur T4 au cours du traitement, autant de fois que l'on veut générer un fichier. Dans le projet *multidoc* de nos exemples, ce travail est effectué par *TemplateProcessor.tt*. En voici le code, qui a été pris sur le site http://www.olegsych.com

```
<#@ output extension=".txt" #>
<#@ template language="C#" hostspecific="True" debug="true" #>
<#@ import namespace="System.IO" #>
<#@ import namespace="Microsoft.VisualStudio.TextTemplating" #>


<#+
  void ProcessTemplate(string templateFileName, string output
FileName)
  {
   string templateDirectory = Path.GetDirectoryName(Host.TemplateFile);
     string outputFilePath = Path.Combine(templateDirectory,
 outputFileName);

    string template = File.ReadAllText(Host.ResolvePath(temp
lateFileName));
    Engine engine = new Engine();
    string output = engine.ProcessTemplate(template, Host);

    File.WriteAllText(outputFilePath, output);
  }
#>
```

Le point à noter est l'utilisation d'une directive que nous n'avions pas encore rencontrée: <#+ #> Le code contenu dans une telle directive devient une méthode de la classe *GeneratedtextTransformation*. Ensuite depuis le fichier *Multidocgenerator,* on demande la transformation de deux templates

```
<#@ include file="TemplateProcessor.tt" #>

<#
 ProcessTemplate("MaClasseDocGenerator.tt", "MaClasseDoc.html");
 ProcessTemplate("MonAutreClasseDocGenerator.tt",
   "MonAutreClasseDoc.html");
#>
```

Vous trouverez le code de ces templates sur notre site : www.programmez.com. Le problème est que maintenant il doit exister aussi un fichier (MonAutreClasseDocGenerator.tt) template pour la documentation de l'autre classe. Autrement dit il faut un template par classe et ceux-ci n'ont plus rien de générique, et notre solution n'est plus acceptable en l'état.

6 PASSER DES ARGUMENTS À UN TEMPLATE

Ce qui nous manque, c'est de pouvoir passer des paramètres à un template. Ce problème est traité dans notre dernier projet d'exemple: *MultidocParams*. Dans *MultidocParams.tt*, nous utilisons la classe *.Net CallContext* qui est un dictionnaire spécifique à un contexte d'exécution :

```
<#@ import namespace="System.Runtime.Remoting.Messaging" #>
<#@ import namespace="System.Reflection" #>
<# String assembly_name = "C:\\Developpement\\Programmez\\Demo
T4\\basic\\bin\\Debug\\basic.exe"; #>
<# String assembly_short_name = "basic"; #>
<# String generator_name = "ClassDocGenerator.tt"; #>
<#@ include file="TemplateProcessorParams.tt" #>

<#
 System.Reflection.Assembly assembly =
   System.Reflection.Assembly.LoadFrom(assembly_name);

 CallContext.SetData("Assembly.Name", assembly_name);

 foreach (Type type in assembly.GetTypes())
 {
     String type_name = type.Name;
     String qualified_type_name = assembly_short_name + "." +
type_name;
     CallContext.SetData("Type.Name", qualified_type_name);
     ProcessTemplate(generator_name, type_name + ".html");
 }

#>
```

Ensuite il suffit de récupérer les paramètres dans ce dictionnaire, ce que nous faisons depuis *ClassDocGenerator.tt* qui est maintenant un template unique, comme nous le souhaitions. (*ClassDocGenerator.tt* est disponible sur notre site). Nous avons atteint notre but, avec des templates génériques et réutilisables.

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

DÉVELOPPEZ VOTRE SAVOIR-FAIRE

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*

*11 numéros par an : 49 €**

Economisez 16,45 €*

**Tarif 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

PROgrammez! Le magazine du développement

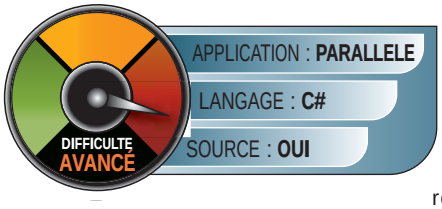
Offre limitée, valable jusqu'au 30 janvier 2010

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 126

Axum, un langage pour la programmation concurrente et distribuée sous .Net

Les machines à microprocesseurs multi-coeurs sont répandues, et avec elles la programmation concurrente. Nous faisons aujourd'hui connaissance avec Axum, un langage Microsoft expérimental, dérivant de C#, et dédié à la programmation concurrente et distribuée.



L'évolution des machines, désormais toutes équipées de processeurs multi-cœurs, voire même de plusieurs, la généralisation des réseaux, Internet en tête, font que la programmation ne pourra bientôt plus être envisagée que comme concurrente et distribuée. Dans ce monde qui évolue rapidement, il y a une chose qui reste elle, bien constante et tenace: la difficulté. Le sujet étant ardu, le travail du programmeur ne doit pas être alourdi inutilement. Ainsi, celui-ci doit pouvoir au maximum être dispensé de gérer le réseau. Il faut donc un environnement ou *runtime* qui assure ce travail. Mais ceci n'empêchera pas le programmeur de se casser la tête avec les classiques problèmes du genre, le *deadlock,* ou verrou mortel, en tête, la *race condition*, ou accès concurrent à une même ressource, n'étant jamais bien loin elle aussi. Si les problèmes sont classiques, les solutions existantes sont nombreuses. Ainsi nous avons par exemple Erlang, langage aussi génial que méconnu. L'air du temps étant à la plate-forme .Net, c'est sur cette plate-forme que des solutions apparaissent en ce moment. Récemment, dans Programmez! 120, nous avons pu apprécier CCR qui est une API Microsoft pour la programmation concurrente sous .Net. Aujourd'hui, nous découvrons Axum, qui est lui un langage. Axum est encore expérimental, en "incubation" chez Microsoft. Axum est un langage qui dérive de C#. On peut dire que c'est un C# avec quelques mots-clés et opérateurs supplémentaires. Ce n'est pas un langage anodin pour autant. Il introduit des outils puissants: les protocoles et les canaux (channels). Plutôt que la programmation orientée objet, on fait avec lui de la programmation orientée agents avec passage de messages, un peu comme on fait de la programmation orientée processus en Erlang. Mais surtout, il introduit quelque chose de très nouveau: sa syntaxe permet de visualiser le flux de données et les étapes de transformation de celui-ci. Si la rapidité d'exécution promise sur son site officiel (http://msdn.microsoft.com /en-us/devlabs/dd795202.aspx) ne semble pas (ou pas encore) vraiment au rendez-vous, l'accès à toutes classes .Net et les apports cités ci-dessus font que Axum vaut le détour.

1 LES OUTILS

Pour travailler avec Axum, l'idéal est de disposer d'un Visual Studio 2008 d'une édition supérieure à l'Express, cette dernière ne supportant pas les plugins. Si on dispose de Visual Studio, on téléchargera le plugin Axum à l'URL donnée plus haut. Sinon on téléchargera à la même adresse AxumLite, qui est une archive contenant des outils en ligne de commande. Dernière alternative: utiliser les deux outils gratuits que sont Visual Studio SDK et Visual Studio Shell, ce dernier supportant les plugins. Le lecteur intéressé par cette possibilité trouvera plus d'information à http://msdn.microsoft.com/en-us/vsx2008/products/bb933751.aspx.

2 HELLO WORLD!

Concurrence ou pas, la tradition est la tradition. Sous Visual Studio, il suffit de créer un projet Axum comme illustré [Fig.1]. Commençons simplement, avec un projet de type Console. Pour cette première application, simplifions le code au maximum. Notamment, enlevons la déclaration *public domain Program*. Au final le code de notre Hello World! doit se réduire à ceci :

```
using System;
using Microsoft.Axum;
using System.Concurrency.Messaging;

namespace HelloWorld
{
 agent Program : ConsoleApplication
   {
     override int Run(String[] args)
     {
       Console.WriteLine("Hello, World!");
     }
   }
}
```

Ce code a l'allure du C# mais déjà nous voyons apparaître un nouveau mot-clé *agent*. Ici notre classe *programme* dérive d'une classe de l'assembly Microsoft.Axum. La simple présence du mot-clé fait de notre classe dérivée un agent. Avec Axum, un agent est une "entité" tout à fait opaque pour le programmeur. Le point important à garder à l'esprit est qu'une application Axum est un ensemble d'agents qui communiquent entre eux par passage de messages. A la différence d'Erlang mentionné plus haut, le programmeur n'envoie pas explicitement les messages et ne doit pas connaître d'identifiants de processus. Ce travail est assuré par le runtime. Le programmeur s'occupe simplement d'écrire dans des canaux de communications appelés Channel dans la terminologie Axum.

3 CHANNELS, PORTS ET POINTS D'INTERACTION

Avec les agents, les channels sont au cœur de la programmation Axum. Ce sont des entités complexes. Pour en avoir une image assez fidèle, on peut les comparer aux pipelines d'affichage des

cartes graphiques. Tout au long d'un pipeline d'affichage, il y a des transformations d'un type de données unique: le pixel. Chaque étape d'un pipeline d'affichage trouve sa correspondance dans les points d'interaction des channels. En revanche, contrairement à un pipeline d'affichage, un channel Axum peut travailler avec des types de données différents. Ces données entrent et sortent du channel par des ports, chaque port étant dédié à un type bien défini. Essayons maintenant de clarifier ces notions avec quelques exemples simples.

4 MANIPULATIONS BASIQUES D'UN CHANNEL

Nous allons écrire un Hello World un peu plus élaboré que le précédent. Nous voulons que notre programme salue la personne dont le nom a été donné dans la ligne de commande de notre programme. Pour passer des arguments à la ligne de commande d'un programme depuis l'environnement de Visual Studio, il faut cliquer avec le bouton droit de la souris sur le nom du projet dans le volet d'exploration de Visual Studio, puis sélectionner 'Propriétés'. Ensuite, comme illustré [Fig.2]. sous l'onglet 'Debug' on trouve une entrée pour donner les paramètres de la ligne de commande. Ecrivons maintenant notre programme :

```
using System;
using Microsoft.Axum;
using System.Concurrency.Messaging;

namespace BasicChannel
{
 agent Program : channel Microsoft.Axum.Application
 {
   public Program()
   {
     // Le runtime envoie un message dans le
     // channel intégré PrimaryChannel
     // Ce message peut être lu depuis le port
     // de sortie CommandLine
     String [] args = receive(PrimaryChannel::CommandLine);
     Console.WriteLine("Hello World {0}", args[0]);
     Console.WriteLine("Appuyez sur [RETURN]");
     Console.ReadLine();
     // Envoyer un message dans
     // le port d'entrée ExitCode:
     PrimaryChannel::ExitCode <-- 0;
   }
 }
}
```

La première chose à noter dans ce code est l'apparition d'un nouveau mot-clé: channel. Nous avons donc maintenant notre classe Program qui est de type agent, mais qui semble dériver d'un autre type channel. En fait notre agent est bien un agent et cette écriture est plutôt une commodité syntaxique qu'autre chose. D'ailleurs nous pouvons remarquer que si dans notre premier exemple nous avons redéfini une méthode de la classe de base, nous ne faisons rien de tel ici. Le guide du développeur Axum explique que cet agent s'attache à l'implémentation du channel proprement dit et qu'il endosse le rôle de serveur de messages pour le channel. Quel channel au fait ? Pour l'instant nous n'avons rien défini explicitement. La classe *Microsoft.Axum.Application* comporte un channel intégré du nom de PrimaryChannel que nous utilisons ici. La première ligne de code, avec l'instruction *receive,* met l'application en attente d'un message émis par le *runtime* et contenant la ligne de commande. Lorsqu'on lance le programme, le *runtime* émet immédiatement le message, et l'application continue son cours. On notera que l'instruction *receive* est bloquante. Il est intéressant maintenant de regarder la syntaxe de plus près :

```
receive(PrimaryChannel::CommandLine);
```

receive n'attend pas "n'importe où" mais sur le port de sortie *CommandLine* du *PrimaryChannel.* Port qui, a priori, a été défini pour fournir un tableau de chaîne. Passons maintenant directement à la dernière ligne du code :

```
PrimaryChannel::ExitCode <-- 0;
```

Cette ligne écrit sur un autre port du *PrimaryChannel.* Ecriture qui provoque la terminaison du programme. Ce port est déclaré comme traitant des entiers. Si l'on expérimente ainsi :

```
PrimaryChannel::ExitCode <-- "0";
```

Le compilateur, intraitable, émet un message d'erreur: "The right-hand operand of '<--' has invalid type 'System.String'. Expected type 'System.Int32'". Nous avons ainsi confirmation qu'un port est affecté à un type de donnée et un seul. Nous allons maintenant voir que les ports sont unidirectionnels. On pourrait être tenté d'écrire dans le PrimaryChannel, pour en quelque sorte nous envoyer un message à nous-mêmes :

```
PrimaryChannel::CommandLine <-- args;
```

Le compilateur est aussi intraitable que précédemment: "The left-hand operand is an output port in this context. Can only send to an input port". Pour l'instant Axum, au stade expérimental n'est pas bien documenté. Mais ses librairies sont des assembly .Net. Visual Studio sait donc examiner les types définis dans ses assemblies. Ainsi voit-on sur l'illustration [Fig.3] que *CommandLine* est un port de sortie: impossible d'écrire dedans.

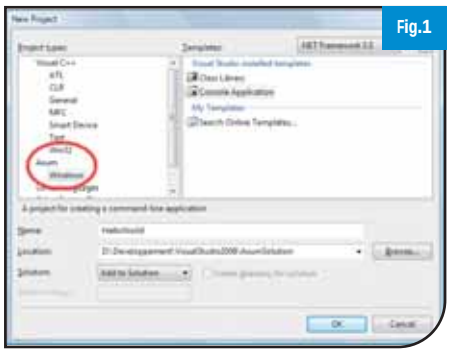


Fig.1

Création d'une application Axum sous Visual Studio.

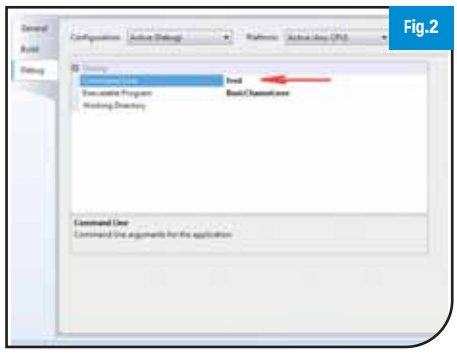


Fig.2

Configuration d'un projet Visual Studio pour passer des paramètres à la ligne de commande de l'application.

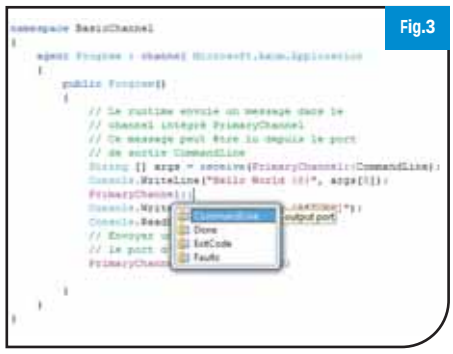


Fig.3

Sous Visual Studio, il est aisé de connaître la direction d'un port.

5 POINT D'INTERACTION DANS UN CHANNEL

Approfondissons un peu par la pratique notre compréhension des channels Axum. Nous allons définir un agent et doter de points d'interactions le channel intégré de cet agent. Le programme écrira une chaîne dans le premier point d'interaction, et cette chaîne sera transformée à chaque point, pour finalement être imprimée sur la console au dernier point.

```
using System;
using Microsoft.Axum;
using System.Concurrency.Messaging;

namespace ProgrammezInteraction
{
 agent Programmez : channel Microsoft.Axum.Application
 {
   function String Magazine(String s)
   {
     return s + " Programmez!";
   }

   function String Abonnement(String s)
   {
     return s + ", abonnez vous! :-)";
   }

   void PrintResult(String s)
   {
     Console.WriteLine(s);
     Console.WriteLine("Fin -- Appuyez sur [RETURN]");
     Console.ReadLine();
     // terminer le programme
     PrimaryChannel::ExitCode <-- 0;
   }

   public Programmez()
   {
     var question = new OrderedInteractionPoint<String>();
     // Creation du pipeline dans le channel
     question ==> Magazine ==> Abonnement ==> PrintResult;
     // Et on démarre effectivement tout le
     // processus de transformation en écrivant
     // dans le premier channel,
     // au premier point d'interaction
     question <-- "Que lire ?";
   }
 }
}
```

Dans cet exemple, tout se passe au sein d'un seul agent attaché à un *channel,* comme dans l'exemple précédent. Cette fois nous définissons des méthodes à l'intérieur de la classe de notre agent. Chacune de ces méthodes, à l'exception bien sûr du constructeur, est un point d'interaction dans le channel intégré. Avec deux de ces méthodes, nous voyons apparaître un nouveau mot-clé: *function*. Une fonction est une méthode sans effet de bord, c'est-à-dire qui n'affecte aucune variable déclarée ailleurs que dans sa portée. Ainsi déclarer le dernier point d'interaction, *PrintResult,* en tant que fonction, aboutirait à une protestation du compilateur :

```
error MA2409: Cannot send or receive messages in a function
```

Notre programme se lance depuis le constructeur de la classe. Nous commençons par instancier un point d'interaction, le premier de la chaîne, en instanciant la classe *OrderedInteractionPoint. Ordered* signifiant que l'ordre des messages arrivant au point d'interaction sera préservé. La deuxième ligne de code est assez formidable en terme d'expressivité :

```
question ==> Magazine ==> Abonnement ==> PrintResult;
```

Cette ligne donne l'ordre des points d'interactions traversés par les données dans le channel. On peut difficilement faire plus clair. Toute la mécanique se met en branle lorsqu'on écrit sur ce premier point d'interaction :

```
question <-- "Que lire ?";
```

Cette chaîne passe alors par tous les points d'interactions. A ce moment, ceux-ci sont autant de threads lancés par le runtime. La documentation, pas vraiment détaillée, au stade expérimental où se situe Axum, ne précise pas comment le runtime gère les threads. Rien ne dit si ce sont des threads lancés sur le système d'exploitation comme avec CCR ou si ce sont des threads gérés par le runtime lui-même comme avec Erlang. Dans le cas d'Erlang le nombre de threads n'est pas limité. Si ce sont des threads Windows, une application sera limitée à 64 threads.

6 CRÉER UN CHANNEL

Jusqu'à maintenant, nous n'avons travaillé qu'avec un seul agent à la fois et avec le channel intégré de cet agent. Si c'est intéressant en soi, c'est assez limité. La programmation Axum prend son sens quand plusieurs agents collaborent entre eux, c'est-à-dire quand un agent peut poster des données dans le channel personnalisé d'un autre agent. C'est que que nous faisons avec l'exemple suivant :

```
using System;
using Microsoft.Axum;
using System.Concurrency.Messaging;

namespace ProgrammezChannel
{
 channel Programmez
 {
   input String mag;
   input String abo;
   output String resultat;
 }

 agent ProgrammezAgent : channel Programmez
 {
   public ProgrammezAgent()
   {
     String chaine = receive(PrimaryChannel::mag) +
       receive(PrimaryChannel::abo);
     PrimaryChannel::resultat <-- chaine;
   }
```

```
}

agent MainAgent : channel Microsoft.Axum.Application
{
  public MainAgent()
  {
    var programmez = ProgrammezAgent.CreateInNewDomain();
    programmez::mag <-- "Programmez!";
    programmez::abo <-- ", Abonnez-vous :-)";
    Console.WriteLine(
    "Simulation d'un travail avant reception du resultat");
    Console.WriteLine("Appuyez sur [RETURN]");
    Console.ReadLine();
    var result = receive(programmez::resultat);
    Console.WriteLine(result);
    // On met fin à l'application
    Console.WriteLine("Fin -- Appuyez sur [RETURN]");
    Console.ReadLine();
    PrimaryChannel::ExitCode <-- 0;
  }
}
}
```

Le premier point important est, au début du code, la déclaration du channel personnalisé, que nous dotons de deux ports d'entrée et un port de sortie. Ensuite nous déclarons un agent qui dérive (pardon pour l'abus de langage) de notre channel. Le constructeur de l'agent lance l'écoute sur les ports. Le second point important se situe dans l'agent principal (MainAgent). Dans le constructeur, nous créons une instance de l'agent "Programmez". Nous faisons cela avec la méthode statique *CreateInNewDomain.* En Axum, un domaine est une sorte de super portée dans laquelle les agents peuvent partager les états. Ceci fait, nous écrivons dans le port de l'agent Programmez, qui se met à traiter les données en arrière-plan. Nous simulons un travail pendant ce temps, puis nous nous mettons en attente bloquante (*receive*) des données de l'agent *programmez* en lisant dans le port de sortie. Vous pourrez constater que nous avons fait de la programmation parallèle sans réfléchir une seconde à la mise en place de thread. Bravo Axum.

7 UN AGENT PERSISTANT

Vous n'êtes pas sans remarquer que l'exemple ci-dessus ne peut écrire qu'une fois dans le port de l'agent *programmez* car celui-ci n'écoute qu'une seule fois son port dans son constructeur. Il n'est pas très intéressant de ne pouvoir écrire qu'une seule fois dans le port d'un agent. Pour remédier à cela, il nous suffit que le code de l'agent qui écoute le port boucle sur lui-même. N'oublions pas que ce code est de facto un thread, grâce au runtime d'Axum. Pour mettre cela en place, nous ré-écrivons notre agent *programmez* en combinant implémentation de channel et point d'interaction (vous trouverez le code complet *ProgrammezChannelInteraction* sur notre site) :

```
agent ProgrammezAgent : channel Programmez
{
 void worker(String s)
 {
   while(true)
   {
     String chaine = receive(PrimaryChannel::mag) +
       receive(PrimaryChannel::abo);
     PrimaryChannel::resultat <-- chaine;
   }
 }

 public ProgrammezAgent()
 {
   var go = new OrderedInteractionPoint<String>();
   go ==> worker;
   go <-- "go";
 }
}
```

Moyennent quoi, le *MainAgent* peut écrire à volonté sur le port de l'agent *programmez.*

8 LES PROTOCOLES

Tout cela est très bien. Axum est clair, expressif. Il est agréable de coder avec lui. Mais si la programmation parallèle c'est bien, la programmation parallèle sans DeadLock c'est mieux. Que se passe-t-il si dans l'exemple précédent le *MainAgent* omet d'écrire dans le port *mag,* avant d'écrire dans le port *abo* de l'agent *programmez* ? Il se mettra en attente sur le port de sortie de l'agent programmez, tandis que celui-ci attendra un message sur son port *mag,* message qui n'arrivera jamais. Et nous avons un deadlock. Pour minimiser le risque qu'une telle situation ne se produise, Axum, via ce qui s'appelle un protocole, permet de définir des états de transitions dans un channel. Dit simplement, cela signifie que l'on peut spécifier un ordre d'écriture sur les différents ports d'un channel. Notre exemple devient alors :

```
channel Programmez
{
 input String mag;
 input String abo;
 output String resultat;
 Start: { mag -> MagRecu; }
 MagRecu: { abo -> AboRecu; }
 AboRecu: { resultat -> End;}
}
```

Avec ce mécanisme, si un agent écrit sur le port *abo* avant d'écrire sur le port *mag,* le runtime d'Axum lèvera une exception.

9 EN GUISE DE CONCLUSION

Axum est un langage de programmation concurrente très expressif. Nous avons vu quelques-uns de ses mécanismes principaux. D'autres aspects restent à aborder, notamment la programmation distribuée, qu'Axum rend très aisée également. Microsoft précise qu'Axum est susceptible d'évoluer, mais tel qu'il est, nous avons les bases d'un grand langage. Nous lui souhaitons longue vie et nous ne manquerons pas de le suivre dans ses évolutions.

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

ARCHITECTURE

Getting started with Oracle SOA Suite 11g R1

*Difficulté : ****
Editeur : Pack Publishing
Auteur : collectif
Prix : 59,99 $

Comment mettre en œuvre l'architecture de services SOA avec la solution dédiée d'Oracle ? Ce livre est une plongée au cœur de cette architecture et de la suite SOA de l'éditeur. Il s'agit tout d'abord de définir la SOA, les concepts de base, l'architecture SCA et les fondations de la SOA Suite 11g. Puis on découvre les premiers projets, les concepts, les déploiements et l'ensemble des process à mettre en place, à connaître. Une bonne initiation. En anglais.

MODÉLISATION

UML 2 2e édition

*Difficulté : ****
Editeur : Eni édition
Auteur : collectif
Prix : 27 €

La modélisation UML reste souvent une énigme pour le développeur ou même le responsable informatique, or la modélisation est aujourd'hui omniprésente. Ce livre s'adresse à toute personne voulant découvrir en douceur les fondamentaux d'UML 2. Cela se fait par plus de 100 QCM, et 60 travaux pratiques, soit en tout près de 30 heures de cours ! Un chapitre particulier présente l'utilisation d'UML en modélisation et en conception (réalisation d'un diagramme d'états-transitions) et une étude de cas complète (librairie en ligne) explique comment mettre en oeuvre UML dans un cadre de commerce électronique. Cette nouvelle édition introduit un chapitre consacré notamment au diagramme de structure composite et à la composition de patterns.

MÉMOS

Collection Open IT Eni édition, *6 €*
Sous forme de court dépliant, la collection OpenIT aborde des sujets précis. Ce mois-ci, la collection s'enrichit des dépliants UML 2 et Shell. Chaque édition comporte les commandes, les explications à connaître. A garder près de son clavier !

LIVRE DU MOIS

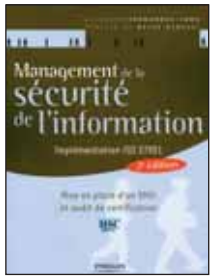
Programmation iPhone OS 3

*Difficulté : *** - Editeur : Eyrolles*
Auteur : Thomas Sarlandie - Prix : 35 €

Incontournable, iPhone continue d'attirer les développeurs et les applications disponibles explosent, plus de 75 000 ! Sans être un guide de référence, cet ouvrage parvient à décrire les fondamentaux du développement iphone, les techniques, les outils. Cela concerne aussi bien les sites web que les applications. Agréable à lire, l'ouvrage permet de comprendre l'architecture et les fonctionnements des applications iphone. Plutôt destiné aux développeurs « avancés », il nécessite une bonne maîtrise des techniques et des astuces de conception à connaître dès le départ !

SÉCURITÉ

Management de la sécurité de l'information 2e édition



*Difficulté : ****
Editeur : Eyrolles
Auteur : Alexandre Fernandez-Toro
Prix : 39 €

La sécurité demeure un secteur sensible et reste encore trop souvent mal implémentée ou mal comprise par les responsables et les développeurs. Ce livre aborde la norme ISO 27001 qui fait référence dans la sécurité informatique. Mais comment une entreprise peut-elle l'utiliser, la déployer et être certifiée ? Une des clés réside dans la mise en place d'un système de management de la sécurité informatique en respectant l'ISO 27001. Cette nouvelle édition va encore plus loin en abordant une autre norme : ISO 27000. Clair et précis, ce livre apporte de précieuses informations sur les bonnes pratiques de sécurité et les procédures à respecter. Nous le conseillons à l'ensemble du service informatique !

CMS

Spip 2

*Difficulté : ***
Editeur : Eyrolles
Auteur : collectif
Prix : 19 €

Comment publier rapidement et simplement son site web ? La réponse est Spip 2. Outil de gestion de contenu open source, certes un peu plus discret depuis quelques mois face à Drupal, mais Spip garde de nombreux adeptes, alors pourquoi pas vous ? Un livre assez complet pour comprendre Spip et sa richesse fonctionnelle. Une excellente approche !

CMS

jQuery

*Difficulté : ***
Editeur : Pearson
Auteur : collectif
Prix : 37 €

Dans cet ouvrage, les auteurs partagent leurs connaissances, leur expérience et leur passion pour jQuery afin de vous aider à comprendre comment cette bibliothèque fonctionne et vous permettre d'en tirer le meilleur parti. Si vos précédentes tentatives de développement JavaScript vous ont laissé perplexe, ils vous aideront à franchir les obstacles dressés par AJAX, les événements, les effets et les fonctionnalités avancées du langage JavaScript. N'attendez plus pour maîtriser la puissance de jQuery !

WEB

GWT

*Difficulté : *** - Editeur : Dunod*
Auteur : Olivier Gérardin - Prix : 25 €

Cet ouvrage s'adresse à tous les développeurs Java qui souhaitent découvrir ce nouvel outil créé par Google pour le développement web. Il intéressera également tous ceux qui utilisent déjà AJAX et JavaScript et qui souhaitent enrichir leurs compétences.
Le principe de GWT est simple : « coder en Java, compiler en JavaScript », et cette simplicité fait aussi sa force. Ce livre se découpe en deux grandes parties : les bases de GWT et les fonctions avancées, que le développeur oublie parfois d'utiliser !

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.

L'INFORMATION SUR MESURE

L'INFORMATION EN CONTINU

www.solutions-logiciels.com