PROGRAMMEZ !
Le magazine du développement
PROgrammez !
www.programmez.com
mensuel n°135 - novembre 2010

PHP

Installer
Déployer
Choisir
Optimiser

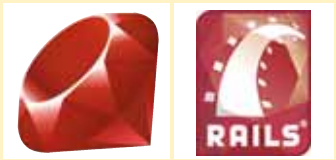
- Choisir son CMS PHP
- Booster PHP sur Windows Server


© Damien Seguy

Webmaster
Expression 4.0 :
la boîte à outil
du développeur web

Code
Faire communiquer
Flex 4 et
Silverlight 4

Langage
Les nouveautés de
Ruby on **Rails 3.0**



© Andresr

Modélisation
Simplifiez-vous le code !

- DSL, UML, MD : à chacun son modèle
- Comment la modélisation aide le développeur ?

Windows Phone 7
Localisez vos applications

NoSQL
Découvrez Cassandra

Langage
Vaadin : le framework survitaminé !

Architecture
La SOA facile avec SCA

M 04319 - 135 - 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

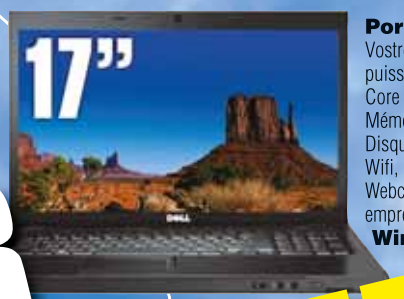
JUSQU'AU 19 DÉCEMBRE

WINDEV®

NOUVELLE VERSION 16

OPÉRATION

1 PC POUR 1 EURO DE PLUS



Portable DELL
Vostro 3700 BTS
puissant Processeur **Intel** Core i5-560
Mémoire **4Go**
Disque **500 Go**
Wifi, Bluetooth, graveur DVD, Webcam 2M, empreinte digitale,...
Windows 7 pro

OU

Station de travail DELL
Precision T1500;
puissant Processeur **Intel** Core i5-750
Mémoire **4Go**
Disque **1.000 Go**
Ecran 20p
Graveur DVD/Bluray
Windows 7 pro 64bits

ACHETEZ WINDEV 16 ET RECEVEZ UN PC DELL POUR 1 EURO DE PLUS
RENDEZ-VOUS SUR WWW.PCSOFT.FR

Offre réservée aux entreprises, administrations, collectivités, indépendants, GIE, assocications,... en France métropolitaine. Chaque élément de l'offre peut être acquis séparément. Tous les détails sont sur www.pcsoft.fr.

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

Environnement de développement professionnel totalement intégré, qui couvre l'intégralité du cycle de développement.

Windows, .Net, Linux, Mac, Internet, Intranet, Java, PHP, Android, Windows Phone 7,...

Fournisseur Officiel de la Préparation Olympique

www.pcsoft.fr

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

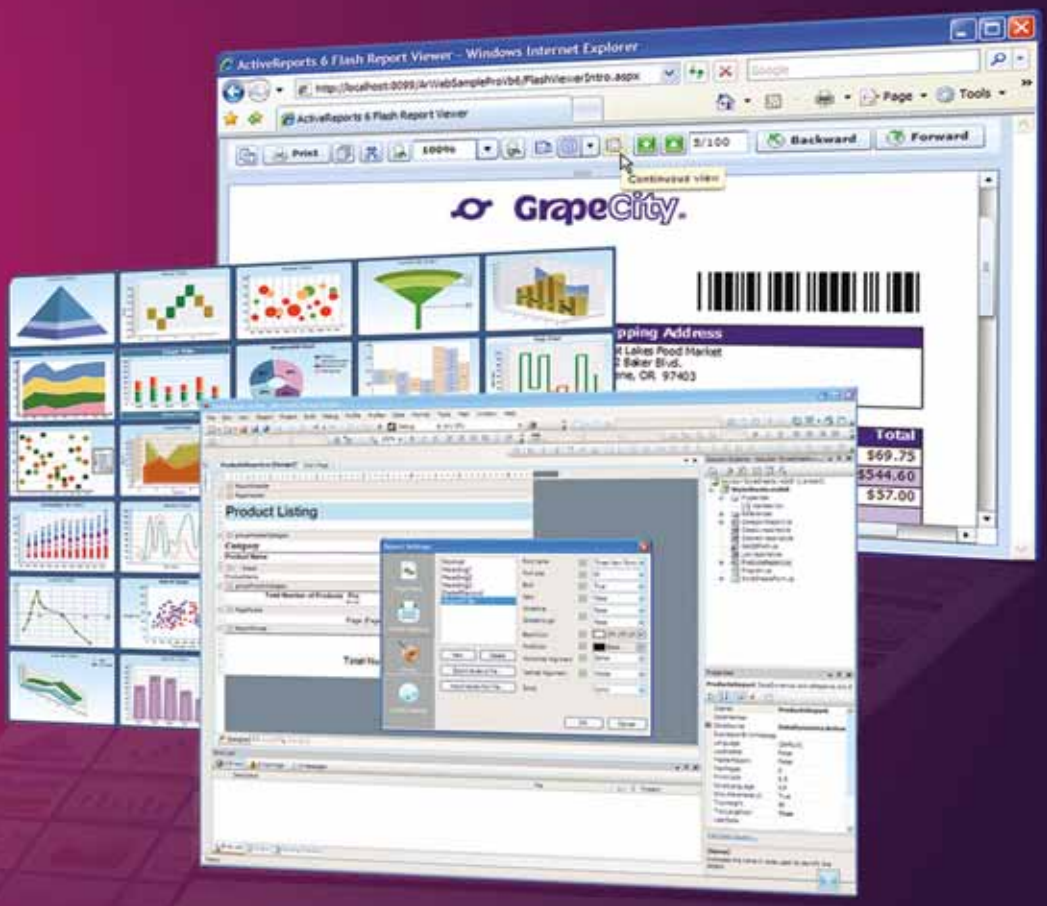
PROgrammez! Le magazine du développement
numéro 135 - novembre 2010

sommaire\\

GrapeCity PowerTools

WE ARE
REPORTING

ACTIVEREPORTS 6



Cet outil de reporting
pour Microsoft Visual Studio.NET
est LE standard.
Il balaie la concurrence, un point c'est tout.

GCPowerTools.com

WE ARE
GrapeCity
Report Analyze Excel

éditO

L'ouverture de la chasse et le dindon de la farce…

Nous l'avions évoqué le mois dernier, le développeur est devenu un enjeu pour tous les éditeurs et constructeurs spécialisés dans les terminaux mobiles. Et ce n'est donc pas un hasard si Microsoft fait de même pour son tout nouveau Windows Phone 7. Le cloud computing n'échappera pas à cette chasse aux développeurs.

Là aussi la réussite ou l'échec d'une plate-forme cloud passera par sa capacité à attirer le développeur. On le voit bien avec la stratégie multiple et ouverte de Windows Azure, et la tentative de VMware de faire la même chose autour de son offre, depuis le rachat de SpringSource et les accords avec force.com et Google. Mais il faudra encore de longs mois avant que cela ne devienne une réelle alternative.

Une autre chasse se dessine depuis quelques mois. La crise a obligé les grands contributeurs des projets open source à rationaliser leurs efforts, ou à en prendre prétexte pour le faire. Et les grands projets cherchent, ceux qui ne l'étaient pas encore, à s'émanciper des éditeurs qui ont beaucoup payé. Le débat symptomatique reste Oracle contre la communauté. La crise autour d'OpenOffice est désormais consommée avec un fork, LibreOffice, qui se positionne franchement en frontal à OpenOffice. Pour les uns, action bénéfique, pour les autres, action dangereuse, brouillant la perception des utilisateurs. Et la guerre n'est pas éteinte. Oracle apporte son soutien à OpenOffice, et souhaite garder la propriété du nom, bien que ce dernier risque de devenir une coquille vide si les départs se succèdent au profit de LibreOffice. Nous ne pensons pas que ces manœuvres soient positives, surtout dans un contexte économique fragile et des développeurs sous pression. Et la situation du monde Java est elle aussi confuse, voire obscure. Malgré, les annonces de l'agenda 2011 – 2012 pour Java 7 et Java 8, la bataille Oracle – communauté – Google se poursuit. Une bataille qui se joue sur plusieurs fronts : on tente de rassurer la communauté avec le futur de Java, on continue à faire pression sur Google et son Android, cette fois-ci en refusant une licence Java à Harmony (source : The Register), et enfin, on travaille main dans la main avec IBM, partenaire incontournable du monde Java depuis 10 ans ! Si le refus de licence Java se confirme envers le projet Harmony d'Apache, ce serait une nouvelle phase de l'action d'Oracle contre Android qui s'appuie sur des éléments du projet Harmony… Et on pourra aussi rajouter les difficultés de Mandriva et de son récent fork ou encore les nouvelles rumeurs de vente de SuSe par Novell… Sans oublier, le fork de MySQL, là aussi, suite au flou stratégique d'Oracle…

Et malgré le succès du récent OpenWorld Forum, la situation de l'Open Source apparaît parfois paradoxale. Ainsi dans le cloud computing, l'Open Source pèse finalement encore bien peu face aux géants du marché et ceux-ci s'appuient souvent sur des briques open source. Mais les projets cloud issus de l'open source ont du mal à s'organiser et à sortir de leurs espaces. Trop d'initiatives cloud émergent, pour faire souvent la même chose, peu ou pas de coordination ou des positions tout bonnement incompatibles. Red Hat dit qu'il faut travailler ensemble pour définir les spécifications, les standards même si cela se fait au détriment de consortiums réellement indépendants. Finalement, il y a de la place pour des acteurs open source et propriétaires. Le monde n'est pas blanc ou noir, il est gris. Nous savons depuis longtemps que le monde informatique ne sera jamais 100 % libre ou 100 % propriétaire. Il sera mixte. Et il faut travailler en bonne intelligence.

S'il y a bien un domaine du monde ouvert qui reste sagement en dehors c'est PHP. Malgré les remous du printemps dernier suite à l'arrêt du projet PHP 6, la communauté continue à évoluer, finalement, sans faire parler d'elle… Et ce n'est pas forcément une mauvaise chose.

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

Le magazine du développement
PROgrammez!
www.programmez.com

Editeur : Go-02 sarl, 21 rue de Fécamp 75012 Paris - diff@programmez.com.
Rédaction : redaction@programmez.com
Directeur de la Rédaction : Jean Kaminsky.
Rédacteur en Chef : François Tonic - ftonic@programmez.com. **Ont collaboré à ce numéro :** F. Mazue, S. Saurel. **Experts :** F. Queudret, A. Mousset, C. Hetier, M. Barbero, G. Rouchon, L. Baumann, J. de Oliveira, P. Desfray, G. Finance, H. Hamon, C.P. de Geyer, J. Renard, P. Couzy, C. Villeneuve, C. Demarey, D. Fournier, N. François, G. Kamgang, A. Petit, A. Vannieuwenhuyze, G. Durelle, J-B Feldis, P. Cauchois,
Illustration couverture : © Damien Seguy
© Andresr.istockphoto
Publicité : Régie publicitaire, K-Now sarl. Pour la publicité uniquement : Tél. : 01 41 77 16 03 - diff@programmez.com.
Dépôt légal : à parution - Commission paritaire : 0712K78366 ISSN : 1627-0908. Imprimeur : S.A. Corelio Nevada Printing, 30 allée de la recherche, 1070 Bruxelles Belgique. **Directeur de la publication :** J-C Vaudecrane
Ce numéro comporte 1 encart libre Component Source sur une partie du tirage.

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

PROCHAIN NUMÉRO

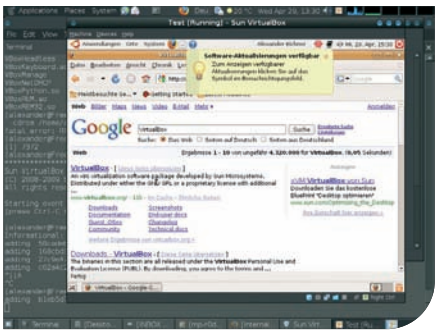
N°136 décembre 2010
parution 30 novembre

✔ **Microsoft innove dans les IDE !**
Découvrez Visual Studio LightSwitch et WebMatrix : deux outils de développement en quelques clics !

✔ **MySQL contre MySQL**
Les forks vont-ils terrasser l'original ?

✔ **GreenIT et le développeur**
Comment concilier écologie et développement ?

■ Le **W3C** tente de calmer les ardeurs autour de HTML5. Depuis un an, les annonces se multiplient, les navigateurs supportent les balises disponibles et quelques outils sortent pour développer en CSS3 et HTML5. Mais pour le W3C cette frénésie n'a pas à avoir lieu. Car tout d'abord, les spécifications ne sont toujours pas définitives et parce qu'il faudra au moins 2 ans avant que HTML5 passe en statut « recommandation ».

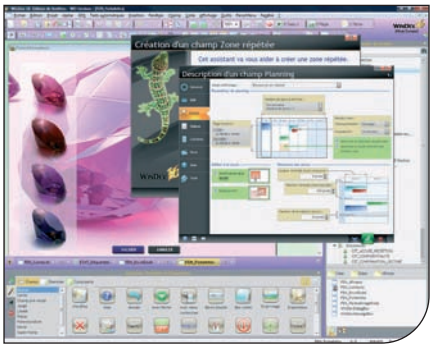


■ **VirtualBox 3.2.1** est disponible depuis mi-octobre. Il s'agit d' une version de maintenance corrigeant de nombreux bugs sur les supports 3D, SATA et iSCSI ou encore le partage réseau. D'autre part, un projet est annoncé : une interface PHP / Ajax pour VirtualBox, phpVirtualBox ! Il s'agit d'une implémentation très complète de la solution !
Site : http://www.virtualbox.org/

■ **Ubuntu** propose à tout le monde de tester durant 60 minutes son dernier système : Ubuntu 10.10 directement dans le cloud sur une instance Amazon EC2 ! On peut choisir le type de Ubuntu Server (avec wordpress, ou wiki ou installation de base). Attention : on peut tester uniquement l'édition serveur. Site : www.10.cloud.ubuntu.com

■ **Amazon Web Services** propose un SDK PHP ! Il inclut des exemples, librairies, documentations pour créer, porter des applications PHP sur Amazon EC2 mais aussi Amazon S3, Amazon SimpleDB, etc. Cependant, Amazon est encore en retard par rapport à un Windows Azure (certes on ne peut pas comparer un IaaS avec un PaaS) ou même un VMware – Spring, même si le couple reste bien trop Java.

PC SOFT dévoile les versions 16 de ses outils



Comme chaque année, PC SOFT dévoile en automne la nouvelle gamme WinDev, WebDev et WinDev Mobile. Et d'emblée, l'éditeur met la barre assez haute : 1000 nouveautés, changements et améliorations. Et parmi les mots clés à retenir : Windows Phone 7, nouvel éditeur de code, un soupçon de HTML 5, 64 bits, des dizaines de nouvelles fonctions dans les langages (WLangage, Java, PHP), meilleur support de Linux, Android. WinDev se positionne plus que jamais comme l'atelier à tout faire pour le chef de projet, architecte, développeur, utilisateur avancé. Et l'outil reste fidèle à lui-même : un environnement, un langage (en français).

L'environnement de développement évolue : amélioration de l'ergonomie, interface plus épurée, nouvelle barre des documents, sélection rapide, historique sur le presse-papier, prise en compte des écrans larges (barres d'outils)... Les améliorations touchent un peu partout l'éditeur. Dans le support de la localisation, on trouve désormais 64 langues. Sur les audits, on gagne en rapidité et sur la granularité (audit statique). Pour la création des interfaces, le designer gagne quelques fonctions comme l'ancrage et le magnétisme et les erreurs de compilation liées à l'interface s'affinent aussi (par exemple sur deux binding identiques). Sur l'éditeur de code en lui-même, le développeur gagne en visibilité avec une colorisation améliorée (ex : surbrillance des variables par exemple). La fonction recherche a été étendue dans la sélection d'un bloc. L'autocomplétion gagne aussi quelques astuces bienvenues : rajout automatique de guillemets, crochets. Concernant le 64 bits, notons que le module debug autorise un debug local ou distant en mode 64, avec l'ensemble des fonctions disponibles en 32. A noter que Hyperfile se dote du support de Linux 64. Et cloud computing oblige, WinDev est capable de travailler avec les bases SQL Azure.

Si on regarde la partie WLangage, mettons l'accent sur le Mutex. Les nouvelles fonctions de gestion des Mutex (MutexCrée, MutexDébut, MutexFin, MutexDétruit) permettent de restreindre les accès à une ressource partagée (une zone mémoire partagée par exemple) afin qu'elle ne soit utilisée que par un seul thread d'une seule application. Autre nouveauté bienvenue, l'appel natif d'une API via la fonction API. Si cela était déjà possible, la v16 simplifie les choses. Dans la partie gestion de sources, là aussi, la v16 étend les possibilités : lancement automatique des tests dans une intégration continue, génération de tous les exécutables, compilation automatique à une heure donnée, compilation distribuée.

La partie mobile subit elle aussi beaucoup de modifications. La grande nouveauté est le support de Windows Phone 7. Ce qui permettra de tirer profit dès maintenant (quand la v16 sera disponible) des terminaux 7. Android est lui aussi mieux supporté : support de la boussole, gestion des réseaux sans fil, notification, etc.
Site : pcsoft.fr

agenda \\

NOVEMBRE

• Les 02 et 03 novembre, Nantes, derniers jours des **Microsoft Days** www.microsoft.com/france/.../microsoft-days.aspx

• Le 05 novembre, Paris 16e, Pavillon Royal, Première édition de **l'Eclipse Day Paris** L'événement parisien de l'année dédié à l'écosystème Eclipse. www.eclipsedayparis

• les 9 et 10 novembre 2010, Cité des Sciences de La Villette, **Forum PHP 2010.** Rasmus Lerdorf, créateur de PHP, sera l'invité d'honneur de cette édition anniversaire (15 ans) http://www.afup.org

• Le 25 novembre, Issy les Moulineaux 8h30 – 18h30, **MD Day 2010.** La journée du « Model Driven », avec l'agilité au cœur des conférences. http://www.mdday.fr

DECEMBRE

• Le 08 décembre, Microsoft France - Centre de Conférences 41 Quai du Président Roosevelt - 92130 Issy Les Moulineaux **Séminaires Business Intelligence en Libre Service avec Microsoft SQL Server 2008 R2 et PowerPivot** http://www.finelog.fr


Rejoignez la communauté de développeurs Nokia,
et créez des applications pour les millions de Smartphones Nokia à travers le monde !
Découvrez les nouveaux SDK Nokia
Et distribuez vos applications sur Ovi Store
auprès de millions d'utilisateurs Nokia.
ovi
Rendez-vous sur
www.forum.nokia.com/develop
NOKIA
Connecting People*
NOKIA
OVI STORE
CATÉGORIES
CHERCHER
COMPTE
Allociné
Actualités et info
GRATUIT
Facebook pour Nokia
Réseaux sociaux
GRATUIT
L'Equipe.fr
Sports
GRATUIT
Twitter
Réseaux sociaux
GRATUIT
Pages Jaunes
Utilitaires
GRATUIT
Elle
Utilitaires
GRATUIT
Menu Store
* Connecting People : pour relier les hommes. © 2010. Tous droits réservés. Nokia, Nokia N8 et OVI Store sont des marques déposées de Nokia Corporation. R.C.S. Paris B 493271522.
NOKIA
ovi

■ **Codendi 4.2** est disponible. Cette version inclut une réécriture totale du système de suivi de Codendi avec une nouvelle interface, la création rapide de tableaux de bord et de workflow. On dispose aussi d'un support de Git ainsi que d'un nouveau plugin pour les statistiques- administrateur.

■ Denyall complète son offre **rWeb**. Il intègre désormais la fonction Edge pour les tests de pénétration. Edge simule en amont les attaques que pourraient subir les applications testées. Les applications Web sont ainsi mises à l'épreuve par ce scan automatique, délivré en mode SaaS. L'outil utilise une gestion des faux positifs (les fausses vulnérabilités) et une approche fonctionnelle.

■ Compuware annonce **Compuware Workbench,** un nouvel environnement de développement ouvert et innovant dédié au monde mainframe. Il s'appuie entièrement sur Eclipse. Outre la reprise du code existant, l'outil permet aux développeurs de profiter de la plateforme Eclipse, de son interface, de ses fonctions de programmation.

■ **Oracle** a publié début octobre, Java 6 update 22. Cette mise à jour corrige 29 vulnérabilités dont 28 sont exploitables à distance sans le moindre processus d'identification. Cette mise à jour est donc très importante. Votre Java Updater s'en chargera si vous ne l'avez pas désactivé. Dans ce dernier cas, ne tardez pas à procéder à une mise à jour manuelle.

■ Dans la 9e édition de son rapport de sécurité semestriel, **Microsoft** a souligné l'importance des botnets (réseaux de pc zombies infectés pour être contrôlés à distance) qui ouvrent la voie au cybercrime. Dans ce cadre, Microsoft appelle à une approche collaborative et innovante afin de créer un Internet plus sûr et plus digne de confiance.

■ **Ingres** dévoile la version 10 de sa base de données. De nombreuses nouveautés sont disponibles. L'un des objectifs est de faciliter la migration d'une base propriétaire vers Ingres, notamment avec un nouveau kit de migration. La gestion de la concurrence a été renforcée. Autre fonction intéressante, le cryptage par colonne. Cela permet à un administrateur de chiffrer les données d'un champ contenant des informations sensibles. On notera aussi l'apparition de JDBC 4, de nouvelles fonctions SQL, et un travail important sur les performances (ex. : sur l'exécution des batchs).

Web

Adobe : en attendant la conférence Max 2010

Lors que vous lirez ces lignes, la conférence Adobe Max 2010 aura fermé ses portes depuis quelques jours. Nous en ferons un compte rendu complet dans le prochain numéro. En attendant, le labs foisonne de projets, de plug-ins. Commençons par le **Flash Player 64-bit** pour Windows, Linux et MacOS X. Connue sous le nom de code Square, cette version apporte un support 64-bit (système et navigateur). Et 2e nouveauté, la prise en compte, de l'accélération matérielle d'Internet Explorer 9.
On trouve aussi un complément à Dreamweaver : **Dreamweaver Widget Browser.** Il s'agit d'une application AIR pour prévisualier et configurer les widgets via une interface graphique. Il doit faciliter le travail en CSS et Javascript. On peut prévisualiser les widgets basés sur OpenAjax, ajouter des widgets provenant de Dreamweaver. Côté Catalyst CS5, on dispose d'une extension pour Photoshop CS5 : **Flash Catalyst CS5 FXG Extension for Photoshop CS5.** Il s'agit de pouvoir, via ce module, répercuter les modifications de fichiers entre Catalyst et Photoshop. Car depuis Catalyst, il est possible de modifier un fichier Photoshop. Ce module doit faciliter les transmissions des modifications. Avec la modification des clauses d'utilisation d'Apple sur la manière de créer des applications iPhone / iPad, Adobe a remis à disposition le projet **Packager for iPhone.** Il s'agit de pouvoir packager un projet Flash pour les terminaux Apple bien que Flash ne soit pas disponible sur iPhone et iPad. Ce packager fonctionne aussi bien sur Windows que MacOS. L'éditeur ne manquera pas de dévoiler d'autres solutions…
Autre projet, la disponibilité d'un plug-in Pixel **Bender** pour Photoshop CS5. Il permet d'utiliser des filtres Pixel Blender en utilisant au mieux CPU et GPU.
En revanche, nous attendons la disponibilité de la première pré-version de Digital Publishing, un greffon à InDesign dédié aux magazines numériques sur iPad (et les autres tablettes tactiles). C'est un environnement très attendu depuis les premières présentations avec le magazine Wired.
N'hésitez pas à faire un tour régulier sur http://labs.adobe.com/ !

Précision

Erratum suite à l'article Android Runtime vs Oracle du n°134

Suite à la parution le mois dernier d'un article sur la polémique Google – Oracle, Bernard Delacrétaz a envoyé la précision suivante : *« il n'y a jamais eu d'accord entre Sun et Apache qui restreindrait l'utilisation des classes du projet Harmony, de telles restrictions ne sont en effet pas possibles dans le cadre de la licence Apache V2. Sun aurait voulu imposer une telle restriction pour permettre à Harmony d'utiliser le TCK Java, ce qui est contraire à leurs promesses antérieures et n'a donc pas été accepté par la fondation Apache ».*

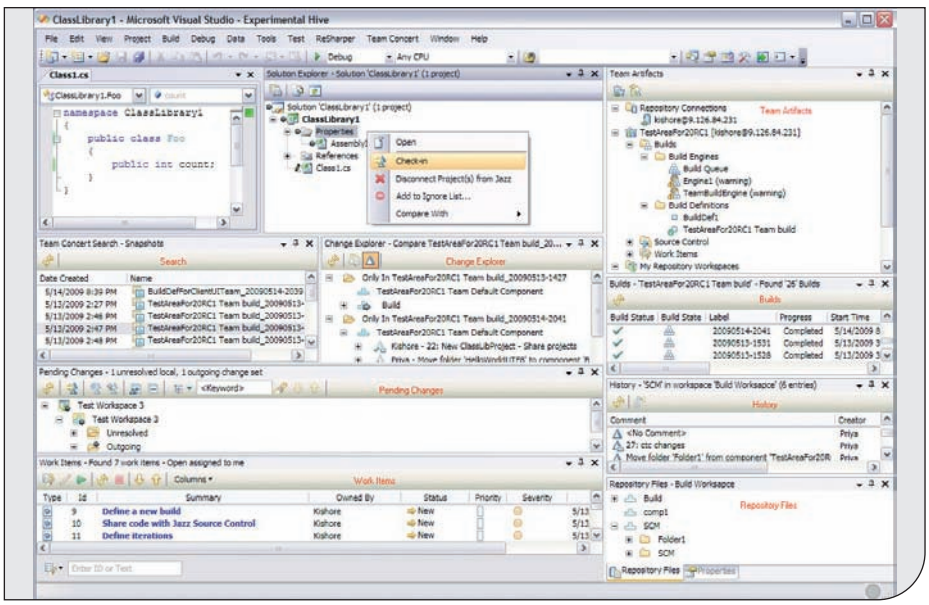
Conférence

Rational Conference Software 2010 : du cloud, du Jazz

Comme chaque année, IBM Rational organise une journée spéciale de discussions et de rencontres autour des technologies et outils Rational. Malgré les grèves dans les transports, il y avait beaucoup de monde. Et la journée fut particulièrement éclectique dans son contenu : cloud, agilité, développement, tests, modernisation des applications, gouvernance, gestion du changement. Bref, tous les thèmes de l'infrastructure moderne et des solutions de gestion de cycle de vie des applications.

Peu de nouveautés mais un soupçon de cloud…

La session plénière n'a pas livré de grandes révélations sur les futurs produits Rational, ni quant à la vision de l'éditeur sur le développement, le logiciel. Martin Nally (CTO Rational) parle d'une ère numérique comparable aux précédentes révolutions industrielles. Il a notamment abordé la question de l'amélioration des développements des projets et leur livraison. Cela passe par les bonnes pratiques, les bons outils, la collaboration (via la plateforme Jazz) ou encore les nouvelles méthodes de développement à trouver. Sans le dire, l'ALM agile fut un des arguments de Martin. Et le web a fourni un profond changement dans la manière de travailler, de relier applications et données.



Jazz fut sans doute l'argument premier de cette introduction avec l'outil maison : Rational Team Concert, l'implémentation Jazz de Rational. Il s'agit d'ouvrir une plate-forme complète de ALM et de collaboration. Véritable défi pour les équipes, les entreprises. Il est tout de même intéressant de noter que le développeur revient, chez IBM aussi, au cœur de l'application, des projets. Si la volonté affichée de Rational est toujours de passer l'ensemble de ses logiciels à Jazz, cela prendra bien plus de temps que prévu car cela représente un travail important.

Et l'idée de faire de Jazz un « eclipse du collaboratif », n'est pas abandonnée mais prendra elle aussi du temps… La seule véritable « annonce » fut quelques mots sur Rational Team Concert 3.0 qui supportera plus de processus (gestion du risque par exemple) et une ouverture vers le mainframe. Pour 2011, l'éditeur prévoit de nouvelles avancées sur la modélisation. Et sans doute aussi sur le cloud qui est déjà stratégique pour Rational. Mais la difficulté, dixit Martin, est de trouver le bon modèle, le bon marketing, la bonne transition. Cependant, l'intérêt de proposer des solutions ALM ou Jazz en cloud a du sens… D'ailleurs, il existe déjà un portage cloud de Rational Team Concert réalisé par CloudOn. Rational permet déjà de développer et de déployer sur le cloud notamment avec Rational Software Architecture.

L'Open source n'est pas une menace, il faut s'appuyer sur les standards

Martin Nally ne voit pas l'open source comme une menace même si les éditeurs s'appuient dessus, Rational le fait aussi. Il s'agit d'apporter, pour un éditeur, de la valeur, un « plus produit » et une parfaite intégration avec les outils utilisés par l'entreprise. Par contre le point important est de pouvoir être ouvert en s'appuyant sur des protocoles et des standards ouverts.

Cette approche est cruciale pour les outils ALM et permet de collaborer notamment pour l'intégration et l'usage des données. C'est pour cela que IBM pousse Open Services for Lifecycle collaboration. Il s'agit de partager, ouvrir, définir des usages, des standards, des protocoles communs. L'éditeur a invité les autres éditeurs à rejoindre l'initiative. Si Microsoft et HP ne participent pas, selon Martin, ils ne ferment pas la porte… Comme quoi, l'interopérabilité est partout. Sur Jazz, la situation est un peu différente. Les équipes souhaitent tout d'abord consolider la technologie avant de proposer un modèle ouvert comme Eclipse. Cependant, le code source de Team Concert est accessible pour développer des solutions Jazz et compatibles.

DESIGN DEVELOPPEMENT EXPERIENCE

NetAdvantage® ULTIMATE

for ASP.NET, Windows Forms, WPF, Silverlight,
WPF Data Visualization, Silverlight Data Visualization

De la conception au développement , nous avons les outils pour créer une excellente expérience utilisateur.

DECOUVREZ COMMENT UTILISER NOS CONTROLES POUR CREER CETTE "KILLER APP" SUR
INFRAGISTICS.COM/IMPRESS

Infragistics

Infragistics Ventes France N° Vert 0800 667 307 • Infragistics Europe Ventes +44 (0) 800 298 9055
Infragistics India +91 80 41518042 • @infragistics

Expression 4 : une gamme survitaminée

Microsoft Expression Studio 4 Ultimate est un ensemble d'outils de design complémentaires aux outils de développement de Microsoft Visual Studio 2010. Utilisés indépendamment par les développeurs, graphistes ou designers interactifs, ces outils permettent de bâtir des applications qui suivent les standards utilisateurs du moment en termes d'ergonomie ou de design.

Ouverts, via des fonctions d'import/export, aux outils concurrents tels que Photoshop ou Illustrator, il est facile d'intégrer et de réutiliser le travail de maquettage réalisé en amont d'un projet ou encore de collaborer dans les deux sens au développement ou au design tout au long du cycle projet. Chaque outil de la gamme Expression est dédié à son domaine de créativité artistique :

- Design pour le graphisme vectoriel,
- Web pour le design d'applications Web,
- Encoder pour la vidéo,
- Blend pour le design interactif des applications basées sur WPF ou Silverlight. [Fig.1]

Expression Studio 4 Ultimate s'installe sur Windows à partir de la version XP SP3 et utilise les versions 4 des technologies Microsoft (.NET 4, Silverlight 4, etc.). Certains produits, comme le SuperPreview, peuvent utiliser FireFox à partir de la version 3.

Dans cet article, nous présentons les fonctionnalités de la suite Ultimate (disponible à l'achat ainsi que dans les abonnements MSDN et WebSiteSpark) mais les outils de cette suite peuvent être achetés séparément et dans différentes versions. Le site de Microsoft propose un comparateur qui vous aidera à choisir la version la mieux adaptée à vos besoins.

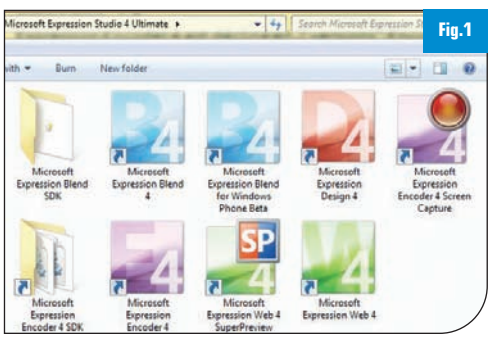


Fig.1

Fig.3

Nouveautés de Blend 4

Blend 4 supporte la création et l'édition des projets Silverlight 3 et 4, WPF 3.5 SP1 et 4.

ListBox

Si vous explorez la ListBox, vous découvrirez de nouveaux VisualStates. En effet, *l'itemContainerStyle* a évolué avec l'ajout du *VisualStateGroup LayoutStates*. Ces états permettent d'animer l'ajout et la suppression d'un élément de la liste. Vous pouvez par exemple définir un *fade in* sur l'ajout d'un élément et un *fade out* sur la suppression. Couplé à l'utilisation du *FluidMoveBehavior*, en quelques clics votre liste devient beaucoup plus vivante.

3 VisualStates composent ce VisualStateGroup :

- ***BeforeLoaded*** se déclenche avant qu'un nouvel item soit ajouté à la ListBox,
- ***AfterLoaded*** se déclenche après l'état BeforeLoaded lorsqu'un item est ajouté à la ListBox,
- ***BeforeUnloaded*** se déclenche avant qu'un item soit supprimé de la ListBox. [Fig.2]

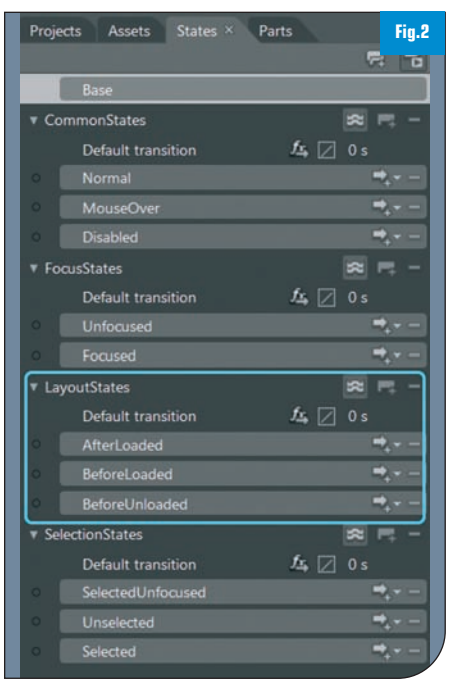


Fig.2

PathListBox

La liste se dévergonde ! Depuis les débuts de WPF puis Silverlight, nombreux sont les clients qui arrivent avec un mot en tête : Carrousel. Blend était assez limité lorsqu'il s'agissait de créer une liste un peu exotique dans sa forme. L'arrivée de la PathListBox et d'un lot de behaviors associés marque un tournant dans la création des listes, l'intégrateur gagne en autonomie et le développeur peut se concentrer sur des questions qui ne touchent pas à la mise en forme. A savoir : Une série de behaviors est disponible sous le nom de ***PathListBoxUtils*** afin de pouvoir, entre autres, gérer le défilement de la liste sur le site http://www.expression-blend.codeplex.com.

La PathListBox est donc une ListBox qui affiche ses éléments en suivant un ou plusieurs UIElements (l'utilisation de tracés vectoriels étant la plus indiquée).

Vous prenez le composant PathListBox, un tracé vectoriel (ou tout autre élément de votre interface) et une multitude de paramètres vous permettront de gérer l'affichage de votre liste : le nombre d'éléments visibles, leur positionnement par rapport au(x) tracé(s), le positionnement des éléments les uns par rapport aux autres… [Fig.3]

Le point de départ de la PathListBox peut être le tracé, en passant par le menu Object > Path > Make Layout Path ou la PathListBox elle-même en passant par le panneau des propriétés LayoutPaths et en désignant le(s) objet(s) UIElement souhaité(s).

Shapes

Blend 4 met à votre disposition une large série de nouvelles formes géométriques. Les outils de combinaison permettaient d'ob-

tenir toutes les formes possibles par addition ou soustraction des formes de base (ellipse et rectangle), mais l'atout indéniable de ces nouvelles formes réside dans leur paramétrage. Les formes Hexagon, Pentagon, Star et Triangle par exemple sont tous des Polygones (RegularPolygon) avec un nombre de points différent. Les formes Callout sont particulières puisqu'elles ont une propriété Content [Fig.4].

Pixel Shader effects

Jusqu'à la version 3, vous aviez la possibilité d'ajouter un effet de flou (blur effect) ou d'ombre portée (drop shadow effect) à un élément. Cette version 4 n'apporte pas moins de 13 effets supplémentaires.

Les effets colorimétriques

Les cinq premiers effets « Bloom » sont des paramétrages différents d'un seul et même effet qui gère plusieurs paramètres affectant la couleur.

L'effet **Bloom-Desaturated** permet de désaturer votre image, de réduire la pureté de la couleur. L'effet est un peu plus fort qu'un effet de désaturation, pour lequel Les zones qui sont en niveau de gris ne devraient pas être affectées. Les zones claires sont accentuées et tirent vers le blanc tandis que les zones plus sombres ne sont pas modifiées – cela amène une lumière intéressante à vos images.

L'effet **Bloom-Gamma** permet de renforcer le contraste de vos objets.

L'effet **Bloom-Saturated** : cet effet permet de retrouver la pureté des couleurs, elles sont plus intenses.

L'effet **Bloom-Soft** permet d'adoucir votre image : les contrastes sont moins élevés, la luminosité plus régulière.

L'effet **Bloom-Subtle** est assez proche de l'effet Soft, avec une saturation moins élevée [Fig.5].

L'effet **ColorTone** vous permet de changer la teinte de votre image, en spécifiant une couleur pour les tons sombres et une couleur pour les tons foncés, en laissant ou non apparaître la teinte d'origine, avec une saturation plus ou moins forte.

L'effet **Monochrome** est idéal pour faire du Sépia, elle redéfinit la teinte de l'objet, sans s'embarrasser de ses couleurs originales [Fig.6].

Les effets d'accentuation

L'effet **Emboss** est un effet d'estampage, il donne du relief à vos images en marquant les contours avec des tons très foncés pour les creux et très clairs pour les bosses, dans une teinte que vous pouvez définir.

L'effet **Sharpen** renforce vos images en appuyant les contours [Fig.7].

Les effets de déformation

L'effet **Magnify**, que l'on appelle aussi souvent le fisheye, permet d'avoir une zone globuleuse, déformée pour avoir au centre une vue zoomée de l'objet.

L'effet **Pixelate** vous sortira du monde vectoriel pour retourner dans un univers pixellisé.

L'effet **Ripple** déforme votre image avec un effet de vagues.

L'effet **Swirl** affecte une rotation en gardant le point d'origine fixe [Fig.8].

N'oubliez pas qu'il est possible d'animer tous ces effets via des VisualStates ou des Storyboard ! (En faisant attention de prendre en compte les éventuelles baisses de performances associées.)

Transition Effects

Si vous connaissez Blend 3, la notion de VisualStates vous est familière. Vous savez qu'il est possible d'animer le passage d'un état à l'autre en définissant une durée et un type d'animation. Dans Blend 4, vous pouvez ajouter un « transition effect » - vous pouvez spécifier une transition en parallèle d'une animation « easing function ». Les effets de transition se rapprochent des transitions que vous pouvez appliquer dans Powerpoint, ou dans les logiciels de montage vidéo. Soyez subtils, ces effets sont à utiliser avec parcimonie !

Easing Function pour WPF

WPF4 intègre les animations « Easing Function », de la même manière qu'elles étaient disponibles pour Silverlight 3.

Les ressources en Design-Time

Pour certains projets, les dictionnaires de ressources sont chargés dynamiquement, à l'exécution du projet. Lorsque vous travaillez sur ce type de projets sous Blend, vous obtenez une interface différente du résultat final et des messages d'erreurs concernant les ressources introuvables. Difficile de travailler dans ces conditions. Pour tester cette nouvelle fonctionnalité de Blend 4, le chemin peut paraître un peu tortueux. En effet vous ne trouverez rien au niveau des menus vous permettant de définir un dictionnaire de ressource à utiliser en design-time. Blend vous le proposera de lui-même si certaines conditions sont réunies : Votre projet doit contenir au moins un dictionnaire de ressource et une ressource introuvable. C'est uniquement au chargement de l'application qui remplit ces conditions que Blend affichera la boîte de dialogue suivante [Fig.9]. Cela aura pour effet d'ajouter un nouveau dictionnaire de ressources dans le dossier Properties de votre projet (DesignTimeResources.xaml) qui référence tous les dictionnaires de ressources à utiliser en design-time. Vous pouvez alors travailler normalement sur votre projet.

Windows Phone 7

Blend 4 SP1 intègre la gestion des applications Silverlight pour Windows Phone 7

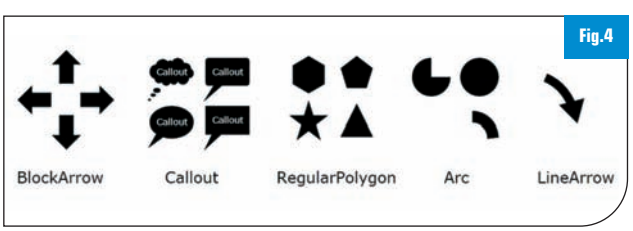


Fig.4

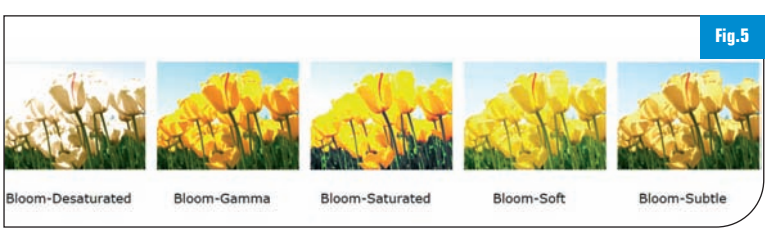


Fig.5

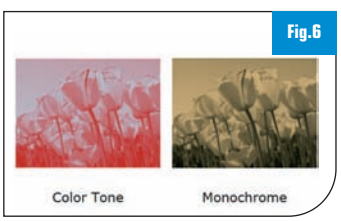


Fig.6

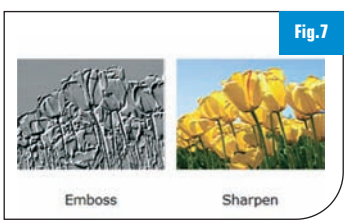


Fig.7

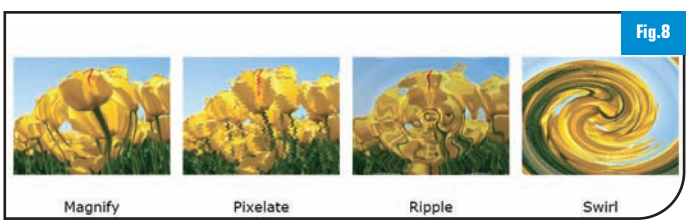


Fig.8

(WP7), le nouvel OS de Microsoft pour mobile [Fig.10].
De nouveaux contrôles accompagnent WP7. Les contrôles Pivot et Panorama sont les contrôles principaux qui permettent de mettre en place une navigation « METRO ». Les contrôles disponibles incluent notamment le contrôle Bing Map. A noter également que tous les composants bénéficient de styles appropriés à Windows Phone 7, c'est-à-dire qu'ils intègrent des marges importantes afin d'éviter le chevauchement des éléments. Etant dans un environnement tactile, il est nécessaire de veiller à ce que les éléments interactifs soient accessibles facilement.

Blend et Pattern M-V-VM

Expression Blend dans sa version 4 apporte son lot de fonctionnalités supplémentaires, mais aussi des améliorations importantes sur la stratégie de développement d'une application en Silverlight. Le pattern MVVM (Model/View/View-Model) bien connu des développeurs dans ce domaine apporte une réelle plus-value en termes de logique et collaboration entre un développeur et un designer. Cette partie ne prétend pas couvrir de façon exhaustive toutes les nouveautés et possibilités d'expression Blend 4 dans le cadre d'un projet qui implémente le pattern MVVM, mais permet plutôt de toucher du doigt l'incroyable efficacité de cet outil au travers de deux exemples simples et concrets.

Exemple 1 : Création d'un annuaire

Nous allons créer dans cette partie un annuaire qui propose une liste de personnes, et détaille les informations sur cette personne lorsqu'une personne de la liste est sélectionnée. Pour cela nous allons utiliser intégralement Expression Blend avec des données bouchonnées au format XML sans une seule ligne de code.

Création d'un sample DataSource

Nous utilisons le fichier XML suivant pour créer notre « SampleDataSource » :

```
<?xml version="1.0" encoding="utf-8" ?>
<Persons>
  <Person
    FirstName="Lincoln"
    LastName="Abraham"
    Birth="12/02/1809"
    Adress="1 wall street"
    ZipCode="77005"
    Phone="02857548" />
  <Person
    FirstName="Jean"
    LastName="de la Fontaine"
    Birth="13/04/1965"
    Adress="12 avenue du corbeau"
    ZipCode="57459"
    Phone="67025804"/>
  [...]
</Persons>
```

Pour créer notre **DataSource** à partir de ce fichier :

1. Sélectionner le type de SampleDataSource à créer : [Fig.11].

2. Donner un nom à la DataSource et un chemin d'accès au fichier XML : [Fig.12].

3. La DataSource apparaît dans l'onglet Data avec la description de son arborescence : [Fig.13].

Création d'une liste d'index

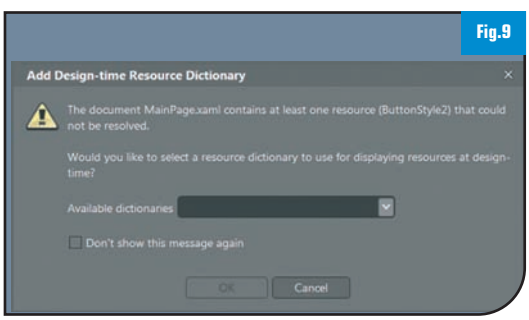
Maintenant que la DataSource a été importée, il ne reste plus qu'à exploiter les données pour créer notre page. Cette étape est très simple puisqu'il suffit de sélectionner les propriétés de la personne à afficher et de les glisser-déposer dans la fenêtre de design : [Fig.14].
Une Listbox contenant les noms des personnes est automatiquement créée et connectée à la DataSource. Il est ensuite possible de mettre en forme la liste par l'édition de son template.

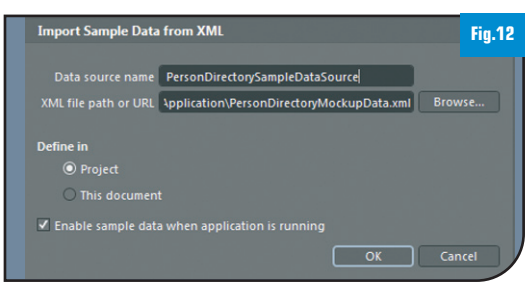
Création du détail de la liste

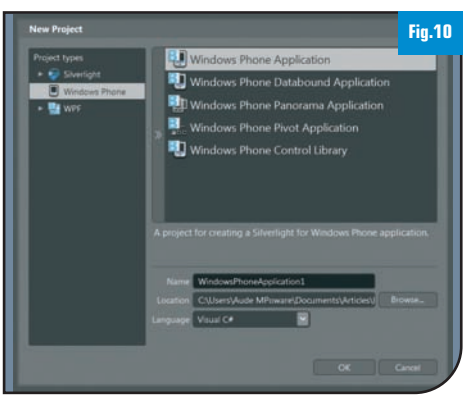
La première chose à faire est de passer en mode « Details » avec le bouton situé en haut de l'onglet « Data » [Fig.15].
Comme pour l'étape précédente, il ne reste plus qu'à faire un glisser-déposer des propriétés à afficher dans la zone de design [Fig.16]. Puis à mettre en forme la zone créée en ajustant les marges [Fig.17].
Il est aussi possible de redéfinir l'ordre et les intitulés des éléments du détail en changeant la valeur des TextBlocks des intitulés, et en faisant à nouveau un glisser-déposer de chaque propriété du DataSource vers le TextBlock correspondant [Fig.18].
Nous venons en un temps record, quelques clics et un minimum d'effort de créer une page aussi fréquente que fastidieuse à implémenter proprement !

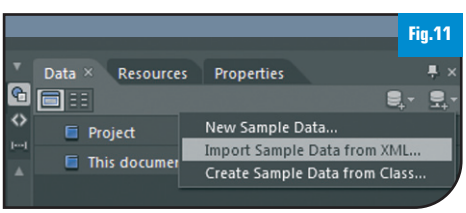
Exemple 2 : Création d'une page de recherche simple

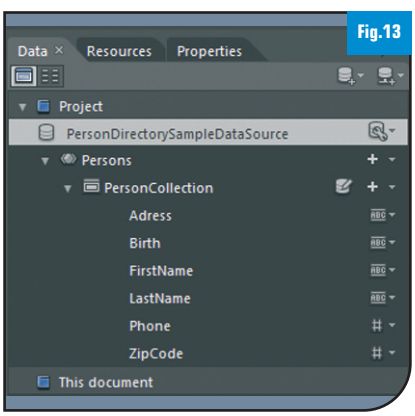
Parmi les grandes nouveautés de Silverlight et Expression Blend 4, on trouve notamment

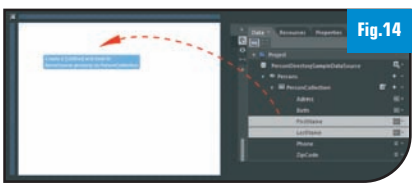












le support des commandes par les contrôles Silverlight (classes implémentant ICommand) et des améliorations au niveau des Behaviors et des TriggerActions, ouvrant ainsi de nouvelles possibilités à l'implémentation du pattern M-V-VM. Pour cette partie, nous supposons que nous disposons déjà d'une couche View-Model qui expose une liste de personnes (à filtrer) et une commande qui prend en paramètre un pattern de nom et qui rafraîchit la liste des personnes filtrées avec ce pattern. Le ViewModel est instancié sous forme de DataSource dans la page XAML principale et la liste résultante est connectée au DataSource [Fig.19].

Invocation de la commande par un TriggerAction

Nous allons lancer une nouvelle recherche à chaque fois que l'utilisateur modifiera la Textbox de saisie (cf. schéma de connexion : «SearchPatternTextBox »). Pour cela, nous allons utiliser un TriggerAction que Blend met à disposition dans ses librairies : « **InvokeCommandAction** » [Fig.20].

Configuré comme ceci : [Fig.21].

(Préférez l'évènement « KeyUp » plutôt que « TextChanged » pour cet exemple, car l'évènement TextChanged se déclenche avant que la modification dans la Textbox ne soit effective, la recherche fonctionnerait alors avec une étape de retard…) Avec la propriété « CommandParameter » liée à la propriété « Text » du Textbox contenant le pattern de recherche : [Fig.22]. A ce niveau, l'application peut déjà être testée : [Fig.23].

Ajout d'une condition à la recherche

A présent, nous voulons modifier cette page de façon à ce que la recherche ne se lance pas avant qu'on ait tapé au moins 2 caractères. Pour cela, nous allons utiliser les Conditions qui se trouvent dans les propriétés du TriggerAction comme ceci : [Fig.24].

En lançant à nouveau l'application, vous pouvez constater que la recherche ne se lance pas avant que 2 caractères soient tapés dans le champ de saisie. Ces 2 exemples nous ont permis d'illustrer de façon concrète la mise en place du M-V-VM avec Expression Blend 4. Cet ensemble de fonctionnalités permet aussi bien au designer qu'au développeur de mettre en place une conception de l'interface selon les bonnes pratiques tout en gagnant en productivité.

Web 4

Expression Web 4 apporte quelques nouvelles fonctionnalités, notamment pour les tests, l'optimisation pour les moteurs de recherche et la création d'add-ins.

SuperPreview

SuperPreview est une fonctionnalité qui existait déjà dans la version 3 mais qui s'enrichit dans la version 4 par la possibilité d'utiliser de nouveaux moteurs de tests de compatibilité de navigateurs en ligne. Pour cela, au lancement de SuperPreview, cliquer sur le lien « Sign Up » pour vous enregistrer sur le service en ligne en utilisant votre adresse email [Fig.25].

Il est nécessaire de valider votre email, puis d'activer l'accès au service en ligne. Vous avez maintenant accès aux navigateurs en ligne, comme par exemple IE9 Beta ou Safari pour Mac [Fig.26].

Exemple de comparaison entre IE9 Beta et Safari 5 pour MAC [Fig.27].

Outils SEO

Ces outils vous permettront d'optimiser l'indexation de votre site pour améliorer vos

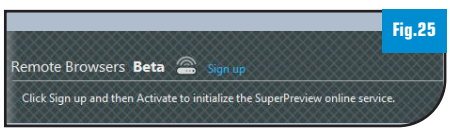


Fig.25

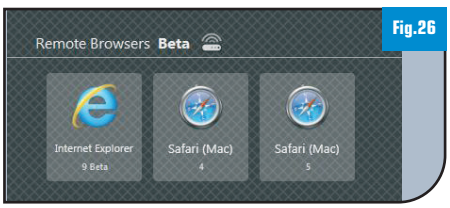


Fig.26

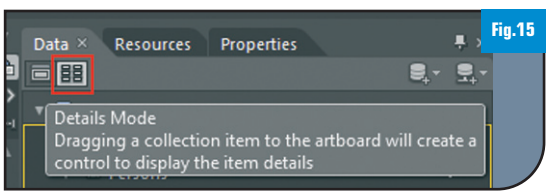


Fig.15

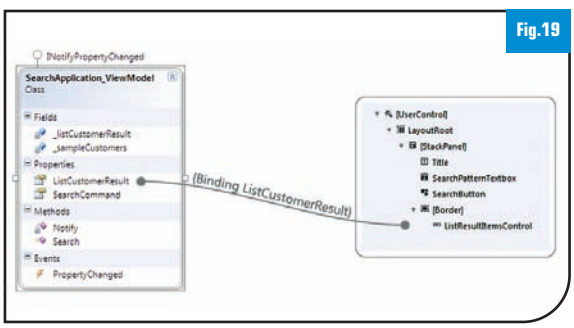


Fig.19

Fig.20

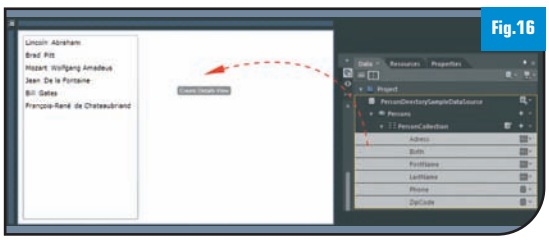


Fig.16

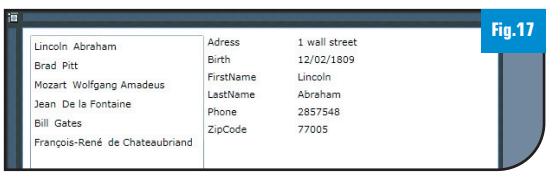


Fig.17

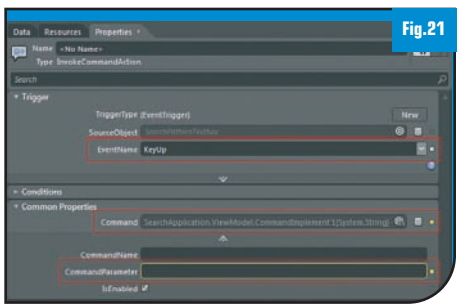


Fig.21

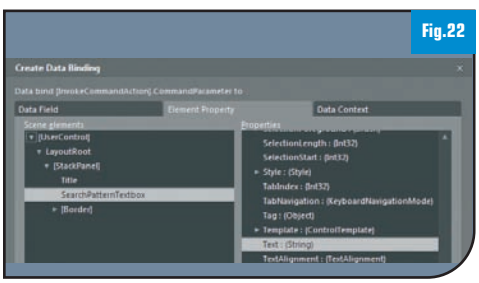


Fig.22

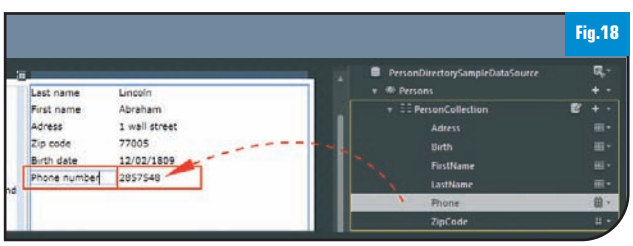


Fig.18

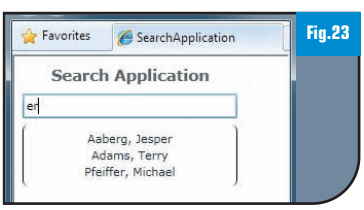


Fig.23

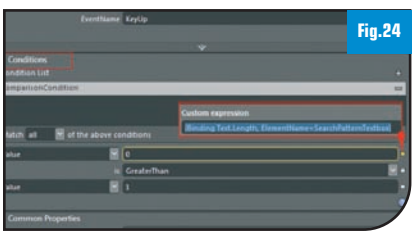


Fig.24

scores de recherche depuis la plupart des moteurs de recherche du Web (Bing, Google…). Le vérificateur SEO de Web 4 utilise des techniques similaires à ces moteurs de recherche pour valider l'indexabilité de votre site (Menu Tools > SEO Checker) [Fig.28].

Extensibilité par les Add-Ins

Il est maintenant possible d'étendre Expression Web 4 avec des add-ins spécifiques développés en HTML, JavaScript et CSS. En utilisant l'outil « Add-In Builder » téléchargeable en ligne (http://gallery.expression.microsoft.com/en-us/AddinBuilder), la création d'add-in devient très simple [Fig.29].

L'Add-In Builder permet de décrire un add-in, ajouter des Panels et des boîtes de dialogues, ainsi que de spécifier des assemblies à embarquer dans l'add-in [Fig.30 et 31].

Il ne reste plus qu'à agrémenter votre add-in pour construire votre outil. Si celui-ci est bien fait, vous pourrez le poster sur la galerie des add-ins (http://gallery.expression.microsoft.com). Un template de projet d'Add-In pour Expression Web est aussi disponible en téléchargement pour Visual Studio 2010 (http://gallery.expression.microsoft.com/fr-fr/xWeb4SDK).

Exemple d'Add-In pour insérer une carte Bing Map dans une page Web sans ligne de code :

1. Choix d'une carte via l'Add-In [Fig.32].
2. Insertion de la carte dans la page Web (l'add-in ajoute automatiquement les références à jQuery) [Fig.33].
3. Test dans un navigateur [Fig.34].

Autres nouveautés de Web 4

Les autres nouveautés de Web 4 :

- La surface de Design utilise les bonnes pratiques HTML, CSS…
- Les options de publication s'enrichissent notamment avec la possibilité de ne publier que les fichiers modifiés par exemple

Cette version ravira à coup sûr le développeur Web par les capacités à intégrer des extensions très productives, à réaliser des tests sur des navigateurs non Windows ou encore à améliorer la qualité des sites Web réalisés.

Encoder 4

Expression Encoder 4 est décliné en 2 versions : Encoder 4 (en téléchargement gratuit) et Encoder 4 Pro. La version Pro propose en supplément les formats d'entrée MPEG-2, AVCHD et Dolby Digital et les formats de sortie H.264, AAC-LC et IIS Smooth Streaming. Les 2 versions proposent la capture d'écran en direct avec une limitation à des segments de 10 minutes pour la version gratuite.

Encoder Screen Capture permet de capturer en vidéo votre écran, indispensable pour les démos et autres web casts. Plusieurs options sont disponibles notamment la capture ou non du pointeur souris, la sortie son, l'incrustation du flux web cam dans la capture écran, … [Fig.35].

Parmi les nouvelles fonctionnalités d'Expression Encoder, vous pourrez découvrir :

- Des espaces de travail adaptés à ce que vous voulez faire (transcodage, projet Silverlight ou Live Broadcasting), [Fig.36].
- Protection des contenus avec DRM,
- L'import multi-langues de fichiers de légendes DFXP, SAMI, SRT, SUB et LRC et l'export en DFXP.

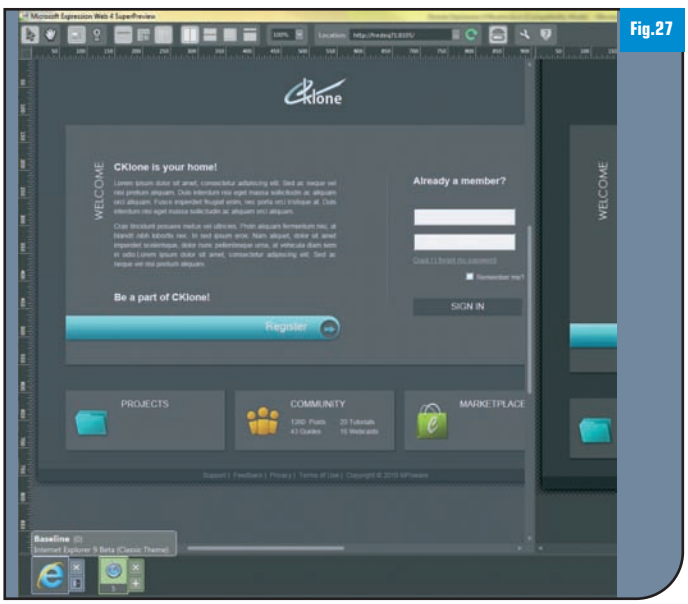


Fig.27

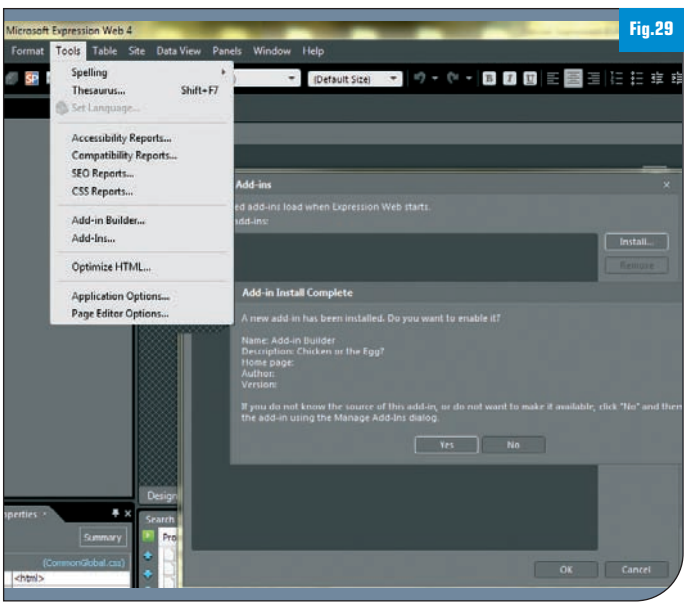


Fig.29

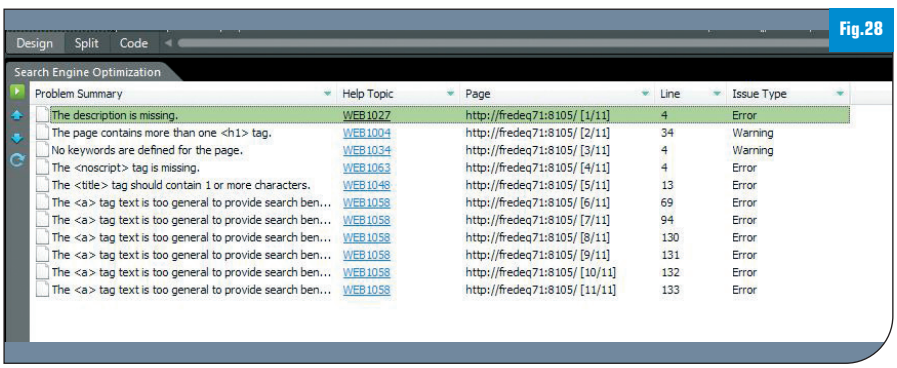


Fig.28

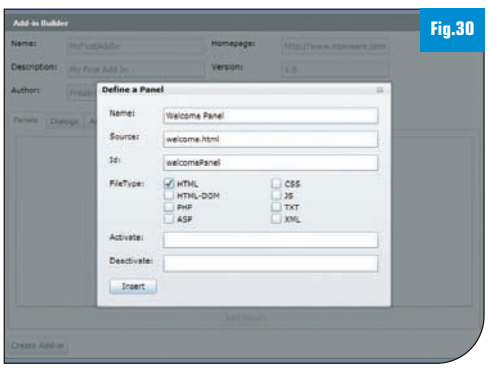


Fig.30

Design 4

Expression Design 4 est l'outil indispensable pour les graphistes qui participent aux projets WPF ou Silverlight. Cette nouvelle version s'enrichit de l'import des fichiers Windows Metafile et Enhanced Metafile, formats en général dédiés aux applications de la suite Office. Dans cette nouvelle version, le graphiste pourra aussi sauvegarder des espaces de travail qu'il aura personnalisés afin d'améliorer sa productivité [**Fig.37**].

Conclusion

Avec cette nouvelle version d'Expression Studio 4 sortie avant l'été 2010, Microsoft continue d'enrichir ses outils de design. Un gros focus est donné à Expression Blend 4 et le support de WPF4, Silverlight 4 et Windows Phone 7, le tout en restant compatible avec Visual Studio 2010. Expression Web 4 se professionnalise et offre un nouveau mode Super Preview avec des services en ligne pour tester des navigateurs non Windows. Quant à Expression Design 4 et Encoder 4, les nouvelles fonctionnalités leur permettent de s'intégrer davantage dans les projets RIA / RDA. Une nouvelle fois, MICROSOFT confirme sa stratégie dans le RIA / RDA et renforce la capacité à faire collaborer les designers et les développeurs au sein d'un même projet !

Références

1. Le site de MICROSOFT pour Expression Studio : http://www.microsoft.com/expression/

2. La galerie en ligne des outils tiers pour Expression Studio : http://gallery.expression.microsoft.com

3. Les exemples avec Expression Blend 4 sur CodePlex : http://expressionblend.codeplex.com/

4. Blog de l'équipe Expression Blend & Design : http://blogs.msdn.com/b/expression/

5. Blog de l'équipe Expression Web : http://blogs.msdn.com/b/xweb/

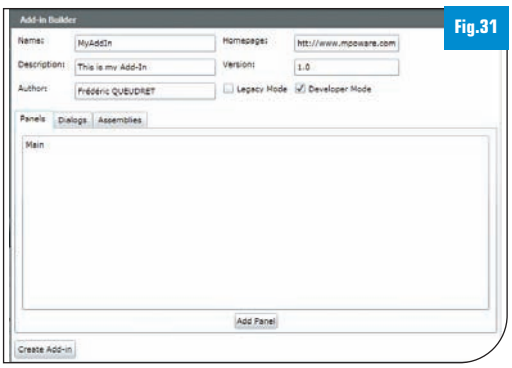
6. Blog de l'équipe Expression Encoder : http://blogs.msdn.com/b/expressionencoder/

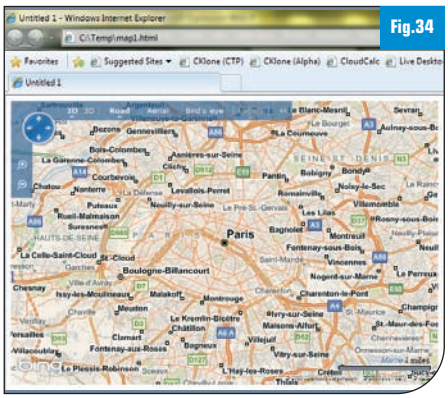
■ Frédéric **Queudret**
MVP Client Application & CTO – MPoware.

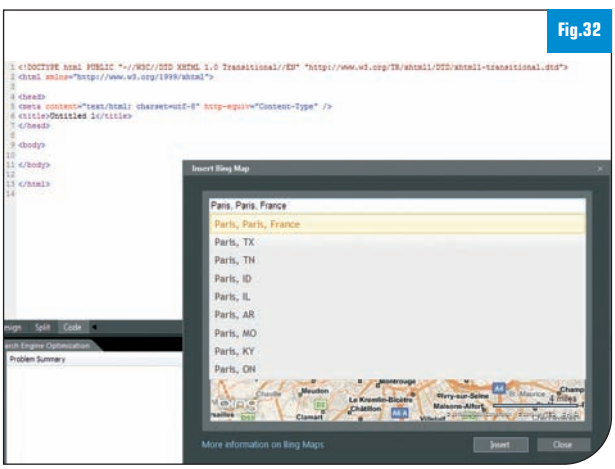
■ Aude **Mousset**
Directrice Artistique MPoware.

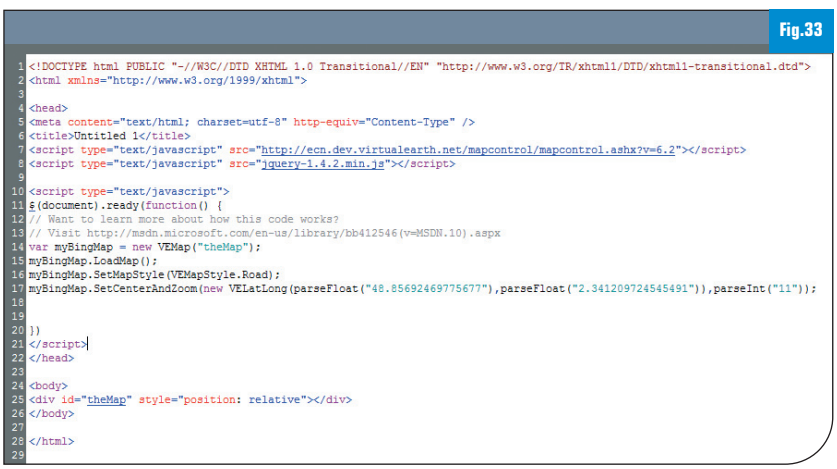
■ Charles **Hetier**
Ingénieur R&D MPoware.

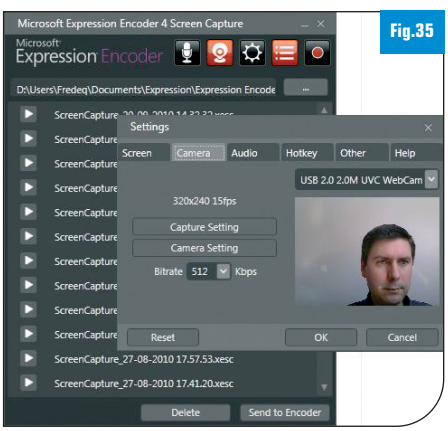
MPoware est une société française d'édition de logiciels et de prestations de services sur les technologies .NET.
http://www.mpoware.com/

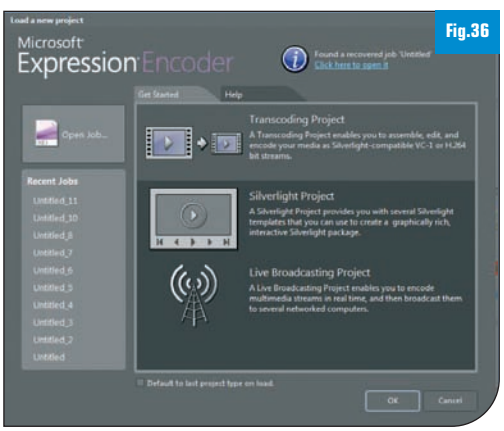


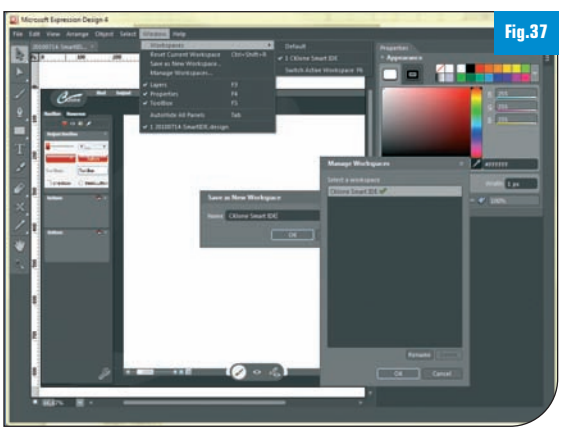












Modélisation et développement : à chacun son modèle

1re partie

Dans Programmez ! nous vous proposons régulièrement des dossiers sur la modélisation, l'UML, le pilotage par modèle. Si longtemps, ces techniques furent peu regardées par le développeur, aujourd'hui, elles se révèlent précieuses. Elles permettent tout d'abord de réduire le code technique à produire mais aussi de mieux structurer l'application et donc son code.

Si l'intérêt de la modélisation est réel, quelle méthode utiliser ? Jusqu'ou aller dans la modélisation ? Il n'est pas simple de répondre à ces questions car cela va dépendre du projet, de l'entreprise, de la proximité du développeur avec ces méthodes. DSL, UML, MD, font grosso modo la même chose, tout du moins dans leur objectif, ces techniques diffèrent dans les outils, l'approche et le contexte d'utilisation. Ce dossier se conclura le mois prochain.

L'avenir d'UML : l'approche Domain Specific Modeler

A ses débuts, le MDA pensait pouvoir utiliser des modeleurs "sur étagère". Aujourd'hui, l'approche Domain Specific Modeler (DSM) bouscule l'hégémonie des modeleurs UML.

Pour expliquer cette tendance, imaginons que vous vouliez peler une pomme. Comme souvent, vous faites confiance à votre couteau suisse. Maintenant, si vous souhaitez peler un millier de pommes, mieux vaut utiliser un épluche-légumes pour être plus efficace. Si l'on fait une analogie en informatique, tout le monde s'accorde à dire qu'il vaut mieux utiliser SQL pour requêter une base de données ou Makefile pour compiler un programme plutôt que Java ou C#. Côté modélisation, UML rentre également dans la catégorie des "couteaux suisses".

Il a un certain nombre de défauts inhérents à cette catégorie : généricité et faible adaptabilité qui l'éloigne trop des préoccupations métier selon les termes mêmes d'un des auteurs de la norme.

UML ou DSL

L'adaptation au contexte métier est possible à travers deux principaux mécanismes : les profils UML qui offrent un mécanisme d'extension de la norme pour introduire des spécialisations et les Domain Specific Language (DSL) qui consistent à définir la liste des concepts métier, autrement nommée le « métamodèle ». Dans le jargon MDA, c'est le métamodèle qui décrit un vocabulaire dédié à une problématique donnée.

La construction de Domain Specific Modeler au dessus de ces mécanismes suit le même schéma :

- DSM over UML, utilisé pour créer des diagrammes personnalisés stockés via un profil UML;
- DSM over DSL utilisé pour projeter à l'aide de points de vue les informations exprimées dans un langage spécifique.

Dans les deux cas, l'utilisateur choisit librement quelle représentation utiliser indépendamment du mode de stockage. Les grosses différences entre les deux approches sont qu'avec un DSL, le modèle est plus simple à manipuler car il ne dépend pas de tous les concepts UML, et il est valide par construction car l'utilisateur ne peut pas modéliser quelque chose de non conforme au langage.

Ne vous êtes-vous jamais dit que malgré tous ces outils de modélisation, PowerPoint est plus simple pour expliquer le fonctionnement d'une application ou une architecture ? Ou, pourquoi me force-t-on à connaître le principe de classe UML pour schématiser simplement des écrans ? Un Domain Specific Modeler laisse à l'utilisateur la liberté de choix de la représentation graphique : diagramme à base d'icônes ou des formes, tableau,

voire même textuelle [Fig.1]. En résumé, un DSM rend simples et naturelles à utiliser la visualisation et l'édition de données ayant un sens sémantique.

Eclipse Modeling

Le projet Eclipse Modeling propose des outils pour créer des DSM. Les plus utilisés sont Graphical Modeling Framework (GMF) pour les représentations graphiques et Xtext pour les représentations textuelles. GMF permet de décrire la représentation graphique de chaque concept et construit un DSM par génération. Les modeleurs ainsi produits possèdent une très bonne ergonomie et une standardisation du format de stockage des informations graphiques. Mais GMF demeure complexe à appréhender et requiert un niveau d'expertise élevé pour construire des modeleurs de qualité industrielle. Xtext quant à lui propose de définir une grammaire et construit le DSL textuel également par génération. L'éditeur produit est conforme à l'état de l'art des éditeurs Eclipse (c'est-à-dire très bon) : coloration syntaxique, quick fix, content assist, folding, formatage, outline [Fig.2 et 3].

Comment fabriquer son modeleur ?

Des modeleurs de qualité industrielle basés sur GMF sont déjà disponibles : Bonita, Fractal, SCA, etc. Mais la difficulté de fabriquer de tels outils demande un effort important. Afin de simplifier cette étape, un studio de construction de DSM basé sur GMF a été mis au point au sein de l'atelier Obeo Designer. L'approche choisie consiste à fournir un environnement de paramétrage des DSM et à l'interpréter pour produire à la volée les vues sous forme de diagrammes ou tableaux. L'utilisateur décrit pour chaque concept :

- sa forme (carré, rond, conteneurs, ...),
- son style (taille ou couleur qui changent en fonction d'un critère, libellé dynamique, ..),
- son affichage au sein de calques (comme dans Gimp ou Photoshop),
- les barres d'outils qui la manipulent,
- son stockage dans le modèle

Il est ainsi possible de construire des DSM sans même connaître GMF ou le développement de plug-ins Eclipse.

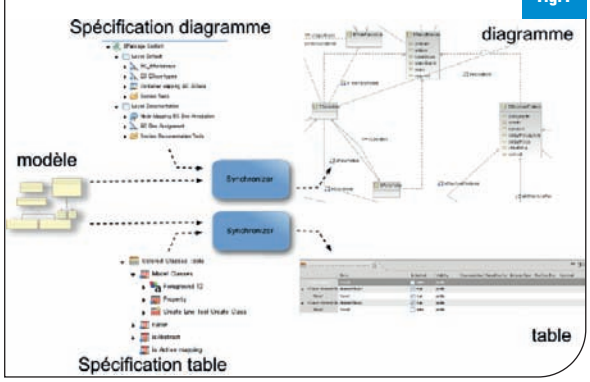
Points de vue

Une fois que plusieurs DSM sont mis au point, il est souhaitable de pouvoir les regrouper selon une préoccupation donnée : par type d'utilisateur ou par problématique. On parle ainsi de « points de vue », c'est à dire selon un ensemble de représentations ciblées qui guide l'utilisateur dans sa démarche de modélisation. Il pourra ainsi appréhender facilement la complexité du système modélisé et celle du langage utilisé, par des représentations plus simples et plus précises.
Le principe des points de vue peut être utilisé dans différents contextes. Par exemple :

- **Modélisation de systèmes d'information.** Chaque couche est représentée par un point de vue, et chaque point de vue regroupe plusieurs diagrammes : couche métier (entités, persistance, accès aux données), couche composant (traitements, services, urbanisation) et couche présentation (cinématique de navigation, IHM).
- **Cartographie.** L'approche par point de vue est particulièrement intéressante dans le contexte de rétro-ingénierie de code source existant. Ce type d'usage s'applique sur des modèles comportant énormément d'informations. L'approche par point de vue offre un gain important pour classifier ces différentes informations par besoins d'analyse : analyse de dépendances, qualité, zoom à travers différents niveaux d'abstraction, ...
- **Modélisation de systèmes embarqués.** Un système embarqué peut être modélisé selon trois points de vue qui sont software, temporel et hardware.

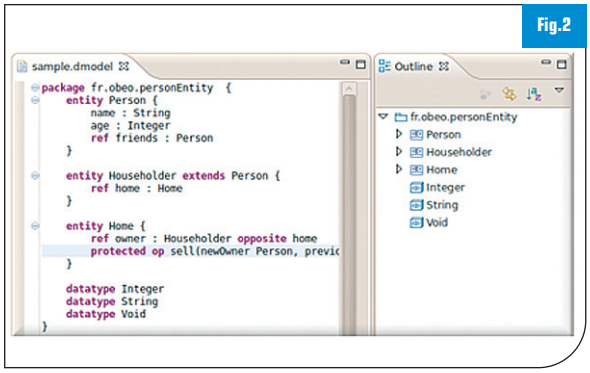
Alors, quid de l'avenir de UML ? Une recommandation simple serait d'utiliser UML pour ce qu'il a été prévu. Si votre sémantique s'en éloigne fortement, évitez la profusion des profils et stéréotypes, et privilégiez un DSL. Ainsi, les DSL ne s'opposent pas à UML et se complètent parfaitement.
Grâce à l'usage de points de vue, il est de plus très facile de retrouver les représentations recommandées par UML (séquences, états/transitions, composants, ...) pour les appliquer à des DSL. Face à l'abondance des concepts et des notations que peut proposer un langage de modélisation, cet article a pour objectif d'initier à l'utilisation de l'approche par points de vue. L'idée de cette approche est de séparer les préoccupations par domaine et d'augmenter ainsi le niveau d'expressivité. L'intérêt d'une telle solution réside en sa simplicité de prise en main afin de rendre accessible à tous, la mise en œuvre de DSM.

Fig.1



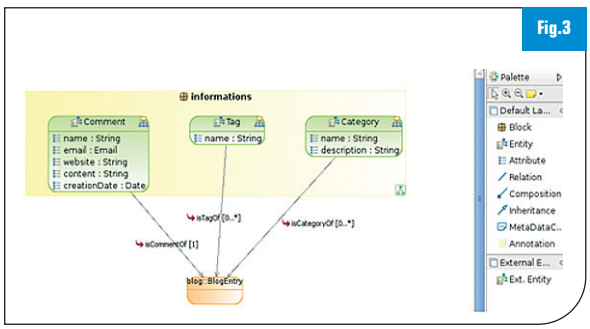
Un même modèle représenté de différentes manières

Fig.2



Exemple de DSL textuel avec xText

Fig.3



Exemple de DSM avec GMF

■ Mikaël **Barbero**
Consultant Eclipse Modeling pour la société Obeo, spécialiste de la migration et la modernisation de systèmes d'information.

Visual Studio 2010 et la modélisation

Avec Visual Studio 2010 les portes de l'architecture et de la modélisation se sont ouvertes à un large public. L'outil n'a pas prétention de rivaliser avec les cadors du genre, mais de démocratiser ces disciplines qui sont aujourd'hui très importantes et d'apporter une valeur ajoutée indéniable à l'ensemble de l'équipe.

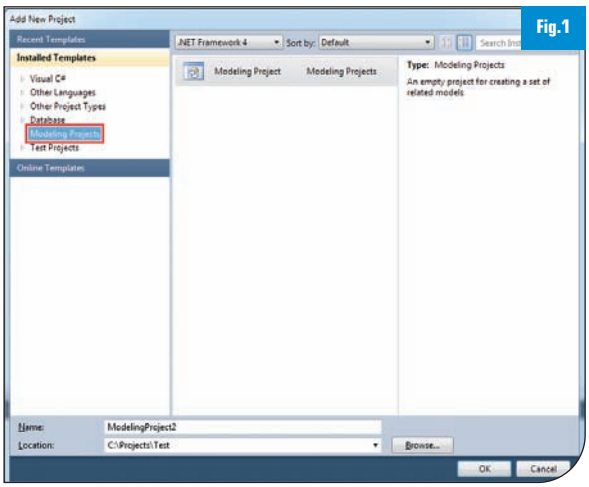


Fig.1

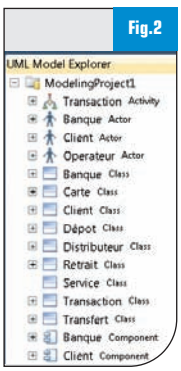


Fig.2

La modélisation est pour beaucoup de développeurs une étape obscure voire inutile. Cette réputation est malheureusement en partie fondée : de nombreux outils de modélisation UML existent sur le marché, mais quasiment tous ont pour effet d'isoler la phase de modélisation du reste de la réalisation d'un projet. Cela est dû au fait que ces logiciels n'offrent pas d'intégration (du moins assez efficace) avec les disciplines adjacentes comme le code ou les tests. Microsoft joue de nouveau les cartes de l'intégration et du pragmatisme avec la brique Modélisation de Visual Studio 2010. Les experts UML ne trouveront certainement pas cette version à la hauteur (malgré la sortie du Feature Pack qui comble bon nombre de lacunes), mais pour ceux qui approchent cette discipline comme un outil permettant de travailler mieux et plus vite il n'y a pas de doute : Microsoft est sur la bonne voie !

Projet de type modélisation

Pour la modélisation, Visual Studio 2010 prévoit un nouveau type de projet spécifique qui regroupe tous les diagrammes. Ce type de projet est disponible dans la catégorie « Modeling Projects » de la fenêtre de dialogue « Add New Project » [Fig.1]. Un projet est nécessaire afin de pouvoir accéder aux différents types de diagrammes, chaque diagramme est constitué d'un ou plusieurs fichiers. En procédant de la sorte, Microsoft vous garantit le même comportement par rapport aux projets classiques (WinForms, Class Library, …) vis-à-vis de votre solution et de la gestion de configuration.

Deux points d'entrés sont alors disponibles : le premier vous est familier puisqu'il s'agit de l'explorateur de solution. Celui-ci vous permet de créer et d'organiser les différents diagrammes comme vous avez l'habitude de le faire et ainsi d'intégrer la modélisation directement au même niveau que les autres projets de la solution : les développeurs ont enfin un accès simple à ces données. Le second est une fenêtre inédite intitulée « Explorateur de modèle UML » qui centralise,comme son nom l'indique, l'ensemble des éléments de type UML.

Chaque élément créé dans un diagramme est référencé dans l'explorateur de modèle UML, cela permet de pouvoir constituer une bibliothèque d'éléments ayant un sens fonctionnel qui peuvent être par la suite réutilisés au travers des différents diagrammes [Fig.2]. Une petite astuce : la centralisation de ces éléments se fait dans un unique fichier XML, ce qui peut parfois aboutir à des fusions difficiles. Un contournement existe en utilisant l'élément UML de type Package : celui-ci et son contenu sont stockés dans un fichier XML à part. Si vous être amené à travailler en simultané sur un gros projet de modélisation, le regroupement des éléments UML par Package vous simplifiera la vie ! (même si ce n'est pas l'utilisation normale des packages). Dans sa version actuelle, Visual Studio 2010 gère 5 diagrammes UML :

- Le diagramme de cas d'utilisation.
- Le diagramme d'activités.
- Le diagramme de séquence.
- Le diagramme de classes.
- Le diagramme de composants.

Deux autres types de diagrammes non UML sont aussi proposés : le diagramme de couches qui vous permet de modéliser et cartographier sous forme de couches logiques votre code et le graphe direct qui, lui, est un modèle générique permettant de représenter des diagrammes tels que le graphe de dépendance. Nous reviendrons en détail sur le premier type de diagrammes par la suite.

Comme nous le disions au début de l'article, l'intégration est un atout majeur de la suite de développement de Microsoft.

Conjuguée avec Team Foundation Server 2010, comme pour les différents éléments qui composent notre développement (documentation, fragment de code source, résultat de test, …), nous avons la possibilité d'associer un ou plusieurs éléments de travail à un élément UML ou directement à un diagramme. Cette relation est très précieuse car nous pouvons de ce fait obtenir une traçabilité qui ira jusqu'au code et aux tests !

Pour les développeurs, c'est la relation inverse qui leur simplifie la vie : pouvoir, à partir d'une tâche, remonter aux différents diagrammes qui décrivent le code à réaliser.

Malheureusement, dans leurs versions initiales, Visual Studio et TFS ne le permettent pas ! Un élément UML peut référencer un élément de travail mais on ne retrouve pas ce lien sur l'élément de travail.

Plus d'une personne ayant trouvé ce problème très gênant, Microsoft n'a pas attendu une nouvelle version ou

1&1 HÉBERGEMENT

TOUS LES PACKS AU PRIX DU MOINS CHER !

TOUS LES PACKS HÉBERGEMENT À SEULEMENT : 1,99 € HT/mois (2,38 € TTC/mois) pendant les 3 premiers mois*

Vous voudriez choisir votre solution d'hébergement sans vous soucier du prix ? Chez 1&1, tous les packs d'hébergement vous sont actuellement proposés au prix du moins cher : seulement 1,99 € HT/mois* durant 3 mois quelque soit le pack !

Faites votre choix sur 1and1.fr, où vous découvrirez également nos serveurs, e-boutiques et noms de domaines à petits prix !

DOMAINES À PRIX CASSÉS :

le .fr à 3,99 € HT/an (4,77 € TTC/an),
le .info à 0,99 € HT/an (1,18 € TTC/an)* !

* Tous les Packs Hébergement sont au prix du Pack Initial (1,99 € HT/mois, soit 2,38 € TTC/mois). A l'issue des 3 premiers mois, les produits concernés sont aux prix habituels (Pack Confort à 5,97 € TTC/mois, Pack Pro à 11,95 € TTC/mois, Pack Premium à 23,91 € TTC/mois). Frais de mise en service : 5,97 € TTC (Pack Confort) ou 11,95 € TTC (Pack Pro, Pack Premium). Offre soumise à un engagement de 12 mois. Offre domaines applicable la première année au lieu du prix habituel de 6,99 € HT/an (8,36 € TTC). Conditions détaillées sur www.1and1.fr. Offres sans engagement également disponibles.

MEMBRE DE united internet

Appel non surtaxé
0970 808 911

www.1and1.fr

1&1

un service pack pour y remédier. Avec le « Visualization & Modeling Feature Pack », il est maintenant possible via un nouveau type de lien d'élément de travail de référencer un élément UML.
Après cette présentation du projet de type modélisation, nous allons voir, au travers d'un exemple de distributeur de banque, chaque diagramme supporté par Visual Studio 2010 et nous apprécierons comment ils peuvent simplifier le développement.

Diagramme de cas d'utilisation

La première phase de tout développement est la définition des besoins. A partir des interviews des utilisateurs, on pourra créer des diagrammes de cas d'utilisation afin de formaliser leurs besoins de manière globale, compréhensible, concise et claire.
On va d'abord définir les acteurs qui vont interagir avec le système. Dans notre exemple, l'opérateur assure la maintenance, la Banque exécute les transactions et le Client utilise le système. La prochaine étape consiste en la définition des cas d'utilisation. La meilleure approche est de trouver les besoins que chaque acteur a envers le système.
On donne ensuite un nom unique et une brève description qui décrit clairement les objectifs pour chaque cas d'utilisation. Appliqué à notre exemple, on peut facilement trouver les cas d'utilisation suivants :

- L'opérateur démarre et arrête le système.
- Le client veut effectuer des transactions telles que retrait, dépôt ou transfert.
- La banque agit comme système extérieur avec lequel les transactions sont exécutées.

Enfin, les associations permettent de définir les liens entre les éléments du diagramme.
Dans Visual Studio 2010, tous ces éléments sont disponibles dans la boîte à outils. Après création d'un nouveau diagramme, il suffit d'utiliser le glisser/déposer pour constituer le diagramme avec les acteurs, les cas d'utilisation et leurs associations [Fig.3]. Jusque-là nous sommes dans le classique. Là où Visual Studio 2010 va apporter un plus, c'est grâce à son intégration avec TFS 2010 en permettant de lier les différents cas d'utilisation aux exigences stockées sous forme d'éléments de travail. Un développeur pourra alors, en partant de ce qu'il connaît le mieux (les éléments de travail), retrouver facilement via ces liens les diagrammes correspondant aux exigences sur lesquelles il va devoir travailler. Les exigences dans TFS pouvant être liées à des cas de tests, on aura donc aussi automatiquement un lien logique entre cas d'utilisation et cas de tests !

Diagramme d'activités

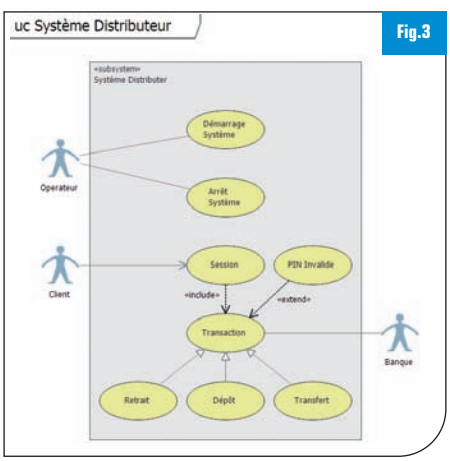
Après la première phase d'analyse et de définition de besoins, on commence à détailler les cas d'utilisation en diagrammes d'activités. Le diagramme d'activités décrit le comportement d'un processus d'un système. Une activité est un déroulement d'étapes séquentielles.
Dans notre exemple, on va étendre le cas d'utilisation « Transaction » par un diagramme d'activités qui va expliquer le comportement attendu :

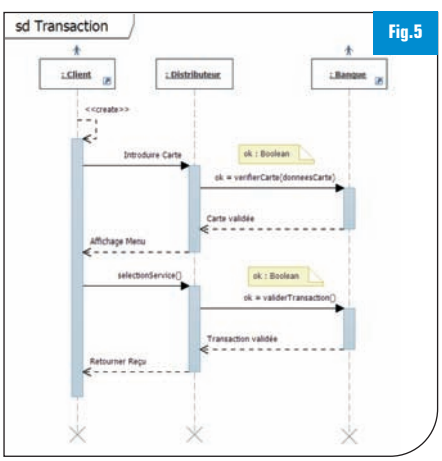
- le distributeur attend l'introduction d'une carte valide par le client.
- Le client sélectionne un service.
- Le distributeur délivre un reçu de la transaction.

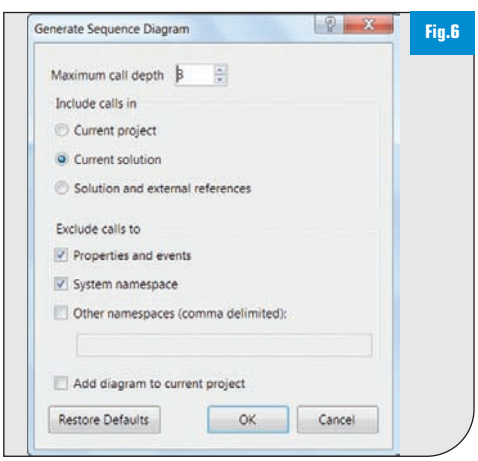
La modélisation de ce diagramme dans Visual Studio 2010 se fait de la même manière que pour la création du diagramme de cas d'utilisation, par glisser/déposer. La boîte à outils contenant maintenant les éléments adaptés tels que les actions, les décisions, les connecteurs … En quelques clics, le diagramme correspondant à notre exemple est créé [Fig.4].
Le diagramme d'activités sert à illustrer et consolider la description textuelle des cas d'utilisation. Il est également utile dans la phase de réalisation, car il donne les lignes directrices pour la programmation et pourra donc être lié aux tâches dans TFS afin que le développeur puisse se simplifier le travail en ayant accès directement au diagramme décrivant l'algorithme à mettre en place.

Diagramme de séquence

Une autre façon de détailler une partie ou l'intégralité des cas d'utilisation est le diagramme de séquence. Ce diagramme a pour but de montrer la séquence des interactions entre objets ou acteurs sur un axe de temps donné. Le diagramme est constitué de deux dimensions. La dimension verticale montre la séquence des messages dans le temps. La dimension horizontale montre les instances des objets auxquelles sont envoyés les messages. Les messages peuvent être synchrones ou asynchrones.
Un diagramme de séquence est très







simple à élaborer dans Visual Studio 2010. En haut du diagramme sont identifiées les instances de classe. Le temps est représenté du haut vers le bas le long des lignes de vie. Les messages sont représentés par des flèches d'un acteur vers un autre. Tout ceci s'effectue rapidement dans Visual Studio 2010 encore une fois grâce au glisser/déposer des objets de la boîte d'outils.

Dans notre exemple de la « Transaction », on identifie les interactions entre Client, Distributeur et Banque. On définit les messages entre les entités comme l'introduction et la vérification de la carte, l'affichage du menu et sélection du service par le client, la validation et l'exécution de la transaction et enfin le retour du reçu [Fig.5]. Comme pour le diagramme d'activités, on pourra lier le diagramme de séquence aux différents éléments de travail.

Une fonctionnalité très intéressante de Visual Studio 2010 est la possibilité de créer un diagramme de séquence à partir de code existant ! Dans le cas d'une application à reprendre ne possédant pas ou peu de documentation, cette fonctionnalité sera très utile afin de se donner une idée de ce que font certaines méthodes. Pour cela, il suffit de faire un clic droit sur la méthode concernée et on peut générer automatiquement le diagramme en choisissant les paramètres voulus [Fig.6].

Diagramme de classes

Le diagramme de classe permet de décrire les informations et les données utilisées par les utilisateurs et le système afin de répondre aux besoins exprimés. Un développeur est généralement plus habitué à ce type de diagramme car il est très proche du code. Visual Studio 2008 possédait déjà des diagrammes de classes mais ceux-ci n'étaient que des représentations visuelles des objets .Net du projet. Dans Visual Studio 2010, le diagramme de classe est un vrai diagramme de classe UML dont le contenu n'est pas lié à une implémentation particulière.

Une fois un nouveau diagramme de classe créé dans notre projet de modélisation, on va pouvoir le remplir avec des éléments classiques tel que :

- Une classe.
- Une interface.
- Une énumération.
- Un package.
- Des liens entre éléments (dépendance, héritage, ...).

Appliqué à notre exemple, on obtiendrait les éléments suivants [Fig.7] :

- Client : une classe représentant le client.
- Distributeur : une classe représentant le distributeur.
- Transaction : une classe abstraite qui sera la classe parente des classes représentant les différentes transactions (retrait, dépôt, transfert).
- Banque : une classe représentant la banque.

A partir des différents diagrammes de classe un développeur a une meilleure vision de ce qu'il faut réaliser en termes d'objet. Ces classes et interfaces peuvent bien entendu être liées, comme pour tous les autres éléments de modélisation, à des éléments de travail dans TFS. Ces éléments sont généralement liés aux tâches de développement et permettent aux développeurs de rapidement trouver les informations nécessaires pour la réalisation de ces tâches. Mieux encore, avec le « Visualization and Modeling Feature Pack » le code peut être généré automatiquement via des template T4 ! L'inverse est aussi possible, générer les diagrammes de classes à partir de code déjà écrit.

Enfin afin de se simplifier la vie, les nouvelles fonctionnalités d'extension de Visual Studio 2010 permettent d'ajouter ses propres éléments dans la boîte à outils. Cela permet, par exemple, de définir un élément représentant un design pattern couramment utilisé afin de ne pas avoir à le re-modéliser à chaque fois.

Diagramme de composants

Une fois que l'on a une vue sur les classes et interfaces nécessaires afin de réaliser les besoins des utilisateurs, on peut commencer à le regrouper sous forme de composants. Ces composants représentent les parties centrales de l'application et les diagrammes de composants permettent de les représenter en spécifiant comment ils dépendent les uns aux autres. On obtient alors une représentation macro de la conception de l'application. Dans le cas de composants complexes, on peut créer des sous-diagrammes de composants pour décrire leurs structures internes. Le principe étant d'apporter une vision globale sur l'architecture de l'application.

Comme pour les autres diagrammes avec Visual Studio, la création se fait par simple glisser/déposer depuis la boîte à outils. Dans notre exemple cela donnerait les composants suivants [Fig.8] :

- Client : composant représente l'utilisateur.
- Distributeur : composant représentant le distributeur.
- Transaction : composant représentant une transaction de manière générale.
- Banque : composant représentant la banque.

Pour l'ajout des interfaces exposées

Fig.4

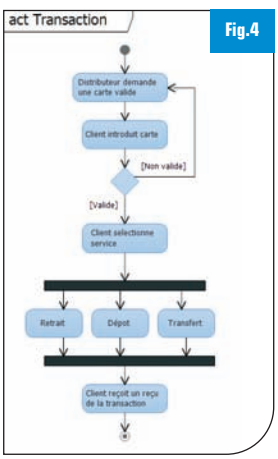
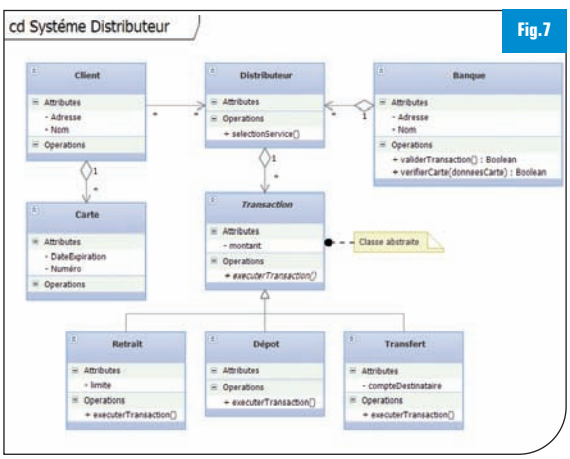


Fig.7



et attendues sur les composants, rien de plus simple, il suffit d'ouvrir l'explorateur de modèle et de glisser/déposer les interfaces qui ont été définies dans les diagrammes de classes ! Bien entendu l'inverse est aussi possible, vous pouvez définir l'interface dans le diagramme de composant et ensuite la glisser/déposer depuis l'explorateur vers un diagramme de classe.

Une fois les diagrammes de composants finalisés, on pourra vérifier que l'on n'a pas d'erreur comme des dépendances circulaires et il sera normalement simple de pouvoir regrouper ces composants en couches (IHM, service, métier et accès aux données par exemple).

Diagramme de couches

Nous quittons l'univers de l'UML pour aborder le dernier (et non des moindres) type de diagramme : celui des couches logiques. Pour beaucoup de personnes qui ont eu la lourde tâche de faire respecter le cloisonnement de l'architecture aux équipes de développement, cette fonctionnalité vaut de l'or !

Que l'on soit architecte, urbaniste ou développeur, avec l'expérience accumulée on parvient à se rendre compte qu'un des plus gros ennemis auquel nous ayons à faire face est le couplage entre les différentes briques de notre application. L'absence de maîtrise dans ce domaine fait que nous perdons totalement le contrôle sur les évolutions ou corrections que nous allons apporter à l'application : il est impossible de déterminer les répercussions de ce que nous faisons !

Combattre ce fléau à mains nues n'est pas une simple tâche et bon nombre d'entre nous finissent par rendre les armes très rapidement, laissant l'application livrée à elle-même, ce qui garantit un résultat accablant !

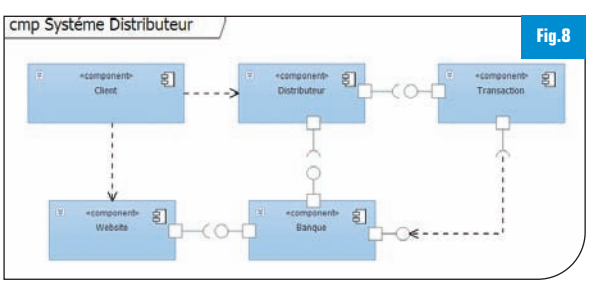


Fig.8

Le diagramme de couche apporte une solution aussi simple que radicale. Dans un premier temps on va représenter les différentes briques de son application sous forme de couches pouvant s'imbriquer puis définir les relations possibles entres chacune d'entre elles [Fig.9].

A ce stade nous avons un diagramme purement logique, nous avons alors la possibilité d'associer des éléments de notre solution aux différentes couches. Selon la granularité recherchée nous pourrons ainsi associer méthodes, classes, espaces de nom ou assemblages.

Là où ce diagramme prend tout son sens, c' est dans la phase dite de Validation. Visual Studio pourra valider pour vous que votre code est toujours conforme aux règles établies dans le diagramme ! Cette Validation peut être effectuée à la demande et lors de l'intégration continue.

Par exemple, si un nouveau développeur vient à appeler une méthode de la couche d'accès aux données à partir de l'interface graphique, la validation du diagramme générera une erreur car aucune relation n'existe entre la couche « UI » et « Accès aux données ».

Cette fonctionnalité permet de gagner un temps considérable, voire de rendre possible ce qui n'était auparavant que grossièrement faisable et de manière relativement inefficace.

Conclusion

Utilisée à bonne escient, la modélisation apporte une réelle simplification au développement et aux développeurs. Elle permet de représenter visuellement et simplement les besoins utilisateurs et comment ceux-ci doivent être réalisés. En intégrant la modélisation UML, Visual Studio 2010 rattrape son retard vis-à-vis de la concurrence mais Microsoft a su ajouter des fonctionnalités très utiles pour apporter de la simplification là où il en manque cruellement :

- Le projet de type modélisation, intégré directement à la solution, permet aux développeurs de s'approprier les diagrammes et de ne plus voir la modélisation comme une discipline obscure et sans intérêt.
- L'intégration avec Team Foundation Server va quant à elle permettre à la fois une meilleure traçabilité entre la définition des besoins et les diagrammes UML et un accès simplifié au diagramme à partir des éléments de travail, grâce aux liens que l'on va pouvoir créer.
- Enfin, en plus de la modélisation UML, Microsoft est allé plus loin et a ajouté le diagramme de couches qui permet non seulement d'architecturer son application mais en plus de la valider vis-à-vis du code !

■ Guillaume **Rouchon**
Architecte .Net / Expert ALM, .Net Rangers by Sogeti
Blog : http://blog.qetza.net

■ Loïc **Baumann**
Architecte .Net / Expert ALM, .Net Rangers by Sogeti
Blog : http://loicbaumann.fr

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

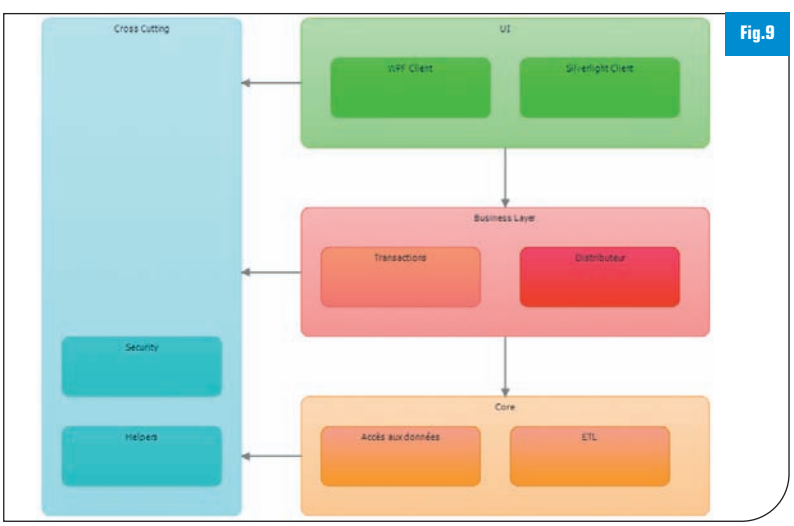


Fig.9

RÉUSSISSEZ VOS CERTIFICATIONS AVEC Microsoft Press

9782100521975 ▪ 800 pages

9782100521968 ▪ 640 pages

39 €
seulement
à partir du 1er novembre 2010

9782100532711 ▪ 624 pages

9782100532728 ▪ 688 pages

9782100530366 ▪ 672 pages

Réalisation : MATEO

Les ***Kits de formation Microsoft Press*** sont les ouvrages de référence, exhaustifs et pratiques certifiés par Microsoft. Ils contiennent toute l'expertise nécessaire pour vous accompagner dans l'évolution de votre métier et pour réussir les examens Microsoft.

Tous nos ouvrages sont disponibles en librairie ou sur commande auprès de votre libraire.

MICROSOFT PRESS : LA RÉFÉRENCE OFFICIELLE

Maîtriser un développement guidé par le modèle dans une démarche projet

Les activités de gestion de configuration, de gestion de version, d'intégration, de validation ainsi que l'organisation des équipes et de la coopération sont souvent peu appréciées des équipes de développement mais restent cruciales pour la maîtrise d'un projet. C'est particulièrement sur ces points durs que sont expérimentées les limites des approches guidées par le modèle si les outils et méthodes ne sont pas adaptés.

S'il est relativement facile de présenter une solution produisant une grande quantité de code, impressionnante sur des cas d'étude, la confrontation avec la réalité des exigences des projets en termes de travail en équipe, de gestion de configuration et version nécessite des procédures et techniques opérationnelles et pragmatiques rarement effectives.

Nous allons prendre l'exemple du fonctionnement interne de l'équipe de développement du produit Modelio (société SOFTEAM) en Java et C++, utilisant le produit Modelio lui-même pour la modélisation et génération de code, pour illustrer un exemple de solution.

Comment gérer le travail coopératif d'une grande équipe sur des modèles de très grande taille ? Cette question est essentielle pour bénéficier des avantages de la modélisation et du développement guidé par le modèle. Outre des points techniques, liés aux capacités de l'atelier de modélisation et aux outillages divers employés, le point organisationnel est fondamental, avec en particulier la nécessaire adhésion de l'ensemble de l'équipe de développement pour une bonne discipline collective.

Le succès ne peut être obtenu que s'il y a des avantages clairement perçus :

- Le développeur doit percevoir le gain, notamment au niveau de sa productivité, de son confort de travail et d'une coopération facilitée avec ses partenaires.
- Le projet doit obtenir un gain qualitatif clairement mesuré et perçu, sensible au niveau des responsables du projet, mais aussi des niveaux supérieurs de management.

Atelier Modelio, module Java designer – Modèle et code sont constamment synchronisés

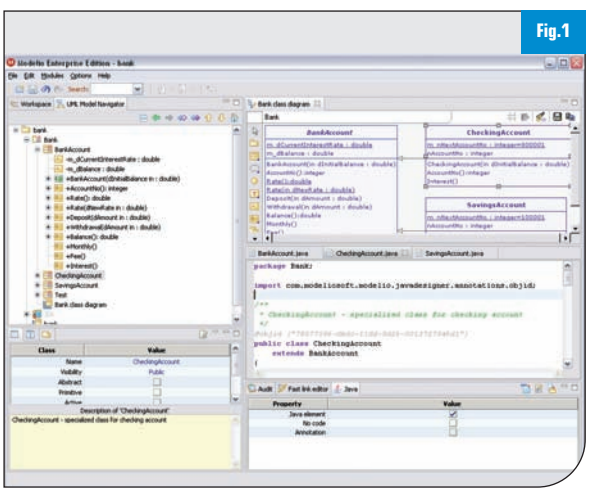


Fig.1

Développement guidé par le modèle : les outils et techniques de productivité

La génération de code pratiquée selon deux modes : « model driven » ou « round trip »

L'approche guidée par le modèle doit être équilibrée par les avantages et inconvénients d'une orientation modèle ou d'une orientation code. Tout modéliser avant de coder peut s'avérer excessif dans le cas de petites classes techniques Java (par exemple), cependant qu'une approche centrée code aboutit à une production volumineuse de code, très productive avec des outils tels que Eclipse, mais souvent mal architecturée si on n'y prend garde.

Modelio permet ainsi deux modes de développement guidés par le modèle :

- « Model driven », où le modèle est élaboré, puis le code généré en autorisant l'écriture du code des méthodes entre des balises dans le code Java. Modelio est donc maître, Eclipse (dans le cas Java) étant utilisé pour compléter le squelette de code produit. C'est en théorie le mode préconisé, car il garantit un respect de la conception faite au niveau du modèle.
- « Roundtrip », ou le code est retouché librement, et où Modelio resynchronise le modèle avec le code. Dans ce cas, Eclipse/Java est maître, et Modelio adapte le modèle au code. C'est un mode apprécié des développeurs, car il leur laisse toute la puissance de l'environnement Java [Fig.1].

Modelio conserve l'ensemble des informations - modèle et compléments code – dans son référentiel. A tout moment, l'ensemble du code peut être régénéré depuis Modelio. Ainsi, que l'approche soit « model driven » ou « roundtrip », la référence reste toujours le modèle, ce qui permet de se contenter de gérer le modèle en version, et non pas le code.

La modélisation, séparée en trois niveaux

Selon la phase de réalisation (analyse, conception, conception détaillée/codage), les parties de modèle mises en œuvre ne sont pas identiques. Dans les phases d'analyse, les modèles permettent de bâtir le dictionnaire de l'application, de définir ses exigences, de réaliser les Use Case, de construire des modèles conceptuels, et d'utiliser la totalité des capacités UML de modélisation.

La conception définit l'architecture générale en construisant des modèles productifs, essentiellement des modèles de classe, qui sont concentrés sur les grands principes et patterns architecturaux, et les classes essentielles du système. Le modèle de conception produit le code applicatif, soit en génération intégrale via les technologies MDA de Modelio, soit en génération de code en mode « model driven ».

La phase de codage/conception détaillée vient ensuite compléter le modèle de conception générale en mode « roundtrip », c'est-à-dire en utilisant massivement les capacités de l'environnement Eclipse/Java et en resynchronisant constamment le modèle.

La conception ne rentre pas dans le détail de classes Java trop spécifiques, typiquement dans le cas de construction IHM. Les classes principales de dialogue sont modélisées en conception, ainsi que les classes génériques IHM, puis générées, et le reste est fait sous Eclipse en réalisation puis reversé sous Modelio.

L'implémentation utilise le mode « roundtrip » plus flexible pour le programmeur, en préservant les classes de conception en mode model driven.

Ainsi, l'analyste réalise des modèles pour une production essentiellement documentaire, ou pour reprise ultérieure en conception. Le concepteur réalise des modèles pour générer du code, qui doivent être pérennes et non remis en cause par le codage plus détaillé. Lors du codage plus détaillé sous Eclipse, le mode « model driven » n'autorise alors de retoucher et compléter que le corps des méthodes, balisé dans le source. Les modifications plus importantes doivent être faites sous l'atelier Modelio au niveau du modèle. Le programmeur intervient soit dans les modifications du code des opérations entre balises pour le mode « model driven », soit peut directement créer des classes techniques ou retoucher le code des classes sans restriction dans le mode « round trip ».

Un même intervenant peut jouer les différents rôles (analyste, concepteur, programmeur). En première itération (Analyse/Conception/Codage), la phase de conception est clairement identifiée. Mais en phase de codage et dans les itérations suivantes, si la phase d'analyse reste bien séparée, il est cependant de la responsabilité de l'intervenant de choisir les modes « model driven » ou « round trip », selon son appréciation du niveau d'intervention (conception ou codage).

Tous les développeurs ont deux écrans sur leur station de travail : un écran est dédié au développement (Eclipse-Java ou Visual Studio-C++), et un écran est dédié à la modélisation (Modelio).

Les capacités MDA de génération dédiée à une architecture

Comme tout système informatique, l'application Modelio est appuyée sur une architecture dédiée. Les aspects systématiques et répétitifs de celle-ci donnent lieu à automatisation depuis une modélisation spécifique vers le code, afin qu'un principe architectural jugé optimal après de premières études et mises en œuvre soit systématiquement appliqué et automatisé.

Outre une qualité garantie, une performance de développement démultipliée, cette approche permet lorsqu'un principe architectural évolue, de le réappliquer sur l'ensemble des cas concernés, et donc de moderniser toute l'application avec peu d'efforts.

Pour notre cible applicative, qui est l'atelier de modélisation Modelio lui-même, les transformateurs et générateurs MDA dédiés permettent à partir du métamodèle (définition par un diagramme de classes spécialisé du modèle supporté : UML, BPMN, modélisation des exigences, …) de produire automatiquement : la gestion de la persistance de ce modèle, la construction de la vue « explorateur » du modèle, l'élaboration des boîtes de saisie des différents éléments du modèle, le contrôle de cohérence du modèle, la production de son interface de manipulation programmatique (API Java), et l'implémentation de différents mécanismes fonctionnels comme la gestion des verrous, les mécanismes de comparaison/fusion de modèles, etc.

Les éditeurs graphiques ne sont pas produits automatiquement, du fait de particularités importantes de chacun des cas, mais un « pattern » de modèle permet d'assister grandement leur programmation en permettant au développeur de se concentrer exclusivement sur les parties spécifiques à coder.

Parfois, de simples macros de manipulation de modèle, faciles à écrire en Python dans Modelio, permettent de traiter des « refactoring » massifs au niveau modèle avant régénération du code.

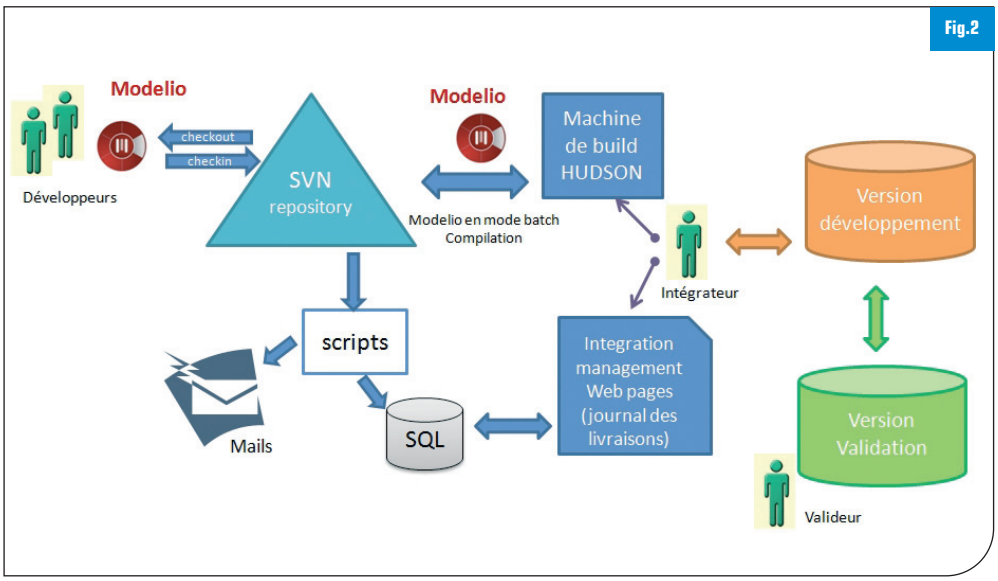
L'application de MDA permet soit de générer totalement le code à partir du modèle (ou du modèle plus compléments de code) via des transformations de modèle intermédiaires et des générations finales de code, soit de générer un modèle augmenté (par exemple via un pattern partiel) nécessitant un travail complémentaire de modélisation et codage. Pour une maîtrise des artéfacts gérés en configuration en phase de conception/codage, seul le modèle produisant du code est géré en configuration. Ainsi dans le deuxième cas, le seul modèle retenu et géré en configuration est le modèle augmenté, les étapes intermédiaires ne sont pas mémorisées, au même titre que l'on ne mémorise pas chaque action de modélisation ou codage en configuration.

Organisation du travail d'équipe

Plusieurs équipes contribuent au développement Modelio, réparties en plusieurs projets. Les développements utilisent des technologies variables selon les projets (C++ pour le cœur de l'atelier, Java pour l'interface homme/machine, projets MDA pour les modules Modelio). Une personne est dédiée à l'intégration, et une équipe séparée gère la validation [Fig.2].

Les outils utilisés ont été configurés et couplés à Modelio :

- L'atelier Modelio : Modélisation, génération de code (www.modelio.fr)
- Les modules Modelio Java Designer, C++ Designer, MDA Designer, et Teamwork manager



Organisation d'un projet

- Des extensions Modelio dédiées au support du processus de développement (modules Java spécifiques) : intégration avec les autres outils, automatisation de procédures, génération automatique vers notre framework interne de gestion de modèles
- Eclipse : (www.eclipse.org) outil open source pour les développements Java
- Visual Studio : outillage Microsoft pour les développements C++
- Subversion (SVN) : (www.subversion.tigris.org) outil open source de gestion de version et configuration. Il est utilisé via le couplage natif Modelio fourni par le module teamwork.
- Mantis : outil open source de gestion d'anomalie. Application web. (www.mantisbt.org)
- Testlink : outil open source de gestion des tests (http://sourceforge.net/projects/testlink/)
- Hudson : outil open source de production automatisée d'applications. (http://hudson-ci.org/)

Au sein d'un projet Modelio, les développements sont très couplés (chaque développeur effectuant plusieurs « commit » par jour) et nécessitent une gestion des conflits d'accès sur les modèles.

Tous les outils de développement sont déployés à l'identique sur chaque poste de travail. L'ensemble des outils, appelé « Toolkit » est versionné, géré par le responsable intégration sous la supervision du chef produit qui décide des mises à jour chez les développeurs.

C'est un point fondamental pour le succès de l'ensemble. De cette façon, les environnements de développement sont homogènes, et les développeurs peuvent facilement intervenir dans les développements d'autres personnes.

Les procédures liées à l'intégration sont grandement facilitées.

Procédures de travail coopératif

L'ensemble du développement est découpé en « projets » (au sens de Modelio), qui sont des unités de coopération homogènes, sur lesquelles coopère une équipe d'au maximum 6 personnes. Le projet contient le référentiel du modèle et du code, et fait l'objet d'une gestion de version et configuration dédiée. Les « composants de modèle » supportés par l'atelier Modelio sont utilisés pour gérer les livraisons entre projets différents. Seul le modèle est géré en configuration, le code pouvant sans cesse être reconstitué à partir du modèle. Un référentiel SVN, directement alimenté par Modelio, est créé par projet. Ceci évite tout problème de taille, simplifie la gestion des branches et évolutions, permet des cycles de vie indépendants pour chaque projet, et permet de configurer chaque projet individuellement (modules, versions, composants de modèles, autorisations…).

La gestion de version et configuration s'effectue à la granularité d'un package ou d'une classe (ou acteur, Use Case, …) qui est l'unité minimale de travail d'un développeur. Avec l'atelier Modelio, les participants réservent sur les modèles les parties sur lesquelles ils travaillent (check out), puis livrent et libèrent celles-ci selon leur avancement (commit, check-in).

Le travail coopératif nécessite, outre les outillages Modelio Teamwork et Subversion, une discipline des participants. Un concepteur/développeur a les responsabilités suivantes lorsqu'il réalise ou modifie une partie de l'application :

- prendre en « checkout » les éléments du modèle qu'il doit modifier (ceci est imposé par Modelio), en minimisant le nombre d'éléments verrouillés,
- modifier ces éléments en respectant les règles méthodologiques citées précédemment (model driven versus roundtrip),
- « livrer » avec Modelio ses modifications pour les mettre à la disposition de l'équipe,
- garantir la « compilabilité » de ses modifications avant leur « livraison » ou « publication »,
- tester ses modifications avant leur « livraison » ou « publication »,
- documenter sa livraison (ce qui a été modifié, pourquoi, comment, impact sur la validation etc.)

Lorsqu'il livre ou publie des modifications, l'intervenant doit renseigner ce qu'il a fait. Ceci est assisté par l'outillage (extension MDA de Modelio Teamwork Manager), qui lui fournit un formulaire à remplir obligatoirement lors de chaque « check in » ou « commit » de ses modifications, avec des indications courtes du type : « Correction de l'anomalie 3765 » ou « Ajout de la fonction –move()- sur les objets graphiques ». Il doit aussi renseigner l'impact sur la validation : en quoi le comportement fonctionnel a été modifié (changement de commande, nouveau menu, …). Les personnes concernées (développeurs impliqués, équipe d'intégration, de validation) sont averties par mail de l'évolution, qu'elles peuvent alors prendre en compte.Cette responsabilisation des acteurs dans le processus est essen-

tielle pour susciter leur adhésion au processus mis en œuvre. Elle leur permet d'ailleurs d'intervenir dans l'amélioration du processus.

En pratique, pour six intervenants simultanés sur un projet de 2800 classes, il y a peu de gêne occasionnée par le mécanisme de verrou. Le mécanisme de « add » de SVN, supporté par Modelio Teamwork Manager, permet en outre de créer de nouvelles parties de modèles au sein du modèle général, sans prendre de verrou tant que l'intervenant n'a pas livré ses premiers travaux.

Toutes les informations saisies dans le formulaire sont stockées dans une base de données relationnelle (MySQL). Ceci permet ensuite de faire des requêtes très utiles sur l'historique des travaux, typiquement pour savoir qui a changé quoi, pourquoi, et quelles sont les évolutions.

Lors d'une correction d'anomalie, le développeur doit accéder à l'outil de gestion des anomalies (Mantis), et mettre à jour le statut de celle-ci. Cette correction fait également l'objet d'une notification aux autres intervenants concernés.

L'intégrateur joue un rôle clé d'orchestration dans le processus : Il voit la synthèse des livraisons. Les livraisons des concepteurs et développeurs sont automatiquement assemblées grâce à l'outillage Hudson, qui reconstruit en permanence l'application dans son état de développement/livraison le plus récent (intégration continue).

L'intégrateur supervise la construction permanente d'applications, et peut intervenir, notamment auprès des développeurs si le « build » n'est pas correct.

Il décide, avec l'assentiment du responsable de produit, de basculer des applications vers l'espace de livraison/validation. A ce stade, les livraisons sont versionnées et fournies à la validation. L'équipe de validation connaît, grâce à Mantis et à la base renseignant les livraisons, ce qu'il faut (re) valider sur cette nouvelle version.

Industrialisation du développement guidé par le modèle

Dans cet exemple, l'ensemble du développement guidé par le modèle a été industrialisé. Au-delà de la génération de code, la coopération de l'équipe en mode projet est outillée par des extensions à l'atelier Modelio, des outils additionnels, et est supportée par des procédures spécifiques.

Il est indispensable d'avoir le triptyque « processus/outillage adapté/adhésion et participation des développeurs ». C'est à cette condition que le développement guidé par le modèle devient opérationnel en pratique, pour des développements professionnels. Pour obtenir une description plus détaillée de l'organisation, un « livre blanc » est publié sur www.modelio.fr, où des démonstrations sont disponibles ainsi qu'une version gratuite de Modelio.

■ Philippe **Desfray**
SOFTEAM
philippe.desfray@softeam.fr

Modélisation SysML

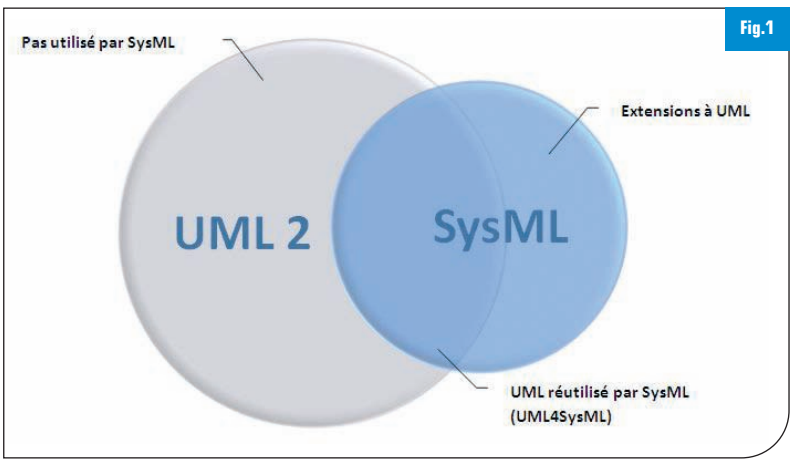
UML, langage de modélisation très répandu pour les développements logiciels, a été utilisé et adapté pour définir un langage de modélisation des systèmes : SysML ou Systems Modeling Language. Dans cet article nous présenterons l'Ingénierie Système, puis nous aborderons l'utilisation des modèles SysML.

L'**Ingénierie Système (IS)**, ou Systems Engineering en anglais (SE), est une démarche méthodologique pour répondre à des problèmes complexes par la réalisation de solutions logicielles et matérielles. L'Ingénierie Système s'adresse aux secteurs suivants : systèmes embarqués (ex : encodage/décodage audio et vidéo, set top box), automobile, ferroviaire, aéronautique, espace, militaire, télécoms, médical, production d'énergie, etc. Les méthodes de l'Ingénierie Système (IS) reposent sur des approches de **modélisation** et de **simulation** pour valider les exigences ou pour évaluer le système. La modélisation a donc été couramment utilisée pour la décomposition fonctionnelle, les flux de données, et la décomposition structurelle du système, faisant appel à des techniques telles que :

- Le diagramme de flux de données (DFD ou Data Flow Diagram) pour définir les données traversant un système et leurs traitements éventuels
- Le diagramme de flux fonctionnel de bloc (FFBD ou Functional Flow Block Diagram) proche du diagramme UML d'activité

SysML : Présentation

SysML est basé sur UML et remplace la modélisation de classes par des **blocs** pour un vocabulaire plus adapté à l'Ingénierie Système. Un bloc englobe tout concept logiciel, matériel, données, processus, et même gestion des personnes. Comme indiqué sur le diagramme de la figure 1, SysML réutilise une partie d'UML2 (UML4SysML), et apporte également ses propres définitions (extensions SysML). SysML n'utilise donc pas les treize diagrammes d'UML2, proposant ainsi un ensemble de diagrammes adaptés à l'IS. SysML offre l'avantage d'être accessible aux développeurs logiciels qui vont retrouver de nombreuses similitudes avec UML2. SysML permet de produire des spécifications dans un langage unique pour des équipes hétérogènes, responsables de la réalisation des blocs matériels et logiciels. Les connaissances sont ainsi modélisées dans un référentiel unique qui améliore la communication entre les différentes équipes participantes, est accessible à tous, et permet la réutilisation des blocs réalisés [Fig.1]. SysML propose les diagrammes suivants :



- Diagrammes structurels
  - Le « Block Definition Diagram » (BDD) remplace le diagramme de classes
  - L' « Internal Block Diagram » (IBD) remplace le diagramme de structure composite
  - Le diagramme paramétrique est une extension SysML pour l'analyse de paramètres critiques du système
  - Le diagramme de paquetage reste inchangé
- Diagrammes dynamiques
  - Le diagramme d'activités est légèrement modifié pour SysML
  - Les diagrammes de séquence, d'états, et de cas d'utilisations restent inchangés
- Le diagramme d'exigences est une extension SysML

Cet ordre ne définit en aucun cas la méthodologie à suivre pour modéliser un système ; le sujet de cet article est la présentation du langage SysML et non les méthodologies appliquées aux projets.

Modélisation structurelle

BDD : Block Definition Diagram

Le diagramme de définition de bloc (BDD, ou Block Definition Diagram en anglais) représente la vue boîte noire d'un bloc. Ainsi le **bloc principal** et la hiérarchie des blocs qui le composent, qu'ils soient logiciels ou matériels, sont spécifiés dans ce diagramme. Le BDD est similaire à la première page d'une notice de montage d'un meuble, indiquant la liste des éléments et des pièces à assembler avec leurs quantités respectives. Par rapport à UML, le BDD de SysML redéfinit le diagramme de classe en remplaçant les classes par des blocs. Le diagramme BDD provient de l'exemple OMG du purificateur d'eau [Fig.2].

Informations liées au BDD :

- Les blocs sont représentés par des classes UML stéréotypées « block ».
- Le bloc principal définit le purificateur d'eau (Distiller) composé de 3 blocs :
  - un échangeur de chaleur (HeatExchanger) qui a un rôle de condensateur (condenser)

- deux bouilloires (Boiler) qui ont toutes les deux un rôle d'évaporateur (evaporator)
- une soupape (Valve) qui a un rôle de drain

- Les trois blocs font physiquement partie du bloc principal, car les liens utilisés sur le diagramme sont des agrégations fortes ou compositions, représentées par un losange plein. Si un bloc n'en faisait pas physiquement partie, on parlerait alors d'une référence, et l'association utilisée serait une agrégation simple, représentée par un losange vide.
- Le port de flux (flow port) est une nouveauté SysML ; il représente ce qui peut circuler en entrée et/ou en sortie d'un bloc, que ce soit des données, de la matière ou de l'énergie. Ainsi le BDD indique que les entrées du bloc « Distiller » sont de l'eau froide et de la chaleur externe.

IBD : Internal Block Diagram

Le diagramme de bloc interne (IBD, ou Internal Block Diagram) décrit la vue interne d'un bloc ou vue boîte blanche, et se base sur le BDD pour représenter l'assemblage final des blocs qui composent le bloc principal. Les blocs composant le bloc principal et définis dans le BDD, sont **instanciés** en **parties**. Ces parties sont assemblées par des **connecteurs**, les reliant directement ou au travers de leurs **ports** (ports standards avec interfaces exposées et/ou ports de flux). Par rapport à UML2, l'IBD de SysML redéfinit le diagramme de structure composite en ajoutant, entre autres, les ports de flux.

Exemple de diagramme IBD pour le purificateur d'eau [Fig.3].

Informations liées à l'IBD :

- Le bloc Distiller du BDD est copié sur ce diagramme
- Les blocs du BDD qui composent le Distiller sont instanciés en **parties** sur l'IBD, et sont intitulées comme suit : « rôle : nom du Bloc ». Le rôle d'une partie doit être cohérent avec les associations d'agrégations du BDD ; ainsi on retrouve le rôle « drain » défini sur l'agrégation du bloc Valve du BDD dans l'IBD via la partie « drain : Valve ».
- La **multiplicité** spécifiée sur une agrégation du BDD est cohérente avec l'IBD, dont la multiplicité est représentée sur les parties entre crochets, par exemple : « evaporator : Boiler [2] »
- Tous les ports du bloc principal sont reliés aux ports des parties internes par des connecteurs. Par exemple l'eau froide du Distiller est transmise en entrée à l'échangeur de chaleur.
- La direction d'un flow port peut être définie en entrée, sortie, ou entrée/sortie.
- Alors que les ports indiquent ce qui peut passer par un bloc, les **flux d'éléments** ou « **item flows** » définissent ce qui passe entre deux blocs reliés par un connecteur. Par exemple l'IBD spécifie que l'élément « externalHeat », de type Heat (bloc), circule entre le port d'entrée du Distiller et le port d'entrée du Boiler.

Value Types

Les Value Types sont une nouveauté SysML pour définir des types de valeurs réutilisables par des propriétés du modèle, par exemple des blocs. De façon similaire à la modélisation UML où les attributs de classes peuvent être typés par d'autres classes, SysML permet de définir des propriétés de blocs typées avec des Value Type. La Value Type a la particularité de contenir deux propriétés optionnelles : une **unité** et une **dimension**. Exemple : Value Type « °C » définie avec unit = 'degrés Celsius' et sans dimension. La unit « degrés Celsius» est définie avec dimension = 'température'.

Diagramme de paquetage

Le diagramme de paquetage permet de structurer le modèle tout comme avec UML.

Modélisation dynamique

La modélisation de l'aspect dynamique du système avec SysML repose sur une sélection de quatre diagrammes UML2 : diagrammes de

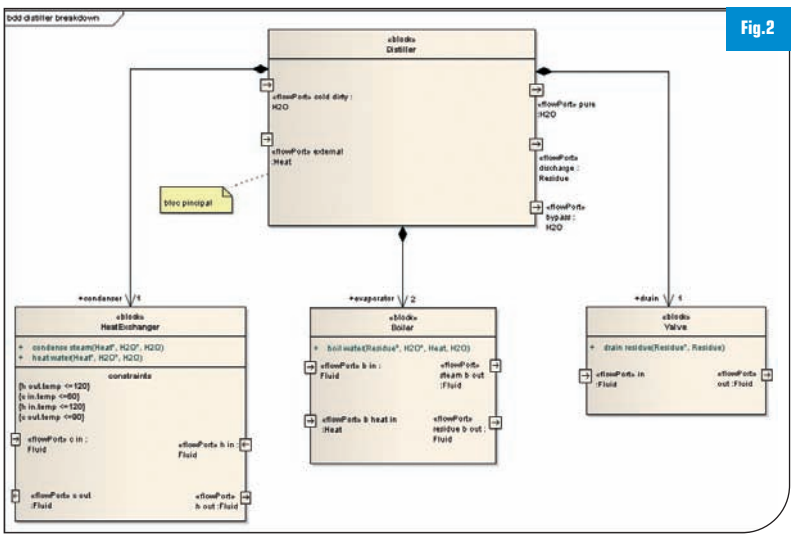


Fig.2

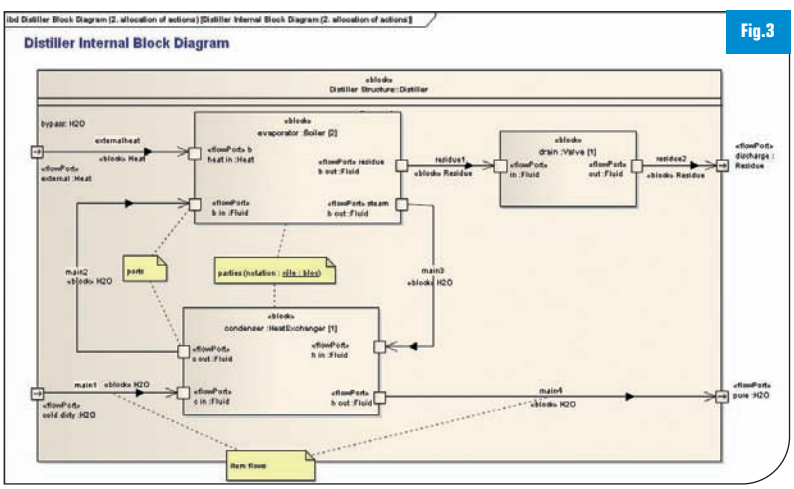


Fig.3

cas d'utilisations, de séquence, d'activité, et d'états.
Parmi ces diagrammes, seul le diagramme d'activité comporte quelques modifications pour SysML.

Diagramme de cas d'utilisations

Tout comme avec UML, on utilise les diagrammes de cas d'utilisations pour identifier les acteurs et les cas d'utilisations d'un point de vue utilisation du système, interactions acteurs/système. Les cas d'utilisations permettent de définir les frontières, le périmètre fonctionnel du système.

Diagramme de séquence

Le diagramme de séquence représente les éléments intervenant dans un scénario ou un processus, ainsi que les messages échangés dans un ordre chronologique.
Les éléments intervenant sont représentés par des **lignes de vie** (lifetime en anglais) pouvant être des instances de blocs du modèle. Cette instanciation de blocs établit un lien avec la modélisation structurelle du système donc une cohérence dans le modèle :

- On peut accéder aux propriétés du bloc d'une ligne de vie
- Chaque message échangé peut être utilisé pour l'identification des opérations de blocs

L'ensemble des propriétés du diagramme de séquence utilisées en UML sont également disponibles avec SysML : messages synchrones ou asynchrones, opérateurs (ex : alt, loop, opt, par), références vers d'autres diagrammes de séquence, etc.

Diagramme d'activité

Le diagramme d'activité est utilisé pour représenter les étapes d'un processus, impliquant en général les « input et output pins » qui correspondent respectivement au type d'élément requis en entrée d'une activité ou action, et à celui généré en sortie.
Si une action ou activité correspond à l'opération d'un bloc, il est alors possible de vérifier que les types d'éléments définis en entrée et en sortie de cette activité soient cohérents avec la signature de l'opération du bloc ou de son interface.
Toutes les propriétés des diagrammes d'activités UML sont également disponibles avec SysML. SysML a rajouté quelques spécificités :

- Notion de contrôle pour activer ou désactiver les actions en cours (Control Value)
- Spécification de la nature du débit sur le flot : système continu ou discret
- Définition de taux et de probabilité sur les flux de contrôle ou d'objets [Fig.4].

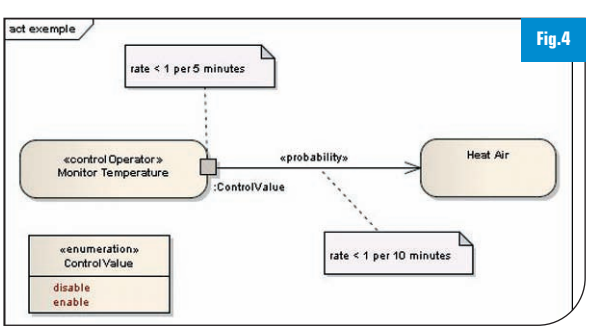
Diagramme d'états

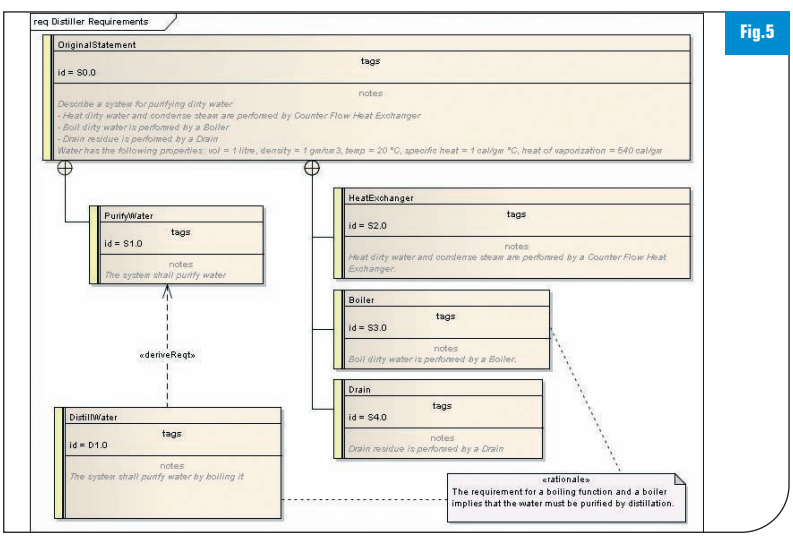
Le diagramme d'états est utilisé avec SysML de la même manière qu'avec UML2, c'est-à-dire qu'il permet de représenter le cycle de vie auquel doivent se conformer toutes les instances d'un bloc donné, ce au travers de la modélisation de tous les états possibles. Seuls les blocs qui sont importants d'un point de vue métier, ou qui sont de nature complexe, devraient avoir un diagramme d'état.
Les propriétés du diagramme d'état UML sont également disponibles avec SysML : conditions sur évènements, effets, activité durable, transitions, états composites, régions concurrentes, etc.

Nouveautés SysML

Exigences

Que ce soit pour l'ingénierie système ou uniquement pour des réalisations logicielles, les exigences sont couramment utilisées pour formaliser les pré-requis du système, se traduisant par des fonctionnalités ou conditions qui doivent ou devraient être satisfaites par le système (selon les éventuelles priorités associées aux exigences).
Pour la **maîtrise d'ouvrage** (MOA), les exigences ont pour objectif d'assurer l'adéquation de la solution (le système réalisé) avec les besoins. Les exigences peuvent être formalisées et catégorisées, par exemple différenciant les exigences fonctionnelles des exigences techniques (performance, fiabilité, ergonomie, etc.).
La formalisation des exigences peut être effectuée avec une feuille Excel, ou avec un outil spécialisé tel que DOORS ou CaliberRM. L'intérêt qu'offrent ces outils est la gestion des exigences dans une organisation structurée. Les exigences sont également utilisées pour la modélisation, par la création d'associations entre exigences et cas d'utilisations, blocs ou tout type d'élément du modèle, établissant la **traçabilité du modèle**. Il est possible avec l'outil Enterprise Architect de définir les exigences ou de les importer depuis un outil tel que DOORS, et de les associer avec les éléments du modèle.
SysML formalise les exigences et leur représentation, s'inspirant des fonctionnalités des outils actuellement





TOUR DE FRANCE

WINDEV® 16

996 NOUVEAUTÉS

ENVIRONNEMENT PROFESSIONNEL DE DÉVELOPPEMENT (AGL)

AGL N°1 en France

VENEZ DÉCOUVRIR LES NOUVEAUTÉS DE LA VERSION 16 PRÈS DE CHEZ VOUS

CALENDRIER

Montpellier	9 Nov
Toulouse	16 Nov
Bordeaux	17 Nov
Nantes	18 Nov
Paris	23 Nov
Lille	24 Nov
Bruxelles	25 Nov
Strasbourg	30 Nov
Genève	1 Déc
Lyon	2 Déc
Marseille	7 Déc

SÉMINAIRE PROFESSIONNEL GRATUIT

destiné aux DSI, donneurs d'ordre, développeurs, Webmasters...
13h45 à 17h15
inscrivez-vous sur ww.pcsoft.fr

Attention: 10.000 places seulement

Venez découvrir les fabuleuses possibilités de WINDEV, WEBDEV et WINDEV Mobile, et toutes les nouveautés de la version 16.

Venez découvrir comment il est possible de développer 10 fois plus vite des applications portables dans tous les environnements: Windows, Linux, .Net, Mac, Internet, Intranet, SaaS, Windows Phone 7, Android,...

www.pcsoft.fr

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

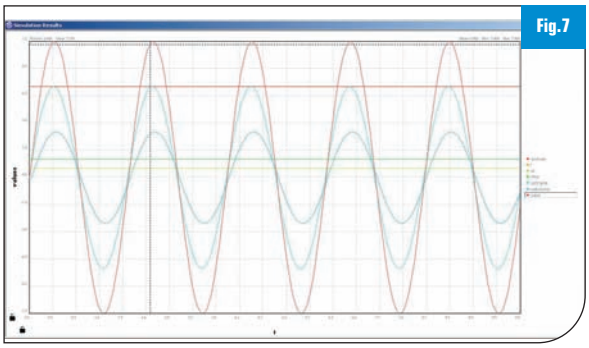


Fig.7

disponibles sur le marché. Ainsi **SysML définit une représentation graphique et visuelle des exigences textuelles, leur organisation hiérarchique, et des associations avec les éléments du modèle.**

SysML définit aussi de nouveaux types d'associations (liens de dépendance stéréotypés) :

- *Derive* : une ou plusieurs exigences dérivent d'une exigence
- *Satisfy* : un ou plusieurs éléments du modèle permettent de satisfaire une exigence
- *Verify* : un ou plusieurs éléments du modèle permettent de vérifier et valider une exigence
- *Refine* : un ou plusieurs éléments du modèle, par exemple un cas d'utilisation, redéfinissent une exigence

SysML définit de nouveaux commentaires stéréotypés permettant d'associer une explication à des associations ou éléments du modèle :

- *Problem* : commentaire dont la description pose le problème ou le besoin
- *Rationale* : commentaire dont la description indique la raison ou la justification par rapport à l'élément ou l'association associé

Exemple de représentation des exigences sous EA pour le système de purificateur d'eau : [Fig.5].

Diagramme paramétrique

Le diagramme paramétrique permet d'intégrer des analyses systèmes (performance, fiabilité, etc.) par des **blocs de contrainte**. Un bloc de contrainte représente une expression mathématique dont les paramètres peuvent faire référence à des éléments du système, par exemple aux propriétés de blocs. Dans un premier temps, de façon similaire à la création du diagramme BDD, les blocs de contraintes sont définis dans un diagramme de classe. Un diagramme paramétrique peut alors être créé :

- Les blocs de contraintes sont instanciés, donnant lieu aux **propriétés de contrainte** (ou **constraint property**) qui héritent ainsi de leurs paramètres (note : il n'y a pas de différenciation entre paramètres d'entrée et paramètres de sortie)
- Des propriétés systèmes (optionnellement liées à des blocs)
- Des connecteurs reliant l'ensemble des propriétés systèmes et paramètres des propriétés de contrainte

Diagramme paramétrique basé sur un exemple de lecteur MP3 proposé par l'éditeur Sparx Systems (Enterprise Architect) : [Fig.6].

Informations liées au diagramme paramétrique :

- Le diagramme comporte six propriétés de contrainte.
- Les paramètres systèmes sont catégorisés entre entrées (vert) et sorties (bleues). Ces paramètres sont associés aux paramètres des propriétés de contrainte (par exemple la fréquence d'entrée f est reliée au paramètre f de la propriété de contrainte SineWave)
- Certaines propriétés de contrainte sont reliées entre elles (ex : le paramètre SineWave.output est relié à Buffer.input et à Add.a)

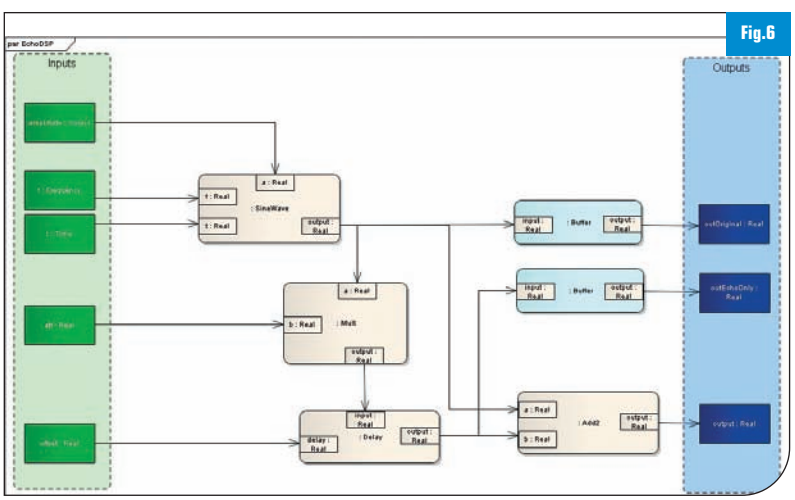


Fig.6

Certains outils permettent d'utiliser ce diagramme dans le cadre de **simulation**. Il est possible avec Enterprise Architect de saisir des expressions pour chacun des blocs de contraintes (par exemple en VBScript ou JavaScript), ainsi que de renseigner les valeurs des paramètres systèmes. L'exécution du module de simulation est illustrée ci-contre : [Fig.7].

Allocations

Le concept d'allocation est repris du vocabulaire des ingénieurs systèmes, indiquant un ensemble d'éléments associés dans un environnement structuré. La modélisation système implique des tentatives d'allocations entre éléments du système. Cet ensemble d'allocations est utilisé pour générer une matrice pour vérifier que les parties du système sont correctement intégrées. La création d'allocations permet de maintenir une cohérence entre les éléments du système, en particulier entre les modèles dynamiques et les modèles structurels.

Outils SysML

Le langage SysML a été intégré par de nombreux éditeurs d'outils commerciaux ou open source:

- Sparx Systems Enterprise Architect (plugin SysML ou version Ultimate requise)
- IBM Rational Software Modeler (plugin d'une société tierce disponible)
- MagicDraw (plugin SysML requis)
- Open source : TopCased (environnement Eclipse)

Pour en savoir plus

Un descriptif complet de la formation est disponible sur http://www.objetdirect.com/html/formation/catalogue/MODSY.html

■ Guillaume **Finance**
Consultant-formateur
Objet Direct
gfinance@objetdirect.com
www.objectdirect.com

NOUVELLE VERSION

WINDEV 16

DÉVELOPPEZ 10 FOIS PLUS VITE

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

VERSION EXPRESS GRATUITE
Téléchargez-la !

996 NOUVEAUTÉS

WINDEV 16 est l'environnement de développement totalement intégré, qui couvre l'intégralité du cycle de développement.

Pour le découvrir, vous pouvez télécharger la version Express, demander le numéro spécial «01 Informatique» de 100 témoignages, ou encore mieux: demander le dossier complet gratuit (avec DVD).

Et vous aussi, développez 10 fois plus vite.

ACTUALITE WINDEV
- *Opération «1 PC pour 1 Euro de plus»*
- *«Tour de France» des versions 16*

Voir les annonces dans ce magazine, merci.

Votre code est unique: Windows, Internet, Mobile, Java, .Net, PHP, J2EE, Mac, XML, Ajax, Webservice, Linux, Android, SaaS,...

Fournisseur Officiel de la Préparation Olympique

www.pcsoft.fr

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

PHP

installation, optimisation, choix !

Loin des tumultes des grands projets open source et libres (MySQL, OpenOffice, Java, Mandriva…), les langages vivent leur vie, avancent à leur rythme. PHP fait partie de ceux-là. Malgré de nombreux remous autour de l'hypothétique PHP 6.0 et des revirements multiples, la communauté et les responsables ne se pressent pas.

Revenons rapidement en arrière. L'une des grosses nouveautés de PHP était Unicode et une refonte de nombreux éléments du langage nécessitant un développement complexe et lourd engagé depuis 2005. Finalement, PHP 6 répondait bien aux attentes de langage et de performances. Et il était apparu qu'il serait plus lent, plus complexe que la v5… Finalement, la v5.3 intègre plusieurs nouveautés prévues dans la v6 : instruction goto, espaces de nom. Puis, en mars 2010, Rasmus annonce l'abandon de PHP 6…

Reste maintenant à remobiliser les développements pour faire évoluer le langage sans se couper des fondations actuelles, d'où la volonté de s'appuyer sur la lignée 5.x. Même si tous les problèmes aperçus avec les ambitions de la v6 et de gouvernance ne sont pas résolus, notamment sur le support, ou non, d'Unicode. Et pour le moment, le calendrier reste vide. A priori, rien de nouveau avant mi-2011. Mais des pistes se dessinent tout de même : épuration du langage, de nouveaux outils comme Dtrace, nouveau module CGI…

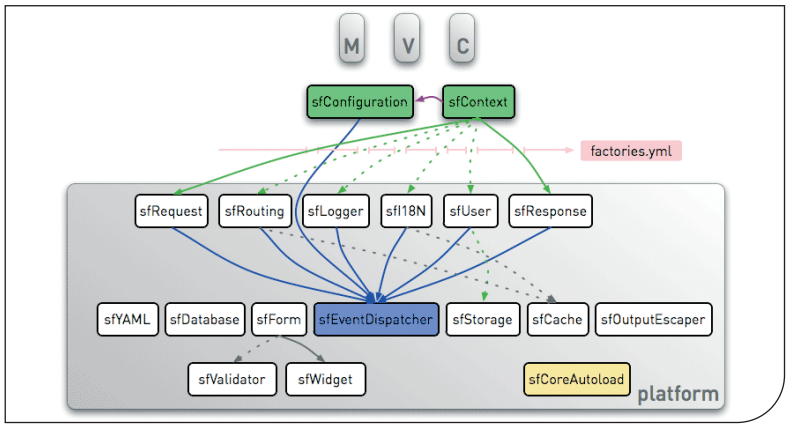
Alors que se déroule une nouvelle édition du Forum PHP en novembre, nous revenons sur PHP. Et particulièrement sur quatre points : optimisation d'exécution, choix du framework, installation et déploiement de PHP sous Windows, et enfin choix du gestionnaire de contenu PHP.

François Tonic ■

Bien choisir son framework PHP

Choisir un cadre de travail, framework en anglais, est une tâche délicate et souvent lourde de conséquences. En effet, il existe des centaines de frameworks différents et les critères de sélection pour les distinguer les uns par rapport aux autres sont d'autant plus nombreux. Par conséquent, il convient de prendre le temps de les comparer et de bien mûrir votre réflexion. En effet, le cadre de travail que vous choisirez sera celui qui correspondra le mieux à vos besoins quotidiens (et futurs), et deviendra la nouvelle base de travail pour toute votre équipe.

Comparer les frameworks les uns par rapport aux autres est une tâche particulièrement difficile en raison du nombre important de critères à prendre en compte. En effet, le choix d'un framework ne se limite pas seulement à son code source ni à la manière de l'utiliser. Bien qu'il s'agisse d'un outil visant à améliorer la productivité et la qualité des développements, un framework se doit avant tout de posséder un large écosystème. C'est le cas de Zend Framework, Symfony et CakePHP étudiés plus loin dans cet article. Ces trois frameworks se distinguent des autres outils car ils possèdent tous un écosystème riche et varié destiné au monde professionnel.



Comparaison des écosystèmes des frameworks

L'écosystème d'un framework professionnel est constitué de nombreux paramètres à ne pas négliger au moment de l'étude comparative. En effet, il convient d'étudier avec attention les quelques points importants listés ci-dessous pour chaque framework :

- la licence (GPL, MIT, BSD… ?),
- l'état de stabilité et de maturité du framework (nombre d'années d'existence ? stabilité de l'API ?),
- la richesse et la fraîcheur de sa documentation (wiki, ouvrages, blogs…),
- l'activité de l'équipe de développement (date de dernière mise à jour ? roadmap précise ?),
- la richesse de ses fonctionnalités (abstraction de base de données, architecture MVC, génération de code…),
- l'importance de sa communauté (contributeurs, événements sociaux, forums…),
- l'intégration dans les principaux EDI du marché (Eclipse, Netbeans, Zend Studio…)
- la durée de support à long terme ou bien la présence d'une société supportant le projet.

A l'heure où sont écrites ces lignes, le Zend Framework est en version 1.10 et publié sous licence BSD. Il bénéficie d'une feuille de route active puisque la version 2.0 est en cours de préparation. Cette nouvelle mouture nécessitera l'utilisation de serveurs en PHP 5.3 minimum. Le framework de Zend jouit d'une communauté importante et d'une documentation de bonne qualité. De plus, la richesse de ses outils couvre les fonctionnalités essentielles du développement web telles que la manipulation d'une base de données, l'authentification et les droits d'accès ou bien encore l'architecture MVC. Son interface en ligne de commande, introduite il y a quelques mois seulement, permet désormais au développeur d'automatiser un certains nombre de tâches fastidieuses et sans valeur ajoutée. Enfin, il est parfaitement supporté en natif par Netbeans et Zend Studio.

Le framework Symfony, publié sous licence MIT, a atteint sa version 1.4 en décembre dernier et est supporté pendant trois ans par l'équipe de développement et la société française Sensio Labs. Ce support à long terme garantit des mises à jour mineures régulières de correctifs de bogues ou de sécurité. Sa documentation est pléthorique et constamment mise à jour tandis que sa communauté est très active puisqu'elle contribue en retour au projet par le développement de plugins gratuits. Symfony bénéficie aussi d'un support par les principaux EDI du marché (Netbeans et PHPEdit) et d'une feuille de route précise puisque la version 2.0 est prévue à la fin de l'année. Des trois frameworks étudiés, il s'agit certainement du plus intéressant car il offre un jeu de fonctionnalités plus riche et une architecture plus stricte, obligeant les développeurs à redoubler de rigueur. Les développeurs Symfony apprécient tout particulièrement l'intégration de Doctrine et Propel comme couches d'abstraction de base de données, les nombreux outils d'aide au débogage, l'interface en ligne de commande ou

bien encore son générateur automatique de backoffice qui accélère considérablement les développements d'interfaces d'administration.
CakePHP bénéficie lui aussi d'une activité permanente puisque la version 1.3.3 est parue en juillet dernier. Sa documentation est également complète mais le framework souffre d'une baisse d'intérêt de la part des développeurs. Malgré sa maturité, CakePHP est contraint par un frein majeur à son adoption. L'intégralité du code source s'appuie sur la syntaxe de PHP4, ce qui l'empêche naturellement de profiter de tous les avantages de la programmation orientée objet de PHP5. Néanmoins, il demeure toujours un framework de qualité offrant une interface en ligne de commande ainsi que des outils de génération de scaffholding (squelette fonctionnel de module). La suite de cet article s'intéresse exclusivement à la partie technique de ces frameworks puisque la comparaison intervient ici au niveau du code. Il s'agit d'identifier les avantages et inconvénients de chacun en termes d'utilisation.

Comparaison technique des frameworks

Symfony, Zend Framework et CakePHP sont trois frameworks qui respectent le motif de conception MVC (Modèle, Vue et Contrôleur). Par conséquent, le traitement d'une requête HTTP est sensiblement le même d'un framework à l'autre. Les contrôleurs sont représentés par des méthodes tandis que les vues prennent la forme de fichiers de templates mélangeant généralement code HTML et quelques instructions PHP. En ce qui concerne la configuration, chacun de ces trois frameworks propose sa méthode.

La configuration

Commençons par étudier la manière de configurer l'accès à une base de données pour chacun des trois frameworks. Le framework Symfony s'appuie sur des fichiers au format YAML comme le montre le listing 1*. Le YAML est un format textuel hiérarchique. De la même manière qu'avec un fichier XML, les données sont représentées hiérarchiquement à l'aide d'indentations. Le framework de Zend, quant à lui, a choisi le format INI comme moyen de configuration. Le listing 2* montre comment configurer une connexion sur une base de données MySQL. Enfin, CakePHP fait usage d'un simple fichier PHP pour configurer les bases de données comme le présente le listing 3*.
Nous allons à présent étudier de quelle manière ces trois frameworks facilitent la création et le traitement des formulaires.

Le traitement des formulaires

Les formulaires constituent l'un des éléments de base des applications web dynamiques modernes. Ils ont pour objectif de favoriser les interactions entre l'utilisateur et le système d'information. Cependant, traiter un formulaire n'est pas une tâche si aisée en apparence. En effet, il est de la responsabilité du développeur de s'assurer que toutes les données saisies par l'utilisateur sont conformes à celles attendues.
Chacun des trois frameworks étudiés dans cet article offre un mécanisme pour simplifier la gestion des formulaires. Commençons par CakePHP qui offre un certain nombre de helpers de vue facilitant la création d'éléments de formulaires comme le présente le listing 4*. Le contrôle des champs est assuré par la définition de règles de validation au niveau de la classe de modèle correspondant. Le listing 5* montre comment déclarer des règles de validation dans le modèle Post afin de valider les champs titre et contenu. Enfin, le contrôleur du listing 6* a pour responsabilité de déclencher la chaîne de validation du formulaire et de sauvegarder l'objet de modèle en base de données en cas de réussite. Les frameworks Symfony et Zend arborent une stratégie différente de CakePHP. Les formulaires sont décrits sous la forme d'une classe PHP qui déclare des champs et des validateurs sur ces derniers. Chaque champ et validateur est aussi un objet PHP. Le listing 7* montre une classe de formulaire pour chaque framework. Les formulaires Symfony déclarent des widgets et des validateurs. Les widgets sont les objets responsables du rendu des champs tandis que les validateurs se chargent de contrôler et de modifier les données saisies. Le composant Zend_Form du Zend Framework met à disposition des classes pour créer des éléments de formulaires, des validateurs et des filtres. Les filtres sont des objets qui modifient une valeur validée en entrée. Par exemple : forcer une valeur en majuscules. L'utilisation de classes facilite la spécialisation et la réutilisation du code contrairement à ce qu'offre CakePHP. Le rendu des formulaires Zend_Form et Symfony est simplifié grâce à la méthode magique __toString() qui a pour rôle de générer le rendu complet du formulaire. Le listing 8* montre comment rendre les deux formulaires dans un template. Enfin, le listing 9* présente le code pour chacun des deux frameworks : une action chargée de déclarer et de déclencher la logique de validation pour chaque formulaire.

Conclusion

Chaque framework dispose de ses forces et de ses faiblesses. Le choix de l'un ou de l'autre ne dépend que de l'affinité que le développeur a avec le code et l'environnement. Le choix d'un framework plutôt qu'un autre est une tâche lourde de conséquences puisque c'est l'outil sur lequel seront développées vos futures applications. Nous ne pouvons que vous conseiller de bien tester chacun de ces outils afin de choisir celui qui correspond le mieux à vos attentes. Bien sûr, le choix d'un framework ne doit pas se limiter uniquement à sa philosophie et au code. Il convient de prendre en compte dès le début l'activité de la communauté et de la feuille de route du framework ainsi que l'état de stabilité de ses API et de sa documentation.

(*) Listings et Code source complet sur le site.

■ Hugo **Hamon**
Secrétaire Général de l'AFUP (Association Française des Utilisateurs de PHP)
Responsable des formations Sensio Labs
Co-auteur de l'ouvrage "Mieux Développer en PHP avec Symfony 1.2 et Doctrine"
http://www.hugohamon.com

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 .

Optimiser PHP ? c'est possible !

Nous allons étudier ensemble comment optimiser une application PHP et lui permettre de répondre à un plus grand nombre de sollicitations. Même si dans la majorité des cas un audit d'optimisation est lié à une situation d'urgence nous allons nous placer dans le meilleur des mondes : celui dans lequel il n'y a pas un client affolé derrière notre dos !

Supposons que nous avons le temps de prévoir et mettre en place notre optimisation sereinement. Dans ce cas nous devons utiliser la méthode suivante :

- mise en place d'outils de monitoring
- mise en place de tirs de charges réalistes
- analyse des données recueillies : recherche des goulots d'étranglements potentiels
- Améliorations de la plateforme : PHP, Apache, MySQL, architecture, ...

Étant donné que Linux est la plateforme recommandée pour héberger des applications PHP professionnelles, nous nous baserons sur cet OS. Il est cependant possible pour certains utilisateurs de l'OS à la fenêtre colorée de mettre en pratique une partie des conseils présentés.

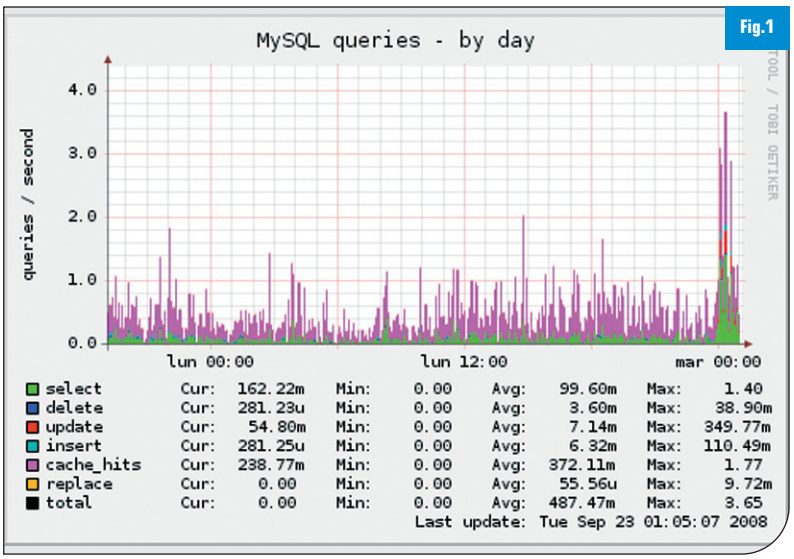


Fig.1

Présentation de Munin

MISE EN PLACE D'OUTILS DE MONITORING

Les fichiers de logs

PHP, Apache et MySQL comme la majorité des logiciels stockent une partie des informations liées à leur activité dans des fichiers. On parle de fichiers de logs et/ou de journalisation. Ces fichiers sont en général une mine d'or pour comprendre ce qui se passe sur votre serveur : statistiques d'accès, enregistrement des erreurs, requêtes SQL lentes... Mais attention, il n'est pas toujours évident de les manipuler et de savoir dans quel fichier aller chercher l'information.

Dans le cadre de notre logique d'optimisation nous allons mettre en avant principalement:

- La journalisation des requêtes SQL, notamment lentes et sans index,
- La détection des erreurs applicatives via les logs d'erreur PHP,
- La création de scénarios des tests via les logs d'accès d'Apache

Généralement, dans une configuration de production on active un minimum de journalisation pour éviter de se retrouver avec des fichiers énormes et souvent inutilisés. Quand on se place dans le cadre d'une logique d'audit d'optimisation il convient de récolter un maximum d'informations. Nous allons donc activer les enregistrements suivants :

- journal d'accès d'Apache
- journal des requêtes MySQL
- journal des requêtes lentes de MySQL
- journal des requêtes sans index de MySQL
- journal des erreurs PHP, MySQL et Apache

Ces fichiers contiennent des informations utiles pour analyser votre application. Si nous les couplons à la mise en place d'outils de monitoring et des tirs de charges, nous serons en mesure de détecter les goulots d'étranglements potentiels. La localisation de ces fichiers dépend de votre installation, nous vous encourageons à faire les recherches vous-même et à les activer.

Les outils de monitoring

Le suivi des logs ne génère qu'une partie des indicateurs à utiliser pour optimiser votre application Web, il faut y ajouter d'autres informations comme les ressources CPU, les statuts de vos serveurs, le load average, ... Ces données sont extrêmement intéressantes mais elles sont complexes à manipuler. Pour un suivi au jour le jour on leur préférera des outils de monitoring permettant un suivi graphique, comme Munin, Cacti ou Nagios.

Grapher les constantes systèmes

Il existe plusieurs outils permettant de suivre l'activité de votre système : Nagios, Cacti, Munin, Zend Server, MySQL Entreprise... et il est hors du cadre de cet article de tous les pas-

ser en revue. Nous allons vous présenter Munin qui est le plus facile à utiliser et configurer.
Munin fournit ses résultats sous forme de graphiques disponibles via une interface web comme indiqué sur la figure 1. Son architecture est modulaire et vous permet de l'adapter à votre environnement : des plugins sont disponibles pour presque toutes les applications qui nous intéressent (Linux, Apache, MySQL) [Fig.1].
L'architecture du système Munin est constituée d'un serveur principal appelé Munin-master, récupérant les informations à intervalle régulier et de plusieurs nœuds appelés Munin-node. Le nœud doit être installé sur le(s) serveur(s) à surveiller. Dans le cadre d'une utilisation simple le master et le noeud seront sur la même machine, mais il est possible de mettre en place une architecture plus poussée telle que présentée dans la [Fig.2].

Figer, restaurer et charger une base de données

Une fois que vous avez mis en place ces outils de monitoring, nous sommes presque prêts à travailler. Il reste une composante importante : l'état de la base de données. Pour être en mesure de comparer les résultats que vous obtiendrez vous devez interroger à chaque fois la même base de données. Il convient donc de la figer à un instant t et de la restaurer avant chaque tir de charges.

Créer des sauvegardes logiques avec mysqldump

L'utilitaire mysqldump est fourni en standard avec toutes les distributions du serveur MySQL, aussi bien sous Windows que sous Linux. Il exporte le contenu des tables vers un fichier texte. Voici une liste de ses principales caractéristiques :

- Il peut exporter au choix : toutes les bases, des bases spécifiques ou bien des tables spécifiques.
- Il peut être utilisé pour sauvegarder une base de données locale ou distante (selon le type de connexion MySQL utilisée).
- Les fichiers de sortie sont écrits au format texte et sont donc facilement portables.
- Il est rapide et pratique pour des petits exports mais limité dans le cadre d'une stratégie de sauvegarde professionnelle.

L'utilisation de mysqldump peut être très simple, il suffit d'exécuter le programme avec le nom de la base de données à exporter. Ainsi, dans notre cas, pour sauvegarder l'intégralité de la base de données « production » du serveur distant « mysql.domaine.com », avec l'utilisateur « sauvegarde » et sont mot de passe « motdepasse », le tout dans le fichier local « backup.sql », la ligne de commande sera:

```
mysqldump -h mysql.domaine.com -u sauvegarde -pmotdepasse production > backup.sql
```

En supposant que vous ayez sauvegardé toutes vos bases de données dans un fichier nommé « backup.sql » avec la commande « mysqldump », vous pouvez les restaurer à l'aide de la commande suivante:

```
mysql -u root -p production2 < backup.sql
```

Ceci aura pour effet d'exécuter la totalité des ordres SQL de backup.sql sur la base de données « production2 ». Le modèle de données sera alors re-créé et les données injectées à l'identique.

Bien entendu cette méthode implique que vous soyez en mesure de réaliser une sauvegarde à froid ce qui n'est pas toujours le cas.
Pour optimiser, vous pourriez également avoir besoin de « remplir » votre base de données : effectivement si vous souhaitez situer vos tests dans un futur plus ou moins proche il est fort probable que votre base de données soit bien plus remplie. D'un blog avec une centaine d'articles vous pourriez souhaitez tester votre application avec plusieurs milliers d'articles ; il y a effectivement fort à parier que votre base de données ne répondra pas de la même façon.
Vous expliquer comment faire est hors du spectre de cet article et demande généralement de l'huile de coude, nous nous contenterons donc dans un premier temps de notre base actuelle.

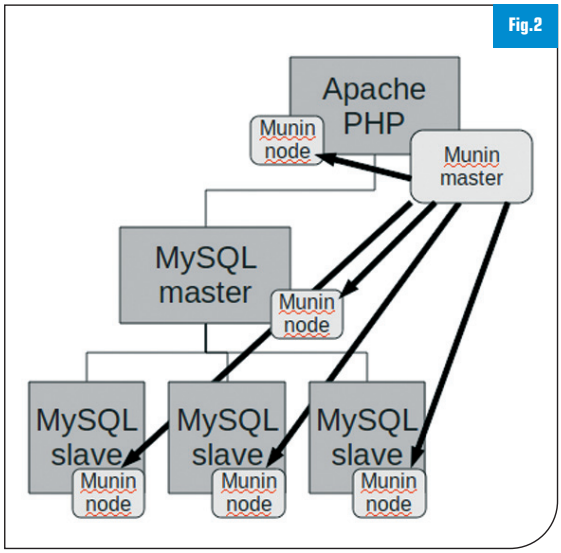


Fig.2

Architecture de Munin

MISE EN PLACE DE TIRS DE CHARGES RÉALISTES

Nous venons de voir comment mettre en place les bases d'une architecture de surveillance. La prochaine étape est de créer des scénarios de tests et de mettre en place des tirs de charges. Pour créer des tirs de charge réalistes on peut soit se baser sur les logs d'accès à Apache soit enregistrer des scénarios ou encore les imaginer. Nous allons utiliser Jmeter pour enregistrer des scénarios et les rejouer.

Qu'est-ce qu'un tir de charge ?

Un tir de charge est un moyen de simuler un nombre prédéfini de requêtes simultanées sur un site web. Il permet de mettre un site en situation de quasi production, permettant ainsi de se rendre compte de son comportement en termes de performances un fois soumis à du trafic. C'est une étape nécessaire à tout projet de site web à vocation commerciale, idéalement c'est une tâche qui sera intégrée dans un processus d'intégration continue, comme les test unitaires et fonctionnels. L'expérience et la réalité montrent que ce n'est presque jamais le cas, et que les tirs de charges sont effectués soit à la toute fin du projet, quand il ne

reste plus assez de temps disponible pour corriger les problèmes, ou alors une fois le site en production, après un moment difficile ou une indisponibilité suite à un pic de trafic.

Présentation de JMeter

Il existe une multitude d'outils permettant de générer du trafic. Tous ont leurs avantages et leurs spécificités, cependant un outil couramment utilisé pour des applications web est JMeter. Projet porté par la fondation Apache, JMeter permet de générer du trafic sur n'importe quel type d'application Web. Initialement conçu pour ne générer de la charge que pour des applications web, le projet s'est développé peu à peu et permet de faire des tirs de charge sur autre chose qu'un site web, par exemple un annuaire LDAP. Du fait de son écriture en Java, il est nativement multiplateforme et ne nécessite rien d'autre qu'un simple téléchargement pour être utilisé. Il peut être téléchargé librement sur son site officiel http://jakarta.apache.org/jmeter/.

Utilisation de JMeter pour simuler du trafic

Maintenant que nous en savons un petit peu plus sur JMeter et sur ses capacités, passons tout de suite à la pratique.

Enregistrer un parcours de navigation

Afin de prendre un exemple réel, nous allons effectuer un tir de charge sur le site officiel de JMeter. Mais pour faire un tir de charge reproduisant un parcours de navigation sur le site, il nous faut enregistrer les URL qui ont été utilisées pour ce parcours. Pour nous simplifier la tâche nous allons utiliser la fonctionnalité « Proxy » de JMeter. Nous allons partir du principe que la navigation sur le site officiel de JMeter se fait de la façon suivante :

- visite de la page d'accueil
- visite de la page Changes
- lecture de la page User Manual

Nous allons préparer notre plan de travail pour qu'il soit prêt à enregistrer notre visite sur le site de JMeter. Il nous faut d'abord créer un nouveau plan de travail, et ensuite installer le système de Proxy, comme montré dans l'image : [Fig.3].

Nous pouvons laisser la configuration par défaut, nous allons simplement exclure les images de l'enregistrement car nous n'en voulons pas dans ce cas précis. Dans la partie « ULS: motifs à exclure », pour pouvez ajouter les expressions régulières suivantes, une par ligne :

```
.*\.jpg
.*\.gif
.*\.png
.*\.css
.*\.js
```

Ensuite il nous faut insérer quelques éléments afin d'avoir un plan de test utilisable.

Tout d'abord, il nous faut créer un nouveau groupe d'unités en faisant un clic droit sur la ligne « Plan de test » puis Ajouter Æ Moteurs d'utilisateurs Æ Groupe d'unités [Fig.4].

Une fois ce groupe d'unités créé, il nous faut utiliser un contrôleur enregistreur pour pouvoir enregistrer dans JMeter ce que le proxy va écouter. Il suffit de faire un clic droit sur le groupe d'unités nouvellement créé et de sélectionner Ajouter Æ Contrôleurs logiques Æ Contrôleur enregistreur [Fig.5].

Maintenant que tout est prêt du côté de JMeter il nous faut aussi reconfigurer Firefox afin qu'il utilise le Proxy que nous venons de mettre en place dans JMeter il suffit de régler les préférences réseau, et configurer le proxy comme étant atteignable sur 127.0.0.1 sur le port 8080 [Fig.6].

Nous pouvons maintenant cliquer sur

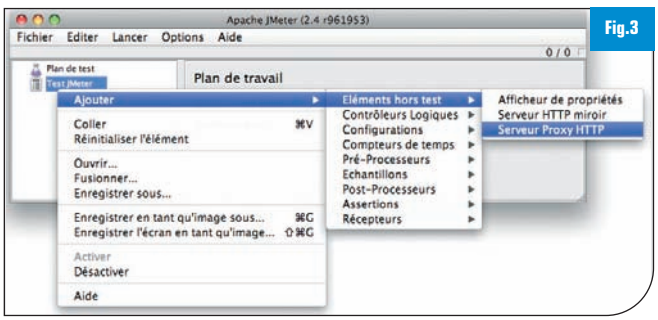


Fig.3

Mise en place du proxy

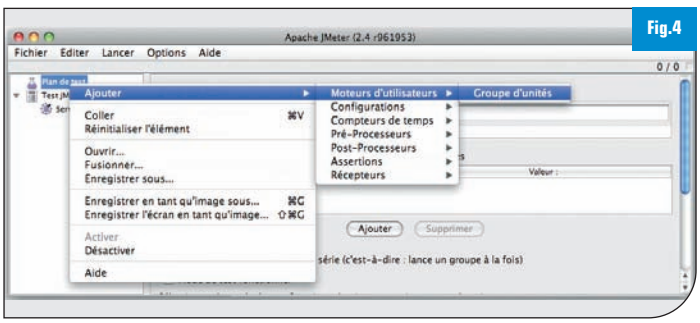


Fig.4

Groupe d'unités

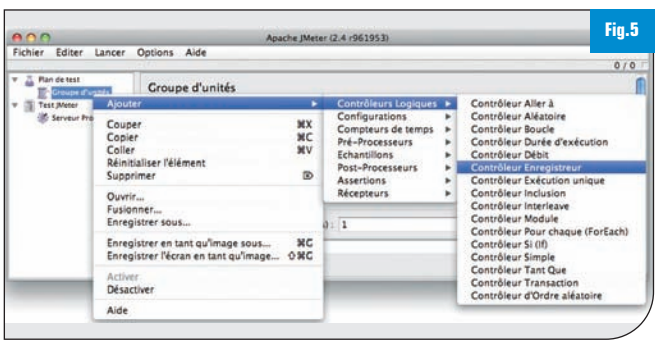


Fig.5

Enregistreur

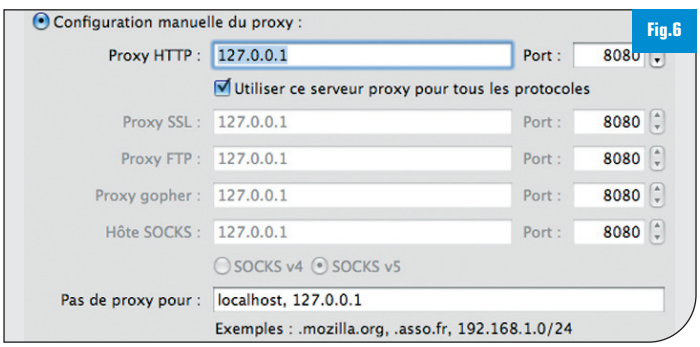


Fig.6

Proxy Firefox

le bouton « Lancer » dans le panneau réservé au Proxy dans JMeter, puis effectuer le parcours de navigation défini précédemment. Après avoir retiré les URL correspondant à de la publicité, vous devriez avoir le résultat final suivant : [Fig.7].

Effectuer le tir de charge

Maintenant que nous avons notre scénario de tir il nous faut ajouter un contrôleur nous permettant d'obtenir les résultats de notre tir de charge. Pour cela, nous allons ajouter le contrôleur « Rapport consolidé » en faisant un clic droit sur « Groupe d'unités » Æ Récepteurs Æ Rapport consolidé.

Concentrons-nous maintenant sur notre scénario de tir, nous voulons simuler 100 utilisateurs simultanés qui effectuent 20 fois le scénario de tir, sur une période de montée en charge de 10 secondes. Nous obtenons donc ceci dans la configuration de notre groupe d'unités [Fig.8].

Une fois le tir configuré, nous pouvons le lancer en allant dans le menu « Lancer » Æ Lancer.

Une fois le tir de charge terminé (le petit carré en haut à droite de la fenêtre JMeter s'éteint) vous devriez obtenir un rapport consolidé qui ressemble à ceci : [Fig.9].

Le nombre total de requêtes effectuées est de 6 000, ce qui est correct puisque nous avons défini 100 utilisateurs effectuant 20 fois le parcours de navigation, cela revient à visiter 2000 (100 * 20) chaque page du parcours de navigation.

Voici la description des colonnes du tableau:

- **Libellé :** le nom de l'élément testé, ici c'est l'URL de chacune des pages testées
- **# Echantillons** : Le nombre de fois que le test a été exécuté, pour l'élément testé
- **Moyenne :** La moyenne des temps de réponse pour l'élément testé
- **Min :** Le temps de réponse minimum pour l'élément testé
- **Max :** Le temps de réponse maximum pour l'élément testé
- **Ecart type :** La dispersion des échantillons autour de la valeur dans la colonne « Moyenne ». Plus l'écart type est grand, plus les données sont dispersées par rapport à la moyenne. Autrement dit, plus la valeur dans la colonne « Ecart type » est élevée, plus il y a de très bons et très mauvais temps de réponses. Cela permet de savoir si les résultats obtenus sont homogènes ou hétérogènes.
- **% Erreur :** Le pourcentage d'erreurs pour les pages testées.
- **Débit :** La bande passante utilisée, représentée par la formule suivante : nombre de requêtes effectuées/ temps total
- **Ko/sec :** La vitesse de transfert en Ko par seconde
- **Moy. Octets :** Le nombre moyen d'octets transférés

Conclusion

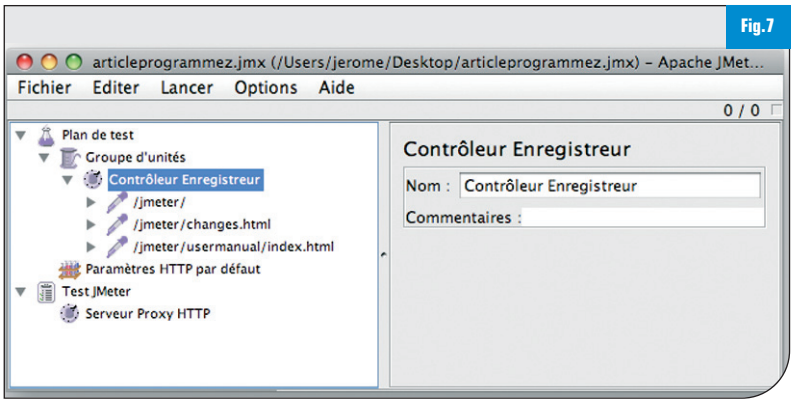
Nous avons vu comment monitorer et stresser une application. Avec ces deux composantes il vous sera désormais possible de tester votre site. Dans de prochaines étapes nous pourrions continuer l'analyse, chercher les goulots d'étranglements, mettre en place des améliorations d'architectures, de configuration et bien sûr, visualiser les améliorations grâce à ce que nous venons de voir.

A l'occasion du salon PHP qui aura lieu les 9 et 10 Novembre à la Villette, une conférence sur l'optimisation PHP mettra en pratique cette approche. Renseignez vous sur www.afup.org/forumphp/

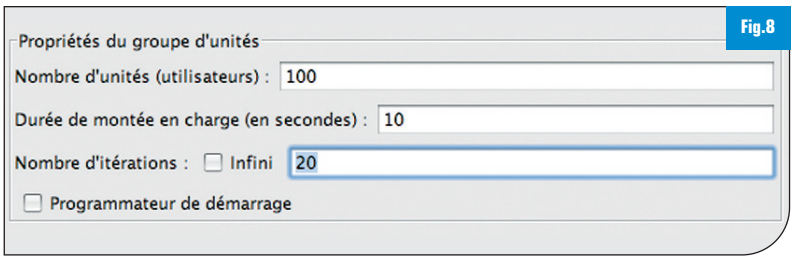
Si cet article vous a plu et que vous souhaitez une suite n'hésitez pas à faire remonter l'information et si vous souhaitez continuer la conversation, direction le forum PHP du site Programmez ou sur nos mails (cyril@php.net , jr@39web.fr)

Cyril **Pierre de Geyer** *est expert Open Source, co-auteur du livre PHP 5 avancé et certifié sur l'ETL Talend, il est l'un des fondateurs et ancien président de l'AFUP. Depuis dix ans il travaille dans la formation.*

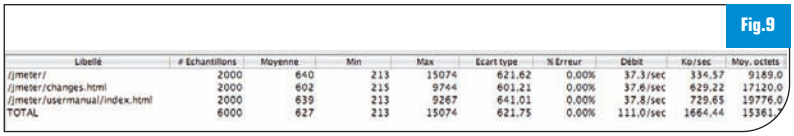
Jérôme **Renard** *est le fondateur de la société 39Web, société spécialisée dans les technologies internet LAMP soumises à de fortes charges.*



final



Configuration du tir de charge



Résultats

Installer et configurer PHP sur Windows

Depuis quelques années, PHP prend de plus en plus d'importance dans le développement Web. Comment faire tourner PHP sur une plateforme Microsoft ? Comment l'installer ? Quels sont les petits (et gros) pièges à éviter ? Il est évidemment impossible d'être exhaustif en quelques pages, mais cet article vous aidera à balayer les fondamentaux.

PHP se compose aujourd'hui de plusieurs grandes briques :

- Le runtime
- Les extensions
- Les frameworks divers

Le runtime est le binaire chargé d'intercepter vos requêtes et de les traiter. La version majoritaire actuelle est PHP 5.2, mais PHP 5.3 devient rapidement la norme, et il existe des différences non négligeables entre les deux environnements, particulièrement sur une plateforme Microsoft.

En effet, les binaires PHP sont composés de plusieurs sources (en gros, le runtime lui-même et les extensions), et pour les unifier il faut un compilateur adapté. Le premier compilateur utilisé fréquemment sous Windows pour produire un binaire PHP est Visual C 6, c'est-à-dire un produit qui a plus de 10 ans. Même si la qualité de ce compilateur n'est pas à prouver, force est de constater qu'il a vieilli : nous en sommes aujourd'hui à la version 9, qui apporte un certain nombre de nouveautés intéressantes pour PHP :

- Compilation 64 bits
- Optimisations de performance
- Meilleure sécurité

Pour n'en citer que quelques-unes. Comme nous sommes en période de transition, vous trouverez PHP 5.3 en plusieurs versions : des versions VC6, pour assurer la compatibilité avec des modules non encore disponibles sous VC9, et VC9.

Au-delà du compilateur, il existe également deux versions de PHP : l'une est thread-safe, et l'autre non. En d'autres termes, la version thread-safe a été conçue pour être ré-entrante, en protégeant certaines sections de code pour garantir qu'une exécution simultanée est impossible et que vous ne pouvez pas corrompre les données d'un utilisateur lorsqu'un autre utilisateur consulte votre site Web. Intéressant ? Oui, mais le prix à payer en performance n'est pas négligeable. Utiliser la version non-thread-safe peut sembler un peu inquiétant mais rassurez-vous, l'isolation des processus est assurée à un autre niveau : dans IIS.

Comment IIS exécute vos applications PHP

IIS a la possibilité d'exécuter PHP dans trois modes, et il est important de bien les comprendre :

PHP en processus CGI

Dans ce mode d'exécution, IIS lance un nouvel exécutable PHP pour chaque nouvelle requête. Ce type de fonctionnement est extrêmement lent, et est très rarement utilisé en production. Si votre serveur reçoit une très faible charge, c'est cependant un mode d'exécution offrant une très grande stabilité.

PHP en mode ISAPI

Cette approche a longtemps été la plus performante sous IIS (et Apache, par le biais de modphp). La raison de cette performance est aussi la cause de sa fragilité : toutes les requêtes PHP sont en effet envoyées dans le même binaire PHP, et ce binaire est directement chargé dans IIS. Si l'une de vos applications se comporte mal et fait planter le binaire PHP, l'exécutable dans lequel il vit va subir le même sort, et votre serveur web entier peut tomber. Cette approche impose d'utiliser la version thread safe de PHP, et peut dans certains cas apporter un peu plus de performance que la suivante, mais au détriment de la stabilité.

PHP en mode FastCGI

Le mode fastCGI est un raffinement de CGI : comme dans CGI, IIS passe le contrôle à un autre exécutable lorsqu'une requête PHP lui parvient. La différence est dans la gestion de l'autre exécutable : dans CGI, il est simplement détruit et un autre sera relancé à la requête suivante. Dans le cas de fastCGI, l'exécutable reste chargé en mémoire et servira plus tard à traiter une autre requête. Pour les curieux, la communication entre l'exécutable et IIS est assurée par des canaux nommés en mémoire (on peut aussi utiliser des canaux TCP mais la performance est moindre). Ce mode de fonctionnement permet de fournir des performances identiques à celles de la version ISAPI, en consommant un peu plus de mémoire mais avec une stabilité très supérieure. Vous pouvez dans ce mode utiliser les versions non thread safe de PHP.

En conclusion, les configurations que vous devez privilégier sont dans cet ordre :

- Version 5.3 de PHP, pour VC9, en mode non thread safe, et avec IIS en mode FastCGI.
- Version 5.2 de PHP, pour VC6, en mode non thread safe, IIS en mode FastCGI

Attention : lorsque vous fonctionnez en mode fastcgi, le processus php-cgi.exe reste en mémoire entre deux appels au serveur Web. Si vous êtes en développement et que vous modifiez le runtime PHP (ajout de nouvelles extensions par exemple), il est important de redémarrer IIS pour forcer l'arrêt et le redémarrage de tous les processus PHP qui pourraient vivre en mémoire. Pour relancer IIS, je vous suggère d'ouvrir un command prompt en mode administrateur et d'y exécuter la commande iisreset chaque fois que vous en avez besoin.

Comment procéder à l'installation de PHP sous IIS

Les pré-requis

Installer PHP sous IIS est relativement simple, et demande simplement de vérifier que quelque pré-requis sont bien respectés. Microsoft met à disposition un utilitaire d'installation que je vous recommande, le Web Platform Installer (téléchargeable sur http://www.microsoft.com/web). Cet outil permet d'automatiser l'installation de toutes les dépendances de PHP, ainsi que de PHP lui-même. Cependant, la version de PHP actuellement disponible sur WebPI n'est pas la plus récente (il s'agit de la version 5.2), et le chemin d'installation par défaut n'est pas celui que vous voudrez forcément utiliser (les binaires sont installés

dans Program Files, ce qui peut être gênant notamment si vous désirez apporter des modifications à PHP.INI). Il faut également noter que l'installation contenue dans WebPI déploie automatiquement un grand nombre d'extensions alors que vous voudrez probablement minimiser celles que vous mettez à disposition de vos applications PHP.

La solution que je vous recommande est de passer par WebPI pour les prérequis et de passer par le site http://windows.php.net pour l'installation de PHP en lui-même.

Pré-requis pour installer PHP sur une machine de production :

IIS+document par défaut+contenu statique+module de réécriture d'URL

CGI - FastCGI (et son hotfix pour IIS 7)

Pour installer ces pré-requis, vous pouvez utiliser ce lien :

http://www.microsoft.com/web/gallery/install.aspx?appid= IIS60;IIS51;StaticContent; defaultdocument;directorybrowse;httperrors;httplogging;loggingtools;requestmonitor;requestfiltering;statuscontentcompression;iismanagementconsole;cgi;FastCGIIIS6;fastcgibackport

Installation du runtime PHP

Pour installer PHP, je vous conseille de réfléchir un peu à votre environnement de production avant de vous lancer dans l'aventure. Les questions les plus importantes sont :

Quelles sont les permissions dont votre application a besoin sur le disque ?

- Quelles sont les extensions dont vous avez besoin pour fonctionner ?
- Quelle version de PHP comptez-vous utiliser ?
- Le serveur est-il dédié à un site PHP ou partagé par plusieurs sites ?

Si votre serveur doit héberger plusieurs applications PHP, je vous suggère de toutes les faire tourner avec la même version de PHP : c'est un peu plus de travail en recette mais beaucoup moins de risques en maintenance.

Une fois les décisions prises, passons à l'installation de PHP. La méthode que je vous recommande est de passer par l'installeur disponible sur http://windows.php.net. Ne prenez pas le zip à moins de savoir exactement ce que vous faites, car il installe un grand nombre d'extensions et ne paramètre pas FastCGI pour IIS.

L'installeur va vous guider pas à pas dans le processus d'installation et gérer pour vous trois aspects importants :

1 Renseigner les variables d'environnement permettant à PHP de savoir où trouver PHP.INI
2 limiter le nombre d'extensions et les configurer dans PHP.INI
3 Configurer FastCGI pour IIS

La checklist est donc la suivante :

a Choisissez votre version de PHP (je recommande chaudement les versions Non Thread Safe)
b Choisissez le chemin d'installation

Si vous comptez utiliser plus d'une version de PHP, je vous suggère de créer l'arborescence suivante :

Vous installerez la version principale de PHP dans \PHP\Default, en vous réservant la possibilité d'installer des binaires spécifiques dans un répertoire correspondant au nom du site que vous souhaitez personnaliser.

c Choisissez les modules que vous voulez installer. Moins vous en aurez, mieux vous maîtriserez la configuration de votre site. A noter que l'installeur proposé sur http://windows.php.net permet de modifier la liste des extensions installées par simple ré-exécution du MSI.
d Demandez l'installation de PHP pour FastCGI. Vous pourrez par la suite faire pointer d'autres sites vers d'autres versions de PHP, mais il est nécessaire de faire une première fois un certain nombre de tâches de configuration dans IIS, et c'est ce que permet cette option.

Paramétrage de PHP pour une bonne exécution sous Windows.

Ce paramétrage doit se faire à deux niveaux : d'une part dans IIS, pour que les requêtes soient correctement passées et que IIS puisse gérer quelques aspects spécifiques à PHP, et ensuite dans votre php.ini, qui doit lui aussi être adapté pour un bon fonctionnement sous IIS.

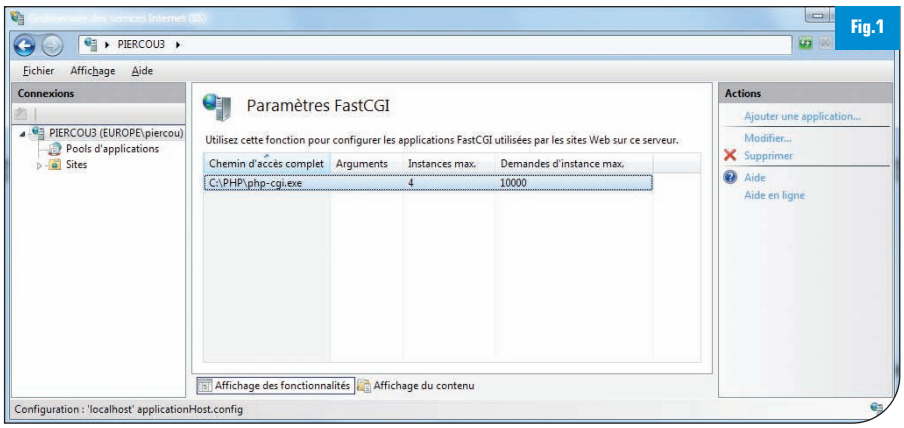


Fig.1

Le paramétrage d'IIS et FastCGI

Pour que PHP se comporte correctement, vous devez définir ou vérifier un certain nombre de paramètres définis dans IIS, et plus spécifiquement dans les paramètres fastCGI : [Fig.1]

Vous voyez dans la copie d'écran ci-dessus quelques-uns des paramètres définis pour l'exécution de PHP via FastCGI. Les paramètres importants sont les suivants :

Nombre max d'instances : c'est le nombre de processus php-cgi.exe que FastCGI est autorisé à exécuter simultanément. Vous pouvez mettre 0 pour laisser FastCGI optimiser ce paramètre en fonction de votre configuration

Demandes d'instance max : ce paramètre définit le nombre de requêtes acceptées par votre processus avant recyclage. Attention, PHP dispose de son propre système de recyclage de processus, et vous devez vous assurer que celui-ci ne prendra pas la main au mauvais moment. Pour ce faire, il vous faut modifier une variable d'environnement définie au lancement du processus php-cgi.exe par FastCGI : [Fig.2].

Vous voyez ici l'ensemble des paramètres vous permettant d'influer sur le comportement de php :

Analyser les modifications du fichier permet de définir le paramètre MonitorFileChanges, il sert à redémarrer les instances de php-cgi.exe en cas de modification du fichier php.ini (ce qui vous évitera de faire iisreset à chaque mise à jour).

Le nombre maximal de demandes d'instances doit être supérieur à PHP_FCGI_MAX_REQUESTS : ce dernier paramètre est celui que PHP utilise pour déterminer qu'il doit redémarrer, et vous ne voulez pas que PHP redémarre sans en avoir reçu l'ordre depuis IIS.

L'autre variable d'environnement est très importante : PHPRC est en effet l'une des

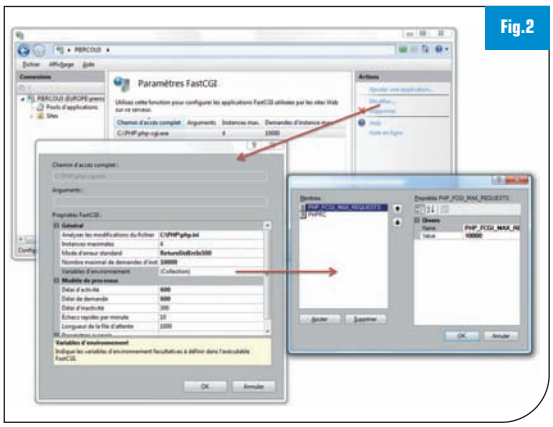


Fig.2

techniques que le runtime PHP utilise pour déterminer le chemin d'accès au fichier PHP.INI qu'il doit prendre en compte. Si vous désirez déployer en production plusieurs versions de PHP, c'est ce paramètre qui vous permettra de faire pointer chaque runtime PHP vers son PHP.INI.

Le paramétrage de php.ini

Ce fichier comporte le paramétrage fin de PHP, ainsi que la liste des extensions chargées au démarrage de PHP. Voici la liste des points à vérifier sur votre installation :

```
fastcgi.impersonate = 1
```

Ce paramètre permet à PHP de fonctionner sous le compte utilisé par IIS. Pour les accès anonymes sous IIS7 et IIS7.5 il s'agit du compte IUSR, et c'est à lui que vous devrez donner les permissions adéquates (par exemple pour utiliser la sécurité intégrée dans Sql S erver, ou accéder à des fichiers)

```
cgi.fix_pathinfo=1
```

Il s'agit d'une correction mineure des variables d'environnement.

```
cgi.force_redirect = 0
```

Pas ou peu de documentation sur ce paramètre, il est apparemment utilisé pour Apache pour mitiger un risque de sécurité

```
extension_dir = "…"
```

C'est l'emplacement où résident les extensions que vous désirez monter dans le runtime PHP.

Quel est le compte qui fait tourner votre runtime PHP ?

Attention, cette section est capitale pour comprendre la sécurité et les droits d'accès de votre application, et les bonnes pratiques sont bienvenues en ce domaine, particulièrement si vous hébergez plusieurs applications Web sur un environnement de production. Afin de simplifier un peu la discussion je vais me limiter aux versions récentes de Windows, les comptes de sécurité par défaut ayant pas mal évolué dans les dernières versions d'IIS.

L'identité de votre processus est déterminée par un ensemble d'étapes. Supposons qu'une requête arrive sur votre serveur, et voyons comment IIS va la traiter et déterminer qui l'exécute.

Le serveur accepte les accès anonymes

Dans ce cas, IIS va utiliser l'identité associée à l'utilisateur anonyme pour déterminer s'il a le droit de poursuivre son chemin jusqu'à la page (ou image) demandée. Le premier compte à connaître est celui qu'IIS tente d'attribuer à PHP-CGI.exe. Dans IIS, les applications Web sont exécutées dans des pools d'applications. Chaque pool d'applications a un paramètre important pour PHP : son identité. Les valeurs courantes de cette identité sont :

- LocalService, NetworkService, ou LocalSystem

Ce sont les comptes de service pré-définis habituels. Il existe cependant un autre type d'identité prédéfinie :

- ApplicationPoolIdentity

Il s'agit d'un compte nommé IIS APPPOOL\NomDuPool qui est créé et maintenu dynamiquement par le système. Choisir ApplicationPoolIdentity (c'est la valeur par défaut sous IIS 7.5) permet d'avoir automatiquement un compte de service distinct par pool d'application, et donc une meilleure isolation entre les différents sites web associés à ces pools.

Tout compte de votre choix

Vous pouvez bien évidemment attribuer un compte de votre choix.

Est-ce que ce compte est celui qui fait tourner mon processus PHP ?

Oui et non : en mode anonyme oui, en mode authentifié c'est une question de paramétrage

Le serveur refuse un accès anonyme

Dans ce cas, IIS et le navigateur vont négocier pour déterminer votre identité. Si cette identité est un compte Windows, PHP va prendre une décision en fonction du paramètre fastcgi.impersonate présent dans PHP.INI. Si cette valeur est à 1, l'identité qui exécute votre requête est celle de l'utilisateur. Si la valeur est nulle, c'est toujours le compte anonyme qui exécute php-cgi.exe. Attention, il est également possible de changer cette valeur dans les paramètres du site, auquel cas le compte défini pour le site sera utilisé en lieu et place du compte défini par le pool d'applications.

Quelques erreurs fréquentes

Attention, les pièges sont nombreux sur les extensions, en voici quelques-uns :

Mauvaise version de PHP.INI : veillez toujours à contrôler que PHP utilise bien le PHP.INI de votre choix, si ce n'est pas le cas, vérifiez la variable d'environnement PHPRC. Le symptôme : vous modifiez PHP.INI et vos modifications ne sont pas prises en compte.

Oubli du redémarrage d'IIS : si vous n'avez pas correctement configuré IIS pour qu'il redémarre automatiquement PHP lorsque vous modifiez le fichier php.ini (par défaut il n'est pas surveillé), les modifications de php.ini ne sont prises en compte qu'au lancement d'un nouveau php-cgi.exe. Symptôme : vous faites des modifications et vous testez, le comportement est variable (vous tombez une fois sur deux sur le php-cgi.exe qui a pris en compte vos modifications, et une fois sur deux sur le précédent).

Mauvaise version des extensions : si vous ne prenez pas la version de l'extension correspondant à votre binaire (version thread safe sur un binaire php non thread safe par exemple), son chargement ou son initialisation échouent. Symptômes : variés, depuis la page blanche jusqu'à un message d'erreur abscons.

```
open_basedir = "…"
```

Cette directive permet de restreindre les droits d'accès fichier du runtime PHP aux répertoires spécifiés (séparateur : point-virgule). Veillez à terminer le nom de chaque répertoire par un antislash.

Pour vérifier (et paramétrer) simplement les aspects PHP.ini, vous pouvez utiliser PHP Manager, un module d'administration pour IIS disponible sur codeplex.

Vous pouvez le télécharger (en version bêta) sur http://phpmanager.codeplex.com/

■ Pierre **Couzy**
Microsoft france

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.

www.solutions-logiciels.com

☐ **OUI**, *je m'abonne* (écrire en lettres capitales)

Envoyer par la poste à : Solutions Logiciels, service Diffusion, 22 rue rené Boulanger, 75472 PARIS - ou par fax : 01 55 56 70 20

1 an : 30€ *au lieu de 36€, prix au numéro (Tarif France métropolitaine) - Autres destinations : CEE et Suisse : 36€ - Algérie, Maroc, Tunisie : 36€ , Canada : 48€ - Dom : 45€ Tom : 60€*

6 numéros par an.

☐ M. ☐ Mme ☐ Mlle Société .....

Titre : ..... Fonction : ☐ Directeur informatique ☐ Responsable informatique ☐ Chef de projet ☐ Admin ☐ Autre .....

NOM ..... Prénom .....

N° ..... rue .....

Complément .....

Code postal : |_|_|_|_|_| Ville .....

Adresse mail .....

☐ *Je joins mon réglement par chèque à l'ordre de SOLUTIONS LOGICIELS* ☐ *Je souhaite régler à réception de facture*

Editeur : K-Now sarl au cap de 8000€ siège social : 21 rue de Fécamp, 75012 Paris

SL016

application dans la catégorie des outils de gestion de contenus.
Par ailleurs, le système de blogs très poussés, propose la possibilité de créer, modifier, classer des contenus sur un ou plusieurs serveurs. Cependant une complexité existe, qui peut poser problème dans le déploiement de certains projets qui ne se limitent pas qu'à la gestion d'un blog. Bien sûr, un large choix de plugins est disponible, pour permettre d'ajouter de nouvelles options et des configurations, tout en gardant une interface simplifiée. La version 3, sortie en mai 2010, a permis de centraliser les thèmes et les plug-ins pour simplifier les installations et les configurations. Par ailleurs, vous pourrez à partir d'une même interface gérer plusieurs sites internet.

CMS Award

Autre bon indicateur, que la majorité des utilisateurs ou des développeurs oublient de regarder : les CMS Award. Il s'agit d'un événement annuel regroupant de nombreux concours et ouvert au marché du CMS, tous langages confondus. Ce concours se distingue par différentes catégories spécifiques mais aussi générales comme :

- Les CMS open source
- Les blogs
- Le projet prometteur
- Les applications Open source E-commerce
- Les interfaces et applications graphiques.

Depuis sa création, les vainqueurs toutes catégories confondues ont vu des CMS remporter plusieurs fois ce prix comme Mambo, Joomla, Drupal. Les résultats 2010 seront dévoilés à la fin de l'année.

Futur

L'utilisation du CMS est assez jeune, même si de nombreuses applications existent depuis fort longtemps, le système s'est démocratisé depuis ces dernières années, grâce à un énorme effort au niveau de l'ergonomie et surtout pour répondre aux attentes des utilisateurs. Le concept se veut évolutif, permettant généralement d'insérer facilement l'ajout de plugins ou d'extensions pour s'ouvrir vers l'extérieur. Par ailleurs pour souder aussi bien les développeurs que les utilisateurs, il est important de prévoir l'utilisation des multi-langues.
Du côté de la structure des CMS, elles sont toutes différentes selon les années de parution et les versions. La tendance s'oriente vers le xHTML et le XML, en s'éloignant du HTML classique. Cependant, avec l'arrivée du HTML 5, certains CMS, non cités dans l'article pourraient revenir sur le devant de la scène.

Hébergement

Lors de la sélection d'un CMS, il est aussi important de tenir compte de la tenue de l'hébergeur. L'ensemble des gestionnaires de contenus présentés dans l'article, tournent dans un environnement AMP (Apache, MySQL, PHP) et peuvent fonctionner chez différents hébergeurs proposant les types suivants :

- Mutualisé
- Dédié
- Les deux.

Ces critères sont à prendre à compte, car le coût financier peut orienter la sélection. Cependant, des astuces et configurations spécifiques (cache par exemple), adaptés au cas par cas peuvent vous aider dans votre sélection.

Tableau récapitulatif

Avant de vous lancer dans un CMS, il est important d'avoir une petite synthèse des CMS présentés dans l'article, surtout si vous n'êtes pas développeur. (voir tableau ci-dessous)

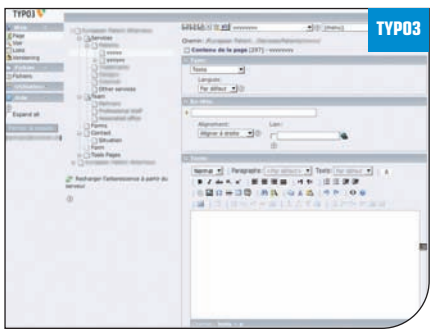
Conclusion

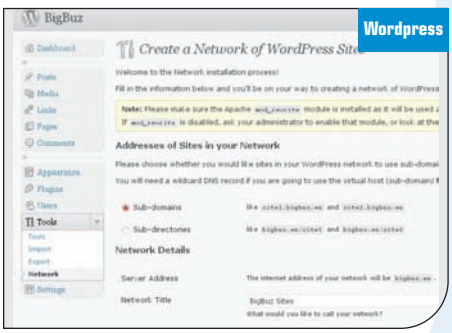
L'ensemble des CMS proposent de nombreuses fonctionnalités propres, tout en gardant le même principe qu'une application Framework. Grâce aux différents plugins, vous pouvez insérer facilement un environnement CRM, ERP, E-commerce, et ainsi répondre à des besoins spécifiques.
Pour finir, de nombreux projets sont lancés autour du CMS et parfois le résultat attendu par les utilisateurs et les internautes peut décevoir, c'est pourquoi il est utile de se faire accompagner par des professionnels.

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

	Installation et configuration	Communautés	Performance et Robustesse	Evolutivité	Documentation
Drupal	***	***	**	***	*
eZ Publish	**	**	***	***	***
Joomla	***	***	**	***	*
Magix CMS	***	**	***	***	***
SPIP	***	**	**	***	*
Typo3	**	**	***	***	*
Wordpress	***	***	**	***	*

Symbole : * Faible ** Moyen *** Facile





de pilote approprié (MySQL, PostgreSQL, Microsoft SQL Server, Oracle). Par ailleurs, il supporte le clustering, et respecte la séparation du contenu et de la présentation par le stockage sous forme XML de tous les contenus. La communauté est assez présente et internationale, permettant de répondre aux différentes attentes des utilisateurs. Depuis ces dernières semaines, de nombreux points et changements ont été dévoilés. Tout d'abord la sortie d'une nouvelle version 4.4, qui autorise le développement et l'évolution du coeur de celui-ci par la communauté, car jusqu'à présent, il était réservé aux employés de la société. Concernant la version 5 (très attendue), son coeur eZ Components à été libéré au profit de la communauté Apache sous le nom de Zeta components.

Joomla

Joomla est un CMS basé sur le CMS Manbo (incontournable dans les années 2000/2005). La version actuelle est soutenue par une importante communauté, qui lui assure une énorme notoriété envers le grand public. Il propose un ensemble d'outils très large en natif qui sont : la gestion de Flux RSS, un éditeur WYSIWYG, la gestion de news, sondages, blogs, une version imprimable avec une gestion de droits pré-configurés.

Un petit temps d'adaptation est nécessaire pour exploiter l'ensemble des fonctionnalités mais cela se réalise très rapidement. Un des nombreux points forts, concerne le templating sous la forme de <table> ou . Bien sûr, vous trouverez un ensemble de plugins déjà installés et paramétrés, et ainsi vous pouvez obtenir un site professionnel en quelques clics, avec très peu de connaissance PHP, tout en gardant la possibilité d'en rajouter suivant vos besoins. La prochaine version 1.6 est actuellement attendue car elle proposera une nouvelle gestion des autorisations (ACL) et aussi l'ouverture aux bases de données PDO.

Magix CMS

Il s'agit d'un nouveau concept dans la famille des CMS en PHP. La dernière version est sortie en 2010 et soutenue par la société « Web solution Way », sur un même constat : il est important de permettre à une personne de créer et de modifier un contenu. Cependant le référencement est souvent mis de côté ou en deuxième partie. Ici le référencement est privilégié pour permettre une meilleure visibilité auprès des moteurs de recherche d'une façon automatique pour le contenu du site et des images.

Par conséquent, il bénéficie de l'ensemble des dernières technologies pouvant exister sur le web, c'est-à-dire l'utilisation de la technologie Ajax pour permettre le « drag and drop » et répondre à la norme Web 2.0. L'interface s'utilise comme une boîte à outils pour faciliter le travail des développeurs, des intégrateurs. Cette boîte à outils contient un module de liens, une newsletter, la gestion d'actualités, des galeries d'images.

SPIP

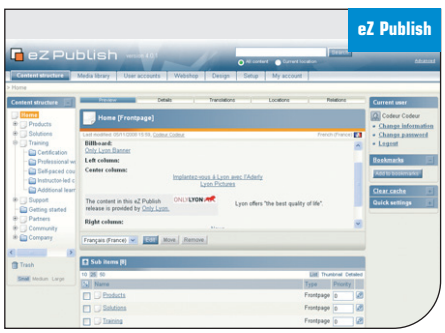
Le CMS Spip possède une mascotte « un écureuil volant » et il s'agit d'un produit français. C'est un des premiers à avoir proposé un environnement complet en français, associé à une messagerie interne. Il supporte plusieurs formats de bases de données MySQL, PostgreSQL, SQLite. Bien sûr, le back est structuré le plus simplement possible et permet à l'ensemble des utilisateurs une prise en main rapide de celui-ci. L'intégration d'un environnement multi-langue a permis de l'utiliser sur des projets internationaux. Depuis la version 2 et actuellement 2.1, sortie en 2010, SPIP peut-être utilisé sur plusieurs serveurs et configurations différentes. Son environnement a intégré Ajax, permettant d'avoir un outil Web 2.0 et l'incorporation d'une gestion de Plugins. Enfin, l'outil vous permettra de déployer facilement un projet web avec un minimum de connaissances.

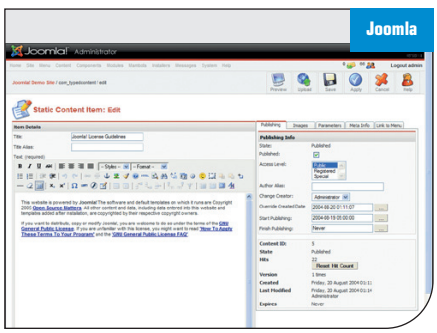
Typo3

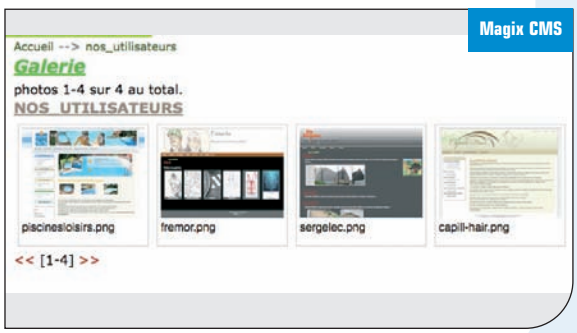
Typo 3 est un autre CMS, qui pourrait répondre à votre attente, si vous devez travailler sur un projet web nécessitant de gros besoins, tout en associant de nombreux liens. Son approche est celle d'un Framework mais avec une interface CMS. Il va vous permettre de développer très rapidement un site web. La version actuelle (4.4.2) sortie en Août 2010, propose une architecture sous forme de plugins, une gestion de multi-site, un processus de publication garantissant la qualité du code et de la sécurité. Le template HTML permet d'utiliser les différentes normes actuelles. Le plugin « TemplaVoila », indispensable pour des utilisateurs non spécialisés, permet de faciliter l'utilisation et la modification des pages, rendant ainsi les contenus flexibles. La prochaine version 5, sera basée sur le Framework Flow3 pour répondre aux attentes des utilisateurs

Wordpress

A l'origine, Wordpress proposait juste la création et la gestion de blogs individuels. Au fil des années, les choix de développement ont positionné cette







Bien choisir son CMS

L'année 2010, peut être considérée comme une année charnière pour les CMS PHP. Parmi les centaines de CMS existants, le choix devient rapidement très difficile. A travers cet article nous allons tenter de vous aider à y voir clair en vous présentant un tour d'horizon de l'ensemble des outils les plus connus.

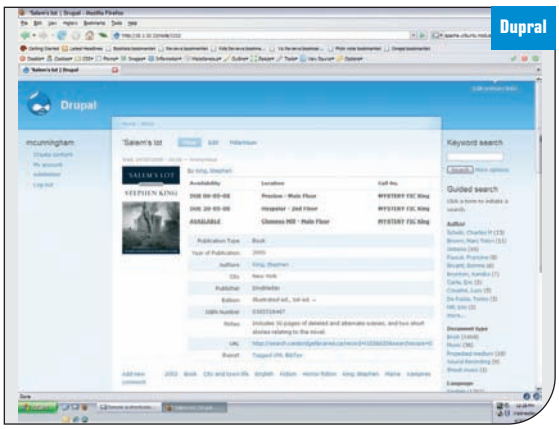
La dénomination de CMS signifie « Content Management System », c'est-à-dire : gestionnaire de contenu. Le but premier de cette catégorie est de vous permettre de gagner du temps au niveau du développement, mais aussi au niveau de la maintenance et de faciliter la migration vers les évolutions futures.

Travail préliminaire

Avant de vous lancer dans l'utilisation d'un CMS, certains critères au niveau des fonctionnalités doivent être pris en compte. Lorsque vous choisissez ce type d'outils, c'est pour vous permettre de créer, modifier et sauvegarder un contenu au format HTML en respectant les standards W3C, d'une façon simple et efficace par l'intermédiaire d'une interface administrative. Celle-ci va vous permettre de gérer dynamiquement un contenu et d'obtenir le résultat instantanément.

Par ailleurs, le CMS doit vous permettre d'ajouter des extensions supplémentaires, tout en gardant la possibilité de développer et concevoir des modules ou extensions spécifiques suivant vos besoins.

Les points principaux lors de la sélection d'un CMS, qui doivent être pris en compte sont :



- La qualité, quel que soit votre besoin
- Il doit être intuitif
- Un back normalisé, logique et organisé
- Il doit posséder les fonctions utiles et non le superflu inutile
- Une prise en main facile et simple
- Il doit posséder un éditeur RTE (Rich text Editor) = WYSIWYG
- Pas d'attente lors de l'affichage d'une page simple
- La possibilité de personnaliser suivant son envie
- Une bonne documentation
- La présence d'une communauté

Par ailleurs, certains points négatifs, pouvant poser des problèmes, sont aussi à prendre en considération :

- Retenir le CMS qui vient de sortir car il n'est pas sûr de répondre à votre besoin
- Un CMS trop technique
- Un CMS trop volumineux (poids des fichiers et de l'espace disque)
- Choisir un CMS trop petit et non évolutif
- Vérifier de ne pas se trouver seul pour l'utiliser
- Permettre aux développeurs de choisir les CMS pour le client suivant leurs envies.

Cependant, si vous essayez de suivre l'ensemble de ces points en consultant individuellement les différents CMS du marché, les réponses seront toujours positives, chacun vous expliquant qu'il est le meilleur. C'est pourquoi dans cet article, nous vous proposons une approche des principaux CMS

Les principaux CMS

Lorsque vous avez défini vos différents besoins, moyens matériels et humains et surtout délais, vous pouvez vous orienter vers le choix d'un des CMS qui vous sont présentés ci-dessous par ordre alphabétique.

Drupal

Le CMS Drupal existe depuis très longtemps, et s'est fait connaître sur le marché grand public voici quelques années. Il s'agit d'un projet Open Source accessible à l'ensemble de la communauté et soutenu par de nombreuses entreprises, proposant une évolution dynamique.

De nombreux points sont mis en avant autour de ce CMS, comme la possibilité de proposer un espace personnalisé pour chaque internaute et un seul back-end personnalisable pour gérer plusieurs types de sites. Par contre, l'organisation de contenu s'effectue par l'intermédiaire d'une taxonomy et le partage de données sous la forme de droits et de rôles. Cependant, la prise en main nécessite un temps d'apprentissage pour exploiter l'ensemble des différentes possibilités offertes.

La version de base (6.x) permet de réaliser simplement un blog, avec une configuration minimale. Mais il offre la possibilité de gérer de nombreux plugins (5 000 modules). La prochaine version exploitera les différentes fonctionnalités de PHP 5 avec l'incorporation de PDO, des plug-ins supplémentaires seront directement intégrés dans le coeur du CMS, ce qui permettra une configuration d'hébergement plus importante.

eZ Publish

eZ Publish, soutenu par une société eZ System, est un des tous premiers CMS proposant un environnement de prise en main très rapide car il peut-être aussi bien utilisé sous la forme de CMS, que d'un Framework. Il a pour vocation de répondre aux différentes attentes de gros projets web car il a été le premier à proposer une couche d'abstraction pour l'accès aux bases de données par l'intermédiaire

SOA facile avec SCA

Comment simplifier le développement d'applications SOA tout en se donnant un cadre architectural ? SCA et notamment FraSCAti que nous utiliserons apportent des réponses à ces préoccupations.

Ecrire des applications SOA, avec de nombreux services web, n'est pas toujours chose aisée. Notamment, la mise en œuvre de services web (WS, REST, etc.) demande du temps et surtout du code technique en plus de vos classes métiers. Que diriez-vous de n'écrire que le code métier et simplement spécifier dans un fichier XML les services que vous voulez exposer sur le web ? SCA rend ceci possible ! Mais ce n'est pas le seul avantage, SCA vous permet aussi de bénéficier d'un cadre architectural pour vos applications orientées services. Enfin, il permet de mixer des applicatifs utilisant des technologies différentes (bundle OSGi, Java, scripts, BPEL, etc.) et des protocoles de communication hétéroclites (SOAP, HTTP, JSON-RPC). Dans cet article, nous ne reviendrons pas sur les fondamentaux de SCA, présentés dans le numéro 110.

Présentation et installation de FraSCAti

Il existe plusieurs implémentations des spécifications SCA (OW2 FraSCAti, Apache Tuscany, IBM WebSphere, etc.). Nous utiliserons FraSCAti, une plate-forme open-source du consortium OW2. Elle ne supporte pas tous les langages et protocoles spécifiés pour SCA (focus sur les technologies Java) mais, en contrepartie, fournit plusieurs fonctionnalités avancées, telles que le support de composants SCA réflexifs (permettant le changement de configuration à chaud des assemblages), des protocoles d'accès originaux comme UPNP, JNA ou encore des outils qui nous aideront à observer l'état et administrer les assemblages de composants pendant leur exécution. FraSCAti est téléchargeable sous la forme d'une archive zip sur le site du projet (http://frascati.ow2.org). Pour installer la plateforme, il suffit d'extraire les fichiers de l'archive.

SCA, cadre architectural

Notre exemple fil rouge, MyWeather, consiste en une orchestration de plusieurs services. Dans un premier temps, nous allons interroger un compte twitter pour récupérer la localisation d'une personne, ensuite nous interrogerons un service météo afin d'obtenir le temps pour ce lieu. SCA propose un langage d'architecture décrit en XML qui permet de construire des applications par assemblage de composants, le composant représentant une fonctionnalité implémentée généralement par une classe. Un assemblage SCA est comparable à la notion de contexte d'application dans Spring, les beans étant assimilés à des composants SCA.

Il est possible de générer le descripteur d'architecture (fichier *.composite) en utilisant l'éditeur STP/SCA d'Eclipse, mais nous allons ici modéliser notre exemple directement en XML [Figure 1.]. Pour cet assemblage (ou composite) SCA nous devons réaliser deux composants :

- le composant Decoder qui traduit les messages XML renvoyés par le service météo,
- le composant Orchestration qui utilise les services twitter, météo et le composant Decoder. Il définit une propriété SCA utilisée pour configurer l'identifiant du compte twitter.

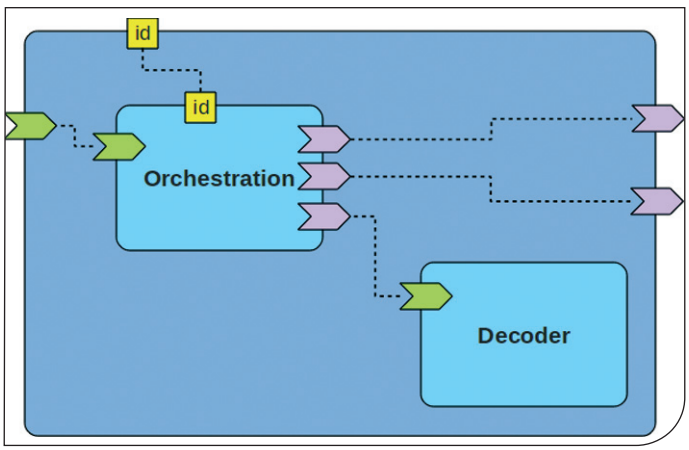


Figure 1: Implémentation du composant de décodage

SCA, vecteur d'interopérabilité et d'intégration technologique

SCA n'impose pas de langage particulier pour ses composants. Il est possible d'implémenter des composants avec le langage Java ou Scala, des scripts Ruby ou JavaScript, ou encore d'utiliser des composants Spring. Cette flexibilité se retrouve pour la création ou l'utilisation de services pouvant être réalisés par le biais des technologies SOAP, HTTP, RMI, JMS, etc. Les nombreux langages et protocoles d'accès supportés par les implémentations de SCA présentent cette technologie comme solution efficace pour faciliter l'intégration de composants logiciels existants.

Implémentation d'un composant SCA

Nous allons, dans un premier temps, nous intéresser à l'implémentation de notre composant de décodage qui sera utilisé pour traduire les réponses du service météo, encodées au format XML. Notons que ce composant ainsi que les objets JAXB sont nécessaires car le WSDL du service décrit le type de retour de ces méthodes comme une chaîne de caractères et non le type réel (liste de Locations).

Le composant Decoder, développé en Java, fournit une méthode 'decode'. Nous commençons par définir une interface Java, nommée 'Decoder', qui représentera le contrat du service. Cette interface a la signature suivante : Locations decode(String message);

On nommera son implémentation DecoderImpl :

```
public Locations decode(String message) {
  InputStream is = new ByteArrayInputStream(message.getBytes());
  try {
    JAXBContext context = JAXBContext.newInstance(Locations.
class);
    Unmarshaller unmarshaller = context.createUnmarshaller();
    return (Locations) unmarshaller.unmarshal(is);
```

```
  } catch (JAXBException e) {
    System.err.println("Unable to decode server message.");
    return null;
} }
```

Cette méthode permet de décoder (via JAXB) le message XML renvoyé par le service météo et donnant la liste des villes pour lesquelles le service météo est disponible. Cette liste est utilisée par le composant d'orchestration pour connaître l'ensemble des villes où les informations météo sont accessibles.
Notons qu'un composant SCA ne peut exposer un service que s'il fournit une interface décrivant ses méthodes (aussi appelée contrat). Ici, le contrat est une interface Java, mais SCA permet aussi de définir les contrats via des descripteurs WSDL.

SOA simplifiée

Grâce à SCA, nous allons très facilement utiliser ou réaliser des services distants sans nous soucier des détails techniques propres aux protocoles d'accès. Nous illustrerons l'intégration et l'exposition de services avec SCA via la création d'un composant d'orchestration qui proposera la mise à jour des informations météorologiques pour des utilisateurs du service twitter.
Pour obtenir les informations relatives à un compte utilisateur, twitter propose une API conforme au principe d'architecture REST (REpresentational State Transfer). Le service de météo est lui, accessible via le protocole SOAP. Nous allons accéder à ces services de nature différente via les connecteurs REST et Web Service fournis par la plateforme FraSCAti. L'utilisation d'un connecteur avec SCA est non intrusive, elle n'impacte pas l'implémentation même du composant. C'est FraSCAti qui se chargera de la création des stubs/squelettes en charge des connexions distantes.

Accès à un service REST : twitter

Nous passons à l'intégration des services twitter et météo afin de les rendre utilisables par notre orchestration. Pour rappel, la technologie SCA propose l'intégration des services via de nombreux protocoles d'accès dont l'utilisation est spécifiée via le descripteur d'architecture. Par exemple, pour accéder aux méthodes exposées par twitter nous sélectionnerons le binding REST en ajoutant, dans la description de notre composite SCA, la balise XML suivante :

```
 <frascati:binding.rest uri="http://twitter.com"/>
```

L'attribut URI permet d'indiquer l'adresse de la ressource web à accéder via REST, dans notre cas "http://twitter.com". Pour permettre à notre composant, implémenté en Java, d'accéder aux informations du profil utilisateur, Nous devons écrire l'interface qui reflètera le service utilisé.

```
import javax.ws.rs.*;

public interface Twitter {
  @GET
  @Path("/users/show/{id}.xml")
  User getUser(@PathParam("id") String id);
}
```

Nous utiliserons la méthode "getUser" qui permet de récupérer les informations contenues dans le profil utilisateur. Cette méthode prend en paramètre l'identifiant de l'utilisateur. Dans le cas d'un service REST, nous devons utiliser les annotations de l'API JAX-RS afin de préciser le chemin d'accès, la commande HTTP, et les paramètres utilisés pour accéder à la ressource. L'appel à la méthode "getUser" renvoie un objet de type User reflétant l'architecture de la ressource REST reçue. La classe User comporte les annotations de l'API JAXB afin de réaliser la correspondance entre les tags XML et les champs de l'objet. Le code java pour la classe "User" est donné ci-dessous :

```
import javax.xml.bind.annotation.*;

@XmlRootElement
public class User {
    public String id;
    public String name;
    public String screen_name;
    public String location;
    ...
}
```

La valeur de l'attribut "location" de l'objet "User" contient les informations de localisation renseignées dans le compte twitter. Nous pouvons passer à l'intégration du service météo.

Accès à un Web Service (SOAP) météo

Le service météo que nous utilisons est accessible via le protocole SOAP, il est publié à l'adresse "http://www.webservicex.net/globalweather.asmx". Ce service web présente deux opérations : "GetCitiesByCountry" qui permet de lister les villes où les informations sur la météo sont disponibles et "GetWeather" pour connaitre la météo d'une ville. Pour utiliser ce service nous allons faire appel au connecteur SOAP de la plateforme FraSCAti. Dans notre descripteur d'assemblage, nous décrivons l'accès à ce service via une référence comportant un binding web service. Le service météo est alors rendu accessible pour les composants SCA de notre assemblage, et donc accessible pour le composant d'orchestration. Dans le descripteur d'assemblage SCA, cette liaison se traduit par la balise suivante :

```
<binding.ws
  wsdli:wsdlLocation="http://www.webservicex.net/globalweather.
asmx?wsdl"
  wsdlElement="http://www.webservicex.net#wsdl.port(GlobalWeather
/GlobalWeatherSoap)" />
```

La présence des attributs wsdlLocation et wsdlElement permettent à la plateforme de retrouver le descripteur de l'interface WSDL et le port d'accès au service web. Pour rendre le service météo utilisable par le composant d'orchestration, nous devons générer l'interface Java reflétant ce service à partir de son descripteur WSDL. Pour réaliser cette opération FraSCAti propose la commande wsdl2java :

```
% frascati wsdl2java -u http://www.webservicex.net/globalweather.
asmx?wsdl -o src/generated
```

Elle permet d'obtenir l'interface "GlobalWeatherSoap" dans le dossier des sources Java. L'interface obtenue est décorée avec les

annotations de l'API JAX-WS, indiquant le nom des opérations, paramètres et messages proposés par le service web. C'est via cette interface que l'on pourra interroger le service météo.

Réalisation de l'orchestration

Nous avons réalisé le composant de décodage des messages, puis écrit/généré les interfaces qui permettent d'interagir avec les services externes : twitter et météo. Nous pouvons, à présent, nous occuper de l'implantation du composant d'orchestration qui se chargera d'agréger les informations récupérées des services externes et de décoder les informations de localisation. Nous commençons par créer la classe "Orchestration" suivante :

```
public class Orchestration {
  @Reference protected Twitter twitter;
  @Reference protected Decoder decoder;
  @Reference protected GlobalWeatherSoap weather;
  @Property protected String userId;
    ...
}
```

Dans notre implémentation, nous utilisons deux annotations spécifiques à SCA. L'annotation "@Reference" précise les attributs de la classe permettant d'utiliser d'autres services, internes (ici le decoder) ou distants (twitter et météo dans notre cas). L'injection d'une référence vers un autre service ou un autre composant est réalisée par la plateforme SCA. La seconde annotation "@Property" définit une propriété configurable du composant, la valeur de cette propriété sera définie dans le descripteur d'assemblage SCA.
Il ne nous reste qu'a implémenter la méthode d'orchestration que nous nommerons "getWeatherForUser()". Nous allons d'abord appeler le service twitter et vérifier que notre utilisateur a renseigné la localisation dans son profil.

```
public String getWeatherForUser() {
 User user = twitter.getUser(userId);
 if (user.location.equals("")) {
 System.err.println("The user " + userId + " did not publish his
 location");
 return "N/A";
    } ...
```

Nous supposons que l'attribut de localisation comporte le nom de ville et du pays séparés par une virgule, que nous récupérons dans les variables "cityName" et "countryName"

```
 ...
 String[] locations = user.location.split("[\\s]*,[\\s]*", 2);
 String cityName = locations[0];
 String countryName = locations[1];
 System.out.println("User '" + userId + "' is living in " + cityName
   + " (" + countryName + ")"); ...
```

On récupère la liste des villes proposées par le service météo en utilisant la référence correspondante : "weather", puis on appelle le composant de décodage. On obtient alors une instance d'objet "Locations".

```
 String cities = weather.getCitiesByCountry(countryName);
 Locations l = decoder.decode(cities);
 ...
```

Enfin, on compare la liste des villes avec la localisation de l'utilisateur. S'il y a correspondance, on appelle de nouveau le service météo pour obtenir les conditions météo détaillées :

```
 String result = "";
 if (l != null) {
   boolean done = false;
   for (Location loc : l.locations) {
     if  (loc.city.value.toLowerCase().contains(cityName.to
LowerCase())) {
       result += "Current weather in " + loc.city.value + ":\n";
       result  +=  weather.getWeather(loc.city.value,  country
Name) + "\n";
       done = true;
   } }
   if (!done) {
     System.err.println("Unable to find '" + cityName
       + "' in available cities of the weather service");
 } }
 return result;
```

Descripteur d'assemblage

Nous avons implémenté les différents composants et interfaces nécessaires pour réaliser notre orchestration. Nous pouvons maintenant décrire l'architecture de notre application SCA. Le descripteur d'architecture appelé "composite" permet de construire notre application en définissant les composants, leurs liaisons, les propriétés de configuration ou encore les connecteurs utilisés.
L'écriture d'un composite commence avec l'élément XML "composite" qui mentionne obligatoirement un attribut "name". Notez également la définition des espaces de nommages via l'attribut "xmlns", qui seront ensuite utilisés dans la description :

```
<composite name="twitter-weather"
 xmlns="http://www.osoa.org/xmlns/sca/1.0"
 xmlns:wsdli="http://www.w3.org/2004/08/wsdl-instance"
 xmlns:frascati="http://frascati.ow2.org/xmlns/sca/1.1">
....
```

On définit ensuite les composants SCA. Dans un premier temps, nous ajoutons le composant responsable du décodage des messages XML. Les lignes suivantes permettent de nommer le composant et de lui associer la classe d'implémentation Java écrite précédemment.

```
 <component name="decoder">
   <implementation.java class="org.ow2.frascati.examples.twitter
weather.lib.DecoderImpl" />
 </component>
```

Nous ajoutons le second composant, responsable de l'orchestration. Outre la classe d'implémentation, nous décrivons les références vers les services nommés twitter et weather, ainsi que vers le composant de décodage. Nous définissons également la propriété

té "userId" qui utilisera la valeur définie dans la propriété nommée "userId" du composite (réutilisation de propriétés).

```
<component name="orchestration">
 <implementation.java class="org.ow2.frascati.examples.twitter
weather.lib.Orchestration"/>
 <reference name="twitter"/>
 <reference name="weather"/>
 <reference name="decoder" target="decoder"/>
 <property name="userId" source="$userId"/>
 </component>
```

Finalement, on décrit les références vers les services en spécifiant les connecteurs utilisés et on définit une valeur pour la propriété userId au niveau du composite.

```
 <reference name="twitter" promote="orchestration/twitter">
   <frascati:binding.rest uri="http://twitter.com"/>
 </reference>

 <reference name="weather" promote="orchestration/weather">
   <binding.ws wsdli:wsdlLocation="http://www.webservicex.net/
globalweather.asmx?wsdl"
     wsdlElement="http://www.webservicex.net#wsdl.port(Global
Weather    /GlobalWeatherSoap)" />
 </reference>

 <property name="userId">userid</property>
</composite>
```

Créer un service web

Pour exposer notre service TwitterWeather et le rendre accessible via SOAP, nous allons ajouter une interface le décrivant. Nous nommerons cette interface "TwitterWeather". Notez l'utilisation de l'annotation @Service pour indiquer que l'interface Java sera utilisée comme contrat pour le service SCA. Voici l'interface qu'implémentera alors la classe Orchestration.

```
@Service
public interface TwitterWeather {
 String getWeatherForUser();
}
```

On ajoute ensuite la définition du service au niveau du composite.

```
<service name="tw" promote="orchestration/TwitterWeather">
 <binding.ws uri="http://localhost:8080/TwitterWeather"/>
</service>
```

Le service "tw" sera ainsi accessible soit via un appel à partir d'un client SOAP, soit localement. Il ne nous reste plus qu'à compiler et exécuter notre intégration de service implémentée avec la technologie SCA. Pour créer le jar de l'application, nous appelons FraSCAti avec la commande de compilation :

```
% frascati compile src myWeather
```

L'exécution est ensuite réalisée grâce à la commande "run". On précisera les noms du service (option –s) et de la méthode (option –m) à appeler ainsi que le chemin vers le jar de l'application.

```
% frascati run myWeather -libpath myWeather.jar -s tw -m get
WeatherForUser
```

Conclusion

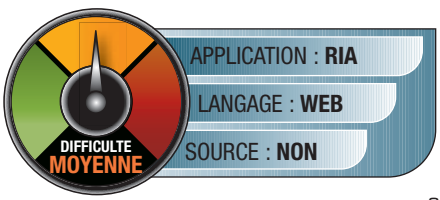
Nous avons illustré que SCA permet de construire très rapidement des architectures orientées services pouvant utiliser différentes technologies de communication. Nous avons créé une petite application, en quelques lignes de code, réalisant une orchestration entre un service SOAP et une ressource REST. Toutefois SCA et FraSCAti ne se limitent pas à cet usage. Dans un prochain article nous irons plus loin avec FraSCAti. Nous utiliserons ses capacités réflexives et son interface d'administration puissante pour montrer, avec un script ou en quelques clics, qu'il est possible de modifier le comportement d'une application SCA, de modifier sa configuration ou d'exposer les services d'une application encore plus rapidement.

Site : http://frascati.ow2.org

■ Christophe **Demarey** et Damien **Fournier**.

Vaadin : un RIA serveur basé sur GWT

Un framework RIA de plus ? Oui, mais basé sur un seul langage (100% Java) côté serveur, et ne nécessitant aucun plugin côté client puisque son rendu se fait avec les standards du web : HTML, Javascript et CSS. Comme GWT ? Vaadin se base sur GWT; mais à la différence de ce dernier, il n'y a pas de compilation pré-déploiement et il est orienté côté serveur. Ces deux derniers points vous intriguent ? Nous allons vous aider à faire vos premiers pas avec Vaadin.



Vaadin c'est le nom de la daine (la femelle du daim) du dieu finlandais Seppo Ilmarinen. C'est aussi le nom choisi en 2009 par la société finlandaise IT Mill quand elle a rebaptisé son produit phare IT Mill Toolkit. Créé en 2002 comme produit interne à leur société, open-sourcé sous licence Apache en 2007, Vaadin permet de construire des applications RIA à partir de composants graphiques. Le projet a depuis gagné en visibilité avec l'intégration de GWT, l'investissement de Monty (Michael Widenius, créateur de Mysql) et son intégration dans LifeRay.

PRINCIPES GÉNÉRAUX

- Full-Java : non au polyglottisme. Plus de fichiers de configuration en XML, ni de templates HTML, ni de Javascript, mais un seul langage (Java) qui facilite et simplifie le développement.
- Orienté serveur : la logique applicative est totalement exécutée côté serveur laissant ainsi moins de failles de sécurité possibles. Toute action côté client provoquera un appel côté serveur (avec potentiellement une mise à jour de données). Même l'interface graphique sera construite côté serveur et transmise par sérialisation au client. Tout ceci entraîne une charge réseau un peu plus importante mais qui reste modeste étant donné le minimum d'informations transmises.
- GWT : GWT est utilisé comme moteur de rendu côté client pour sa multicompatibilité avec les navigateurs.
- Communication Client - Serveur : les changements d'état côté serveur sont envoyés au client avec le langage UIDL (User Interface Definition Langage), basé sur Json.

ARCHITECTURE

Le point d'entrée d'une application Vaadin est une classe héritant de "com.vaadin.Application". Elle se charge de créer les composants graphiques et de leur apporter les modifications nécessaires. Nous verrons celle-ci dans notre exemple de code. Une application Vaadin fonctionne dans une servlet au sein d'un serveur web java. Quand le serveur reçoit la 1re requête d'une application, il crée une instance de la classe ApplicationServlet qui hérite de HttpServlet. Elle gère les sessions par l'interface HttpSession et associe les instances de Application avec chaque session. Pendant la durée de la session, Vaadin transmet les actions de l'utilisateur à la bonne instance applicative [Fig.1]. Les principales parties de l'architecture de Vaadin sont les suivantes :

Interface graphique

Elle est à base de composants créés et gérés par l'application. Ces derniers possèdent à la fois une partie serveur et une partie cliente avec lequel l'utilisateur interagit. La partie serveur se sérialise au travers de la connexion en utilisant le Terminal Adapter. La partie client, quant à elle, sérialise les interactions de l'utilisateur qui seront reçues comme des évènements par la partie serveur et qui les transmettra à la logique applicative.

Client Side Engine

Un moteur de rendu client se charge de l'affichage au sein du navigateur en utilisant GWT. Il communique les interactions de l'utilisateur et reçoit les modifications graphiques au Terminal Adapter sous

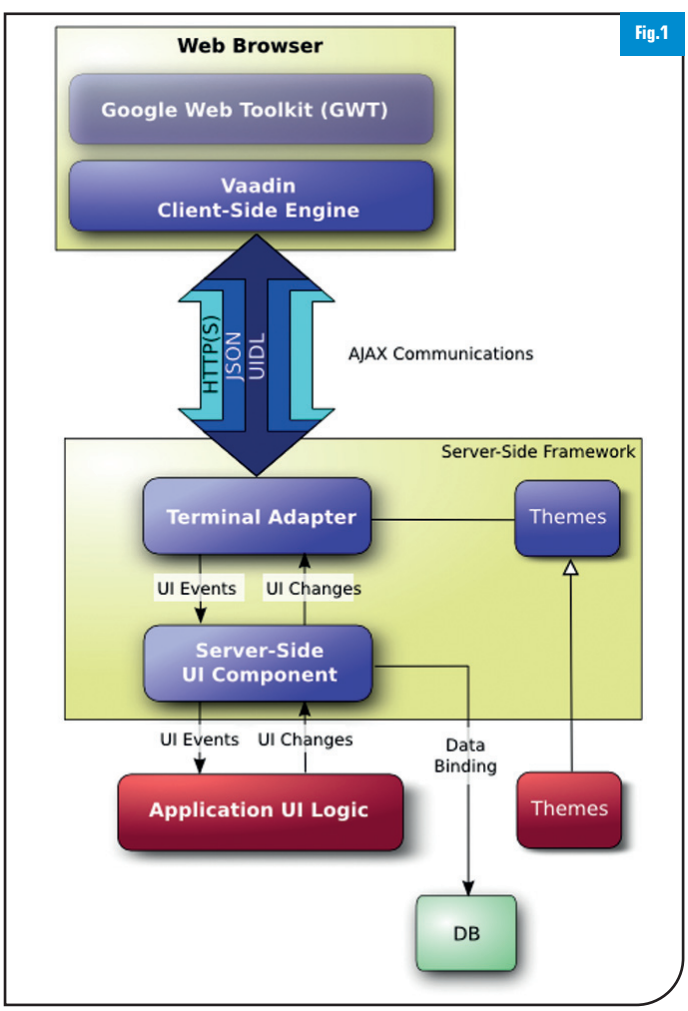


Fig.1

forme UIDL. La communication est faite par des requêtes asysnchrones http ou htpps.

Terminal Adapter

C'est cette couche qui permet l'utilisation d'une application Vaadin avec n'importe quel navigateur. Pour permettre ce genre d'abstraction, les composants graphiques lui communiquent leurs modifications et il gère le rendu pour le navigateur. C'est cette couche qui va utiliser un moteur de rendu GWT. Lorsque l'utilisateur interagit avec la page web, les événements lui sont communiqués par des requêtes ajax asynchrones.

UIDL :

Il s'agit du langage utilisé par le Terminal Adapter pour afficher l'interface graphique dans la page web ainsi que n'importe quelle modification de celle-ci. Ce langage basé sur Json est un format d'échange de données léger très efficace pour son intégration javascript côté client. Ce qui transite sur le réseau, ce ne sont pas les données correspondant au résultat d'une requête, mais les modifications d'interface graphique qu'engendre cette dernière.

Événement :

Les interactions de l'utilisateur sur les composants graphiques produisent des évènements qui sont d'abord traités en javascript dans la partie client, puis passés au Terminal Adapter par le serveur web, puis à la couche composants de l'application.

Modèle de données

En plus du modèle d'interface graphique, Vaadin fournit aussi un modèle pour interfacer les données qu'affichent les composants graphiques. Ce modèle permet à l'application la mise à jour automatique des données. Tous les composants graphiques utilisent ce modèle de façon interne, mais peuvent être liés à une source de données externe. Il est par exemple possible de lier un tableau au résultat d'une requête SQL.

INSTALLATION ET CRÉATION DU PROJET

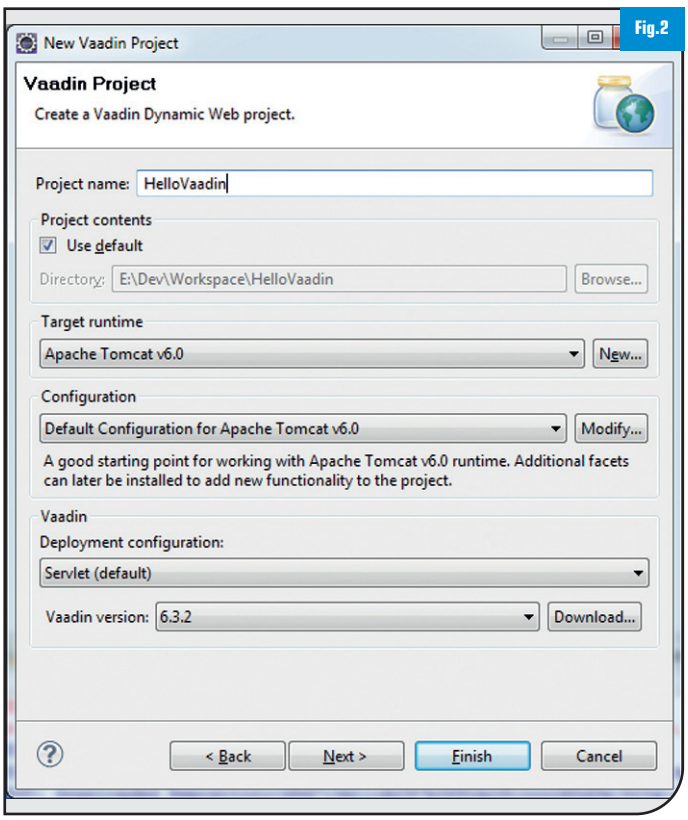
Pour développer avec Vaadin, nous allons installer le plugin Eclipse (aussi disponible pour NetBeans).

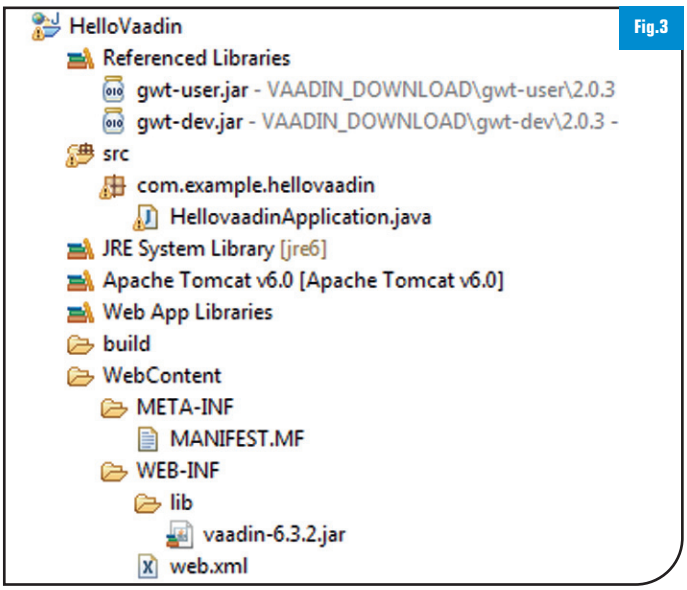
1. Assurez vous de disposer d'un serveur correctement configuré via WTP. Dans notre cas, nous utiliserons Apache Tomcat 6.0.
2. Installez le plugin Eclipse (Help/Software Updates/Add site) en pointant sur cette adresse : http://vaadin.com/eclipse
3. Une fois le plugin installé, nous avons la possibilité de créer un *"Vaadin Project"* [Fig.2]
4. Nous n'avons qu'à indiquer le nom de notre projet. D'autres réglages sont possibles :
   - *"Target Runtime"* nous permet de sélectionner le serveur local de développement.
   - *"Configuration"* nous permet de configurer les serveurs web.
   - *"Deployment configuration"* nous permet de sélectionner le type de déploiement. Les choix possibles sont Servlet, Google App Engine ou Portlet. Ce dernier est disponible à la fois en 1.0 (Old Portlet) ou en 2.0 (Generic Portlet).
   - *"Vaadin version"* tout simplement la version de Vaadin utilisée pour le projet. A noter le bouton "Download" qui permet de télécharger des versions plus récentes ou même plus anciennes.
5. Il ne reste plus qu'à cliquer sur *"Finish"*.

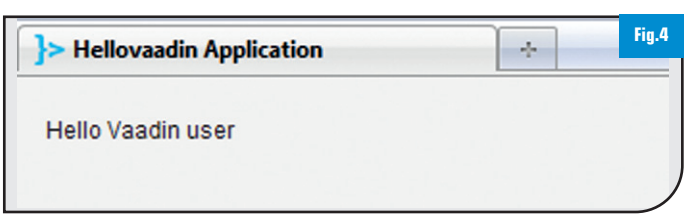
Et voilà, vous avez votre premier projet ! [Fig.3]

Nous remarquons tout de suite la structure de projet web avec : le code java, le MANIFEST.MF, le web.xml, un répertoire de bibliothèques (un seul jar en l'occurence), … mais sans fichier html ! Ce projet est pourtant parfaitement fonctionnel : [Fig.4]

Et oui, pas de polyglottisme, donc pas de html : c'est Vaadin qui se







chargera de tout cela. Commençons par regarder le code qui a produit cette page :

```
public class HellovaadinApplication extends Application {
 @Override
 public void init() {
   Window mainWindow = new Window("Hellovaadin Application");
   Label label = new Label("Hello Vaadin user");
   mainWindow.addComponent(label);
   setMainWindow(mainWindow);
 }
}
```

Il se comprend assez facilement pour les habitués de GWT ou de Swing. Cette classe est le point d'entrée de notre application Vaadin. Elle se doit donc d'étendre la classe Application et de surcharger en conséquence la méthode init(). C'est assez analogue à l'EntryPoint de GWT. Window est un conteneur d'interface graphique dans un navigateur. Etant aussi dans le paradigme "single page" typique des applications Ajax, une application Vaadin doit donc posséder une seule Window principale. Cela ne l'empêchera pas de posséder une Window fille. Celle qui est créée a pour titre "Hellovaadin Application" et contient un Label ayant pour texte "Hello Vaadin user".
Intéressons nous maintenant à html que le moteur de rendu de Vaadin à généré;. Il y a très peu de html, l'essentiel étant du javascript. Observons plus particulièrement les lignes suivantes :

```
document.write('<iframe tabIndex="-1" id="__gwt_historyFrame"
style="position:absolute;width:0;height:0;border:0;overflow:
hidden;" src="javascript:false"></iframe>');
document.write("<script language='javascript' src='/HelloVaadin/
VAADIN/widgetsets/com.vaadin.terminal.gwt.DefaultWidgetSet/com.
vaadin.terminal.gwt.DefaultWidgetSet.nocache.js?1275149013441
'><\/script>");
```

Cet exemple illustre bien l'utilisation de GWT faite par Vaadin. En effet, nous y retrouvons :
- L"history frame".
- Une déclaration de fichier javascript en *.nocache.js.

Côté hml, nous n'avons qu'une simple div :

```
<div id="HelloVaadin-692233477" class="v-app v-app-loading v-
theme-reindeer v-app-HellovaadinApplication" ></div>
```

C'est elle qui sera le conteneur de toute l'interface graphique qui sera construite.

HELLO ++

Faisons évoluer un peu notre code, pour lui donner un peu plus d'interactivité : un bouton simple sur lequel le clic permet d'afficher un message de notification.

```
Window mainWindow = new Window("Hellovaadin Application");
Button button = new Button("Click me");
button.addListener(new ClickListener() {
 @Override
 public void buttonClick(ClickEvent event) {
   getMainWindow().showNotification("Hello Vaadin user");
 }
 });
mainWindow.addComponent(button);
setMainWindow(mainWindow);
```

Le code a très peu évolué et reste simple. Nous avons juste remplacé le Label par un bouton qui possède un listener invoquant la méthode de notification qui affiche notre hello.
Comme nous l'avons déjà évoqué, la moindre interaction passe par un appel serveur. Observons donc ce qui a transité sur le réseau, la requête suivante a été faite au serveur :
http://localhost:8080/HelloVaadin/UIDL/?windowName=293611331
Et la réponse, sous forme UIDL :

```
for(;;);[{"changes":[["change",{"format": "uidl","pid": "PID0"},
["0",{"id": "PID0","caption": "Hellovaadin Application","name":
"293611331","theme": "","resizable":true,"main":true,"layoutRelative
Width":true,"v":{"scrollLeft":0,"scrollTop":0,"positionx":-1,
"positiony":-1,"close":false}},["1",{"id":  "PID3","cached":
true}],["notifications",{},["notification",{"caption": "Hello Vaadin
user","position":1,"delay":0}]]]], "meta" : {}, "resources" : {},
"locales":[]}]
```

Cette réponse sera interprétée par le Client Side Engine qui modifiera en conséquence la page. En GWT, un appel au serveur aurait été géré par un appel asynchrone, et nous aurions dû implémenter 2 callback : onSucces() et onFailure correspondant respectivement à la réussite/échec de l'appel. Comment alors gérer cette problématique ? Identifions différents types de problèmes pouvant survenir :
- Problème technique : Le serveur injoignable est un bon exemple. Il suffit de couper le serveur pour tester. Nous obtenons un message de notification d'erreur nous disant que le serveur n'a pu être joint.
- Exception non contrôlée : Lançons par exemple une NullPointerException dans le listener. Un petit logo d'erreur apparaît dans le bouton et au survol de la souris, on peut voir la trace de l'exception apparaître.
- Exception contrôlée : Elle est tout simplement gérée dans le code en java comme nous le ferions avec n'importe quel framework.

Vaadin permet de s'affranchir très facilement de la complexité de l'interaction avec le serveur. Ainsi, nous sommes plus proches du développement tel qu'on le ferait pour un client lourd.

CONCLUSION

Si les effets graphiques présentés sur la page de démo de Vaadin nous ont séduit au premier abord, comprendre l'architecture interne de Vaadin nous a convaincu sur ses possibilités. Vaadin reprenant des concepts déjà utilisés dans d'autres frameworks (Swing, GWT), sa prise en main se fait tout naturellement et est facilitée par l'excellente documentation disponible (Livre, Demo, JavaDoc). Vaadin est assurément un sérieux choix à considérer pour toute application RIA.

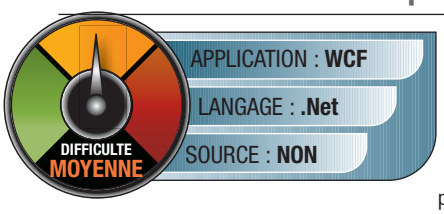
Ressources

Site : http://vaadin.com
Book of Vaadin : http://vaadin.com/book/
Show case : http://vaadin.com/demo

■ Nicolas **François**, *Ingénieur Etudes et Développement à Sfeir*
■ Ghislain **Kamgang**, *Ingénieur Etudes et Développement à Sfeir*

WCF RIA Services : Comment manipuler des données ?

Lors du développement d'applications de type RIA (Rich Internet Application) la question se pose de l'architecture à mettre en place. WCF RIA Services est proposé aux développeurs afin de leur simplifier leur développement pour des architectures de type n-tiers et fortement orientées données. Nous allons voir comment utiliser WCF RIA Services dans la manipulation de données.

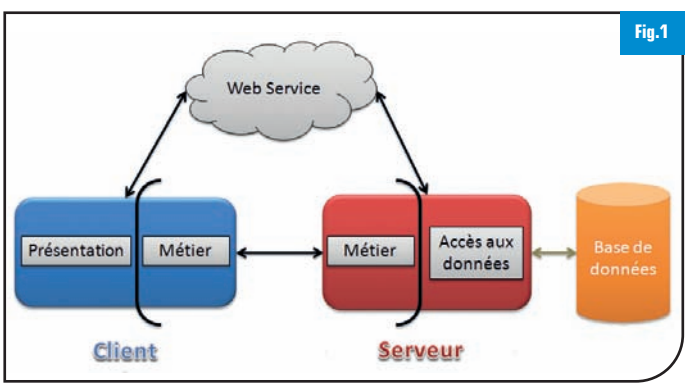


Le schéma [Fig.1] montre une architecture n-tiers simplifiée. WCF RIA Services est représenté par les accolades regroupant la partie métier du côté client et la partie métier du côté serveur, ce qui va permettre de faciliter le processus de développement d'une application RIA (Rich Internet Application). WCF RIA Services permet de dupliquer la couche métier du serveur vers le client sans aucune manipulation supplémentaire pour le développeur. Elle sera mise à jour à chaque recompilation de la solution.

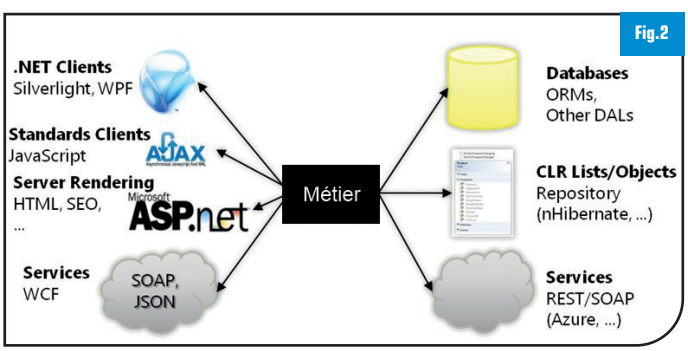
Notre exemple possède une base de données qui peut être hébergée ou non sur le même serveur que notre application. La partie serveur est composée d'une couche d'accès aux données (DAL) qui peut être générée grâce à Entity Framework, par exemple. Ensuite, il faut implémenter la couche métier, avec les règles métiers et y ajouter toute la logique applicative. Avec WCF RIA Services, cette couche métier est automatiquement répliquée du côté Client, de manière transparente pour le développeur, et elle permet à la couche présentation de pouvoir accéder à toute la logique métier de l'application comme si elle se trouvait elle-même sur la partie serveur. WCF RIA Services est une technologie qui peut être utilisée avec tout type de donnée et tout type de technologie RIA (Rich Internet Application). La plupart des exemples que l'on peut trouver sur Internet l'utilisent avec Silverlight et Entity Framework, mais WCF RIA Services peut totalement être utilisé avec les technologies réunies en exemple dans ce schéma : [Fig.2].

Nous allons maintenant voir WCF RIA Services par des exemples d'utilisation grâce à la base de données Northwind et ses tables Customers et Orders. Tout d'abord à la création de l'application Silverlight 4, la boîte de dialogue (cf. [Fig.3]) apparaît. Afin de pouvoir utiliser WCF RIA Services, il suffit de cocher la case correspondante.

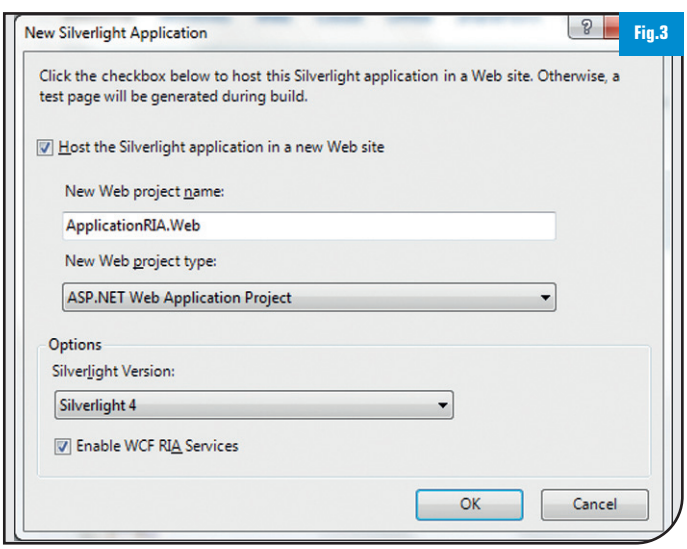
Puis nous allons générer un ADO.NET Entity Data Model via Entity Framework contenant nos deux tables dans le projet Web de l'application [Fig.4]. Ensuite, pour utiliser WCF RIA Services, il faut créer un **Domain Service.** Pour cela ce nouvel élément doit être ajouté dans le projet Web de la solution [Fig.5]. Suite à cela, un ou deux fichiers (selon que l'on a coché la case permettant de générer les classes associées concernant les metadata) est (sont) créé(s) dans le projet Web. Ce sont ces fichiers qu'il faudra modifier et/ou compléter pour implémenter la couche métier de notre application. Après l'ajout de notre **Domain Service** et la compilation de notre solution, on peut remarquer dans l'arborescence de cette dernière (cf. [Fig.6]), du côté client (Silverlight dans notre cas) qu'un dossier caché a été ajouté se nommant « Generated_Code ». Il possède un fichier qui a été généré contenant la réplication de la couche métier présente du



Architecture n-tiers d'une application RIA (Rich Internet Application)



Différentes technologies utilisables avec WCF RIA Services



Création d'une solution Silverlight 4 et WCF RIA Services avec Visual Studio 2010

côté serveur. Ce fichier va permettre d'accéder du côté client à toute cette logique métier qui a été mise en place avec notre Domain Service. Voici l'interface que l'on souhaite réaliser, une ComboBox et une DataGrid. Suivant le client (Customers) sélectionné dans la ComboBox, les commandes correspondantes (Orders) s'afficheront dans la DataGrid : [Fig.7].

Pour réaliser cela, il faut se servir des **DomainDataSource** fournis par WCF RIA Services. Ils permettent de relier facilement le code XAML, les interfaces utilisateurs et les données exposées via un **Domain Service.** Dans notre projet Silverlight, il ne faut pas oublier d'ajouter les espaces de noms dont on a besoin (attention les appellations des espaces de nom ont évolué tout au long des différentes versions intermédiaires) :

```
xmlns:riaControls=»clr-namespace:System.Windows.Controls;assembly
=System.Windows.Controls.DomainServices»
xmlns:data=»http://schemas.microsoft.com/winfx/2006/xaml/
presentation/sdk»
xmlns:domain=»clr-namespace:RIADomainSourceParam.Web»
```

System.Windows.Controls.DomainServices : pour utiliser les DomainDataSource.

RIADomainSourceParam.Web : le projet Web qui contient le Domain Service déclaré auparavant qui va nous permettre d'interroger notre Entity Data Model.

Ensuite il faut déclarer un premier **DomainDataSource** pour le contrôle ComboBox :

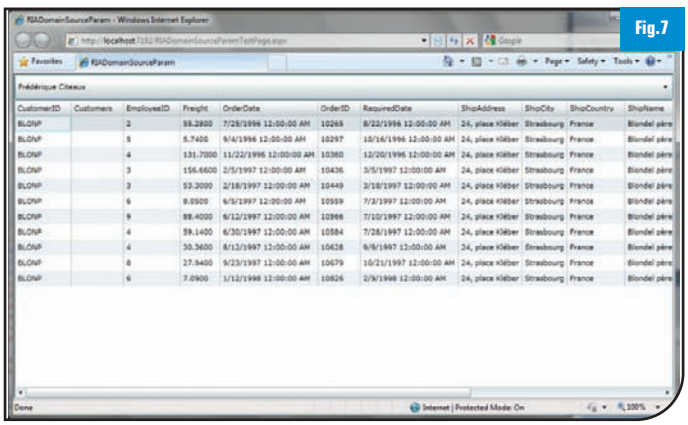
```
< riaControls:DomainDataSource x:Name=»sourceCustomers» Query
Name=»GetCustomers» AutoLoad=»True»>
  <riaControls:DomainDataSource.DomainContext>
    <domain:NorthwindContext />
  </riaControls:DomainDataSource.DomainContext>
</riaControls:DomainDataSource>

<ComboBox x:Name=»CbCustomers» ItemsSource=»{Binding Data,Element
Name=sourceCustomers}»
      ItemTemplate=»{StaticResource   DataTemplateComboBoxItel
Customers}»
      HorizontalAlignment=»Stretch» VerticalAlignment=»Stretch»
 Grid.Row=»0»
     SelectionChanged =»CbCustomers_SelectionChanged» />
```

Dans les propriétés du **DomainDataSource,** la propriété **QueryName** doit être renseignée avec le nom de la requête générée et exposée dans notre **Domain Service,** ici «GetCustomers», afin de lister tous les clients présents dans la table Customers.

```
public IQueryable<Customers> GetCustomers()
{
return this.ObjectContext.Customers;
}
```

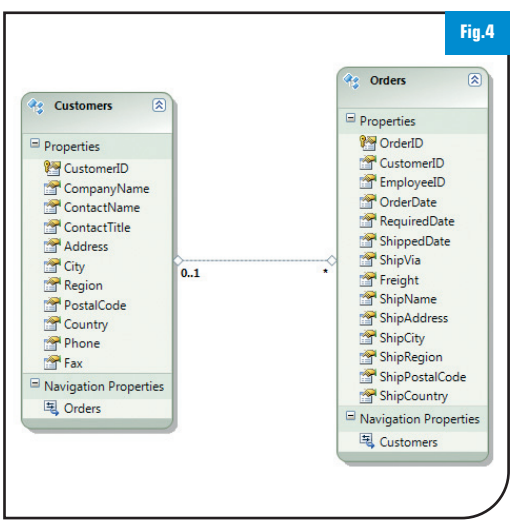
La propriété **AutoLoad,** par défaut à «true», permet de charger les données automatiquement au chargement de l'interface. Ensuite, on renseigne le **DomainContext,** ici à la valeur «NorthwindContext». Pour la ComboBox, on effectue le binding suivant pour la propriété **ItemsSource,** en faisant un lien vers le nom de notre **DomainDataSource** «sourceCustomers». Puis on abonne notre contrôle à l'événement SelectionChanged afin de pouvoir savoir lorsque l'utilisateur sélectionne un client. Et voilà, notre ComboBox est maintenant reliée à nos données et elle peut afficher tous les noms des clients de la base de données ! Maintenant nous souhaitons mettre à jour notre DataGrid suivant le client sélectionné. Pour cela, une requête Linq doit être créée, permettant de sélectionner toutes les com-

Fig.7



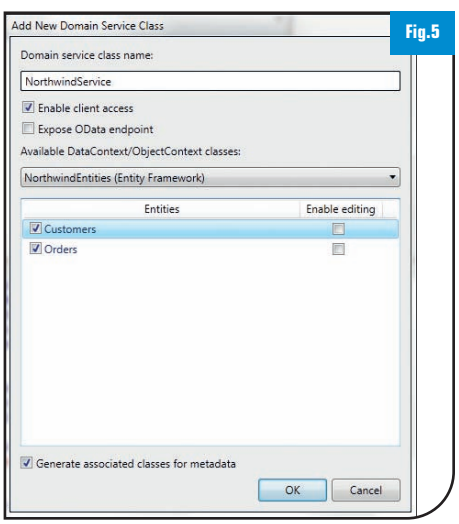
Interface de l'application exemple

Fig.4



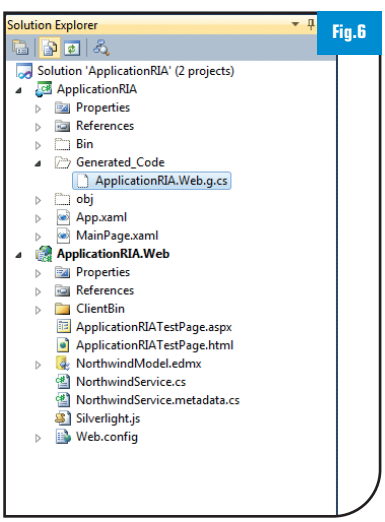
ADO.NET Entity Data Model via Entity Framework des tables Customers et Orders

Fig.5



Ajout d'un Domain Service

Fig.6



Arborescence d'une solution Silverlight 4 et WCF RIA Services dans Visual Studio 2010

mandes de ce client. Pour cela, nous allons ajouter une méthode dans notre **Domain Service** :

```
public IQueryable<Orders> GetOrdersWithParams(string myCustomerID)
{
 var query = from cust in this.ObjectContext.Orders
       where cust.CustomerID == myCustomerID
       select cust;
 return query;
}
```

Notre méthode prend en paramètre une chaîne de caractères correspondant à l'identifiant du client sélectionné. Nous allons pouvoir déclarer un nouveau **DomainDataSource** pour notre DataGrid.

```
<riaControls:DomainDataSource x:Name=»sourceOrders» QueryName=
»GetOrdersWithParams» AutoLoad=»False»>v
 <riaControls:DomainDataSource.QueryParameters>
   <riaControls:Parameter ParameterName=»myCustomerID»
                  Value=»{Binding ElementName=CbCustomers ,
Path=SelectedItem.CustomerID}» />
 </riaControls:DomainDataSource.QueryParameters>
 <riaControls:DomainDataSource.DomainContext>
   <domain:NorthwindContext />
 </riaControls:DomainDataSource.DomainContext>
</riaControls:DomainDataSource>

<data:DataGrid x:Name=»DgOrders» ItemsSource=»{Binding Data,
ElementName=sourceOrders}»
               AutoGenerateColumns=»true» Grid.Row=»1» Horizontal
Alignment=»Stretch»
               VerticalAlignment=»Stretch» />
```

On renseigne la propriété **QueryName** avec le nom de la méthode que l'on a déclaré auparavant «GetOrdersWithParams». La propriété **AutoLoad** est mise à «false», afin de ne charger les informations des commandes seulement lorsque l'utilisateur aura sélectionné un client. Au déclenchement de l'événement **SelectionChanged** de notre ComboBox, nous pourrons alors effectuer le Load du **DomainDataSource** :

```
private void CbCustomers_SelectionChanged(object sender, Selection
ChangedEventArgs e)
{
 sourceOrders.Load();
}
```

On ajoute ensuite le bloc **DomainDataSource.QueryParameters** dans notre DomainDataSource, afin de pouvoir renseigner le paramètre qui va être utilisé dans notre requête Linq. Pour cela les propriétés suivantes doivent être renseignées :

ParameterName : le nom du paramètre dans notre méthode **myCustomerID**.

Value : permet de donner la valeur du paramètre soit en passant directement la valeur, soit en la « bindant » avec la propriété d'un contrôle, ici notre ComboBox. Puis nous pouvons effectuer le binding de notre DataGrid, avec sa propriété **ItemsSource** en renseignant l'ElementName avec le nom de notre **DomainDataSource** «sourceOrders». Du XAML et une seule ligne de code-behind pour une interface Maître-Détail ! On peut aussi accéder aux données disponibles grâce aux liaisons entre les tables (les Orders d'un Customer par exemple), la méthode Include permet de faire cette liaison du côté du code-behind. Pour cela, il faut créer un **DomainService,** où il ne faut pas oublier de cocher la case suivante concernant la génération des metadata : [Fig.8]. Cela permet de générer la création d'un fichier **metadata** contenant les classes et les propriétés associées aux membres de nos entités. Dans notre **DomainService**, la méthode **GetCustomers** générée doit être modifiée en lui ajoutant la méthode **Include** pour qu'elle puisse récupérer les informations concernant les commandes effectuées par le client :

```
public IQueryable<Customers> GetCustomers()
{
 return this.ObjectContext.Customers.Include(«Orders»);
}
```

Ensuite dans le fichier metadata généré, le tag **[Include]** est ajouté au-dessus de la propriété Orders de la class CustomersMetadata sur laquelle on souhaite effectuer **l'include** :

```
[Include]
public EntityCollection<Orders> Orders { get; set; }
```

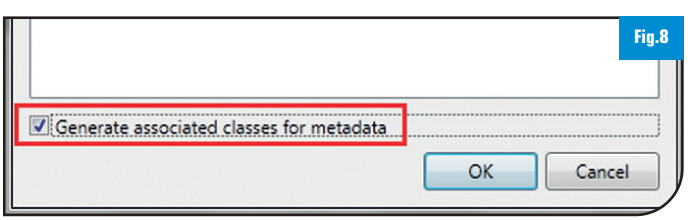
Et enfin, il ne reste plus qu'à ajouter le **DomainDataSource** correspondant et le **Binding** côté XAML :

```
<RIAControl:DomainDataSource x:Name=»source» QueryName=»Get
Customers» AutoLoad=»True»>
 <RIAControl:DomainDataSource.DomainContext>
   <domain:NorthwindContext />
 </RIAControl:DomainDataSource.DomainContext>
</RIAControl:DomainDataSource>
<ComboBox ItemsSource=»{Binding Data,ElementName=source}» Name=
»cbCustomers» />
<StackPanel DataContext=»{Binding ElementName=cbCustomers, Path
=SelectedItem}» >
   <RIAData:DataGrid ItemsSource=»{Binding Orders}» AutoGenerate
Columns=»True» Name=»dgOrders» /></StackPanel>
```

Maintenant on souhaite pouvoir utiliser des paramètres afin d'affiner les résultats à afficher dans la DataGrid. Tout d'abord en utilisant le tag Include et la méthode Where. Du côté XAML, l'événement **SelectionChanged** est ajouté à notre ComboBox afin de pouvoir savoir lorsqu'un **Customer** est sélectionné :

```
<ComboBox x:Name=»CbCustomers» SelectionChanged=»CbCustomers_
SelectionChanged» />
<my:DataGrid x:Name=»dgOrders» />
```

Du côté Code-Behind, le tag **Include** doit être ajouté dans le fichier Metadata comme indiqué précédemment, et la requête de la méthode **GetCustomers** est modifiée pour y inclure la liaison avec les **Orders**. Puis le code suivant est ajouté :



Générer les classes associées pour les metadata

```
public MainPage()
{
 InitializeComponent();
 contextCustomers = new NorthwindContext();
}
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
 CbCustomers.ItemsSource = contextCustomers.Customers;
 contextCustomers.Load(contextCustomers.GetCustomersQuery());
}
private void CbCustomers_SelectionChanged(object sender, Selection
ChangedEventArgs e)
{
dgOrders.ItemsSource = contextCustomers.Orders.Where(c => c.
CustomerID ==
((Customers)CbCustomers.SelectedItem).CustomerID);
}
```

Et les Orders correspondant au Customer sélectionné pourront s'afficher dans la DataGrid. Une autre méthode est possible, on peut aussi utiliser des paramètres avec le **DomainDataSource**. Du côté Code-Behind, il faut tout d'abord créer une nouvelle méthode qui prend en paramètre le **CustomerId** afin de pouvoir sélectionner les **Orders** que l'on souhaite afficher :

```
public IQueryable<Orders> GetOrdersWithParam(string customerId)
{
 return this.ObjectContext.Orders.Where(c => c.CustomerID ==
customerId);
}
```

Puis le code suivant est ajouté :

```
NorthwindContext contextCustomers;
NorthwindContext contextOrders;

public MainPage()
{
 InitializeComponent();
 contextCustomers = new NorthwindContext();
 contextOrders = new NorthwindContext();
}

private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
 CbCustomers.ItemsSource = contextCustomers.Customers;
 contextCustomers.Load(contextCustomers.GetCustomersQuery());
}

private void CbCustomers_SelectionChanged(object sender, Selection
ChangedEventArgs e)
{
 dgOrders.ItemsSource = contextOrders.Orders;

 contextOrders.Load(contextOrders.GetOrdersWithParamQuery(((
Customers)CbCustomers.SelectedItem).CustomerID));
}
```

Un ou plusieurs paramètres peuvent être utilisé(s) pour la requête associée au **DomainDataSource**. A noter qu'il est aussi possible de les utiliser via le code-behind de l'application.

Côté Code-Behind :

```
NorthwindContext contextCustomers;

public MainPage()
{
 InitializeComponent();
 contextCustomers = new NorthwindContext();
}

private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
 CbCustomers.ItemsSource = contextCustomers.Customers;
 contextCustomers.Load(contextCustomers.GetCustomersQuery());
}

private void CbCustomers_SelectionChanged(object sender, Selection
ChangedEventArgs e)
{
 // Remise à zéro des paramètres du DomainDataSource
 source.QueryParameters.Clear();

 // Déclaration et instanciation d'un nouveau paramètre
 Parameter p = new Parameter();

 // Nom du paramètre (mettre le même que celui déclaré dans
 le prototype de la méthode contenant l'appel aux données)
 p.ParameterName = «customerId»;

 // Valeur du paramètre
 p.Value = ((Customers)CbCustomers.SelectedItem).CustomerID;

// Association de la méthode du DomainContext au DomainData
Source
 source.QueryName = «GetOrdersWithParamQuery»;

// Ajout des paramètres
 source.QueryParameters.Add(p);

// Chargement du DomainDataSource
 source.Load();
}
```

Et les **Orders** correspondant au **Customer** sélectionné dans la ComboBox pourront alors s'afficher dans la DataGrid.

Maintenant nous allons voir comment insérer, mettre à jour et supprimer des données. Pour l'insertion de données : Voici l'interface utilisée pour ajouter un nouveau Customer : [Fig.9]. Lorsque l'utilisateur clique sur le bouton **OK** après avoir correctement rempli le formulaire, un nouveau Customer est créé grâce à la méthode **Add**, puis est validé avec la méthode **SubmitChanges** qui permet de répercuter l'ajout jusque dans la base de données :

```
private void OKButton_Click(object sender, RoutedEventArgs e)
{
 (customersDomainDataSource.DomainContext as NorthwindContext).
```

```
Customers.Add(new Customers()
 {
   Address = addressTextBox.Text,
   City = cityTextBox.Text,
   CompanyName = companyNameTextBox.Text,
   ContactName = contactNameTextBox.Text,
   ContactTitle = contactTitleTextBox.Text,
   Country = countryTextBox.Text,
   CustomerID = customerIDTextBox.Text,
   Fax = faxTextBox.Text,
   Phone = phoneTextBox.Text,
   PostalCode = postalCodeTextBox.Text,
   Region = regionTextBox.Text
 });

 customersDomainDataSource.DomainContext.SubmitChanges();
}
```

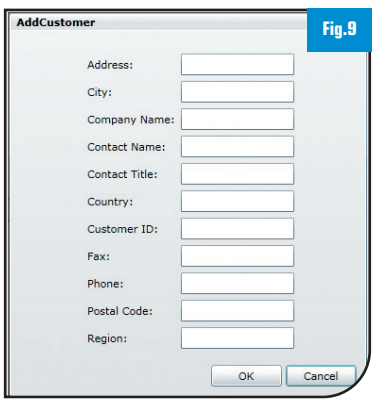


Fig.9

Interface d'ajout des données

Pour la mise à jour de données :
L'interface permettant de visualiser les données de notre table Customers est composée d'une DataGrid pour afficher les données, et d'un formulaire permettant de modifier les données sélectionnées de cette dernière avec un bouton « OK » pour valider les modifications, un bouton « Ajouter » pour afficher l'interface, et un bouton « Supprimer » pour supprimer le Customer sélectionné. Voici le code-behind pour le bouton « OK » permettant de mettre à jour les données du Customer sélectionné dans la DataGrid, après qu'elles aient été modifiées dans le formulaire. La méthode **SubmitChanges** valide ces modifications et les applique dans la base de données :

```
private void btn_Ok_Click(object sender, RoutedEventArgs e)
{
 customersDomainDataSource.DomainContext.SubmitChanges();
}
```

Suppression de données :
En appuyant sur le bouton « Supprimer », on souhaite supprimer le Customer sélectionné. La méthode Remove permet de le supprimer et la méthode **SubmitChanges** permet de valider cette modification et de propager cette modification dans la base de données.

```
private void btn_Supp_Click(object sender, RoutedEventArgs e)
{
 Customers myCustomer = dgCustomers.SelectedItem as Customers;

 (customersDomainDataSource.DomainContext as
NorthwindContext).Customers.Remove(myCustomer);
 customersDomainDataSource.DomainContext.SubmitChanges();
}
```

AUTHENTICATION DOMAIN SERVICE

WCF RIA Services propose les **Authentication Domain Service** qui permettent de simplifier l'accès aux méthodes afin de gérer l'authentification et l'identification d'utilisateurs à une application. Deux types d'authentifications sont proposés, l'authentification par formulaire et l'authentification Windows.
Tout comme le Domain Service, un fichier est ajouté du côté serveur et est à compléter suivant les besoins de l'application. Lors de la compilation, le code client correspondant est généré et est ajouté dans le même fichier contenant le code généré de notre Domain Service, se trouvant dans le répertoire caché « Generated_Code » comme vu auparavant.

ODATA

OData (Open Data Protocol) est un protocole qui permet d'exposer des données via HTTP pour différents types de clients (RIA, Web, Mobile, etc.) afin de pouvoir utiliser et modifier ces données à distance. Ces données sont alors requêtables avec Linq depuis le client. Grâce à WCF RIA Services, il est possible d'exposer ses propres données facilement. Lors de l'ajout d'un DomainService, il suffit de cocher la case «Expose OData endpoint», et les données des tables sélectionnées pourront être exposées.
Dans notre exemple, on utilise les tables Customers et Orders de la base de données Northwind. Le fait de cocher cette case apporte les modifications suivantes à notre code :

- Dans le **DomainService**, chaque méthode sans paramètre possède désormais un tag :

```
[Query(IsDefault = true)]
public IQueryable<Customers> GetCustomers()
{
 return this.ObjectContext.Customers;
}
```

- Dans le Web.Config, un **endpoints** est ajouté dans les **domainServices** :

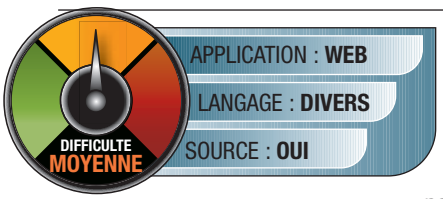
```
<system.serviceModel>
 <domainServices>
   <endpoints>
     <add name=»OData»
       type=»System.ServiceModel.DomainServices.Hosting.ODataEndpointFactory,
         System.ServiceModel.DomainServices.Hosting.OData,
         Version=4.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35» />
   </endpoints>
 </domainServices>
</system.serviceModel>
```

Pour voir plusieurs types de données exposés grâce à OData, une liste de quelques flux est disponible sur le site OData (http://www.odata.org)
Pour conclure, WCF RIA Services est un framework d'aide à la conception d'applications fortement orientées données, qui permet un gain de temps dans le développement et une simplification de la mise en place des mécanismes de communication entre les différentes couches de l'architecture n-tiers d'une application Web.

■ Audrey **Petit**
MVP Client Application Development
Ingénieur consultante .NET chez MCNEXT

Faites communiquer Flex 4 et Silverlight 4

Actuellement les deux principaux acteurs en termes de RIA sont Flex et Silverlight. Dans cet article, certes un peu provocateur je vous invite à découvrir comment il nous est possible de communiquer entre une application réalisée en Flex et l'autre en Silverlight.



Comme toute communication (humaine, machine…), pour que celle-ci ait lieu, il faut avant tout l'existence d'un canal commun. Dans notre cas, le premier canal qui nous vient à l'esprit est le navigateur web.

En effet, une application Flex ou Silverlight est contenue dans le navigateur Web. Depuis une application Flex il nous est possible d'interagir avec le navigateur via le composant Ajax Bridge. Ce composant, étant en mesure de mettre en relation l'applicatif Flex et les méthodes Javascript présentes dans la page web l'hébergeant, ce qui permet ainsi le déclenchement de méthodes ActionScript. Pour Silverlight, ce mode de fonctionnement est également possible grâce à l'utilisation de classes et méthodes situées dans le package System.Windows.Browser

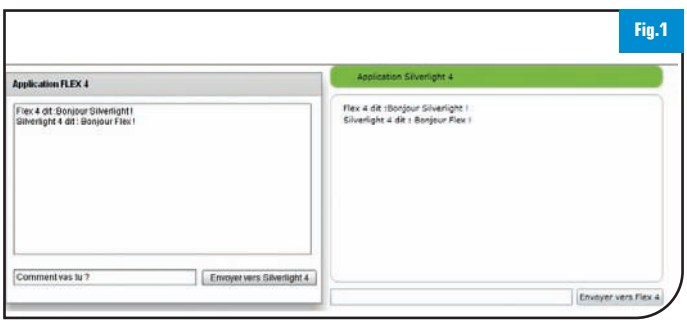
UN EXEMPLE ?

Pour illustrer ceci, nous vous proposons de créer une application de Chat, permettant la communication entre une application Flex et Silverlight [Fig.1].

Avant d'aller plus loin, nous allons décrire le processus d'exécution. Dans le cas où nous souhaitons envoyer un message depuis l'application Flex vers l'application Silverlight :

1 - L'utilisateur saisit son texte et clique sur le bouton envoyer.
2 - Une action ActionScript est alors déclenchée qui, par le biais de la classe ExternalInterface, fait appel à une méthode Javascript présente sur la page web.
3 - La méthode Javascript appelée, fait à son tour un appel à une méthode écrite en C# sans omettre de lui envoyer par la même occasion le texte saisi par l'utilisateur
4 - La méthode C# réceptionne le message et l'affiche dans l'application Silverlight.

Maintenant, plaçons-nous dans le cas où nous souhaitons envoyer un message à l'application Flex depuis l'application Silverlight :



Application de Chat

1 - L'utilisateur saisit son texte et clique sur le bouton envoyer.
2 - Une action C# est alors déclenchée faisant appel à une méthode Javascript présente sur la page web.
3 - La méthode Javascript fait à son tour un appel à une méthode écrite en ActionScript située dans l'application Flex sans omettre de lui envoyer le texte saisi par l'utilisateur. Cet appel et réalisé par le biais d'AjaxBridge qui permet de mettre en relation une fonction Javascript et une procédure ActionScript
4- La méthode ActionScript réceptionne alors le message et l'affiche dans l'application Flex.

Environnement
Pour que le projet fonctionne, il faut bien évidemment que celui-ci soit hébergé sur un serveur IIS de Microsoft et non un serveur Apache.

ET LE CODE DANS TOUT ÇA ?

Les lignes de code suivantes permettent l'appel de la fonction javascript nommée silverlight4 pour l'envoi de messages. Elles permettent également la notion d'implémentation de la librairie AjaxBridge pour réaliser l'exécution de la procédure AfficherMessageDeSilverlight depuis une fonction javascript contenue dans la page web :

A noter que l'implémentation d' Ajaxbridge se fait de façon automatique dans FlashBuilder 4 en réalisant un clic droit sur le nom du projet et en choisissant l'option « Créer un composant Ajax Bridge »

```
<?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/mx» minWidth=»955»
 minHeight=»600» width=»456» height=»335» xmlns:bridge=»bridge.*»
        >


 <fx:Declarations>

   <bridge:FABridge/>
 </fx:Declarations>



 <fx:Script>
   <![CDATA[

     // Importation de la classe ExternalInterface
     import flash.external.ExternalInterface;
```

```
      // Procédure d'envoi de message vers SilverLight
      // Par appel de la fonction Javascript nommée silverlight4
      public function chatVersSilverLight4():void{
        if (ExternalInterface.available) {
          var procedureJavascript:String = «silverlight4»
          var s:String = ExternalInterface.call(procedureJavas
cript,»Flex 4 dit :»+txt_message.text);
          lst_messagesenvoyes.text += «Flex 4 dit :»+txt_message
.text+»\n»;
          txt_message.text = «»;
        }
      }



      // Procédure de reception de message depuis Silverlight
      // Appelée par Ajax Bridge depuis Silverlight
      public function afficherMessageDeSilverlight(s:String):void{
        lst_messagesenvoyes.text += s+»\n»;
      }



    ]]>
  </fx:Script>


  <s:Panel x=»12» y=»10» width=»433» height=»315» title=»Application
  FLEX 4»>
    <s:TextArea x=»10» y=»10» width=»407» height=»206» id=»lst_messages
envoyes»/>
    <s:TextInput x=»10» y=»236» width=»250» id=»txt_message»/>
    <s:Button x=»268» y=»237» label=»Envoyer vers Silverlight 4»
  click=»chatVersSilverLight4()»/>
  </s:Panel>


</s:Application>
```

Une fois la saisie terminée, il convient de réaliser une version de déploiement validée du projet, en réalisant un clic droit sur le nom du projet puis en choisissant l'option exporter.
Passons à présent à l'écriture de l'application Silverlight.

CODE DE L'APPLICATION SILVERLIGHT

Pour créer l'application Silverlight, nous vous conseillons d'utiliser l'outil Visual Studio 2010 web developper dans sa version Express, qui vous fournira l'ensemble des bibliothèques nécessaires à la création de projet Silverlight.
Nous ne nous étendrons pas davantage sur la méthode de création de l'application, mais nous allons vous présenter le contenu des fichiers principaux de l'application :
Fichier MainPage.xaml, permettant le design de l'application :

```
<UserControl x:Class=»SilverlightApplication1.MainPage»
  xmlns=»http://schemas.microsoft.com/winfx/2006/xaml/presentation»
  xmlns:x=»http://schemas.microsoft.com/winfx/2006/xaml»
  xmlns:d=»http://schemas.microsoft.com/expression/blend/2008»
  xmlns:mc=»http://schemas.openxmlformats.org/markup-compat
ibility/2006»
  mc:Ignorable=»d»
  d:DesignHeight=»335» d:DesignWidth=»456» xmlns:dataInput=»
clr-namespace:System.Windows.Controls;assembly=System.Windows.
Controls.Data.Input»>

  <Grid x:Name=»LayoutRoot» Background=»White» Width=»456» Height
=»335»>
    <Border BorderBrush=»Silver» BorderThickness=»1» Height=
»31» HorizontalAlignment=»Left» Margin=»-1,0,0,0» Name=»border1»
VerticalAlignment=»Top» Width=»456» CornerRadius=»10» Background=
»#FF81B300»>
      <dataInput:Label Height=»30» Name=»label1» Width=»380»
Content=»Application Silverlight 4» />
    </Border>
      <TextBox Height=»23» HorizontalAlignment=»Left» Margin=»0,304,
0,0» Name=»txt_message» VerticalAlignment=»Top» Width=»331» />
      <Button Content=»Envoyer vers Flex 4» Height=»23» Horizontal
Alignment=»Right» Margin=»0,304,1,0» Name=»button1» Vertical
Alignment=»Top» Width=»118» Click=»button1_Click» />
    <Border BorderBrush=»Silver» BorderThickness=»1» Height=
»256» HorizontalAlignment=»Left» Margin=»0,40,0,0» Name=»border2»
VerticalAlignment=»Top» Width=»455» CornerRadius=»10»>
      <TextBlock Height=»234» Name=»txt_listemessages» Text=»»
Width=»420» />
    </Border>
  </Grid>
</UserControl>
```

Le fichier MainPage.xaml.cs, comportant les méthodes écrites en C# d'appel aux fonctions javascript :

```
using System.Windows.Media.Animation;
using System.Windows.Shapes;

//Utile pour la communication avec Javascript :
using System.Windows.Browser;

namespace SilverlightApplication1
{

  public partial class MainPage : UserControl
  {
    public MainPage()
    {
      InitializeComponent();
      //Utilisation de l'application comme Objet Scriptable
      HtmlPage.RegisterScriptableObject(«Page», this);
    }



    private void button1_Click(object sender, RoutedEventArgs e)
    {
      //Appel de la fonction Javascript liée à l'application Flex
      HtmlPage.Window.Invoke(«envoyerMessage», «Silverlight 4
dit : « + txt_message.Text);
```

```
      //Mise à jour de la liste des messages envoyés
      txt_listemessages.Text += «Silverlight 4 dit : « + txt_
message.Text+»\n»;

      //On efface le message envoyé
      txt_message.Text = «»;

    }


    //Methode appelée depuis Javascript
    [ScriptableMember()]
    public void AfficherMessage(string message)
    {
      txt_listemessages.Text +=  message + «\n»;
    }
  }
}
```

A noter que notre application doit être portée au titre de Scriptable lors de son initialisation. En faisant cela, nous précisons que l'application peut être appelée depuis la page conteneur via des fonctions javascript. S'ensuit après l'écriture de la méthode C# permettant l'appel (Invoke) de la méthode javascript *envoyerMessage*, chargée d'appeler la méthode ActionScript de l'application Flex. Puis par le biais de la décoration [ScriptableMember()], nous déclarons la méthode AfficherMessage, appelable depuis la page web permettant alors d'afficher le message reçu de l'application Flex.
Mettons tout ceci en musique en écrivant le fichier aspx, contenant les applications Flex, Silverlight et l'ensemble des méthodes Javascript :

```
<%@ Page Language=»C#» AutoEventWireup=»true» %>

<!DOCTYPE html PUBLIC «-//W3C//DTD XHTML 1.0 Transitional//EN»
«http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd»>
<html xmlns=»http://www.w3.org/1999/xhtml» >
<head runat=»server»>
 <title>SilverlightApplication1</title>
 <style type=»text/css»>
 html, body {
   height: 100%;
   overflow: auto;
 }
 body {
   padding: 0;
   margin: 0;
 }
 #silverlightControlHost {
   height: 100%;
   text-align:center;
 }
 </style>
 <script type=»text/javascript» src=»Silverlight.js»></script>
 <script type=»text/javascript»>
   function onSilverlightError(sender, args) {
     var appSource = «»;
     if (sender != null && sender != 0) {
     appSource = sender.getHost().Source;
     }

     var errorType = args.ErrorType;
     var iErrorCode = args.ErrorCode;

     if (errorType == «ImageError» || errorType == «MediaError») {
       return;
     }

     var errMsg = «Erreur non gérée de l'application Silver
light « +  appSource + «\n» ;

     errMsg += «Code : «+ iErrorCode + «     \n»;
     errMsg += «Catégorie : « + errorType + «        \n»;
     errMsg += «Message : « + args.ErrorMessage + «      \n»;

     if (errorType == «ParserError») {
       errMsg += «Fichier : « + args.xamlFile + «      \n»;
       errMsg += «Ligne : « + args.lineNumber + «      \n»;
       errMsg += «Position : « + args.charPosition + «     \n»;
     }
     else if (errorType == «RuntimeError») {
       if (args.lineNumber != 0) {
         errMsg += «Ligne : « + args.lineNumber + «      \n»;
         errMsg += «Position : « +  args.charPosition + «    \n»;
       }
       errMsg += «Nom de la méthode « + args.methodName + «    \n»;
     }

     throw new Error(errMsg);
   }
 </script>
```

Pour les méthodes, voir le code source complet sur le site.

Si vous exécutez à présent l'application, vous devriez constater qu'une application Flex est en mesure de communiquer avec une application Silverlight

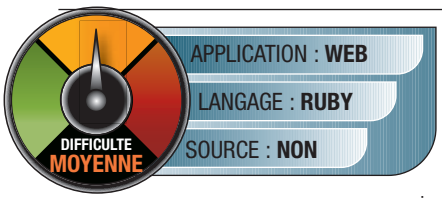
QUE DIRE DE TOUT CELA ?

Non partisan des traditionnels clivages informatiques : « Vive Linux, Non à Windows ! », « Vive J2EE, Non à .Net !, comme je l'ai évoqué en introduction de cet article, ce dernier peut être source de polémique. Mais n'oublions pas qu'il faut garder un esprit ouvert sur le monde qui nous entoure car Flex ou Silverlight ne sont pas les seules technologies RIA existantes et il se peut qu'un jour vous soyez amené à faire cohabiter les deux mondes par souci de migration ou de choix politique interne à votre entreprise.

■ Aurélien **Vannieuwenhuyze**

Les bases avec Rails 3 1re partie

Ruby On Rails est un framework open-source écrit en Ruby par 37signals et créé pour développer des applications web rapidement et facilement.



Ruby est un langage open-source créé par Yukihiro Matsumoto. Inspiré de plusieurs langages dont Smalltalk, Perl, et Python; il est interprété (comme PHP), objet (comme Java), et multi-paradigme (comme le C++). L'objectif principal de ce langage est de rendre la programmation lisible et surtout..."naturelle". Fort de tout ceci, Rails se base sur 3 principes fondamentaux :

DRY : Don't Repeat Yourself

Ré-écrire encore et encore les mêmes morceaux de code est une mauvaise chose !

Convention Over Configuration

Rails est basé sur une somme de choix architecturaux et impose une direction par des mécanismes automatisés. Pas de fichiers de configuration sans fin !

REST : Representational State Transfer

REST est l'architecture à la base du web. Il se base sur les verbes du protocole HTTP (GET, POST, PUT, DELETE) pour accéder à des ressources (URI spécifique) et les manipuler.

ex : http://monsite.com/users/4/avatar, permettra à la fois de créer l'avatar (POST) du user dont l'id en base est 4, de l'obtenir (GET), le modifier (PUT), ou le supprimer (DELETE)

RAILS+MERB=RAILS3 ET POURQUOI C'EST IMPORTANT

Rails a très récemment fusionné avec un autre framework web écrit en Ruby : Merb. C'est ce qui a donné naissance à Rails 3. Il s'agit d'un évènement clé techniquement mais aussi humainement :

- Merb apporte à Rails le découplage de ses composants: modularité et puissance sont désormais les mots d'ordre.
- La communauté Rails peut s'affranchir des désaccords (lot commun de toute communauté) et agir rapidement pour aller de l'avant.

Créer un premier projet

Rails englobe plusieurs outils utilisables via un terminal. Si vous êtes sous Windows, il est temps d'installer Cygwin ou PowerShell ! Evidemment, utiliser un IDE est possible avec Rails, mais un bon éditeur de texte (Textmate sous Mac ou Notepad++ sous Windows) et quelques lignes de commande feront des miracles !

Prérequis :

Assurez vous aussi d'avoir Ruby (1.8.7 minimum) et Rails 3.0.0 installés sur votre machine. Pour Mac OS X il suffi de taper : sudo gem install rails. Les sources du projet sont déjà disponibles sur : http://github.com/gdurelle/myblog

C'est parti !

La commande "rails" permet de créer des projets mais aussi de générer la plupart des briques dont vous aurez besoin.

Ouvrez votre terminal et tapez : rails new myblog. Un dossier "myblog" est créé, avec toute l'arborescence de votre projet rails.

Le dossier "app" est le coeur business de votre application.

Rails suit le pattern MVC, et vous retrouverez ici les contrôleurs, modèles, et vues. Les "helpers" servent à garder vos vues lisibles et DRY. Ils contiendront des méthodes (tris, calculs, mise en forme...) spécifiques à l'affichage. Le dossier "config" est le centre névralgique.Il contient les fichiers de configuration des différents environnements, les routes, les traductions, etc.

Le dossier "public" de votre application est généralement directement exposé au monde. Il contient les fichiers statiques servis directement par le serveur (images, css, javascript...).

Si vous utilisez ActiveRecord (ORM par défaut) la convention veut que vous utilisiez une base de données. Il est donc temps d'en créer une et d'en informer Rails via le fichier config/database.yml

SQLite est le SGBD par défaut, aussi "rails new" a dû tout faire pour vous ! Regardons de plus près ce fichier config/database.yml :

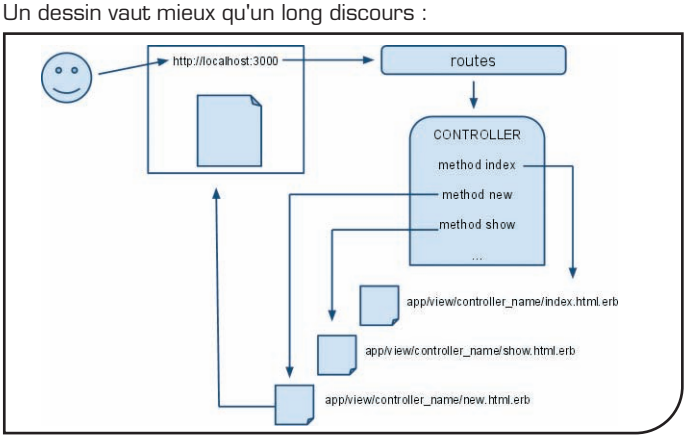
Il s'agit d'un fichier YAML qui définit le lien avec la base de données.

```
# SQLite version 3.x
#   gem install sqlite3-ruby (not necessary on OS X Leopard)
development:
  adapter: sqlite3
  database: db/development.sqlite3
  pool: 5
  timeout: 5000
```

Vous pouvez dès à présent lancer le serveur en tapant : rails server et accéder à celui-ci sur http://localhost:3000

Comment ça marche ?

Un dessin vaut mieux qu'un long discours :



Les routes sont la porte d'entrée pour accéder à un couple controller/action. Par défaut, le nom de la vue correspond à l'action appelée. Pour que le contenu d'une vue soit affichable, il faut définir une route d'accès à une méthode de controller. Ces routes sont soit statiques soit dynamiques. A chaque méthode d'un contrôleur peut correspondre une vue qui servira à afficher du contenu HTML. Une vue a accès aux variables définies dans la méthode qui lui correspond dans le controller. **Exemple :**

Dans routes.rb: map.uneroute "uncontrolleur#uneaction"

http://localhost:3000/uncontrolleur/uneaction devrait correspondre au controller Uncontroller, à l'action Uneaction et à la vue uneaction.html.erb.

Gemfile

Ouvrez le fichier : *Gemfile*. Ce fichier contient la liste des gems utilisés dans votre projet. Cela facilite les projets en équipe et assure une homogénéité des environnements. Vous pouvez déjà voir les lignes pour Rails et SQLite.

Le scaffold

Dans la console/terminal tapez la commande suivante :

```
rails g scaffold author name:string ("g" est un alias pour "generate")
```

Cette commande ajoute une ligne dans le fichier routes.rb permettant d'accéder aux fonctions REST et génère :

- le contrôleur nommé AuthorsController et ses méthodes liées à REST (Create Read Update Delete)
- les vues permettant de gérer les ressources du contrôleur
- le fichier helpers/authors_helper.rb, pour vos éventuelles méthodes spécifiques à l'affichage.
- le modèle Author
- la migration (fichier Ruby qui va intervenir sur la BDD à votre place) qui va créer la table "authors" avec la colonne "name"

Et maintenant les articles: rails g scaffold article title:string content:text author_id:integer. La commande *"rake db:migrate"* va lancer les migrations et ainsi créer les tables dans la BDD. La majorité du travail est terminé ! Vous pouvez vous concentrer sur l'essentiel : votre code métier. Lancez votre server ("rails server") et visitez http://localshot:3000/ authors pour jouer avec votre nouvelle application :). Attention : le champ correspondant à author_id vous demandera l'id de l'auteur et pas son nom. C'est cela qui permettra de faire la jointure entre l'article et son auteur, ce que nous allons voir tout de suite.

ORM et associations

Modifiez les modèles *article.rb* et *author.rb* du dossier *app/models/* comme ceci :

```
class Author < ActiveRecord::Base
  has_many :articles
end
```

```
class Article < ActiveRecord::Base
  belongs_to :author
end
```

Ceci va permettre à ActiveRecord de faire les liens relationnels entre vos objets et nous apporter toutes les fonctions dont nous avons besoin. Si vous êtes familiers d'UML ou E/R vous devriez trouver tout ça plutôt logique. Ici on indique que la table "article" correspondant au modèle aura une clef étrangère vers l'auteur. Rails va considérer par défaut que la clef sera nomDuModel_id, ici author_id
Ex : *@article.author* renverra l'auteur de l'article.
Allez sur http://localhost:3000/authors, et créez des auteurs.

ALLONS PLUS LOIN

Les routes

Le fichier *config/routes.rb* définit comment accéder aux ressources de l'application via des URL/URI. Nous allons créer la page d'accueil générale du blog, celle que vous verrez en tapant seulement : http://localhost:3000. Modifions le fichier config/*routes.rb* :

```
# You can have the root of your site routed with "root"
# just remember to delete public/index.html.
root :to => "welcome#index"
```

Dé-commentez la ligne ci-dessus ; elle indique que la racine du site ("/") sera gérée par le controller "welcome" et son action (et donc sa vue) index. La prochaine étape est donc de créer le controller et ses vues :

```
rails g controller welcome index about
```

Tous ce que vous écrirez à présent dans la vue *app/view/welcome/index.html.erb* sera affiché par défaut à la racine du site. N'oubliez pas de supprimer le fichier *index.html* dans le repertoire *public/* . Ce fichier est affiché par défaut puisque le repertoire public est la "racine" du site pour le navigateur. Allez sur http://localhost:3000, qui vous affichera la vue welcome/index.html.erb

Arel

Editons welcome_controller.rb :

```
def index
  @articles = Article.all
  @johnny = Article.select(:title, :content).joins(:author).where("authors.name = 'Johnny'")
end
```

Arel (ActiveRelation) permet de créer des requêtes puissantes en enchaînant des méthodes simples. Tout d'abord nous sélectionnons tous les articles de la base. La requête SQL générée est : "SELECT `articles`.* FROM `articles`"

- Puis nous sélectionnons les articles (titre et content) dont l'auteur (jointure) a pour "name" "Johnny".
  Arel attend le dernier moment pour nous générer une belle requête : "SELECT title, content FROM `articles` INNER JOIN `authors` ON `authors`.`id` = `articles`.`author_id` WHERE (authors.name = 'Johnny')"

Utilisons @articles dans la vue *app/views/welcome/index* :

```
<% @articles.each do |article| %>
  <h3><%= article.title %> <span class="date">(<%= article.created_at.strftime("%d %m %Y") %>)</span> </h3>
  <%= article.content.truncate(100) %>
<% end %>
```

Utilisons @johnny :

```
<p>
  <h2>Articles by Johnny :</h2>
  <% @johnny.each do |article| %>
    <h3><%= article.title %></h3>
    <%= article.content.truncate(100) %>
  <% end %>
</p>
```

Rendez vous sur http://localhost:3000 Pour voir les changements.

Scope (anciennement appelés named_scope)

Ajoutons au model Article la ligne :

```
scope :recent, where("created_at >= ?", Time.zone.now - 2.days )
```

Cette ligne va me permettre d'effectuer des requêtes très facilement, par exemple dans la méthode index du contrôleur :

```
@articles = Article.recent
```

De plus, les scopes peuvent s'enchaîner avec Rails3 :

```
scope :recent, where("created_at >= ?", Time.zone.now - 2.days )
scope :by_johnny, where("author_id = ?", Author.find_by_name("Johnny").id)
scope :rock, recent.by_johnny
```

Partials & Layouts

Un *partial* est une vue destinée à faire partie d'une ou plusieurs autres vues. C'est un autre exemple d'application du principe DRY. Par convention, un fichier partial est toujours nommé par le préfixe underscore('_'), afin de le distinguer d'une vue régulière. Cependant il est appelé sans le underscore ('_')
ex : <%= render "menu" %> rendra la vue du fichier *_menu.html*
Dans *app/views/authors* regardez le code des vues new et edit puis regardez le code du fichier *_form*.

```
<%= render 'form' %>
```

Ces vues insèrent le code "rendu" du partial _form dans leur code

propre. Un autre utilisation concrète des partials est la création de vos headers et footers de page. Par convention (on adore ça), on placera ce type de fichier dans un repertoire *'shared/'* : app/view/shared/. Créons un repertoire shared dans *app/view/*, puis ajoutons y un fichier *_headers.html.erb*
Placez-y le code suivant :

```
<div id="header">
  <%= link_to "Home", root_path %>
  <%= link_to "Articles", :controller => "articles", :action => "index" %>
  <%= link_to " + Add article", :controller => "articles", :action => "new" %>
  <%= link_to "Authors", :controller => "authors", :action => "index" %>
</div>
```

Ce sont simplement les liens vers les pages les plus importantes que nous venons de créer… Il ne vous reste plus qu'à faire en sorte que ce header s'affiche partout, et plutôt que de faire

```
<%= render "shared/header"%>
```

sur toutes vos vues (ce qui ne serait pas du tout DRY ;)), ouvrez plutôt le fichier *app/view/layouts/application.html.erb*. Cette vue est un "layout", elle est le conteneur par défaut de vos vues.

```
<body>

<%= render "shared/header"%>

<%= yield %>

</body>
</html>
```

La ligne <%= yield %> est ce qui permet d'afficher vos vues.
Il nous suffit donc d'ajouter notre <%= render "shared/header" %> au-dessus de cette ligne pour avoir le même header sur l'ensemble de notre application.

Helpers

Les helpers contiennent des petites portions de code réutilisables dans nos vues. Supposons, par exemple, que nous voulions la liste des auteurs mais avec le nombre d'articles écrits par chacun. Le contrôleur va générer la liste des auteurs :

```
class WelcomeController < ApplicationController
  def index
    @articles = Article.rock
    @johnny = Article.select(:title, :content, "articles.created_at").joins(:author).where("authors.name = 'Johnny'")
    @authors = Author.all
  end
```

mais l'affichage sera calculé dans le helper :

```
module WelcomeHelper
  def name_with_count(author)
    count = author.articles.count
    if count > 0
      author.name + " (#{count})"
    else
      author.name
    end
  end
end
```

Cette méthode compte simplement le nombre d'articles écrits par auteur. Si l'auteur a écrit des articles, ce nombre est affiché entre parenthèses à côté du nom de l'auteur. Finalement, dans app/view/welcome/index.html.erb, nous utilisons cette méthode :

```
<fieldset id="authors_list" class="right">
  <legend>Authors</legend>
  <ul>
    <% @authors.each do |author| %>
      <li><%= name_with_count(author) %></li>
    <% end %>
  </ul>
</fieldset>
```

La méthode de ce helper peut être réutilisée à plusieurs endroits, nous allons voir cela tout de suite. Notez que la technique utilisée ici peut être améliorée (au niveau performances) en effectuant le "count" dans la requête du contrôleur, le helper s'occupant alors uniquement de l'affichage.

JOUONS UN PEU

Allez sur http://localhost:3000/articles/new. Pour indiquer l'auteur de l'article, le champ text attend l'id en base de l'auteur, ce qui n'est pas très sympa ! Nous allons donc modifier la vue de ce formulaire, afin de sélectionner l'auteur par son nom dans une liste, qui s'occupera de retourner l'id correspondant. Dans le fichier _form.html.erb, nous allons changer :

```
  <div class="field">
    <%= f.label :author_id %><br />
    <%= f.text_field :author_id %>
  </div>
```

en ceci :

```
  <div class="field">
    <%= f.label :author_id %><br />
    <%= f.select(:author_id, @authors.map{|a| [a.name, a.id]}) %>
  </div>
```

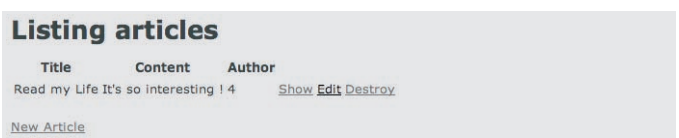
Editez également le fichier *articles_controller.rb* :

```
class ArticlesController < ApplicationController

  before_filter :find_authors, :only => [:new, :edit]
```

Le *before_filter* va appeler la méthode voulue avant tout autre méthode.

```
  protected

  def find_authors
    @authors = Author.all
  end
```

Créez un article et sélectionnez l'auteur dans la liste, puis retournez http://localhost:3000/articles . Vous remarquerez sûrement que l'auteur indiqué est l'id de l'auteur.



Encore une fois, il serait plus "user-friendly" d'afficher le nom de l'auteur. Editez simplement *views/articles/index.html.erb* :
Et changez

```
<td><%= article.author_id %></td>
```

par

```
<td><%= article.author.name %></td>
```

Et voilà ! Ceci est possible grâce aux relations que vous avez indiquées plus tôt. Arel sait automatiquement que *"author"* indique la clef étrangère *"author_id"* grâce aux conventions de nommage et retourne l'objet correspondant à la relation : Author.

Dry, dry, dry…

Nous avons créé un helper pour la "Home", réutilisons le ici ! Dans/views/articles/_form, remplacer le a.name par un appel au helper :

```
<%= f.select(:author_id, @authors.map{|a| [name_with_count(a), a.id]}) %>
```

Résultat :

On pourra également effectuer ce changement dans le listing (app/views/articles/index) :

```
<td><%= name_with_count(article.author) %></td>
```

CONCLUSION

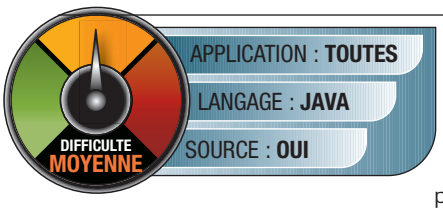
Nous avons ici survolé les principes de bases et les nouveautés apportées par Rails 3 mais nous espérons vous avoir donné l'envie de tester plus en profondeur ce super outil qu'est Rails :). Nous verrons par la suite les outils permettant de faciliter la productivité et la mise en production rapide.

■ Gregory **Durelle** - ■ Jean-Baptiste **Feldis**

Studio Melipone

Profilez vos applications JEE avec TPTP

Dans cet article, nous vous proposons un exemple de mise en œuvre de l'outil TPTP pour identifier les goulots d'étranglement d'une application JEE.



Soutenu par la fondation Eclipse qui en a fait un top projet, il se décompose en 4 sous-projets principaux :

- TPTP Platform qui regroupe le coeur du projet ainsi que les services communs aux autres sous-projets
- Testing Tools qui amène un ensemble d'outils permettant la mise en oeuvre de tests
- Monitoring Tools qui propose des services permettant de monitorer les ressources d'une application et du système sous-jacent
- Tracing and Profiling Tools qui propose des services facilitant la collecte de données permettant ainsi le profiling d'applications

Dans le cadre de cet article, nous nous intéresserons à ce dernier sous-projet puisque nous allons mettre en œuvre TPTP pour réaliser le profiling d'une petite application JEE. Son architecture [Fig.1] peut paraître complexe de prime abord mais elle est parfaitement intégrée à l'ensemble des frameworks de base de l'écosystème Eclipse, ce qui lui confère sa richesse et sa puissance.

Cette architecture se divise en 2 grandes parties :

- La partie présentation qui est située sur le système sur lequel on souhaite exploiter le client TPTP pour visualiser et exploiter les données collectées.
- La partie qui est déployée sur le système cible, c'est-à-dire sur lequel on souhaite profiler une application par exemple.

Installation

L'installation est assez simple et se passe en 2 temps. Sur le poste client, il faut disposer d'une installation d'Eclipse sur laquelle on installera le plugin TPTP disponible via l'update site correspondant de votre distribution. Sur la machine distante, qui accueille l'application à profiler, il faut installer l'Agent Controller téléchargeable à l'adresse suivante : http://www.eclipse.org/tptp/home/downloads/

Configuration

Comme pour l'installation, la configuration se passe en 2 temps. Dans un premier temps, on va décompresser l'Agent Controller sur la machine distante. Ensuite, on va l'installer en lançant le fichier SetConfig contenu dans le répertoire bin/ situé à la racine de l'arborescence de l'Agent Controller. Le lancement se fait via un terminal et il vous sera demandé de configurer votre Agent Controller.

Une fois ces quelques informations renseignées, l'Agent Controller est prêt à être lancé via le fichier ACServer du répertoire bin/ mentionné précédemment. Il est important de préciser que l'Agent Controller doit être lancé en premier avant que le serveur d'applications soit lancé. Avant de lancer le serveur d'applications il faut procéder au positionnement de quelques variables. Voici un exemple pour un lancement du serveur sous Windows :

- set TPTP_AC_HOME=<Path vers l'Agent Controller>
- set JAVA_PROFILER_HOME=%TPTP_AC_HOME%\plugins\org.eclipse.tptp.javaprofiler
- set PATH=%JAVA_PROFILER_HOME%;%PATH%;%TPTP_AC_HOME%\bin

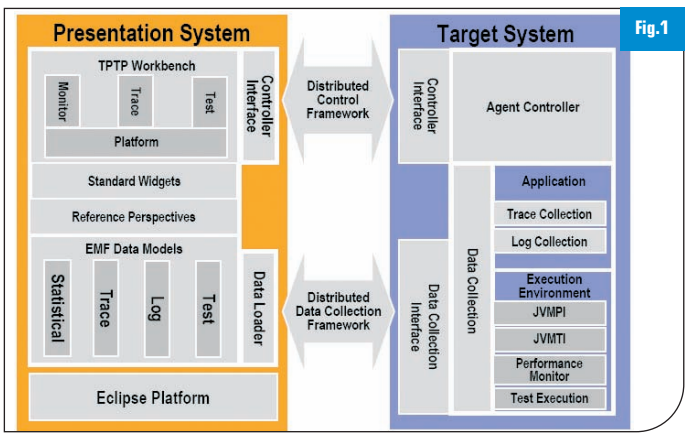


Fig.1

Architecture de TPTP

- set PATH=%PATH%;%JAVA_HOME%\bin

Le plus simple étant de positionner ces variables dans le fichier de lancement de votre serveur d'applications. Dans le cas du serveur Tomcat qui sera utilisé dans la suite de cet article on place ces variables dans le fichier startup. Pour terminer la configuration du serveur d'applications il faut placer la déclaration de l'agent dans ses options de lancement. Ceci se réalisant en plaçant la ligne suivante dans le fichier startup :

```
set JAVA_OPTS=-agentlib:JPIBootLoader=JPIAgent:server=enabled;
<Profile-Option>
```

Les options de profiling sont les suivantes :

- CGProf pour profiler le temps d'exécution de l'application
- HeapProf pour profiler la mémoire utilisée par l'application
- ThreadProf pour profiler l'utilisation des threads faits par l'application durant son exécution

Un seul type de profiling peut être réalisé à la fois. Une fois votre choix effectué, le serveur d'applications peut être lancé.

Application JEE

Pour mettre en évidence les possibilités de profiling de TPTP, nous avons développé une petite application JEE sans prétention qui se contente d'afficher une liste de produits chargée depuis des fichiers XML contenus sur le serveur d'applications. Bien que très simple cet exemple va permettre de mettre en évidence un goulot d'étranglement lors du chargement des données des fichiers XML. Le code susceptible de poser des problèmes de performance est contenu dans le DAO en charge de la récupération des produits contenus dans les fichiers XML. Voir le code de base sur www.programmez.com.

À première vue rien d'exceptionnel dans ce DAO. Cependant, en y regardant de plus près on s'aperçoit qu'une instance de parser SAX est créée à chaque lecture de fichier XML par la méthode parseContent. Sur un nombre de fichiers très importants à traiter cela se paiera forcément en temps d'exécution. Nous allons voir maintenant comment ce goulot d'étranglement va être détecté à l'aide du profiling de l'application.

Profiling de l'application

Maintenant que nous avons mis en place l'Agent Controller sur la machine distante sur laquelle vont être lancés le serveur d'applications et notre application, nous allons pouvoir utiliser la partie cliente de TPTP. Comme vu précédemment, la partie cliente se présente sous la forme d'un plugin Eclipse. Une fois Eclipse lancé, il faut sélectionner l'entrée Profile Configurations …. Une fois l'écran de configuration lancé, il faut alors créer une nouvelle entrée dans la partie Attach to Agent dans le menu de gauche. Une fois la nouvelle entrée créée, il faut ajouter le serveur sur lequel l'application distante tourne ainsi que le port par lequel l'Agent Controller va communiquer avec la partie cliente d'Eclipse (par défaut le port 10002). Une fois ces informations saisies, il peut s'avérer intéressant de tester la connexion entre ces 2 parties en cliquant sur le bouton Test Connection. Il faut ensuite passer à l'onglet Agents afin de voir les options proposées pour le profiling par l'agent distant. Dans notre cas, on sélectionne Execution Time Analysis puisque l'on souhaite profiler l'application JEE au niveau de son temps d'exécution. Un click sur le bouton Edit Options permet de définir des filtres sur les packages que l'on souhaite exclure du profiling. Dans notre cas, la configuration par défaut est suffisante. Elle exclut les packages Java de base. Ces filtres mis en place, le profiling est lancé via le bouton Profile. Il suffit ensuite d'utiliser notre application web pour que l'Agent Controller collecte les données d'exécution qui sont ensuite envoyées à la partie cliente qui est à l'écoute. Dans la perspective de Profiling lancée précédemment on a donc la vue de présentation des statistiques d'exécution.

Cette vue se présente sous la forme d'un tableau détaillant pour chaque package de l'application les informations suivantes :

- Base Time : temps d'exécution propre d'une méthode (en excluant les appels à d'autres méthodes)
- Average Base Time : temps d'exécution moyen d'une méthode (Average Base Time = Base Time / Calls)
- Cumulative Time : temps d'exécution total de la méthode (en incluant les appels à d'autres méthodes)

Ici, on se rend compte que la majorité du temps d'exécution provient du package fr.productweb.dao . Un passage en mode pourcentage et une analyse un peu plus profonde nous permettent d'affiner cette affirmation. On se rend compte ici que les appels à la méthode parseContent représentent 70 % du temps d'exécution global de notre application. Afin de déterminer si cette valeur est logique ou non, on double-clique sur la méthode pour obtenir le détail de son exécution et notamment de ses appels éventuels à d'autres méthodes. Le détail de l'invocation de parseContent nous indique clairement que l'appel à createParser est ce qui coûte le plus de temps d'exécution dans cette méthode. En allant inspecter le code de cette méthode, on se rend compte que createParser réalise une création d'instance de parser SAX à chaque appel de parseContent. Le problème est donc bien identifié.

Résolution du problème

Notre problème identifié, nous passons maintenant à la modification du code source de la classe ProductDAO afin de rendre unique la création du parser SAX. Le code modifié ne diffère guère de l'exemple montré précédemment. Il consiste simplement à garder une référence vers le parser SAX dans la classe ProductDAO et à réaliser son instanciation à la construction du DAO :

```
public class ProductDAO {

  private static final String DATA_PATH = «/fr/productweb/data/»;
  private SAXParser parser;

  public ProductDAO(){
    try {
      parser = createParser();
    } catch (ParserConfigurationException e) {
      // TODO
    } catch (SAXException e) {
      // TODO
    }
  }

  // … etc …
}
```

Une fois sorti de la méthode parseContent, la création du parser SAX sera désormais unique dans notre DAO. De ce fait, les appels à la méthode parseContent se contenteront d'utiliser le parser déjà instancié. Afin de mesurer l'impact de ce refactoring sur la classe ProductDAO, nous effectuons de nouveau le profiling de l'application. Les résultats montrent clairement une amélioration significative des performances d'exécution. Le temps d'exécution moyen pour la méthode parseContent et plus globalement pour le DAO Product est désormais plus en rapport avec le traitement qu'ils accomplissent.

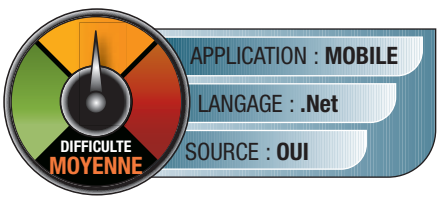
Conclusion

Ce bref exemple nous aura permis de mettre en place le profiling avec TPTP afin de mesurer le temps d'exécution d'une application JEE.

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

Internationalisez votre application Windows Phone 7

Le Marketplace de Windows Phone 7 autorise une diffusion mondiale de l'application, mais aussi des diffusions spécifiques dans des marchés localisés. Par exemple vous pouvez avoir envie de diffuser une version Française de votre application sur les terminaux en langue française, et en Allemand sur les terminaux allemands. C'est ce qu'on appelle la localisation.



De la même manière, certaines valeurs se représentent différemment d'un pays à l'autre : par exemple la date s'écrit au format mm/jj/aaaa dans les pays anglo-saxons mais jj/mm/aaaa dans les pays francophones. En utilisant les bons outils, on peut indiquer au téléphone d'adapter son affichage en fonction des options choisies par l'utilisateur. C'est ce qu'on appelle la globalisation. Localisation et globalisation forment les deux mamelles de l'internationalisation d'une application, et se servent de plusieurs notions qu'il faut connaître :

La notion de culture se réfère à un ensemble de préférences (langage, monnaie, affichage des dates…) qu'on identifie par un code en deux parties : <langage>-<region>. Par exemple, en-us pour anglais-états-unis ou fr-fr pour français-France qui n'est pas la même chose que fr-ca (français-canada). Ce code peut aussi porter le nom de « **Locale** ».

Il y a 3 types de « culture » dans le framework : « invariant » qui désigne justement l'absence de culture choisie, « neutral » qui représente un langage sans association à une région particulière (fr par exemple) et specific qui représente une culture associée à la fois à un langage et une région.

La classe CultureInfo nous permettra de récupérer les informations à propos de la culture comme celle qui a été paramétrée dans le système, le formatage des différents types de données (date, monnaie, etc.) et jusqu'à un calendrier spécifique (et oui, le calendrier hébreu est différent du calendrier japonais !!)

La notion d'encodage, qui est un outil pour représenter les différents caractères utilisés par une langue de façon numérique : les plus connus sont ASCII, UTF-8, Unicode… Il est bon de savoir que Windows Phone 7 utilise l'Unicode. La classe Encoding vous aidera à vous abstraire des problèmes potentiels d'encodage (par exemple lors du téléchargement de données qui ne sont pas encodées en Unicode)

LA GLOBALISATION ET L'UTILISATION DE LA CLASSE CULTUREINFO

La première (et peut-être la seule) chose à savoir est qu'il existe deux instances de la classe CultureInfo dont se sert votre application : il s'agit des propriétés CurrentCulture et CurrentUICulture de l'objet Thread.CurrentThread. Cela permet de savoir par exemple comment un objet de type « DateTime » sera affiché lors de l'appel à sa méthode « ToString() ». Pour choisir la façon d'afficher ces types de données d'une manière qui n'est pas celle choisie à l'échelle du système (dans le menu Settings du terminal Windows Phone 7) il suffit de créer une nouvelle instance de la classe CultureInfo en lui passant en paramètre la locale souhaitée (par exemple fr-FR) et assigner cette instance aux propriétés CurrentCulture et CurrentUICulture.

```
CultureInfo ci = new CultureInfo(«fr-FR»);
Thread.CurrentThread.CurrentCulture = ci;
Thread.CurrentThread.CurrentUICulture = ci;
```

LA LOCALISATION – BIEN RÉPARTIR SES RESSOURCES POUR SUPPORTER PROPREMENT DIFFÉRENTS LANGAGES

Comme par le passé avec le .NET Compact Framework, la première étape de la localisation consiste à utiliser des fichiers de ressources (extension .resx). On peut ajouter ces fichiers avec dans le menu « Add -> Add New Item » en choisissant « Resource File » [Fig.1].

Attention, pour que le framework retrouve ses petits il faut adopter une stratégie de nommage particulière : d'abord un fichier de ressources par défaut qu'on nomme <NomDuFichierRessources>.resx puis ensuite les fichiers de ressources localisées qu'on nommera de la façon suivante : <NomDuFichierRessources>.<Locale>.resx.

Je vous conseille de commencer par créer uniquement le fichier par défaut, y insérer toutes vos chaînes de caractères, et ensuite seule-

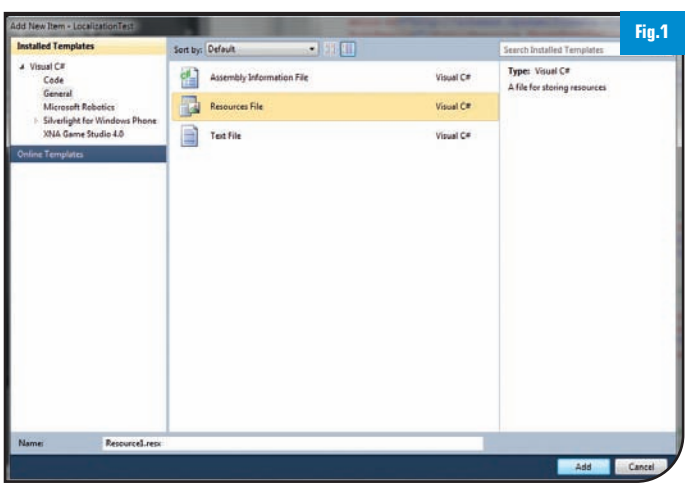


Fig.1

ment, une fois que toutes les ressources par défaut sont créées, copier/coller et renommer le fichier pour les différentes cultures à supporter, avant de changer le contenu des chaînes de caractères : de cette manière vous êtes sûr que toutes les chaînes de caractères localisées ont le même nom, indépendamment de la culture…
Il faut ensuite définir la culture par défaut supportée par l'application (celle qui utilise le fichier <NomDuFichierRessources>.resx). Pour cela dans l'explorateur de solution, cliquez sur le nom du projet et choisissez ses propriétés (Properties). Dans l'onglet « Application », cliquez sur le bouton « Assembly Information » et dans la liste « Neutral language », choisissez la culture par défaut [Fig.2].

Dernière opération « manuelle » s'il en est, fermez votre projet et ouvrez le fichier .csproj dans un éditeur de texte : c'est en fait un fichier xml. Trouvez le tag « <SupportedCultures> » et ajoutez toutes les locales que votre application doit supporter (y compris la locale par défaut) dans ce tag, séparées par des points-virgules.

```
<SupportedCultures>en-US;fr-FR;</SupportedCultures>
```

Il ne reste plus qu'à utiliser ces fichiers de ressources dans votre projet : là encore, il faut suivre quelques étapes : c'est un peu plus compliqué qu'avec le .NET Compact Framework car il faut que ces chaînes de caractères localisées soient accessibles à la fois depuis le code et le XAML sous forme de ressources [Fig.3].
D'abord, ouvrez chacun de vos fichiers ressources et dans la liste « Access Modifier » en haut de l'éditeur, sélectionnez la valeur « Public ». Il faut le faire pour chacun des fichiers ressources.
Ensuite, il faut créer une nouvelle classe dans votre application (qu'on appellera comme on veut, j'utilise ici « LocalizedStrings » car c'est ce qui est suggéré par la documentation MSDN) dans laquelle on créera une propriété qui renverra vers ces fichiers de ressources fraîchement créés : dans le bout de code suivant, LocalizedResources est le nom que j'ai choisi pour mon fichier de ressources, on retrouve donc une classe de ce type dans l'espace de nommage de mon application.

```
public class LocalizedStrings
{
  public LocalizedStrings()
  { }

  private static LocalizedResources locTestResources = new Localized
Resources();
  public LocalizedResources LocTestResources { get { return locTest
Resources; } }
}
```

Après ça, il faut ouvrir le fichier App.xaml et dans la section « <Application.Resources> » il faut rajouter le bout de code suivant :

```
<Application.Resources>
  <local:LocalizedStrings xmlns:local=»clr-namespace:Localization
Test» x:Key=»LocalizedStrings» />
</Application.Resources>
```

Si votre projet dispose de plusieurs fichiers de localisation, il suffit de rajouter des propriétés dans la classe LocalizedStrings.
Enfin, il n'y a plus qu'à utiliser ces ressources dans le code XAML avec un simple Binding de la façon suivante :

```
{Binding Path=LocTestResources.PageName, Source={StaticResource
 LocalizedStrings}}
```

Et dans le code C#, c'est d'une simplicité déconcertante : il suffit d'accéder à la classe LocalizedResources :

```
PageTitle.Text = LocalizedResources.PageName;
```

Vous voilà maintenant en mesure de supporter plusieurs langages dans une même version de votre application, effort qui saura sans aucun doute séduire vos utilisateurs. Attention cependant à bien réfléchir à vos traductions !!

■ Pierre **Cauchois**

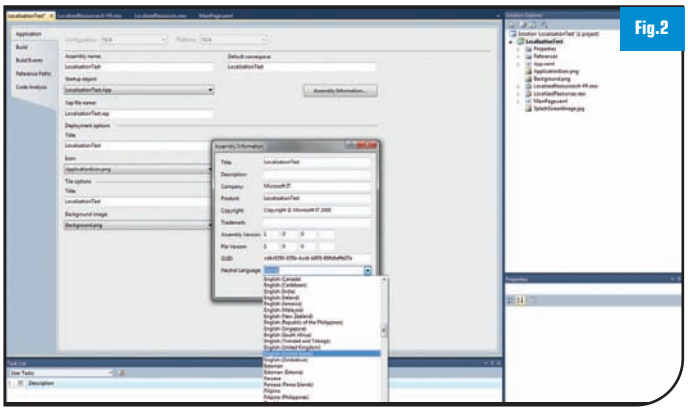


Fig.2

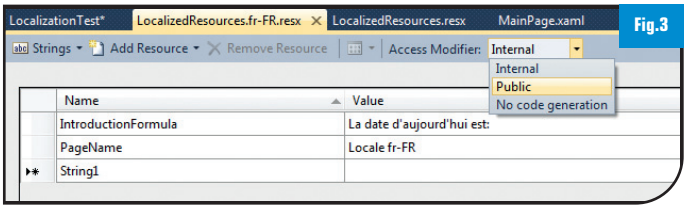
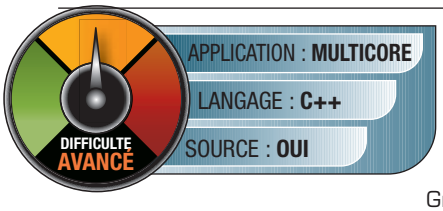


Fig.3

Programmation parallèle avec OpenCL 2e partie

Le mois dernier nous avons découvert les maniements de base de l'API OpenCL. Nous abordons aujourd'hui des notions plus avancées telles que la synchronisation des kernels et le profiling des applications



Résumons-nous. OpenCL est une API dédiée à la programmation concurrente et spécifiée par un consortium d'industriels, le Khronos Group. Ce consortium définit également l'API 3D OpenGL et nous ne serons donc pas surpris qu'OpenCL soit proche du graphisme. De fait OpenCL a pour but d'exploiter les possibilités des processeurs multi-cœurs, mais aussi les capacités des processeurs graphiques modernes, qui sont de plus en plus programmables et puissants, et de faire ainsi de votre PC de bureau une vraie bête de calcul.

Une application OpenCL s'articule autour de la notion de Platform, qui est l'implémentation d'OpenCL, de Devices qui sont les composants programmables (CPU, GPU), d'un contexte qui est une abstraction des deux notions précédentes, et d'une file de commande qui dans la terminologie OpenCL est appelée une *Command Queue*. Dans cette file de commande, le programmeur injecte des instructions de transferts bidirectionnels de données entre le corps de l'application et les Devices. En outre, et c'est l'essence d'une application OpenCL, le programmeur injecte des kernels qui sont de petites routines que le runtime se chargera de répartir sur les coeurs des Devices pour exécution parallèle. Les kernels sont compilés à la volée lors de l'initialisation de l'application. Dans nos exemples du mois dernier, nous nous sommes appuyés sur une particularité d'OpenCL qui est, par défaut, d'assurer que tous les commandes poussées dans la file soient exécutées l'une après l'autre. On peut ainsi voir la file de commande comme une pile FIFO (First In First Out). Les commandes sont exécutées de façon séquentielle et dans ce cas le programmeur n'a pas à se préoccuper de problèmes de synchronisation. Cependant OpenCL permet une autre approche.

1 LA PROGRAMMATION DANS LE DÉSORDRE

Il est possible de configurer une file de commande afin que les commandes qui y sont poussées ne soient plus exécutées dans l'ordre selon lequel elles sont poussées. La question est alors : *comment tout cela va-t-il se comporter ?* Et la réponse est : *on ne sait pas du tout, mais ce qui est certain, c'est que le programmeur devra utiliser des objets de synchronisation.* Ainsi, si un kernel B doit manipuler des données qui doivent elles-mêmes être produites par un kernel A, alors il faudra s'assurer que l'exécution du kernel A est terminée, et pour cela le programmeur dispose de plusieurs possibilités. Dans quelles situations utiliser ce mode particulier ? A priori dans le cas d'une application constituée de nombreux kernels ayant des temps d'exécution très inégaux. On est alors tenté d'imaginer que le runtime va exécuter ces kernels parallèlement sur les cœurs à disposition et qu'ainsi les kernels rapides ne seront pas contraints d'attendre la fin de l'exécution de kernels plus lents. C'est du moins ce qu'on se dit intuitivement, mais l'expérimentation à ce niveau apporte son lot de surprises. En effet les spécifications d'OpenCL sont assez vagues pour ce mode désordre (out-of-order). Ces spécifications nous disent que l'ordre d'exécution des kernels n'est plus garanti, mais elle laisse l'implémentation libre de faire ce qu'elle veut. Ainsi il n'est pas dit que l'implémentation DOIT paralléliser des kernels inégaux. En outre une implémentation va travailler avec des composants GPU et/ou CPU et ceux-ci sont très différents. Un CPU multi-cœur se prête bien à la parallèlisation de kernels différents, tandis qu'un GPU se prête bien à l'exécution parallèle d'un très grand nombre d'instances d'un même kernel. Il est donc extrêmement difficile d'anticiper le comportement 'désordonné' d'une application OpenCL et nous touchons là les limites de la portabilité de cette API. En mode séquentiel pas de problème, mais en mode désordre c'est une autre histoire, que notre premier exemple va illustrer.

2 UN HELLO WORLD PARALLÉLISÉ

Notre programme d'exemple doit produire la chaîne Hello World! dans un buffer au moyen de trois kernels. Les données initiales sont séparées en Hello et World, chacun dans un buffer. Un premier kernel (Hello) va copier les données Hello dans un buffer intermédiaire. Ce premier kernel est ralenti par une énorme boucle. Le second kernel (World), copie les données du buffer intermédiaire, dans un second buffer intermédiaire et y ajoute les données World!, enfin le dernier kernel (HelloWorld) copie le contenu du second buffer dans le buffer final. En théorie le premier kernel est très lent, et l'on doit être sûr que son exécution est terminée lorsque le second kernel démarre, sinon celui-ci produira un résultat incorrect. Voici maintenant un extrait du code. Nous avons retiré la partie qui initialise l'application et qui est identique à ce que nous avons vu le mois dernier. Comme le mois dernier, nos exemples ne gèrent pas les erreurs afin d'alléger le code. Bien entendu le code complet de cet exemple est disponible sur notre site.

```
#include <iostream>
using namespace std;
#include <CL/opencl.h>

const char* program_source[] =
{
 «__kernel void Hello( __global const char* a, __global char* tempa)»,
 «{«,
   «long i;»,
   «float f;»,
   «for(i=0 ; i<10000000; i++)»,
   « f = sin(0.5) + cos (0.4) + tan (0.3);»,
   «int iGID = get_global_id(0);»,
   «tempa[iGID] = a[iGID];»,
 «}»,

 «__kernel void World(__global const char* tempa, __global const
 char* b, __global char* tempb)»,
```

```
 «{«,
   «int iGID = get_global_id(0);»,
   «tempb[iGID] = tempa[iGID];»,
   «tempb[iGID+6] = b[iGID];»,
 «}»,

 «__kernel void HelloWorld(__global const char* tempb, __global
 char* c)»,
 «{«,
   «int iGID = get_global_id(0);»,
   «c[iGID] = tempb[iGID];»,
 «}»,
};

int main(int argc, char **argv)
{
 // Voir le code sur le site pour
 // l'initialisation de l'application
 const cl_char a[] = «Hello «;
 const cl_char b[] = «World!»;
 cl_char c[] = «————»; // le résultat de l'addition sera
stocké ici


 // Buffer pour la donnée a
 cl_mem buffera = clCreateBuffer (context,
   CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR,
   sizeof(a)-1, // pas de zéro final dans le buffer
   (void*)&a,
   &error);

 // Buffer temporaire a
 cl_mem buffertempa = clCreateBuffer (context,
   CL_MEM_READ_WRITE | CL_MEM_COPY_HOST_PTR,
   sizeof(a)-1, // pas de zéro final dans le buffer
   (void*)&c,
   &error);

 // Buffer pour la donnée b
 cl_mem bufferb = clCreateBuffer (context,
   CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR,
   sizeof(b)-1, // pas de zéro final dans le buffer
   (void*)&b,
   &error);

 // Buffer temporaire b
 cl_mem buffertempb = clCreateBuffer (context,
   CL_MEM_READ_WRITE | CL_MEM_COPY_HOST_PTR,
   sizeof(c)-1, // pas de zéro final dans le buffer
   (void*)&c,
   &error);


 // Buffer pour la données resultat c
 cl_mem bufferc = clCreateBuffer (context,
   CL_MEM_WRITE_ONLY,
   sizeof(c)-1, // Pas de zéro final dans le buffer
   (void*)&c,
   &error);

 cl_program program = clCreateProgramWithSource(
   context,
   sizeof(program_source)/sizeof(char*),
   program_source,
   NULL,
   &error);

 cl_int result = clBuildProgram(program, 0, NULL, NULL, NULL, NULL);
 cl_kernel kernel_hello = clCreateKernel(program, «Hello», NULL);
 cl_kernel kernel_world = clCreateKernel(program, «World», NULL);
 cl_kernel kernel_helloworld = clCreateKernel(program, «Hello
World», NULL);


 result = clSetKernelArg(kernel_hello, 0, sizeof(cl_mem), &buffera);
 result = clSetKernelArg(kernel_hello, 1, sizeof(cl_mem), &buffer
tempa);

 result = clSetKernelArg(kernel_world, 0, sizeof(cl_mem), &buffer
tempa);
 result = clSetKernelArg(kernel_world, 1, sizeof(cl_mem), &bufferb);
 result = clSetKernelArg(kernel_world, 2, sizeof(cl_mem), &buffer
tempb);

 result = clSetKernelArg(kernel_helloworld, 0, sizeof(cl_mem),
 &buffertempb);
 result = clSetKernelArg(kernel_helloworld, 1, sizeof(cl_mem),
 &bufferc);

 result =  clSetCommandQueueProperty(command_queue,
   CL_QUEUE_OUT_OF_ORDER_EXEC_MODE_ENABLE,
   CL_TRUE,
   NULL);

 const size_t global_work_size_hello[] = {
   sizeof(a)-1,
 };

 const size_t global_work_size_world[] = {
   sizeof(b)-1,
 };

 const size_t global_work_size_helloworld[] = {
   sizeof(c)-1,
 };


 result = clEnqueueNDRangeKernel (command_queue,
         kernel_hello,
         1,
         NULL,
         global_work_size_hello,
         NULL, 0, NULL, NULL);


 result = clEnqueueNDRangeKernel (command_queue,
```

```
        kernel_world,
        1,
        NULL,
        global_work_size_world,
        NULL, 0, NULL, NULL);

result = clEnqueueNDRangeKernel (command_queue,
        kernel_helloworld,
        1,
        NULL,
        global_work_size_helloworld,
        NULL, 0, NULL, NULL);


result = clEnqueueReadBuffer(command_queue,
  bufferc,
  CL_TRUE,
  0,
  sizeof(c)-1,
  &c,
  0, NULL, NULL);

cout << «Resultat: « << c << endl;

clReleaseKernel(kernel_hello);
clReleaseKernel(kernel_world);
clReleaseKernel(kernel_helloworld);
clReleaseProgram(program);


clReleaseMemObject(bufferc);
clReleaseMemObject(buffertempb);
clReleaseMemObject(bufferb);
clReleaseMemObject(buffertempa);
clReleaseMemObject(buffera);
clReleaseCommandQueue(command_queue);
clReleaseContext(context);

cin.get();
}
```

On remarquera, au milieu de ce code, que l'on bascule en mode désordre au moyen de la fonction *clSetCommandQueueProperty*. On remarquera aussi qu'aucune synchronisation n'est faite. Le premier kernel étant volontairement ralenti par un code idiot, avons-nous le comportement supposé, à savoir l'exécution du deuxième kernel avant la fin du premier ? Ayant fait de nombreux essais avec le SDK OpenCL 1.0 proposé par NVIDIA je dois répondre non à cette question, étant entendu que cette réponse n'est valable que pour ce SDK et que pour l'exécution de kernels, car je n'ai pas fait d'essais sur les opérations de transferts de données.

Que se passe-t-il ? D'abord, le code ralentissant le kernel étant un code mort, on peut envisager que le compilateur du SDK comporte un optimisateur qui supprime ce code. A la lumière d'autres expérimentations qui dépassent le cadre de cet article, il semble que cela ne soit pas le cas. L'explication la plus plausible tient tout simplement dans la nature de l'implémentation d'OpenCL par NVIDIA, qui ne tient compte que des GPU. Ceux-ci sont bien exploités lorsqu'ils exécutent parallèlement un grand nombre de petits kernels identiques à exécution courte, cas de figure qui est classique dans l'univers du graphique. Du coup NVIDIA ne semble pas s'être intéressé au cas de deux ou trois kernels différents et à temps d'exécution très différents, ce qui est le cas de notre exemple et qui est un cas qui trouve son application sur les CPU. Pour le coup nous déduisons que NVIDIA a profité de la liberté offerte par les spécifications: les commandes peuvent être exécutées dans le désordre mais rien n'y oblige et NVIDIA ne semble n'avoir rien implémenté et garde le comportement séquentiel. Bien sûr, avec une autre implémentation d'OpenCL le comportement peut-être complètement différent. Nous savons que le SDK ATI Stream d'AMD prend, lui, les CPU en charge. Nous invitons donc chaleureusement les lecteurs qui auraient observé des comportements différents avec ce SDK à partager leurs informations sur notre forum, dans le fil de discussion dédié à cet article. Nous poursuivons maintenant cet article en faisant «comme si» et nous passons en revue les possibilités de synchronisation.

3 EXEMPLE DE SYNCHRONISATION EN CASCADE

Le runtime OpenCL peut fournir un objet de synchronisation (event) pour chaque commande poussée dans la file. De même, on peut demander à toute commande poussée dans la file de ne démarrer que quand un objet de synchronisation le permet. Notre nouvel exemple, *DemoNDRangeEvent* est une modification de l'exemple précédent, pour assurer une exécution en cascade (ou séquentielle) des commandes en mode désordre. Voici l'extrait intéressant (le code complet est disponible sur notre site).

4 PLACER DES BARRIÈRES

Le code précédent montre qu'une commande peut attendre plusieurs événements, ceux-ci étant placés dans un tableau. Une commande *clEnqueueNDRange kernel* peut donc se comporter comme une barrière. Toutefois il existe une commande spécifique pour tenir ce rôle. Notre exemple suivant, *DemoEnqueueWaitForEvents* disponible sur notre site, illustre comment mettre en œuvre cette commande particulière qui se nomme *clEnqueueWairForEvents*. Nous n'en donnons qu'un très court extrait ici :

```
cl_event eventa, eventb, eventc;
cl_event event_list[1];

result = clEnqueueNDRangeKernel (command_queue,
        kernel_hello,
        1,
        NULL,
        global_work_size_hello,
        NULL, 0, NULL, &eventa);

event_list[0] = eventa;
cl_uint num = sizeof(event_list)/sizeof(cl_event);

result = clEnqueueWaitForEvents (command_queue, num, event_list);
```

Il existe encore la commande *clEnqueueBarrier* qui permet d'attendre la fin de l'exécution de toutes les commandes qui ont été placées avant elle dans la file.

5 SYNCHRONISER DEPUIS LE THREAD PRINCIPAL DE L'APPLICATION

Il est important de remarquer que les opérations précédentes sont obtenues à partir de commandes. La synchronisation est donc effectuée par le device, ou dans le thread du driver de celui-ci. Toutefois on peut préférer que les synchronisations soient effectuées dans le thread principal de l'application. Cela devient une obligation, au moins avec le SDK NVIDIA, en ce qui concerne les opérations de profiling, ainsi que nous le verrons plus loin. Pour synchroniser depuis le thread principal de l'application, on emploie la fonction *clWaitForEvents* comme le démontre l'exemple *DemoWaitForEvent,* disponible sur notre site, et dont nous ne donnons qu'un extrait minimum :

```
cl_event eventa, eventb, eventc;
cl_event event_list[1];

result = clEnqueueNDRangeKernel (command_queue,
        kernel_hello,
        1,
        NULL,
        global_work_size_hello,
        NULL, 0, NULL, &eventa);

event_list[0] = eventa;
cl_uint num = sizeof(event_list)/sizeof(cl_event);

result = clWaitForEvents (num, event_list);
```

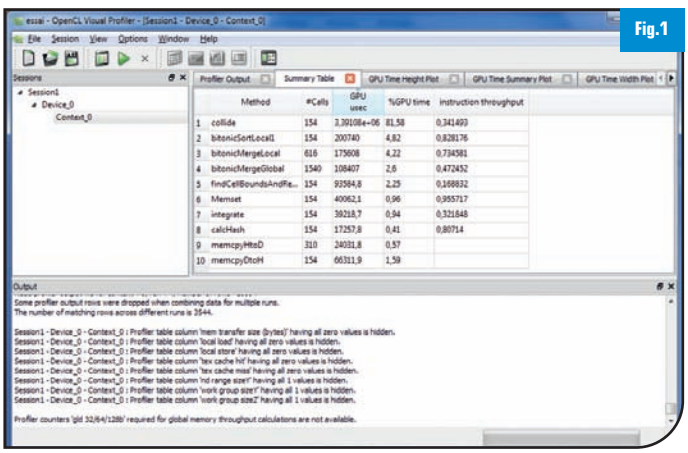


Fig.1

Le SDK OpenCL NVIDIA vient avec un outil de profiling

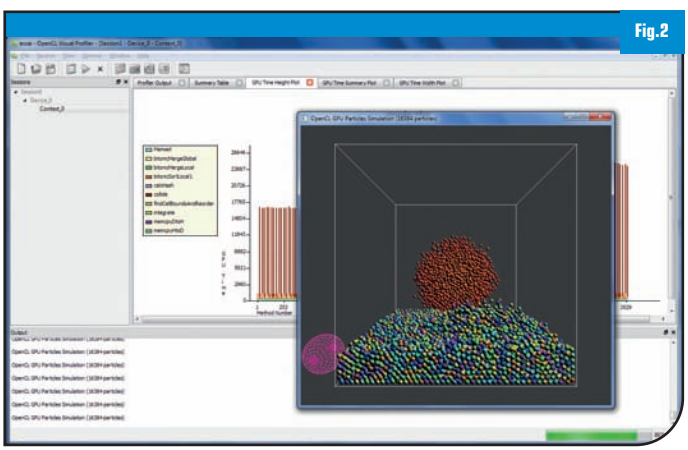


Fig.2

L'exemple oclParticles du SDK en cours de profiling

6 OBTENIR LE STATUT DES COMMANDES

Il est théoriquement possible d'obtenir le statut des commandes. Il y quatre états possibles : *queued, submitted, running* et *complete*. Dans la pratique, avec le SDK NVIDIA, nous n'avons pu obtenir que les états extrêmes: *queued* et *complete.*

Les autres sont sans doute trop éphémères pour les rencontrer autrement que sur un coup de chance.

Nous renvoyons le lecteur au code de l'exemple *DemoEventInfo*, dont le code est disponible sur notre site, et qui montre comment utiliser la fonction *clGetEventInfo*.

7 PROFILER L'EXÉCUTION DES KERNELS

OpenCL prévoit qu'il soit possible d'obtenir des informations sur les temps d'exécution des commandes pour peu que la file soit configurée pour cela. Si l'on est paresseux, on peut se contenter d'utiliser l'outil OpenCL Visual Profiler fourni par le SDK NVIDIA.

L'illustration [Fig.1] montre le résultat du profiling de l'exemple *oclParticles* du SDK, et qui est un démonstration de collisions de 16384 objets. [Fig.2] Si l'on est courageux, on peut écrire un peu de code :-)

Notre dernier exemple, DemoProfiling montre comment procéder. Il est calqué sur les exemples précédent, mais ne lance que le kernel Hello. Le code complet est disponible sur notre site.

Ce code ne lance le kernel qu'une fois. Après avoir essayé ainsi, nous suggérons au lecteur de lancer le kernel 500 fois parallèlement en modifiant la taille du NDRange comme ceci :

```
const size_t global_work_size_hello[] = {
  500,
};
```

Sur ma machine, le temps demandé pour 500 fois plus de travail n'est que 2.4 fois plus grand, ce qui en dit long sur les performances des GPU en calcul parallèle.

Dans les spécifications OpenCL, il est théoriquement possible d'accéder aux informations de profilage à tout moment et de n'importe où. Dans la pratique, avec le SDK NVIDIA, ces informations semblent être une structure de données qui est rafraîchie en temps réel par le runtime, ce qui est normal, mais cette structure n'est lisible que depuis le thread principal de l'application.

C'est pourquoi nous lisons ces informations en aval d'un appel à la fonction *clWaitForEvent*.

8 CONCLUSION

Même si les spécifications d'OpenCL laissent des libertés aux implémentations qui peuvent nuire à la portabilité des applications, cette API se révèle facile et agréable à utiliser une fois prise en main. En outre elle possède la caractéristique unique de permettre de travailler avec CPU et GPU, et selon les tâches à effectuer, de profiter au mieux des atouts de l'un ou de l'autre, à condition de disposer d'une implémentation complète.

OpenCL devrait rapidement s'imposer dans le monde de la programmation parallèle.

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

DÉVELOPPEZ VOTRE SAVOIR-FAIRE

Economisez jusqu'à 50%

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

Pour votre développement personnel et professionnel, abonnez-vous à Programmez !
www.programmez.com

1 -25%

Abonnement 1 an

49€ au lieu de 65,45 € tarif au numéro - *Tarif France métropolitaine*

2 +0,8€ par mois

Abonnement Intégral : + archives

1 an au magazine + archives sur Internet et PDF

59€ *Tarif France métropolitaine*

3 jusqu'à -50%

Abonnement 2 ans + 1 livre numérique ENI

- **79€** au lieu de 130,90 (valeur de 22 numéros) *Tarif France métropolitaine*
  + un livre d'une valeur de 23,9 € à 31,9 €, soit un total de 154,8 € à 162,8 €
- **89€** 2 ans au magazine + archives sur Internet et PDF + 1 livre numérique ENI

OUI, *je m'abonne* — *Vous pouvez vous abonner en ligne et trouver tous les tarifs* www.programmez.com

- ☐ **Abonnement 1 an au magazine : 49 €** (au lieu de 65,45 € tarif au numéro) *Tarif France métropolitaine*
- ☐ **Abonnement Intégral : 1 an au magazine + archives : 59 €** *Tarif France métropolitaine*
- ☐ **Abonnement 2 ans au magazine + livre numérique ENI : 79 €** *Tarif France métropolitaine*
- ☐ **Abonnement 2 ans au magazine + livre numérique ENI + archives : 89 €** *Tarif France métropolitaine*

Livres à Choisir : ☐ Visual Studio 2010 ☐ PHP5.3 ☐ Bing Maps ☐ MySQL 5, Administration et optimisation
☐ Java et Spring, Concevoir, construire et développer une application Java/J2EE avec Spring. *Détails sur www.programmez.com/abonnement.php*

☐ M. ☐ Mme ☐ Mlle Entreprise : Fonction :

Prénom : Nom :

Adresse :

Code postal : Ville :

Tél : **(Attention, e-mail indispensable)**

E-mail : @

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

A remplir et retourner sous enveloppe affranchie à : Programmez ! - Service Abonnements - 22 rue René Boulanger - 75472 Paris Cedex 10.
abonnements.programmez@groupe-gli.com

Offre limitée, valable jusqu'au 30 novembre 2010

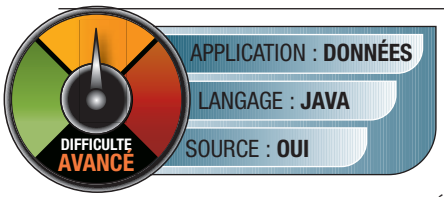
Le renvoi du présent bulletin implique pour le souscripteur l'acceptation pleine et entière de toutes les conditions de vente de cette offre. Conformément à la loi Informatique et Libertés du 05/01/78, vous disposez d'un droit d'accès et de rectification aux données vous concernant. Par notre intermédiaire, 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 135

Le magazine du développement PROgrammez!

Cassandra, une base de données distribuée NoSQL

Développée à l'origine par les ingénieurs de Facebook, distribuée, tolérante aux fautes et NoSQL, Cassandra est une base de données très originale. Découvrons là.



Stocker des données est un besoin récurrent en informatique. La pratique établie consiste à utiliser un système de base de données relationnelle (SGDBR) tel que MySQL, SQL Server, PostgreSQL, Oracle ou autre. Avec ces systèmes, les données sont organisées dans des tables, conformément à un modèle relationnel. Il est alors possible de retrouver une information quelconque dans la table au moyen de l'algèbre relationnelle, exprimée par les fameuses requêtes SQL.

Cette approche du stockage de données a largement fait ses preuves. Mais les très gros volumes de données et les architectures distribuées (clusters) font apparaître de nouvelles solutions, même si les bases de données classiques se sont adaptées aux clusters. Les bases de données NoSQL, si elles ne sont pas nouvelles, les premières ayant vu le jour en 1998, commencent à faire leur apparition sur la scène informatique, en situation de production. Ainsi en est-il de la base de données Cassandra qui a été initialement développée par les ingénieurs de Facebook. Offerte à la communauté Open Source, Cassandra est actuellement un des projets phare de la fondation Apache et elle est utilisée par de très gros de l'Internet comme Twitter ou Digg. Cassandra est écrite en Java. C'est une base de données distribuée qui rend très facile la mise en place d'un cluster.

Un nœud défaillant peut être réparé à chaud et une défaillance d'un nœud dans le cluster ne provoque pas une défaillance générale du système puisque les données sont automatiquement répliquées sur tous les nœuds qui sont tous identiques. Cassandra est une base de données intéressante et amusante à découvrir, même pour un amateur qui voudra mettre en place un mécanisme de réplication de données sur son réseau personnel.

Le seul point noir est la carence d'une documentation claire, et écrite à l'attention de ceux qui ne savent pas déjà, aussi bien pour la base de données elle-même qu'en ce qui concerne les librairies client. Votre serviteur espère que cet article vous aidera à vous y retrouver.

1 MANIPULATIONS DE BASE

Cassandra, écrite en Java, vient sous la forme d'une archive qu'il suffit de décompresser à l'endroit de votre choix. Les classes Java résident dans le sous-répertoire lib de l'arborescence ainsi obtenue. Dans le répertoire bin vous trouverez des scripts shell (Unix / Linux) et batch (Windows) qui permettent de lancer et administrer la base de données. Enfin, le sous-répertoire conf et les fichiers storage-conf.xml et log4j.properties méritent toute notre attention ainsi que nous le verrons un peu plus loin. Même si Cassandra est destinée aux clusters, il est parfaitement possible de l'utiliser sur un seul poste. Sous Windows, c'est même la seule application possible au moment de la rédaction de cet article qui est basé sur Cassandra 0.6.1. Lancer Cassandra en monoposte est très simple. D'abord on définira deux variables d'environnement JAVA_HOME et CASSANDRA_HOME, la première pointant sur le répertoire racine du Java 1.6 qui doit être installé sur votre système, la seconde pointant sur le répertoire racine de Cassandra. Ensuite Cassandra utilise les ports 9160 et 7000 par défaut. Votre pare-feu doit donc ouvrir ces ports. Enfin on doit éditer le fichier storage-conf.xml pour définir les répertoires de travail de Cassandra. Par défaut le fichier contient :

```
<CommitLogDirectory>
  /var/lib/cassandra/commitlog
</CommitLogDirectory>
<DataFileDirectories>
  <DataFileDirectory>
    /var/lib/cassandra/data
  </DataFileDirectory>
</DataFileDirectories>
```

Cela ne conviendra peut-être pas à votre système Linux, et cela ne convient pas à Windows. Par exemple pour ce dernier, donnez :

```
<CommitLogDirectory>
  c:\cassandra-logs
</CommitLogDirectory>
<DataFileDirectories>
  <DataFileDirectory>
    c:\cassandra-data
  </DataFileDirectory>
</DataFileDirectories>
```

cassandra-logs et cassandra-data étant deux répertoires que vous aurez pris soin de créer. Enfin dans le fichier log4j.properties qui se situe dans le répertoire conf de Cassandra, vous trouverez :

```
# Edit the next line to point to your logs directory
log4j.appender.R.File=/var/log/cassandra/system.log
```

Là aussi vous devez adapter cette ligne selon votre système. Il est à noter que je suis tombé sur un bug sous Linux avec Cassandra 0.6.1. J'avais donné cassandra-logs comme répertoire, ce qui avait pour effet de provoquer une exception out-of-memory faisant échouer le démarrage de la base de données. Tout va bien en spécifiant le répertoire de base de Cassandra, donc par exemple :

```
# Edit the next line to point to your logs directory
log4j.appender.R.File=c:\cassandra\system.log
```

Tout est prêt, vous pouvez lancer Cassandra avec le script cassandra (.sh ou .bat) du répertoire bin :

```
cassandra -f
```

Le commutateur -f provoque la sortie de messages sur le terminal lors du démarrage de Cassandra. Si vous ne voyez pas de messages d'exception et que le dernier dit "Cassandra startup up...", tout est en ordre. [Fig.1] Sinon vérifiez le contenu des deux fichiers de configuration, et vérifiez que tout est en ordre du côté de votre pare-feu.

2 PREMIER CONTACT AVEC LE MODÈLE DE DONNÉES

Prenons contact avec le modèle de données de Cassandra, à travers le client en ligne de commande qu'elle propose, à l'instar de tous les systèmes de base de données. Ce client se lance par le script cassandra (.sh ou .bat) du répertoire bin.

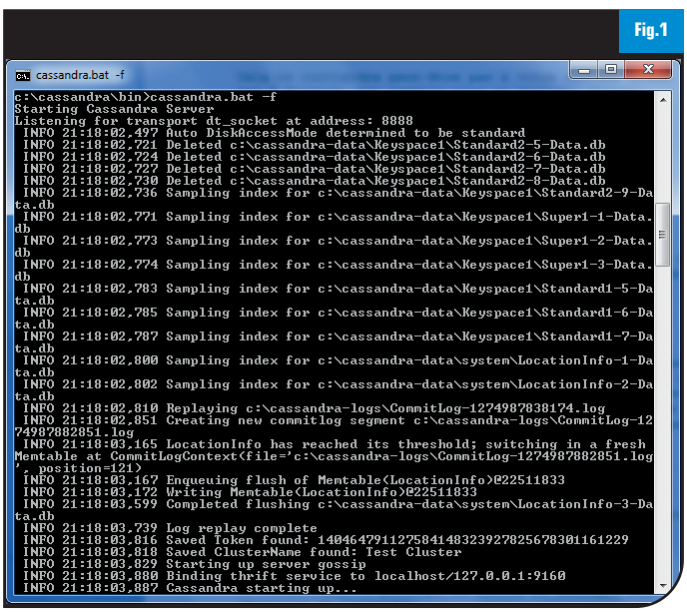
```
cassandra-cli -host localhost -port 9160
```

Une fois le client lancé, vous pouvez saisir la commande help pour connaître les commandes à votre disposition. Essayons une commande :

```
cassandra> get Keyspace1.Standard2['fred']
Returned 0 results.
```

Les termes étranges que sont Keyspace1 et Standard2 seront expliqués plus loin. Aucun résultat n'est retourné, puisqu'il n'existe encore aucun enregistrement sous la clé fred. Créons un enregistrement :

Fig.1



Démarrage de Cassandra.

```
cassandra> set Keyspace1.Standard2['fred']['Nom'] = 'Mazue'
Value inserted.
```

Nous pouvons maintenant le lire :

```
cassandra> get Keyspace1.Standard2['fred']
=> (column=Nom, value=Mazue, timestamp=1274992588147000)
Returned 1 results.
```

Nous remarquons que 'Nom' est un intitulé de colonne, bien que nous n'ayons pas du tout spécifié de colonne explicitement. Nous reviendrons sur ce point. Je n'ai pas utilisé de caractères accentués (Mazue au lieu de Mazué) parce que le terminal Windows ne les affiche par correctement. Mais, et fort heureusement, Cassandra les supporte très bien. Enfin Cassandra a ajouté automatiquement un timbre à date à notre enregistrement. Continuons un petit peu :

```
cassandra> set Keyspace1.Standard2['fred']['Prenom'] = 'Frederic'
Value inserted.
cassandra> get Keyspace1.Standard2['fred']
=> (column=Prenom, value=Frederic, timestamp=1274992896586000)
=> (column=Nom, value=Mazue, timestamp=1274992588147000)
Returned 2 results.
cassandra> get Keyspace1.Standard2['fred']['Prenom']
=> (column=Prenom, value=Frederic, timestamp=1274992896586000)
```

Nous voyons qu'à partir de la clé fred, nous pouvons soit obtenir toutes les données, ou plus exactement toutes les colonnes, soit obtenir une colonne particulière. Cassandra est une base de données orientée colonne, nous expliquerons cette notion après avoir mis en place un cluster.

3 METTRE EN PLACE UN CLUSTER

Pour cela vous devez disposer de machines sous Linux/Unix connectées en réseau, au moins deux, et connaître leurs adresses IP. Bien entendu, cela fonctionnera parfaitement avec des noms de machines si un DNS est configuré, ou si les fichiers host sont correctement renseignés.

Chaque machine tournant avec Cassandra constitue un noeud du cluster. Les nœuds ont besoin de communiquer entre eux. Ils le font par défaut sur le port 8080, ce qui est un choix malheureux car ce port est souvent occupé par un autre service sur une machine de développement. On spécifiera donc un autre port, par exemple 9090. Ceci se fait dans le fichier cassandra.in.sh du répertoire bin :

```
# Arguments to pass to the JVM
# etc
-Dcom.sun.management.jmxremote.port=9090 \
```

Nous devons maintenant retourner au fichier storage-conf.xml de chaque nœud pour spécifier l'adresse (ou le nom d'hôte) depuis lequel le nœud est à l'écoute. Cela revient à remplacer

```
<ListenAddress>localhost</ListenAddress>
```

Par, par exemple

```
<ListenAddress>192.168.1.10</ListenAddress>
```

Puis nous devons dire à chaque nœud combien de fois il doit répliquer ses données. Donner un nombre égal au nombre de noeuds assure la réplication sur tous les nœuds :

```
<ReplicationFactor>2<ReplicationFactor>
```

Enfin les nœuds doivent être capables de se trouver et recenser les uns les autres. Pour cela chaque nœud doit connaître l'adresse IP ou le nom d'hôte d'un autre nœud au moins, cet autre est appelé seed. Ainsi dans le fichier storage-conf.xml de la machine d'adresse 195.168.1.10, on donnera par exemple :

```
<Seeds>
  <Seed>192.198.1.54</Seed>
</Seeds>
```

Tout est prêt, même si nous n'avons pas épuisé les secrets de storage-conf.xml, loin s'en faut. Il vous reste à lancer Cassandra sur chaque machine. Ceci fait, vérifiez votre cluster avec l'outil nodetools, qui fera l'inventaire des nœuds actifs :

```
./nodetool --host 192.168.1.10 --port 9090 ring
```

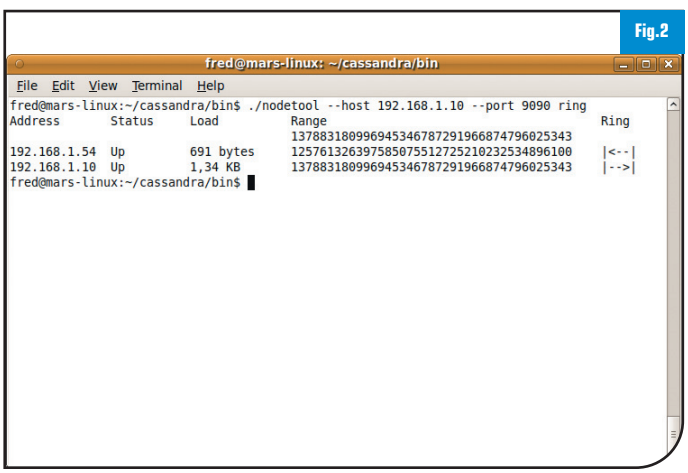
Si tout va bien, vous obtiendrez un résultat semblable à celui montré par la capture. [Fig.2] Maintenant vous pouvez créer un enregistrement depuis le client à partir d'un nœud, et vous verrez, en interrogeant un autre nœud depuis un autre client que les données y ont été dupliquées :-)

4 COMPRENDRE LE MODÈLE DE DONNÉES DE CASSANDRA

Nous avons dit que Cassandra est une base de données orientée colonnes. Voyons maintenant ce que cela signifie. Avec une base de données classique, les données sont rangées dans des tables :

Clé (ou index)	Nom	Prenom	Salaire
fred	Mazué	Frédéric	1
ft	Tonic	François	1000
jk	Kaminsky	Jean	

Fig.2



L'outil nodetool vous confirmera que votre cluster est bien configuré.

Dans cette organisation classique, on ne modifie pas la table lors des opérations courantes, et un NULL, comme par exemple ici le salaire de jk, coûte de l'espace de stockage.
Avec une base de données orientée colonnes, les colonnes sont spécifiées pour chaque clé (ou index), et pour chaque index le nombre de colonnes peut-être variable et n'est pas limité. Une colonne peut être ajoutée à la volée à volonté. Un NULL ne coûte donc pas d'espace de stockage. Fondamentalement, une colonne est un tuple, c'est-à-dire une paire nom de colonne / valeur. Dans le cas de Cassandra, il s'agit en fait d'un triplet en raison de l'ajout du timbre à date. Voici comment la table ci-dessus peut trouver son équivalent avec Cassandra, et avec deux colonnes seulement pour l'index jk :

fred	(Nom : Mazué) (Prénom : Frédéric) (Salaire : 1)
ft	(Nom : Tonic) (Prénom : François) (Salaire : 1000)
jk	(Nom : Kaminksy) (Prénom : Jean) (Fonction : Big Boss)

Sous Cassandra, il existe des espaces de clés (ou Keyspace) dans lesquels des familles de colonnes sont définis: nom, encodage cache, etc. Cette définition se situe elle aussi dans le fichier storage-conf.xml. Par défaut celui-ci définit un espace de clé du nom de Keyspace1, et des familles de colonne Standard1, Standard2, etc. Nous comprenons donc que quand nous avons donné dans le client :

```
Keyspace1.Standard2['fred']['Nom'] = 'Mazue'
```

La clé fred s'est trouvée rangée dans l'espace Keyspace1 et la colonne 'Nom' est de famille Standard2, c'est-à-dire encodée en utf-8 d'après storage-conf.xml. L'administrateur de Cassandra a toute liberté pour redéfinir les Keyspace et les familles de colonnes. Enfin, il est important de savoir que les index sont toujours triés, ce qui permet d'interroger sur des intervalles (slice) d'enregistrement). Ce tri se fait à chaque insertion.

5 ECRIRE UN CLIENT EN JAVA

Ce travail est ardu, en raison de la carence de documentation. Cassandra est conçue pour communiquer à travers le protocole thrift, qui sort du cadre de cet article. Le lecteur trouvera sur notre site un exemple, demothrift.Main, qui utilise les classes (non documentées) de Cassandra. Nous ne reproduisons pas cet exemple ici, car il dérive directement du seul exemple fourni sur le site de Cassandra. Plutôt que les classes de Cassandra, je pense qu'il est mieux d'utiliser l'API Hector (http://prettyprint.me) écrite par Ran Tavory, qui semble être un des membres du projet Cassandra. Cette API est de haut niveau, mais elle n'est pas documentée elle non plus… La seule ressource disponible pour y comprendre quelque chose est un code de test de l'API fourni sur le site précité. Le code ci-dessous a été écrit de cette manière :

```
package me.prettyprint.cassandra.service;

import static me.prettyprint.cassandra.utils.StringUtils.bytes;
import static me.prettyprint.cassandra.utils.StringUtils.string;


import static org.mockito.Mockito.mock;
import java.net.UnknownHostException;
```

```
import java.util.NoSuchElementException;
import org.apache.cassandra.thrift.Column;
import org.apache.cassandra.thrift.ColumnPath;
import org.apache.cassandra.thrift.ConsistencyLevel;
import org.apache.cassandra.thrift.NotFoundException;
import org.apache.cassandra.thrift.SuperColumn;
import org.apache.thrift.TException;
import org.apache.thrift.transport.TTransportException;

public class DemoHector {

 private CassandraClient client;
 private Keyspace keyspace;
 private CassandraClientPool pools;
 private CassandraClientMonitor monitor;

 public void setup() throws TTransportException, TException,
   IllegalArgumentException, NotFoundException,
   UnknownHostException {

   pools = mock(CassandraClientPool.class);
   monitor = mock(CassandraClientMonitor.class);
   client = new CassandraClientFactory(pools,
   new CassandraHost("localhost", 9160), monitor).create();
   keyspace = client.getKeyspace("Keyspace1",
     ConsistencyLevel.ONE,
     CassandraClient.DEFAULT_FAILOVER_POLICY);
 }


 public void demoColonne() throws IllegalArgumentException,
   NoSuchElementException, IllegalStateException,
   NotFoundException, Exception {

   // Insérer les données
   ColumnPath cp = new ColumnPath("Standard1");
   cp.setColumn(bytes("Prenom"));
   keyspace.insert("fred", cp, bytes("Frederic"));
   cp.setColumn(bytes("Nom"));
   keyspace.insert("fred", cp, bytes("Mazue"));

   // lire les données
   // D'abord le prénom
   cp.setColumn(bytes("Prenom"));
   Column col = keyspace.getColumn("fred", cp);
   String value = string(col.getValue());
   System.out.println(value);
   // Ensuite le nom
   cp.setColumn(bytes("Nom"));
   col = keyspace.getColumn("fred", cp);
   value = string(col.getValue());
   System.out.println(value);
 }

 public void demoSuperColonne() throws IllegalArgumentException,
   NoSuchElementException, IllegalStateException,
   NotFoundException, Exception {

   // Insérer les données
   ColumnPath cp = new ColumnPath("Super1");
   cp.setColumn(bytes("Prenom"));
   cp.setSuper_column(bytes("Auteur"));
   keyspace.insert("superfred", cp, bytes("Frederic"));
   cp.setColumn(bytes("Nom"));
   keyspace.insert("superfred", cp, bytes("Mazue"));
   cp.setSuper_column(bytes("Article"));
   cp.setColumn(bytes("Titre"));
   keyspace.insert("superfred", cp, bytes("Decouvrez Cassandra"));

   // Lire les données
   // D'abord celles relatives à l'auteur
   cp = new ColumnPath("Super1");
   cp.setSuper_column(bytes("Auteur"));
   SuperColumn sc = keyspace.getSuperColumn("superfred", cp);
   int n = sc.getColumns().size();
   System.out.println("Il y a " + n
     + " colonnes dans la colonne Auteur");
   Column col;
   String name;
   String value;
   for(int i=0; i<n; i++)
   {
     col = sc.getColumns().get(i);
     name = string(col.getName());
     value = string(col.getValue());
     System.out.println(name + " " + value);
   }
   // puis les données relatives à l'article
   cp.setSuper_column(bytes("Article"));
   sc = keyspace.getSuperColumn("superfred", cp);
   System.out.println("Il y a " + sc.getColumns().size()
     + " colonne dans la colonne Article");
   col = sc.getColumns().get(0);
   value = string(col.getValue());
   System.out.println("Titre article: " + value);
 }

 public static void main(String[] args) throws NotFoundException,
   Exception {
   DemoHector dh = new DemoHector();
   dh.setup();
   dh.demoColonne();
   dh.demoSuperColonne()
 }
}
```

Ce code illustre en outre le travail avec les super colonnes de Cassandra, qui sont des conteneurs de colonnes. Tout n'est pas dit sur Cassandra, nous invitons le lecteur qui voudrait approfondir à se replonger dans storage-conf pour définir mot de passe et stratégies fines de réplications, et côté client, sur les prédicats qui permettent d'obtenir des intervalles de colonnes.

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

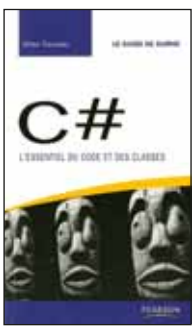
par François Tonic

LIVRE DU MOIS

Les EJB 3

*Difficulté : **** - Editeur : éditions Eni*
Auteur : Celinio Fernandes - Prix : 39 €

Pas toujours très bien considérés, les EJB apparaissent comme lourds et complexes. Ce livre aborde les EJB 3 via l'utilisation de frameworks (Struts, JSF, Flex). Et l'explication se fait par la conception d'une application afin de comprendre les différentes couches et la mise en œuvre concrète des EJB. L'auteur aborde chacun des différents frameworks pour bien marquer les différences et les points communs. Bien entendu, les bases des EJB 3 sont présentées. On y trouvera aussi des informations sur les EJB 3.1, et les nouveaux apports de Java EE 6. La partie Flex est intéressante, particulièrement ce qui concerne l'usage du framework GraniteDS et comment les EJB contribuent aux interfaces riches RIA !



LANGAGE

Le guide de survie C#

*Difficulté : ****
Editeur : Pearson
Auteur : Gilles Tourreau
Prix : 19,00 €

Vous avez une panne ? Vous avez oublié une classe ? Ce guide de survie est fait pour vous. Il permettra aux développeurs déjà familiers de l'algorithmique ou de la programmation orientée objet de s'initier rapidement aux technologies du .NET Framework. Une centaine d'exemples de code est donnée pour améliorer la compréhension. Ce guide couvre les versions 2, 3, 3.5, 4.

SÉCURITÉ

Gestion des risques en sécurité de l'information

*Difficulté : ****
Editeur : Eyrolles
Auteur : Anne Lupfer
Prix : 39,90 €

Comment évaluer le risque pour les SI d'entreprise ? Quels risques doit-on accepter de prendre ? La gestion des risques en sécurité de l'information, recommandée par de nombreux référentiels comme la norme ISO 27001, est en train de devenir une obligation pour les responsables de la sécurité de l'information (RSSI), les directeurs des systèmes d'information (DSI), et bien d'autres acteurs de l'entreprise. Après les démarches locales, comme Ebios et Mehari en France, la norme ISO 27005, première méthode de gestion des risques structurée et normalisée, est appelée à s'imposer à l'échelle planétaire. Facilement accessible, elle propose une approche continue (au quotidien, dans la durée), systématique, pragmatique et adaptée à la réalité complexe des entreprises actuelles. S'appuyant, tout comme la norme, sur des scénarios d'incidents réels, ce guide de mise en oeuvre ISO 27005 dévoile l'essence des années d'expérience et de savoir-faire de l'auteur, et constituera une aide précieuse pour la certification ISO 27005 Risk Manager.

SÉCURITÉ

la boîte à outils du pilote des systèmes d'information

*Difficulté : ****
Editeur : Dunod
Auteur : J-L Foucard
Prix : 25,00 €

Il paraît difficile de concilier la continuité du système d'information et la création de valeur dans ce même SI. L'un des problèmes d'évolution vient de l'incertitude technologique : sa vitesse d'évolution, sa profusion. Cela nécessite de la réflexion sans pour autant pénaliser son SI. Au-delà de la technique, nous retrouvons les problèmes de communication entre les personnes et les départements de l'entreprise. Se pose aussi la question du choix des outils, des méthodes et des prestataires externes. Découvrez 74 outils et méthodes pour piloter efficacement la construction des systèmes d'information et satisfaire les exigences des clients par un schéma synthétique, le contexte d'utilisation, les étapes de mise en oeuvre, des conseils, les avantages et précautions à prendre. Par cette boîte à outils, l'auteur veut vous aider à comprendre le pilotage et le faire de manière raisonnée.

BLOG

Wordpress 3

*Difficulté : ***
Editeur : Pearson
Auteur : divers
Prix : 33,00 €

Wordpress constitue une des meilleures plateformes de blog et site web actuellement aussi bien pour un usage personnel que professionnel. Grâce à une communauté active et à de nombreuses extensions, Wordpress peut réellement tout faire ! Et il est gratuit et open source. Le must du CMS ! Ce livre, écrit par trois personnalités de la communauté francophone et internationale de WordPress, est un véritable guide de référence.
Il vous fera entrer de plain-pied dans la communauté des utilisateurs aguerris en vous livrant l'ensemble des informations utiles pour comprendre le logiciel et en exploiter tout le potentiel. Cette édition aborde longuement la version 3. Bonus : la présence d'un CD-Rom.

WEB

HTML5 pour les web designers

*Difficulté : ****
Editeur : Eyrolles
Auteur : Jereny Keith
Prix : 12,00 €

Evolution pour les uns, révolution pour les autres, HTML 5 anime les débats depuis plus d'un an alors que les spécifications ne sont même pas fixées et qu'une première spécification test ne sera pas disponible avant fin 2010 (au moins) pour une version finale pas avant fin 2011.
Mais déjà, des balises HTML5 apparaissent un peu partout notamment pour les animations, la vidéo. Que doivent en retenir les web designers et les développeurs ? Comment exploiter toute la puissance de l'HTML5 dans les navigateurs actuels ? L'auteur tente de répondre aux questions des développeurs et designers.

Kevin, responsable du pôle **formation d'ISIMEDIA :**

"La nature ne m'a pas fait blonde à forte poitrine et je n'ai pas eu la chance de grandir sous le soleil d'Hawaï. Mais je développe tous les jours avec WINDEV et WEBDEV, et j'aime communiquer mon savoir-faire"

ISIMEDIA. Le spécialiste WINDEV / WEBDEV & WINDEV Mobile

Formations WINDEV / WEBDEV

Paris • Lyon • Montpellier • Nantes

Savez-vous qu'il est désormais possible de vous former à WINDEV et WEBDEV avec des professionnels du développement ?

Nos formations sont réalisées par **des collaborateurs d'ISIMEDIA** : ce sont avant tout des professionnels du développement qui vous feront bénéficier d'**un vrai retour d'expérience**. Leur expérience « terrain » est déterminante : WINDEV et WEBDEV sont en effet des outils riches qui disposent de nombreuses fonctionnalités destinées à raccourcir les temps de développement. Or, l'expérience montre que certaines de ces fonctionnalités nécessitent d'être employées avec précaution dès lors que l'on souhaite réaliser une application d'envergure, performante et maintenable. Nos formateurs sont donc des interlocuteurs privilégiés qui sauront mieux que des formateurs professionnels, vous conseiller pour réaliser des applications de qualité.

Formations inter-entreprises (cours standards)

INTITULÉ	DURÉE	TARIF HT
Prise en main	2 jours	890 € HT
Perfectionnement	3 jours	1 350 € HT
Prise en main + Perfectionnement	5 jours	1 990 € HT
Expert	3 jours	1 350 € HT

Plans de cours et calendriers disponibles sur notre site Web www.isimedia.com

INTITULÉ	DURÉE	TARIF HT
Prise en main	2 jours	890 € HT
Perfectionnement	3 jours	1 350 € HT
Prise en main + Perfectionnement	5 jours	1 990 € HT

Formations intra-entreprises

Le contenu est entièrement personnalisable : du plan de cours standard au transfert de compétences préalable à la réalisation d'un nouveau projet, nous pouvons réaliser des zooms approfondis sur certaines fonctionnalités de WINDEV et WEBDEV. Ces formations, dont la durée varie entre 2 et 5 jours, font l'objet d'un devis spécifique.

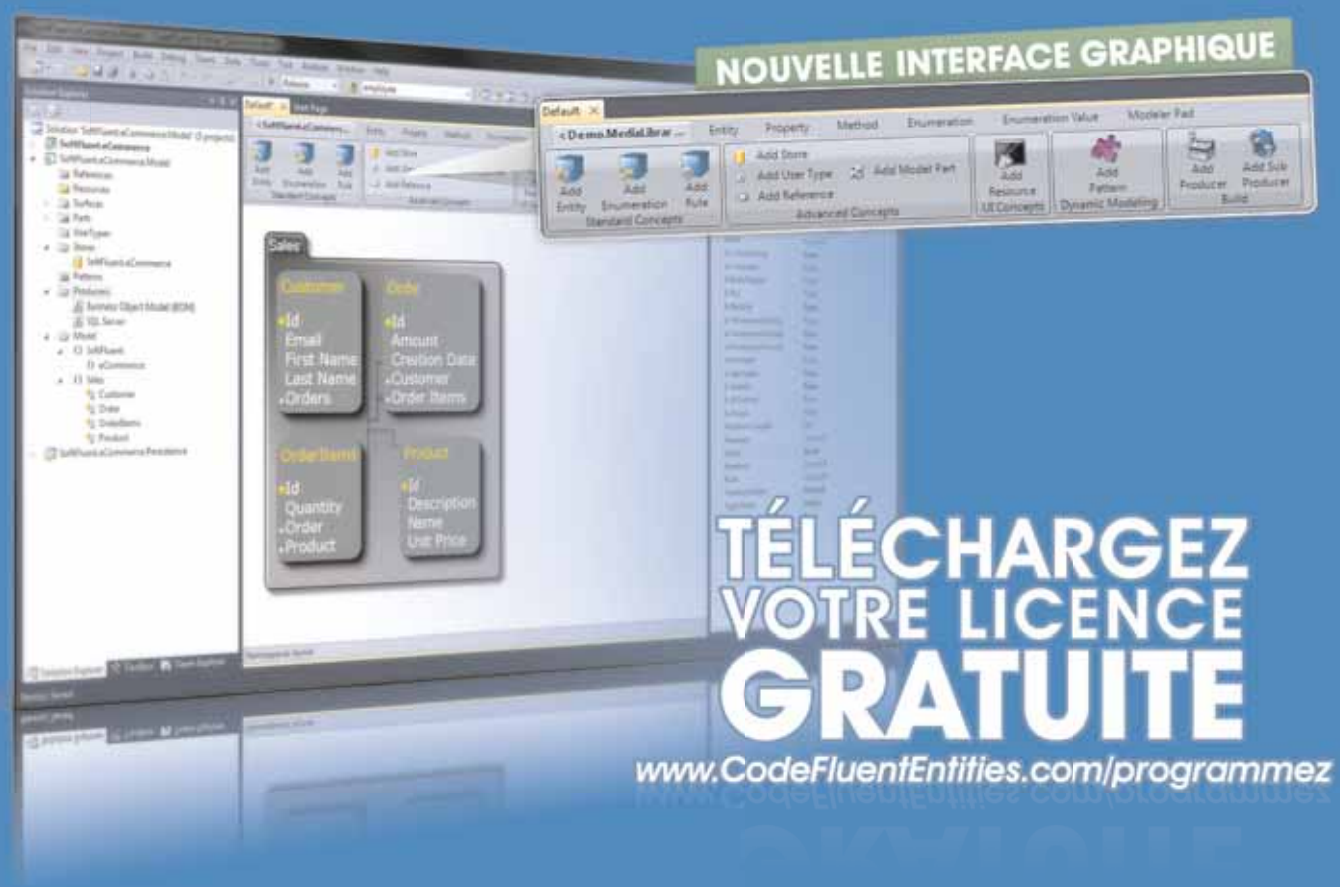
Les + d'une formation inter-entreprise dispensée par ISIMEDIA :

1. Une formation réalisée par un professionnel du développement WINDEV / WEBDEV
2. Une totale indépendance vis-à-vis de l'éditeur PC SOFT
3. Une machine par participant (équipée d'une clé WINDEV ou WEBDEV)
4. Une mise en application de chacun des thèmes abordés sur un projet concret
5. Les supports de cours remis sur clé USB
6. Des sessions organisées sur Paris, Lyon, Montpellier et Nantes
7. Des sessions limitées à 8 participants

Besoin d'informations?

Contactez-nous directement : Tél : 04 67 55 81 55 • E-mail : formation@isimedia.com

Simplifiez-vous la vie avec CODEFLUENT ENTITIES !



ELÉMENTS APPLICATIFS

- *Localisation*
- *Liaison aux données*
- *Règles & Validation*
- *Concurrence d'accès*
- *Sécurité*
- *Cache*
- *Gestion des BLOBS*

ARCHITECTURES

- *SOA, Client intelligent*
- *Client riche, RIA*
- *Web, Webparts*
- *Client/Serveur, N-Tier*
- *Office*
- *SharePoint*
- *SaaS, Cloud*

TECHNOLOGIES

- *.NET, C#, Linq, Visual Studio*
- *ASP .NET WebForms, MVC*
- *Silverlight, WCF, ASMX*
- *WPF, WinForms*
- *Microsoft SQL Server*
- *Oracle Database*
- *Excel, Access, SharePoint*

LE 1er ENVIRONNEMENT POUR LES DÉVELOPPEURS, PAR LES DÉVELOPPEURS

CodeFluent Entities offre aux développeurs une ***méthode structurée*** et un ***environnement intégré*** à Visual Studio pour faciliter et accélérer le développement des applications .NET.

CodeFluent Entities
the Model Driven Software factOry

Contact: info@softfluent.com
www.CodeFluentEntities.com
SoftFluent *3 rue de la Renaissance 92160 ANTONY*