mensuel n°113 - novembre 2008

PROgrammez !

Nouvelle Formule

www. programmez.com

Le magazine du développement

Google™ révolutionne le développement

- Android : vos applications mobiles
- Exploitez GWT
- Gears - App Engine : le Cloud Computing arrive
- Protocol Buffers : le tueur de XML

Salaires
Les compétences gagnantes

Développeur, boostez votre PRODUCTIVITÉ !

Silverlight 2.0 adopte Eclipse

.net
Concevez vos cartes avec Virtual Earth

Java
Des applications natives Java avec JNA !

Multiprocesseur
Utilisez MPI en Python

SGBD
SQL Server 2008 Les mécanismes de compression

Web
Coder avec le framework Play!

M 04319 - 113 - 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

Nouvelle version 14

Venez découvrir WINDEV 14

Vous êtes invité ! (11 villes)

WINDEV

Environnement professionnel intégré de développement (IDE & ALM). Windows, Internet, Mobile.

501 NOUVEAUTÉS

Inscrivez-vous vite pour découvrir WINDEV 14 et vous aussi *développez 10 fois plus vite* !

Montpellier	jeudi 13 novembre
Nantes	mardi 18 novembre
Bordeaux	mercredi 19 novembre
Toulouse	jeudi 20 novembre
Bruxelles	mardi 25 novembre
Lille	mercredi 26 novembre
Paris	jeudi 27 novembre
Genève	mardi 2 décembre
Lyon	mercredi 3 décembre
Strasbourg	jeudi 4 décembre
Marseille	mardi 9 décembre

de 13h45 à 17h30

Attention: 10.000 places seulement

Détail du programme et invitation sur www.pcsoft.fr

www.pcsoft.fr

Logiciel professionnel. Version annoncée. Document non contractuel. Support technique gratuit. ...ta version en cours de commercialisation.

PROgrammez ! le magazine du développement
numéro 113 - novembre 2008

sommaire

CD ROM

.NET

Silverlight 2.0
Installez et développez dès aujourd'hui avec la version finale du très attendu Silverlight 2. La plate-forme RIA incontournable ! Windows & MacOS X

Eclipse4sl
Utilisez Eclipse pour créer vos applications Silverlight 2.0 ! Pré-version technique - Windows

Mono 2.0
Découvrez la toute dernière version du Framework .Net open source ! Supporte entièrement les API 2.0.

MonoDevelop 2.0 alpha1
L'IDE dernière génération pour développer rapidement en Mono 2.0 !

Training kit pour Visual Studio 2008 et .Net 3.5
Apprenez simplement et rapidement les API, les nouvelles fonctions de la plate-forme .Net.

PHP

Symfony 1.1.4
La dernière version du framework vedette PHP

Web

Alfresco Labs 3b
Créez votre gestion de contenu compatible SharePoint avec l'environnement référence : Alfresco. Windows – Version communautaire

Seamonkey 2.0 alpha
La nouvelle suite web intégrée. Inclut la messagerie, agenda, navigateur, chat… Versions Windows et Linux

Outils

Java 7 JDK
Le futur de Java disponible dès maintenant. Découvrez la dernière pré-version.

Python 2.6
La nouvelle version du langage Python

Realbasic 2008 r4
Développez et déployez rapidement vos applications Basic sur toutes les plates-formes. Version Linux 30 jours.

Conversion de données par glisser-déposer

ALTOVA® mapforce® 2008

Découvrez Altova MapForce® 2008, le célèbre outil de mappage de données conçu par les créateurs de XMLSpy®. Utilisez le glisser-déposer pour mapper, convertir et transformer vos données entre:

XML | Databases | EDI | Flat Files | Web Services

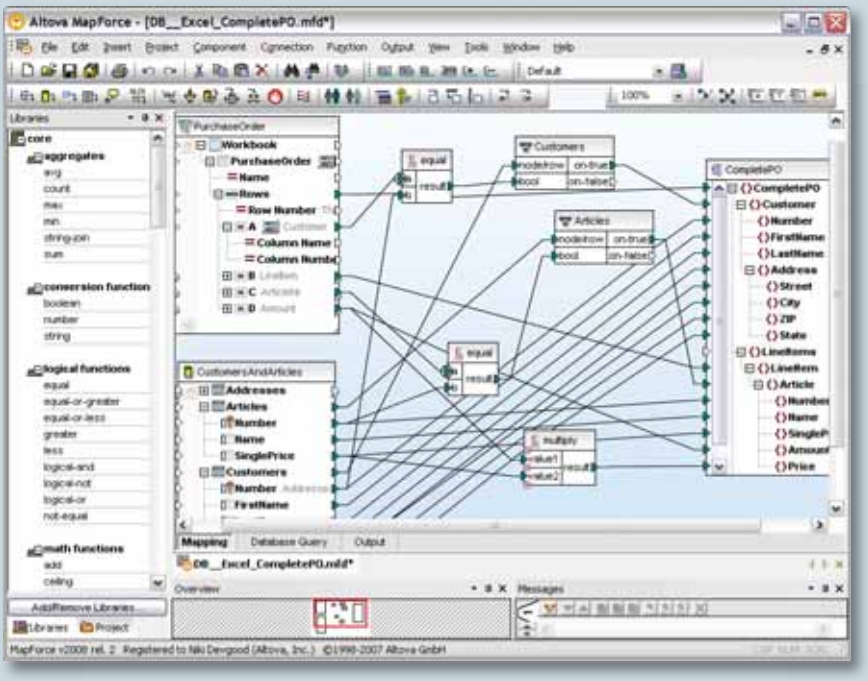
- **Mappage de données Excel 2007 (OOXML)**
- **Prise en charge de flux de données en code généré**
- **Prise en charge d'instructions SQL SELECT**
- **Création en série de clés primaires / étrangères**
- **Prise en charge de SOAP 1.2**
- **Et bien plus encore**

Nouveautés dans la Version 2008 v2

Une fois que vous avez défini un mappage de données dans MapForce, il vous suffit de cliquer sur la fenêtre de sortie pour instantanément convertir les données. Vous pouvez également générer du code libre de droits d'utilisation et le déployer sans frais supplémentaires ou adaptateurs de déploiement.

Grâce à MapForce, implémentez l'intégration de vos données et vos applications de type services Web **sans avoir à écrire de code!**

- Mappage et conversion de données par glisser-déposer
- Prise en charge des principales bases de données relationnelles
- Fenêtre de requêtes sur bases de données avec éditeur SQL
- Utilitaire FlexText pour l'analyse des fichiers plats
- Prise en charge des messages EDIFACT et X12 EDI
- **Nouveau!** Mappage de fichiers OOXML / Excel 2007 (.xlsx)
- Mappage de fichiers OOXML / Excel 2007 (.xlsx)



ALTOVA®

Jugez MapForce par vous-même – Téléchargez gratuitement une version d'évaluation de 30 jours sur www.altova.com

éditoll

On éclipse tout

Octobre serait-il le mois d'Eclipse ? On pourrait le croire suite à l'avalanche des annonces, des mouvements. Et surtout, indirectement, Microsoft qui fait du pied au monde d'en face, dans l'espoir de proposer une nouvelle maison à son Silverlight. Nous l'avions dit en filigrane dans le précédent numéro avec l'opposition, toujours légitime, entre Eclipse et NetBeans, Eclipse est plus que jamais la plate-forme de développement par excellence open source, et Java. L'arrivée de Microsoft ne fait que conforter cette impression.

Au-delà de l'annonce, que nous développons dans ce numéro, il faut bien comprendre qu'il s'agit d'une réelle révolution dans l'approche de l'éditeur pour démocratiser ces technologies. Car un des défauts de Silverlight résidait dans son approche mono-IDE. Et comme de nombreux développeurs travaillent sous Eclipse, il fallait trouver une solution. Microsoft avec eclipse4sl pousse en fait deux pions : Silverlight sur d'autres outils que les siens et Java. Car finalement, on oublie de dire que ce projet (open source) vise aussi à assurer une intéropérabilité entre Silverlight et Java...

Mais Eclipse confirme chaque jour son omnipotence dans l'univers Java. Ce n'est pas IBM Rational qui dira le contraire en dévoilant sa plate-forme collaborative Jazz, via Team Concert, et en sortant les produits sous forme de plug-in. Le futur sera-t-il de posséder une unique plate-forme, un socle, sur lequel on ajoutera des modules, des composants ? Oui la tendance est bien réelle et cela fournit une grande souplesse. Et d'autre part, on tombe dans la notion d'usine à logiciels. Eclipse sera la chaîne de montage sur laquelle on ajoute des robots et ces robots intègrent, assemblent.

En parlant d'usines à logiciels, n'oublions pas un autre aspect particulièrement important et qui n'est pas à prendre à la légère, surtout en période de crise, la génération automatique. Aujourd'hui, cette génération vise à générer à partir des modèles (UML) une application complète. Et d'ores et déjà, des solutions existent et fonctionnent, il n'y a qu'à voir l'efficacité d'un environnement comme Blu Age de Netfective. Mais finalement, cette tendance de fond se vérifie aussi dans les données, l'architecture, le développement au quotidien. Et nous pensons que l'arrivée de Microsoft dans le monde OMG et l'annonce d'une prise en compte de plusieurs diagrammes UML vont permettre à ce marché d'exploser. Le développeur n'aura plus d'excuse pour avancer à reculons.

D'un côté, nous croulons sous les outils, comme nous venons de le voir, et de l'autre, nous faisons face à un vide, presque une reculade. Google sait faire des API, des technologies mais on peut se demander où sont les outils pour coder en toute souplesse, en toute productivité. Eclipse (encore lui) est bien utilisé par Android mais pour Gears, pour App Engine ? La puissance 10 de l'éditeur se vérifie au quotidien avec les librairies mais le manque d'outils, les absences fonctionnelles gênent les développeurs et les changements continuels dans les API (bêta certes) comme Gears ou App Engine freinent l'enthousiasme légitime qu'elles engendrent. Regardez Chrome, une euphorie durant les premières semaines et le calme plat rapidement après. Mais c'est bien normal pour une application en plein développement. Et si finalement, Google se servait de Chrome pour en faire son centre de développement ? Pourquoi pas !

Allez, promis, le mois prochain, ce sera le mois .Net !

■ François **Tonic**

PROgrammez! Le magazine du développement

Rédaction : redaction@programmez.com
Directeur de la Rédaction : Jean Kaminsky
Rédacteur en Chef : François Tonic
Ont collaboré : F. Mazué, L. Guillois, F. Dewasmes, C.Remy, O. Thery.
Experts : P. Belaud, D. Mera, F. Barbier, P. Mage, V. Bostoen, A. Dupuis, R. Sabin Mompelat, P. Lonvaud, S. Saurel, F. Colin , G. André, D. Cohen-Zardi
Crédit photo : istockphoto, Crabmet
Maquette : AJE Conseils
Publicité : Régie publicitaire, K-Now sarl
Pour la publicité uniquement : Tél. : 01 41 77 16 03
coordination@programmez.com
Editeur : Go-02 sarl, 6 rue Bezout - 75014 Paris
Coordination@programmez.com - Dépôt légal : à parution - Commission paritaire : 0707K78366
ISSN : 1627-0908 - Imprimeur : ETC - 76198 Yvetot
Directeur de la publication : J-C Vaudecrane
Ce numéro comporte 1 CD Rom

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

L'INFO PERMANENTE
WWW.PROGRAMMEZ.COM

PROCHAIN NUMÉRO

N°114 décembre, parution 29 novembre

- **Dossier PHP**
  Les formations,
  Gérer la montée en charge,
  Open Office et PHP.
- **Les révolutions Microsoft**
  Présentation et analyses des annonces faites à la PDC et au Tech Ed !

Votre potentiel, notre passion.™

Microsoft®

■ **Ilog** annonce la nouvelle version des composants Java, JViews. Cette version, la 8.5, s'intègre avec Eclipse et notamment avec des modules tels que EMF et GEF. La partie Ajax connaît elle aussi un lifting important avec un meilleur support des frameworks basés JSF. Facelets fait aussi son apparition. Disponibilité : décembre.

■ **ASP.Net MVC :** Microsoft vient de livrer la première version bêta du modèle MVC pour ASP.Net 3.5. Il doit aussi faciliter le développement selon le modèle TDD, test driven development.

■ **Qt Software** lance un grand concours de créativité de widgets, pimp my widgets. La date limite de soumission du projet est fixée au 31 décembre 2008. De nombreux lots à gagner !

■ **MiniUML 1.0** est un projet sur CodePlex. Il s'agit d'une application WPF pour créer des diagrammes UML. Le projet est en cours de développement et reste pour le moment très limité dans son support. Site : http://www.codeplex.com/miniuml

■ **DemoCamp Eclipse** fait une halte à Paris le 20 novembre de 18 à 22h. Mission : montrer ses développements, partager ses idées, ses travaux ! Un rendez-vous incontournable ! site http://wiki.eclipse.org/Eclipse_DemoCamps_November_2008/Paris

■ **Aladdin** a créé un nouveau centre sur la sécurité sur Internet pour lutter contre la cybercriminalité : Attack Intelligence Research Center. Il emploie des chercheurs en sécurité et des spécialistes en matière de cybercriminalité et d'application des réglementations. Site : www.aladdin.com/AIRC

■ **Microsoft Web Platform Installer** est disponible en bêta. Il s'agit de pouvoir déployer dans un package unique (avec sa configuration) la plate-forme Web Microsoft incluant IIS7, Visual Web Developer 2008 Express, SQL Server 2008 Express, .Net Framework !
site : http://www.microsoft.com/web/channel/products/Web-PlatformInstaller.aspx

■ **Microsoft Web Application Installer** est disponible en bêta. Cet outil doit vous aider à utiliser les applications web sur un serveur Windows. L'outil supporte ASP.Net, PHP, DotNetNuke, WordPress, Drupal. Voici une excellente initiative ! Compatible avec MySQL. site : http://www.codeplex.com/wai

■ **JAX-RS** est disponible en version finale. Pour mémoire, Jax-RS est une API pour RESTful (web services). Elle est connue sous le nom de : JSR 311. Le projet Jersey (lui aussi disponible) est l'implémentation Sun de la JSR 311.

■ **NetBeans 6.5** sera disponible dans à peine 3 semaines ! Au menu : nombreuses améliorations et nouveautés sur PHP, Ajax, JavaScript, JavaEE, Groovy, Grails, Rails, Glassfish 3, et le support attendu de JavaFX.

Atelier

PC Soft livre la 14e version de WinDev

La fin d'année approche et PC Soft, éditeur français bien connu, prépare les fêtes et le début 2009 avec la 14e version de son atelier de développement complet : WinDev. Disponible depuis la mi-octobre dernier, WinDev 14, comme dans les précédentes moutures, propose 501 améliorations et nouveautés, et ce, dans tous les compartiments et modules. Ces nouveautés concernent les éditions WinDev, WebDev et WebDev Mobile. Commençons par les graphiques qui subissent un lifting avec de nouveaux modèles comme le radar, les aires, le beignet. On bénéficie dessus d'un zoom automatique. WinDev 14 profite aussi des buzzworlds actuels pour se mettre à niveau. Ainsi, le mashup fait une entrée en force. Le SaaS fait lui aussi son apparition, avec la possibilité de connaître son mode de distribution (de l'application). La SOA est également présente, avec de nouvelles fonctions dans le langage propre à WinDev. Le confort du développeur n'a pas été oublié avec une refonte de l'interface et une meilleure adaptabilité aux grands écrans. On bénéficie d'un nouvel explorateur de projet, d'une recherche multiprojet ! Et surtout d'une centralisation des options et paramètres de l'environnement, ce qui n'est pas un luxe pour le développeur, pour gagner du temps ! Toujours dans les performances, le compilateur subit une optimisation pour mieux tirer parti des processeurs multi-coeurs. Et on peut maintenant automatiser le build en le configurant à une heure donnée, idéal pour faire des builds chaque nuit... On bénéficie aussi d'un lien natif avec Google, les applications WinDev peuvent maintenant s'interfacer facilement et rapidement avec les services en ligne Google. Pour mieux contrôler son application, on dispose d'un robot de surveillance et d'un moniteur. On dispose aussi d'un support de PostgreSQL, des champs Flex et Silverlight 2, du support PHP 4 et 5. La mobilité a été améliorée avec notamment le mode VGA. Une version express est aussi disponible gratuitement sur le site.
Le Tour de France aura lieu du 13 novembre au 9 décembre.
Site : http://www.pcsoft.fr/

Rapport

France numérique 2012 : quoi de neuf ?

Présenté avec plus d'une semaine de retard, le plan de développement de l'économie numérique est un gros rapport de 80 pages contenant 154 mesures dans tous les domaines touchant à l'informatique, au numérique. Si on peut saluer la généralisation du haut débit fixe et mobile, la partie développement / logiciel est plus décevante. Le secteur du jeu vidéo est une nouvelle fois mis en avant. Secteur important mais qui concerne quelques milliers de personnes alors que le développement informatique représente des dizaines de milliers d'emplois ! Sur cette partie, le plan préconise de renforcer les aides, les conseils à l'exportation qui est une des sources de profits des éditeurs français dont le paysage est très segmenté et verticalisé. Le plan repère par contre le dynamisme européen autour de l'open source dont la France est un des utilisateurs les plus assidus et contribue énormément aux projets ouverts. Ce domaine est à soutenir, à développer encore plus si on veut garder notre place. Notons que les standards et l'interopérabilité sont préconisés dans la protection des contenus mais aussi au niveau documentaire. Nous n'avons pas vu d'actions spécifiques pour améliorer le secteur du développement informatique à part l'utilisation plus importante de logiciels ouverts sans que cela donne lieu à mesures spécifiques (partie sur le développement logiciel). A la place, nous avons une action 64 sur l'affichage des prix séparés des logiciels et des systèmes d'exploitation pré-installés.

Et une action 65 qui préconise le découplage de la vente de l'ordinateur et du système d'exploitation pré-installé (cela vise clairement Windows qui domine le marché). Il y a bien la création d'un réseau " Logiciel " (action 63) dans une dizaine de villes clés de l'industrie logicielle mais est-ce suffisant ? Quelle sera sa mission ? On pousse surtout l'administration.

Notons aussi de nombreuses mesures pour la sécurité des données et la lutte contre la cybercriminalité.

On notera aussi les encouragements à améliorer le numérique et son usage dans les TPE et PME ou encore les efforts pour développer le télétravail en France.

Modélisation

Objecteering Software dévoile Document Publisher

L'éditeur annonce avec la sortie du service pack 2 de son atelier de modélisation (version 6.1), la disponibilité d'un nouvel outil orienté production documentaire : Document Publisher. Car le projet dépend aussi de la qualité de la documentation (contenu, rédaction, présentation, etc.), l'outil aide à mieux formaliser cette documentation en guidant les développeurs, utilisateurs. *" Convivialité, qualité et couverture du cycle de vie sont les points clés de Objecteering Document Publisher "* déclare Philippe Desfray Directeur R&D chez Objecteering Software. Disponible pour Windows et Linux.

■ **IBM** a dévoilé un nouvel outil pour traquer les failles de sécurités dès le développement des applications ! IBM Rational AppScan Developer Edition peut être utilisé dès la conception du projet pour détecter rapidement les risques, les failles dans le code. Tarif : 2 650 dollars par poste.

■ **Intersystem** a sorti un nouvel outil DeepSee. Il s'agit de disposer d'un moteur de BI dans son application. Fonctionnant de concert avec Caché, la base de données maison, et ayant un modèle objet de bout en bout, DeepSee doit simplifier la création des tableaux de bord, rapports, etc.

■ **Ingres l'éditeur de SGBD** a annoncé fin septembre dernier qu'il rejoignait le consortium OW2 (ex-ObjectWeb). Il vise à contribuer à la mise en place d'une initiative BI dans le consortium. Annonce importance car la BI bouge beaucoup et les gros éditeurs de SGBD se positionnent rapidement sur ce marché.

■ **Ever Team** vient de lancer une solution de gestion de contenus totalement compatible SharePoint : EverSuite.Net.

■ **Netqost** innove en lançant une solution de test de charge des applications web en mode SaaS ! On dispose alors d'un outil d'analyse générant les rapports sur les temps de réponses, la charge des sessions, les performances, etc. www.netqost.net

■ **La Fondation Linux** a dévoilé la bêta des Linux Standard Base 4.0. Cette version comprend le nouvel outil de vérification App Checker ainsi qu'un outil de vérification des scripts shell et surtout un tout nouveau SDK qui doit aider le développeur à utiliser les LSB. D'autre part le SDK supporte les versions antérieures. On y trouve aussi : Mozilla Network Security, SSL, la partie Sample Implementation a été entièrement refaite. Sortie prévue : courant automne !

Conférence

Rational mise tout sur le collaboratif

C'est le 14 octobre dernier que s'est tenue l'édition parisienne de la RSDC (Rational Software Developer Conference). Sur une journée, il s'agissait de montrer les nouveautés et les prochaines évolutions des plates-formes Rational. L'après midi fut surtout consacrée aux sessions techniques, Rational et de partenaires.

La stratégie de Rational est toujours d'adresser les besoins de l'entreprise, en prenant en compte les usages internes et externes. Comme l'a précisé Martin Nally (Chief Technology Officer IBM Rational et "Distinguished Engineer"), les entreprises ont besoin d'outils distribués notamment dans des projets d'outsourcing. Il faut aussi mettre en place des méthodes agiles. La stratégie dévoilée cet été avec Jazz pour la collaboration et les premiers outils Rational comme Team Concert doit répondre à ce défi. *" On s'occupe des outils. Il faut s'occuper des données, des tests, des process. "* affirme Martin Nally. Car dans tout projet, la donnée est l'élément vital. Il faut pouvoir la créer, la modifier, la communiquer aisément, même si l'outil (de développement) reste important. Il faut aller au-delà et c'est le chemin pris par Rational, indique Martin.

La plate-forme Jazz va donc dans ce sens. Et cela va aussi dans la mise en place des bonnes pratiques. Mais cela démontre aussi toute l'importance du rôle de l'intégration. *" On développe le logiciel comme on le faisait il y a 50 ans. "* précise Martin Nally. Aujourd'hui il faut donc aller bien au-delà dans les process, les méthodologies, les données, l'architecture, bref tout ce qui aide à l'industrialisation du développement constitue un gage crucial. D'ailleurs, l'éditeur Nefective qui avait un stand et faisait une conférence, formulait la même remarque avec son outil de génération : Blu Age.

Une partie de l'état major de Rational : de g-à-d, Olivier Roubine, pourl'Europe, Martin Nally, CTO, Patrice Poiraud, directeur de la division Rational France, Michel Speranski, chef de produit.

Conserver les compétences en interne

Un des maîtres mots de la journée, Team Concert oblige, fut collaboration et travail en équipe. Ce que Martin Nelly nous a souvent répété. Et cela passe notamment " par l'exploitation du Web ". Aujourd'hui, il ne s'agit plus de tout faire soi-même mais d'assembler différents composants provenant de différents fournisseurs. Mais n'oublions pas un élément important. *" Il faut garder les compétences (en interne) "* avertit le CTO. Bref, si on fait le code technique à l'extérieur ou avec des outils automatiques, l'intelligence de l'application, le fonctionnel, doivent impérativement rester dans l'entreprise, dans l'équipe ! Et cette mutation favorise les usines à logiciels. *" Les SSII mettent en place de telles usines. Internet a permis cela. Il faut des outils très web. "* poursuit Martin Nally. L'autre maître mot de la journée fut process ! Et dans cette orientation, Rational a revu RUP (Rational Unified Process) pour qu'il soit plus simple à mettre en place, plus adapté aux équipes. Assurer un lien entre l'outil et le processus... et s'assurer que les utilisateurs savent l'utiliser au quotidien. Bref, de belles idées durant les plénières, qui restent à mettre en œuvre, et que les 200-250 personnes présentes ont pu apprécier, notamment dans les sessions de l'après midi. ■

valtech training

Au plus court vers vos nouvelles compétences

90 formations au développement logiciel

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

Architecture et intégration
- Introduction à l'architecture orientée service (SOA) (1 jour)
- Ingénierie logicielle objet (3 jours)
- Introduction aux architectures et technologies du Web (1 jour)
- Architectures .Net multi-niveaux (3 jours)
- Urbanisation du système d'information (2 jours)
- Architecture orientée service (SOA) (3 jours)
- Architecture d'entreprise avec Java EE (4 jours)

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

Microsoft .Net
- C# et la conception objet (5 jours)
- Programmation avec VB.NET et conception objet (5 jours)
- Programmation intensive avec le Framework .Net (5 jours)
- Développement d'applications Web avec ASP.NET (5 jours)
- Développement d'applications Windows Forms sur la plate-forme .Net (5 jours)
- Développer des applications graphiques avec WPF (3 jours)
- Gestion des données avec Linq (3 jours)
- Communication inter-applicative avec WCF (3 jours)
- Développer un client riche avec Silverlight 2 (3 jours)

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

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

Gestion de projet
- Gérer des projets avec un processus itératif (4 jours)
- Coacher son équipe projet (3 jours)
- Du recueil des besoins aux exigences : rédiger le cahier des charges (2 jours)
- La conduite de projet (3 jours)
- Gestion de projet informatique (3 jours)
- Manager des hommes dans le cadre d'un projet (2 jours)
- Management de projet (5 jours)
- MSProject (3 jours)

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

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

Méthodes et pratiques agiles
- Test Driven Requirement ou la gestion des exigences dirigée par les tests (2 jours)
- Test Driven Development ou la programmation pilotée par les tests en Java (2 jours)
- Stratégie de test, vérification et validation (3 jours)
- Méthodes agiles de développement logiciel (1 jour)
- Gérer les projets agiles avec Scrum (2 jours)
- Mettre en œuvre le Lean Software Development (3 jours)
- Tests unitaires en Java avec JUnit (2 jours)
- Devenir Scrum Product Owner (2 jours)
- Qualité du code dans les projets Java (2 jours)
- Gestion de configuration logicielle avec Subversion (1 jour)
- Usine logicielle, des concepts à la pratique (3 jours)

Développement Web
- Développement de pages Web avec HTML et CSS (2 jours)
- Développement client avec JavaScript et Prototype (3 jours)
- Développement d'applications Web avec PHP (5 jours)
- Ajax, pour dynamiser le poste client (2 jours)
- Programmation Ajax avec Dojo Toolkit (2 jours)
- Développement d'applications Web avec Struts 2 (3 jours)
- Hacking des applications Web (1 jour)
- Développer des applications avec Adobe Flex 3 (5 jours)
- Conception d'applications Web d'entreprise avec Java EE (5 jours)
- Développement d'applications Web avec JSF (3 jours)
- Développement de composants JSF (2 jours)
- Utilisation du Framework Struts pour le développement d'applications Web (3 jours)
- Développer une application Web avec Ajax et GWT (3 jours)
- Développer des applications pour Adobe Integrated Runtime (AIR) (2 jours)

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

■ **Microsoft** a lancé officiellement Windows Embedded Standard 2009 auprès des constructeurs. Cette version propose notamment Silverlight, le protocole RDP, .Net Framework 3.5. site : www.microsoft.com/windowsembedded

■ **Tidal Software** annonce un outil de gestion des applications web (disponibilité : mi-novembre). Il s'agit de réaliser du monitoring temps réel sur les applications et les transactions. L'outil prend en compte Java, .Net, SOA et supporte les principaux serveurs d'application.

■ **Infragistics** a dévoilé durant la dernière PDC les nouveaux contrôles Silverlight pour l'affichage des données. On disposera de plus de 28 graphiques différents. Supporte Silverlight 2. Plus de détails dans le prochain numéro.

■ **Perforce** propose depuis quelques semaines la version 2008.1 de Perforce, son outil de gestion de sources. On bénéficie ainsi d'un nouveau Visual Client, il supporte les contenus graphiques, facilitant la comparaison de différentes versions. Les utilisateurs mobiles sont aussi mieux reconnus dans l'environnement grâce à serveur proxy plus performant. Site : www.perforce.com

■ **BusinessObjects** propose un lecteur (viewer) standalone pour les fichiers rpt (fichier Crystal Reports). Il permet d'explorer le rapport, de le partager plus facilement, il supporte les nouveautés 2008 et offre un accès au site crystalreports.com.

■ **Talend et JasperSoft** travaillent ensemble pour packager une offre commune dans le cadre de Jasper ETL, powered by Talend. Le meilleur des deux environnements est utilisé pour fournir aux développeurs et utilisateurs un ensemble le plus complet possible dans l'intégration des données et le BI.

RIA
Silverlight, compatible Eclipse !

Sur le CD du numéro !

Si l'annonce mi-octobre de la sortie de Silverlight 2.0 n'est pas une surprise en soi, car attendue, l'annonce d'un plug-in Eclipse a surpris beaucoup de monde. Microsoft a choisi de soutenir le développement du projet via l'éditeur français Soyatec, connu pour son travail autour de Java, UML, XAML. Il s'agit d'offrir aux développeurs sous Eclipse, un environnement complet de conception et de codage Silverlight ! Le projet s'appuie sur l'IDE et surtout sur la plate-forme RCP d'Eclipse. Eclipse2SL ne signifie pas la fin des outils visual studio ou Expression mais cela permet une alternative intéressante pour les développeurs Java par exemple. D'autre part, le projet supporte pleinement la structure projet MSBuild permettant une bonne interopérabilité avec les IDE Microsoft.
Le projet supporte aussi les services web Java. On dispose d'un gestionnaire de projet propre à Silverlight et d'un compilateur. Le développeur dispose d'un éditeur XAML avec autocomplétion du code. La version actuelle supporte C#, le build & run automatique, une configuration web simplifiée. La prochaine étape est prévue fin de l'année avec un meilleur support des projets Silverlight, une interopérabilité accrue avec les middlewares Java. La version finale est attendue pour le printemps 2009. Pour le moment, le plug-in fonctionne uniquement sous Windows.
Site : http://www.eclipse4sl.org/

Conférence
Paris Capitale du Libre

Malgré les difficultés d'organisation et l'absence de la mairie de Paris, l'événement open source de la rentrée, Paris Capitale du Libre, fut une réussite avec plusieurs milliers de visiteurs, des conférences de haute volée et de nombreux stands commerciaux et associatifs. L'éditeur Ingres qui s'investit beaucoup sur le marché depuis quelques mois, était fortement présent durant tout l'événement, dans les conférences et par son stand imposant. Occasion pour montrer son tout nouvel environnement de développement basé sur Eclipse : *Ingres Café*. Un des produits les plus intéressants que nous avons vu, est *blackduck code center* (ainsi que la déclinaison Protex). Il permet de tracer le code, de l'analyser par rapport à des métriques mais surtout par rapport aux licences open source, et de pouvoir identifier l'origine du code ! Ainsi, l'analyse vous dira si tel code vient de telle librairie et le pourcentage de certitude.
Un outil à suivre ! *Symfony* était bien entendu présent, son éditeur travaille actuellement à la version 1.2 en cours de finalisation et la v2 est même déjà en bêta test chez un gros site de diffusion vidéo français ! Côté Business Intelligence, Talend et Jasper étaient très présents avec leurs dernières versions. On pouvait voir une démonstration particulièrement intéressante d'étudiants de l'Epitech sur un développement d'interface virtuelle utilisant la wii remote de Nintendo ! La partie associative / communautaire était elle aussi bien garnie : April, AFUP, Wikipedia, framasoft, jamendo. Enfin, notons aussi la présence de l'outil collaboratif : codendi (édité par Xerox), Mindtouch (environnement wiki particulièrement puissant et personnalisable), Xwiki (autre environnement de wiki, startup française).
Rendez-vous en 2009.

agenda \\

NOVEMBRE

• **MSDN & Technet Tour 2008** - séminaire. Marseille 03 et 04 novembre, Nantes 05 novembre. http://technet.microsoft.com/fr-fr/cc184917.aspx

• 4 novembre, **soirée GWT** au JUG : pour passer 3 heures en compagnie des experts GWT au club utilisateurs Java de Paris. http://www.parisjug.org

• Du 13 novembre au 9 décembre, **tour de France technique WinDev 14** : http://www.pcsoft.fr/pcsoft/tdf-com/2008/index.html

• Le 19 novembre, Paris, **Conférence MySQL 2008** http://www.mysql.fr/news-and-events/european-conferences/2008/

• Du 19 Novembre 2008 au 20 Novembre 2008, Paris - La Défense, CNIT - **Infosecurity 2008** http://www.infosecurity.com.fr/

• Du 19 Novembre 2008 au 20 Novembre 2008, Paris - La Défense, CNIT - **Storage expo 2008** http://www.storage-expo.fr/

• Les 22 et 23 novembre, **concours TopDev.** http://www.topdevone.com

• Les **Roadshows DRM d'Aladdin** se tiendront : le 6 novembre à Strasbourg, le 13 à Nantes, le 25 à Paris. Pour découvrir les dernières nouveautés de la solution HASP SRM. www.aladdin.fr

• Le 25 Novembre, Paris La Défense, toit de la grande Arche, **MD Day 2008**, la 2e édition de la journée du Model Driven http://www.mdday.fr/

DÉCEMBRE

• 1er décembre, **Paris On Rails** : la plus grande conférence français sur le langage Ruby et le Framework Rails ! http://paris.onrails.info/

• Les 8 et 9 décembre, Paris : **Forum PHP**, http://www.afup.org/pages/forum-php2008/

ETRANGER

• Du 10 au 14 novembre, Barcelone, **Microsoft Tech-Ed 2008**,
Le rendez-vous de référence pour plonger au cœur des technologies Microsoft http://www.microsoft.com/emea/teched2008/developer/

• Du 1er au 4 décembre, **Adobe Max**, Milan : la grand messe Adobe sur ses technologies et les prochaines versions et projets des labs ! http://max.adobe.com

Perforce, le système de Gestion de Configuration Logicielle rapide

Le support technique Perforce

Des traitements rapides, des réponses précises

Nos équipes supports sont toujours disponibles afin de partager leurs expertises en personne - pas de réponses automatiques, de répondeurs ou de centres d'appel. Nos ingénieurs supports hautement qualifiés se font une fierté de traiter rapidement vos appels et de vous apporter des réponses précises.

Réaliser vos projets dans les délais nécessites des équipes supports prêtes à vous aider quand vous en avez besoin. Vous pouvez compter sur le système de GCL rapide Perforce et son support technique réputé, pour vous donner un atout gagnant.

PERFORCE
SOFTWARE

Téléchargez sans conditions une copie gratuite de Perforce sur www.perforce.com. Un service d'assistance technique gratuit est offert pendant toute la période d'évaluation.

Sur le CD du numéro !

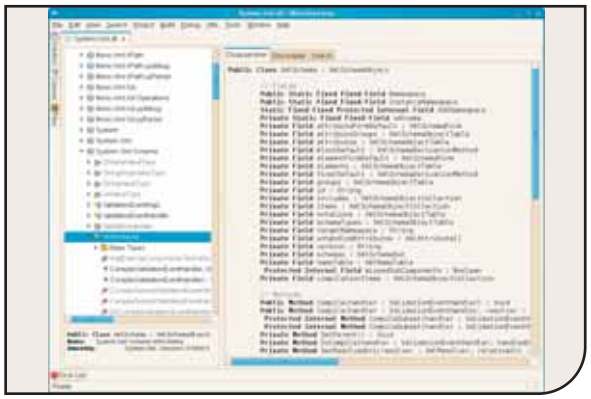
Mono 2.0 enfin disponible !

Mono 2.0 est disponible depuis le 6 octobre 2008. Nous allons faire le point sur les différentes évolutions de l'API ainsi que les outils offerts par Mono. Nous allons également découvrir la nouvelle version de MonoDevelop, un IDE de référence pour le développement sous Gnome, et en particulier en .NET.

Loïc **Guillois**

Mono 2.0 est arrivé en version stable et celle ci confirme la règle : bien que en retrait sur certains points, les développeurs de chez Novell ont permis à l'implémentation .NET open source de prendre de l'avance. Voici la liste des principales nouveautés au niveau de l'API :

- ADO.NET 2.0 API pour l'accès aux bases de données
- ASP.NET 2.0 API pour les applications web
- Windows.Forms 2.0 pour les clients lourds
- System.XML 2.0
- Support du Language Integrated Query (LINQ)
- System.Xml.Linq
- System.Drawing 2.0



La portabilité des applications graphiques bénéficie de l'API Mono.Cairo, un binding de la librairie graphique Cairo qui supporte de nombreuses plates-formes, dont le Serveur X (Linux, Unix...), Windows mais également Quartz et les fichiers SVG, PDF et PostScript. Un support expérimental d'OpenGL, BeOS et OS/2 est notamment intégré.

La solution alternative à Windows.Forms, Gtk#, passe en version 2.12. Cette API graphique sera utilisable aussi bien sous Linux, MacOS que Windows. Elle permet d'accéder aux librairies Gtk+ et Gnome. Les développeurs Linux bénéficieront de l'API Mono.Posix qui permet d'accéder aux fonctionnalités de bas niveau du système d'exploitation. Pour en faciliter l'utilisation, des méthodes de plus haut niveau permettent de dialoguer avec le noyau.

Cela existait depuis longtemps dans le monde Java, avec Mono.Cecil, la manipulation de code compilé devient possible. Ceci vous sera utile si vous souhaitez implémenter un mécanisme de mise à jour ou que vous n'avez pas accès au code source de l'application.

Cela vous permettra également d'implémenter un framework ou des outils de programmation orientée aspect.

Du coté des bases de données, Mono offre désormais le support de SQLite : un mécanisme simple et léger qui permet d'embarquer un SGBD dans vos applications.

Le compilateur et ses outils

Au niveau du compilateur, C# 3 est disponible avec le support complet de LINQ. Les tableaux de très grande taille sont maintenant supportés grâce à l'exploitation des capacités des processeurs 64 bits. Certaines sections de la spécification ECMA ne sont pas encore implémentées. Seule la version 8 de Visual Basic est supportée. Enfin, Dtrace est désormais disponible pour MacOS et Solaris.

Mono 2.0 est la première version à offrir un debugger pour le code managé. Les développeurs apprécieront. Un autre outil fait son apparition : il s'agit du Gendarme. Un moteur de règle qui permet une inspection précise du code pour déterminer les problèmes que le compilateur n'est pas en mesure d'identifier. Les testeurs pourront ainsi mettre en oeuvre des stratégies de tests plus efficaces. Pour terminer, nous parlerons du linker. Celui ci permet une réduction de la taille des exécutables et des librairies en supprimant les portions d'assembly qui ne sont pas utilisées.

Conclusion

Mono a évolué sur le plan de son API et de ses outils mais nous n'avons pas parlé des performances. Les changements depuis la version 1.9 sont remarquables. Les fonctions mathématiques ainsi que le support du multithreading ont été améliorés. Ceci permet de relever le niveau de performances des applications Mono. Mono 2.0 est donc une belle avancée pour le projet qui ravira les développeurs .NET cherchant la portabilité.■

MonoDevelop 2.0 en Alpha

Ça bouge aussi du côté de MonoDevelop. Les développeurs de l'IDE vont nous proposer une version 2.0 d'ici la fin de l'année. Dans la liste des nouveautés, on remarque l'arrivée du support natif du format MSBuild. La gestion des modèles de projet à été repensée pour correspondre à ce format de Microsoft. Ainsi on retrouve la notion de workspace et de solution. MonoDevelop est maintenant capable d'ouvrir plusieurs solutions en même temps. Ceci sera bien utile pour la revue de code. La complétion du code ASP .NET et HTML a été revue et améliorée. Désormais, l'éditeur de code source nous permettra de réduire les portions de codes. La barre de recherche est améliorée et on constate l'apparition de la gestion de thèmes pour la colorisation syntaxique. De nouveaux outils ont été greffés à MonoDevelop afin de tirer la quintessence de Mono 2.0 (navigateur d'assembly, debugger...).

tribune libre

Spéculations sur le futur

Gilbert **Vidal** - C.T.O. idSM

Après une carrière à la direction Europe de Sybase, puis de Versant, il est co-fondateur et CTO de la start-up idSM, éditeur de logiciel, basé à New York.

Quand on vous demande brutalement comment vous avez vécu les dix dernières années de l'évolution de l'informatique, vous avez l'impression d'écrire votre propre *nécro*. Puis, bien sûr, votre vie défile sous vos yeux, et vous concluez par : *à quoi bon ?* Chers lecteurs fidèles, vous qui lisez ce magazine depuis 10 ans, vous qui arrivez à comprendre la totalité des articles, exploit que je serai bien en peine de réaliser, vous en savez bien plus que moi sur ce qui s'est passé !

La seconde impression est *" tout change, rien ne change "*. Nous parlons toujours d'interface utilisateur, même s'il faut maintenant plus de doigts pour la contrôler, nous sommes confrontés aux mêmes Transactions par Secondes, TCP / IP est toujours le protocole de base de l'internet, et le fantôme du 3 270 rôde encore dans certains couloirs.

Ce qui pourrait toutefois arriver serait identique à la fameuse bataille mérovingienne CISC contre RISC, dont certains d'entre vous se souviennent peut-être. Va-t-on voir le Système d'exploitation se réduire à son strict minimum, c'est-à-dire le multitâche, l'accès réseaux et périphériques, et les interfaces utilisateurs " riches " s'installer sur les différents outils disponibles ? Safari et Chrome semblent montrer la voie. Ou bien les OS prendront encore 2 ou 3Go supplémentaires pour moudre le café et faire la vaisselle ? J'aimerais bien la première solution, mais je préférais QUEL à SQL alors.... Finalement, pour regarder vers l'avenir faisons appel aux spécialistes que sont les auteurs de science fiction. Le premier qui nous vient à l'esprit est Frédéric Pohl, avec " l'Ere du Satisfacteur ". Bien sûr, mon Iphone ne m'injecte aucune drogue, mais fait à peu près tout le reste. Je ne regarde plus par la fenêtre pour savoir le temps qu'il fait, et je communique avec ma femme sans me soucier de savoir si elle est dans le salon ou dans la chambre.

World of Warcraft est-il le futur de l'humanité ? Le premier pas vers la vie virtuelle ? Cette pseudo réalité qui vient soit s'additionner, soit carrément remplacer notre bon vieux monde. Clifford Simak " Demain les chiens " (en 1944 !) prévoyait une époque agoraphobique. Quant à Isaac Asimov " Les cavernes d'aciers ", il nous décrivait un monde ou les seules rencontres socialement tolérables se feraient à distance. On pourrait sans doute multiplier les exemples à l'infini, en oubliant opportunément ceux qui se sont trompés. Finalement, l'informatique va-t-elle accélérer ou ralentir les modifications de nos comportements ? Mon avis strictement personnel est une accélération. La généralisation du Cloud Computing, et l'ubiquité de la communication renforcent l'isolement physique, et dégradent la densité des échanges (lol, mdr !). Quels peuvent être les changements de paradigmes qui viendraient casser la simple accélération mécanique de cette évolution ? ". Elle est peut être déjà là, ayant compris à sa naissance en une milliseconde que la paranoïa de son créateur entraînerait une destruction immédiate généralisée si Elle se faisait connaître.

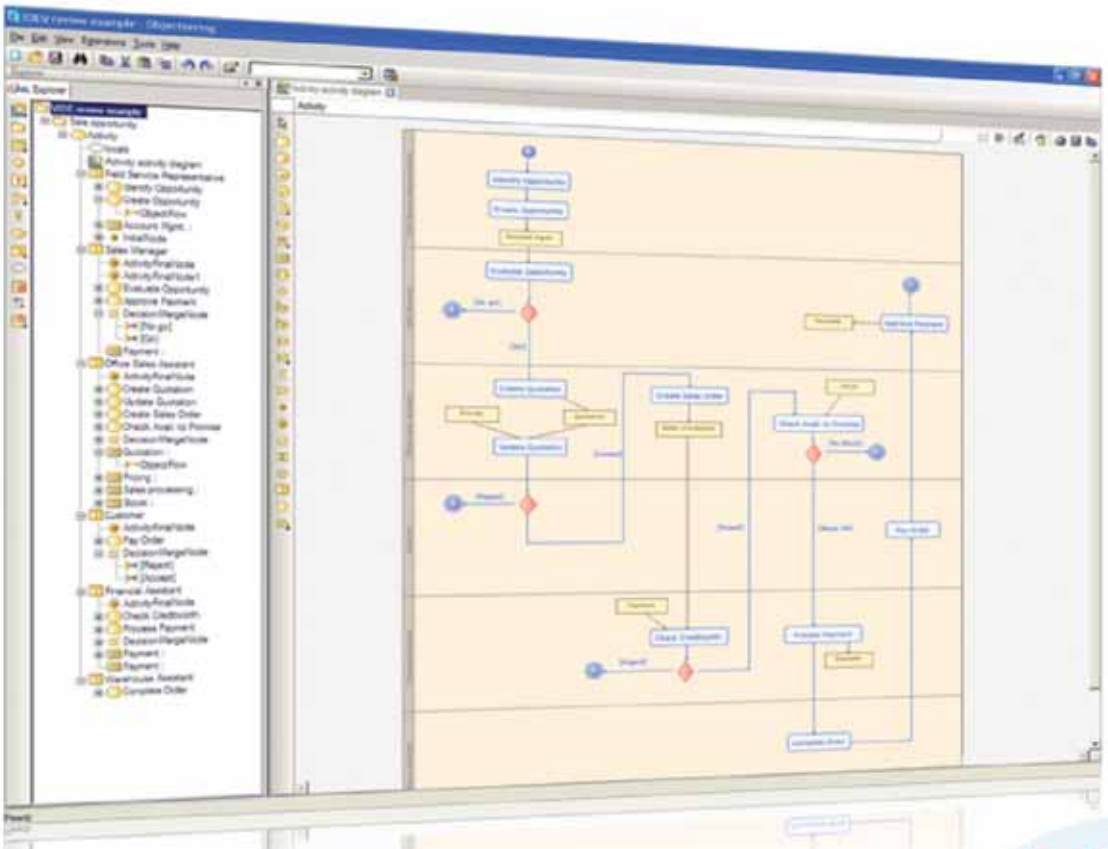
> *" Certains prévoient l'apparition ex-nihilo d'une Intelligence Artificielle née de l'interconnexion de millions de CPU "*

Plus sérieusement, je crois que l'évolution la plus fondamentale viendra de l'interface directe entre le Réseau et l'Individu. William Gibson " Neuromancer " (1984) montre une direction possible de notre évolution. Pas aussi radicale que celle proposée par le biologiste Axel Kahn, qui prévoit une modification volontaire de notre ADN, mais une extension de nos capacités cérébrales par fusion de la conscience avec le Réseau. La vitesse des découvertes scientifiques semble être une fonction exponentielle du nombre de chercheurs liés, et le gestalt résultant d'une meilleure intégration pourrait donner des résultats surprenants. Le nombre de projets réalisés pour connecter directement les prothèses aux influx nerveux, de même que l'apparition de manettes de jeux vidéos interfacées aux ondes cérébrales, démontrent que cette évolution intéresse diablement nos investisseurs.

Maintenant, à plus long terme, il semblerait naturel de trouver un support plus durable pour notre cerveau et notre conscience. Le silicone n'ayant pas la plasticité nécessaire, quelques recherches seront sans doute indispensables.

Your projects deserve a tool*

Objecteering Modeler



Objecteering 6.1

guide vos développements par les modèles sur toute la portée du système

Objecteering 6.1 est une solution de modélisation complète et ouverte supportant les standards OMG de modélisation UML 2.1, BPMN, et SysML, la modélisation de l'Architecture d'Entreprise et la modélisation d'une architecture SOA.
Il intègre le support de l'analyse des besoins et de la définition du dictionnaire, assurant ainsi une traçabilité complète sur tout le cycle de vie.
La technologie MDA associée à une ouverture de l'outillage en Java permet de guider le développement à chaque contexte utilisateur, chaque infrastructure technique cible. Ses générateurs sur étagère automatisent le développement d'applications Java, J2EE, .Net C#, C++, SQL...

Objecteering Software, éditeur de l'atelier Objecteering 6, est le spécialiste français UML/MDA pour le développement d'applications guidé par le modèle. Son offre modulaire couvre le cycle de vie de la gestion des exigences jusqu'au déploiement d'application.
Pour plus d'information sur Objecteering 6.1, pour télécharger Objecteering **Free Edition** ou Objecteering **Enterprise Edition**, rendez-vous sur : www.objecteering.com

Architecture d'Entreprise
UML2 BPMN
C# MDA SysML
Java SQL
J2EE SOA
C++

Objecteering SOFTWARE

Venez découvrir
Objecteering 6.1 à

<MDDAY/> 2008

le 25 novembre
Toit de la Grande Arche
Inscription gratuite sur
www.objecteering.com

* Vos projets méritent un outil

www.objecteering.com - Tél. : 01 30 12 16 60 - sales@objecteering.com

SQL Server 2008 : compression et développement

2e partie

Dans le numéro précédent nous avions exploré le Resource Governor. Ce mois-ci, nous allons revenir, toujours avec Pascal Belaud, sur les nouveautés de compression et le développement.

Data Compression

Les données que gèrent les entreprises au quotidien sont de plus en plus nombreuses. Ceci a pour conséquence des tailles de base de données de plus en plus conséquentes. Cette remarque s'applique également aux fichiers de sauvegarde. Avec SQL Server 2008, une nouvelle fonctionnalité de compression a été mise en place et elle porte le nom de " Data Compression ". Elle s'applique aux tables ainsi qu'aux index de vos bases de données. Deux stratégies d'économie d'espace disque ont été mises en place dans SQL Server 2008.

Row Compression

Toutes les colonnes dont le type impose une taille fixe sont stockées sous forme de type variable. Ceci n'a aucune incidence pour vos applications qui continuent de voir ces colonnes comme des colonnes de taille fixe. Un exemple est le type " int " qui est stocké sur 4 octets. Si la valeur entière stockée est inférieure à 255, un seul octet suffit pour la stocker (comme pour le " tinyint "), soit une économie de 3 octets, c'est-à-dire finalement 75% d'économie. De la même manière, si vous avez une colonne de type " char(100) ", ceci impose de stocker 100 caractères systématiquement, même s'ils ne sont pas entièrement utilisés. Dans ce cas, la compression de données offerte par SQL Server 2008 va nous permettre de stocker cette colonne sous forme variable. Dans le cas où cette colonne hébergerait les valeurs " Football " ou encore " Olympique de Marseille ", on voit que le stockage ne nécessiterait que 8 caractères dans le premier cas et 22 caractères dans le second, soit une économie de 92% et 78%, ce qui est loin d'être négligeable.

Page compression

Dans ce cas, toute la stratégie repose sur le fait de ne pas dupliquer les valeurs contenues dans les colonnes. Supposons que nous ayons une table définie par le script DDL suivant :

```
Table Employe
(
  nom varchar(100),
  statut varchar(30) default 'Contrat à Durée Illimitée (CDI)'
)
```

Dans le cas précédent, on voit que la valeur " Contrat à Durée Illimitée (CDI) " sera systématiquement positionnée sur la colonne si aucune valeur n'est saisie. Dans ce cas, la compression de page va nous permettre de ne stocker cette valeur qu'une seule fois et d'y faire référence autant de fois que nécessaire. Il est évident que l'efficacité de cette stratégie va directement dépendre du niveau de redondance des données. Si cette stratégie est choisie, celle-ci inclura obligatoirement la stratégie précédente, à savoir " Row Compression ".

Paramétrage

Il y'a deux manières de mettre en place la compression de données, l'une via l'outil d'administration de SQL Server 2008, le " Management Studio " ou via un script DDL. [Fig.1 et 2]
Ceci peut être également effectué via un script DDL :

```
Use [bMSpress]
Alter Table [Dbo].[Ouvrages] rebuild partition = All
with (Data_Compression = Row)
```

Il est également possible de déterminer à l'avance une estimation des gains que l'on peut obtenir en décidant de compresser telle table ou tel index. Il suffit d'appeler pour cela la procédure stockée " sys.sp_estimate_data_compression_savings " :

```
exec sys.sp_estimate_data_compression_savings  'dbo','Ouvrages',
NULL,1,ROW
```

[Fig.3]

Transparent Encryption

La protection des données de l'entreprise est un sujet qui prend de plus en plus d'importance. Par exemple, dans certains cas, la loi oblige les entreprises à garantir un grand niveau de confidentialité pour

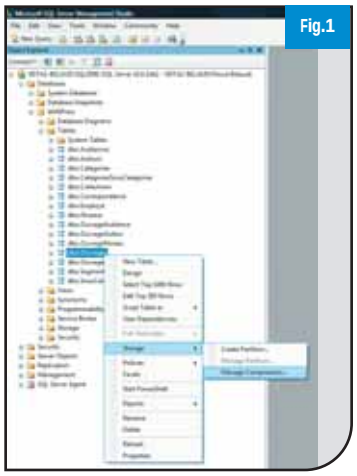


Fig.1


Fig.2

Fig.3

	object_name Ouvrages
schema_name	dbo
index_id	1
partition_number	1
size_with_current_compression_setting (KB)	2 912
size_with_requested_compression_setting (KB)	2 896
sample_size_with_current_compression_setting (KB)	2 648
sample_size_with_requested_compression_setting (KB)	2 640

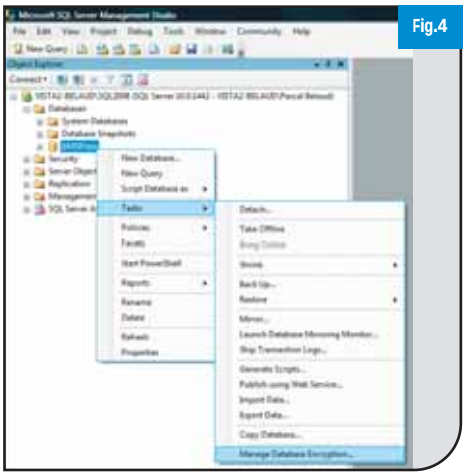


Fig.4

certains types de données. Dans SQL Server 2005, il était possible de crypter manuellement les données présentes dans certaines de nos colonnes, mais ceci devait être réalisé de manière explicite et les applications interrogeant ces données devaient avoir été prévues pour lire ces données cryptées. Avec SQL Server 2008 apparaît cette notion de chiffrement transparent des données. Ceci va permettre de se prémunir contre le vol des fichiers de données (mdf et ldf) ou encore des fichiers de sauvegarde de notre base de données en chiffrant, de manière transparente, ceux-ci. Pour la démonstration de cette fonctionnalité, nous allons créer à la fois une clé de chiffrement basée sur un mot de passe mais également un certificat :

```
Use master
Create Master Key Encryption By Password = 'Marseille jouera
la Ligue des Champions en 2008/2009 !'
Create Certificate MonCertificat With Subject = 'Mon certificat
 de chiffrement'
GO
```

Ensuite, nous pouvons maintenant encrypter la base de données en utilisant l'outil d'administration de SQL Server 2008, le " Management Studio " : [Fig.4 et 5]
Ceci peut également être obtenu en exécutant le script DDL suivant :

```
Create database encryption key with algorithm = Aes_128 encryption
 by server certificate [MonCertificat]
Alter database [bMSPress] set encryption on
GO
```

Désormais, les fichiers de données, comme tous les fichiers de sauvegarde qui seront créés à partir de maintenant, sont chiffrés et protégés contre tout accès indésirable.

DÉVELOPPEMENT

Linq To SQL

Avec le Microsoft .NET Framework 3.5 (et Microsoft Visual Studio 2008) est apparue une nouvelle technologie qui permet aux développeurs, dans leur langage de développement préféré (C# ou VB par exemple) de requêter des données pouvant être sous la forme de tableaux en mémoire, de documents XML, de DataSet ou encore de base de données de type SQL Server.
Cette fonctionnalité, baptisée " LINQ " pour " Language INtegrated Query ", permet aux développeurs d'exprimer ce qu'ils souhaitent faire plutôt que d'expliquer comment le faire. On se rapproche clairement des langages dits fonctionnels (tel que " F# " par exemple).

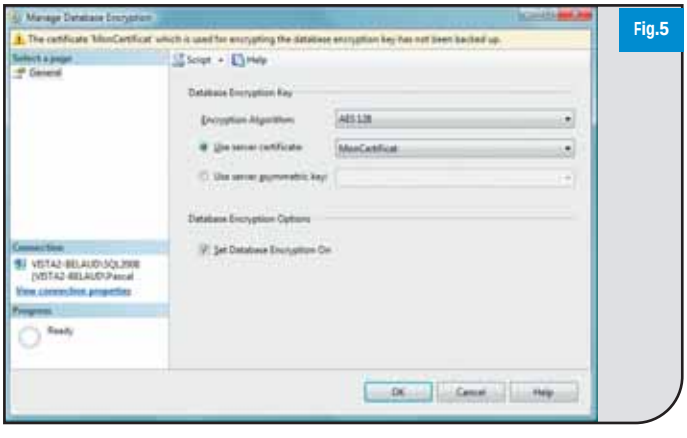


Fig.5

L'expression de ce besoin se fait en utilisant une syntaxe proche de la syntaxe SQL (" Query Expression "). Et pour cause, le langage SQL a clairement été conçu pour aider au requêtage des données. Voici un exemple en VB du requêtage d'un tableau d'objet en mémoire. Nous souhaitons récupérer l'ensemble des processus actuellement chargés en mémoire, dont l'empreinte mémoire est supérieure à 10Mo, le tout classé par ordre croissant de nom de processus :

```
Imports System.Diagnostics
Dim mesProcess = From monProcess In Process.GetProcesses() _
                 Where monProcess.WorkingSet64 > 10 * 1024 * 1024 _
                   Order By monProcess.ProcessName Ascending _
                    Select New With _
                    { _
                         .Nom = monProcess.ProcessName, _
                         .Mémoire = monProcess.WorkingSet64 _
                    }
```

Une des " versions " de " LINQ " s'appelle " Linq To SQL " et permet de faire la même chose mais en attaquant une base de données SQL Server. Une fois qu'on a déclaré les tables auxquelles on voulait avoir accès (en ne retenant que les colonnes qui nous intéressent et en renommant celles choisies si on le souhaite), on peut requêter ces tables de la même manière que l'exemple précédent :

```
Dim monDataContext = New MSPressDataContext()
Dim mesLivres = From monLivre In monDataContext.Ouvrages _
                 Where monLivre.Ouv_CurPrix > 5 AndAlso _
                       monLivre.Ouv_CurPrix < 90 _
                 Order By monLivre.Titre Ascending _
                 Select monLivre
```

La question désormais est : pourquoi avoir prévu une telle technologie alors qu'elle existe déjà dans SQL Server et se nomme le langage SQL lui-même ? Et bien, il y'a plusieurs arguments en faveur de LINQ et l'un d'entre eux tient au fait qu'on puisse effectuer des requêtes qui impliquent des données aussi différentes que des tableaux en mémoire, des documents XML, des DataSets ou encore des bases de données SQL Server, le tout en même temps ! Par exemple, on aurait pu mélanger la requête précédente sur les processus chargés en mémoire avec des descriptions des processus qui se trouveraient dans des documents XML et des statistiques d'utilisation de ces processus qui se trouveraient quant à elles dans une base de données SQL Server.

CONCLUSION

Comme on l'a vu dans cet article, SQL Server 2008, sans remettre en cause les acquis de SQL Server 2005, fournit son lot de nouveautés, et ce, dans des domaines aussi variés que la performance, la gestion administrative au quotidien ou encore le développement d'applications. SQL Server 2008 est disponible en version finale et il ne vous reste plus qu'à tester vous-même toutes ces nouveautés pour vous faire votre propre idée sur la question !

■ Pascal **Belaud**, *Microsoft France*
Pascal est en charge, depuis plus de 10 années, de la relation technique avec les développeurs chez Microsoft France. Vous pouvez le retrouver sur son blog à l'adresse : http://blogs.msdn.com/Pascal

Soyez productif !

Comment être un développeur pragmatique ?

François Tonic

Voilà un titre et une promesse ambitieuse pour le développeur. Il faut avouer que lorsque nous avons évoqué ce dossier au sein de Programmez !, l'enthousiasme a rapidement tourné à l'interrogation. Car finalement, un développeur peut-il être productif et si oui, comment ? Le déclic est venu du livre de Neal Ford intitulé *The Productive Programmer* (O'Reilly, 2008). Il est rapidement devenu notre fil rouge pour ce dossier.

La notion de productivité cache une extraordinaire diversité : les méthodes, l'organisation de son bureau, créer des scripts d'automatisation, faire moins de clics, appliquer des bonnes pratiques, des métriques, imposer les tests dès le départ, rester concentré sur son travail, etc. Mais il y a des petits détails qui peuvent changer la vie d'un développeur, au quotidien. Quelques minutes ici, moins de gestes là-bas, suffisent à améliorer son confort de travail. Et ces minutes s'accumulant peuvent faire la différence à la fin du mois… Aujourd'hui, nous pouvons dire, sans faire hurler, que les grandes tendances sont : le pragmatisme de développement (réutilisation), faire du code utile rapidement (= le code n'a pas besoin d'être beau) répondant à la fonction, à une demande.

La productivité se heurte-t-elle à la production de code ? Notre sentiment, et expérience personnelle et professionnelle, est : *" oui, la production de code peut nuire à la productivité, à l'efficacité du développeur "*. Bien sûr quand on code un projet personnel, à la maison, à ces heures perdues, par " amusement ", peu importe. Par contre, prenez un développeur freelance, peut-il se permettre de perdre des heures à peaufiner le code d'une fonction parce qu'il veut faire un code carré, parfait, esthétique ? Non, car ce n'est plus rentable pour lui et peut nuire aux délais de livraison. Le développeur pragmatique n'est pas un vain mot. Et durant notre longue enquête, souvent, ce terme est revenu. Il faut produire le plus rapidement possible, le mieux possible, en réutilisant du code existant, en s'appuyant sur des librairies, frameworks. Bref, minimiser le code technique à écrire car il est souvent verbeux et sert la plupart du temps de glue, de lien technique. Le développeur doit se concentrer sur l'essentiel : le codage des fonctions. Et cela est d'autant plus vrai quand on travaille en équipe.

Nous verrons dans ce dossier que l'on peut optimiser son poste de travail, ses outils, automatiser les tâches récurrentes, constituer une bibliothèque de code, savoir communiquer efficacement quand on travaille en équipe, etc. Vous vous direz en refermant Programmez !, *" c'est tout bête ! "*. Oui, cela peut paraître basique mais c'est souvent avec ces petits détails, ces petits réflexes, que l'on gagne du temps !

Alors soyons productif dès maintenant !

Le mois prochain : les outils de collaboration

“Faites les choses bien, et rapidement”

Qu’entend-on au juste par : “ être productif ” ? Chez Uperto, on pourra vous dire : *“Demandez que l’on vous développe une page de code, chiffrée à 2 jours de travail. Certains développeurs mettront à peine 3/4 de journée, d’autres plus des deux jours prévus”…* Comme nous le verrons ici et dans les prochains numéros, la productivité dépend de beaucoup de facteurs.

“Il ne faut pas avancer sans réfléchir à produire du code. Il faut savoir gérer son temps, être pragmatique” indique **Pierre Stockmann** (développeur, Neos-SDI). Faut-il tout coder, utiliser des frameworks pour gagner du temps, même si au début on en perd car il faut l’apprendre, le maîtriser ? L’utilisation de framework est un élément de la productivité.

Il est important d’éviter de devoir créer un code récurrent ou un code purement technique. Il faut se concentrer sur le code fonctionnel. Les bonnes pratiques ne sont pas des prérequis incontournables, il faut déjà réfléchir à la fonction à développer, puis la réaliser de manière la plus rapide possible.

Des astuces partout pour gagner du temps partout !

Si beaucoup de développeurs, notamment Java et .net, utilisent des IDE possédant des centaines de fonctions, il est malaisé de les maîtriser et de les configurer à sa manière.

Et il est vrai qu’un développeur novice risque de perdre beaucoup de temps et d’énergie à prendre en main l’outil. Outre cette phase de formation, obligatoire et incontournable, il faut aussi apprendre toutes les petites astuces pour aller plus vite. Les raccourcis claviers, la colorisation du code, les *scriptages* pour automatiser certaines tâches peuvent faire économiser un précieux temps. Cela n’a l’air de rien, mais quelques minutes économisées par jour font de grosses heures à la fin d’un projet.

Standardiser, automatiser son poste de travail

De bonnes économies de temps peuvent être faites sur le poste de travail. Neal Ford indique plusieurs éléments qui simplifient les accès aux outils et aux fonctions.

1. savoir utiliser et configurer son lanceur, cela peut être Dock sous OS X, le menu démarrer de Windows, etc. Cela a l’air tout bête mais au lieu d’avoir plusieurs sous-menus pour accéder à son programme, il faut avoir un seul niveau, ou tout regrouper dans un unique menu, quitte, à faire des raccourcis. C’est un confort d’utilisation du système.
2. Avoir un ordinateur en adéquation avec ses développeurs. La performance de vos outils dépend de la mémoire vive installée, de la vitesse du disque dur, de la qualité de la carte graphique, de la propreté du système, etc. Selon le langage, il faudra plus ou moins de ram… Et si un Windows XP suffit pour vos développements, pourquoi passer à Vista ?
3. User et abuser des raccourcis du système, des outils !
4. La souris n’est pas toujours votre meilleure alliée, le clavier est souvent bien plus rapide. Passez alors

Pierre Stockmann

Contre-exemple : attention à la motivation

Dans ce qui suit, je prends l’exemple d’une entreprise T existant au Danemark où la productivité des développeurs s’avère décroissante en raison de certains facteurs directement imputables à la politique informatique de cet établissement. En premier lieu, les développeurs de l’entreprise se voient attribuer diverses tâches (que je qualifie de) computo-administratives en sus de leur activité stricte de développement. Ces tâches consistent, par exemple, à installer des logiciels – afin de faciliter le travail du personnel administratif –, à résoudre des problèmes informatiques (virus, " système instable ", fichiers endommagés etc.), à signaler l’absence de certaines applications ou le renouvellement du matériel informatique. Cette diversification de l’activité du développeur entraîne une démotivation et un non-enrichissement personnel, car le développeur fait " plusieurs choses " sans pour autant approfondir ses connaissances dans d’autres domaines que le sien. Dans cette entreprise, un développeur m’a fait part de sa frustration concernant l’absence de documentation détaillée pour les applications dont il fait usage quotidiennement et quasi-machinalement. La documentation est présente dans cette entreprise, mais elle reste très succincte et il n’y a pas de description de l’architecture du système, de l’utilisation de certaines fonctions, de la structure du code utilisé. A propos précisément du code, notre développeur a accès à celui d’un certain système d’information utilisé par l’entreprise, mais le code est très complexe, car il se compose d’un mélange de diverses technologies – telles que la technologie Unix (avec les sockets, les sémaphores) – que ne maîtrise pas forcément le développeur lambda – et la programmation C++…

■ Dr. Rodrigue Sabin Mompelat
Enseignant-Chercheur, Ingénieur Logiciel, Copenhague – Danemark

par la ligne de commande… pour exécuter, rechercher…

5 Utiliser les outils complémentaires des éditeurs. Sous Windows, vous disposez de PowerToys, de Powershell.

6 Gardez un poste de travail toujours propre sans y installer des pré-versions, des outils dont vous ne connaissez pas le comportement avec votre outillage de production. Installez-les dans une machine virtuelle ou une autre machine. Votre poste de développement doit rester neutre. Et pensez à le sauvegarder, à créer une image de votre environnement de travail !

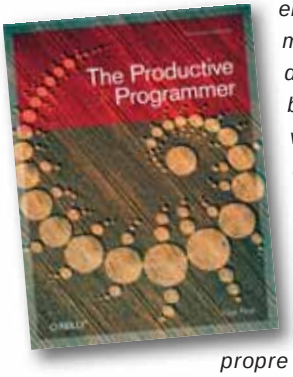
7 Quand c'est possible, créez des macros, des scripts pour automatiser

8 Regrouper les codes réutilisables pour éviter de les chercher !

"Il faut optimiser son poste de développement. J'ai créé des scripts, j'utilise une police de caractères qui me convient. On utilise aussi les codes snippet. Il faut également voir ce que l'on pourrait réutiliser pour un autre projet, ou au moins en reprendre l'idée. Il faut aller vite sur les éléments simples et répétitifs. Je note systématiquement mes idées sur One Notes," poursuit **Pierre Stockmann** (développeur, Neos-SDI). La factorisation du code est donc incontournable pour avancer plus vite. Par exemple, si vous devez créer des dizaines de formulaires avec les mêmes objets comme des textbox, pourquoi ne pas créer son propre textbox préconfiguré pour aller plus vite, avec un validateur configuré.

Neil Ford pointe aussi un autre élément non négligeable : la distraction. Il faut pouvoir garder un haut niveau de concentration pour résoudre les problèmes, trouver les bonnes solutions. Il faut éviter les distractions visuelles et auditives. Par contre : bloquez le téléphone, préférez la messagerie instantanée ou le mail, mais là encore à vous de bien les utiliser, de répondre à des moments définis. En environnement ouvert, ou à la maison avec un enfant en bas âge, il faut savoir s'isoler pour garder la concentration. Cette phase peut passer par des plages horaires de tranquillité où vous êtes réellement isolé. Cela peut être utile quand on travaille en équipe. Et là il faut éviter téléphone, mail, réunion, etc. Mais il faut aussi savoir aménager des moments de détente pour se reposer, décompresser. Car la concentration s'estompe rapidement, au fil des heures.

Benoît Gueuniot



Stéphane Malbequi

Les bons outils !

"C'est primordial ! Par exemple, en Java, il faut disposer des bons plug-in, en .Net on pourrait utiliser un module comme Resharper. Le développeur doit constituer sa boîte à outil. Il doit aussi découvrir les raccourcis clavier pour aller plus vite. Mais en équipe, les développeurs doivent disposer des mêmes outils ! Le choix est immense et il y en a beaucoup. Mais les clients peuvent aussi imposer leur propre choix. L'un des premiers outils à choisir est le gestionnaire de source, puis les plug-in de développement. Et chaque membre de l'équipe peut apporter son vécu, son expérience. L'équipe partage et choisit de concert." précise Stéphane Malbequi (consultant sénior et architecte et scrum master chez Valtech).

Et en équipe ? La collaboration !

Travailler en équipe n'est jamais simple et il faut que chaque développeur y mette du sien pour s'intégrer, interagir, avancer efficacement. Mais au-delà de cela, il y a aussi toute la notion de collaboration, de partage des documents, de l'information, des codes. Pour le partage, un des outils incontournables est le wiki. Un contre-exemple, la SSII Neos SDI sur sa plateforme de développement ne l'utilise pas. *"cela ne nous est pas venu à l'esprit ! Le wiki oblige les gens à le remplir, à le faire vivre",* explique Pierre Stockmann. *"Par contre la messagerie instantanée est utile ". " Chez Valtech, nous possédons un wiki, c'est notre point d'accès rapide à la documentation, tout le monde y apporte des données."* précise Stéphane Malbequi.

Dans une équipe, il faut homogénéiser les postes de travail surtout quand il y a cohérence dans les technologies et outils utilisés pour les projets. En effet, il faut éviter que chaque développeur utilise un outil différent. Cela peut engendrer des difficultés d'intégration des codes. *"Nous utilisons Visual Studio 2008 et nous fournissons à nos développeurs une machine virtuelle unique. On maîtrise ainsi les usines à développement"* intervient **Benoît Gueuniot** (Business Manager du Centre de Services de Dijon, Neos-SDI). Il faut alors définir les outils qui seront utilisés par le projet. On package l'ensemble, on préconfigure selon les "normes" définitions, puis le développeur charge la machine virtuelle sur son poste. Mais ce package nécessite une concertation entre le chef de projet, l'architecte et les développeurs. Ces derniers doivent pouvoir donner leur avis même si le client peut imposer tel outil. *"Cependant, une machine virtuelle est gourmande en ressources. Mais cela reste un formidable outil !"* affirme Benoît Gueuniot.

Bonnes pratiques, méthodes agiles : oui mais.

"Il faut bien organiser le projet !" précise avec vigueur Pierre Stockmann. Si dès le départ, dans le cas d'une architecture n-tiers, les couches sont mal définies, le projet partira à la dérive. L'utilisation d'outils de type MVC peut aider à cette rigueur mais aux développeurs de respecter la philosophie de ces frameworks. Avec des cycles de développement de plus en plus courts, les méthodes agiles peuvent aider.

Et d'autre part, le client peut aussi les imposer, donc vous n'aurez pas le choix ! *"Un environnement de gestion du cycle de vie et des moyens pour y arriver"* indique Benoît Gueuniot. Il faut une démarche globale à mettre en place et toute l'équipe, le chef de projet, l'architecte, le client doivent pouvoir s'y connecter et travailler dessus. C'est un moyen d'être plus réac-

tif et plus transparent. Mais, les bonnes pratiques peuvent rendre le code rigide si on les suit trop strictement, il faut savoir prendre ce qu'il faut et ne pas les appliquer strictement. Un des préceptes qui nous a souvent été répété durant notre enquête : le développeur doit sortir du code purement technique pour aller vers le fonctionnel, là où est l'intelligence de l'application. Le développeur purement technique aura parfois du mal à comprendre les attentes du client, ce que l'on attend d'une fonction. Or, une fonction mal comprise nuit ensuite au développement.

Le développeur doit s'informer des bonnes pratiques, il doit savoir faire valider son code régulièrement, de nombreux outils existent pour cela. Il ne faut pas attendre la fin du développement pour le faire, surtout en développement en équipe qui nécessite une intégration de différents codes. Les composants peuvent être un autre moyen d'être productif même s'ils nécessitent un apprentissage. Mais au final, ces composants graphiques simplifient beaucoup la conception et le codage de l'interface.

"Les méthodes agiles améliorent la productivité, comme le TDD" analyse d'emblée Stéphane Malbequi. *"On implémente dans le projet ce qui est demandé et sur la phase développement, on gagne du temps ! Le TDD permet de tester le code par rapport aux tests initiaux sans aller dans l'ensemble de l'application. Mais selon le projet, il faut adapter le niveau de la méthode agile même si elle s'adapte à peu près à tout !".*

Cyril Durand : le quotidien d'un développeur freelance

"Je travaille principalement de chez moi, le reste du temps chez mes clients, généralement une journée toute les 2 semaines mais cela varie en fonction de la mission et de leur localisation géographique.

Lorsque je suis chez moi, j'utilise beaucoup Skype, le télétravail ne signifie pas travailler tout seul. Je préfère poser une question au client plutôt que de partir sur une fausse piste. Pour les questions urgentes et simples, j'utilise la messagerie instantanée mais s'il le faut, je n'hésite pas à appeler le client lorsque le problème devient plus épineux. La messagerie instantanée est souvent à l'origine de quiproquos. Pour les questions moins urgentes, j'envoie un mail. Chaque semaine, j'organise une réunion avec le client, par Skype ou de vive voix. Chaque réunion possède son propre compte-rendu sur OneNote, que j'utilise beaucoup (je m'en sers comme d'un " journal de projet ", toute mon activité autour du projet y est notée). L'avantage, par rapport à un wiki, est de pouvoir collaborer sur un même document à plusieurs, en live. Je trouve cet outil très pratique, mes clients aiment cette approche et font souvent de même par la suite.

Travailler de chez soi est parfois difficile : il faut se fixer des objectifs à réaliser, s'occuper d'une tâche à la fois, éviter de se disperser. C'est pour cela que chaque semaine je définis mes objectifs et note chacune de mes tâches une fois celle-ci accomplie. Cette liste de tâches, sur le OneNote du projet, cela permet de donner de la visibilité aux clients sur ce que je fais. Je sais que l'après- midi j'ai du mal à rester concentré, je fais de la veille : je lis, je blogue, je bêta-test, ou alors je sors prendre l'air. Du coup mes horaires sont peu conventionnels ;-)

Coté code, j'utilise principalement Visual Studio, je n'ai pas de Framework personnel, le code étant souvent lié à un projet client. Cependant, j'utilise de nombreux bouts de code que je fais évoluer au fur et à mesure des projets lors de mes veilles technologiques. Pour le partage des sources, je m'adapte au client, souvent Team System. J'envisage de passer à Subversion pour mes projets personnels : cela correspond mieux à mes besoins. Pour mon poste de travail, j'ai récemment acheté une grosse machine hébergeant hyper-v sur laquelle je fais tourner différentes machines virtuelles. C'est très pratique pour les tests projet ou pour tout simplement tester des outils, des technologies que je ne connais pas, ou en pré-version. Je ne veux pas polluer ma machine de travail avec des outils dont je ne connais pas les incidences.

Lorsque je rencontre un problème, j'utilise un cahier papier qui sert à gribouiller mes idées, j'y pose le problème, l'erreur et j'essaie de réfléchir hors du clavier. Cela me permet d'aller à l'essentiel, de mieux comprendre le problème et trouver une solution.

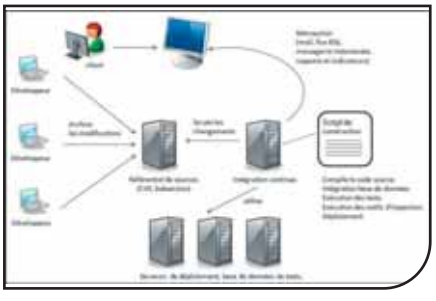
Un conseil pour gagner en productivité ? Maitrisez vos outils ! Visual Studio regorge de fonctionnalités méconnues, soyez curieux !"

François **Tonic**

Netapsys mise sur l'industrialisation et la productivité

Nos projets de développement concernent essentiellement des réalisations au Forfait, d'applications informatiques basées sur les nouvelles technologies. Dans ce cadre, nous travaillons sur plusieurs axes afin d'optimiser la productivité de nos projets : les outils, la méthodologie et les hommes.

Les outils s'inscrivent dans une démarche d'industrialisation des développements. Nous travaillons par exemple sur une plate-forme d'intégration continue. Les outils d'industrialisation sont primordiaux mais ils ne sont pas suffisants : ils ont des impacts méthodologiques importants et doivent s'inscrire dans une culture d'entreprise adaptée. Nous considérons qu'il n'y a pas une méthode universelle qui s'applique avec succès à tous les contextes de projets. L'organisation dépend de la culture professionnelle des différents acteurs, des méthodes et habitudes de travail existantes, du type de projet, ... Mais nous sommes convaincus que les méthodes agiles nous permettent d'améliorer nos performances globales. Nous introduisons



donc des éléments méthodologiques agiles à chaque fois que cela s'avère possible.

Productivité et tests automatisés sont-ils compatibles ?

L'automatisation des tests engendre nécessairement un coût sur un projet, et se trouve de ce fait susceptible de ralentir le développeur. Un tel examen superficiel conduit à la conclusion que l'automatisation des tests a tendance à diminuer la productivité.

■ Dominique **Méra** - *Objet Direct - Consultant – Directeur de projets*
■ Illustrations par Ludivine **Alligier**

Ce raisonnement ne tient pas compte du gain obtenu au moment du passage des tests, puisqu'un test automatique ne coûte presque rien à passer. La question fondamentale est donc de déterminer si le gain compense l'investissement initial. Sur ce point les avis sont partagés.

A priori, un test donné a de bonnes chances d'être exécuté un nombre de fois non négligeable. Prenons l'exemple d'un écran : le développeur mettra deux heures à le mettre en place. Ensuite, il convient d'effectuer un premier test pour vérifier qu'il fonctionne. Que ce test soit manuel ou automatique, il est fort probable que le développeur identifie à travers lui une erreur dans le code écrit. Cette erreur rectifiée, le développeur effectuera de nouveau le même test. Ce cycle sera répété tant que le test ne passera pas avec succès. Le nombre de cycles est en général corrélé avec l'expérience du développeur, et varie de 1 à 10, voire plus. Ultérieurement, une seconde passe de tests pour ce même écran sera exécutée juste avant la livraison. Or la phase de tests conduite par les utilisateurs suite à cette livraison amène la plupart du temps à la détection d'un certain nombre de demandes de changement (bugs ou évolutions).

La prise en compte de ces demandes conduit à une nouvelle livraison. En toute rigueur, l'ensemble des tests devrait être exécuté de nouveau préalablement à la re-livraison, pour garantir l'absence de régressions.

En pratique, le coût de cette nouvelle phase de tests globale étant rédhibitoire, seul un sous-ensemble de ces tests est effectivement exécuté, dans le meilleur des cas.

Il apparaît alors un risque sur la qualité du produit. Bien entendu, le cycle de tests utilisateurs / corrections / relivraison se produit en pratique plusieurs fois et est susceptible de retarder la mise en production (ou la distribution) de la version définitive de logiciel. Il n'est pas rare qu'un même écran soit testé 10 ou 20 fois dans le processus. Sans compter les tests qui seront nécessaires pour les versions successives pendant la durée de vie de l'application.

Figure 2 : Lorsque les tests ont été automatisés, la batterie complète peut être exécutée avec un effort modeste.

Optimiser les coûts

Le coût initial de l'automatisation des tests peut très bien s'avérer inférieur au coût global des tests exécutés manuellement. Il s'agit d'une certaine manière d'un investissement : le temps passé sur l'automatisation, qui représente un coût immédiat, devra rester inférieur au temps global des tests manuels.L'évaluation de rentabilité n'est possible que pour un projet donné, en tenant compte du contexte. L'expérience montre que dans la majorité des projets, l'automatisation des tests permet de fournir un logiciel de qualité supérieure à un coût inférieur et dans un délai plus court. ■

Nos recommandations

- La totalité des tests ne peut pas être automatisée. La règle des 80 / 20 s'applique : 80% des tests peuvent être automatisés avec 20% de l'effort. Il convient donc de ne pas automatiser les tests qui seraient onéreux à l'être.
- Construire un nombre de jeux de données restreint, de telle sorte que plusieurs tests puissent s'appuyer sur le même jeu de données (éventuellement modifié légèrement dans la phase de préparation du test pour les besoins propres de ce test).
- Privilégier les tests écrits dans un langage par rapport à des outils d'enregistrement : ces derniers outils ne peuvent être mis en œuvre que lorsque le code est suffisamment abouti, et ne sont donc pas disponibles pour les tests unitaires du développeur.
- Dans la mesure du possible, utiliser un test fonctionnel pour faire un test unitaire ou d'intégration, pour des raisons évidentes d'économie.
- Porter l'effort sur les fonctions les plus importantes du logiciel, et les plus visibles des utilisateurs. Automatiser les tests sur une fonction qui sera en pratique utilisée par une petite poignée d'utilisateurs une ou deux fois pas an est probablement peu rentable.
- Concevoir le test de manière à ce qu'il soit peu sensible aux évolutions ultérieures du code adjacent au code testé. Par exemple, dans un écran, si le test clique sur un bouton identifié à partir de son emplacement sur l'écran, la moindre modification sur cet écran, comme l'ajout d'un bouton, risque de nécessiter une mise à niveau du test qui pourtant ne concerne pas le bouton ajouté.

Les 10 commandements du développeur productif

N°1
La plate-forme parfaitement tu connaîtras
La connaissance du domaine dans lequel on développe et de la plate-forme sous-jacente est le premier facteur de productivité. Lors de nos audits, nous trouvons fréquemment des développements de fonctions qui existent déjà dans.NET, ou la réécriture inappropriée de services techniques déjà présents dans la plate-forme.

N°2
Des composants tu rechercheras et utiliseras
Dans le même ordre d'idée, il faut veiller en permanence à rechercher l'existence de solutions au problème que l'on cible. Il faut toujours inventer ce qui n'existe pas, après avoir vérifié l'absence de composants répondant au besoin ou de " patterns " de code déjà disponibles. Si l'on trouve des composants répondant à son besoin, c'est presque toujours une meilleure solution économique de les acheter.

N°3
Des outils de génie logiciel tu profiteras
Sans prêcher seulement pour notre paroisse, il est bien évident que les efforts de R&D déployés par les éditeurs d'outils de génie logiciel sont importants et fortement focalisés sur l'automatisation des tâches répétitives. Se priver de ces investissements, c'est prendre le risque d'en essuyer le coût, et de ne le mesurer qu'après avoir fortement avancé dans son cycle de développement.

N°4
Ton code comme une API bien nommée tu concevras
Le " quick and dirty " est une expression malheureuse, car une mauvaise écriture fait perdre plus de temps au final. " Dirty " n'est donc jamais " quick ". A l'inverse, une vraie conception orientée-composant fait gagner beaucoup de temps. En concevant bien ses objets, on doit pouvoir se poser en client de son propre code. Il faut faire comme si on devait vendre son code comme un composant utilisable en tant que tel. Il est conseillé pour cela de soigner le nommage de ses classes, méthodes et propriétés. Une bonne astuce est de regarder dans MSDN.

N°5
Du code testable tu écriras
Il est crucial d'écrire du code qui puisse être facilement testé, et le plus possible de manière automatisée. L'implémentation de tests unitaires, nominaux et aux limites, prépare les tests ultérieurs de non régression en phase de maintenance. Il est également souhaitable, afin d'éviter des découvertes tardives, de se mettre en situation d'infrastructure réelle, en évitant d'être administrateur de son poste et en croisant langue du système et paramètres régionaux. Les tests d'interface utilisateur doivent être le plus réduits possible car ils sont longs et plus coûteux à automatiser. Ils doivent donc ne servir qu'en fin de chaîne d'intégration, une fois le code fonctionnel bien testé.

N°6
D'une machine performante tu t'équiperas
Les outils de développement à jour technologiquement sont consommateurs de mémoire. Il est souhaitable de disposer d'une machine puissante, avec un disque dur rapide et beaucoup de mémoire vive. Une machine 64 bits est souhaitable lorsque cela est possible, ainsi qu'un grand écran.

N°7
Ton environnement de travail tu optimiseras
Choisir une police adaptée dans l'outil de développement, organiser ses répertoires de manière à retrouver ses fichiers très vite, créer les raccourcis qui pointent vers les outils utilisés quotidiennement et configurer les options appropriées de son outil de développement (avec des raccourcis bien choisis, par exemple) sont autant de petits gestes qui rapportent beaucoup sur la durée.

N°8
Toute tâche répétitive systématiquement tu automatiseras
Il faut profiter de son statut de développeur pour réaliser ses propres outils. L'informaticien partage avec le forgeron l'avantage de pouvoir réaliser des outils pour sa propre productivité. Sans tomber dans le piège de passer son temps à faire des outils, il faut systématiquement 'scripter' toute tâche que l'on est amené à reproduire au moins 3 fois.

N°9
Le recours au débogage tu limiteras
Bien que ce point soit controversé, nous pensons que le recours systématique au débogage fait souvent perdre plus de temps qu'en gagner. Lorsque nous entrons dans une pièce où travaille une équipe de développement, nous observons qu'en moyenne un développeur sur deux a le " débogueur " ouvert. Or, quelqu'un qui débogue ne produit pas, ne capitalise rien et ne travaille que pour lui. Nous préférons recommander le recours aux tests unitaires et à la trace.

N°10
En permanence tu capitaliseras
Il faut capitaliser en permanence. Les développeurs les plus performants sont ceux qui n'ont de cesse d'accumuler leur savoir, à la fois en connaissance de la plate-forme, en consolidation des " patterns " utiles et en capitalisation de leur propre code, qu'ils écriront naturellement de plus en plus comme des composants réutilisables. Avec cette approche, on devient chaque jour plus productif.

■ Daniel **COHEN-ZARDI**
Président de SoftFluent et de la commission R&D de l'AFDEL

Avis d'expert

Productivité, agilité, industrialisation : même combat ?

Quel constructeur automobile fabrique ses véhicules de façon artisanale dans des ateliers peu robotisés avec du personnel d'excellence ? Ferrari, Rolls-Royce… Peut-on reproduire un tel schéma en développement logiciel ? Oui, mais à condition de vendre ces applications au prix des Ferrari et de disposer de programmeurs hors pair.

Pour les applications communes et les " simples " développeurs, le besoin d'usines à logiciel est aujourd'hui crucial pour rationaliser le cycle de développement, établir les gisements potentiels de productivité et plus généralement accroître la maintenabilité, la réutilisabilité et la fiabilité des logiciels. Un outil comme BLU AGE incarne cette usine qui prône : le modèle = l'application, et l'application = le modèle. Un changement radical dans la manière de penser le développement. On pense modèle et non code. Et dans cet esprit, on parlera donc d'IME (Integrated Modeling Environment) et non d'IDE. La modélisation se substitue donc à la programmation pure. Une hérésie ?

La productivité est en jeu

Livrer une application dans un temps limité n'est pas " être productif " si le coût d'adaptation face aux utilisateurs fait exploser les budgets. Du point de vue financier, développer un logiciel n'existe pas. Seul l'acte de " maintenir un logiciel " existe ; les facteurs discriminants sont donc agilité et industrialisation. Tout programmeur sait que ce qu'il code est amené à changer dès que les utilisateurs prennent possession de l'application. Il faut donc des environnements de production de logiciel réactifs à l'évolution des besoins, à la réparation si le logiciel est défaillant. C'est pour cela que le IME facilite la maintenance des modèles et celle-ci est moins coûteuse que celle d'un code car celui-ci est dépendant de la technologie, des évolutions des outils, etc.

Réutiliser, c'est évidemment être productif, surtout quand le code repris a été intensément testé et donc prouvé fiable. Réinventer la roue reste souvent le quotidien du programmeur. L'éclectisme du code résultant des cultures et sensibilités diverses et variées des programmeurs se paye aussi comptant au moment du calcul des prix de revient des applications. Comment faire pour qu'un programmeur réutilise le code d'un autre ? Comment faire pour qu'un programmeur maintienne sans effort conséquent le code d'un autre ? Comment faire pour qu'un programmeur, retouchant celui d'un autre, n'introduise pas, par simple incompréhension, des bogues ? Réponse : le transformer en créateur de modèles et générer le code à partir de ses modèles !

Reste l'organisation de tout cela, c'est-à-dire par quelles méthodes agiles exprimer les besoins applicatifs qui par transformation conduisent à du code ? MDA (Model-Driven Architecture) propose de distinguer modèles métier (Platform-Independent Models) et modèles technologiques (Platform-Specific Models) tous deux écrits dans un langage graphique favorisant le travail collaboratif (entre concepteurs et utilisateurs), la communication, la traçabilité. Selon l'adage qui dit qu'un dessin (un modèle UML) vaut mieux qu'un long discours (le code généré), maintenir un logiciel c'est raisonner sur des modèles UML sans présumer d'une technologie spécifique. Les composants techniques sont réutilisés au gré des plates-formes et technologies (.NET, ASP.NET, Java EE, Struts…) retenues pour déployer l'application finale. En revenant à la métaphore de l'industrie automobile, les modèles technologiques correspondent aux grands composants techniques (châssis, moteur…) avec leur plan précis, clair et détaillé de fabrication et d'utilisation. Les modèles métier correspondent eux aux lignes d'assemblage final où il faut pouvoir, au plus tard et de façon personnalisée, changer le tissu des sièges, ajouter ou supprimer une option, un agrément sur le véhicule… cela, sans avoir à reconsidérer la façon dont ce dernier, dans son ensemble, a été bâti.

En fait, la productivité grâce au MDA n'est plus aujourd'hui une simple déclaration d'intention. Des industriels, la grande distribution, des sociétés de service, des opérateurs télécoms… disposent aujourd'hui d'applications intégrées à leur système d'information pour lesquelles toutes les lignes de code ont été générées. Un outil tel que BLU AGE rend totalement pérennes les modèles qui représentent ces applications et instrumente de bout en bout leur gestion : industrialisation et agilité, des concepts à la réalité.

■ Franck **Barbier** - *Netfective Professeur à l'Université de PAU Auteur de "UML 2 et MDE - Ingénierie des modèles avec études de cas" (Dunod) Conseiller technique BLU AGE*

DES EXPÉRIENCES INTERFACES UTILISATEUR QUI INSPIRENT

Introducing...

NetAdvantage® .NET ASP.NET, WinForms, WPF, Silverlight

NetAdvantage® win client Windows Forms, WPF

NetAdvantage® web client ASP.NET, Silverlight

NetAdvantage® Select ASP.NET, Windows Forms

NetAdvantage® Data Visualization Silverlight

NetAdvantage® ICONS Office Basics

Pour vos applications .NET actuelles… et futures

En tant que développeur, votre défi est de concevoir des applications en utilisant les plates-formes les plus matures et robustes disponibles maintenant, Windows Forms ou ASP.NET, tout en planifiant les nouvelles technologies émergentes telles que WPF et Silverlight™. C'est la raison pour laquelle nous alignons nos produits selon vos besoins.

Dans un monde en constant changement, vous pouvez toujours faire confiance aux valeurs durables de NetAdvantage : la puissance et la qualité de ses composants UI ; la documentation efficace ; les tutoriaux instructifs ; l'assistance technique fiable ; et des services de consultation récompensés afin de permettre aux développeurs de livrer à leurs utilisateurs une meilleure expérience dans vos applications d'aujourd'hui et celles de demain.

Pout tout renseignement:
infragistics.com/thefutureisnow

Appelez dès aujourd'hui

N° Vert 0800 667 307

Infragistics®

sales-europe@infragistics.com

Google révolutionne le développement

Dans ce dossier, nous avons voulu nous focaliser sur les dernières API proposées par Google et qui restent parfois obscures pour le développeur et encore plus pour l'utilisateur : App Engine, Gears, la mobilité avec Android, Protocol Buffers, et bien entendu, l'incontournable GWT

A regarder la trentaine d'API disponibles, on peut dire qu'il y a l'embarras du choix, et dans tous les domaines : agenda partagé, messagerie, document, application web, cartographie, vidéo, données, mobilité, etc. Pourtant, il y a un sérieux manque de cohérence dans tout cela, avec l'impression sur certaines API que les techniciens Google parlent aux seuls techniciens et omettent de nombreuses fonctionnalités. Ou laissent passer des failles grossières comme ce fut le cas avec Chrome.

Mais alors, où veut aller Google ? Comme nous l'avons dit avec Chrome, une des facettes de Google est de constituer une plate-forme complète pour créer des applications riches internet de type RIA tout en proposant son propre conteneur : Chrome, avec les API adéquates. Sauf que le support est différent sur Android... Mais l'éditeur veut aussi se positionner dans le nuage informatique, le fameux Cloud Computing en proposant sa propre plate-forme et un hébergement sur ses machines. Mais si Google complète ses API, il n'en va pas de même pour les outils qui restent le parent pauvre de l'offre. Il y en a bien parfois quelques uns, mais ils demeurent très rudimentaires.

A quand un véritable Google IDE ? Et il faudrait aussi que des API comme App Engine étendent rapidement le support des langages.

Nous reviendrons dans les prochains numéros sur les fonctions avancées de GWT (qui réserve aux développeurs une impressionnante richesse fonctionnelle), Gears, App Engine, Chrome, les API YouTube, etc.

■ François **Tonic**

Introduction à Google Web Toolkit

Google Web Toolkit est la boîte à outil Java proposée par Google pour simplifier le développement des applications Web 2.0. Le coeur du framework est son compilateur qui transforme du code Java en JavaScript optimisé. Le compilateur GWT permet également de s'abstraire d'un bon nombre de problèmes de compatibilité des navigateurs.

La compilation du code Java en JavaScript s'appuie sur une émulation des classes de la JRE. Cependant, toutes les classes ne peuvent pas être retranscrites en JavaScript. Par conséquent, GWT souffre de quelques limitations. Par exemple, la classe BigDecimal n'est pas émulée et ne peut pas être utilisée dans le code qui sera interprété par le compilateur GWT. Pour les veinards qui vont se mettre à GWT après avoir lu cet article, sachez que depuis la version 1.5, les génériques, annotations, énumérations et autres nouveautés du langage Java liées à la version 1.5 (de Java) sont enfin supportés ! Les composants graphiques de GWT sont les Widgets et les Panels. Les panels sont des conteneurs de Widget auxquels sont associés des layout spécifiques. Une critique récurrente du framework est liée à la pauvreté des Widgets et Panels livrés d'office avec GWT. Cette faiblesse initiale est fortement compensée par des projets annexes comme l'incubateur GWT et les nombreux projets de la communauté. [Fig.1].

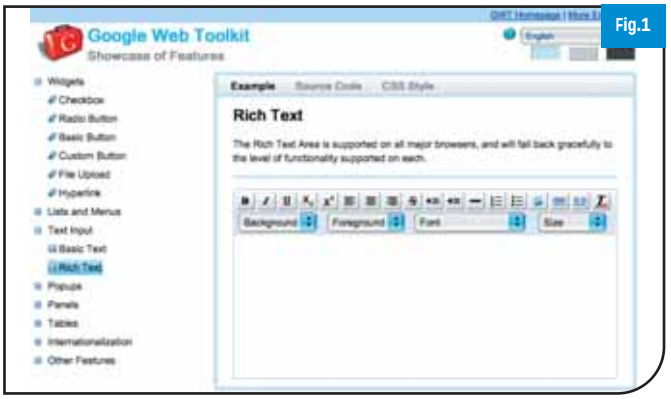


Fig.1

La puissance de GWT

Java met à la disposition des développeurs GWT une vaste palette d'outils. Le passage d'une technologie AJAX plus traditionnelle à GWT n'implique pas de jeter tout votre code JavaScript. GWT inclut un module, nommé JSNI pour JavaScript Native Interface, qui permet d'intégrer facilement des librairies JavaScript ou du code JavaScript natif dans votre code Java. Ce module gère également la communication entre le code Java et JavaScript. Pour faciliter le développement, GWT inclut un navigateur qui permet de visualiser rapidement, c'est-à-dire sans compilation, les modifications que vous apportez à votre projet. Ce navigateur est appelé *Hosted Mode Browser.* Le moteur de rendu de ce navigateur dépend de la plate-forme de développement : Internet Explorer pour Windows, Safari pour Mac OS et Firefox pour Linux.

La communication entre le client et un serveur Java est facile à mettre en place avec le module RPC de GWT. Ce module s'occupe des problèmes de sérialisation et de l'invocation de la méthode sur le serveur. L'utilisation de GWT-RPC impose l'utilisation de servlets Java sur le backend. Dans le cas où l'utilisation de servlets n'est pas possible ou souhaitée, GWT est outillé pour travailler directement avec des requêtes HTTP et propose des classes qui simplifient le traitement de données JSON ou XML. Pour l'internationalisation ou I18N, GWT fournit une technique robuste qui repose sur des interfaces, des fichiers de propriétés et la fonctionnalité " deferred binding " du compilateur GWT.

Le " deferred binding " est une fonctionnalité très puissante du compilateur GWT qui consiste à générer plusieurs versions du code à la compilation. Les fichiers générés sont optimisés pour un navigateur donné et prennent en compte l'internationalisation, par exemple " Firefox en anglais ", " Firefox en français ", " Opéra en anglais ", etc. Le résultat est de minimiser le temps de chargement à l'exécution. GWT gère la fonction d'historisation des navigateurs c'est-à-dire les boutons *précédent, suivant* et les signets. Parce qu'une application de qualité est une application bien testée, GWT intègre le framework de test JUnit. Depuis la version 1.5, GWT prend également en compte la problématique d'accessibilité.

DÉBUTER AVEC GWT

GWT se présente avec des générateurs qui permettent de rentrer très vite dans la technologie : projectCreator, applicationCreator, junitCreator, i18nCreator. La communauté GWT propose d'autres outils. La société Instantiations propose le plug-in GWT Designer pour Eclipse (payant) qui est la référence du moment en matière d'outil de développement GWT. A noter, l'équipe GWT à annoncé la sortie (prochaine) d'un plug-in Eclipse de leur cru.

Du code, du code !

Dans cet exemple, *projectName* est remplacé par *Test, entryPointName* par *com.google.gwt.example.client.TestViewer, moduleName* par *com.google.gwt.example.TestViewer, testName* par *com.google.gwt.example.client.MyFirstGwtTest* et *interfaceName* par *com.google.gwt.example.client.MyFistConstants.* Par convention, le package " client " contient le code source Java qui sera transformé en JavaScript par le compilateur GWT.

Hello World !

Pour commencer, utilisons projectCreator et applicationCreator. Dans une ligne de commande, exécutez les lignes suivantes :

```
GWT_HOME/projectCreator -eclipse projectName -out project
Directory
GWT_HOME/applicationCreator -eclipse projectName -out project
Directory entryPointName
```

La première ligne génère le squelette d'un projet GWT pour Eclipse tandis que la seconde génère une application basique qui servira de base au développement de notre exemple. [Fig.2].

Addition de deux champs

Rendons l'application un peu plus utile en lui faisant faire l'addition de deux champs de saisie. Pour cela, il suffit de modifier la classe *TestViewer.java* comme suit :

```
public class TestViewer implements EntryPoint {

  TextBox firstInput, secondInput;
  Button additionButton;
  Label resultLabel;

  /**
    * Point d'entrée de l'application (comparable à main).
    */
  public void onModuleLoad() {
    // Instantiations des Widgets.
    firstInput = new TextBox();
    secondInput = new TextBox();
    additionButton = new Button("=");
    resultLabel = new Label();

    // Instantiation du Panel.
   HorizontalPanel horizontalPanel = new HorizontalPanel();

    // Ajout des Widgets au Panel.
    horizontalPanel.add(firstInput);
    horizontalPanel.add(new Label("+"));
    horizontalPanel.add(secondInput);
    horizontalPanel.add(additionButton);
    horizontalPanel.add(resultLabel);

    // Attachement du Panel.
    RootPanel.get().add(horizontalPanel);

    // Gestion du clic sur le bouton.
    additionButton.addClickListener(new ClickListener() {
      public void onClick(Widget sender) {
        resultLabel.setText(addition(firstInput.getText(),
secondInput.getText()));
      }
    });
  }

  /**
    * Effectue l'addition de deux String.
    */
   public String addition(String firstInput, String second
Input) {
    int firstValue = Integer.valueOf(firstInput);
    int secondValue = Integer.valueOf(secondInput);
    return String.valueOf(firstValue + secondValue);
  }
}
```

Internationalisation

Le dernier générateur, i18nCreator, s'utilise comme suit :

```
GWT_HOME/i18nCreator -eclipse projectName -out project
Directory interfaceName
```

Les fichiers créés sont un fichier de propriétés et le script de synchronisation qui génère à son tour l'interface associée. 'internationalisation n'est pas nécessaire dans cette application mais regardons tout de même comment ça marche. Pour commencer, modifions le fichier *MyFirstConstants.properties* :

```
plus: plus
equals: equals
```

L'exécution du script de synchronisation génère l'interface *MyFirstConstants.java* :

```
public interface MyFirstConstants extends Constants {

  @DefaultStringValue("equals")
  String equals();

  @DefaultStringValue("plus")
  String plus();
}
```

Utilisation de l'interface dans *TestViewer.java* :

```
// Instantiation de MyFirstConstants via deferred binding.
MyFirstConstants constants = GWT.create(MyFirstConstants.
class);
...
// Utilisation de MyFirstConstants
additionButton = new Button(constants.equals());
...
horizontalPanel.add(new Label(constants.plus()));
```

Les valeurs du fichier *MyFirstConstants.properties* sont celles utilisées par défaut lorsque la locale n'est pas spécifiée. Définissons maintenant les valeurs associées à la locale " fr ". Pour cela, il faut ajouter la ligne suivante dans le module GWT de notre application, soit *TestViewer.gwt.xml :*

```
<extend-property name="locale" values="fr" />
```

Puis il reste à créer le fichier *MyConstants_fr.properties* au même endroit que *MyConstants.properties* et avec le contenu suivant :

```
plus: plus
equals: égal
```

Désormais, en ajoutant le paramètre *?locale=fr* à l'URL de notre application, les textes doivent correspondre aux valeurs du fichier *MyConstants_fr.properties*.

Mon premier service RPC

Pour compliquer un peu l'exemple, effectuons le calcul sur le serveur en utilisant le module GWT-RPC. Pour comprendre le travail à réaliser, regardons le schéma suivant : [Fig.3]. L'interface de base :

```
public interface MyFirstRPC extends RemoteService {
  public String addition(String firstInput, String secondInput);
}
```

L'implémentation :

```
public class MyFirstRPCImpl extends RemoteServiceServlet
 implements MyFirstRPC {
```

```
  public String addition(String firstInput, String secondInput) {
    int firstValue = Integer.valueOf(firstInput);
    int secondValue = Integer.valueOf(secondInput);
    return String.valueOf(firstValue + secondValue);
  }
}
```

L'interface asynchrone :

```
public interface MyFirstRPCAsync {
  public void addition(String firstInput, String secondInput,
AsyncCallback<String> callback);
}
```

Appel au service dans *TestViewer.java* :

```
// Gestion du clic sur le bouton.
additionButton.addClickListener(new ClickListener() {
  public void onClick(Widget sender) {
    // Référence vers le proxy.
    MyFirstRPCAsync service = (MyFirstRPCAsync) GWT.create
(MyFirstRPC.class);
    // Spécification de l'URL.
    ServiceDefTarget target = (ServiceDefTarget) service;
    target.setServiceEntryPoint(GWT.getModuleBaseURL() + "/
Addition");
    // Création d'un callback.
   AsyncCallback<String> callback = new AsyncCallback<String>
() {
      public void onFailure(Throwable caught) {
        resultLabel.setText(caught.getMessage());
      }
      public void onSuccess(String result) {
        resultLabel.setText(result);
      }
    };
    // Appel du service.
    service.addition(firstInput.getText(), secondInput.get
Text(), callback);
  }
});
```

Pour utiliser le service en mode hosted, il faut déclarer le service dans le module GWT :

```
<servlet path="/Addition" class="com.google.gwt.example.rpc
.MyFirstRPCImpl" />
```

Pour utiliser le service en mode Web, il ne faut pas oublier de déclarer la servlet dans le fichier WEB-INF / web.xml

Déploiement sur Tomcat

La procédure de déploiement de notre application sur un serveur Tomcat est très simple. Pour commencer, il faut créer le répertoire " Test " dans le dossier *webapps* de Tomcat. Puis il suffit de respecter l'arborescence : [Fig.4]
Le dossier *com.google.gwt.example.TestViewer* est généré lors de la compilation du projet Test par GWT. Le contenu du fichier *web.xml* est le suivant :

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
  <servlet>
    <servlet-name>Addition</servlet-name>
    <servlet-class>com.google.gwt.example.rpc.MyFirstRPCImpl
</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>Addition</servlet-name>
   <url-pattern>/com.google.gwt.example.TestViewer/Addition</
url-pattern>
  </servlet-mapping>
  <welcome-files>
    <welcome-file>index.html</welcome-file>
  </welcome-files>
</web-app>
Et voici le contenu du fichier index.html :
<html>
  <meta http-equiv="refresh" content="0;com.google.gwt.example.
TestViewer/TestViewer.html"/>
</html>
```

Le déploiement via un un *war* n'est guère plus compliqué. Un *war* n'est rien de plus qu'une archive utilisée pour le déploiement sur un serveur d'application. Pour obtenir un *war* à partir du dossier *Test* créé ci-dessus, il suffit de zipper son contenu puis de modifier l'extension du zip obtenu par *war*. Mais il est tout de même mieux de déléguer cette tâche à un outil de *build* comme Maven par exemple. Ceci nous emmène au chapitre suivant.

Aller plus loin

Vous connaissez maintenant les fondamentaux de GWT mais il reste encore beaucoup à découvrir. Par exemple, la communication client-serveur via JSON / XML, l'utilisation de JSNI, les tests unitaires et les problématiques de sécurité. L'univers GWT est très vaste grâce à une communauté très dynamique.

■ Pierre **Mage** , Sfeir

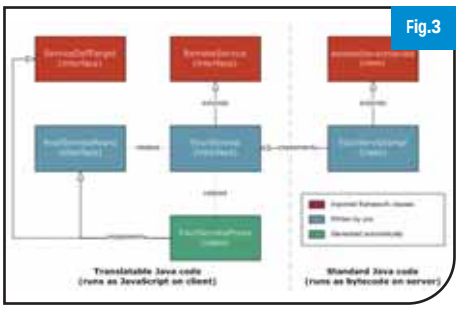


Fig.3

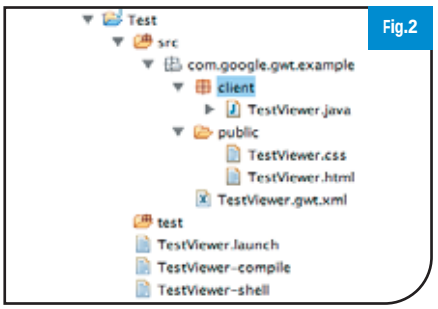


Fig.2

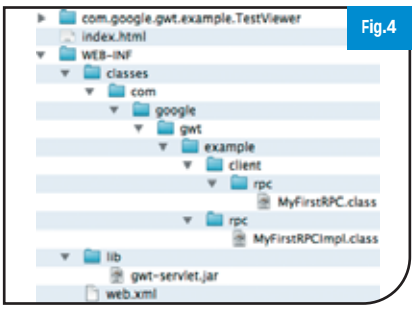


Fig.4

Mettre en oeuvre le Google Protocol Buffers

Protocol Buffers est un mécanisme de sérialisation de données mis au point par Google dans un souci de légèreté, de rapidité et d'économie de bande passante. Nous le découvrons en le mettant en oeuvre avec Java.

Les mécanismes de sérialisation, les formats d'échanges de données sont légion. Sans doute tous ont-ils leur raison d'être et leurs points forts. Et aussi leurs points faibles. Avec son Protocol Buffers, Google a voulu créer pour son usage interne, avant de le diffuser en open source, un mécanisme extensible de sérialisation de données structures indépendant du langage et de la plate-forme, à des fins d'utilisation dans des protocoles de communication, de stockage de données ou autre. Google, pour des raisons que l'on comprend aisément, à mis l'accent sur la légèreté et l'économie de la bande passante. Il n'était sans doute pas question d'utiliser SOAP, plutôt exigeant côté serveur. Google explique n'avoir pas voulu de XML seul, non plus, car celui-ci est trop lourd, trop long et difficile à parser, et peu économe en termes de bande passante. Il est vrai qu'avec XML, le conditionnement des données est souvent plus volumineux que les données elles-mêmes :) Comme tous les mécanismes de ce genre, Protocol Buffers est indépendant du langage … pourvu qu'une implémentation existe pour votre langage préféré. Pour l'instant Google ne propose des implémentations que pour C++, Java et Python, ce qui fait peu, en regard de toutes les implémentations existantes pour JSON par exemple. Bien sûr la communauté open source est réactive et il est fort probable que de nombreuses implémentations verront le jour rapidement. Il en existe déjà une pour le langage Haskell (http://hackage.haskell.org/cgi-bin/hackage-scripts/package/protocol-buffers).

Nous allons nous faire la main avec un langage plus couramment utilisé: Java. Pour nos exemples nous allons nous baser sur le tutorial fourni à http://code.google.com/apis/protocolbuffers/docs/javatutorial.html mais en l'enrichissant. En effet, l'exemple fourni par Google échange des données via un fichier, ce qui est un peu dommage pour un mécanisme voué au réseau. Nous ferons l'échange de données à travers un socket. Enfin, Google met en avant l'existence d'un mécanisme de réflexion et invite, sans donner le moindre exemple, à consulter la documentation pour en savoir plus et le comprendre. Hélas cette documentation est épouvantable, n'explique rien, et est manifestement encore en cours d'écriture au moment de la rédaction de cet article. Mais Programmez! a débroussaillé le terrain pour vous :)

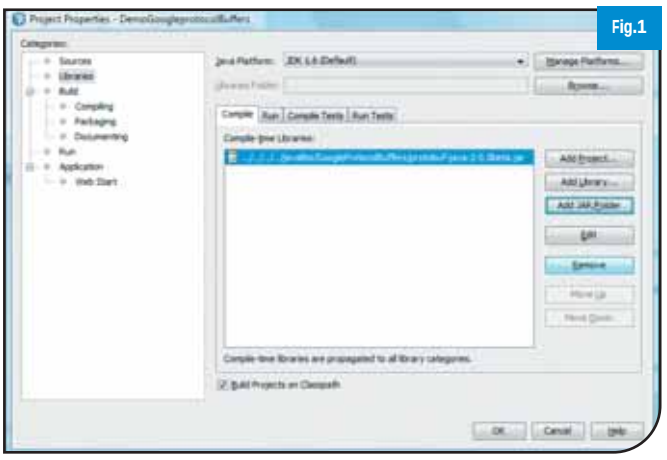


Fig.1

1 INSTALLATION

Une fois l'archive de Protocol Buffers téléchargée à http://code.google.com/p/protobuf/downloads/list, on commencera par compiler … le compilateur avec la séquence de commandes classiques ./configure, make, etc. Ceci si l'on travaille sous un système de type Unix bien sûr. L'opération aboutira à la génération du compilateur *protoc,* requis quel que soit le langage utilisé. Si l'on travaille sous Windows, le plus simple est de télécharger un binaire de *protoc.*

Une fois le compilateur *protoc* prêt, on se rendra dans la branche Java de l'arborescence de l'archive de Protocol Buffers et on suivra les instructions du fichier *Readme.txt,* pour compiler les sources Java et aboutir à la création d'un jar. Pour cela l'installation de *Maven* (http://maven.apache.org/) sera nécessaire. Une fois le jar généré, on veillera bien sûr à ce que celui-ci soit pointé par le CLASSPATH. Nos exemples ont été écrits sous Netbeans 6.1. Sous cet environnement, on ajoutera le jar en tant que librairie dans les propriétés du projet, comme illustré. [Fig.1]

2 STRUCTURATION DES DONNÉES À ÉCHANGER

Nous voulons êtes capables de conditionner des données relatives à un ou plusieurs auteurs de Programmez! accompagnées des données relatives à un ou aux articles que ceux-ci ont écrits. Nous décrivons cela dans un fichier texte baptisé *programmez.proto*, dont le contenu est très intuitif:

```
package programmez;
option java_outer_classname = "ProgrammezGPB";

message Auteur {
 required string nom = 1;
 required int32 id = 2;
 optional string email = 3;
```

```
 enum ArticleType {
   DOSSIER = 0;
   CODE = 1;
  }
  message Article {
 required string titre = 1;
 optional ArticleType type = 2 [default = DOSSIER];
  }
 repeated Article article = 4;
}
message InfosAuteurs {
 repeated Auteur auteur = 1;
}
```

Les deux premières lignes sont, on s'en doute, propres au langage utilisé, ici Java. Si l'on ne donne pas de nom pour la classe englobante, le nom du fichier est utilisé par défaut.
Pour notre exemple cela donnerait une classe *Programmez* au lieu de *ProgrammezGPB.* Cette classe Java contiendra un nombre de choses tout à fait impressionnant, mais avant tout trois classes imbriquées, chacune correspondant à l'un des messages déclarés plus haut. Le message *InfosAuteurs* contient tout ce qui concerne un ou plusieurs auteurs, le "plusieurs" trouvant ici son expression dans le mot-clé "repeated".
On remarque que chaque élément d'un message doit être doté d'un identificateur de rang. Les données propres à un auteur sont empaquetées dans le message *Auteur*. Les deux premiers éléments, nom et id sont requis, l'email est optionnel et enfin un message *Auteur* peut contenir un nombre variable d'articles, dont les données sont à leur tour décrites dans le message *Article.* Le principe est donc extrêmement simple et intuitif.
Une caractéristique intéressante de Protocol Buffers est qu'il est toujours possible a postériori d'ajouter des éléments dans une définition comme celle ci-dessus. Pourvu que l'on ne modifie rien de l'existant, le mécanisme continuera de fonctionner. On pourrait par exemple ajouter ceci dans le message *Auteur* :

```
optional string prenom = 5;
```

3 COMPILATION DE LA STRUCTURE

Nous devons maintenant compiler ce fichier *programmez.proto.* Sous Netbeans, c'est une bonne idée de placer le fichier à la racine de l'arborescence de source du projet. Puis, depuis ce répertoire, on invoquera le compilateur ainsi :

```
protoc -I=. --java_out=. programmez.proto
```

Ceci aboutira à la génération des classes Java et autres stubs dans le fichier *ProgrammezGPB.java* que Netbeans intègrera automatiquement au projet.

4 CONSTRUIRE ET SERVIR UN MESSAGE

Le code ci-dessous est un serveur tout ce qu'il y a de classique, basé sur la classe *Java ServerSocket*. Chaque connexion entrante est traitée dans un thread dédié :

```
package programmez;

import java.io.IOException;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import programmez.ProgrammezGPB.Auteur;
import programmez.ProgrammezGPB.InfosAuteurs;

class SocketThread extends Thread {
 private Socket s;

 SocketThread(Socket s) {
   this.s = s;
 }

 @Override
 public void run() {
   try {
     InfosAuteurs.Builder infosauteurs
       = InfosAuteurs.newBuilder();
     // Définir un auteur et ses articles
     Auteur.Builder auteur = Auteur.newBuilder();
     // id:
     auteur.setId(1);
     // nom:
     auteur.setNom("Tonic");
     // e-mail:
     auteur.setEmail("ftonic@programmez.com");
     // Définir un article
     Auteur.Article.Builder article
       = Auteur.Article.newBuilder();
     article.setTitre("Developper Fun!");
     // On ne spécifie pas le type de l'article, donc
     // le type DOSSIER est attribué par défaut
     // Puis on attribue l'article à l'auteur
     auteur.addArticle(article);
     // On construit l'ensemble Auteur/Articles
     Auteur a = auteur.build();
     // Et on attribue les données au message
     infosauteurs.addAuteur(a);

     // On répète les opérations pour un autre auteur
     // ayant deux articles, dont un de type CODE
     auteur = Auteur.newBuilder();
     auteur.setId(2);
     auteur.setNom("Mazue");
     auteur.setEmail("fmazue@programmez.com");
     // Article 1
     article = Auteur.Article.newBuilder();
     article.setTitre("Le langage Dada");
     auteur.addArticle(article);
     // Article 2
     article = Auteur.Article.newBuilder();
     article.setTitre("DB API Python");
     article.setType(Auteur.ArticleType.CODE);
     auteur.addArticle(article);
```

```
      a = auteur.build();
      infosauteurs.addAuteur(a);
      // Enfin construire le message
      InfosAuteurs ia = infosauteurs.build();
      // envoyer les données
      OutputStream os = s.getOutputStream();
      ia.writeTo(os);
      // puis fermer le socket
      s.close();
      System.out.println("Message transmis");
    }
    catch(IOException ieo) {
      ieo.printStackTrace();
    }
  }
}

public class GPBServeur {

    private void go() throws IOException {
      ServerSocket ss = new ServerSocket(6969);
      while(true) {
        Socket s = ss.accept();
        System.out.println("Connexion entrante");
        new SocketThread(s).start();
    }
  }


  public static void main(String[] args) throws IOException {
    GPBServeur gpbs = new GPBServeur();
      System.out.println(
        "Serveur demarre -- Ctrl-c pour arreter");
    gpbs.go();
  }
}
```

Le point particulier de la construction de messages est que ceux-ci ne sont pas instanciés directement comme on le ferait d'une banale classe Java. Observons, au début de la méthode *run* du code ci-dessus, comment est construite l'instance de la classe *InfosAuteurs*. On commence, par le biais d'une méthode statique de la classe, *newBuilder*, par obtenir une instance d'un constructeur (Builder) de la classe. Le terme constructeur n'est pas à prendre au sens de constructeur de classe, mais plutôt au sens de Factory. Cette fabrique sert à ajouter les messages Auteurs au sein de la classe InfosAuteurs à venir. Quand ceci est fait (cf. la fin du corps de la méthode *run)*, on appelle la méthode *build* du *Builder*, appel qui retourne enfin une instance de la classe *Java InfosAuteurs*. Cette classe se sérialise dans un flux via sa méthode writeTo. Dans notre exemple le flux est celui du socket. Le procédé est le même pour les classes *Auteur* et *Article* imbriquées dans la première. On obtient l'instance d'un *Builder*, puis on appelle la méthode *build* pour avoir l'instance de la classe. Ceci à une exception près: Il n'est pas besoin d'appeler *build* pour la classe *Article*. En effet, celle-ci est déclarée à l'intérieur de la classe *Auteur* dans le fichier .proto. De ce fait, appeler build de la classe *Auteur* construira aussi les instances *d'Article* qu'elle doit contenir. Le lecteur trouvera sur le CD-Rom ou notre site, la classe *GBPClient* pour lire et afficher les données de l'autre côté du socket. Ce code étant globalement la réplique de celui du tutoriel de Google, nous ne le reproduisons pas ici.

5 LE MÉCANISME DE RÉFLEXION

Abordons ce point, plutôt obscur et mal documenté. A la fin de chaque tutoriel sur le site de Google, on trouve un paragraphe "usage avancé" affirmant que si l'on explore la documentation on trouvera le moyen d'écrire du code capable de parcourir un message, sans rien en connaître à l'avance. Sauf erreur ou incompréhension de votre serviteur, ceci n'est pas tout à fait vrai. Il faut au moins connaître le nom de la classe englobante (ProgrammezGPB pour nous) et le nom de la classe constitutive principale des messages (InfosAuteurs pour nous). Moyennant quoi, il est alors vrai qu'il est possible de parcourir des messages, par exemple à la recherche de champs qui y auraient été ajoutés côté serveur et qu'un client pourrait alors découvrir. Mais de l'humble avis de votre serviteur on ne peut pas dire que l'on a là un vrai mécanisme de réflexion car un message sérialisé n'est de toute façon pas entièrement auto-descriptif, ainsi que chacun pourra le constater en examinant en dur les données sérialisées. Cette réserve faite, voici un exemple de client analysant un message à partir des fonctionnalités des classes *Descriptors* et *Message* du runtime de Protocol Buffers :

```
package programmez;

import com.google.protobuf.DescriptorProtos;
import com.google.protobuf.Descriptors;

import com.google.protobuf.Message;
import java.io.IOException;
import java.io.InputStream;
import java.net.Socket;

import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Set;

public class GPBClientReflect {

  public static void main(String[] args) throws IOException {
    Socket s = new Socket("localhost", 6969);
    s.setSoTimeout(5000);
    InputStream is = s.getInputStream();

    Message.Builder mb
      = ProgrammezGPB.InfosAuteurs.newBuilder();
    mb.mergeFrom(is);
    s.close();
    System.out.println("Message recu\n-----------------");

    Map<Descriptors.FieldDescriptor, Object> all_fields
      = mb.getAllFields();
```

```
//Message wrapper = mb.build();

Set<Descriptors.FieldDescriptor> dset = all_fields.keySet();
// Normalement ici il y a un seul enregistrement,
// celui de la classe englobante.
System.out.printf("Il y a %d enregistrement "
  + "enveloppe dans le message\n", dset.size());
// On obtient les informations sur le fichier .proto
Iterator<Descriptors.FieldDescriptor> itdfd
  = dset.iterator();
Descriptors.FieldDescriptor fd = itdfd.next();
Descriptors.FileDescriptor dfd = fd.getFile();
System.out.printf(
  "Le nom du fichier .proto est %s\n", dfd.getName());
System.out.printf("Le package est %s\n", dfd.getPackage());
// On extrait les options du fichier .proto
// pour obtenir le nom de la classe englobante.
DescriptorProtos.FileOptions fo = dfd.getOptions();
System.out.printf("La classe enveloppe %s\n",
  fo.getJavaOuterClassname());

// Obtenir le type de l'enregistrement décrit.
Descriptors.FieldDescriptor.JavaType tp
  = fd.getJavaType();
System.out.printf(
  "Le type de l'enregistrement est %s\n", tp.toString());

// Maintenant on obtient une description des champs
// des messages englobés
Descriptors.Descriptor dd = fd.getMessageType();
List<Descriptors.FieldDescriptor> ls = dd.getFields();
System.out.println(ls.size());
for (Descriptors.FieldDescriptor item : ls) {
  System.out.print(item.getName());
  System.out.println(" " + item.getJavaType());
}

// Maintenant on prend le premier descripteur de la liste
// et on récupère son nom pour afficher
// uniquement cette valeur depuis le message

itdfd = ls.iterator(); // Voir plus haut déclaration itdfd
String pour_recherche = itdfd.next().getName();
System.out.println("Pour recherche: " + pour_recherche);

// Puis on obtient et imprime les valeurs
List msgs = (List) mb.getField(fd);
System.out.printf(
  "Nb messages contenus: %d\n", msgs.size());
for (Object item : msgs) {
  System.out.println(item);
  Message m = (Message) item;
  Map<Descriptors.FieldDescriptor,Object> all_desc
    = m.getAllFields();
  Set<Descriptors.FieldDescriptor> all_desc_set
    = all_desc.keySet();
  for(Descriptors.FieldDescriptor item_bis : all_desc_set)
  {
        if (item_bis.getName().equals(pour_recherche)) {
          Object nom = m.getField(item_bis);
          System.out.println(nom);
        }
      }
    }
  }
}
```

Nous commençons par obtenir une instance d'un Builder de la classe *ProgrammezGPB.InfosAuteurs.* Cette étape est inévitable. Cette classe doit être connue et accessible à ce moment et c'est pourquoi nous ne pouvons pas parler de véritable réflexion. Une fois le Builder obtenu, nous invoquons sa méthode *mergeFrom* qui récupère les données issues du flux du socket client. Ensuite l'analyse peut commencer. La méthode *all_fields* du builder retourne un dictionnaire dont les clés sont des classes *Descriptors.FieldDescriptor,* autrement dit des classes décrivant les champs du message. Les valeurs du dictionnaire sont, sous couvert de la classe cosmique *Object,* les valeurs des champs du message. Au départ notre dictionnaire ne contient qu'une paire, à savoir un message *InfosAuteurs* et son descripteur. Ce dernier permet d'obtenir un *Descriptors.FileDescriptor* donnant des informations sur le nom du fichier .proto, le nom du package et le nom de la classe englobante. Nous pouvons aussi obtenir le type de données conditionnées dans *ProgrammezGPB,* à savoir un objet de type *Message* (sous la forme d'une instance de *InfosAuteurs*). A partir de ce point les choses deviennent plus intéressantes car nous plongeons véritablement dans les données. Pour cela, la démarche est d'abord d'obtenir un *Descriptor* décrivant le type du Message, au sens .proto du terme, contenu dans *InfosAuteurs.* Ce descripteur permet alors d'obtenir les descripteurs des champs des instances du *Message* contenu :

```
Descriptors.Descriptor dd = fd.getMessageType();
List<Descriptors.FieldDescriptor> ls = dd.getFields();
```

Le programme boucle sur les éléments de cette liste pour afficher quelques unes de leurs caractéristiques, puis sélectionne via un itérateur un élément dont la propriété nom servira à filtrer un affichage. Vient ensuite cette ligne :

```
List msgs = (List) mb.getField(fd);
```

fd, sans doute est-il bon de le rappeler, est le descripteur de champ de ce qui est contenu dans *InfosAuteurs,* c'est-à-dire des objets de *type Message.* C'est pourquoi, lorsque nous parcourons ensuite cette liste, nous pouvons transtyper les objets (Object Java) en instance de la classe *Java Message.* Finalement nous pouvons répéter les opérations, pour itérer sur les champs du message et éventuellement n'afficher que ceux dont nous avons sélectionné plus haut le descripteur. Tout ceci est assez ardu, et nous ne partageons pas vraiment l'enthousiasme de Google en ce qui concerne la réflexion de Protocol Buffers. En revanche, dans le cadre d'une utilisation classique, sa légèreté, son côté intuitif et ses performances à l'exécution en font un outil intéressant.

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

LEONARDi

W4-LYRIA : la modélisation au coeur des applications métier

L'ihm

EN TOUTE SIMPLICITÉ !

Pour implémenter des applications riches, accédant à des données hétérogènes (applicatifs métier, BD, et depuis la V4.1, Hibernate, GED et silos documentaires), déployées en Swing, plugin Eclipse ou DHTML/Ajax et s'insérant naturellement dans les processus d'entreprise...

Adoptez la puissance et l'agilité de l'approche Model-Driven

NOUVELLE VERSION LEONARDI V4.1 open source

Concentrez-vous sur votre métier et dotez votre entreprise d'avantages compétitifs durables: amélioration du cycle de vie du logiciel, démarche itérative par prototypage pour coller aux besoins, découplage technologie/métier, évolutivité, agilité et évolutivité, le tout sans expertise technique pointue !

LYRIA

Pour en savoir plus sur la solution Leonardi, rendez-vous sur notre site www.lyria.com ou envoyez-nous un courriel à info@lyria.com

Code source sur www.programmez.com

Localisez vos contacts avec Android

L'application détaillée dans cet article permet de localiser un ensemble de contacts extraits du carnet d'adresses.

Elle met en œuvre les mécanismes introduits par Android pour réaliser une application mobile : gestion des écrans, des données et accès aux fonctions du téléphone. Elle utilise aussi une des possibilités phare de la plate-forme : le contrôle d'une carte " Google Maps " depuis Java.Les applications Android sont composées d'Activity, de BroadcastReceiver, de Service et de ContentProvider :

- *Activity :* l'activité gère une interaction entre l'utilisateur et le téléphone, en pratique c'est un écran.
- *Service :* le service permet d'implémenter des fonctionnalités s'exécutant en arrière-plan sur une longue durée de temps.
- *BroadcastReceiver :* le récepteur de demandes de traitement permet d'effectuer des traitements sans que l'application soit nécessairement active.
- *ContentProvider :* le fournisseur de contenu permet de fournir l'accès à une ressource qui est partagée entre plusieurs applications. La ressource peut être une base SQLite, un ensemble de fichiers, des données sur le réseau …

Ces différents modules s'activent selon divers modèles :

- Un modèle événementiel basé sur des *Intent* qui sont des objets représentant une demande de traitement, c'est ainsi que les *Activity,* les *Service* et les *BroadcastReceiver* sont activés.
- Un modèle procédural semblable à un accès à la base de données pour les *ContentProvider.*
- Un modèle de RPC pour les fonctions exposées à l'intérieur des *Service.*

Les différents composants de l'application: activités, services, receveurs de demande de traitement et fournisseurs de contenu sont déclarés dans un fichier: le *manifeste*. Le *manifeste* précise également les droits de l'application et les interactions de ses composants avec les événements *(Intent)* systèmes ou applicatifs. Pour construire une application mobile, la première étape est de définir les écrans que l'on souhaite mettre en oeuvre. Dans le cas de l'application de localisation, deux écrans sont utilisés :

- Le premier écran présente la carte du monde avec les contacts sélectionnés indiqués par des icônes, un menu pour sélectionner les contacts et un bouton pour demander leur localisation.
- Le second écran présente une liste de contacts sélectionnables et deux boutons pour valider ou non la sélection.

Chacun de ces écrans est implémenté par une *Activity.*

Une activité suit un cycle de vie géré par sept événements, les méthodes suivantes sont à surcharger pour gérer les transitions d'état de l'activité :

- *onCreate :* l'activité est créée.
- *onStart :* l'activité devient ou redevient visible.
- *onRestart :* l'activité reprend après un arrêt.
- *onResume :* l'activité prend / reprend le contrôle des interactions avec l'utilisateur.
- *onPause :* l'activité passe la main à une autre dans les interactions avec l'utilisateur.
- *onStop :* l'activité n'est plus visible.
- *onDestroy :* l'activité s'arrête.

L'écran associé à une activité est décrit dans un fichier XML : le layout. Le layout définit la disposition des vues (View) qui sont les briques de base pour la construction de l'interface utilisateur: bouton, barre de progression… Pour la création d'une liste avec deux boutons: ok et cancel, le layout suivant est décrit :

```
<LinearLayout xmlns:android="http://schemas.android.com/apk
/res/android"
  android:orientation="vertical" android:layout_width="fill_parent"
  android:layout_height="fill_parent">
  <ListView android:id="@android:id/list" android:layout_width
="fill_parent"
    android:layout_height="wrap_content" android:text="@string
/timer" />
  <LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"   android:layout_width=
"fill_parent"
    android:layout_height="wrap_content">
    <Button android:id="@+id/contact_ok" android:text="@string/ok"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />
    <Button android:id="@+id/contact_cancel"
      android:text="@string/cancel"   android:layout_width=
"wrap_content"
      android:layout_height="wrap_content" />
  </LinearLayout>
</LinearLayout>
```

Une gestion de menu peut être associée à l'activité, dans ce cas les méthodes *onCreateOptionsMenu* et *onOptionsItemSelected* de l'activité doivent être surchargées. La création d'un menu pour lancer la gestion des contacts **est disponible sur notre site** www.programmez.com.

La gestion des frappes clavier est également obtenue en surchargeant une méthode: *onKeyDown :*

```
public boolean onKeyDown(int keyCode, KeyEvent event) {
```

```
    if (keyCode == KeyEvent.KEYCODE_S) {
      mMap.setSatellite(!mMap.isSatellite());
      return true;
    }
    return false;
  }
```

L'affichage d'une carte avec Google Maps

Dans le cas de notre application, une activité spécifique doit être utilisée: la *MapActivity qui* gère à la fois le cycle de vie de l'activité mais aussi les interactions avec Google Maps. La *MapActivity* utilise une vue particulière: la *MapView.* Le *layout* de la *MapActivity* doit comporter un objet de type *MapView.* Le *layout* suivant positionne un bouton " localise " en haut de l'écran, la partie basse étant réservée à la carte :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk
/res/android"
    android:layout_width="fill_parent"
    android:orientation="vertical"
    android:layout_height="fill_parent">

<Button android:id="@+id/localise"
android:text="@string/localise"
android:layout_width="fill_parent"
android:layout_height="wrap_content" />

<com.google.android.maps.MapView
        android:id="@+id/map"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:enabled="true"
        android:clickable="true"
        android:apiKey="secret"
        />
</LinearLayout>
```

Ce layout est utilisé dans le onCreate de l'Activity : **code complet sur notre site** www.programmez.com

setContentView(R.layout.main) permet de positionner le layout, R est l'objet créé par le SDK depuis la description textuelle du layout. *FindViewById* permet de récupérer les objets du *layout* par leur identifiant. On utilise le bouton récupéré pour lui positionner un handler déclenchant sur un clic la récupération des localisations. La *MapView* possède un nombre important de méthodes permettant de contrôler la carte. Elle permet d'activer le contrôleur de zoom, de positionner le niveau de zoom et la position du point central sur l'écran :

```
    MapController mapController = mMap.getController();
    mapController.setZoom(6);
    mapController.animateTo(new GeoPoint(48333333, 2333333));
```

La fonctionnalité essentielle pour pouvoir positionner des points sur la carte est fournie par les *Overlay, Overlay* est un calque sur lequel on peut dessiner et qui se superpose à la carte. L'affichage de la localisation des contacts consiste donc, à ajouter un *Overlay* à la vue *MapView :*

```
    List<Overlay> list = mMap.getOverlays();
    list.add(new PositionPoints());
```

C'est ensuite la classe *PositionPoints* qui positionne les différents contacts. Pour cela, la méthode *onDraw d'Overlay* est surchargée :

```
public void draw(Canvas canvas, MapView mapView, boolean shadow) {
 super.draw(canvas, mapView, shadow);
 Projection projection = mapView.getProjection();
 ...
 while (numeros.hasMoreElements()) {
   String numero = numeros.nextElement();
   Position position = mHash.get(numero);
   GeoPoint point = new GeoPoint(
     (int) (position.getLatitude() * 1000000),
     (int)(position.getLongitude() * 1000000));
   Point p = projection.toPixels(point, null);

   canvas.drawBitmap(bitmap, p.x, p.y – bitmap.getHeight(),
 new Paint());

   String nom = ContactsActivity.getNom(numero);
   if (nom != null) {
     Paint paint = new Paint();            paint.setTypeface
(Typeface.create((String)null, Typeface.BOLD));
     paint.setStyle(Paint.Style.FILL_AND_STROKE);
     Rect bounds = new Rect();
     paint.getTextBounds(nom, 0, nom.length(), bounds);
     bounds.left += p.x + bitmap.getWidth() / 3; bounds.right
 += p.x + bitmap.getWidth() / 3;
     bounds.top += p.y – bitmap.getHeight(); bounds.bottom +
= p.y - bitmap.getHeight();
     canvas.save();
     canvas.clipRect(bounds);
     canvas.drawARGB(200, 255, 255, 255);
     canvas.restore();
     canvas.drawText(nom, p.x + bitmap.getWidth() / 3, p.y
       - bitmap.getHeight(), paint);
     }
   }
 }
 }
```

onDraw donne accès à l'objet Canvas qui correspond à l'affichage de la vue, Canvas possède toutes les méthodes nécessaires au dessin. La classe *Projection* fournit la méthode permettant de passer des coordonnées GPS aux coordonnées pixels :

```
projection.toPixels(point, null);
```

La gestion du positionnement sur la carte est ainsi réalisée avec deux classes et moins de 100 lignes de code !

L'accès au répertoire du téléphone

Sur Android, les données peuvent être gérées soit localement par l'application dans une base SQLite ou dans un fichier plat, soit elles peuvent être partagées dans un *ContentProvider* qui, lui-même, peut s'appuyer sur une base locale.

Le carnet d'adresses est une application " système " qui utilise le

mécanisme de ContentProvider pour mettre à disposition ses données aux autres applications. Le *ContentProvider* du carnet d'adresses est accédé par une URI :

```
android.provider.Contacts.People.CONTENT_URI
```

Les données de contacts sont récupérées par une requête query au *ContentProvider* et parcourues par un *Cursor,* semblable au curseur d'accès aux données d'une base :

```
Cursor cursor = getContentResolver().query(People.CONTENT_
URI, null, null, null, null);
startManagingCursor(cursor);
int nomIndex = cursor.getColumnIndexOrThrow(People.NAME);
int numeroIndex = cursor.getColumnIndexOrThrow(People.NUMBER);
cursor.moveToFirst();
do {
 String nom = cursor.getString(nomIndex);
 String numero = PhoneNumberUtils.stripSeparators(cursor.get
String(numeroIndex));
 mHashNew.put(nom, numero);
} while (cursor.moveToNext());
```

L'affichage des données récupérées est facilité par le mécanisme *d'Adapter, l'Adapter* permet de faire le lien entre une vue et des données. Un adapteur (*CursorAdapter*) existe, par exemple, pour faire le lien entre une vue en liste (*ListView*) et les données associées à un curseur (*Cursor*). Dans le cas de la sélection des contacts, la nécessité d'ajouter une case à cocher devant le nom des contacts nécessite de spécialiser un *Adapter,* basé dans ce cas sur l'*ArrayAdapter* : (voir le code complet sur notre site www.programmez.com) L'*ArrayAdapter* fournit les données de la liste avec les valeurs ad hoc grâce à la méthode *getView* qui retourne la *Checkbox* et son état en cours.

Les SMS et la localisation

Le dernier point important de l'application est la récupération de la position GPS des contacts, pour cela il faut bien sûr communiquer avec les téléphones des contacts sélectionnés et avoir déployé sur ceux-ci une petite application " espion ".

Le mécanisme choisi pour la communication est le SMS qui a l'avantage d'être universel sur les réseaux GSM, à l'instar d'une communication TCP / IP entre mobiles qui peut être interdite par le réseau de l'opérateur.

Le principe de l'échange des données de localisation est le suivant :

- Le demandeur envoie un SMS " localise "
- Le récepteur du SMS " localise " retourne un SMS " position " avec ses coordonnées GPS.

Android possède un ensemble complet d'API pour gérer les fonctions téléphoniques: prise d'appel, récupération des informations de signal, accès à la SIM et gestion de SMS. L'envoi des SMS se fait de la façon suivante :

```
SmsManager manager = SmsManager.getDefault();
manager.sendTextMessage(numeros.nextElement(), null, "Localise",
 PendingIntent.getBroadcast(this, 0, null, 0), null);
```

La réception d'un SMS se fait par l'intermédiaire d'un Broadcast Listener, qui reçoit le SMS et active un ServiceProvider pour le traiter, le code de la partie " espion " est sur www.programmez.com.

Le *BroadcastListener* doit être déclaré dans le *manifeste*, le *manifeste* de l'application " espion "est le suivant :

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res
/android"
 package="org.ot.espion">
 <uses-permission
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:name="android.permission.ACCESS_FINE_LOCATION">
 </uses-permission>
   ...
 <uses-permission
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:name="android.permission.WRITE_SMS">
 </uses-permission>
 <application android:icon="@drawable/espion"
   android:label="@string/app_name">
 <service android:name="GPSEspionService"></service>
   <receiver android:name="SMSBroadcastReceiver"
     android:enabled="true">
       <action
       android:name="android.provider.Telephony.SMS_RECEIVED" />
     </intent-filter>
   </receiver>
 </application>
</manifest>
```

Le *manifeste* garantit la discrétion de l'application : elle n'apparaît pas dans le site des applications disponibles et démarre automatiquement quand un SMS est reçu (filtre sur l'action " android.provider.Telephony.SMS_RECEIVED "). Une fois la communication établie, il reste à récupérer la position. Là encore, Android possède une batterie d'API qui permet de contrôler la nature de la localisation et la façon de la récupérer. Dans le cas qui nous intéresse, c'est la méthode la plus simple qui est employée :

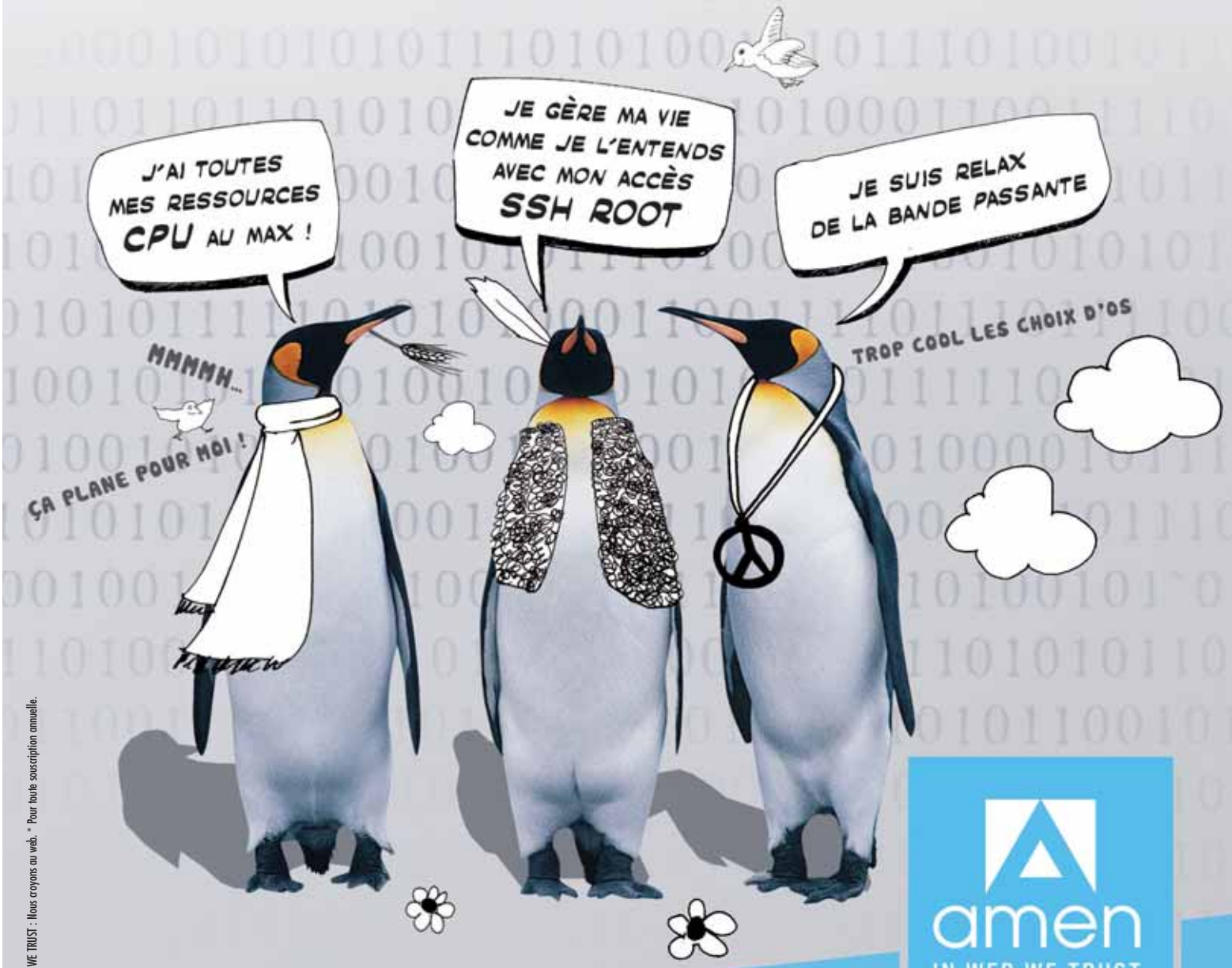
```
   String providerName;
   LocationManager locationManager;
   locationManager = (LocationManager) getSystemService(Context
.LOCATION_SERVICE);
   providerName = locationManager.getBestProvider(new Criteria
(), true);
   Location location = locationManager.getLastKnownLocation
(providerName);
```

Un système conçu pour les développeurs

La mise en oeuvre des différentes API montre la facilité de développer avec Android des applications complexes alliant des fonctionnalités mobiles: SMS, Localisation, carnet d'adresses et des fonctionnalités internet: service de cartographie, instant messaging... En simplifiant l'accès aux ressources du téléphone et à celles d'Internet, avec un langage Java en version 5, Android permet de s'affranchir de la barrière technique pour se concentrer sur le fonctionnel de l'application. Le développeur ne peut être que séduit par l'étalage des fonctionnalités d'Android et confiant dans le succès à venir du " google phone " et de ses applications.

■ Olivier **Théry**

NOUVEAU VDS+ d'AMEN :
le bonheur est dans le serveur !



amen
IN WEB WE TRUST

À PARTIR DE
5 € HT /MOIS
soit 5,98 € TTC/MOIS*

SERVEURS PRIVÉS AMEN : BÉNÉFICIEZ DE RESSOURCES GARANTIES QUI VOUS SONT PROPRES (PROCESSEUR, MÉMOIRE, DISQUE DUR...) TOUT EN PROFITANT D'UNE PLATEFORME INFOGÉRÉE 24H/24 - 7J/7.

- Hébergement multi-sites/multi-domaines
- Interface d'administration : Plesk 8.6
- Systèmes d'exploitation : Fedora Core 8, Suse 10.3, Debian 4.0, Ubuntu 8.04 ou CentOS 5
- Part CPU minimum : de 1 à 6
- Mémoire garantie : de 256 Mo à 1 Go
- Espace disque : de 5 Go à 30 Go
- Bases de données : illimitées
- 1 adresse IP fixe
- Accès Root

PARTENAIRE INDUSTRIEL
entrepreneurs, faites le choix de l'économie numérique
www.econumerique.pme.gouv.fr

Pour plus de renseignements : 0892 55 66 77 (0.34 €/mn) ou www.amen.fr
NOMS DE DOMAINE - EMAIL - HÉBERGEMENT - CRÉATION DE SITE - E-COMMERCE - RÉFÉRENCEMENT

App Engine : le *cloud computing* selon Google !

App Engine permet à n'importe quel développeur de déployer ses applications WEB sur les infrastructures de Google. Il est donc possible qu'une application passe de un à un million d'utilisateurs sans que cette montée en charge n'ait d'impact pour le développeur ou les utilisateurs. Les mécanismes qui permettront cette montée en charge sont transparents. Il n'y a donc pas besoin de modifier son application.

Cette offre se positionne sur le marché du *Cloud Computing*. Il s'agit de l'externalisation de la puissance de calcul dont on délègue la gestion à un prestataire externe. Ce type d'offre présente un certain nombre d'intérêts, parmi lesquels la réduction du coût des infrastructures internes à l'entreprise. C'est aussi l'occasion d'éviter bien des casse-têtes et d'être plus agile, on ne consomme et ne paye que ce que l'on utilise. Google n'est pas le seul à proposer ce genre d'offre, cependant, celle-ci présente deux avantages principaux : simplicité et gratuité.

Google App Engine est simple pour plusieurs raisons :

- Le déploiement d'une application tient en une ligne de commande
- Il n'est plus nécessaire de configurer un quelconque serveur WEB (Apache, Tomcat)
- Des API facilitant le développement sont intégrées.

L'offre d'entrée à ce service est gratuite. Ainsi, un développeur peut disposer gracieusement de 500 mo d'espace pour ses données persistantes et suffisamment de bande passante et de CPU pour supporter 5 millions de pages vues par mois. A ce jour, l'offre payante n'a pas encore été mise en place mais pourrait ressembler à ceci :

- 10 à 12 cents par heure de CPU
- 15 à 18 cents par GB de données par mois
- 11 à 13 cents par GB de bande passante sortante
- 9 à 11 cents par GB de bande passante entrante

Chaque développeur bénéficie d'un quota de dix applications pouvant être déployées. Actuellement, seules les applications WEB Python sont supportées sur Google App Engine. D'autres langages devraient être supportés par la suite (java a été évoqué).

Déployer une première application

La liste des pré-requis au développement et au déploiement d'une application sur cette plate-forme est relativement courte :

- Créer un compte sur Google App Engine
- Télécharger et installer Python ainsi que le SDK

Créer un premier projet n'est ensuite qu'une simple formalité. En effet, un projet Google App Engine tient dans sa plus simple expression en deux fichiers. Un premier qui servira à décrire le projet en question, il s'agit d'un fichier YAML. Un second qui contiendra le code source du projet (Python pour l'instant). Ainsi, un Hello World se présenterait de la manière suivante :

- Un répertoire contenant deux fichiers.
- Un fichier app.yaml décrivant l'application courante :

```
application: helloworld
version: 1
runtime: python
api_version: 1
handlers:
- url: .*
  script: helloworld.py
```

La syntaxe des fichiers YAML se veut plus lisible par l'œil humain qu'une syntaxe de type XML.

Les informations " application ", " version ", " runtime " et " api_version " figurent toujours dans ce fichier. La valeur associée à la clef " application " correspond à l'identifiant de celle-ci. Il est possible de spécifier le langage utilisé pour le projet (" runtime "), cette valeur permettra à l'avenir de faire son choix entre les différents langages supportés. L'API de cette plate-forme évolue constamment. Afin d'assurer la pérennité de votre application en cas de modification majeure de l'API, la version de celle que vous souhaitez utiliser doit être spécifiée (api_version).Le fichier app.yaml permet enfin de préciser les scripts Python qui doivent être exécutés en fonction du pattern d'URL d'appel. Dans l'exemple ci-dessus, l'intégralité des URL conduit vers le code source du fichier helloworld.py (fichier ci-dessous).

```
print 'Content-Type: text/plain'
print ''
print 'Hello, world!'
```

Une fois le développement du projet terminé, une simple ligne de commande permet de le tester en local. Le SDK intègre un script permettant d'émuler l'environnement de déploiement. La ligne de commande est la suivante :

```
dev_appserver.py nomdurépertoire
```

Ici, " nomdurépertoire " correspond au répertoire dans lequel se trouve votre projet. Vous n'avez ensuite plus qu'à constater le résultat au travers de votre navigateur WEB (http://localhost:8080).

Il est temps de déployer l'application sur App Engine. Le déploiement tient également en une ligne de commande, mais une opéra-

tion préalable est nécessaire. Il vous faut en effet créer l'application via la console d'administration de Google App Engine (http://appengine.google.com). Attention l'identifiant d'application saisi sur cette interface doit être reporté dans le fichier app.yaml de votre projet. La ligne de commande pour le déploiement est celle-ci :

```
appcfg.py update nomdurépertoire
```

Tout le monde pourra alors consulter votre application à l'adresse http://nomdevotreapplication.appspot.com.

L'interface d'administration

Pour chaque application déployée, il vous est possible de consulter un certain nombre d'informations les concernant depuis l'interface d'administration. On trouve sur celle-ci des données relatives à l'utilisation de votre application. Un certain nombre d'indicateurs sur le taux d'usage par rapport aux quotas sont également renseignés. Il est également possible de consulter les logs selon différents niveaux de gravité (erreur, warning, information, ...). Un peu à la manière d'un PHPMyAdmin, il vous est possible de consulter les données persistées, de les modifier, les supprimer et même d'en ajouter de nouvelles. Toujours depuis cette interface d'administration, vous pouvez inviter des développeurs à collaborer sur votre projet. Enfin, un historique des différentes versions déployées d'une même application est accessible. Un lien vers chacune des versions permet d'accéder à l'application telle qu'elle était à une version donnée. Il est possible d'en sélectionner une par défaut parmi toutes celles de l'historique.

Les API

Gestion d'utilisateurs

Users API se base sur Google Account pour authentifier les utilisateurs. Le Google Account est le compte utilisateur que vous utilisez pour vous connecter à Gmail ou toute autre application Google. C'est une aubaine pour les développeurs, ceux-ci n'ont plus besoin de gérer leur propre système d'authentification. C'est également une chance pour les utilisateurs qui peuvent ainsi se connecter à plusieurs applications Google App Engine développées par différentes personnes, avec un seul et unique compte utilisateur.

```
from …
class MyHandler(webapp.RequestHandler):
  def get(self):
    user = users.get_current_user()
    if user:
   self.response.headers['Content-Type'] = 'text/plain'
   self.response.out.write('Hello, ' + user.nickname())
    else:
   self.redirect(users.create_login_url(self.request.uri))
…
```

Dans l'extrait de code source ci-dessus, on constate que récupérer l'utilisateur courant est très aisé : *users.get_current_user()*, il est ensuite très simple de consulter les informations relatives à celui-ci. On peut ainsi récupérer son e-mail ou encore son identifiant avec *user.nickname()* et *user.email()*. L'API propose de plus un certain nombre de mécanismes bien pratiques, par exemple, si l'utilisateur ne s'est pas encore authentifié, il est possible de l'envoyer vers un formulaire d'authentification géré par Google. Pour cela, il suffit d'utiliser la fonction *create_login_url()* de l'objet User, le mécanisme inverse existe également pour déconnecter l'utilisateur courant *(create_logout_url())*.

Accès et persistance de données

Google App Engine n'intègre pas de base de données relationnelle, la persistance et l'accès aux données se fait via la Datastore API. Celle-ci repose sur BigTable, il s'agit d'un système de stockage distribué. Les données sont stockées dans une sorte de Map, réparties et multidimensionnelles.

```
class Comment(db.Model):
  content = db.StringProperty(multiline=True)
  date = db.DateTimeProperty(auto_now_add=True)
...
    comments = db.GqlQuery("SELECT * FROM Comment "
                           "ORDER BY date ASC LIMIT 10")
...
    self.response.out.write(comment.content)
...
    comment = Comment()
    comment.content = self.request.get('comment')
    comment.put()…
```

La Datastore API est sans doute la plus complète des API actuellement disponibles sur App Engine. Bien que celle-ci propose un grand nombre de fonctionnalités, il est parfois difficile d'en appréhender le fonctionnement lorsque l'on est un habitué des bases

de données relationnelles. Les objets que l'on persiste sont appelés " Entity ", ils héritent de db.Model. Dans l'exemple ci-dessus, la classe " Comment " est déclarée comme telle. On remarque que celle-ci est constituée d'un attribut de type texte : " content " et d'une date qui est affectée automatiquement à chaque fois que l'objet est persisté. Au travers de cette API, il est possible d'interroger les objets persistés par l'intermédiaire de requêtes dont la syntaxe ressemble à du SQL, il s'agit en fait de GQL (Google Query Language). Persister un objet est également très simple, l'appel à la fonction " put() " sur un objet de type Entity suffit à ce que celui-ci soit sauvegardé. Cette API ne se limite pas aux quelques fonctionnalités présentées ici, je vous invite chaudement à aller consulter la documentation en ligne pour de plus amples informations.
Pour en savoir plus sur le fonctionnement de BigTable, vous pourrez trouver un document à l'adresse : http://labs.google.com/papers/bigtable.html.

Gestion d'un cache mémoire

Les développeurs peuvent utiliser la Memcache API pour stocker en mémoire le résultat de requêtes lourdes ou répétitives.

```
from …
compteur = memcache.get("counter")
if compteur is not None:
   memcache.incr("counter")
else:
   memcache.add("counter", 1)
self.response.out.write(memcache.get("counter"))
…
```

L'API Memcache est également très simple d'utilisation, il est possible d'ajouter, récupérer, incrémenter, … une donnée présente dans le cache. Ces données sont stockées sous la forme de " Clef, Valeur ". L'utilisation de cette API est particulièrement recommandée pour alléger les accès au datastore et ainsi augmenter la capacité de montée en charge de vos applications. Libre à vous bien entendu de le mettre en œuvre dans d'autres usages tels que la création d'un compteur.

Envoi d'un e-mail

L'envoi de mails depuis App Engine relève du jeu d'enfant grâce à la Mail API.

```
mail.send_mail(sender_address, user_address, subject, body)
```

L'exemple ci-dessus se suffit à lui-même pour ce qui est de l'envoi d'un e-mail depuis App Engine. Une ligne de code dans laquelle on précise les informations relatives à l'adresse de l'émetteur, du destinataire, le sujet ainsi que le corps de l'e-mail suffisent. Le nombre d'e-mails qu'il est possible d'envoyer par jour est actuellement limité à 2000.

Communication avec des hôtes distants via HTTP

La URL FETCH API permet d'exécuter des requêtes synchrones au travers du protocole HTTP.

```
from …
 url=http://www.insideit.fr/feed/atom
 result = urlfetch.fetch(url)
 if result.status_code == 200:
    self.response.out.write(result.content)
…
```

Cette API peut être utilisée pour communiquer avec des Web Services ou tout simplement récupérer un flux RSS. Elle se limite aux protocoles HTTP et HTTPS. Comme précisé précédemment, les requêtes s'effectuent de manière synchrone, des précautions sont donc à prendre dans le cas où la communication s'effectuerait avec un hôte au temps de réponse important. En effet, votre application devra attendre une réponse de ce dernier avant de pouvoir continuer son exécution.
Dans l'exemple ci-dessus, on utilise la fonction *fetch* de *urlfetch* pour récupérer le flux RSS du blog *Inside-it*. On vérifie que le code de retour de la requête est 200, c'est-à-dire que tout s'est bien passé. On renvoie ensuite le résultat dans la réponse.

Manipulation d'images

Redimensionner, ajuster, faire tourner des images n'est pas un problème avec l'API de gestion d'images intégrée à App Engine.

```
images.resize(image_data, width, heigth)
```

Encore une fois, la simplicité est de mise, on le voit dans l'exemple ci-dessus avec la fonction de redimensionnement, la manipulation des images ne présente pas de problème particulier.

Un laboratoire à projets

De par la facilité de prise en main de l'environnement de développement et la présence d'API, Google App Engine n'a pas mis beaucoup de temps avant de se voir adopté par une communauté de développeurs. De plus, cette offre, contrairement à d'autres est un KIT tout en un, c'est-à-dire que tous les outils nécessaires à la réalisation d'une application sont réunis sur la plate-forme. Ce n'est donc pas un hasard si aujourd'hui, on trouve déjà de nombreux projets hébergés sur cette plate-forme. Sur Google App Engine, on peut trouver des applications telles qu'un espion de médias sociaux, un générateur de nuage de tags, des DigLikes, un réseau social, …
App Engine apparaît un peu comme un laboratoire à projets, les développeurs peuvent tester facilement l'adhésion à leurs produits. On ne peut donc que souhaiter un bel avenir à cette technologie à laquelle se joindront encore d'autres développeurs avec le support de nouveaux langages.
Pour que sans plus attendre, vous aussi puissiez déployer votre première application App Engine, voici quelques ressources incontournables :

- Blog : http://googleappengine.blogspot.com
- Site officiel : http://code.google.com/appengine
- Cookbook : http://appengine-cookbook.appspot.com
- Liste de diffusion : http://groups.google.com/group/google-appengine
- http://www.appengine.fr

■ Vincent **Bostoen**
Ingénieur Développement chez SFEIR

Déconnectez votre application avec Google Gears

1re partie

Gears est sorti depuis maintenant plus d'un an. Peu d'applications en font aujourd'hui usage et pourtant cela permet d'améliorer considérablement les performances et le confort utilisateur. Voyons de près comment faire en réalisant une application de gestion de vos contacts Google pouvant travailler en mode déconnecté.

Google Gears est une boîte à outils qui s'installe en tant que plug-in dans votre navigateur et est accessible par javascript. Dans cette boîte à outils on trouve :

- un cache de fichiers
- une base sqlite
- un moyen d'accéder à quelques fonctionnalités du desktop
- un émetteur de requêtes http
- de quoi géo-localiser l'utilisateur
- des timers
- un framework pour lancer des tâches en arrière plan

Google Data et Google Contacts sont des API permettant de réaliser des opérations CRUD sur les données stockées chez Google et notamment vos contacts. Les informations échangées entre le client et le serveur sont formatées sous la forme de fils ATOM qui, d'après Google, sont standard et expressifs. ExtJS est une librairie javascript permettant de réaliser des applications AJAX aussi riches et interactives que des applications desktop. Avec GWT, cette librairie s'est imposée comme une référence durant ces derniers mois. Pour réaliser cet article, j'ai simplement pris une application ExtJS existante et lui ai ajouté la fonctionnalité de déconnexion. Cette application est disponible sur le site ExtJS et son URL fournie à la fin de l'article.

Avant toute chose

Il est important de comprendre que seules les applications fonctionnant principalement sur le client (le navigateur) sont éligibles. Si vous avez une application principalement codée sur le serveur et qui renvoie uniquement du 'simple' HTML, alors il vous faudra changer un peu votre fusil d'épaule en codant des parties déportées sur le client et qui utiliseront de l'AJAX. Quant à savoir si la déconnexion de l'application peut se faire toute seule, si c'était le cas, je n'écrirais pas cet article ;) Voyons donc ce que cela implique.

La stratégie

Il faut tout d'abord définir sa stratégie de déconnexion. Il suffit de regarder quelques exemples d'application et voir comment chacun fait. Google Reader par exemple a choisi de faire ce que certains appellent du 'modal', c'est-à-dire que c'est l'utilisateur qui choisit le moment où l'application se déconnecte, permettant à celle-ci de télécharger à ce moment précis tout ce qui lui permet de fonctionner en mode déconnecté (dans le cas de Google Reader, il s'agit de télécharger essentiellement tous les articles des fils RSS). D'autres, comme Remember The Milk, ont choisi de faire du 'modeless' c'est-à-dire que l'application est en permanence synchrone avec le serveur permettant à tout moment de travailler en connecté ou en déconnecté. Mais il faut aussi définir ce qu'on veut permettre de faire offline : est-ce tout ou partie de l'application ? Une fois cela en tête, il faut réaliser pas à pas la déconnexion. Voyons le cheminement sur notre application.

1 RAPATRIER LES RESSOURCES

Pour pouvoir fonctionner sans connexion, il faut d'abord s'assurer que nous avons en local toutes les ressources nécessaires pour l'affichage de la page. Pour cela, nous allons avoir besoin du cache de fichier local de Gears. Celui-ci a simplement besoin d'un fichier *manifest* contenant dans un format spécifique les URL des ressources à conserver en local. Pour savoir facilement quelles sont ces ressources, il suffit tout simplement d'utiliser l'outil *informations de page* de Firefox (outils > informations sur la page) et de regarder les onglets Médias et Liens. [Fig.1]

Pensez également à inclure bien sûr la page HTML elle-même mais aussi et surtout les scripts javascript. Il ne vous reste plus qu'à écrire le fichier *manifest* qui doit ressembler à ceci :

```
{
  "betaManifestVersion": 1,
  "version":"ContactGears_V1.4",
  "entries": [
    {"url":"index.htm"},
    {"url":"lib/ext/resources/images/default/toolbar/bg.gif"},
[...]
    {"url":"GContactProxy.js"},
    {"url":"GearsContactProxy.js"}
  ]
}
```

Certaines options peuvent être positionnées dans le fichier comme par exemple les redirect mais elles ne sont pas nécessaires ici et donc pas couvertes. Il nous faut maintenant tout de même ajouter Gears à notre application. Pour cela il suffit d'inclure le script gears_init.js fourni sur la page 'resources and tools' de Gears. Tant que vous êtes sur le site de Gears prenez également le zip 'samples and tools' qui contient deux fichiers HTML, WebCacheTool et dbQuery qui vont nous être utiles. Mais pour l'heure créez un nouveau fichier javascript et insérez le code suivant dedans :

```
var STORE_NAME = "contacts_offline_docset";
var MANIFEST_FILENAME = "app_manifest.json";
var localServer;
```

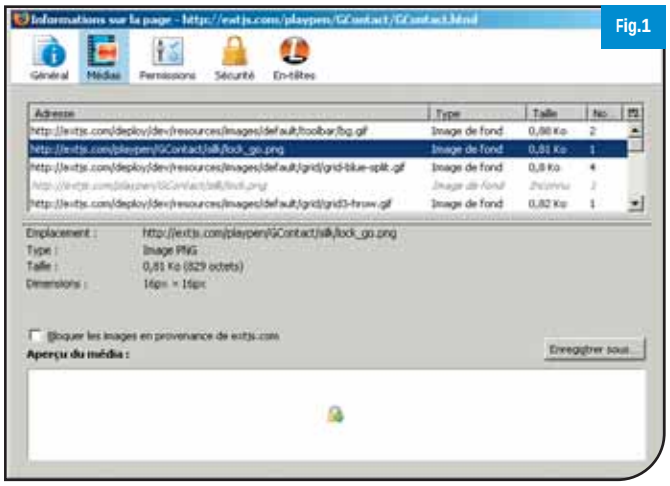


Fig.1

```
var store;

function initGears(){
 if (!window.google || !google.gears) {
   alert("NOTE:  You must install Gears first.");
 }
 else {
   if (!localServer)
     localServer = google.gears.factory.create("beta.localserver");
   if (!store) store = localServer.createManagedStore(STORE_NAME);
   if (!localServer.openManagedStore(STORE_NAME)) {
     store = localServer.createManagedStore(STORE_NAME);
     storeLocalFiles();
   }
 }
}
```

Nous avons créé le serveur local en nous servant de la fabrique de Gears et créé un *managedStore* à partir du serveur local. La fonction *storeLocalFiles* réalise le vrai travail à savoir télécharger les fichiers :

```
store.manifestUrl = MANIFEST_FILENAME;
store.checkForUpdate();

var timerId = window.setInterval(function(){
 if (store.currentVersion) {
   window.clearInterval(timerId);
   alert('Done');
 } else if (store.updateStatus == 3) alert("Error: "+store.
lastErrorMessage);
}, 500);
```


Fig.2

La méthode checkForUpdate permet de vérifier si la version du fichier manifest a changé par rapport à la version locale et si oui, vérifier quels sont les fichiers mis à jour et les télécharger. [Fig.2]

2 MODIFIER L'APPLICATION POUR PRENDRE EN COMPTE LA DÉCONNEXION

Il s'agit ici d'implémenter côté client, en javascript, les fonctionnalités accessibles par l'utilisateur. Cette étape est très rapide pour ce qui nous concerne puisque l'application est entièrement en ExtJS et donc prévue pour entièrement fonctionner côté client, n'utilisant le réseau que pour rafraîchir ses données. Pour des applications n'utilisant pas encore AJAX, cela peut se révéler plus long car il va falloir coder une nouvelle partie de l'application en Javascript et AJAX. Nous avons juste ici ajouté deux boutons à la barre d'outils de l'application, l'un permettant de forcer l'application à passer offline et donc appeler notre fonction initGears, l'autre permet de supprimer le store.

Afin de vérifier la présence des fichiers locaux, vous pouvez aller voir votre système de fichiers (sous Linux l'emplacement est ~/.mozilla/firefox/<profile id>/Google Gears for Firefox) mais il est possible aussi d'utiliser l'outil webCacheTool de Gears que vous pouvez mettre dans un répertoire 'tools' de votre site sur votre serveur. Entrez alors le nom du store (contacts_offline_docset) dans le champ *name* et cliquez sur *OpenManagedStore*. Entrez ensuite dans le champ url l'url de la page HTML qui est censée être rendue par le serveur de fichier de Gears plutôt que par le vrai serveur et cliquez sur *canServeLocally*. Vérifiez que la valeur de retour (tout en bas) est bien 'true'. Vous pouvez aussi cliquer sur *showInfo* pour vérifier la version que vous possédez.

3 GÉRER LES DONNÉES

Notre application passe donc en mode déconnecté mais n'a pas de données à afficher puisque celles-ci ne proviennent pour le moment que du réseau. Il nous faut donc nous créer une copie locale de ces données. Nous allons utiliser ici la base sqlite fournie par Gears.

```
var CONTACT_URL = 'http://www.google.com/m8/feeds/contacts
/default/full';

function updateContactsFromServer(){
 var db = google.gears.factory.create('beta.database');
 db.open('gcontacts');
 db.execute('create table if not exists Contacts (Name text,
 Email text,'+
   'Phone text, selflink text)');
 getAllContacts();
}
```

Ici nous utilisons encore la fabrique de Gears pour obtenir une instance de la classe Database, puis nous ouvrons une base 'gcontacts' qui sera créée si elle n'existe pas. Enfin nous créons une table dans la base et appelons la fonction qui va nous remplir les données :

```
function getAllContacts(){
 var cs = contactService;
 var query = new google.gdata.contacts.ContactQuery("");
 query.setMaxResults(1000);
```

```
  query.setStartIndex(1); // Starting idx (1 not 0)
  cs.getContactFeed(query, storeResults(results), GContact.
errorHandler);
}
```

Nous utilisons ici l'API Contact pour créer une requête paramétrée demandant un très grand nombre de contacts pour être sûr de bien recevoir toutes les données. Puis les données sont insérées en base :

```
function storeResults(result){
  var entries = result.feed.entry;
  for (i = 0; i < entries.length; i++) {
    primaryEmail = processMailAdresses(entries[i]);
    primaryPhone = processPhones(entries[i]);
    db.execute('insert into Contacts values (?, ?, ?, ?)',
      [entries[i].title.getText(), primaryEmail,'+
      'primaryPhone,  entries[i].getSelfLink().getHref()]);
  }
}
```

Le rapatriement des données peut être fait au moment de l'appui sur le bouton qui nous permet de passer offline.
Vérifions que les données sont bien présentes en base. Pour cela, vous pouvez utiliser l'outil *dbQuery* fourni par Gears en uploadant le fichier html sur votre serveur mais je vous recommande l'utilisation de *SqliteStudio* qui est Open Source et compatible Linux et Windows. [Fig.3]
Une fois *SqliteStudio installé*, ajoutez une base en pointant sur le répertoire contenant les fichiers locaux et en ouvrant le fichier *gcontacts#database*.
Pour faire apparaître les résultats sur ExtJS, il vous faudra changer de dataStore. Créez donc un autre data store :

```
var offlineStore = new Ext.data.Store({
        proxy: new Ext.data.GearsContactProxy({
            limit: 15
        }),
        reader: new Ext.data.JsonReader({
            totalProperty: 'total'
        }, [{
            name: 'primaryEmail'
```

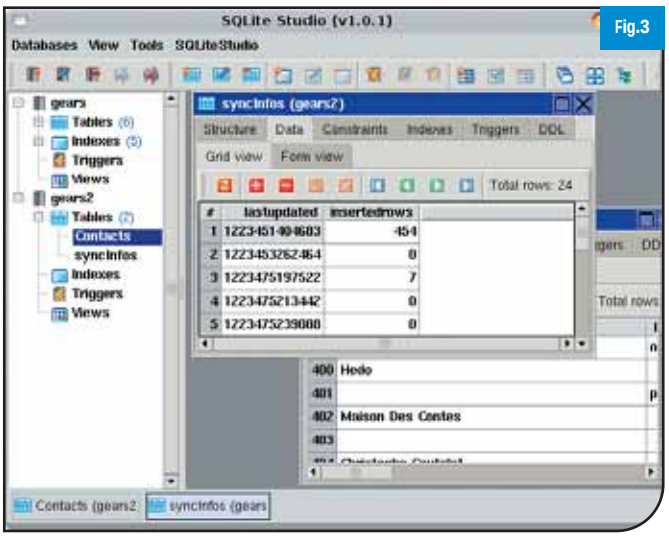


Fig.3

```
        }, {
            name: 'primaryPhone'
        }, {
            name: 'title'
        }]),
        storeId: 'offline'
    });
```

Ce data store s'appuie sur un data proxy qui émet des requêtes sur la base (voir pour cela le fichier *GearsContactProxy.js* dans les sources de l'application). Enfin, il suffit de substituer le store 'déconnecté' au store 'connecté' :

```
var offlineStore = Ext.StoreMgr.lookup('offline');
Ext.getCmp('pager').bind(offlineStore);
Ext.getCmp('gc-grid').reconfigure(offlineStore, columnModel);
offlineStore.load();
```

Notre application peut donc fonctionner sans connexion internet ! Assurons-nous en, en passant le navigateur en mode hors ligne… et en purgeant le cache ! Et naviguons dans les données.
La suite le mois prochain.

Fabrice **Dewasmes**
Blog : http://fdewasmes.free.fr - Fabrice.dewasmes@gmail.com

SALAIRES : Quelles compétences paient le mieux ?

Les métiers de l'informatique ont toujours le vent en poupe. Cela se traduit par des progressions de salaires, d'une année à l'autre, généralement plus fortes que dans les autres métiers. Sauf indication contraire, les salaires sont indiqués en valeur brute par an.

Les salaires des informaticiens sont assez différenciés, mais la plupart se situent dans une fourchette entre 20 k€ et 50 k€ (cf. baromètre Lesjeudis / Harris Interactive, du 1er semestre 2008 - et tableau 1). Le groupe de recrutement Hays a fait une étude de rémunération 2008 sur les secteurs de l'informatique et télécoms, mettant en évidence la dépendance de la rémunération par rapport au métier et à l'expérience (tableaux 2 : Salaires-Hays).

Tableau 1 : *une " photographie " des moyennes de salaires indiqués par les recruteurs sur lesjeudis.com pour quelques fonctions (salaire brut annuel en €)*

Fonction	2 à 5 ans d'expérience	5 à 10 ans d'expérience
Bases de données	46 000	50 000
Etudes/développement	41 400	53 300
Exploitation/production	50 000	60 000
Progiciels (ERP/CRM)	35 000	52 800
Internet/intranet/messagerie	36 500	-
Systèmes/réseaux/télécoms	50 000	-
Méthodes/qualité/process	-	45 000
Direction technique/DSI	40 000	77 500

Des salaires en augmentation générale

L'augmentation des salaires des informaticiens reste supérieure de plusieurs points à la croissance nationale. Ainsi, on a observé des augmentations de + 4,3% en 2005, + 4,1% en 2006, selon Oberthur Consultants. Pour 2007, **Alexandre Xiradakis,** directeur marketing et communication du Groupe Les Jeudis, constate : *" L'augmentation de salaire est supérieure à 10% entre 2006 et 2007 pour des profils très recherchés comme le développement Java, .Net, les experts ERP, Oracle, Business Intelligence ou double compétence, par exemple le développement et l'expertise dans la banque-assurance. Sur les autres profils, l'augmentation est raisonnable. "*

" Depuis 6 mois, nous assistons à une hausse considérable des salaires. En effet, un ingénieur jeune diplômé qui demandait auparavant 33 à 34 k€ souhaite aujourd'hui un salaire brut annuel de 40 k€ ", note **Amina Altiti,** responsable RH chez Sybase. Ce que modère l'étude Hays : *" On aurait pu s'attendre naturellement à une hausse importante des salaires. Cependant les salaires en 2007 ont légèrement augmenté sans s'envoler, malgré l'écart entre l'offre et la demande. "* La variation dépend aussi du poste occupé. Ceux dont les salaires croissent le plus sont les plus pointus, tels les études et développement, l'expertise, la sécurité ou le support technique. Les rémunérations des postes à forte responsabilité, direction ou maîtrise d'ouvrage, continuent à connaître une croissance plus importante que la moyenne. A l'opposé, les postes liés à l'exploitation sont assez peu valorisés, selon Oberthur Consultants. Malgré cette forte progression, la moyenne des salaires des ingénieurs informaticiens, de l'ordre de 50 k€, reste inférieure au salaire médian des ingénieurs de la catégorie *" production et fonctions connexes ",* qui s'élève à 52,6 k€, des postes de direction générale bien entendu (103 k€) et même des fonctions marketing et commerciale (66 k€) et administration et gestion (65 k€), selon la 19e enquête socio-économique conduite

Tableau 2 : *salaires de différents métiers et fonctions selon l'expérience – extrait de l'étude Hays (salaire brut annuel en k€)*

1re partie : Etudes et développement

Fonction	0-2 ans	2-5 ans	5-10 ans	>10 ans
Analyste-programmeur	25-30	30-35	35-40	40-43
Ingénieur développeur	39-35	35-40	40-45	45-50
Ingénieur qualité et méthode	30-33	33-38	38-42	42-45
Ingénieur intégration	30-35	35-40	40-45	40-45
Concepteur / développeur web	22-26	26-30	30-35	35-45
Responsable R&D	-	-	45-50	50-60
Chef de projet études & développement	32-37	37-45	45-50	50-55
Architecte	-	37-45	45-55	55-65
Directeur de projet	-	-	55-65	65-80
Directeur informatique	-	-	60-70	70-85
DSI	-	-	65-75	75-100
Développeur ABAP (SAP)	25-30	30-35	35-40	40-50
Chef de projet ERP	-	35-40	45-55	55-70
Directeur de projet ERP ou décisionnel	-	-	50-60	60-80

par le Conseil national des ingénieurs et scientifiques de France auprès des ingénieurs diplômés des écoles françaises.

Les technologies porteuses

Les technologies les plus demandées sur lesjeudis.com sont SAP, Java / JEE, Réseau, .Net, DBA, PHP. *" Les offres de postes en sécurité, architecture, expertise SAP, conseil métier sont les plus difficiles à pourvoir. Les experts et managers de projets sont également courtisés ",* constate Oberthur. Ce que confirme l'étude Hays : *" Nous constatons un maintien de la pénurie de compétences notamment chez les développeurs Java / JEE et .Net, les consultants ERP et les ingénieurs réseaux et télécoms entre autres. "* Chez Acti, société de conseil et services informatiques, *" les développeurs Java, JEE et .Net sont des profils très demandés ",* indique **Marie-Mercedes Allongue**, directrice marketing et communication de ce groupe. *" Les rémunérations pour ces postes vont de 35 à 55 k€ suivant l'expérience, 1 an à plus de 5 ans. "* C'est pourquoi ce type de poste est fortement valorisé par les salaires. Les réseaux et la sécurité sont aussi des domaines très demandeurs. *" Un ingénieur avec certification a, aujourd'hui, après 2 à 3 ans d'expérience, des prétentions de l'ordre de 40 à 45 k€ selon le secteur d'activité "*, précise **Manuela Mucheche**, chargée de recrutement chez l'intégrateur réseau Telindus. Chez Acti, un administrateur système avec un an d'expérience est payé environ 25 k€, et un ingénieur environ 35 k€.

En ce qui concerne les ERP, notamment SAP, et le décisionnel ou BI (business intelligence), *" les entreprises sont extrêmement demandeuses des différents outils informatiques de gestion ",* relève **Laetitia Langer**, manager Hays. *" La rémunération des candidats est ainsi favorisée à la hausse pour ce secteur. Cette tendance 2007 se poursuivra en 2008. "*Toutefois, même si la demande est plus forte sur certaines technologies, la rémunération n'est pas nécessairement liée à ce critère. *" La grille de rémunération peut être très large et s'étire de 30 k€ à plus de 50 k€. Mais à l'intérieur d'une même fonction, nous valorisons surtout, d'une part, les compétences liées à des savoir-être et, d'autre part, le potentiel qui est notamment lié à la capacité et à l'envie d'apprendre. Cette logique reste indépendante de la technologie ",* souligne **Marine Hamon**, responsable RH chez Netapsys, SSII spécialiste des nouvelles technologies orientées objets. *" En informatique, nous observons souvent l'émergence de bulles technologiques fortement valorisées financièrement mais qui peuvent tout aussi rapidement dégonfler. Nous préférons privilégier une politique de rémunération à long terme sans faire de différence entre les technologies. C'est-à-dire la volonté et la capacité d'apprendre, plus qu'une technologie particulière. " " La technologie .Net est particulièrement prisée, mais ce n'est pas significatif pour la rémunération ",* confirme **Caroline Grangeon**, DRH chez ITS Group, spécialiste des infrastructures IT. Pour cette société, l'expert technique, dont le salaire avoisine les 50 k€, doit avoir *" une culture globale de l'informatique, il peut s'agir d'un ingénieur systèmes et réseaux, qui a été formé et a développé une expertise sur un produit. "*

Un cas particulier : le logiciel libre

Pour le marché de l'open source, *" les fourchettes salariales, notamment sur les populations de développeurs et d'ingénieurs, restent nettement en deçà de celles de leurs homologues [Java / J2EE, technologies Microsoft, C / C++] ",* indique **Edwige Taulin**, manager Hays. Pourquoi cette différence ? *" Une des caractéristiques des développeurs en logiciel libre, c'est leur passion et leur forte motivation ",* répond **Philippe Montargès,** PDG d'Alterway. *" La motivation par le salaire est importante, mais ce n'est pas le seul critère. "* Avis partagé par **Alexandre Zapolsky**, PDG de Linagora : *" Nous sommes une boîte de 'geek', l'objectif est de donner de la profondeur dans les carrières techniques. "* Pour un profil expert et chef de projet, chez Alterway, le salaire est de l'ordre de 40-50 k€ ; pour les ingénieurs d'études et développement, il est un peu en dessous de la moyenne des SSII, ainsi que pour les jeunes diplômés, administrateurs de systèmes et techniciens d'exploitation. *" Malgré cela, la pression est forte en termes de recrutement ",* souligne Philippe Montargès. D'ailleurs, le salaire n'est pas limité pour les postes à forte expertise et grande responsabilité. Ainsi chez Linagora il peut atteindre les 100 k€ pour une personne qui se trouverait à l'intersection de plusieurs aspects : *" Une technologie que peu de gens maîtrisent, un certain niveau d'ancienneté, et la participation à des fonctions d'encadrement de 10 à 15 personnes ",* précise Alexandre Zapolsky. **Cyril Pierre de Geyer**, directeur technique et responsable de l'offre Anaska, organisme de formation du groupe Alterway, admet que le métier de formateur soit moins payé que le conseil : la grille s'étend entre 20 et 30 k€, indépendamment du type de technologie, pour un formateur à plein temps. La plus faible rémunération est largement compensée par d'autres avantages, mis en avant par Cyril Pierre de Geyer : *" Des satisfactions personnelles, pas de contraintes comme les développeurs, une partie du temps axée sur la veille technologique... "* Comme les développeurs en logiciel libre, ce sont des passionnés de technologie.

Tableau 2 : *salaires de différents métiers et fonctions selon l'expérience – extrait de l'étude Hays (salaire brut annuel en k€)*

2e partie : Systèmes, réseaux et télécoms

Fonction	0-2 ans	2-5 ans	5-10 ans	>10 ans
Administrateur systèmes réseaux	24-28	28-32	30-35	35-40
Ingénieur systèmes réseaux	28-32	32-38	38-43	43-48
Analyste d'exploitation	22-25	25-30	30-35	35-40
DBA	30-35	35-45	45-55	50-60
Architecte réseaux et télécoms	-	38-45	45-55	55-70
Ingénieur sécurité	28-32	33-40	40-45	45-55

Marine Hamon
Responsable Ressources Humaines NETAPSYS

" Dans un secteur informatique en perpétuelle évolution, pour paraphraser Montaigne, une tête bien faite est mieux rémunérée qu'une tête bien pleine. "

Caroline Grangeon
Directrice Ressources Humaines - ITS GROUP

" Pour le développement, .Net est particulièrement prisé, mais pas significatif dans la rémunération. "

Une rémunération en fonction de la séniorité...

Les salaires dépendent, la plupart du temps, des années d'expérience. *" La plupart des offres d'emploi mentionnent 2 à 10 années d'expérience maximum. Les entreprises recherchent des personnes opérationnelles rapidement ",* précise Alexandre Xiradakis. Selon lesjeudis.com, la moyenne est de 44 113 € pour 2 à 5 ans d'expérience, et 55 659 € pour 5 à 10 ans d'expérience. Cette dernière fourchette d'expérience semble la plus appréciée, une expérience supérieure à 10 ans étant quasiment inexistante ou du moins non mentionnée. *" Les niveaux de rémunérations tiennent bien sûr compte de la fonction : les nouveaux embauchés débutants entrent Ingénieur développement afin de consolider leurs connaissances des technologies objets, les plus expérimentés occupent des postes de chef de projet technique avec trois grands domaines de compétence : l'architecture, la gestion de projet, le management ",* confirme Marie Hamon (Netapsys).

Egalement avide de profils techniques, Valtech privilégie l'expérience : à partir de 18 mois pour les consultants, et jusqu'à 8-10 ans dans le domaine concerné pour les experts. En effet, la rémunération est d'abord fonction de l'expérience, depuis 35-37 k€ pour 2 à 3 ans d'expérience, jusqu'à plus de 60 k€ pour 8 à 12 ans. *" La rémunération est une synthèse entre savoir-faire technique, potentiel, capacité à être consultant ",* précise **Henri Petitet,** DG de Valtech. *" Elle est en cohérence avec notre perception de la valeur et du potentiel de la personne, et nous sommes capables de l'expliquer. "* Les métiers les plus fortement rémunérés sont – on s'en douterait – ceux à plus forte responsabilité, notamment " architecte ", " directeur de projet ", " maîtrise d'ouvrage " ou " assistance à maîtrise d'ouvrage ", quelle que soit la technologie, où les salaires tournent plutôt autour de 50 k€, pour culminer à 77,5 k€ pour un directeur technique ou un DSI expérimenté (5 à 10 ans). Ainsi, chez ITS Group, l'échelle part de 30-40 k€ pour un administrateur systèmes et réseaux bac+2 expérimenté, et atteint 60-65 k€ pour un directeur de projet. Chez Alterway, il y a trois niveaux de postes à forte expertise : architecte, chef de projet et consultant expert.

Par ailleurs, les chefs de projets ont souvent une double compétence, technique et fonctionnelle, ce qui justifie une rémunération de 48 à 50 k€, notamment pour un projet de BI.

Henri Petitet
Directeur Général VALTECH

" La rémunération variable peut atteindre 30% grâce à un 'menu' de 'primes potentielles' parmi lesquelles le consultant peut choisir. "

... et de la motivation

Si la formation et l'expérience ont leur rôle à jouer, c'est aussi la valeur intrinsèque de l'individu qui est reconnue par la rémunération. C'est en tout cas le point de vue de Valtech qui concrétise l'engagement et la motivation du consultant par une rémunération variable, laquelle peut atteindre 30% du fixe, notamment pour les consultants " très senior ". *" Nous proposons un 'menu' de 'primes potentielles' parmi lesquelles le consultant peut choisir ",* expose Henri Petitet. Dans ce " menu ", on trouve l'activité, la cooptation, le business, la contribution. Par exemple, le collaborateur à l'origine d'une embauche empoche 2000 €. Il peut aussi arrondir ses fins de mois en animant un cours du soir, en aidant à la préparation des Valtech Days, en apportant une nouvelle affaire, etc. *" Tout est transparent et connu à l'avance ",* insiste Henri Petitet, qui concrétise ainsi les valeurs qu'il apprécie chez ses consultants : leur autonomie, leur capacité d'être force de proposition, leur soif de savoir, leur capacité à rechercher des informations... *" Nous ne faisons pas de surenchère sur les salaires. Nous recrutons sur le profil : c'est l'individu qui nous intéresse ",* ajoute-t-il.

Netapsys offre également une rémunération variable, quoique plus faible, à ses développeurs et chefs de projets. *" Nous souhaitons que la rémunération traduise la culture d'entreprise plus que des technologies particulières, et valorise la compétence plus que le savoir ",* commente Marine Hamon. Ainsi, pour un développeur, à un fixe compris entre 30 et 37 k€ s'ajoute une partie variable, de 1 à 3 k€. Pour un chef de projet, le fixe est de 40 à 48 k€, la part variable de 2 à 4 K€. De plus, cette SSII privilégie *" l'intérêt collectif consistant à partager les résultats et associer les collaborateurs à la bonne marche de l'entreprise ",* poursuit Marine Hamon. Chez ITS Group, seul le " top management " a une partie variable.

Chez Alterway, les salaires, plutôt inférieurs à la moyenne des SSII, sont complétés par des modes d'intéressement aux résultats des entreprises du groupe. *" Avec l'intéressement, les niveaux de rémunération sont conformes à ce qui se passe sur le marché ",* indique Philippe Montargès. Chez Linagora, les architectes participent à l'avant-vente et à la production de projet, et à ce titre, ils ont une part de rémunération variable.

Les " premiers jobs "

Les prétentions salariales des jeunes et futurs diplômés se situent majoritairement entre 20 et 30 k€, avec des disparités selon le niveau d'études : majoritairement entre 25 et 30 k€ pour les bac+2 / +3 et de 30 à 35 k€ pour les bac+4 / +5, selon une *" Etude sur les motivations et les attentes des jeunes et futurs diplômés en informatique ",* réalisée par lesjeudis.com et Randstad en septembre 2008. Cette étude place en 2e position le critère de rémunération (60%), juste derrière la possibilité d'évolution de carrière (61%).

Les salaires d'embauche au niveau bac+2 ne varient guère. En revanche, pour les jeunes ayant une qualification élevée (bac+5), les salaires sont en nette augmentation, notamment dans les SSII. *" A la sortie des écoles, les salaires demandés sont largement revus à la hausse ",* observe **Isabelle Willame**, directeur technique chez Readsoft.

Le salaire moyen des jeunes diplômés de Supinfo (promos 2005 à 2007) à la sortie de l'école est de 32,7 k€, généralement en SSII. La plupart des rémunérations se situent entre 30 et 38 k€, elles dépassent exceptionnellement 40 k€ ou, à l'autre extrême, se situent rarement en dessous de 25 k€. Pour les jeunes diplômés de l'Epitech, le salaire moyen dépasse 40 k€. *" Le salaire moyen a beaucoup augmenté ",* confie **Nicolas Sadirac**, directeur de cette école, dont 30%

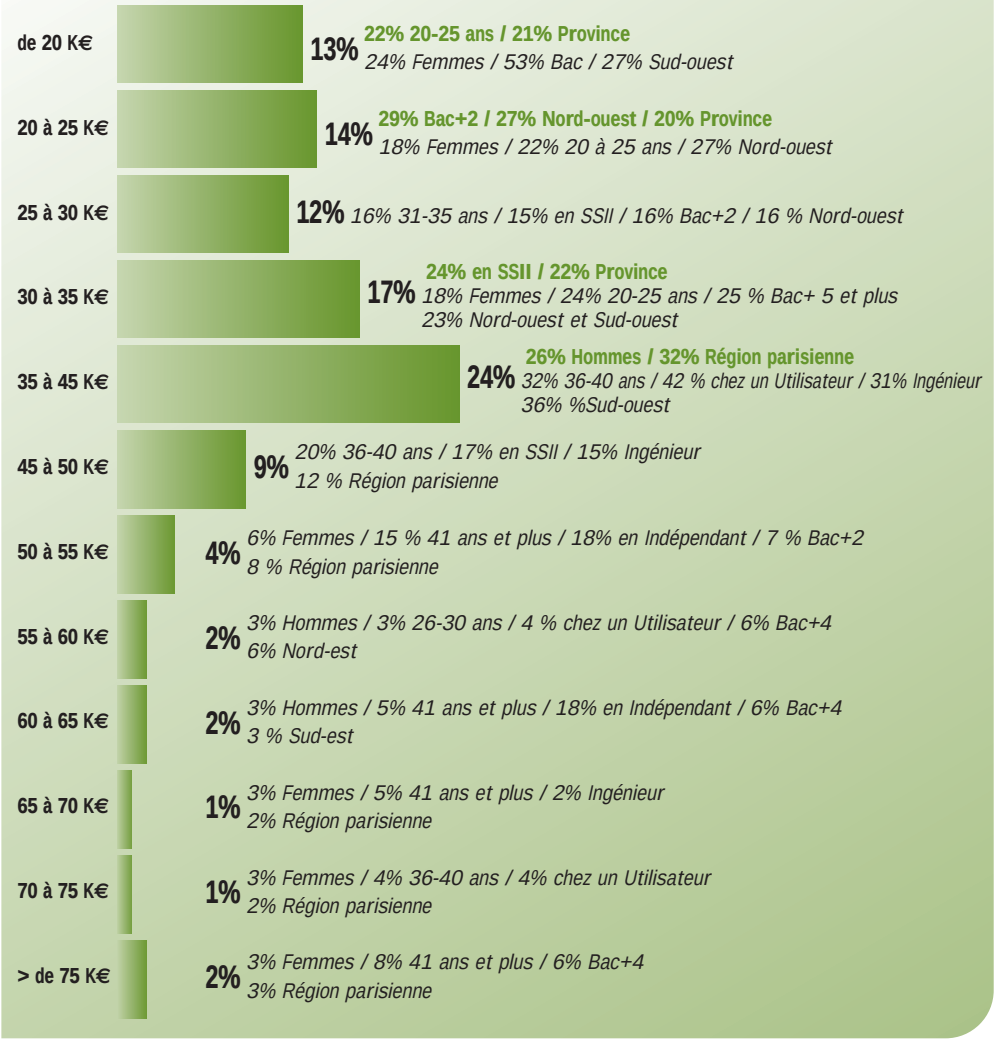
des diplômés vont vers les SSII. L'augmentation de salaire est aussi liée au fait que, depuis 3-4 ans, les jeunes diplômés partent souvent à l'étranger, notamment aux Etats-Unis, au Japon, en Australie, où les rémunérations sont plus élevées. Acti apprécie particulièrement les diplômés de Polytechnique, Centrale ou Mines : *" Pour ceux-ci, le salaire est supérieur à 40 k€ à l'embauche, avec un an d'expérience "*, précise Marie-Mercedes Allongue. Chez Acti, le salaire d'embauche des jeunes diplômés varie de 25 à 36 k€ en fonction du niveau du diplôme (bac+2 à bac+5), des stages effectués et du poste.

Localisation géographique

Il y a souvent des différences considérables entre les salaires d'Ile-de-France et des autres régions. Selon l'étude Oberthur Consultants, elle peut varier entre 14% et près de 20%, selon les secteurs (SSII, éditeurs ou entreprises utilisatrices). Chez Valtech, implanté à Paris et Toulouse, la différence est de 15-20%. *" La population est plus jeune à Toulouse et la concurrence est plus forte qu'à Paris "*, explique Henri Petitet. Selon André Zapolsky, la différence de salaire entre Paris et les régions est d'au moins 20%.

■ Claire **Rémy**

Baromètre Lesjeudis.com + Harris Interactive (bilan du 1er semestre 2008)



Supinfo, une ambition internationale

Plus de 40 ans après sa naissance, Supinfo a su évoluer avec les technologies et le contexte économique. Reconnue comme la première Grande Ecole de l'informatique en France, elle est classée par les entreprises dans le " top 5 " des écoles d'ingénieurs françaises.

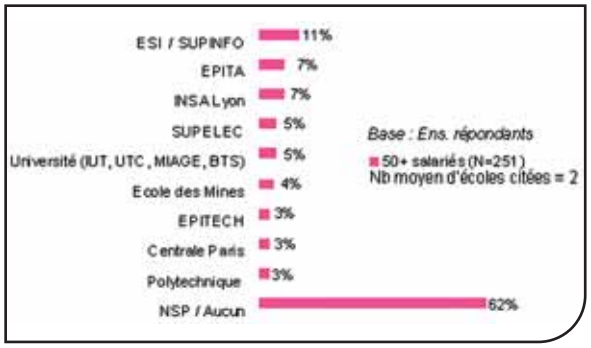
Claire **Rémy**

L'Ecole Supérieure d'Informatique (ESI) est créée en 1965 par un ingénieur Supélec, d'où l'idée de dénommer cette école Supinfo. *" La nouvelle dénomination, plus explicite que le sigle, devient une marque distinctive valable en France et qui passe aussi bien à l'international ",* commente **Alick Mouriesse**, ancien élève de Supinfo et président du groupe depuis 1999. Dédiée aux technologies de l'information, l'école revendique une " vocation professionnalisante ", en formant des concepteurs et praticiens des systèmes d'information immédiatement opérationnels dans les entreprises. D'où un cursus plus pratique que théorique. Dès 1972, date de sa reconnaissance par l'Etat, l'école conclut un accord avec IBM qui installe une ancienne machine dans l'école et donne des cours aux étudiants. Car le constructeur ne doute pas du profit qu'il pourra tirer de compétences sur ses systèmes.

Le nombre d'élèves augmente progressivement, avec une accélération à la fin des années 80 pour culminer en 1990-1991 à près de 1000 élèves. Mais vers 1993, c'est la chute rapide des effectifs, liée à deux phénomènes : mauvaise conjoncture économique et crise de l'enseignement supérieur privé. Supinfo est connoté " grands comptes à la française ", avec leurs équipements grands systèmes IBM, Bull GCOS 7, Cobol... Pendant ce temps, d'autres technologies ont pris le relais : C, C++, objets, Unix / linux, etc. En 1998-1999, Supinfo ne compte plus que 50 étudiants, alors que l'Epita, la concurrente plus " sexy ", créée au début de la décennie, en a déjà 1000. L'existence de Supinfo se pose alors en termes de survie. Fin 1997, Alick Mouriesse décide, avec d'autres anciens, de reprendre l'école sous forme d'une association dont il sera le principal investisseur. *" Nous décidons de moderniser l'école, tout en respectant un certain nombre de règles pour assurer une continuité de l'établissement reconnu par l'Etat ",* raconte-t-il. *" Le dossier est accepté et nous avons pu conserver les labels. "*

> *Supinfo pourrait devenir le leader mondial de la formation IT*

Enquête TNS Sofres 2007
Notoriété spontanée
des grandes écoles spécialisées en informatique.



La stratégie de rupture

L'école renaît ainsi de ses cendres, mais il faut tout reconstruire : nouveaux locaux, nouveaux programmes, nouveaux professeurs surtout, puisque la plupart des anciens sont partis et que ceux qui restent acceptent mal de remettre en cause leur enseignement traditionnel (Une usine à profs). L'esprit reste le même, former des cadres opérationnels, mais à jour. C'est l'époque où Microsoft décide de lancer en Europe son programme de collaboration avec les grandes écoles,

" Compétences 2000 ". Alors que les universités françaises ne jurent que par le libre, Supinfo accueille à bras ouverts le programme de Microsoft. C'est une voie toute tracée pour faire redémarrer l'école : *" Nous redémarrons avec eux pour la formation et la préparation de certifications Microsoft ",* indique le président (cf. encadré : Les laboratoires). A la même époque, l'école déménage de la Porte de Montreuil à Château-Landon. Dès mars 1999, elle peut sortir les nouvelles brochures Supinfo, qui affichent sur la couverture les partenariats avec non seulement Microsoft, mais aussi Oracle et Cisco. *" Le succès est phénoménal, chaque année les effectifs grossissent ",* se félicite Alick Mouriesse. Si bien qu'en 2007, 1600 nouveaux élèves intègrent l'école et, en 2008, le nombre de nouveaux dépasse 2350. Au total, Supinfo affiche plus de 6000 élèves en formation initiale et voit sa notoriété bien reconnue (cf. figure).

A partir de 2001, l'école se démulti-

La nouvelle appellation internationale de l'école

Supinfo à Bordeaux

Supinfo à San Francisco

plie : Alick Mouriesse lance la première Supinfo hors de Paris, en Martinique, son pays d'origine. Le test s'avère concluant et d'autres écoles voient le jour : Strasbourg et la Réunion en 2002 ; puis Bordeaux, Rennes, Montpellier, Lille, Marseille, etc. Les dernières ouvertes sont Metz, Reims et la Guadeloupe en 2008. Hors de France, des écoles sont ouvertes au Canada, en Chine, au Maroc, à Londres ; et la dernière-née à San Francisco. Aujourd'hui, Supinfo, c'est 33 écoles en tout, dont 25 en France métropolitaine et DOM, avec un rythme d'ouverture d'environ 7 écoles par an. *" Plutôt que de faire une grosse école à Paris, nous avons voulu créer des petites écoles avec le même état d'esprit ",* explique Alick Mouriesse. Même si Paris continue à grossir, cet essaimage donne aux étudiants la possibilité de mobilité, ce qui est hautement apprécié par nombre d'entre eux. Le réseau d'écoles porte désormais le nom de " Supinfo International Universities ", dont Supinfo fait partie, sans pour autant impacter la structure juridique de l'école en France. A la formation initiale diplômante, Supinfo ajoute un volet de formation professionnelle et de formation au management.

Pour Alick Mouriesse, l'expansion doit se faire désormais à marches forcées : 5 campus en chantier au Maroc, à la suite des accords entre gouvernements français et marocain et déclinaison mondiale de la marque " Supinfo ".Ce nouveau visage de leader en puissance, c'est aussi ce nouveau siège, un hôtel particulier à deux pas des Champs-Elysées, et cette nouvelle identité : " Supinfo, international University "

Site : www.supinfo.com ■

Une "usine à profs"

"Comment sauver Supinfo ?" Au lendemain de la grande implosion des années 1990, il s'agit de recruter des enseignants et des élèves, car seuls quelques enseignants restaient, et très peu d'élèves. La direction de l'école met en place " le système Supinfo ". Celui-ci est organisé en 3 secteurs – fondamental scientifique ; technologie ; matières cadres (management) – et 4 types de savoirs – savoir théorique ; savoir-faire ; savoir être ; savoir transmettre (faire savoir). Ce chantier pédagogique considérable commencé en 1999-2000 n'est toujours pas achevé.

Ce système remet en cause la légitimité des professeurs. Pour ce qui concerne le savoir théorique, pas de souci ; le savoir-faire pratique, en revanche, est beaucoup plus difficile à obtenir ; quant au savoir être comportemental, il implique une évaluation, ce que refusent généralement les professeurs en place ; enfin, le savoir transmettre est un véritable choc.

Le programme est modifié chaque année à partir des préconisations résultant d'une grande enquête. Il en est ressorti que les enseignants n'étaient pas les mieux placés dans ces différents secteurs et savoirs. En particulier, le " savoir transmettre " (pédagogie), aucun professeur ne l'avait appris. La logique de formation des professeurs était

absente. D'où l'idée de créer une " Teachers Factory ", devenue " SCT Academy " (Supinfo Certified Trainer Academy). Elle part de l'idée que le savoir-faire et le savoir transmettre peuvent être assurés par des étudiants évalués et formés à la pédagogie. Chaque année, certains étudiants sont ainsi sélectionnés, et l'école fait en sorte, qu'à terme, ce soit un passage obligé pour tous les étudiants d'apprendre à transmettre dans le cadre du programme. Tous les SCT sont formés à la pédagogie, ils apprennent à gérer un étudiant récalcitrant, une classe bruyante, etc. Le SCT fait partie des études.

Si ce dispositif a permis d'abord de pallier le déficit en professeurs, il a bien d'autres vertus, en particulier les étudiants acquièrent ainsi plus de responsabilité, plus de maturité, plus de professionnalisme. Il permet de " booster " des étudiants et de gommer les écarts générationnels. ■

Les laboratoires, une ouverture sur l'industrie

Le 1er laboratoire créé au sein de Supinfo est le laboratoire Microsoft. C'est le plus connu et le plus populaire. Par la suite, d'autres labos se sont créés, et Supinfo dispose aujourd'hui de 7 autres labos : Cisco, .Net, Apple, Sun, Oracle, Linux, IBM, Mandriva. L'idée de ces laboratoires est de vulgariser le savoir et de trouver les meilleurs méthodes. *" Une technologie, un labo pédagogique "*, résume Mauricio Diaz Orlich, directeur des laboratoires Supinfo. Les labos sont une communauté, ce n'est pas un endroit physique. Ils rassemblent des personnes passionnées, disponibles, engagées. On y prépare les cours et les supports de cours. Il y a quelques centaines de membres par labo. " Ces membres devront faire plus que les autres étudiants ; pour remplir leur mission, ils ont accès à des informations techniques pointues ; le partenaire peut leur apporter un soutien, organiser des événements. " Certains laboratoires sont devenus des sites de référence pour les partenaires industriels.

Sur le CD du numéro !

Silverlight 2.0 : du .Net et du dynamisme

De nombreuses critiques avaient fusé sur les limitations fonctionnelles et architecturales de Silverlight 1.0. Et la confusion régnait, car Microsoft avait lancé des pré-versions en même temps que la v1. Entre temps rebaptisée v2, l'ex Silverlight 1.1 était ce qu'aurait du être la v1 : support de .Net, Scripting, etc. Disponible depuis la mi-octobre, revenons sur cet événement RIA de cette fin d'année.

Guillaume **André** (Wygwam),
en collaboration avec François **Tonic** (Programmez)

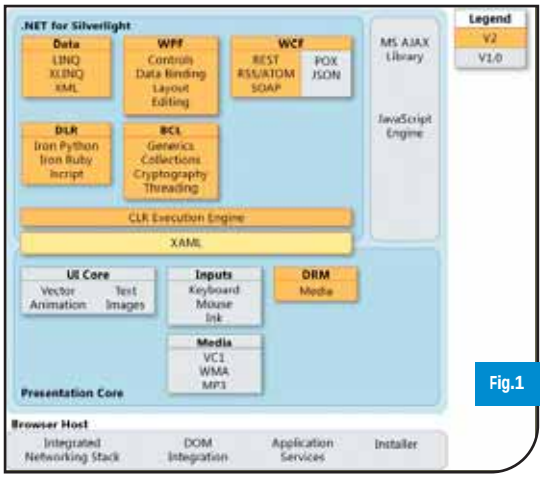


Fig.1

Pour rappel, Silverlight 2.0 est une plate-forme multi-système, multi-navigateur de Microsoft servant à développer, déployer et utiliser des applications riches web ou RIA selon la terminologie. Pour l'utilisateur, il prend la forme d'un plug-in pour Internet Explorer, Safari et Firefox principalement. Il fonctionne sur Windows et MacOS X. Pour Linux, il faut passer par le projet Open Source, Moonlight qui sera prochainement compatible Silverlight 2. Détail important : ce plug-in est totalement autonome et ne nécessite en aucun cas de framework .NET installé sur la machine pour pouvoir visualiser une application Silverlight.

Un concentré de nouvelles fonctions pour le développeur

Silverlight 2 offre la possibilité aux développeurs de développer des applications dans de nombreux langages tels que VB.NET, C#, JavaScript, IronPython, IronRuby. Silverlight expose un nombre important et intéressant de fonctionnalités facilitant le développement d'applications RIA. Ainsi, n'importe quel développeur habitué des langages cités ci-dessus et donc de la plate-forme .NET est capable de réaliser ce type d'application avec les mêmes méthodes de travail. Travaillant avec le format XAML pour la partie rendu des interfaces, Silverlight expose aussi son propre framework .NET pour la partie programmation. Ainsi les contrôles, les fonctionnalités de réseaux, la gestion des fichiers, la mise en page et l'application de thèmes sont tout à fait envisageables et sont natifs à Silverlight. Microsoft conseille de désinstaller toutes les anciennes v2 en pré-version avant d'installer la version finale. Important : cette v2 supporte deux modèles de développement : managé (.Net) et javascript.

Une architecture complète ! [Fig.1]

Comme le montre le schéma, l'architecture est bien plus complète. La v2 rajoute le support de .Net grâce à l'implémentation d'une CLR et surtout d'une DLR pour les langages dynamiques (Javascript, Ruby, Python). C'est une des grosses révolutions de cette version. On voit aussi que l'arrivée de .Net permet d'avoir accès aux API et librairies de données (LINQ, XML), à WPF pour l'interface, à WCF pour la communication ou encore la gestion des fameux DRM.

On remarque aussi que l'on dispose d'un sous-ensemble .net complet. Donc on peut développer sans problème en C# ou VB.Net. Et cette version introduit aussi plusieurs grosses nou-

Compatibilité WPF

Si l'arrivée de WPF est un élément important, on ne dispose pas d'un support total. On peut transférer des projets WPF entre Silverlight et le desktop, mais il faudra tout de même se méfier et considérer les absences.

Par exemple, on ne dispose pas du support des commanding de WPF et actuellement les ressources dynamiques en WPF ne sont pas disponibles. Toutes les ressources XAML sont statiques. D'autre part, plusieurs fonctions WPF se voient limitées (fonctionellement parlant) dans Silverlight 2.0, on citera notamment :

- data binding : pas de disponibilité du binding direct vers les données XML, de l'UI vers un binding UI
- limitation du support des triggers
- styles : pas de support de BasedOn, ni de TargetType

Enfin, plusieurs aspects WPF sont présents mais avec une implémentation différente dans Silverlight (ex. : rendu subpixel). Si vous voulez porter en Silverlight 2 une interface WPF, vérifiez la compatibilité et modifiez le code en conséquence. Vous disposez aussi de fonctions Silverlight 2.0 non présentes dans WPF comme DeepZoom, le contrôle data grid.

Les composants de présentation (Core Presentation)

Input	Accès aux périphériques matériels comme le clavier, la souris, etc.
Rendu User Interface	Rendus vectoriels et bitmap, animation, texte
Média	Lecture, écriture des fichiers audio – vidéo (différents formats)
Contrôles	Développer et personnaliser les contrôles
Layout	Positionnement dynamique des éléments
Data binding	Assurer le lien dynamique entre les objets données et les éléments UI
DRM	Disponibilité des mécanismes de gestion des droits numériques
XAML	Parser pour utiliser XAML

veautés de développements que le programmeur appréciera :

- isolated Storage : accès sécurisé au stockage local, au cache de données
- développement asynchrone
- gestionnaire de fichiers : disponibilité (en mode sécurisé) d'un dialogue d'ouverture de fichier, pour charger un fichier local
- HTML / code managé : manipulation en HTML DOM des éléments d'interface. Ainsi, il est possible en javascript d'appeler directement un objet managé.
- Sérialisation : support de la sérialisation (types CLR) vers JSON et XML
- Packages : création de packages xap contenant l'application et l'entrée servant au plug-in Silverlight du navigateur pour exécuter l'application

L'un des intérêts de Silverlight est de n'avoir pas spécialement de modules serveurs à installer, hormis pour les sources de données le nécessitant. Ainsi, si vous utilisez ASP.Net Ajax, il ne faudra pas oublier les contrôles MediaPlayer Server et ASP.Net Silverlight Server. On peut utiliser IIS et Apache.

Les outils [Fig.2]

Bien entendu, Microsoft a mis à niveau les outils de développement pour tenir compte de Silverlight 2. Pour le développeur il existe différents outils, outils payants et gratuits. Voici un panorama rapide des gammes.

Visual Studio

Nous retrouvons l'incontournable Visual Studio 2008. Cette fois-ci en version Service Pack 1, il est accompagné de son plug-in " Silverlight Tools for Visual Studio " (http://go.microsoft.com/fwlink/?LinkId=129043) qui permet, entre autres, de créer des projets Silverlight avec les templates appropriés directement dans votre Visual Studio. Comme je vous l'ai énoncé ci-dessus, Microsoft propose aussi une solution totalement gratuite. Celle-ci est identifiée par le logiciel Visual Studio Express et le plug-in " Silverlight Tools for Visual Studio ". Ce logiciel est disponible à l'adresse suivante : http://msdn.microsoft.com/fr-fr/express/de fault.aspx

Plug-in Eclipse (eclipse2sl)

La réelle nouveauté ! La possibilité de coder son application Silverlight directement sous Eclipse (uniquement en version Windows pour le moment) ! Le projet, encore en développement est particulièrement prometteur. Deux modèles de développement s'offrent alors aux développeurs : un modèle *full Eclipse* avec installation du Framework, des runtime et SDK Silverlight 2, et un modèle *mixte* (Blend, VS 2008, eclipse4sl). [Fig.3] et [Fig.4]

Pour approfondir la partie Silverlight – Java, plusieurs exemples très parlants sur le blog de Stève : http://blogs.msdn.com/silverlight%5Fplus% 5Fjava/

Site : http://www.eclipse4sl.org/

Les utilitaires [Fig.5]

Avec cette nouvelle version de Silverlight ont été mis à jour certains logiciels et certains SDK qui étaient dédiés à cette technologie. Ainsi un logiciel tel que Deepzoom qui permet de réaliser des applications à base de " zoom infinis " http://memorabilia.hard-drock.com/en est un exemple. Vous retrouverez la version courante de ce logiciel à cette adresse : http://www.microsoft.com/downloads/details.aspx?FamilyID=457B17B7-52BF-4BDA-87A3-FA8A4673F8BF&displaylang=en

Le SDK dédié aux langages dynamiques supportés par Silverlight a lui aussi été mis à jour et est à présent téléchargeable sur : CodePlex (http://www.codeplex.com/sdlsdk/Release/ProjectReleases.aspx?ReleaseId=14254)

Outils designer et intégrateurs

La gamme Expression se met elle aussi à niveau pour devenir compatible avec Siverlight 2.0. L'outil de prédilection sera Blend 2.0. Mais attention, il faut impérativement installer le Service Pack 1 pour supporter Silverlight 2.0. Le SP1 correspond en réalité à la version 2.5 de Blend. Les autres outils seront ou sont déjà mis à jour en SP1.

Moonlight ?

Il faut attendre que les développeurs Mono mettent à niveau la pile open source Silverlight (projet Moonlight). Cela arrivera sans doute dans les prochaines semaines, même si aucune date n'est précisée. A suivre de près !

Quelques nouveautés

Qui dit nouvelle version dit aussi nouveautés. Les modifications entre la version 1 de Silverlight et la version 2 sont colossales. Voici quelques points à retenir.

Qu'en est-il de la compatibilité ?

Silverlight hérite de la logique de compatibilité descendante. Ainsi, toutes les applications déjà réalisées et développées et présentes sur la toile resteront compatibles avec le nouveau plug-in. Bien entendu les développeurs ayant réalisé des applications avec les versions de transition telle que la béta doivent suivre la notice d'instruction pour rendre leurs applications compatibles avec la version finale. Cette compatibilité descendante s'applique aussi sur les applications Silverlight 1.0. Pour récapituler, une application développée en Silverlight 1.0 sera compatible avec la nouvelle version du plug-in 2.0.

Fig.2

Fig.5

Fig.3

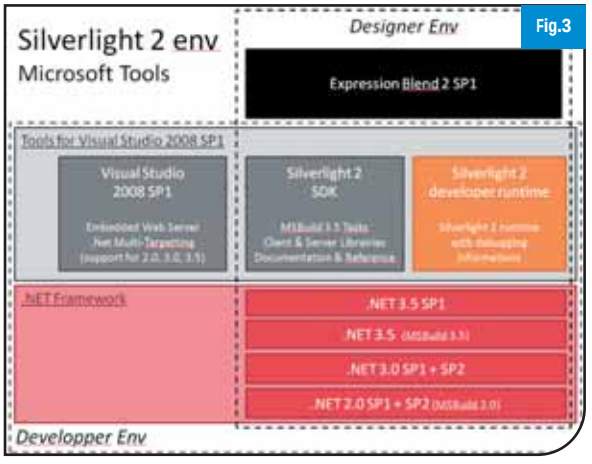
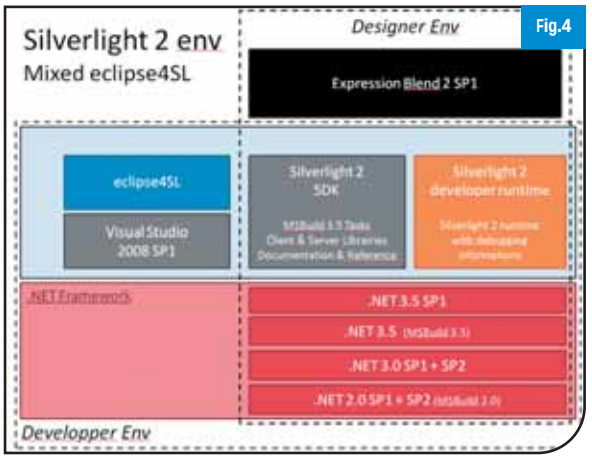


Fig.4



Un nouveau look [Fig.6]

Microsoft a profité de cette nouvelle version pour modifier le rendu par défaut des différents contrôles compris dans le plug-in. Microsoft a visiblement opté pour un design neutre et pro à la fois. <IMG> Pour les graphistes nous relèverons aussi que certaines modifications de rendu ont été apportées au niveau des contrôles de layout, ainsi nous aurons moins de cas ou l'interface ressemblera à un buvard. Le cas SnapePixel est donc en partie corrigé.

De nouveaux contrôles

Deux nouveaux contrôles indispensables voient le jour avec ce nouveau plug-in. Ils sont intitulés *ComboBox* et *ProgressBar*. Tout comme les autres contrôles natifs de Silverlight ces deux là sont totalement personnalisables, tant au niveau graphique qu'au niveau des données. Ainsi vous pourrez faire des Styles, des Templates, du Databinding et du DataTemplate avec deux contrôles.

De nouvelles fonctionnalités

ClipBoard

La réussite d'une application RIA relève bien entendu de sa fonctionnalité initiale mais aussi en grande partie de son ergonomie. Ainsi, avec le support du ClipBoard, Microsoft Silverlight met un pied de plus dans l'intégration du plug-in au sein de l'environnement de l'utilisateur. En effet, avec Silverlight vous allez permettre à vos utilisateurs de réaliser, entre autres, des copier / coller au sein même de votre application et de l'environnement de l'utilisateur. Ceci est réalisable en partie grâce à la classe *HtmlPage* qui expose différentes méthodes de communication avec le DOM. D'un point de vue programmation voici comment cela fonctionne :

```
window.clipboardData.setData('Text'
, 'Votre texte') ;
window.clipboardData.getData('text');
window.clipboardData.clearData();
```

FontManager et Blend 2 SP1

Cette fois-ci, plus orienté graphiste et designer, le FontManager va permettre de gérer très facilement l'ajout et l'import de nouvelles *Font* au sein de vos projets Silverlight. Toujours avec des menus élégants, Expression Blend vous propose de gérer votre pack de *Font* très facilement. C'est lui qui gèrera automatiquement la notion de ressource au sein de votre projet. L'ajout de *Font* sera donc totalement transparent pour les graphistes.[Fig.7]

La datagrid

L'affichage de données tabulaires est primordial dans tout type d'application. Silverlight étant orienté application métier dans le monde professionnel, Microsoft se devait d'offrir un contrôle capable de réaliser un rendu digne de ce nom.

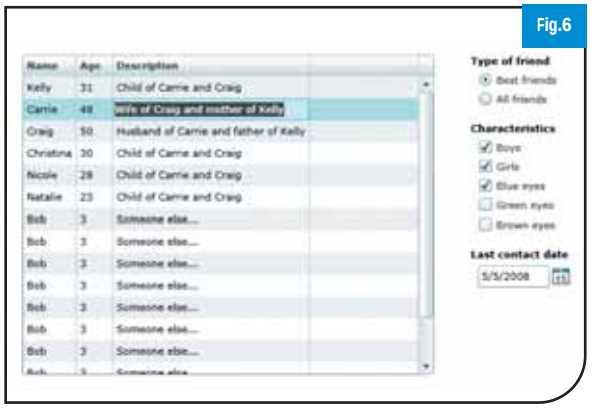
Voici la liste des nouveautés de ce contrôle natif :

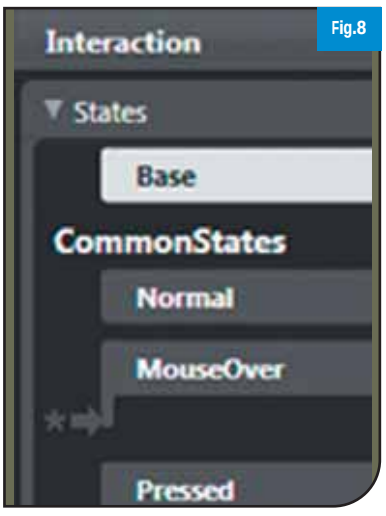
- Support de l'accessibilité. La datagrid est tout à fait lisible par un logiciel tiers.
- La datagrid supporte à présent les *VisualStateManager,* autrement dit, il est possible de venir personnaliser chaque état de vos cellules, que ce soit sur le clic ou le survol de la souris. [Fig.8]
- Il est possible de trier chaque colonne et ce, nativement.
- Il est possible de ré-agencer les colonnes comme l'utilisateur le souhaite.
- Dans la même optique que la réorganisation de colonnes il est à présent possible de les redimensionner, bien entendu, ceci n'est pas obligatoire en soi. C'est au développeur de choisir.
- Il est possible d'afficher une ligne détail à chaque ligne sous forme de " collapse / expand "
- Il est possible d'éditer, à partir de *datatemplater* et de *databinder,* facilement, le contenu de vos datagrids. Ainsi on peut aisément imaginer le scénario suivant. Lorsque l'utilisateur double-clique sur une cellule, celle-ci passe en mode édition et un slider vient se substituer à la valeur de la cellule afin de pouvoir éditer correctement cette valeur.

Pour voir un projet Template Silverlight 2, reportez-vous au projet Arcane de Valentin Billotte : http://www.codeproject.com/KB/silverlight/Silverlight_treeview.aspx

Nous reviendrons très rapidement sur Silverlight 2, les nouvelles fonctions. En attendant, bonne découverte ! ■

*Nos remerciements à **Stève Sfartz** (Architecte en système d'informations - Microsoft France, et un des responsables du projet eclipse4sl).*





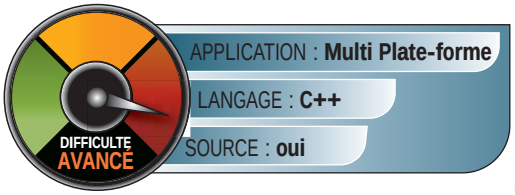
Programmation Qt

Code source sur www.programmez.com

3e partie

Suite et fin de la série d'articles sur la programmation Qt. Nous allons continuer, dans ce numéro, le développement de l'application QuickDesktopNotes en créant le " tray icon ".

Note : Par endroits le code n'apportant pas de nouveauté est tronqué. Nous l'avons signalé par / ...*/.*



Le " tray icon " est ce petit icône qui se fait oublier dans la barre des tâches. Il est disponible sur tous les gestionnaires de fenêtres évolués. Dans la majorité des cas, il s'agit d'un icône dans le dock auquel est adjoint un menu contextuel. C'est bien entendu ce que nous allons mettre en place, et le rendu final sera de ce type : [Fig.1].

Fig.1

Le fichier d'en-têtes ne présente pas de difficulté particulière, ce qui nous permet de nous concentrer sur le code source. Notez cependant, que la classe *Ptray* hérite de *PNoteEditor* qui constitue le widget principal de cette application.Commençons par inclure les en-têtes nécessaires à notre programme :

```
#include <QtGui>
#include <QtDebug>
#include "ptray.h"
#include "pconfigdialog.h"
#include "pxmlhandler.h"
```

Puis nous implémentons le constructeur. Dans celui-ci nous initialisons une variable booléenne que nous utiliserons plus tard. Puis nous lisons et appliquons les préférences de l'utilisateur, créons les différentes actions (QAction) qui sont en réalité les entrées de notre menu contextuel. Nous prenons aussi soin de mettre en place les connexions signals / slots de l'application, pour finalement charger les notes issues d'une précédente session. Bien sûr nous affichons aussi l'icône dans le dock et nous initialisons une note vide afin d'être prêt à travailler !

```
Ptray::PTray() : PNoteEditor()
{
    warnOnClose=false;
    readSettings();
    createActions();
    createTrayIcon();
     connect(trayIcon, SIGNAL(messageClicked()), this, SLOT
(messageClicked()));
   connect(trayIcon, SIGNAL(activated(QSystemTrayIcon::Activa
tionReason)),
           this, SLOT(iconActivated(QSystemTrayIcon::Activati
onReason)));
    connect(this, SIGNAL(noteTitleChanged(const QString&)),
            this, SLOT(on_noteTitleChanged(const QString&)) );
    connect(this, SIGNAL(noteTextChanged(const QString&)),
            this, SLOT(on_noteTextChanged(const QString&)) );
    trayIcon->show();
    setVisible(true);
    setNoteTitle("");
    setNoteText("");
    loadRssFile(notesFileName);
}
```

MIEUX SUPPORTER L'UTILISATEUR

Il est très important lors du développement d'une application graphique de porter une grande attention à la sauvegarde et à la restauration des préférences et du travail de l'utilisateur. En effet, c'est la condition nécessaire à l'utilisateur de s'approprier l'application. Cela lui permet de sentir que l'application se plie à ses désirs d'ergonomie et c'est la seule chose qui garantisse que l'application va être utilisée. Plus exactement, cela devrait être la seule garantie. Dans la réalité, de nombreuses applications (en particulier les applications métiers) sont développées sans aucune considération pour l'utilisateur car ce dernier est obligé d'utiliser ledit programme.Voyons comment lire les préférences utilisateur avec readSettings() :

```
void PTray::readSettings()
{
    QSettings settings("Programmez", "QuickDesktopNotes");
   QPoint pos = settings.value("pos", QPoint(200, 200)).toPoint
();
   QSize size = settings.value("size", QSize(800, 600)).toSize
();
     notesFileName = settings.value("notesFileName","desktop
_notes.rss").toString();
     iconFileName = settings.value("iconFileName",":/images/
burn.png").toString();
    setWindowIcon(QIcon(iconFileName));
    resize(size);
    move(pos);
}
```

Dans un premier temps nous créons un objet QSettings (faites appel à l'assistant pour le détail de l'API). Le constructeur prend 2 arguments : un nom d'entreprise et un nom de programme. L'intérêt de cette classe par rapport à une implémentation personnalisée est qu'elle s'adapte au système sur lequel elle évolue. En effet, les préférences seront sauvegardées dans le registre sous windows, dans le système de préférences XML sous Mac OS et dans $HOME / .config / Programmez / QuickDesktopNotes.conf sous Unix / X11. Une fois l'objet construit, nous récupérons les valeurs sauvegardées grâce à la méthode value(). Cette dernière retourne un QVariant et peut prendre deux arguments : le nom du paramètre (la clé) et une valeur par défaut (facultative). Il est fortement recommandé de ne pas oublier cette valeur par défaut sous peine de comportement erratique lors du premier lancement de l'application… QVariant est l'un des types de données extrêmement intéressant de Qt, il permet de représenter la plupart des types de valeurs Qt et C++. Bien entendu, QVariant possède une panoplie complète de méthodes de conversion vers des types plus spécifiques. Nous utilisons cette possi-

bilité pour transformer le paramètre " pos " en QPoint par exemple. Le lecteur désireux d'en apprendre plus sur cette classe se référera avantageusement à l'assistant. Comme vous le constatez à la lecture de cette courte fonction, en peu de lignes nous avons mis en place un système qui enregistre et restaure jusqu'à la position qu'avait la fenêtre quand l'utilisateur a fermé l'application la dernière fois. Voyons maintenant comment écrire ces préférences :

```
void PTray::writeSettings()
{
      QSettings settings("Programmez", "QuickDesktopNotes");
      settings.setValue("pos", pos());
      settings.setValue("size", size());
      settings.setValue("notesFileName",notesFileName);
      settings.setValue("iconFileName",iconFileName);
}
```

Aucune grosse difficulté ici non plus, il suffit de construire un objet QSettings de la même façon que lors de la lecture puis, nous appelons la fonction *setValue()*. Celle-ci prend 2 paramètres, le nom de l'option et sa valeur (qui peut être n'importe quoi du moment que cela rentre dans un QVariant). Afin que l'utilisateur se sente vraiment dans ses pantoufles avec notre application, il est de bon ton de ne pas perdre les notes qu'il saisit…

SAUVEGARDER...

Nous commençons par vérifier que nous pouvons ouvrir le fichier en écriture. Si c'est le cas nous avons un QFile valide et nous créons un flux texte vers ce fichier à l'aide de QTextStream. Nous modifions ensuite le curseur de la souris pour qu'il prenne sa forme d'attente (un sablier, une montre, etc.), puis nous écrivons le XML directement. Tout d'abord le début (statique) du fichier RSS, puis la liste des news. Nous faisons appel à la macro Qt foreach(variable, itérateur) pour cycler dans la liste des actions. Nous allons voir plus loin la petite astuce mise en place pour différencier les actions qui représentent l'historique des notes, des actions représentant les entrées du menu. Lors de la génération du XML nous avons recours à la méthode Qt::escape() qui échappe les caractères ennuyeux du XML. Une fois que le fichier est intégralement écrit nous restaurons le curseur. Veuillez noter qu'il est bien souvent plus "propre" d'utiliser les classes de Qt pour écrire le XML (QDomDocument, QdomElement, etc.). Cependant, ces classes suivant le modèle de programmation Java XML, elle sont très verbeuses... Charger le fichier précédemment écrit est beaucoup plus concis. Nous ouvrons le fichier *spécifier,* s'il existe, puis nous en faisons une source d'entrée XML, on instance un *parser* puis le *handler* développé au numéro précédent. Ensuite, nous ajoutons les actions à l'historique. En moins de 70 lignes nous avons fait en sorte que l'utilisateur aime travailler avec notre application ! Attelons nous maintenant à la mécanique interne de QuickDesktop-Notes. Tout d'abord nous devons surcharger le slot setVisible() afin d'ajuster l'activation des différentes entrées de notre menu.

```
void PTray::setVisible(bool visible)
{
    minimizeAction->setEnabled(visible);
    maximizeAction->setEnabled(!isMaximized());
    restoreAction->setEnabled(isMaximized() || !visible);
    PNoteEditor::setVisible(visible);
}
```

Une fois ceci fait, nous passons le relais au slot setVisible() de PNoteEditor. Un autre point à ne pas négliger : l'information donnée à l'utilisateur. Nous ne pouvons pas considérer que ce dernier sait ce qu'est une application résidant dans le dock. Quand il ferme une fenêtre pour la première fois, nous devons l'avertir, c'est pour cela que nous surchargeons le gestionnaire de l'évènement de fermeture.

```
void PTray::closeEvent(QCloseEvent *event)
{
    if (trayIcon->isVisible() && !warnOnClose) {
      QMessageBox::information(this, tr("Quick Desktop Notes"),
                               tr("The program will keep
running in the "
                                  "system tray. To terminate
 the program, "
                                  "choose <b>Quit</b> in the
 context menu "
                                "of the system tray entry."));
        warnOnClose=true;
        hide();
        event->ignore();
    }
    else if(warnOnClose){
        hide();
        event->ignore();
    }
}
```

Le message affiché, nous positionnons warnOnClose à true afin de ne pas afficher un popup invasif à chaque fois que l'utilisateur ferme une fenêtre d'édition. La gestion des clics sur le tray icon est prise en charge par le slot iconActivated() qui gère l'affichage de la fenêtre d'édition et des messages (voir le code source du projet).
Le slot showMessage() appelé en cas de clic du milieu déclenche le slot showMessage() qui affiche une infobulle. Il est a noter que la disponibilité de cette fonctionnalité est fortement dépendante de la plate-forme. Nous y reviendrons plus tard. Si l'utilisateur clique sur le menu "Configure", cela déclenche le slot showConfigDialog().
[Fig.2] Il s'agit là d'une instanciation classique d'une fenêtre enfant (ici un QDialog) avec connexion des signaux à des slots (pour la mise à jour des préférences et la suppression d'une note) et mise en place de l'aspect de la fenêtre. Il eut été plus efficace d'instancier la fenêtre de configuration une seule fois dans le constructeur et de conserver celle ci dans une variable membre. Cependant, nous avons préféré cette approche pour les besoins de la démonstration. Si le signal settingsUpdated() est émis, le slot on_settingsUpdated() est appelé. Voyons de quoi il retourne :


Fig.2

```
void PTray::on_settingsUpdated(){
    readSettings();
    trayIcon->setIcon(QIcon(iconFileName));
}
```

Si les préférences sont modifiées nous les relisons et, détail d'importance, nous modifions l'icône dans le dock ! De même si le signal noteRemoved(QAction*) est émis, le slot removeActionFromHistory(QAction *) est appelé :

```
void PTray::removeActionFromHistory(QAction *action){
    if(action != currentNoteAction){
        history->removeAction(action);
        trayIconMenu->removeAction(action);
    }
}
```

Simple et efficace : si l'action n'est pas l'action courante, nous supprimons la note du menu et de l'historique. Il est maintenant temps de revenir sur l'affichage du message, le slot showMessage() ne fait qu'appeler son homonyme de la classe QSystemTrayIcon :

```
void PTray::showMessage()
{
   trayIcon->showMessage(noteTitle(), noteText(), QSystemTray
Icon::Information,15 * 1000);
}
```

CONSIDÉRER LES SYSTÈMES

Comme il a été précédemment mentionné, le comportement de ce "tray icon" dépend beaucoup du système hébergeant. Si sous X11 le comportement attendu correspond au comportement effectif, ce n'est pas le cas sous Windows en revanche (le temps d'affichage est négligé quand la fenêtre à le focus) ni sous Mac OS X (il faut carrément installer Growl pour profiter de la fonctionnalité). Voyons maintenant comment sont gérés les changements dans le titre et dans le corps :

```
void PTray::on_noteTitleChanged(const QString &str){
    QAction *ita = history->checkedAction();
    ita->setText(str);
}

void PTray::on_noteTextChanged(const QString &str){
    QAction *ita = history->checkedAction();
    ita->setToolTip(str);
}
```

Un slot s'occupe de récupérer les modifications et de les répercuter sur l'action. Avec les connaissances accumulées jusqu'ici, l'ajout d'une note pourrait être qualifié de trivial. Je vous laisse donc découvrir cela dans le code source. L'édition d'une note, par contre, fait appel à un mécanisme auquel nous avons souvent recours lorsque plusieurs objets sont connectés au même signal. Voyons le code :

```
void PTray::editNote(){
    QAction *action = qobject_cast< QAction* >( sender() );
    if(action){
        setNoteTitle( action->text() );
        setNoteText( action->toolTip() );
    }
}
```

Nous savons qu'il n'y a que des QActions qui sont connectées à ce slot. Nous utilisons donc la fonction de cast propre à Qt, *qobject_cast()*, bien meilleure que le cast standard C++, (particulièrement en cas d'héritage multiple) sur l'objet qui envoie le signal. Nous avons connaissance de cet objet grâce à la méthode sender() qui retourne un QObject. Une fois l'action récupérée nous approvisionnons la fenêtre d'édition avec les valeur adéquates. Presque à la fin, voyons 2 méthodes appelées dés le début de notre programme : createActions() et createTrayIcon(). Le code source n'est pas reproduit au complet car de nombreuses actions sont redondantes.

```
void PTray::createActions()
{
    minimizeAction = new QAction(tr("Mi&nimize"), this);
    minimizeAction->setIcon(QIcon(":/images/minimize.svg"));
    connect(minimizeAction, SIGNAL(triggered()), this, SLOT(hi
de()));
    /* ... */
    history = new QActionGroup(this);
    history->addAction(currentNoteAction);
    currentNoteAction->setCheckable(true);
    currentNoteAction->setChecked(true);
}
```

Dans un premier temps, nous créons l'ensemble des QAction nécessaires puis nous les connectons aux slots adéquats et nous ajoutons les icônes pour chacune d'elles. Pour les QAction qui symbolisent une note, nous les rendons "checkable" afin de permettre la sélection unique. Et pour nous simplifier la vie, nous créons un groupe d'actions qui permet la sélection exclusive des actions marqués "checkable".

```
void PTray::createTrayIcon()
{
    trayIconMenu = new QMenu(this);
    trayIconMenu->addAction(minimizeAction);
    /* ... */
    trayIconMenu->addSeparator();
    trayIconMenu->addAction(configAction);
    /* ... */
    trayIcon = new QSystemTrayIcon(this);
    trayIconMenu->addSeparator();
    trayIcon->setContextMenu(trayIconMenu);
    trayIcon->setIcon(QIcon(iconFileName));
}
```

createTrayIcon() quant-à-elle ajoute les actions précédemment créées au menu contextuel du tray icon. Nous constatons qu'il suffit de faire appel à la classe *QSystemTrayIcon* pour créer un icône de dock multi-plate-forme ! Pour finir, le slot appelé lorsque l'utilisateur désire quitter le programme :

```
void PTray::quitTray(){
    writeNotes();
    writeSettings();
    qApp->quit();
}
```

Celui-ci écrit les notes dans le fichier RSS ainsi que les préférences, puis quitte l'application grâce à la macro qApp. Je vous invite à jeter un oeil au main.cpp et à reprendre le premier article de cette série pour la partie compilation. J'aimerai remercier Lucie, mon épouse, pour avoir relu tous mes articles tout étant totalement étrangère au milieu informatique !

■ Arnaud **Dupuis**
Consultant - Uperto (Open Source Business Unit) - Devoteam group

La programmation déclarative : le loup, la chèvre et le chou en prolog

Si vous avez essayé de programmer le loup, la chèvre et le chou dans un langage conventionnel, vous avez probablement trouvé cela difficile. Pourtant, nous allons voir que Prolog et sa perception de la programmation déclarative rend tout très simple. Toutefois, il faut bien comprendre que nous n'allons pas écrire un algorithme pour chercher la solution, mais nous traduirons simplement notre problème sous la forme d'un programme en Prolog !

PRÉLIMINAIRES

Pour nous échauffer, écrivons deux petits prédicats que nous utiliserons plus loin. Prolog prédéfinit un prédicat member / 2 : member(E, L) est vrai si la liste L contient l'élément E. Nous disposons également du prédicat delete / 3 : delete(L1, E, L2) est vrai si L2 représente la liste L1 dans laquelle on a supprimé toutes les occurrences de l'élément E. Nous pouvons employer celui-ci pour écrire le prédicat not_member / 2, inverse de member / 2 :

```
not_member(Elem, Liste) :-
  delete([Elem | Liste], Elem, Liste).
```

Que signifie cette règle ? Si on place Elem en tête de la liste Liste, puis que l'on supprime toutes les occurrences d'Elem, on doit retrouver Liste… c'est-à-dire, qu'au départ, Liste ne doit pas contenir Elem ! Le second prédicat se révèlera encore plus simple : pas_ensemble(L, X, Y) sera vrai si la liste L contient soit X, soit Y (mais pas les deux à la fois). Deux règles évidentes permettent de le spécifier :

```
pas_ensemble(L, X, Y) :-
  member(X, L),
  not_member(Y, L).

pas_ensemble(L, X, Y) :-
  not_member((X, L),
  member(Y, L).
```

Examinons bien tous les aspects du problème. A chaque fois, le passeur peut tantôt embarquer un animal et l'amener de l'autre côté, tantôt traverser seul la rivière pour aller chercher quelqu'un sur l'autre rive. Cependant, il ne doit jamais laisser la chèvre seule avec le chou, ni le loup seul avec la chèvre : cela signifie qu'il ne peut embarquer le loup que si la chèvre ne se trouve pas sur la même rive. Il en va de même pour le chou. On remarquera également qu'il nous suffit, pour décrire à tout moment la situation, de préciser l'identité de ceux qui figurent sur une rive, par exemple celle de gauche (car ceux qu'elle n'accueille pas se trouvent forcément sur celle de droite), ainsi que la rive sur laquelle repose la barque (gauche ou droite).

TRADUCTION EN PROLOG

Pour résoudre le problème, nous allons écrire un prédicat solution(*Rive*, *Cote*, *Sol*). La première variable, *Rive*, donne la liste des passagers présents sur la rive de gauche, par exemple [loup chou]. *Cote* désigne la rive sur laquelle la barque a provisoirement élu domicile (droite ou gauche). *Sol* symbolisera enfin la liste des coups permettant d'arriver à la solution, à partir de celle qu'ont définie *Rive* et *Cote*. D'emblée, nous nous voyons en mesure d'écrire une première règle pour ce prédicat :

```
solution([ ], droite, [ ]).
```

Celle-ci stipule que solution / 3 se vérifiera si la rive de gauche est vide (donc, si tout le monde se trouve à droite), si la barque se situe sur la droite et si la liste des coups à effectuer se résume à la liste vide. Voilà qui se révèle logique : si tout le monde a passé la rivière et que la barque est à droite, il ne reste plus rien à faire, car on a résolu le problème ! Deux autres règles définissent les deux coups éventuels :

```
solution(Rive, Cote, Sol) :-
  barque(Rive, Cote, Sol).


solution(Rive, Cote, Sol) :-
  transport(Rive, Cote, Sol).
```

Les deux prédicats barque / 3 et transport / 3 correspondent respectivement au voyage de la barque seule et à celui d'un passager. Le code du premier reste très simple :

```
barque(Rive, gauche, [barque_droite |  Sol]) :-
  pas_ensemble(Rive, loup, chevre),
  pas_ensemble(Rive, chevre, chou),
  transport(Rive, droite, Sol).


barque(Rive, droite, [barque_gauche |  Sol]) :-
  pas_ensemble(Rive, loup, chevre),
  pas_ensemble(Rive, chevre, chou),
  transport(Rive, gauche, Sol).
```

La première règle s'applique lorsque la barque repose sur la gauche. Elle stipule que, dans ce cas, le premier coup consiste à amener la barque à droite (barque_droite) à condition que le loup et la chèvre ou que la chèvre et le chou ne figurent pas sur la même rive. On force ensuite un " transport ", afin d'éviter deux voyages consécutifs sans passager (dans le cas contraire, la barque pourrait accomplir indéfiniment des allers et retours à vide…). La seconde règle, symétrique, traite le cas du trajet effectué en sens contraire.

LE TRANSPORT DES PASSAGERS

Examinons à présent les règles qui existent pour le prédicat transport / 3. Commençons par le cas du loup :

```
transport(Rive, gauche, [loup-droite |  Sol]) :-
 member(loup, Rive),
 not_ member(chevre, Rive),
 delete(Rive, loup, NR),
 solution(NR, droite, Sol).


transport(Rive, gauche, [loup-droite |  Sol]) :-
 member(loup, Rive),
 not_ member(chou, Rive),
 delete(Rive, loup, NR),
 solution(NR, droite, Sol).
```

La première règle s'applique lorsque la barque figure sur la gauche et que l'on désire amener le loup sur la droite. Le loup devra par conséquent se tenir sur la gauche (membre de la liste Rive). On exige également que la chèvre ne se trouve pas au même endroit (afin d'éviter de la laisser éventuellement seule avec le chou). On construit alors la nouvelle liste NR. Le loup a quitté désormais la rive de gauche tandis que la suite des coups à jouer (Sol) part dorénavant de cette nouvelle situation : rive de gauche privée du loup et barque placée sur la droite.
La seconde règle reste identique, mais interdit cette fois la présence du chou.
Ces deux règles permettent ainsi de transporter le loup tout en laissant la chèvre ou le chou à gauche, mais pas les deux en même temps ! En outre, deux règles symétriques aux précédentes contrôlent le passage du loup à gauche :

```
transport(Rive, droite, [loup_gauche |  Sol]) :-
 not_member(loup, Rive),
 member(Rive, chevre),
 solution([loup |  Rive], gauche, Sol).


transport(Rive, droite, [loup_gauche |  Sol]) :-
 not_member(loup, Rive),
 member(Rive, chou),
 solution([loup |  Rive], gauche, Sol).
```

Cette fois, on obtient la " nouvelle situation " en ajoutant le loup à la population de la rive de gauche. Il faudra donc tenir compte en tout de quatre règles pour transporter le loup. A présent, vous vous trouvez en mesure d'écrire vous-même les quatres règles concernant le chou ! Le transport de la chèvre reste beaucoup plus simple. Puisqu'il nous est permis de laisser le loup seul avec le chou, nous pourrons embarquer la chèvre sans avoir à nous plier à la moindre contrainte, ce qui se traduit simplement par deux règles :

```
transport(Rive, gauche, [chevre_droite  |  Sol]) :-
 member(chevre, Rive),
 delete(Rive, chevre, NR),
 solution(NR, droite, Sol).


transport(Rive, droite, [chevre_gauche  |  Sol]) :-
 not_member(chevre, Rive),
 solution([chevre |  Rive], gauche, Sol).
```

Une fois que vous aurez saisi toutes ces règles dans un fichier, chargez celui-ci sous Prolog. Pour lancer le programme, entrez :

```
solution([chevre, chou, loup], gauche, S).
```

On donne la situation initiale (tout le monde occupe la rive de gauche et la barque se situe du même côté) et Prolog tentera d'instancier S de manière à satisfaire le prédicat solution / 3 à partir des règles que nous avons définies. En quelques fractions de seconde, vous obtenez la solution du problème :

```
S  =  [chevre_droite, barque_gauche, loup_droite, chevre_gauche,
 chou_droite, barque_gauche, chevre_droite]
Yes
```

Vous pouvez vérifier la validité de la solution proposée. Tout cela a des chances de paraître un peu magique, mais là réside justement la toute-puissance des langages déclaratifs. Nous n'avons pas créé ici de boucles, ni de tests, ni de structures de données complexes ; il nous a fallu simplement énoncer un problème et le compilateur s'est chargé du reste.

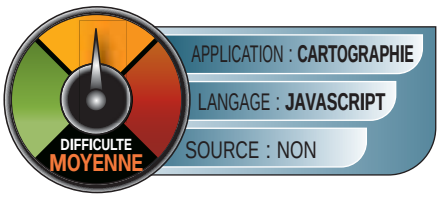
RÉFÉRENCE

R. Mompelat, Advanced programming in Prolog for Computational Linguistics and Artificial Intelligence, 2007, Lincom GmbH

■ Rodrigue Sabin **MOMPELAT**, PhD
Docteur en Sciences Cognitives-Intelligence Artificielle - Danemark.

Bien démarrer avec Virtual Earth

La présentation d'informations géolocalisées a pris ces dernières années une importance croissante, et ce mouvement va en s'accélérant, notamment grâce à la démocratisation de l'utilisation des GPS, et à leur intégration dans les mobiles par exemple. Que les applications soient destinées au grand public (recherche localisée, itinéraires automobiles) ou aux professionnels (transport, agriculture, urbanisation, etc.) les services proposés se multiplient dans ce domaine.



Dans ce contexte, les grands acteurs de l'Internet ont déployé des solutions de recherche, de calcul d'itinéraires, des sociétés se spécialisent dans certains métiers, des standards sur les formats des données commencent à apparaître : c'est dans cet environnement qu'évolue Microsoft, comme acteur majeur, via sa plate-forme " Virtual Earth ". Cette introduction ne lève le voile que sur une partie de la plate-forme, l'intérêt est avant tout de pouvoir, finalement, afficher une carte en maîtrisant son rendu tout en y ajoutant une information géographique qui provient d'une source tierce. Dans cette optique, seuls les objets et méthodes principaux seront évoqués.

AFFICHAGE DE LA CARTE

Virtual Earth est avant tout une technologie basée sur une librairie de contrôles JavaScript qui permet de manipuler des objets, dont le principal est la carte. Cette technologie est donc indépendante des autres technologies développées par Microsoft, notamment indépendante de Asp.Net. Le développement d'une application web utilisant Virtual Earth commence donc par l'intégration dans le code HTML du lien vers la librairie JavaScript.

```
<script type="text/jscript" src="http://dev.virtualearth.net/
mapcontrol/mapcontrol.ashx?v=6"></script>
```

C'est à ce stade que l'on peut commencer à utiliser Virtual Earth par l'intégration dans la page web du contrôle principal, **VEMap,** dédié à la gestion de la carte. C'est à travers cette classe que seront, notamment, gérées et affichées les données images et vecteurs. Ce contrôle sera aussi le lien principal avec l'utilisateur, lien qui sera exploité au travers, principalement, de la gestion des évènements de la carte. La carte peut être, par exemple, encapsulée dans un bloc div et initialisée lors du chargement de la page via l'appel d'une méthode JavaScript. La fonction OnPageLoad nous permet ici d'instancier la classe **VEMap,** puis de configurer la première visualisation de la carte (position, zoom, type de donnée, etc.)

```
<body onload="OnPageLoad();">
<div id="divMap"/>
[…]
function OnPageLoad()
{
    var BW = new VELatLong(43.5552, 1.4830);
    map = new VEMap('divMap');
    map.LoadMap(BW, 11, VEMapStyle.Shaded , false, VEMapMode.
Mode2D, true, 0);

    map.SetScaleBarDistanceUnit(VEDistanceUnit.Kilometers);
}
```

[Fig.1]

Regardons de plus près les paramètres utilisés pour initialiser la carte :

Nom	Description
BW	Il s'agit du point sur lequel est centrée la carte. Ce point est une instance de la classe VELatLong qui possède comme propriétés principales la latitude et la longitude du point (des informations d'altitude peuvent également être précisées). VE travaille uniquement dans le système géodésique WGS84 ; cela signifie que les coordonnées d'un point ne peuvent être que des coordonnées géographiques (un point sur la sphère définie par ce système), donc pas de coordonnées cartographiques (un point sur un plan) comme c'est le cas pour une carte IGN dans le système de coordonnées Lambert II. Ces coordonnées sont représentées, techniquement, en degrés décimaux ({43.5552, 1.4830} équivaut à {43°33'18.72" 1°28'58.8"}) de manière à pouvoir être manipulées à travers un flottant (la manipulation de données dans le système sexagésimal n'est pas le plus amusant …).
11	Il s'agit du niveau de zoom de la carte. Il s'étend de 1 (le plus large) à 19 (le plus précis). Sans rentrer dans les détails du calcul du zoom : le niveau 1 équivaut à peu près, à l'équateur, à une résolution de 78,2 km/pixel ; le niveau 19 à 30cm/pixel.
VEMapStyle.Road	La nature de la carte est choisie au chargement mais peut être modifiée par la suite. Elle peut être prise dans la liste suivante : *Road* : Style de carte 'route'. *Shaded* : Style de carte 'route' avec visualisation du relief. *Aerial* : Vue satellite de la zone. *Hybrid* : Affichage hybride image satellite et routes. *Oblique* : Style de carte oblique, équivalent à Birdseye. *Birdseye* : Style de carte oblique.
False (paramètre 'fixed')	L'utilisateur a-t-il la possibilité de modifier la carte (la carte affichée est elle statique ?) ?
VEMapMode. Mode2D	Visualisation en mode 2D, peut être paramétré en 3D.
True (paramètre 'showSwitch')	Affichage ou non des contrôles de navigation sur la carte.
0 (paramètre 'tileBuffer')	Permet de pré-charger les images adjacentes à la vue courante ; c'est un tampon permettant de fluidifier la manipulation de la carte.

A ce stade, la carte souhaitée est affichée à l'utilisateur, cependant son rendu reste entièrement contrôlable à travers des méthodes de **VEMap** (notez l'appel à *SetScaleBarDistanceUnit* pour passer en système métrique).

LE CONTRÔLE DE LA CARTE

La manipulation d'une représentation géographique fait appel à différents contrôles qui modifient les paramètres du rendu final :

- L'échelle : l'échelle de la carte est initialisée lors du chargement (voir ci-avant). Elle peut être modifiée via les méthodes *ZoomIn, ZoomOut, SetZoomLevel* de la carte. La valeur du zoom est un entier de 1 à 19 (voir ci-avant).
- La position : de la même manière, la position centrale de la carte peut être modifiée via la méthode *SetCenter* ou *SetCenterAndZoom* pour prendre en compte également l'échelle. Le centre de la carte est une instance de **VELatLong.**
- Le type de carte : il est possible de modifier la manière dont la carte est rendue via la méthode *SetMapStyle* qui attend comme paramètre un des éléments de l'énumération **VEMapStyle** décrite plus haut.

Notons qu'il est possible d'afficher ou non des contrôles supplémentaires sur la carte ; les plus courants sont :

- les contrôles standard de navigation : *ShowDashboard, HideDashboard*
- une mini-carte donnant un aperçu de la zone affichée dans la carte : *ShowMiniMap, HideMiniMap*
- plus intéressant, de contrôles personnalisés construits au besoin (barre de recherche par exemple) : *AddControl, DeleteControl, ShowControl, HideControl.*

Exemple en vue oblique *map.SetMapStyle(VEMapStyle.Oblique)* sans dashboard *map.HideDashboard()* : [Fig.2]

Voilà, nous savons afficher et contrôler le rendu de notre carte depuis le code de la page ; voyons maintenant comment maîtriser les actions de l'utilisateur final ...

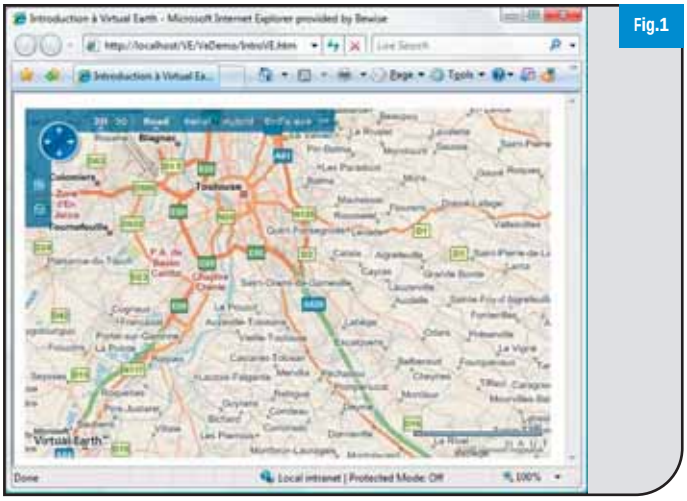


Fig.1

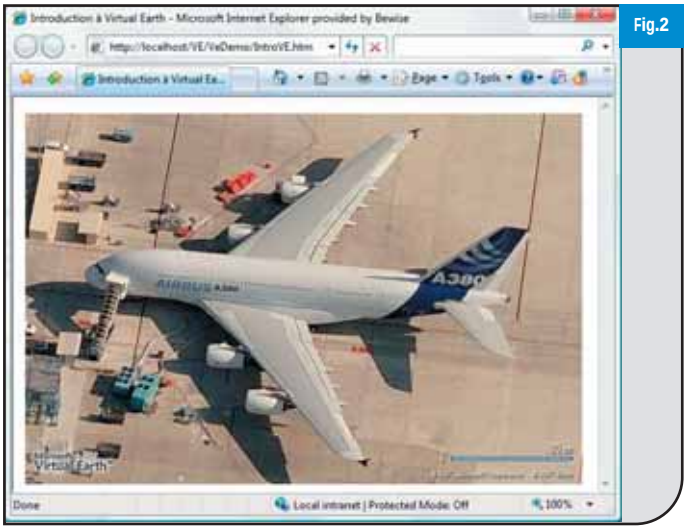


Fig.2

GESTION DES ÉVÈNEMENTS

La classe **VEMap** expose une liste d'évènements sur lesquels il est possible d'attacher une fonction JavaScript. Ces évènements appartiennent à trois catégories :

- Clavier (touche pressée),
- Souris (clic, déplacement, molette),
- Virtual Earth (changement de vue, zoom, etc.).

Ce sont sur ces évènements que seront construites les principales fonctionnalités d'une application exploitant Virtual Earth (saisie de point, détourage d'une zone géographique, déplacement, etc.).

Par exemple, le code suivant permet de visualiser les coordonnées du curseur de la souris via l'évènement *'onmousemove'*.

Dans la fonction *OnPageLoad()* que nous avons définie au début, est rajoutée la ligne suivante :

```
map.AttachEvent("onmousemove", mouseMoveHandler);
```

Un affichage des coordonnées est ajouté sur la page et la fonction *mouseMoveHandler,* appelée sur chaque déplacement souris, permet de mettre à jour le texte affiché dans la page par les coordonnées pointées par le curseur :

```
Latitude : <span id="CurseurLat">00.0000</span>
Longitude : <span id="CurseurLng">00.0000
function mouseMoveHandler(e)
{
    var loc = map.PixelToLatLong(new VEPixel(e.mapX, e.mapY));
   document.getElementById("CurseurLat").innerHTML = loc.Latitude.
toFixed(4);
   document.getElementById("CurseurLng").innerHTML = loc.Longitude.
toFixed(4);
}
```

[Fig.3]

Notons que l'objet passé en paramètre de la fonction ne comporte pas les coordonnées géographiques du curseur de la souris mais la position en pixel sur la carte. C'est pourquoi nous devons utiliser la méthode *PixelToLatLong* de la carte qui nous renvoie une instance de **VELatLong,** c'est-à-dire la position géographique.

Finalement, les bases sont posées, il ne reste plus qu'à comprendre comment ajouter à notre carte une donnée géographique, une donnée métier, qui reste la valeur ajoutée de notre application...

Fig.3

GESTION DES OBJETS 'VECTEUR'

Avant tout, revenons rapidement sur la différence entre données vectorielles et images. Les données images sont le fond de la carte, par exemple des photos aériennes ou satellites, qui sont physiquement des 'fichiers image' (concrètement matrice de pixels). Dans Virtual Earth ces images sont au format png et issues, par défaut, de la base d'images offerte par Microsoft et très régulièrement mise à jour. Il est également possible d'ajouter d'autres sources d'images. Les données vectorielles sont basées sur l'instanciation d'objets géométriques qui peuvent être, dans VE, des points, des lignes ou des polygones. Cette géométrie assez simple ne reflète pas la complexité des géométries disponibles dans d'autres systèmes, mais elle suffit à répondre à une majorité de problématiques. Regardons de plus près comment Virtual Earth nous propose de travailler avec cette géométrie. Les types d'objets disponibles dans VE sont référencés dans l'énumération **VEShapeType**

Pushpin	Un point	Une instance VELatLong
Polyline	Ensemble de lignes	Un ensemble de couples de VELatLong
Polygon	Polygone	Une liste de VELatLong dont le premier et le dernier ne sont pas égaux (contrairement à la définition d'un polygone dans d'autres géométries) ; le polygone est fermé.

Techniquement, un objet vecteur est une instance de la classe **VEShape** qui est caractérisée par un **VEShapeType** et un ensemble de données. Par exemple, la fonction suivante crée un polygone ayant quatre côtés :

```
var polygon = new VEShape(VEShapeType.Polygon,
[new VELatLong(43.6049,1.4428), new VELatLong(43.6050,1.4437),
new VELatLong(43.6040,1.4440), new VELatLong(43.6038,1.4429)]);
```

[Fig.4]

La plupart des systèmes d'information géographique font appel à un ensemble d'objets géométriques ; Virtual Earth implémente également cette notion de couche 'vecteur'.

LA SÉPARATION EN COUCHE

Une couche 'vecteur', tel un calque dans les applications de traitement d'images, est un ensemble de données que l'on peut afficher ou non, éditer indépendamment des autres objets ou encore instancier en fonction de sources différentes ; bref, c'est un outil permettant de manipuler de manière efficace les données en fonction de leur objet, de leur source ou autre.

Dans VE, une couche 'vecteur' est une instance de la classe **VEShapeLayer** qui permet de gérer une collection de **VEShape** ; de la même manière, chaque carte **VEMap** possède une collection de **VEShapeLayer.**

Dans les deux cas, un ensemble de méthodes permet d'éditer les objets de la collection (Add, Delete, Find, etc.).

Une couche dédiée aux points caractéristiques d'une ville peut être instanciée comme suit. La fonction *createToulouseLayer* crée ici l'instance de **VEShapeLayer** qui est affichée ou non en fonction du choix de l'utilisateur saisie dans la check box.

```
function createToulouseLayer()
{
    toulouseLayer.SetTitle("toulouseLayer");

     var shape000=new VEShape(VEShapeType.Pushpin, new VELat
Long(43.60436310, 1.44300700));
    shape000.SetTitle('Place du Capitole');
   shape000.SetDescription('<img src="./images/toulouse_vignettes/
capitole.jpg" width="220" height="120"> <p class="styleCopy
right">Images © panomir.com</p>');
    toulouseLayer.AddShape(shape000);

     var shape001=new VEShape(VEShapeType.Pushpin, new VELat
Long(43.60689500, 1.44147300));
    shape001.SetTitle('La cinémathèque');
   shape001.SetDescription('<img src="./images/toulouse_vignettes/
cinematheque.jpg" width="220" height="120"> <p class="styleCopy
right">Images © panomir.com</p>');
    toulouseLayer.AddShape(shape001);

 […]

    map.AddShapeLayer(toulouseLayer);

}
<input id="chkToulousePOI" type="checkbox" checked="checked"
onclick="onChangeCheckBox(this.checked)" /> Couche vecteur 'Toulouse'
```

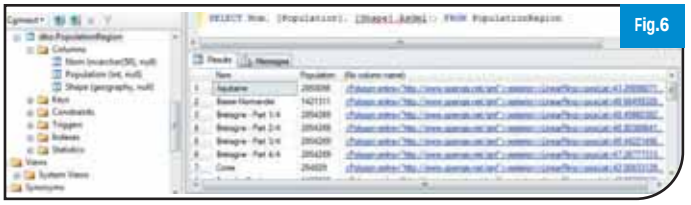


Fig.6

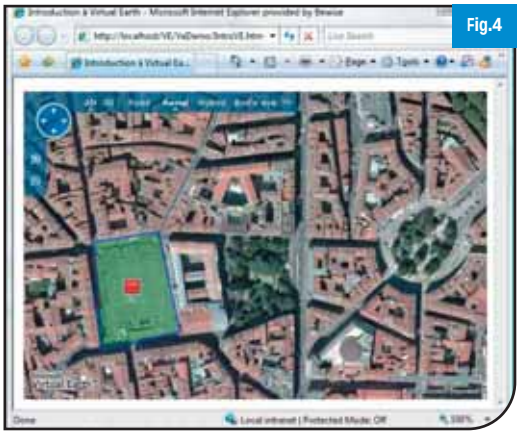


Fig.4


Fig.5

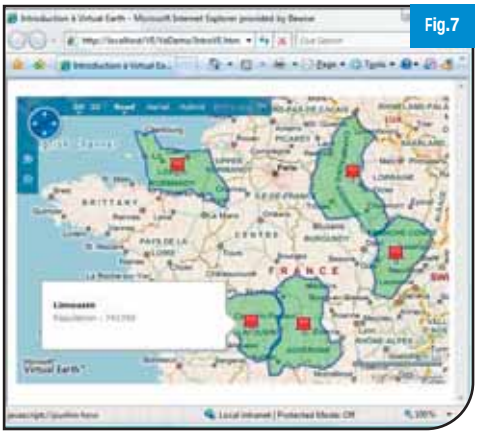


Fig.7

```
function onChangeCheckBox(isChecked)
{
    if( isChecked ) toulouseLayer.Show();
    else            toulouseLayer.Hide();
}
```

Notons que, comme **VEMap,** la classe **VEShape** propose un ensemble de méthodes permettant de mieux définir l'objet. Il est ainsi possible via la méthode SetDescription d'éditer le code html utilisé lors du rendu du ballon jaune affiché. De même, la méthode SetCustomIcon permettrait de changer l'icône (punaise rouge) utilisée pour représenter un Shape sur la carte ... [Fig.5]

Pouvoir afficher des données vectorielles est intéressant, mais les récupérer de sources externes dans un format standard, l'est d'autant plus que les fournisseurs de données géo localisées se multiplient.

L'IMPORT DE DONNÉES EXTERNES

Loin d'être anecdotique, l'import de données externes est proposé par la méthode ImportShapeLayerData de **VEMap** qui prend en entrée une source de données de type **VEShapeSourceSpecification.** Cette source définit le type de données, l'URL de la source et une instance de **VEShapeLayer** (couche dans laquelle seront importés les objets).

Virtual Earth propose donc d'importer des données aux formats KML (Keyhole Markup Language, language de description de données géographiques développé par Google et très largement répandu), GeoRss (données géographiques insérées dans un flux Rss) ou issues d'une collection Live Search Maps (cf. le service maps.live.com). Les deux premiers formats sont importants dans la mesure où les principaux clients permettant de représenter des données géographiques supportent au moins ces deux formats, ce qui amène nombre de fournisseurs de données à converger vers ceux-ci.

Ainsi, quoi de plus simple que de représenter dans une carte les dernières images géo-référencées postées sur le service d'hébergement de photos Flickr :

```
var l = new VEShapeLayer();
var url = "http://www.flickr.com/services/feeds/geo/France";
var veLayerSpec = new VEShapeSourceSpecification(VEDataType.
GeoRSS, url , l,true);
map.ImportShapeLayerData(veLayerSpec, null);
```

Bien évidemment, les formats GeoRss ou KML étant basés sur Xml, il est tout à fait envisageable de créer rapidement un Handler qui fournisse les données que l'on souhaite afficher ... D'ailleurs, si notre application utilise SQL Server 2008, c'est avec d'autant plus de facilité qu'il est possible de réaliser ce Handler, grâce à la *fonction geography::AsGml()* qui renvoie une valeur du nouveau type geography introduit dans SQL Server 2008. Cette fonction renvoie la valeur sous forme de GML, le langage de description de données géographiques défini par l'Open Geospatial Consortium (organisme international à but non lucratif ayant pour mission de définir les standards utilisés pour les services de géo localisation) et sur lequel KML a été construit.

Ainsi, la fonction suivante permet de déporter la génération des données (liste des régions françaises peu peuplées), c'est-à-dire la couche métier, dans un Handler et de ne réserver dans l'interface que l'appel à ces données. L'exemple suivant est construit sur une base SQL Server 2008 possédant une table des régions de France (avec population et Shape) : [Fig.6] On accède aux données grâce à un Handler qui renvoie un flux GeoRss construit à partir de l'encapsulation d'éléments Gml. Côté client, la fonction suivante permet d'ajouter simplement la couche 'vecteur' grâce au Handler qui vient d'être défini :

```
function LoadGeoRss()
{
    var l = new VEShapeLayer();
    var url = "http://localhost/VE/GeoServices/GeoRssHandler
.ashx";
    var veLayerSpec = new VEShapeSourceSpecification(VEData
Type.GeoRSS, url , l,true);
    map.ImportShapeLayerData(veLayerSpec, null);
}
```

Le résultat permet de représenter la carte des régions de France dont la population est inférieure à 1 500 000 habitants : [Fig.7]

CE QUI N'A PAS ÉTÉ DIT ...

Les notions de carte, objet vectoriel ou couche d'objets vectoriels étant définies, il faut savoir que Virtual Earth permet également de travailler sur d'autres type d'informations, et non des moindres.

Il est possible, comme pour une couche 'vecteur', de manipuler des couches de données image et donc de visualiser, en sus du fond classique, des plans d'immeubles ou des cartes métier générées (indice de végétation dans le domaine de l'agriculture de précision, zones de danger pour la gestion des risques industriels, etc.).

Comme beaucoup de systèmes 'grand public', Virtual Earth offre une fonctionnalité de calcul d'itinéraires routiers, dont la gestion des préférences offre une souplesse proche de celles des meilleurs 'GPS' automobiles. Cette fonctionnalité est bien sûr accompagnée d'une méthode permettant le géocodage d'un lieu (transformation d'une adresse postale en coordonnées géographiques).

Enfin, notons qu'une application construite sur Virtual Earth peut faire appel aux Services Web MapPoint. Service cartographique historique proposé par Microsoft, il est accessible via des Web Services standard (SOAP / XML) et offre certaines méthodes plus riches, comme la fonctionnalité de géocodage.

CONCLUSION

Cette initiation nous a permis de voir les fonctionnalités de base de Virtual Earth. Il faut avant tout retenir la facilité avec laquelle l'outil peut être exploité pour représenter un ensemble d'informations géographiques et offrir à l'utilisateur un ensemble conséquent de contrôles lui permettant de travailler le rendu des cartes.

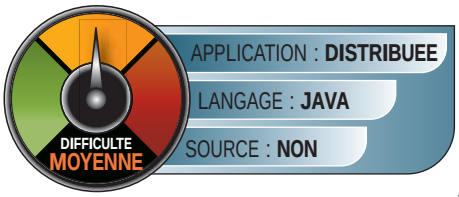
La richesse de l'outil fait qu'il ne serait pas possible d'en faire une description exhaustive en un article, mais la base telle que nous l'avons décrite et les autres fonctionnalités évoquées montrent que Virtual Earth a un degré suffisant de maturité pour être intégré à des applications professionnelles.

Ainsi, que ce soit pour appuyer des restitutions de Business Intelligence, équiper une application de géomarketing ou visualiser des informations terrains, Virtual Earth est une solution simple répondant à la plupart des problématiques concernant la géo localisation.

■ Philippe **Lonvaud**
Ingénieur-consultant Bewise - philippe.lonvaud@bewise.fr

Exécuter du code natif en Java avec JNA

Le slogan de Java – " Write once, run anywhere"*(écrire une fois, exécuter partout)* – a toujours mis en avant la portabilité des programmes écrits dans ce langage. La raison de cette portabilité ? Les programmes Java ne sont pas exécutés directement sur la plate-forme cible mais dans la machine virtuelle Java. Et c'est l'implémentation de cette dernière qui change suivant la plate-forme cible.



Si cette abstraction a conféré à Java sa portabilité, elle lui a du même coup ôté un certains nombre de fonctionnalités spécifiques aux systèmes cibles et communément appelées de "bas niveau". Pour répondre à cette problématique, les concepteurs du langage Java ont très rapidement intégré dans le JDK une API permettant l'exécution de code natif : il s'agit de l'API JNI (Java Native Interface).

LA SOLUTION JNI

JNI est une couche de programmation qui va permettre l'appel de code natif depuis des programmes Java et qui permet également de réaliser des appels Java dans du code natif ! L'appel au code natif réalisé par JNI n'est pas direct. En effet, il est nécessaire de définir des méthodes intermédiaires qui vont encapsuler les appels aux méthodes natives à utiliser. Ainsi, la mise en place d'une architecture JNI dans une application peut être décomposée suivant le processus suivant :

1 / Déclaration Java des méthodes intermédiaires encapsulant les méthodes natives C / C++ à utiliser
2 / Génération du fichier header C / C++ correspondant aux fonctions natives à appeler en respectant le prototype défini à l'étape 1
3 / Implémentation du code natif dans les méthodes déclarées dans le fichier header défini à l'étape 2
4 / Compiler le code implémenté
5 / Générer la bibliothèque dynamique correspondant au code compilé.
6 / Charger la bibliothèque dynamique générée à l'étape 5 dans le code Java afin de pouvoir faire appel au code natif

Ce processus est résumé brièvement sur le schéma de la figure 1.

Suivant la nature du code natif à utiliser, ce processus peut se révéler plus ou moins pénible. D'autre part, l'emploi de JNI oblige à compiler une bibliothèque dynamique qui devra être recompilée suivant le système cible. Ainsi, utiliser JNI pour des codes natifs relativement simples peut sembler disproportionné.

LA SOLUTION JNA

Partant de cet état de fait, des membres de la communauté Java ont mis au point une alternative permettant d'accéder à n'importe quelle bibliothèque partagée du système sans avoir à utiliser explicitement JNI. Nommée JNA (Java Native Access), cette API permet de s'abstraire de l'utilisation de JNI en se chargeant de l'interfaçage avec cette dernière API. De fait, les étapes liées à la manipulation de code C / C++, qui pouvaient s'avérer assez fastidieuses pour des développeurs Java ne connaissant pas ces langages, ne sont plus présentes, et l'interfaçage entre Java et le code natif devient bien plus simple. La mise en place d'une architecture JNA dans une application se décompose tout simplement en 2 étapes :

1 / Déclaration des méthodes natives à utiliser dans une interface Java qui doit obligatoirement étendre l'interface com.sun.jna.Library (ou tout autre sous-interface fournie par l'API) qui fait office de marqueur.
2 / Instanciation de l'interface qui charge dynamiquement la bibliothèque contenant le code natif.

Ce processus est résumé brièvement sur le schéma de la figure 2.

Dans le meilleur des cas - lorsque la bibliothèque contenant le code natif est déjà existante -, ce processus permet d'éviter d'avoir à manipuler et à compiler du code C / C++.

Dans le cas où la bibliothèque contenant le code natif n'existe pas, il faut la créer mais cela se réalise sans manipulation de code natif. A charge ensuite à JNA de charger la bibliothèque dynamiquement puis de générer et de manipuler du code JNI. Ceci permet de s'affranchir d'une compilation pour chaque plate-forme cible et de réaliser des appels de code natif avec un minimum d'effort pour le programmeur Java.

INSTALLATION DE JNA

L'installation de JNA ne pose pas de souci majeur. En effet, l'API se présente sous la forme d'une seule archive nommée jna.jar. Elle est téléchargeable sur le site du projet JNA à l'adresse http://jna.dev.java.net

Une fois l'API téléchargée, il vous suffit de la placer dans le classpath de votre application lorsque vous la compilerez et que vous l'exécuterez. Et le tour est joué ! Vous pouvez maintenant exécuter du code natif en Java.

PREMIER EXEMPLE AVEC JNA

Ce premier exemple assez simple va permettre de vérifier que vous avez correctement installé JNA sur votre système d'exploitation. Nous allons donc essayer d'utiliser quelques fonctions de base du langage C et notamment la fonction *printf* pour afficher le fameux message "Hello World !" à l'écran. La première étape du processus consiste à définir l'interface contenant les méthodes natives que l'on souhaite appeler. Voici notre interface Java dans le cas de cet exemple :

```
public interface ICLibrary extends Library{

    public static final String winCLib = "msvcrt";
    public static final String otherCLib = "c";

    /** declaration de l'instance de notre bibliothèque */
   public ICLibrary instance = (ICLibrary) Native.loadLibrary
(Platform.isWindows() ? winCLib : otherCLib, ICLibrary.class);

    /** declaration des fonctions natives a utiliser */
    public int printf(String format, Object... args);

    public int toupper(int c);

    public int tolower(int c);

    public double sqrt(double x);

    public int atoi(String s);
}
```

La déclaration de cette interface comporte le chargement dynamique de la bibliothèque que nous allons utiliser, qui ici dépend de la plate-forme utilisée. Ainsi, sur les systèmes de type Windows, on chargera la bibliothèque *msvrct.dll* et sur les autres systèmes, on chargera la bibliothèque *libc.so*. Le restant de l'interface concerne la déclaration des méthodes natives que nous allons utiliser dans notre programme Java.

Nous définissons donc la fonction *int printf*(String format, Object... args) qui vient mapper la fonction C éponyme *int printf*(const char* format, ...). On peut d'ores et déjà remarquer que JNA réalise automatiquement le mapping entre le type String de Java et les chaîne des caractères C définies sous la forme de char*. Le second argument de la fonction C *printf* permet de préciser que la fonction peut recevoir un nombre variable d'arguments. L'ellipse proposée depuis Java 5 permet de définir une fonction ayant la même vocation, JNA se chargeant ensuite de faire le mapping entre les paramètres fournis lors de l'appel de la fonction Java. Enfin, notre interface comporte d'autres fonctions de base du langage C qui seront utilisées dans ce premier exemple.

Fig.2

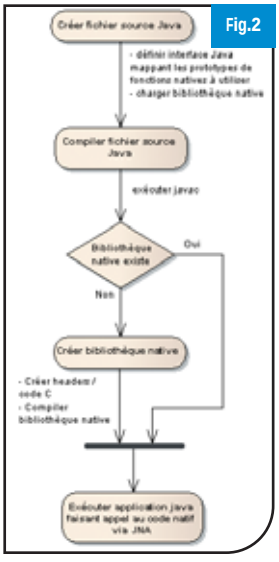


Fig.3

```
Console | Problems | Javadoc | Declaration | Properties | History
Hello World !
Valeur de entier : 00015
Valeur de la chaine : Chaine
Char mis en majuscule : A
Char mis en minuscule : b
Conversion de la chaine en entier : 128
Racine de 2 : 1.4142135623730951
```

Notre interface Java définie, nous pouvons passer à l'écriture de notre programme réalisant quelques appels natifs. Voici son code :

```
public class FirstExample {
 public static void main(String[] args) {
   ICLibrary cLibrary = ICLibrary.instance;
   // Le traditionnel Hello World ! ;)
   cLibrary.printf("Hello World !\n");
   // Utilisation avec des arguments
   int entier = 15;
   String str = "Chaine";
   cLibrary.printf("Valeur de entier : %05d\nValeur de la chaine :
 %s", entier, str);
   char c = 'a';
   cLibrary.printf("\nChar mis en majuscule : %c", cLibrary
.toupper(c));
   c = 'B';
   cLibrary.printf("\nChar mis en minuscule : %c", cLibrary
.tolower(c));
   str = "128";
   cLibrary.printf("\nConversion de la chaine en entier : %d", cLibrary
.atoi(str));
   cLibrary.printf("\nRacine de 2 : " + cLibrary.sqrt(2));
 }
}
```

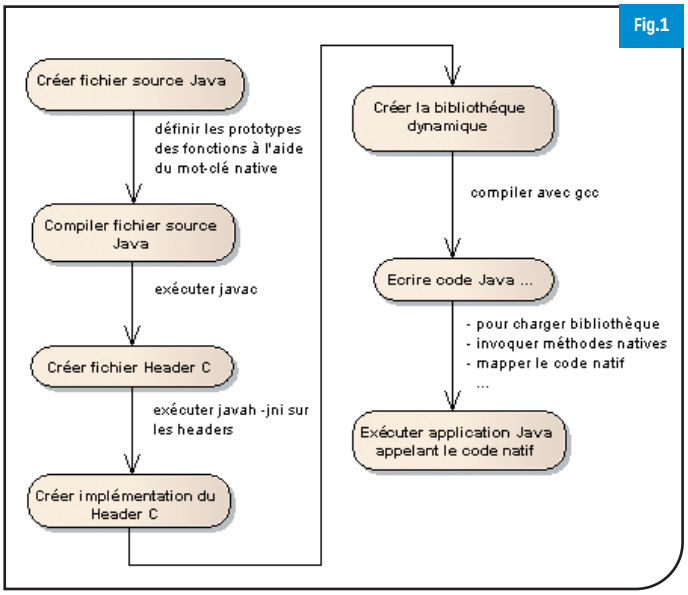
L'exécution de ces quelques lignes de code nous donne à l'écran le résultat présenté à la figure 3.

Le résultat obtenu nous confirme que la bibliothèque dynamique a bien été chargée par l'API et que les différents appels natifs ont été réalisés avec succès. Cet exemple permet de constater que JNA réalise automatiquement le mapping entre les types primitifs Java et ceux du langage C.

UN EXEMPLE A PRIORI PLUS COMPLEXE

L'installation de JNA validée par notre premier petit exemple, nous allons pouvoir détailler un exemple en apparence plus complexe. Son objectif est de réaliser la lecture d'un fichier texte à l'aide de fonc-

Fig.1



tions natives C. Cet exemple peut paraître délicat en apparence puisqu'il demande de manipuler correctement le pointeur de fichier renvoyé par la méthode C *fopen* notamment, mais nous allons voir qu'avec JNA, cela se fait très facilement. Pour réaliser cet exemple, on suit le processus défini précédemment et l'on définit dans un premier temps l'interface qui contient les méthodes natives que l'on souhaite appeler. Son code est le suivant :

```
public interface ILibraryIO extends Library{

  public static final String winCLib = "msvcrt";
  public static final String otherCLib = "c";

  /** déclaration de l'instance de notre bibliothèque */
  public ILibraryIO instance = (ILibraryIO) Native.loadLibrary
(Platform.isWindows() ? winCLib : otherCLib, ILibraryIO.class);

  /** déclaration des fonctions natives a utiliser */
  public void printf(String format, Object... args);

  public FILE fopen(String filename, String mode);

  public int fread(Pointer p, int size, int nbObj, FILE stream);

  public int feof(FILE stream);

  public int fclose(FILE stream);

}
```

Les développeurs Java maîtrisant le C auront cru reconnaître le type FILE qui est renvoyé par la fonction *fopen* du langage C. En réalité, il ne s'agit pas d'un mapping natif qui serait réalisé par JNA mais d'une définition de classe Java réalisée explicitement comme suit :

```
public class FILE extends PointerType{
    @Override
    public Object fromNative(Object nativeValue, FromNative
Context context) {
          Object o = super.fromNative(nativeValue, context);
          return o;
    }
}
```

Héritant de la classe abstraite PointerType, cette classe est un pointeur au sens JNA et permet ainsi de gérer l'adresse qui sera renvoyée par la fonction *fopen* et d'utiliser ce pointeur lors de futurs appels natifs.
Son nom sert simplement à se rappeler dans le programme quel est le type du pointeur du langage C que l'on manipule.
Le code permettant la lecture d'un fichier texte à l'aide de JNA et l'affichage de son contenu dans la console Java est présenté ci-dessous :

```
public class SecondExample{

  public static void main(String[] args) {
    ILibraryIO iLibraryIO = ILibraryIO.instance;
    // ouverture du fichier en lecture
    FILE file = iLibraryIO.fopen("test.txt", "r");
    // création d'un objet mémoire JNA
    Memory m = new Memory(1);

    // on parcourt le fichier tant qu'on a pas atteint la fin
    while(iLibraryIO.feof(file) == 0){
      // on essaie de lire 100 bytes que l'on va placer dans l'objet m
      int read = iLibraryIO.fread(m, 1, 100, file);
      // on recupère le contenu de m dans un objet String
      String buf = m.getString(0).substring(0,read);
      // on l'affiche
      System.out.println(buf);
    }

    // fermeture du fichier
    iLibraryIO.fclose(file);
  }
}
```

L'utilisation des fonctions natives se fait de manière assez naturelle et similaire au langage C. On ouvre tout d'abord le fichier à lire en lecture et on récupère le pointeur de type FILE renvoyé suite à cette ouverture. On crée ensuite un objet Memory JNA qui va permettre de récupérer les informations lues dans le fichier.
Une fois ces initialisations réalisées, on parcourt le fichier dans sa totalité en lisant des blocs de données de taille 100 bytes à chaque passage. Les données lues étant stockées dans l'objet Memory qui va nous permettre par la suite de les récupérer dans le but de les afficher dans la console Java. Enfin, le programme se termine avec la fermeture du flux de lecture ouvert sur le fichier.
En considérant un fichier test.txt présent dans le même répertoire que notre classe Java, on obtient en sortie l'affichage du contenu de ce fichier.

CRÉATION DE LA BIBLIOTHÈQUE DYNAMIQUE À UTILISER

Dans les 2 petits exemples présentés jusqu'à maintenant, la situation était quasi idéale avec l'existence de bibliothèques dynamiques contenant les fonctions natives que l'on souhaitait utiliser. De ce fait, il nous suffisait de les charger via JNA dans notre programme Java et de les utiliser telles quelles sans avoir à manipuler de code C / C++.
Dans un certain nombre de cas, il se peut que vous disposiez d'un code natif que vous souhaitiez utiliser dans un programme Java mais que vous ne disposiez pas encore de la bibliothèque dynamique correspondant à ce code. Il existe 2 solutions à ce problème :

1 / La première est applicable dans le cas où votre code natif serait en C ou en C++ privé de sa partie objet. Dans ce cas là, il vous suffit d'utiliser un compilateur tel que gcc pour créer votre bibliothèque dynamique. Il faudra bien entendu effectuer des compilations différentes de cette dernière pour vos différents systèmes d'exploitation cible. Cependant, vous n'aurez toujours qu'un seul code Java à réaliser puisqu'au sein de ce dernier, JNA vous permettra de connaître la plate-forme d'exécution et ainsi de charger la bibliothèque adaptée.
2 / La seconde est applicable dans le cas où votre code natif serait

en C++ objet. Ce cas de figure n'est pas réellement supporté par JNA. De ce fait, vous allez devoir écrire un wrapper C++ permettant de manipuler les différentes fonctions associées à ces objets. Vous aurez ensuite à compiler ce wrapper sous la forme d'une bibliothèque dynamique qui sera utilisée dans votre programme Java.

Afin de mieux comprendre la solution expliquée au point 2, nous allons détailler un petit exemple. Premièrement, considérons que nous avons à notre disposition une classe C++ IntegerMath dont voici le prototype :

```
class IntegerMath{

    public :

        int addition(int, int);
        int soustraction(int, int);

};
```

Cette classe triviale a pour but de fournir une implémentation de quelques opérations mathématiques élémentaires pour des entiers. Ces opérations sont celles que nous souhaitons utiliser dans notre programme Java. L'API JNA ne supportant pas la gestion des objets C++, la réalisation d'un wrapper C++ permettant l'utilisation de ces fonctions s'impose. Son code est le suivant :

```
#include "IntegerMath.hpp"

extern "C"{

    // instance de l'objet IntegerMath utilisée par le wrapper
    static IntegerMath* integerMath = NULL;

    // création d'une instance
    void create(){
       if(integerMath == NULL)
     integerMath = new IntegerMath();
    }

    /* Opérations proposées par la bibliothèque */
    int addition(int op1, int op2){
 create();
 return integerMath->addition(op1, op2);
    }

    int soustraction(int op1, int op2){
 create();
 return integerMath->soustraction(op1, op2);
    }

    // destruction de l'instance utilisée
    void destroy(){
 if(integerMath != NULL){
    delete integerMath;
     integerMath = NULL;
 }
    }

}
```

Les fonctions du wrapper C++ sont encapsulées dans un bloc externe "C" qui permet de préciser au compilateur qu'il ne doit pas décorer les fonctions C d'éléments C++ lors de la compilation, car ces derniers ne sont pas reconnus par JNA et cela poserait des problèmes lors de l'utilisation de la future bibliothèque dynamique. La phase de compilation va imposer la création de la bibliothèque dynamique *libIntegerMath.so* et enfin de *libWrapperIntegerMath.so*. Les commandes suivantes permettent cette compilation à l'aide de gcc :

```
g++ -o libIntegerMath.so IntegerMath.cxx -shared -fPIC
g++ -o libWrapperIntegerMath.so WrapperIntegerMath.cxx -shared
 -fPIC -L./ -lIntegerMath
```

Ces commandes produisent notamment la bibliothèque dynamique *libWrapperIntegerMath.so* utilisable en l'état sous un système Unix / Linux. Afin de pouvoir l'utiliser sous un système Windows, il faut réaliser la même compilation à l'aide de l'implémentation gcc sous Windows à savoir MinGW en nommant les bibliothèques comme suit : *IntegerMath.dll* et *WrapperIntegerMath.dll*.

L'utilisation de la bibliothèque dans un programme Java se fait ensuite très simplement comme pour les 2 premiers exemples de cet article en définissant tout d'abord une interface contenant les fonctions natives à appeler (ici create, addition, soustraction et destroy) et le chargement de la bibliothèque nouvellement créée. Avant l'exécution du programme Java correspondant, il convient de positionner la variable d'environnement sur laquelle le système hôte se base pour le chargement des bibliothèques natives, comme il faut, en lui ajoutant le chemin vers les répertoires où se trouvent nos bibliothèques dynamiques.

Dans le cas d'un système Unix / Linux, il s'agit de la variable LD_LIBRARY_PATH et pour les systèmes de type Windows de la variable PATH.

CONCLUSION

Cet article aura mis en avant les possibilités offertes par l'API JNA pour l'exécution de code natif en Java. Son avantage principal par rapport à JNI se situe au niveau de la manipulation du code C / C++ dont on peut s'abstraire avec JNA si l'on a à disposition une bibliothèque contenant les fonctions natives que l'on souhaite appeler. D'autre part, même dans le cas où cette bibliothèque n'existerait pas, sa création en code C / C++ s'avère également bien plus simple que le code natif que JNI oblige à produire ! Cependant, les possibilités offertes par JNA se limitent à une communication unidirectionnelle du monde Java vers le monde C / C++.

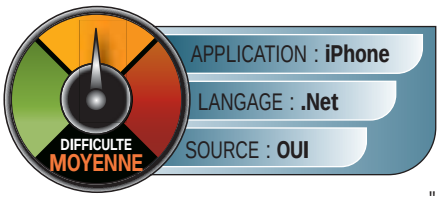
Quelquefois, il peut s'avérer nécessaire de communiquer depuis un programme C / C++ avec un programme Java et donc la machine virtuelle Java. Dans ce cas de figure, l'utilisation de JNI s'impose encore. Pour la plupart des autres cas d'utilisation, le choix de JNA se révèle payant et la légère perte de performance observée par rapport à JNI est bien vite compensée par un temps de développement plus rapide et une maintenance plus aisée.

■ Sylvain **SAUREL** - *Sopra Group. Agence Provence*

2e partie

Piloter votre Windows Media Center avec un iPhone

Le développement d'applications pour Windows Media Center (WMC) est quelque peu facilité avec l'arrivée du SDK 5.3 et ce, même si l'on sent un modèle objet bien lourd derrière. Il n'en reste pas moins qu'avec les technologies .NET il devient plus facile d'exposer les fonctionnalités de Windows Media Center sous la forme de services WCF (Windows Communication Foundation).



GESTION DE LA MISE EN VEILLE

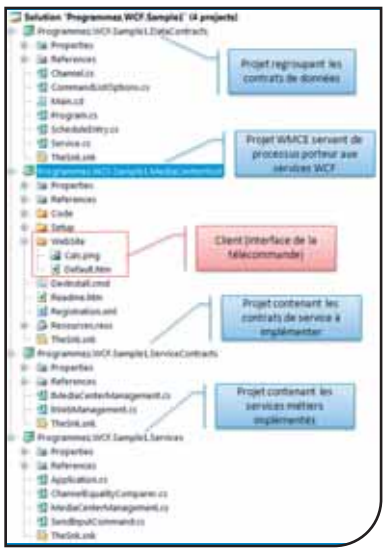
Le métier en rapport avec cette partie utilise pleinement la classe " Application " du Framework .NET :

```
Application.SetSuspendState(
System.Windows.Forms.PowerState.Suspend, true, true);
```

Le premier paramètre permet de choisir entre le mode suspendu et le mode hibernation, le second de forcer la machine à passer dans l'état suspendu immédiatement et le troisième de désactiver la possibilité de réveil événementiel de la machine.

GESTION DES AUTRES TOUCHES

Cette partie a de loin été la plus complexe à élaborer. En effet, le modèle objet de Media Center n'offre pas ces services en standard (déplacement Haut, bas, ok, menu TV, etc.). La solution qui m'avait initialement effleuré l'esprit était d'utiliser la classe " SendKeys " afin de simuler les touches du clavier. Malheureusement, ce fut impossible du fait que c'est l'add-in Média Center qui se chargerait d'envoyer les ordres (puisque cet add-in est le processus porteur WCF) et non pas directement la fenêtre Windows. Il m'a donc fallu recréer une classe permettant de réaliser cette opération directement en appelant les API bas-niveau du système d'exploitation.



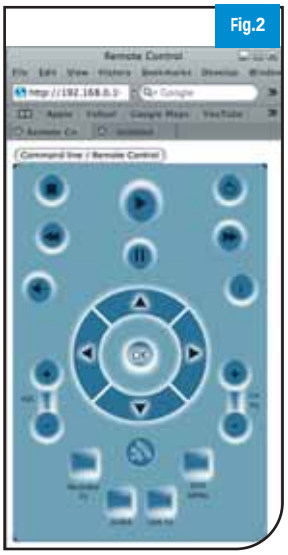


Fig.2

La logique fut donc la suivante :

1. Définir les combinaisons de touches gérées au travers d'un contrat de données afin de faciliter l'utilisation et la description
2. Récupérer le handle de la fenêtre Windows supportant le processus porteur de WMC
3. Eventuellement restaurer la fenêtre à l'écran si elle avait été iconifiée dans la barre de tâche. Pour cela, j'utilise l'API :

```
[DllImport("user32.dll")]
private static extern bool ShowWindow(IntPtr hWnd, WindowShow
Style nCmdShow);
```

4. Eventuellement, passer la fenêtre au premier plan si elle ne l'était pas afin que les ordres clavier lui soient transmis. Pour cela, j'utilise l'API :

```
[[DllImport("user32.dll")]
public static extern bool SetForegroundWindow(IntPtr hWnd);
```

5. Il reste à gérer les combinaisons de touches de types, Shift, Ctrl, Alt, Win. Le tout étant encapsulé dans une classe dédiée appelée " SendInputCommand ". Pour cette dernière, je me suis basé sur une application développée en 2005 (http://sourceforge.net/projects/mcecontroller/) et qui fournissait une partie du service que je recherchais. J'ai donc fait évoluer cette classe afin qu'elle réponde complètement à mon besoin initial. Amateurs de P / Invoke, vous serez servis ...

LA GESTION DE L'IHM

Afin d'éviter tout problème d'appel " Cross Domain " entre la page HTML qui exécute les appels HTTP et le Host hébergeant les services

Fig.1

WCF, j'ai embarqué la page HTML ainsi que l'image afférente directement dans le service et l'ai rendu accessible au travers du contrat de service suivant " IWebManagement " décrit précédemment.

```
public Stream GetFile(String fileName, String encoding)
{
 DirectoryInfo info
   = new DirectoryInfo(
     Environment.GetFolderPath(Environment.SpecialFolder.
System));

 WebOperationContext.Current.OutgoingResponse.ContentType = encoding;

 if (Directory.Exists(info.Parent.FullName + @"\ehome"))
 {
   return
     File.OpenRead(
       String.Format(@"{0}\{1}\{2}",
         info.Parent.FullName, "ehome", fileName));
 }

 return null;
}
```

" WebOperationContext " est une classe *helper* permettant d'accéder aux propriétés contextuelles (Request et Response) des appels WCF courants.

LE HOSTING WCF

Le processus porteur est directement géré par l'add-in créé en tâche de fond. J'ai donc réalisé ce traitement lors du lancement de l'add-in, c'est à dire lors de l'appel de la Méthode " Launch " qui sert de point d'entrée. Pour la mise en place d'une architecture REST avec WCF, j'utilise une nouvelle classe " WebServiceHost " qui n'est de toute façon qu'une classe fille de " ServiceHost " :

```
var TheHost
 = new System.ServiceModel.Web.WebServiceHost(
   typeof(Programmez.WCF.Sample1.Services.MediaCenterManagement));

TheHost.Open();
```

Tout le paramétrage du service est réalisé en fichier de configuration (" ehexthost.exe.config ") : [Fig.1]

LE CLIENT DE TEST

Comme indiqué précédemment, n'étant absolument pas versé dans l'utilisation du SDK natif IPhone, le plus simple pour moi était d'utiliser les standards du Web afin d'implémenter un client Javascript, en ciblant aussi bien Internet Explorer que Safari et Safari Mobile.
Pour cela, je devais résoudre deux problématiques :

1. Utiliser le minimum syndical en Javascript pour des raisons évidentes de portabilité
2. Utiliser " XmlHttpRequest " pour les requêtages des services REST.

Je ne me suis pas trop focalisé sur la qualité de cette IHM. Tout juste me suis-je quelque peu frotté à Microsoft Expression Design pour générer l'image de ma télécommande et y associer une Image Map ! [Fig.2]

Voici l'URI REST utilisée pour récupérer cette page :
http://192.168.0.10:8000/MediaCenter/IWebManagement/GetFile?filename=default.htm&encoding=text/html

Voici le code de la fonction Javascript utilisée pour requêter :

```
function Execute(serviceUri)
{
 var proxy = null;

 if (window.ActiveXObject)
   proxy = new ActiveXObject("Microsoft.XMLHTTP");
 else if (window.XMLHttpRequest)
   proxy = new XMLHttpRequest("");
   proxy.open('POST',
     window.document.getElementById("txtServiceURL").value
       + serviceUri, false);

   proxy.send(null);

   window.document.getElementById("divStatus").innerHTML = proxy.
status;
   window.document.getElementById("divStatusText").innerHTML
   = proxy.statusText;

   window.document.getElementById("divResponse").innerHTML
     = proxy.responseText;
}
```

Vous noterez l'utilisation d'un requêtage de type POST et synchrone.
Voici un exemple d'url pour la récupération du guide :
" http://MonServeur:8000/MediaCenter/IWebManagement/GetGuideByDate?fromDate=05/08/2008&toDate=05/09/2008 "

CONCLUSION

Au travers de cet article, je me suis attaché à démontrer plusieurs choses :

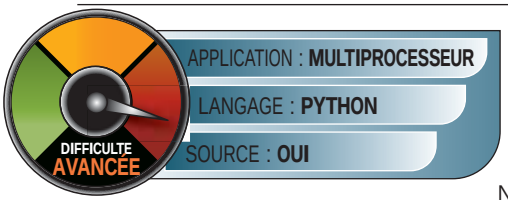
- La capacité offerte par le SDK de WMC à développer des add-in
- La capacité à créer un host WCF différent de ceux que l'on trouve habituellement
- La facilité offerte par WCF à mettre en place et gérer des services via une architecture REST
- Accessoirement, à développer les services d'une télécommande accessibles via un IPhone

L'utilisation d'une architecture REST permet de simplifier au maximum les protocoles utilisés entre un client et un service. Ces derniers ne viennent pas remplacer les Services WCF d'avant, mais plutôt les compléter judicieusement. Enfin, l'unification WCF offre au développeur .NET une nouvelle corde à son arc en intégrant les architectures REST de manière très simple.
Pour ce qui est du développement de la maison numérique de demain, Microsoft offre une première brique à tout développeur Microsoft .NET pour développer des applications au sein de WMC. Encore un nouvel axe d'unification de la plate-forme .NET après l'unification de l'exécution, des langages, des modèles de développement, de l'IDE de développement et de la bibliothèque de classe.

■ Frédéric **Colin** - *Bewise* - *frederic.colin@bewise.fr*

La programmation MPI avec Python et la plate-forme .Net

MPI, ou Message Passing Interface est une spécification dédiée à la programmation concurrente et distribuée et prévue initialement pour les langages de bas niveaux que sont C, C++ et Fortran. Cependant des implémentions pour Python et .Net ont vu le jour. Nous les découvrons aujourd'hui.



Dans un précédent numéro, nous avons faire connaissance avec les fondamentaux de la programmation MPI. Nous avons écrit tous nos exemples en C. A cette occasion le lecteur aura pu remarquer que le code de ces exemples n'était pas particulièrement concis, ni très facile à écrire, notamment en ce qui concerne la gestion des tampons de mémoire, entièrement à la charge du programmeur. Le problème est a priori inhérent à MPI puisque cette spécification a été définie pour les langages C, C++ et Fortran. Le choix de ces langages trouve tout simplement une raison historique, puisqu'en 1992, date de la première version des spécifications MPI, C, C++ et Fortran étaient à la fois très populaires et très utilisés. Depuis, des langages de plus haut niveau ont fait leur chemin. Pourquoi alors ne pas implémenter les spécifications MPI pour ces langages ? Techniquement rien ne s'y oppose. Tout ceci n'a pas échappé à la communauté du logiciel libre et c'est pourquoi nous avons aujourd'hui le plaisir de travailler avec deux implémentations Open Source de MPI, une pour Python, l'autre pour la plate-forme .Net

1 PYTHON ET PYMPI

Mais pourquoi Python au fait ? Après tout MPI a pour vocation d'améliorer les performances globales d'une application en répartissant son flot d'exécution sur les machines d'un cluster et / ou sur les coeurs d'un processeur multi-coeur. Python étant loin, en termes de performances brutes, de langages tels que C, C++ et Fortan, pourquoi l'employer dans ce contexte ? De l'humble avis de votre serviteur, Python a son utilité ici, au moins pendant la phase de développement. En effet, avec MPI comme ailleurs, l'écriture en Python est très rapide et concise et ce langage permet la mise au point d'applications en un temps minimum. Après quoi, on peut transposer le code en C si les performances ne sont pas satisfaisantes, ou on utilisera des outils tels que Pyrex ou Psyco, qui vous ont été présentés dans Programmez! 109. pyMPY est librement téléchargeable à l'adresse http://pympi.sourceforge.net/index.html. pyMPY est un interpréteur Python, supportant MPI, mais il n'est pas autonome. pyMPI constitue une sorte de surcouche à l'implémentation native de MPI dont votre système Linux (ou Cygwin sous Windows) doit disposer. Nous avons travaillé avec l'implémentation Open MPI qui vous a été présentée le mois dernier. Installer pyMPI est simple grâce à la suite traditionnelle de commandes :

```
./configure
make
make install
```

Lors de la configuration, l'implémentation MPI présente sur votre système est automatiquement recherchée et détectée. Toutefois vous avez la possibilité d'en imposer une particulière, comme ceci :

```
env CC='mon_mpicc' ./configure
```

pyMPI est remarquablement facile à utiliser mais remarquablement mal documenté. (Ceci explique-t-il cela ?) Cependant, si l'on a saisi le principe de la programmation MPI, la prise en main de pyMPI est aisée. On lancera une première fois pyMPI dans un terminal. On se retrouve alors dans l'interpréteur en mode interactif. On tape :

```
>>> import mpi
>>> dir(mpi)
```

Ce qui permet de connaître le nom de toutes les fonctions et constantes à notre disposition. [Fig.1] Ces informations, alliées à l'unique (!) script d'exemple qui vient avec la distribution, sont suffisantes pour travailler et nous écrivons l'incontournable Hello World :

```
# basic.py

import mpi
print "hello", mpi.rank, "sur", mpi.size
```

Deux lignes de codes seulement.... On exécutera ce script, par exemple sur un processeur à quatre coeurs, de cette manière :

```
mpirun -np 4 pyMPI basic.py
```

Ainsi mpirun (cf. le mois dernier) instancie quatre interpréteurs Python, chacun d'eux exécutant notre script.

2 PARTIE DE PING PONG

Voici maintenant l'équivalent du programme helloworld à messages, donné le mois dernier en langage C

```
# SendRecv.py

import mpi

if __name__ == "__main__":
  if mpi.rank == 0:
    for i in range(1, mpi.size):
      mpi.send("Hello", i)
  for i in range(1, mpi.size):
    msg, status = mpi.recv()
    print msg, "\n"
```

```
else:
  msg, status = mpi.recv()
  msg += " World de "
  msg += str(mpi.rank)
  mpi.send(msg, 0)
```

Treize lignes de code seulement, alors que l'équivalent en C en compte une grosse quarantaine. Le code Python est tellement immédiat qu'il ne mérite pas plus d'explications. Par contre, nous reviendrons à pyMPI à la fin de cet article.

3 MPI.NET

Nous nous intéressons maintenant à la programmation MPI dans le contexte de la plate-forme .Net. Il existe plusieurs implémentations de MPI pour cette plate-forme. Nous avons choisi MPI.Net disponible à http://www.osl.iu.edu/research/mpi.net/. Son installation est une formalité, mais elle nécessite la présence du *Microsoft Compute Cluster Pack* sous Windows. On se procurera gratuitement ce dernier sur le site de Microsoft. Après installation de ce dernier, il est pertinent de faire pointer le sous-répertoire *bin* par le PATH de votre système. Voici l'inévitable Hello World en MPI .Net et avec C# :

```
using System;
using MPI;

class Program
{
 static void Main(string[] args)
 {
   using (new MPI.Environment(ref args))
   {
     if(Communicator.world.Rank == 0)
       Console.WriteLine(
         "Le communicateur compte {0} processus",
         Communicator.world.Size);
     Console.WriteLine("Hello World, processus "
       + Communicator.world.Rank.ToString());
   }
 }
}
```

Là encore, le code parle de lui-même. On remarquera simplement que tout le code MPI est englobé dans un bloc de code dans lequel la durée de vie d'une instance de MPI.Environnment est assurée grâce au mot clé *using*. Ceci est incontournable. Pour compiler ce code, il faut ajouter une référence sur l'assembly MPI dans le projet, comme illustré [Fig.2]. Demander l'exécution de ce code sous Visual Studio n'a pas de sens. De manière analogue à ce que nous avons vu sous Linux avec Open MPI, on doit passer par les binaires du *Microsoft Compute Cluster Pack*. Ainsi pour exécuter un Hello World sur chacun des coeurs du microprocesseur à quatre coeurs on fera :

```
mpiexec -n 4 HelloWorld.exe
```

Lorsqu'il s'agit de lancer une application distribuée, on ne passe pas implicitement par *ssh* comme sous Linux mais par l'utilitaire *smpd*. Supposons que nous voulions lancer l'exemple *hostsnames* (fourni par la distribution de MPI.Net) sur les machines *Soleil* (1 coeur) et *Venus* (4 coeurs) d'un réseau, on commencera par lancer *smpd* sur chacune de celles-ci :

```
smpd -p 8677 -d 4
```

Puis, en supposant que l'exécutable sur *Venus* réside dans le répertoire courant, on donnera la commande :

```
mpiexec -n 1 -host soleil -wdir G:\bin Hostsnames.exe : -n 4
 Hostsnames.exe
```

On remarquera le commutateur *wdir* qui indique à *smpd* dans quel répertoire se situe l'exécutable devant être lancé sur l'hôte distant, *Soleil* pour notre exemple.

4 PROGRAMMATION POINT À POINT AVEC MPI.NET ET C#

Voici à nouveau le fameux Ping Pong :

```
using System;
using MPI;

class Program
{
 static void Main(string[] args)
 {
   using (new MPI.Environment(ref args))
   {
     Communicator world = Communicator.world;
     if (Communicator.world.Rank == 0)
     {
       for (int i = 1; i < Communicator.world.Size;i++ )
       {
         world.Send<String>("Hello", i, 0);
       }
       String data;
       for (int i = 1; i < Communicator.world.Size; i++)
       {
         world.Receive<String>(Communicator.anySource, 0, out data);
         // essayer
         // world.Receive<String>(i, 0, out data);
         // Pour constater l'ordre ou le désordre
         // d'arrivée des messages
         data += " recu par root";
```

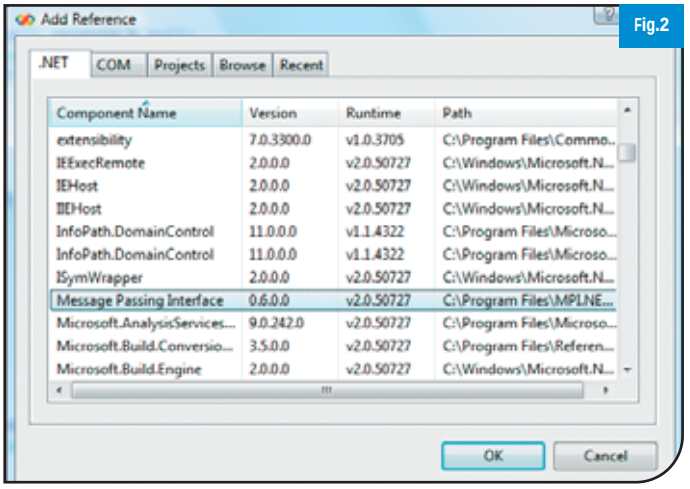


Fig.2

Ajout d'une référence sur l'assembly MPI.

```
          Console.WriteLine(data);
        }
      }
      else
      {
        String data;

        world.Receive<String>(Communicator.anySource, 0, out data);
        data += " World de ";
        data += world.Rank;
        world.Send<String>(data, 0, 0);
      }
    }
  }
}
```

Alors qu'en C le programmeur doit se soucier de près des types transmis via les messages, et qu'en Python, grâce au typage dynamique, la question ne se pose pas, nous voyons qu'avec C# les choses se passent très simplement grâce aux génériques. On remarquera encore l'usage, inévitable, du mot clé *out*, pour la récupération des valeurs reçues par messages.

5 PROGRAMMATION COLLECTIVE AVEC MPI.NET ET C#

Nous commençons par l'envoi d'un message à tous les membres d'un communicator, ici le communicator par défaut. On obtient ce dernier par la propriété *world* de l'objet Communicator.

```
using System;
using MPI;

class Program
{
 static void Main(string[] args)
 {
   using (new MPI.Environment(ref args))
   {
     int valeur=0, result;

     Communicator world = Communicator.world;
     if (world.Rank == 0)
       valeur = 10;
     // ici nous avons une barrière implicite
     // car l'exécution s'arrête tant qu'un message
     // n'a pas été envoyé à tout le monde
     world.Broadcast<int>(ref valeur, 0);
     // Puis l'exécution reprend
     result = valeur + world.Rank;
     Console.WriteLine("Résultat: {0}", result);
   }
 }
}
```

La méthode *Broadcast* est, elle aussi, générique. On notera bien l'emploi du mot clé pour le passage de paramètres par message. *Broadcast* se comporte comme une barrière implicite. Concrètement, cela signifie qu'aucun processus de *world* ne poursuivra son exécution au delà de l'appel de la méthode, tant que tous les processus, y compris l'émetteur *root* de rang 0) n'auront pas reçu le message émis. En programmation collective MPI il est très important de toujours garder à l'esprit que le processus *root* est impliqué dans les barrières implicites ou explicites. Pour illustrer cela, voyons un programme à barrière explicite qui ne fonctionne pas :

```
using System;
using System.Threading;
using MPI;

// Ce programme ne fonctionne pas car Barrier attend
// pour l'éternité le processus 0
class Program {
 static void Main(string[] args) {
   using (new MPI.Environment(ref args)) {
     int tempo = 0;
     Communicator world = Communicator.world;
     if(world.Rank != 0) {
       tempo = world.Rank * 1000;
       Thread.Sleep(tempo);
       world.Barrier();
       Console.WriteLine(
       "Processus {0} terminé", world.Rank);
     }
   }
 }
}
```

L'idée de départ de ce code est la suivante: chaque processus attendra pendant un délai dont la valeur est calculée en fonction de son rang. Jusque là tout va bien. Et on se dit, erreur classique, que puisque le processus *root* a la valeur 0, calculer son délai d'attente n'a pas de sens, et on place donc le code dans un bloc de test. Cette erreur est très courante en programmation parallèle. Le problème est que l'appel à *Barrier* attend tous les processus à l'endroit même de l'appel. En raison de la présence du test, cet endroit n'est jamais atteint par le flux de *root* et le programme attend à l'infini. Bien sûr, calculer le délai d'attente de *root* n'a pas de sens et un test doit être présent, mais l'appel à *Barrier* doit être sorti du bloc de code. Voici l'écriture correcte :

```
using System;
using System.Threading;
using MPI;

// Ce programme fonctionne car world.Barrier est invoqué
// a l'extérieur de tous tests de rang de processus
class Program {
 static void Main(string[] args) {
   using (new MPI.Environment(ref args)) {
     int tempo = 0;
     Communicator world = Communicator.world;
     tempo = world.Rank * 1000;
     if (world.Rank != 0) {
       Thread.Sleep(tempo);
     }
     world.Barrier();
     Console.WriteLine(
       "Processus {0} terminé", world.Rank);
```

```
    }
  }
}
```

Nous nous intéressons maintenant à une *réduction,* ce qui nous montrera un autre aspect de la programmation MPI avec .Net. En programmation parallèle, on appelle *réduction* une opération finale appliquée à chacun des résultats émis par les processus. Dans l'exemple qui va suivre, chaque processus émet son rang, et on fait la somme de tous les rangs :

```
using System;
using MPI;

class Program
{
  static void Main(string[] args)
  {
    using (new MPI.Environment(ref args))
    {
      int valeur, result;

      Communicator world = Communicator.world;
      valeur = world.Rank;
      result = world.Reduce(valeur, Operation<int>.Add, 0);
      if(world.Rank == 0)
        Console.WriteLine(result);
    }
  }
}
```

Le point intéressant de ce code est le passage à la méthode *Reduce* d'un délégué générique propriété de l'objet *Opération*.

6 PROGRAMMATION MPI AVANCÉE AVEC MPI.NET ET C#

Le standard MPI propose des fonctions telles que *MPI_Gather* ou *MPI_Scatter* dont le rôle est de collecter des résultats dans un tableau ou d'éclater un tableau à destination des processus, respectivement. En C la gestion des tableaux, et donc de leur mémoire, et surtout si les tableaux sont multi-dimensionnels, tourne très vite au cauchemar. C'est pourquoi nous nous sommes prudemment abstenus d'aborder ce point le mois dernier ;) Avec un langage comme C# les choses sont considérablement simplifiées. Voici le code qui collecte les rangs de tous les processus dans un tableau. Ce dernier est dimensionné automatiquement :)

```
using System;
using MPI;

class Gather
{
  static void Main(string[] args)
  {
    using (MPI.Environment env = new MPI.Environment(ref args))
    {
      Communicator world = Communicator.world;
      int[] ranks = world.Gather(world.Rank, 0);
      if (world.Rank == 0)
      {
        for (int i = 0; i < ranks.Length; ++i)
          Console.Write(" " + i);
        Console.WriteLine();
      }
    }
  }
}
```

Un peu plus fort maintenant, toujours avec Gather : chaque processus crée un tableau de valeurs, tableau qui est collecté dans un tableau. Le tableau final est donc multi-dimensionnel. Avec C# cela reste facile. Le lecteur trouvera le code GatherBis sur le site. Enfin, pour le plaisir, nous donnons encore sur le site un code qui illustre la méthode *Scatter.* Un tableau d'entiers est constitué avec des valeurs aléatoires, puis les valeurs sont dispatchées vers les processus. Le lecteur qui aura consulté ces exemples remarquera que Gather et Scatter se comportent comme des barrières implicites. C'est pourquoi dans le code elles se situent à l'extérieur de tout test de rang de processus ;) Par ces exemples nous avons constaté qu'utiliser un langage .Net pour la programmation MPI facilite grandement la tâche, ceci en raison de la gestion automatique de la mémoire, de la présence de la généricité et des types de relativement haut niveau. C'est très bien. Mais ne nous quittons pas sans revenir brièvement à Python, et écrivons, en un seul script l'équivalent des exemples Gather et GatherBis vus plus haut :

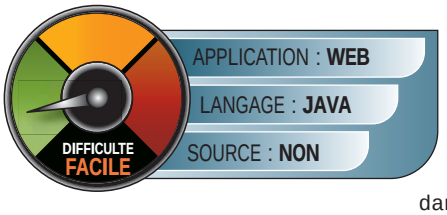
```
# -*- coding: iso-8859-1 -*-
import mpi
import sys

if __name__ == "__main__":
  if mpi.size != 4:
    print "Ce programme doit être lancé avec 4 processus"
    sys.exit(0)

  valeurs = []
  valeurs.append(mpi.rank + 10)
  valeurs.append(mpi.rank + 20)
  valeurs.append(mpi.rank + 30)

  results = mpi.gather(valeurs)
  # ou bien, pour avoir une
  # liste profonde:
  # results = mpi.gather([valeurs])
  if mpi.rank == 0:
    print results
```

Quelle concision en comparaison du code C# (et nous n'osons pas parler du code C)! Pour passer de la construction d'une liste (en Python la liste remplace le tableau) à une liste profonde, il suffit de modifier une seule ligne de code, en tout et pour tout! Python montre ici sa supériorité en raison de sa gestion automatique de la mémoire, de son typage dynamique et surtout de ses types de très haut niveau. En dépit de toutes les qualités de la plate-forme .Net, lorsque la vitesse d'exécution n'est pas critique ou lorsqu'il s'agit de mettre au point des algorithmes, Python, de l'humble avis de votre serviteur, est le choix à considérer en priorité.

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

Play! développez rapidement avec Java

Dans un précédent numéro, nous avions abordé le framework Grails. Aujourd'hui, nous allons nous intéresser à une nouvelle solution pour le développement d'applications web : Play!. Ce framework, distribué sous licence open source, est le fruit du travail d'un cabinet de conseil en architecture logicielle qui a su fédérer les bonnes pratiques du web.



Le framework Play! facilite la construction d'applications web en Java. Il s'agit d'une alternative aux solutions traditionnellement utilisées dans le domaine, à savoir les piles applicatives Java EE. Les concepteurs de Play! ciblent tout particulièrement la productivité des développeurs et font part d'un attachement particulier à l'architecture RESTful, l'architecture technique du web. La plate-forme Java est malheureusement connue pour sa faible productivité principalement due aux phases de compilations, packaging et de déploiements. Play! résout ce problème en optimisant les cycles de développement. Le développement avec Play! est réellement transparent. Le framework compile directement vos sources java et recharge les différents composants à chaud sans relancer le serveur. Vous pouvez ainsi éditer votre code et voir les modifications instantanément prendre effet comme ce serait le cas dans un environnement PHP. Les concepteurs de Play! l'ont élaboré à partir d'applications java réelles (iVelib…) et possèdent de nombreuses fonctionnalités qui permettront le développement d'applications web modernes.

ARCHITECTURE DE PLAY!

Une application Play! repose sur le patron de conception MVC : Modèle-Vue-Contrôleur. Rien de surprenant à cela, la majorité des frameworks web respectent ce principe. L'application est donc découpée en différentes couches qui permettent aux développeurs de mieux se focaliser sur le fonctionnel de l'application :

- Le modèle constitue la représentation des données que l'application manipule. Celui-ci est spécifiquement adapté au métier de l'application. Concrètement, ici nous sommes en présence d'objets Java ayant uniquement des attributs et un constructeur. Dans la pratique, il se trouve que ces objets sont persistés avec Hibernate.
- La vue est ce qui permet l'affichage du modèle. Les vues peuvent prendre différentes formes, plus ou moins complexes suivant l'application. Typiquement, avec Play!, on retrouve les formats standard du web, à savoir : HTML, XML ou JSON. Mais il est tout à fait possible de créer des vues permettant la génération d'un PDF, d'une image ou autres documents. Les vues sont majoritairement dynamiques, Play! utilisant une syntaxe proche des JSP.
- La gestion des événements est assurée par les contrôleurs. Dans le monde du web, ces événements sont typiquement des requêtes HTTP de différentes natures (GET, POST…). Elles permettent à l'application de modifier le modèle et l'affichage des vues. Avec Play!, l'implémentation est la suivante : chaque contrôleur est représenté par une classe Java dont chaque méthode correspond à une requête. Le mapping entre ces requêtes et les URL se fait dans un simple fichier texte pour un maximum de souplesse.

Une application Play! possède deux fichiers de configuration. Ceux-ci sont placés dans le dossier conf. Il n'y a rien de plus à savoir pour configurer votre application. Voici les deux fichiers en question :

- **application.conf**
  - Fichier de configuration principal. Il contient notamment la configuration de la base de données, *memcached* et bien sûr le port d'écoute du serveur.
- **routes**
  - Définit les routes, c'est-à-dire le mapping entre les url et les contrôleurs.

MISE EN PRATIQUE : UN CARNET D'ADRESSES EN LIGNE

Nous allons créer une petite application qui nous permettra d'aborder les concepts fondamentaux de Play par la pratique. Le choix s'est porté sur un outil web permettant la gestion de contacts. Celui-ci permettra la création, l'affichage et la suppression de contacts. Chaque contact aura des méta données qui lui sont propres : numéro de téléphone, adresse… Avant de commencer, nous devons avoir un environnement java configuré. Ici, nous utiliserons Netbeans, mais vous êtes libre d'utiliser Eclipse ou un éditeur texte. Play! ne nécessite aucune configuration spécifique pour fonctionner, récupérez l'archive sur www.playframework.com. Play! fonctionne en environnement Unix, les scripts seront donc fonctionnels nativement sous Linux et Mac OS, en revanche les utilisateurs de Windows devront se procurer Cygwin pour l'exécution des commandes qui suivent. Nous pouvons à présent créer notre premier projet. Pour cela, nous allons utiliser la commande play, qui nous permettra également par la suite de générer un projet Netbeans ainsi que de lancer l'application.

```
> play new carnet
```

La commande précédente crée un nouveau projet dans le dossier

Web oriented architecture

La WOA se définit par un ensemble de protocoles Web avec lesquels on construit une application. Ces protocoles ont la caractéristique commune d'être dynamique, d'assurer la montée en charge, et d'être interopérables. Lorsque l'on parle de WOA, on fait référence essentiellement à HTTP et à XML. La WOA se positionne directement dans l'esprit des architectures SOA, mais propose en plus un modèle d'implémentation technique basé sur le Web. Play! repose sur cette approche, il s'agit donc d'un framework sans état qui met en avant une architecture RESTful.

carnet. Le nom du projet vous sera ensuite demandé. Pour générer le projet Netbeans, utilisez la commande suivante :

```
> play netbeansify carnet
```

De la même manière, il est possible de créer le projet pour Eclipse avec la commande eclipsify. Nous pouvons maintenant lancer cette application.

```
> play start carnet
```

L'application web sera exécutée sur le port 9000. Vous pouvez y accéder avec votre navigateur. Par la suite, nous n'utiliserons plus ces commandes. Vous pouvez arrêter l'application et ouvrir le projet avec votre éditeur.

CRÉATION DE NOS PAGES WEB

Pour commencer nous allons définir les routes de notre application. Conformément à l'esprit de Play!, on choisira une syntaxe RESTful. Éditez le fichier routes du dossier conf. Les méthodes GET correspondent aux affichages tandis que les méthodes POST nous permettront d'effectuer des actions côté serveur.

```
GET / Application.index
POST     /{id}/delete Application.delete
GET     /contact/new Application.newContact
POST    /contact/save Application.save
GET     /contact/{id} Application.contact
```

A présent, nous allons créer nos pages, il y en a trois en tout : accueil, ajout d'un contact et visualisation d'un contact. La page d'accueil comportera la liste des contacts ainsi qu'un lien pour ajouter un contact au carnet d'adresses. Vous remarquerez la syntaxe spécifique des pages, il s'agit en réalité de code Groovy avec quelques tags spécifiques :

```
<h2>Liste des contacts</h2>

#{if contacts.size() == 0}
<br />
Aucun contact
#{/if}

<table class="contact-table">
#{list items:contacts, as:'contact'}
<tr>
    <td class="light"><a href="@{Application.contact(contact.
id)}">${contact.name}</a></td>
   <td class="light"><a href="mailto:${contact.email}">${contact.
email}</a></td>
     <td><form action="@{Application.delete(contact.id)}" met
hod="POST"><input type="submit" value="Supprimer" /></form></td>
</tr>
#{/list}
</table>
```

La page d'ajout de contact comporte un simple formulaire qui envoie les données afin de créer le contact et de le persister.

```
<form method="POST" class="cssform" action="@{Application.save()}">
    <p>
        <label for="name">Nom</label>
       <input type="text" id="name" name="contact.name" value
="${contact.name}" />
        <label for="email">Email</label>
       <input type="text" id="email" name="contact.email" value
="${contact.email}" />
        <label for="phone">Téléphone</label>
       <input type="text" id="phone" name="contact.phone" value
="${contact.phone}" />
        <label for="address">Adresse</label>
       <input type="text" id="address" name="contact.address"
 value="${contact.address}" />
    </p>

    <input type="submit" value="Enregistrer" />
</form>
```

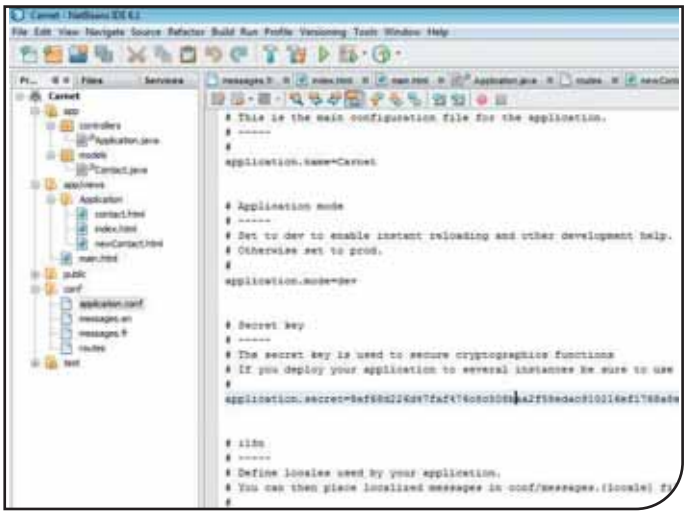
La dernière page que nous allons développer est celle qui affiche les informations d'un contact. Nous verrons juste après comment les informations sont poussées sur la page.

```
<h2>Contact : ${contact.name}</h2>

<p class="contact-info">
     <label for="email">Email</label><span>${contact.email}</
span><br />
   <label for="phone">Téléphone</label><span>${contact.phone}
</span><br />
   <label for="address">Adresse</label><span>${contact.address}
</span><br />
</p>
```

ECRITURE DU CODE MÉTIER

Notre couche modèle de données est ici limitée à sa plus simple expression. Play! repose sur JPA et offre un support permettant une implémentation plus rapide du code métier :

```
package models;

import javax.persistence.Entity;
import play.db.jpa.JPAModel;
```



```
@Entity
public class Contact extends JPAModel {

    public String name;

    public String email;

    public String phone;

    public String address;
}
```

LE CONTRÔLEUR

La partie la plus importante de l'application se situe ici. Ce sont les contrôleurs qui ont le rôle le plus important. Pour simplifier les choses nous n'en avons qu'un seul dans cet exemple. Cependant sur les applications plus consistantes, il est primordial de bien séparer le code.

```
package controllers;

import java.util.List;
import models.Contact;
import play.mvc.Controller;

public class Application extends Controller {

    public static void index() {
        List<Contact> contacts = Contact.findAll();
        render(contacts);
    }

    public static void contact(Long id) {
        Contact contact = Contact.findById(id);
        render(contact);
    }

    public static void delete(Long id) {
        Contact contact = Contact.findById(id);
        contact.delete();
        index();
    }

    public static void newContact() {
        Contact contact = new Contact();
        render(contact);
    }

    public static void save(Contact contact) {
       contact.save();
       index();
    }

}
```

RENDONS L'APPLICATION MULTI-LANGUE

Play! permet une internationalisation rapide des applications. Pour cela, il s'appuie sur le standard *i18n* en offrant une classe de support dédiée. Afin de supporter l'anglais et le français, nous allons créer deux fichiers dans le dossier conf : *messages.fr* et *messages.en*. Ceux ci seront remplis comme de simples fichiers de propriétés, par exemple :

```
home=Accueil
```

Ensuite au niveau du fichier de configuration *application.conf,* nous mettrons la propriété suivante afin d'activer le support de l'internationalisation :

```
application.langs=fr,en
```

Au niveau de nos pages html, il ne nous reste plus qu'a utiliser le tag pour chaque champ. Par exemple :

```
&{'home'}
```

Afin de laisser l'utilisateur choisir la langue, nous ajouterons deux boutons dans notre template :

```
<div id="lang">
    <a href="@{Application.changeLanguage('fr')}"><img src="/
public/images/french_flag.jpg" /></a>
    <a href="@{Application.changeLanguage('en')}"><img src="/
public/images/english_flag.jpg" /></a>
</div>
```

Vous prendrez soin d'ajouter la route suivante :

```
GET     /changeLanguage/{lang}  Application.changeLanguage
```

Ainsi que la méthode qui effectuera le changement de langue. Celle-ci redirigera vers l'accueil :

```
    public static void changeLanguage(String lang) {
        Lang.change(lang);
        index();
    }
```

TESTONS L'APPLICATION

En ce qui concerne les tests unitaires, Play! Repose sur JUnit 4. Les tests seront développés dans le dossier test et vous pourrez utiliser la classe de support ApplicationTest. Play! vous permettra de lancer les tests par la commande suivante :

```
play test
```

Les tests seront lancés dans un mode particulier, ainsi il vous sera possible de définir une configuration spécifique pour le bon déroulement des tests, par exemple :

```
%test.db=mem
%test.jpa.ddl=create-drop
```

Pour effectuer des tests automatisés de votre interface graphique, je vous conseille Selenium, une solution open source basée sur Firefox qui produit des résultats remarquables.

CONCLUSION

Comme vous avez pu le constater, Play! est une solution facile à prendre en main et efficace en terme de productivité. Vous vous apercevrez à l'utilisation que son serveur est particulièrement performant et qu'il permet une bonne montée en charge. Bien que récent, ce projet est déjà utilisable en production et son avenir est réellement prometteur.

■ Loïc **Guillois**

DEVELOPPEZ VOTRE SAVOIR-FAIRE

Programmez ! *est le magazine du développement*

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

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

11 numéros par an : **45 €***

soit **3 Numéros GRATUITS**

**Tarif France métropolitaine*

Choisissez votre formule

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

+ Abonnement INTÉGRAL

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

Cette option est réservée aux abonnés pour 1 an au magazine, quel que soit le type d'abonnement (Éco, Numérique, Etudiant). Le prix de leur abonnement normal est majoré de 12 € (prix de lancement, identique pour toutes zones géographiques). Pendant la durée de leur abonnement, ils ont ainsi accès, en supplément, à tous les anciens numéros et articles /dossiers parus.

OUI, *je m'abonne*

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

PROGRAMMEZ

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

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

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

☐ M. ☐ Mme ☐ Mlle Entreprise : ..... Fonction : .....

Nom : ..... Prénom : .....

Adresse : .....

Code postal : ..... Ville : .....

Tél : ..... E-mail : .....

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

A remplir et retourner sous enveloppe affranchie à :
Programmez ! - Service Abonnements - 22 rue René Boulanger - 75472 Paris Cedex 10.

abonnements.programmez@groupe-gli.com

PROgrammez ! Le magazine du développement

Offre limitée, valable jusqu'au 30 novembre 2008

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

Conformément à la loi Informatique et Libertés du 05/01/78, vous disposez d'un droit d'accès et de rectification aux données vous concernant.
Par notre intermédiaire, vous pouvez être amené à recevoir des propositions d'autres sociétés ou associations.
Si vous ne le souhaitez pas, il vous suffit de nous écrire en nous précisant toutes vos coordonnées.

PROG 113

Devenez un petit génie du jeu vidéo... en Java

Code source sur www.programmez.com

2e partie

Le mois dernier, dans Programmez ! n°112, nous avions abordé les frames et le projet du jeu de course. Aujourd'hui nous allons plus loin avec les threads et les classes internes. Bonne course !

La course du courage : Les voitures

Thread et classes internes

Pour créer un Thread, vous devez d'abord programmer une classe interne. Pour faire cela, saisissez le code suivant à l'intérieur d'une classe, mais en dehors de toute méthode ou de tout constructeur :

```
public class <nom> extends Thread
{
}
```

<nom> peut être remplacé par un nom quelconque différent du nom de la classe principale. Dans cette nouvelle classe, il faut créer une méthode *run,* obligatoire pour que le programme puisse compiler :

```
public void run(){}
```

Dans cette méthode, nous allons faire avancer une des voitures. Il faut y créer une boucle *while* infinie :

```
while (true)
{
}
Dans la boucle while, ajoutez le code suivant :
try
{
  //code
}
catch (Exception e)
{
  break;
}
```

Ces quelques lignes permettent de vérifier s'il y a des erreurs dans le bloc encadré par *try.* Si une erreur survient, l'ordinateur exécute le code se trouvant dans le bloc encadré par *catch*. Dans ce cas, on appelle break, ce qui permet de sortir de la boucle infinie.

Les contrôles faisant avancer la voiture doivent être placés à la place de / / code. Une fois ce Thread terminé, appelez-le depuis le constructeur avec la syntaxe suivante, dans laquelle *Mouvement* est le nom de votre classe interne :

```
Mouvement m = new Mouvement();
m.start();
```

Créer le jeu

Commencez par dessiner un *Rectangle* pour représenter la voiture). Puis, programmez un Thread (grâce à une classe interne). Ajoutez la méthode *run,* la boucle *while* et le code *try*.... Créez ensuite une variable globale pour contenir la vitesse de la première voiture. Dans le code *try*..., augmentez lentement la vitesse jusqu'à la vitesse finale (ici, 4). Puis, faites bouger la voiture en modifiant la valeur **y** de son rectangle en y ajoutant la valeur de la vitesse de la voiture. Pour finir, rafraîchissez l'écran et ajoutez un délai (grâce à Thread.sleep(75);). Lorsque tout ceci fonctionne, essayez d'ajouter la deuxième voiture (**Vous trouverez le code complet sur notre site**). La figure 2 illustre notre projet de course du courage. Le projet est bien entendu personnalisable !

La course du courage : collisions !

Détection de collisions

La détection de collisions (figure 1) est simple en Java. Comme vous avez créé tous vos graphiques avec des classes *Rectangle,* détecter des collisions est facile : la classe *Rectangle* fournit une méthode *intersects.* Pour vérifier une collision entre r1 et r2, on utilise cette méthode de la manière suivante :

```
if(r1.intersects(r2))
```

Pour gérer plusieurs collisions, vous pouvez utiliser les opérateurs && (et) et || (ou) dans la condition du bloc *if.* Par exemple, pour voir si r1 est en contact avec r2 ou avec r3, on écrit :

```
if(r1.intersects(r2) || r1.intersects(r3))
```

KeyListener

La classe *KeyListener* permet de recevoir des saisies clavier venant de l'utilisateur. Celles-ci peuvent par exemple servir à contrôler les directions des voitures. Pour utiliser un *KeyListener,* il faut ajouter les éléments suivants à la déclaration de votre classe (ligne public class...) :

```
implements KeyListener
```

Cela s'applique aussi aux classes internes dérivées de Thread. Puis, dans le constructeur (ou dans la méthode run si vous utilisez des Thread), ajoutez cette ligne :

```
addKeyListener(this);
```

Elle indique au clavier de se "réveiller" et de commencer à écouter les commandes. Vous devez aussi ajouter trois méthodes à votre code :

```
public void keyPressed(KeyEvent e){}
public void keyReleased(KeyEvent e){}
public void keyTyped(KeyEvent e){}
```

Ces méthodes sont appelées automatiquement lorsqu'une touche est tapée. *keyPressed* est appelée lorsque l'utilisateur appuie sur

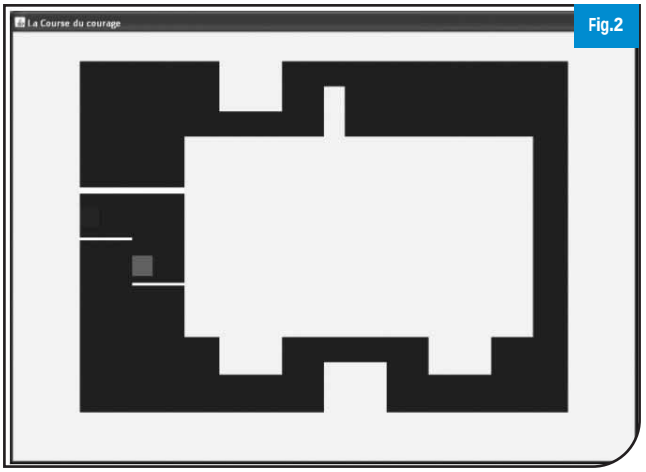


Fig.2

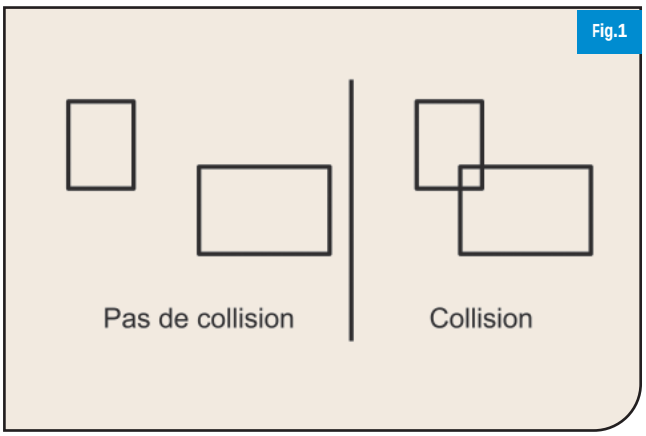


Fig.1

une touche. *keyReleased* est appelée lorsqu'une touche est relâchée. *keyTyped* est appelée lorsqu'une touche est pressée puis relâchée. *keyTyped* est probablement la méthode que vous utiliserez le plus. Vous pourrez alors modifier le jeu en changeant certaines valeurs comme la direction de la voiture en fonction de la touche. Pour savoir quelle touche a été tapée, on utilise :

```
e.getKeyChar()
```

Créer le jeu

Créez des variables globales de type *int* pour la direction de la voiture. Si leur valeur est 0, la voiture va vers le haut, si sa valeur est 1, la voiture va vers la gauche, etc. Pour simplifier, créez des constantes avec l'attribut final ; par exemple une valeur de type *int* nommée HAUT dont la valeur est 0. Pour savoir si la voiture va vers le haut, il suffit alors de comparer sa direction avec HAUT plutôt qu'avec 0. Puis, utilisez un *KeyListener* pour que le joueur puisse modifier sa direction. Modifiez la direction en testant la direction actuelle de la voiture et modifiez x si la voiture va vers la gauche ou vers la droite, y si elle va vers le haut ou vers le bas. Vérifiez également si le joueur est entré en collision avec un mur ou avec une autre voiture. Si c'est le cas, la vitesse du joueur doit être négative pour que la voiture recule avant de pouvoir reprendre de la vitesse. Le code complet est disponible sur le site de Programmez !. Suite et fin le mois prochain.■

Extrait de : **Ian Cinnamon**, *devenez un petit génie des jeux vidéo,* Pearson, 2008. Avec l'aimable autorisation de l'éditeur.

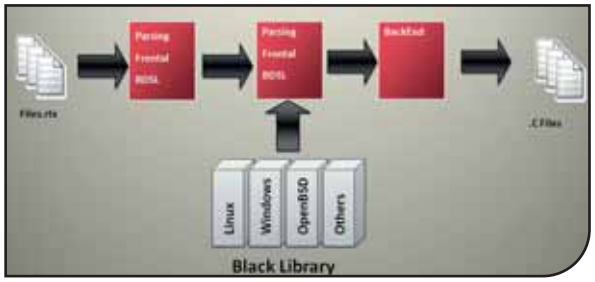
Epitech
Innovative Projects
Une promotion 2009 imaginative !

François **Tonic**

Club de photographie Ephémère. Photo Jimmy Durand Wesolowski.

Les 17 et 18 octobre dernier, sur son campus, l'Epitech organisait l'Innovative Projects. Il s'agissait de montrer une trentaine de projets de fin d'études, internes à l'école ou liés à des projets avec des partenaires, entreprises, institutions. Il faut avouer que nous avons été particulièrement impressionnés par plusieurs d'entre eux.

Notons tout de suite quelques tendances dans les développements : Rails, Python, PHP, .Net, C++. Java étaient bien entendu présents mais un peu en retrait.L'un des projets que nous avons retenu est **Rathaxes** (http://www.rathaxes.org/). Il s'agit de créer un environnement de génération de pilotes de périphériques multi- plates- formes. Pour cela, l'équipe a conçu un langage de description, on passe ensuite par un compilateur qui s'appuie sur des templates spécifiques à chaque système, sous Windows ce sera le DDK. Pour pouvoir réaliser un tel environnement, RTX utilise le parser CodeWorker. Et les responsables ont conçu le langage de description comme un DSL pour mieux structurer l'environnement. L'architecture est donc assez simple : écriture du code RTX, compilation, génération des fichiers .c.



Architecture de RTX

Autre projet particulièrement intéressant, **Scope.** Réalisé dans un cadre européen et sous l'égide de l'INRIA, Scope doit permettre de créer des voitures à conduite automatique, capables de se guider elles-mêmes par rapport aux autres voitures. Techniquement intéressant, Scope repose sur deux serveurs, deux systèmes Linux et un système Windows XP. Un boîtier, routeur wifi, sert de serveur de communication basé sur le protocole OLSR (Optimized Link State Routing). L'équipe a utilisé un système Linux embarqué (OpenWRT). Dans la prochaine version en cours de développement, il est prévu d'utiliser le GPS, la 3G et le passage à IPv6. Toujours côté écologie, le projet **DriveMeUp** est un système de covoiturage. Il s'agit de trouver des voitures participant au programme selon les trajets des personnes, de localiser sur une carte les voitures, de définir des trajets, etc. L'application a été développée en Rails et utilise une base MySQL. Pour la cartographie, on utilise Google Maps et Street Views est en cours d'implémentation. La solution sera gratuite au lancement (http://www.drivemeup.fr/) mais le projet vise aussi (et surtout) l'entreprise.

Les dernières technologies sont mises en avant !

Une équipe d'étudiants s'est aussi lancée dans la conception d'un webos, à l'image d'exo Platform. Le projet **Touareg** est donc un webos permettant d'accéder à ses données, à son espace de travail de n'importe où. Seule condition : une navigation compatible Silverlight car le projet est conçu en .Net (C#) et Silverlight 2.0. D'ailleurs, l'un des intérêts du projet fut la collaboration poussée entre développeur et designer (sous Blend). Le projet dispose d'un agenda, d'une messagerie instantanée, d'une gestion de fichiers, etc. Une API sera prochainement disponible pour les développeurs afin de créer ses propres modules ! Un peu dans le même esprit, nous avons déniché le projet **Everlastick** (http://everlastick.com/). L'objectif est d'avoir avec soi son bureau sur une clé (cryptée) USB. Il intègre par défaut différentes applications (calendrier, navigateur basé sur WebKit, tableur, etc.). Une API sera disponible pour étendre l'environnement mobile. L'ensemble a été codé en C++ et pour l'interface, il s'agit de Qt 4.4. Du beau travail ! Dans le domaine de l'optimisation de l'utilisation des ressources des machines, le projet **Aditam** (http://www.aditam.org/) se propose de généraliser la distribution des tâches sur plusieurs machines et non plus de réserver cela à quelques applications. La partie serveur et les agents sont codés en Python, par contre la partie web l'a été en PHP. Une librairie de mapping de données est utilisée. Les tâches étant stockées dans une base de données. Il fonctionne sous MacOS X, Linux et Windows. La sécurité n'a pas été absente, loin de là. On citera ainsi le projet **NSAT** (nsat.fr). Il s'agit d'un outil d'audit de sécurité de son réseau. Il vérifie l'ensemble des protocoles, accès, versions des logiciels, etc. L'outil a été codé en Java et fonctionne sous Linux et Windows. Citons encore le projet **Nasame.** Il combine un Shell et une navigation. Il doit pouvoir remplacer son explorateur et pour les plus exigeants être modulaire pour l'adapter à ses besoins ! Il a été écrit en C# et Gtk# (Mono). Approche intéressante ! Enfin, terminons avec **Beingenious.** Il s'agit d'une société innovante créée par plusieurs étudiants, avec l'aide d'industriels. Son but est d'aider des projets innovants et technologiquement nouveaux en apportant audit complet, conseils, business plan, etc. Mais la société peut aussi investir dans un projet et le développer en interne. Une des premiers projets réalisés est mesphotosoffertes.com. Un grand bravo à tout(e)s les étudiant(e)s. Un excellent cru 2009 pour l'Epitech. ■