PROGRAMMEZ !
Le magazine du développement
PROgrammez !
www.programmez.com
mensuel n°124 - novembre 2009

Le match des applications internet "riches"

Photo © istockphoto.com/Auteur Ljupco

Flex
JavaFX
Silverlight

Programmer avec quel langage ?

Java

Microsoft .NET

php

DOSSIER
Windows 7
(2e partie)

- ✔ Les nouveautés C++
- ✔ Intégrer le kit d'assistance
- ✔ IE 8 pour booster vos sites

WEB
La révolution Google Wave

JAVA
Utiliser OpenOffice avec Java

ANDROID
Personnalisez votre Nabaztag

Linux
Les signaux sous Linux

Outil
Créer son compilateur

Données
Maîtrisez Hibernate

Virtualisation
Utilisez VirtualBox pour vos développements

M 04319 - 124 - 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

PROgrammez! le magazine du développement

numéro 124 - novembre 2009

sommaire

éditoll

En ton langage tu croiras…!

Ici même, il y a un mois, nous abordions « l'utopie du langage » et son embonpoint constant. Nous retournons le propos en nous demandant comment choisir le bon langage, au bon moment, pour le bon développement. Vous verrez qu'à cette question simple, la réponse l'est beaucoup moins.

Selon Salomon Brys, étudiant à l'Epitech et grand spécialiste du C++, l'objectif du développeur est de chercher les optimisations possibles du langage choisi. Et quand il y a deux langages faisant la même chose pour un même développement, l'expert regarde les optimisations et la facilité de programmation (syntaxe, approche, fonctions). Cette dernière constitue-t-elle un argument massue pour un étudiant, un développeur, lors du choix d'un langage ? Pas forcément. Ainsi à l'Epitech, l'étudiant plonge dans la piscine C++ pour écrire, vivre, respirer objet… Ce qui ne l'empêche pas d'aller voir ailleurs si j'y suis… mais avec une bonne fondation C++…

Le choix d'un langage est-il pragmatique ou idéologique ? Les deux. Il y a le « bon » et le « méchant », pour faire simple. Il arrivera souvent qu'un développeur PHP jette un regard noir sur d'autres langages web, n'y voyant pas l'intérêt. Mais on retrouve finalement ce sectarisme avec C#, VB, C++, ASP.Net, Ruby, etc.
Cependant, en milieu professionnel, le pragmatisme bouscule souvent les préjugés du jeune diplômé. Le projet, le client final oriente régulièrement le choix de la plate-forme et du langage. On se fond dans un moule. Pis, dans certains contextes, le langage, les outils s'imposeront d'eux-mêmes, ou bien on s'enfermera dans une plate-forme comme avec les RIA. Changer de langage en cours de route d'un développement ? Si on aime la programmation S-M pourquoi pas ?

> **“S'il existe une quantité impressionnante de langages pour faire tout et n'importe quoi, seule une poignée émerge réellement”**

On le constate à la lecture des offres d'emplois, à la vitalité des communautés, aux outils disponibles. Choisir un langage pointu et performant, c'est bien, mais s'il est confidentiel et très peu actif, pour un développement, c'est un risque qu'en milieu professionnel personne, ou presque, ne voudra prendre. Ruby s'est fait sa place sur le Web avec Rails, mais bien peu sur le desktop. Le langage fonctionnel attire, intrigue mais demeure largement ignoré car sortant de la demande du marché.

Alors, est-ce que l'on garde les stéréotypes « *C++, vrai développeur* », « *VBiste, développeur amateur* » ? Non, cette vision a évolué. Même si le C++ demeure LE langage performant par définition et incontournable pour la 3D, les jeux, l'embarqué, l'enfoui.
Eh oui, il y a des domaines qui ne changeront jamais. Heureusement. J'aurais une petite peur si je devais voir s'afficher « *Mise à jour de Java 6… merci de patienter* » ou encore « *L'applet Starter se charge* » en appuyant sur le bouton démarrage de ma voiture. Sauf si le café était inclus.

François Tonic
Rédacteur en chef

PROgrammez! Le magazine du développement

Rédaction : redaction@programmez.com
Directeur de la Rédaction : Jean Kaminsky
Rédacteur en Chef :
François Tonic - ftonic@programmez.com
Ont collaboré à ce numéro : F. Mazué, C. Remy, W. Bories, L. Guillois, J-B. Boisseau.
Experts : F. Goldgewicht, P. de Saint Staban, C. Combelles, R. Clark, E. Vernié, P-L. Coll, H. Darmet, S. Saurel, C. Phu, E. Robles, A. Berardino.
Illustration couverture : Microsoft, Photo © istockphoto.com/Auteur Ljupco
Publicité : Régie publicitaire, K-Now sarl
Pour la publicité uniquement :
Tél. : 01 41 77 16 03 - diff@programmez.com
Editeur : Go-02 sarl, 21 rue de Fécamp 75012 Paris - diff@programmez.com
Dépôt légal : à parution - Commission paritaire : 0712K78366 ISSN : 1627-0908
Imprimeur : ETC - 76198 Yvetot
Directeur de la publication : J-C Vaudecrane

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

L'INFO PERMANENTE
WWW.PROGRAMMEZ.COM

PROCHAIN NUMÉRO
N°125 - décembre 2009, parution 28 novembre

✔ **Maîtriser PHP !**
Sécurité, performance, optimisation et bonnes pratiques

✔ **Windows Phone**
La contre-attaque !

✔ **Langages**
C # 4.0, VB 10, Visual Studio 2010 : .Net c'est plus fort que toi !

■ **Qt,** la librairie d'interface C++ multi-plate-forme, a été portée par Nokia sur

son Maemo 5 et sera incluse par défaut dans la v6.

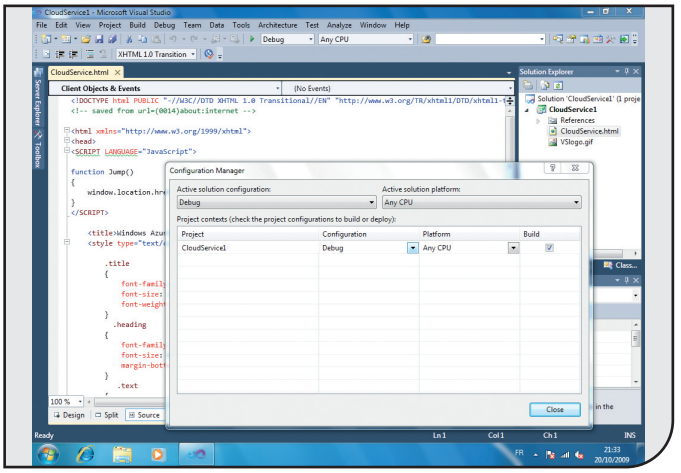
■ **CodeFluent**, édité par Softfluent, ajoute une couche Silverlight pour la partie interface. On peut ainsi, grâce à ce support, disposer de règles métiers sur le client ou encore avoir le binding Siliverlight et générer dynamiquement des interfaces asynchrones.Cela se réalise via le Siliverlight Automatic Asynchronous Proxy.

■ **Turbogears** est un framework web en Python,utilisant l'approche MVC. Le but est d'avoir des applications web plus rapides. Turbogears contient plusieurs composants existants : Mochikit pour la partie javascript, Kid pour la partie Template, CherryPy pour faciliter les entrées / sorties et SQLObject pour les données. Deux lignées sont disponibles : la 1.1.x et la 2.x. Site : http://turbogears.org

■ **Infragistics** dévoile **Gort** pour stimuler l'ingéniosité des développeurs sur l'interface graphique ! Le but est de défendre votre œuf et de découvrir, en cas de succès, tout le potentiel des composants NetAdvantage. Site : www.labs. infragistics.com/gort

Visual Studio 2010 : whaou !

C'est ce que l'on peut dire en installant la version Ultimate de Visual Studio 2010 bêta 2, disponible depuis le 19 octobre dernier. Il faut avouer que c'est du très lourd : test, Team System, partie architecte (avec notamment la modélisation UML), outils d'analyses, versioning, Azure, F#, développement web (Asp, Silverlight...)... Pour cette version majeure, Microsoft a simplifié la gamme : Professional, Professional avec MSDN, Premium avec MSDN et enfin l'édition Ultimate qui comprend tout (pour la modique somme de 11 924 dollars US).



Azure en standard

VS 2010 b2 propose une étape importante pour l'environnement de développement. Il s'intègre à Windows 7, supporte Sharepoint 2010, Office 2010, .Net 4 et tout l'arsenal pour le développement parallèle. Côté multitarget, on a le choix de .Net 2 à 4. Détail particulièrement intéressant, avec l'arrivée du cloud dans les templates projets, chaque édition de VS 2010 arrive avec un volume d'heures d'utilisation Azure, allant de 50 à 250 ! Voici une excellente nouvelle. Désormais VS 2010, et .Net 4 qui l'accompagne, arrivent avec 4 principaux langages : C#, VB, C++ et F#. C'est la grande venue sur la plate-forme .Net de la programmation fonctionnelle. A cela s'ajoutent toute une couche de programmation et de librairies parallèles dont ParallelFX et l'ensemble des fonctions parallèles. Cette v2 apparaît assez véloce et performante. Côté extensibilité, Microsoft veut remettre en avant les capacités d'extension et de personnalisation de l'environnement. Bien entendu, via les plug-in tiers, mais aussi avec la capacité d'écrire ses propres extensions ou même de bâtir son propre Visual Studio, via le kit de développement VS ou encore Visual Studio Shell. Parmi les absents de cette bêta 2, le développement mobile et toute la partie Mesh.

N'oublions pas non plus que VS 2010 inaugure un moteur graphique et une interface basée sur WPF. Si des craintes avaient été émises, avec raison, avec les pré-versions, notons que les premiers tests de la bêta 2 sont plutôt concluants. Réactivité correcte, voire bonne, un accès à la toolbox plutôt rapide, changement de fenêtre immédiat, etc. A l'usage, ce passage à WPF et la nouvelle ergonomie devraient plaire aux développeurs malgré les 6 Go de la version intégrale... D'autre part, le support par défaut du multi-écran (si, si) va enfin apporter un réel confort de développement et de productivité car on se sentait parfois à l'étroit. Dans cette version, tout ou presque a été revu, amélioré, réécrit.

Disponibilité prévue : fin mars 2010 !

agenda \\

NOVEMBRE

• Le 02 novembre à Lyon, le 03 Novembre à Nantes **Microsoft Days 2009** / Rencontres techniques
Pour booster vos compétences avec plus de 20 sessions techniques. http://msdn.microsoft.com/fr-fr/microsoft-days.aspx#

• Le 03 novembre à Lyon, Microsoft, **Journées Plateforme Editeurs**. https://www.microsoft.com/france/External_code_cms/msdays/agenda.aspx#plateforme

• Du 12 au 13 Novembre 2009, Cité des Sciences de La Villette – Paris
9e édition du Forum PHP, organisé par l'AFUP
Pour faire le point sur les évolutions fonctionnelles et techniques, communautaires et entreprises de PHP. http://www.afup.org/pages/forumphp2009/

• Du 12 novembre au 08 décembre,
Tour de France PC SOFT. www.pcsoft.fr

• Du 17 au 19 novembre 2009, au parc des expositions de Paris-Nord Villepinte, l'Internet des objets tient salon sur **Cartes & IDentification 2009** http://www.cartes.com

• Le 17 Novembre, Paris La Défense, **Valtech Days 2009.** Evénement axé autour de 3 thèmes : Agilité, E-business et Technologies. http://www.valtech-days.fr

• Le 26 novembre, Paris, Centre des Congrès Microsoft. **MD Day 2009,** L'ingénierie Model-Driven : une approche pragmatique, des projets concrets, des retours d'expérience positifs. http://www.mdday.fr/

Conférence IBM Rational : Jazz et Agile

IBM organisait la Rational Software Conférence 2009, pour la dernière fois à la Tour Descartes à Paris La Défense, avant son déménagement.

Tous les grands responsables de l'éditeur étaient présents : Danny Sabbah (grand Ordonnateur de Rational), Martin Nelly (le directeur technique) ainsi que Peter Morykon, spécialiste de la sécurité chez Rational.

Du Saas pour 2010

La session plénière n'a pas apporté de révélation sur les futurs outils de l'éditeur. L'un des rappels de cette session fut la complexité du monde logiciel et des développements toujours plus courts, plus complexes, des multiplications des terminaux ainsi que le manque de flexibilité des processus de développement. Rational a alors rappelé un des crédos : rendre plus flexible le développement, faire collaborer les développeurs, les équipes. Et cela passe notamment par sa plateforme Jazz, bâtie pour répondre à ces demandes. Mais sans pour autant faire peur avec la notion d'usine à logiciels ou encore l'industrialisation des développements. Car le développeur reste au cœur du dispositif mais on lui permet de se concentrer sur sa mission en automatisant certaines tâches, ou du moins en les simplifiant dans une certaine mesure. Et surtout un rappel simple mais essentiel : les outils ne font pas tout ! La sécurité a été l'autre fil rouge de la plénière en rappelant les risques sécuritaires des applications. Pour l'éditeur, le manque de sécurité est un bug qu'il faut résoudre. Malheureusement, le développeur reste insuffisamment sensibilisé et les contraintes temps et financières bloquent souvent le développement sécurisé car il est plus coûteux (en théorie).

Pour Rational, ces outils doivent aider l'entreprise à se moderniser sans pour autant rompre totalement avec le passé, l'existant qui pose de nombreuses contraintes. Et l'un des enjeux est de garder la connaissance des développeurs. Trop souvent, quand un développeur part de l'entreprise, celle-ci perd la connaissance de la personne. En conclusion de la session, le Saas et le Cloud ont été longuement mentionnés comme avenir de Rational avec la disponibilité de services en ligne Rational pour début 2010... L'éditeur essaie de repenser la notion de produit, de solutions, dixit Dany Sabbah. L'ultime objectif sera de déplacer la plate-forme collaboration toute entière sur le web. Il s'agit d'un chantier ambitieux qui prendra environ deux ans. Et au cœur de cette réflexion d'ensemble : Jazz.

L'agilité à l'honneur

Durant toutes les journées, des sessions de 45 minutes se succédaient sur la qualité, l'agilité, la gouvernance des développeurs, la modernisation, la réduction des coûts ou encore le développement de systèmes complexes. L'agilité était à l'honneur notamment via les sessions de modélisations agiles, de collaboration et de Lean. La modélisation agile fut une des sessions les plus intéressantes par l'intérêt d'une telle combinaison pour les équipes de développement sans occulter les limites et les risques liés à cette approche.

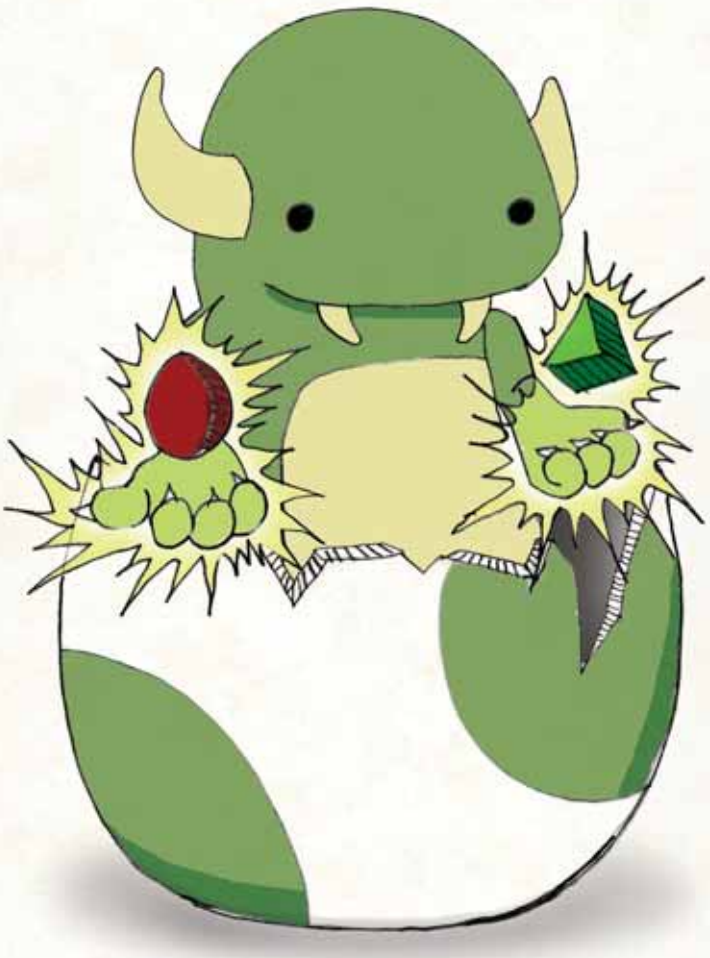
La modélisation agile et Jazz

La modélisation agile est le fait de combiner la modélisation UML et les notions d'agilité (Scrum, XP, etc.). Mais avant tout, il s'agit de fournir un modèle : simple, compréhensible, avec une implication des utilisateurs, être capable de recueillir des commentaires pour améliorer le modèle, accepter les changements. Le modèle ne doit surtout pas être figé et chaque artefact du modèle doit avoir une raison d'être, une utilité. Bref, rendre le modèle agile. Et cela n'est pas aussi simple qu'il y paraît, notamment à cause de l'UML qui est trop complet, et qu'il faut maîtriser l'ensemble des diagrammes pour utiliser le diagramme approprié, au moment voulu. D'autre part, le modèle doit apporter une valeur à un projet, à un développement. Dans cette idée, on parle aussi de Model Driven Development Agile. Surtout, le modèle sait désormais modéliser une architecture de services (SOA) via SOAml que Rational supporte déjà.

Côté Jazz, la journée en a abondamment fait état. Pour IBM il s'agit de le généraliser sur l'ensemble de la gamme Rational afin de disposer d'une couche de communication et de collaboration unique. L'autre objectif est de développer la communauté et son adoption par les éditeurs et les développeurs. En 2010, une sandbox sera disponible pour tester directement en ligne l'outil Team Concerto et Jazz. Un projet Eclipse est en cours et activement soutenu. Le support de Visual Studio n'est pas oublié.

■ François **Tonic**

SON NOM EST GORT
IL FAIT DES KILLER APPS



Par une nuit sombre et orageuse, une étrange réaction chimique déclenchée par une tempête d'éclairs a mis au monde un champion pour les développeurs du monde entier qui créent des killer apps. Visitez infragistics.com/killerapps pour connaitre l'histoire de Gort et apprendre comment créer votre propre killer app.

Infragistics
KILLER APPS. NO EXCUSES.

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

Adobe Max : du Flash, du Flex, de l'iPhone !

Comme chaque année, Adobe organise sa grande conférence développeur et designer : Max. Mais contrairement aux autres années, l'Europe a été délaissée (la conférence s'est tenue à Los Angeles du 4 au 7 octobre). Et c'est bien dommage car de nombreuses annonces et présentations ont eu lieu. Débutons avec la future Creative Suite 5, CS5, qui fera son apparition en 2010. L'éditeur a dévoilé la première version de Flash Professional CS5 qui sera disponible en bêta dans quelques semaines. Cette évolution incorpore plusieurs nouveautés très attendues : support iPhone, nouveau framework pour éditer et manipuler les textes (Text Layout Framework), incorporation du XML dans les fichiers FLA, nouvelle palette de code snippets, intégration avec Flash Builder, idéal pour le développement ActionScript, et un éditeur ActionScript lui aussi revu.

Du Flash natif pour iPhone

Sur la partie iPhone, Adobe ne propose pas d'exécuter du code dans un runtime dans le navigateur, élément que la licence d'utilisation Apple refuse.

Il s'agit donc de générer une application native iPhone que l'on téléchargera depuis AppStore et que l'on installera directement sur le « bureau » de son iPhone. Comme si on faisait du AIR... Il s'agira d'un Flash 10 « mobile ».

Cela ne dispense pas le développeur d'être inscrit au programme développeur Apple iPhone. Ce Flash fonctionnera sur tout modèle tournant sous iPhone OS 3.x. Cela a nécessité pour Adobe de créer un nouveau compilateur dans son environnement Flash afin de générer du Flash natif.

Le développeur accèdera à l'ensemble des API d'AIR et de Flash 10.1.

Un travail d'optimisation sera nécessaire aux Flasheurs et Flexeurs pour générer une application véloce sur iPhone. Cette intégration permet de supporter le tactile, l'orientation de l'écran, l'accéléromètre, la géolocalisation. Il sera même possible d'utiliser le format H.264 pour la vidéo.

Vers la fluidité des vidéos diffusées en continu

Adobe dévoile également sous le nom de code Zeri, un projet visant à ajouter le support d'un nouveau protocole de diffusion multimédia avec le streaming HTTP sur la Flash Platform et qui inclura la protection des contenus basé sur Flash Access 2.0. Cette technologie HTTP sera un standard ouvert basé sur les standards du marché. Elle apportera aux éditeurs de contenus, distributeurs et partenaires les outils dont ils ont besoin pour utiliser des infrastructures HTTP et assurer une diffusion multimédia de qualité dans Flash Player 10.1 et AIR 2 sur les postes de travail comme sur les terminaux mobiles. Zeri prendra en charge tous les codecs multimédias actuels et demeurera compatible avec les contenus de qualité - en direct et en différé - grâce à sa gestion des débits variables et à l'enregistrement vidéo numérique dans Flash Player. Avec le projet Zeri et Flash Media Server 3.5, Flash Platform déploie des contenus multimédias de nouvelle génération à grande échelle.

Et Flex Builder perd son Flex

Adobe prépare aussi activement ActionScript 4, Flex 4 et l'environnement de développement : Flex Builder 4, pardon, Flash Builder 4. Cette version arrive en bêta 2 et cible la productivité, les flux développeur – designer et le développeur des applications données.

Il s'intègre au futur Flash Catalyst qui tarde à arriver. Cette bêta 2 bénéficie d'une amélioration générale et de la mise à jour de nombreuses fonctions. Pré-version disponible dès maintenant !

Catalyst, la tour de contrôle des projets web et d'intégration entre le code et le design, arrive lui aussi en bêta 2. Parmi les nouveautés, retenons l'expert vers AIR, la création de fichiers SWF directement déployables, l'import de la vidéo. Le lancement du produit est prévu pour le premier semestre 2010 !

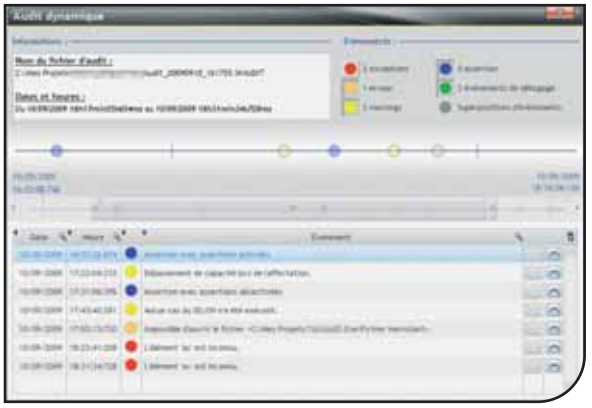
Du cloud pour AIR, Flex et Flash ?

Le cloud computing a été aussi un des thèmes de MAX 2009. Mais quel intérêt dans un contexte basé sur les technologies Adobe ? Pour les technologies de type RIA, le cloud doit être comme un environnement serveur et de services avec du http, du web services, de l'accès distant, avec des API spécifiques. C'est intéressant de mettre du LiveCycle, du BlazeDS, du WebORB, du GraniteDS dans le cloud... Dans un contexte cloud, cela pourrait permettre une montée en charge bien plus performante selon la demande. Mais tout cela a un coût à considérer. Pour les RIA, le cloud peut être vu comme un lieu de stockage en plus d'être une plate-forme serveur, d'exécution, à condition bien entendu de faire du provisionning et d'utiliser des API standard, comme REST et SOAP, par exemple pour stocker les données, utiliser du Google AppEngine pour son DataStore ou via JPA. Connecter du RIA et du Cloud n'est pas forcément difficile. Basiquement, on passe par du http, remoting, web services, avec des services de persistance de données, du load-balancing automatique. Ainsi, BlazeDS fonctionne très bien dans AppEngine et WebORB dans Amazon EC2. Reste à intégrer les notions de cloud directement dans les outils, ce qui n'est encore pas fait...

François **Tonic**

Outil

WinDev 15 : bientôt sur nos PC



Comme à chaque version majeure, l'éditeur PC SOFT fait évoluer profondément sa solution de développement. Cette version 15 n'échappe pas à la règle avec 555 nouveautés et améliorations. Plusieurs se détachent nettement du reste : HyperFileSQL, installation en mode multi-sites et à distance, support d'Android et de Windows 7, création de web services simplifiée et facilitée. Côté objet d'interface, WinDev 15 continue à proposer de nouveaux champs et objets : jauge circulaire, champ agenda par exemple. L'administration a été grandement améliorée avec le superviseur d'infrastructure logicielle, en remplacement du Centre de contrôle topologique. L'objectif est de garder un œil précis et complet sur les applications WinDev et Webdev en production. Vous pouvez aussi générer un rapport d'activité et auditer automatiquement l'état de production. Les responsables de production peuvent ainsi avec la v15 mieux comprendre ce qui se passe réellement, les composants utilisés, etc. D'autre part, on dispose d'un audit statique du projet capable de proposer des optimisations et surtout de voir le code mort, les problèmes de structures, avec bien entendu des fonctions de métrique de code. Côté base de données, les nouveautés sont nombreuses. Notons, par exemple, l'arrivée de HyperFileSQL sur Mac, l'utilisation d'un mode cluster.

Côté déploiement, PC SOFT innove en proposant en standard un déploiement distant et multisite des applications. La fonction push installation s'appuie sur ActiveDirectory sous Windows pour voir les postes et les groupes, définir les postes à déployer, déployer les mises à jour. Fonction intéressante : la possibilité de revenir à une version antérieure.

WinDev Mobile suit le marché en supportant Windows Phone, Windows Mobile 6.5 et surtout Android. Depuis son environnement, on peut générer une application Android en quelques clics, tout en prenant en charge les écrans tactiles (fonction de défilement). Android bénéficie d'un bon support : avec plus de 250 fonctions WLangage compatible. Côté iPhone, on dispose désormais d'un affichage plein écran pour les projets WebDev.

Vous pourrez découvrir la v15 durant le tour de France PC SOFTdu 12 novembre au 8 décembre.

Site : www.pcsoft.fr

PLATEFORME PROFESSIONNELLE DE DÉVELOPPEMENT (AGL)

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

W

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

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

Vos utilisateurs et clients apprécieront immédiatement ces évolutions.

Flex, Silverlight, JavaFX le match

Perdus au milieu d'une offre de technologies RIA aussi touffue que confondante, bon nombre de développeurs préfèrent encore s'en remettre à leur bon vieux web que de se perdre dans cette jungle étrange... et c'est bien dommage tant ces technologies leur permettraient d'aller loin ! Alors soyons concrets : quelle est la meilleure technologie pour votre projet ?

Puisqu'on est en droit d'attendre d'une Rich Internet Application plus que d'une simple application web, je me propose de définir la RIA par rapport aux lacunes actuelles des technologies web. A l'heure d'aujourd'hui, le web - c'est-à-dire HTML 4, CSS 2 et Ecmascript 3 - ne permet pas nativement :

- le multimédia audio et vidéo (on s'en remet la plupart du temps à Flash pour cela)
- le dessin vectoriel (à moins d'utiliser SVG qui est mal supporté par Internet Explorer)
- les animations (ou alors de manière limitée avec Javascript)
- la 3D (moteur 3D, utilisation de l'accélération matérielle)
- la possibilité d'utiliser l'application en mode déconnecté (« offline mode »)
- la possibilité d'accéder aux ressources du système

Cette liste ne signifie pas que ce sont là les seuls critères pour juger de la pertinence d'une technologie RIA, mais elle montre les nouvelles possibilités que celle-ci est susceptible de vous offrir.

Au-delà de ces fonctionnalités, le gros plus des technologies RIA que nous passons en revue aujourd'hui, c'est surtout ce qu'elles apportent en termes de productivité pour le développeur. En effet, Sun, Adobe et Microsoft proposent des ateliers logiciels permettant de construire rapidement des interfaces riches au sein de leur RIA... et c'est probablement la différence la plus flagrante par rapport à la réalisation d'interfaces web modernes (Ajax) dont le développement peut s'avérer très long dans bien des cas.

Les forces en présence

Flex 3.2

Adobe Flex est la technologie sur laquelle nous avons le plus de recul et pour cause, Flex premier du nom est sorti en mars 2004. C'est toutefois lors de la refonte qui a abouti à la version 2 en juin 2006 que Flex explose véritablement : Adobe prend alors un virage radical en distribuant son SDK gratuitement. Au niveau technique, les choix sont les suivants :

- un langage de description d'interface, MXML
- un langage de script, Action Script 3
- un environnement de développement, Flexbuilder (basé sur Eclipse)
- un environnement pour le design d'interfaces avancées : le bien connu Flash CS4
- un runtime côté client qui n'est autre que Flash

Depuis 2008 et l'arrivée de Flex 3, la philosophie reste la même, la plus grosse innovation étant l'introduction du runtime AIR pour faire fonctionner des applications Flex directement sur le bureau.

Silverlight 3.0

Le parcours de Silverlight a été difficile : conçu comme étant une version embarquée dans le navigateur de Windows Presentation Foundation, il apparut dans un second temps comme un concurrent de Flash, notamment en matière de vidéo. Depuis Silverlight 2 et plus encore depuis Silverlight 3 sorti en juillet dernier, les choses sont plus claires : Silverlight est bel et bien une vraie technologie RIA. D'un point de vue technique, on y retrouve une philosophie proche de celle de Flex :

- un langage de description d'interface, XAML
- la technologie de développement .NET incluant tous les langages supportés par la CLR (C#, VB.NET...) mais aussi les langages de la DLR (Jscript, IronPython, IronRuby)
- un environnement de développement, Visual Studio 2008
- un environnement pour la création d'interfaces avancées : Expression Blend
- un runtime côté client : le plug-in Sliverlight.

Il existe également un équivalent à Adobe AIR depuis la dernière version de Silverlight : il est possible de faire fonctionner une application Silverlight offline et en dehors de tout navigateur.

JavaFX 1.2

Dernière technologie RIA majeure d'un point de vue chronologique (la version 1 datant de décembre 2008), JavaFX est loin d'être dénué d'intérêt. JavaFX a l'atout énorme de s'appuyer sur l'univers Java dont la maturité et la robustesse ne sont plus à démontrer. Java ayant néanmoins une réputation d'austérité dans ses interfaces, JavaFX va tenter de changer une fois pour toutes cet a priori, pour entrer véritablement en concurrence avec Flash / Flex et Silverlight. JavaFX, bien que complètement nouveau, s'appuie sur des technologies bien connues du côté de chez Sun :

- un langage de script qui se nomme également JavaFX et ressemble à un Java simplifié

- un environnement de développement : NetBeans ou Eclipse équipé du plug-in adéquat
- un moteur d'exécution qui s'inclut dans la machine virtuelle Java

Comme ses concurrents, la technologie JavaFX permet de faire fonctionner des applications hors du navigateur, toujours en s'appuyant sur votre machine virtuelle Java.

RIA en action

Afin d'être aussi concret que possible, regardons la philosophie de chacune de ces technologies par l'exemple.

Le classique Hello World

Il s'agit ici une simple application envoyant un message à l'écran lors d'un clic sur un bouton. Cela nous permettra de faire connaissance avec les bases nécessaires à toute application dans chacune des technologies comparées.

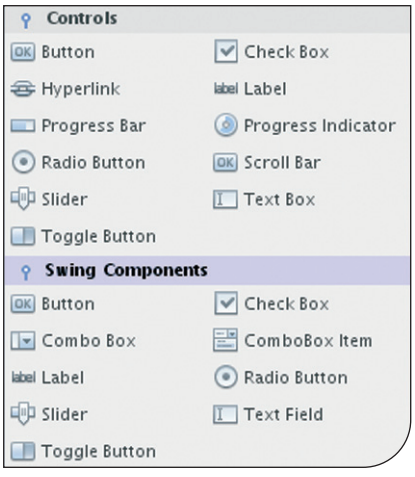
Flex

Dans cet exemple, élaboré à l'aide de Flex Builder, nous n'avons qu'un seul fichier mxml dans lequel est encapsulé notre code actionscript.

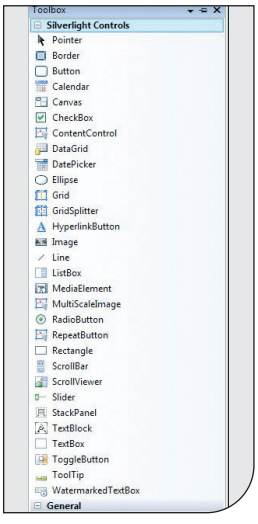
```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout
="absolute">
      <mx:Script>
      <![CDATA[
            private function hello():void{
                  myText.text = "Hello world !";
            }
      ]]>
      </mx:Script>
      <mx:Text id="myText" x="10" y="10" text="..." width=
"209"/>
     <mx:Button x="154" y="36" label="Click me" enabled="true"
click="hello()"/>
</mx:Application>
```

Silverlight

On retrouve ici les principes du développement basé sur XAML.
Tout d'abord notre interface :



Quelques contrôles JavaFX disponibles sous NetBeans



Liste des contrôles Silverlight proposés par Visual Studio

MainPage.xaml

```
<UserControl x:Class="HelloWorld.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
 xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="640" d:DesignHeight="480">
  <Grid x:Name="LayoutRoot">
        <TextBox x:Name="TheTextBox" Width="100"/>
        <Button x:Name="TheButton" Content="Click Me !" Width=
"100" Height="50" Click="TheButton_Click"/>
    </Grid>
</UserControl>
```

La gestion des événements relatifs à cette interface se trouve dans le « code behind » que nous avons réalisé ici en vb.net :

MainPage.xaml.vb

```
Partial Public Class MainPage
    Inherits UserControl

    Public Sub New()
        InitializeComponent()
    End Sub

   Private Sub TheButton_Click(ByVal sender As System.Object,
 ByVal e As System.Windows.RoutedEventArgs)
        TheTextBox.Text = "Hello World !"
    End Sub
End Class
```

JavaFX

Contrairement à ses concurrents, dans JavaFX, interface et gestion des événements sont codés dans le même langage.
On peut donc retrouver l'ensemble du code dans un seul et même fichier :

```
/*
 * Hello World en JavaFX.
 */

package helloworld;

import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.text.Text;
import javafx.scene.text.Font;
import javafx.scene.control.Button;
import javafx.scene.input.MouseEvent;

/*
 * Stage est le conteneur de base pour
 * un script JavaFX
 */
Stage {
   title: "Cliquez sur le bouton"
   width: 300
```

```
    height: 200
    /*
     * On déclare une Scene (un écran)
     * directement dans le Stage. Elle
     * contiendra différents "nœuds"
     * (contrôles).
     */
    scene: Scene {
        content: [
            /*
             * Un label
             */
            Text {
                font : Font {
                    size : 16
                } // Font
                x: 10
                y: 30
                content: "Cliquez sur le bouton"
            } // Text,
            /*
             * Un bouton qui quitte l'application
             * lorsque l'on clique dessus.
             */
            Button {
                width: 100
                height: 20
                text: "Cliquez moi"
                onMouseClicked: function( e: MouseEvent ):Void {
                    coucou.visible = true;
                    coucou.toFront();
                }
                translateX: 100
                translateY: 100
            } // Button

        ]
    }
}

/*
 * Fenêtre contenant notre message
 */
def coucou = Stage {
    title: "Hello World"
    width: 300
    height: 200
    scene: Scene {
        content: [
            Text {
                font : Font {
                    size : 16
                } // Font
                x: 10
                y: 30
                content: "Hello World"
                id: "coucou"
            } // Text
            /* Bouton quittant l'application */
            Button {
                width: 100
                height: 20
                text: "Fermer"
                onMouseClicked: function( e: MouseEvent ):Void {
                    FX.exit();
                }
                translateX: 100
                translateY: 100
            } // Button
        ]
    },
    visible: false
}
```

L'application « BubbleMark »

Le Bubblemark est un projet lancé à l'initiative d'Alexey Gavrilov : c'est une application réalisée dans différentes technologies RIA afin de les comparer, en particulier en termes de performances. Son principe est le suivant : plusieurs bulles se déplacent et rebondissent les unes sur les autres. Notre technologie RIA doit donc permettre le calcul du mouvement de chaque bulle en temps réel puis leur rendu dans le navigateur. Vous pouvez retrouver les codes sources complets et regarder les différentes démonstrations à l'adresse bubblemark.com.

Flex

Le projet Bubblemark a sous Flex, une architecture simple et efficace :
Flexballs.mxml : interface générale de l'application
BallsTest.as : gestion des événements
Ball.as et Flexball.as : classes métiers pour définir les boules et leur comportement.
Le code est dense, tout en restant clair : c'est là une des forces de l'alliance du mxml et de l'Actionscript 3. C'est d'autant plus simple pour le développeur que l'utilisation de Flexbuilder rend les choses encore plus claires. L'examen des classes Ball et FlexBall est assez intéressant : il montre à ceux qui en doutaient encore que le développement sous Actionscript 3 peut donner lieu à de véritables développements objets qui n'ont rien à envier à la concurrence.

Silverlight

Sans rentrer dans le détail du code, il est intéressant de regarder comment celui-ci est structuré :

Sun présente JavaFx sur son site... avec une vidéo Flash !

«Tour de France» WINDEV 15

VENEZ DÉCOUVRIR WINDEV 15 PRÈS DE CHEZ VOUS

Développer 10 fois plus vite (et faire développer 10 fois plus vite) ?

Venez découvrir WINDEV 15 et ses 555 nouveautés près de chez vous.

Inscrivez-vous sur www.pcsoft.fr (gratuit)

CALENDRIER

Montpellier	12 Nov
Nantes	17 Nov
Bordeaux	18 Nov
Toulouse	19 Nov
Genève	24 Nov
Lyon	25 Nov
Strasbourg	26 Nov
Paris	1 Déc
Bruxelles	2 Déc
Lille	3 Déc
Marseille	8 Déc

de 14h00 à 17h30

TOUR DE FRANCE

VOTRE CODE EST MULTI-PLATEFORMES:
Windows, .Net, Java, PHP, J2EE, XML, Internet, Ajax, Pocket PC, SmartPhone, Client riche ...

Séminaire gratuit pour Décideur, Développeur et Chef de Projet

GRATUIT
inscrivez-vous sur www.pcsoft.fr
Seulement 10.000 places disponibles

www.windev.fr

Page.xaml : interface générale de l'application
Page.xaml.cs : code behind de gestion des évènements de l'appli : start, stop, etc.
BallControl.xaml : user control graphique.
BallControl.xaml.cs : code behind gérant les évènements du user control, utilise les classes définies dans Ball.cs
Ball.cs : classes métiers pour définir les boules et leur comportement.

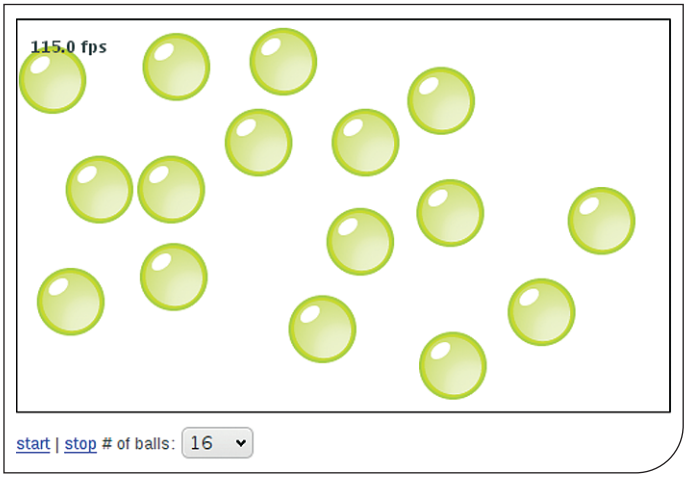
Comme on le voit sur ce petit projet, la structure proposée dans les projets Silverlight est particulièrement structurante et rend l'ensemble du code assez aisément lisible mais aussi maintenable. Tout naturellement, les habitués de .Net y retrouveront les mécanismes qu'ils connaissent déjà, peut-être grâce à leur expérience sur asp.net ou WPF par exemple. L'ensemble du code produit est assez « verbeux » par rapport à ses concurrents, mais la différence ne se fait pas trop sentir pour le développeur dans la mesure où les outils de développement lui facilitent la tâche.

JavaFX
Sous JavaFX, la structure du code du projet est assez différente dans la mesure où il n'y a qu'un langage pour décrire l'interface et gérer les événements liés à celle-ci.
JavaFXBalls.jfx : interface générale de l'application et gestion des événements
Le code est là aussi assez compact (c'est la technologie qui a réclamé le moins de lignes de codes), mais l'ensemble se lit peut-être de manière moins évidente que sur les autres technologies étudiées : la logique déclarative des éléments de l'interface pouvant en particulier s'avérer spéciale à appréhender pour des développeurs habitués au HTML ou autres langages de description d'interface basés sur XML. L'examen du code de l'interface montre ici l'utilisation d'éléments Swing bien connus des développeurs du monde Java et qui trouvent naturellement leur place dans JavaFX.

RIA sur le gril…

Productivité du développement

La question de la productivité pourrait faire longtemps débat tant les paramètres influant sur celle-ci sont nombreux, bien au-delà de la technologie choisie. Nous avons donc observé les outils et les méthodes de travail proposés par chacune au cours de nos tests pour nous mouiller un peu…



Le bubblemark en action !

Flex
Parmi les RIA testées, on voit très rapidement que Flex est la plus mature et probablement la plus efficace dans la conception d'interface. Flex Builder est un outil complet et intuitif à la fois pour la mise en place des interfaces et l'écriture du code Actionscript.
Cela permet d'être réellement très productif sur des projets courts tout en ayant une faible courbe d'apprentissage. Cette apparente facilité peut cependant devenir un vrai défaut : il est si facile de coder n'importe quoi… qu'il l'est encore plus de coder n'importe comment ! C'est probablement une des raisons expliquant certaines déceptions exprimées par des développeurs par rapport à l'utilisation de Flex sur de grands projets.
Petit désavantage par rapport à ses concurrents, Flex n'est qu'une technologie cliente qu'il faut éventuellement intégrer avec une technologie serveur (PHP, .NET, Java…)… intégration qui n'est, de fait, pas native là où Silverlight s'intègre tout naturellement avec .NET alors que JavaFX facilite grandement le travail avec JEE.

Silverlight
Comme d'habitude chez Microsoft, le développement est bien outillé : Visual Studio 2008 est toujours un plaisir à utiliser et Visual Web Developper permet une première approche pour ceux qui ne souhaitent pas investir dans une licence (300 € HT environ pour une version standard). Expression Blend s'avère vite indispensable pour une approche plus « créative », d'autant que Visual Studio ne fournit pas de concepteur d'interface en mode graphique. Les composants du framework MS .net 3.5 sont très nombreux et il est possible d'utiliser d'autres composants commerciaux tels ceux de Telerik. En bref, Silverlight dispose de tout ce qu'il faut pour réaliser des développements correctement et rapidement… à condition d'investir un peu en temps et en licences dans les outils proposés.

JavaFX
Les plugins pour Eclipse ou Netbeans sont relativement similaires. La création d'un projet permet d'avoir un Stage préparamétré.
L'auto-complétion du code fonctionne bien, et comme d'habitude sous les IDE java, un survol d'un élément donne accès à la documentation.

Moonlight semblait encore loin d'être au point pour le Bubblemark…

Il est également possible d'ajouter des éléments dans les sources directement par « cliquer-déposer » ou double clic grâce à une barre d'outils latérale. Il existe d'ailleurs une bibliothèque de composants assez riche à la base (composants standard, éléments graphiques) qui est encore renforcée par l'opportunité d'utiliser des composants Swing éprouvés. On regrette par contre, et c'est le gros point faible de cette solution par rapport à la concurrence, l'absence d'éditeur graphique pour la création d'interface. L'approche « tout script », est en effet à la fois assez peu intuitive et moins productive pour bien des développeurs. Finissons avec l'incontestable point fort de JavaFX : la possibilité d'utiliser toutes les bibliothèques Java existantes… et par là même d'avoir accès à un nombre de ressources techniques existantes considérables !

Performances

Les performances ne constituent certainement pas l'Alpha et l'Oméga du développeur moderne, tant la majorité des développements sont peu exigeants en termes de ressources. Il arrive néanmoins régulièrement que cette question compte dans le monde des RIA : en effet, dès lors qu'il est question d'aller loin en termes d'applications graphiques (vidéo, dessin vectoriel calculé en temps réel, 3D), il devient dès lors important de compter sur des performances et une stabilité suffisantes.

Flex

Les performances de Flash ne sont plus à vanter - en particulier depuis Flash 10 -, tant il est aujourd'hui le plug-in de référence pour les interfaces riches (on pense par exemple aux jeux en ligne) et le multimédia dans le monde du web. Flash est d'ailleurs probablement victime de son succès et de ses limites puisque sa tendance à consommer beaucoup de ressources système couplé à sa forte présence sur les pages web visitées provoque régulièrement des crashes navigateurs… En termes de performances pures, nos tests semblent confirmer que le couple Flex / Flash est un peu en deçà de Silverlight… impressions corroborées par d'autres benchmarks récents et qui montrent surtout les récents progrès de Silverlight.

Silverlight

Silverlight est particulièrement performant, comme nous l'ont par exemple montré les tests que nous avons effectués sur le BubbleMark (... et que nous vous invitons à reproduire !). Cette haute performance est probablement induite par l'exécution dans un plugin / machine virtuelle .net fortement intégrée au système Windows. Et sur les autres systèmes ? Mettons de côté Linux puisque, nous y reviendrons, Moonlight 2.0 n'est encore qu'en bêta. Autres bonnes surprises durant nos tests : la consommation des ressources système reste raisonnable et le temps de chargement initial des applications paraît assez faible.

JavaFX

La technologie JavaFX est jeune mais les performances sont déjà au rendez-vous puisque nous avons observé de ce point de vue un comportement comparable à celui de Flash… auquel on ajoutera un bémol, les temps de chargement initiaux, qui restent toujours importants par rapport à ses adversaires. Le BubbleMark incluant des développements Java, sans JavaFX, dans son comparatif, on constatera que JavaFX reste un peu en retrait par rapport aux bonnes vieilles applets embarquées… ce qui montre aussi que des améliorations sont sans doute encore possibles.

Fonctionnalités « RIA » avancées

Comme nous l'avons vu en introduction, on est en droit t'attendre de nos RIA qu'elles nous proposent de vrais « plus » fonctionnels par rapport au web : en particulier en termes de capacités graphiques avancées et de fonctionnalités dites « desktop » (hors navigateur).

Flex

Les fonctionnalités proposées par le couple Flash / Flex en matière graphique sont des références depuis plusieurs années, en particulier dans le monde de la vidéo où Silverlight tente aujourd'hui de s'imposer. Flash 10 a apporté plusieurs améliorations de ce point de vue : la création d'effets 3D et l'amélioration de l'accélération matérielle sont quelques avancées marquantes. Plusieurs frameworks de création 3D, indépendants d'Adobe, sont apparus ces dernières années pour tirer parti des possibilités de Flash. On peut citer *Papervision*, *Sandy*, *Away3D* ou encore *Alternetiva3D*. Les fonctionnalités desktop reposent sur la plate-forme Adobe Air qui est aujourd'hui disponible aussi bien sur Windows, MacOS que sur Linux. Air est une plate-forme permettant tout ce dont un développeur peut avoir besoin sur une application de bureau : fonctionnement offline, base de données SQLite embarquée, possibilité d'accéder aux ressources systèmes (pour, par exemple, lancer une application locale)… d'autant qu'Air ne se limite pas à l'exécution de Flash mais supporte également des développements Ajax ou PDF. Le tableau serait idyllique si Air n'était pas un runtime « de plus » à installer sur les postes clients… alors que JavaFX ou Silverlight ne demandent pas de runtimes supplémentaires pour supporter leur mode offline.

Silverlight

Silverlight est remarquablement bien doté en fonctionnalités graphiques. Côté multimédia, la vidéo tient ses promesses : streaming vidéo intelligent, prise en charge de la vidéo HD (H264, AAC et 720p), zoom et redimensionnements optimisés. Un autre vrai point fort est la disponibilité de « Perspective » qui permet aisément la mise en oeuvre d'une 3D ou plus exactement d'une « 2.5D ». Ces fonctionnalités graphiques sont renforcées par le support de l'accélération matérielle qui ouvre des possibilités considérables en la matière. Pour ce qui est des fonctionnalités desktop, Silverlight propose depuis sa version 3.0 un support du mode déconnecté. Il est d'une part possible de faire fonctionner l'application hors du navigateur (fonctionnalité « out-of-browser »), tout en permettant d'autre part d'y accéder sans connexion au web plus tard : une fois installé, l'application est capable de détecter si elle est online ou non et peut se mettre à jour automatiquement. Notons de plus que tout cela est simplissime aussi bien pour le développeur que pour l'utilisateur. Seul regret : pas de solution de stockage SQL intégré comme c'est le cas avec Air ou le plug-in navigateur Google Gears.

JavaFX

L'approche de JavaFX diffère assez nettement de ses concurrents pour les réalisations graphiques. L'absence d'outils de design intégré à l'environnement de développement est ainsi compensée par

l'existence de plug-in dans les outils de création (par exemple Photoshop) pour exporter les graphiques de manière exploitable par JavaFX. Côté possibilités graphiques, outre quelques effets et l'existence de composants « out-of-the-box » comme le player video (assez efficace mais qui connaît des problèmes sous Linux), JavaFX s'appuie essentiellement sur les possibilités du runtime Java existant. On peut par exemple intégrer Java3D -dont on attend toujours qu'il intègre la JRE par défaut - au sein d'applets JavaFX.
Tout comme pour Silverlight, il est assez simple de faire d'une application JavaFX « on-line » une application « desktop ». Comme pour le reste, l'intégralité des fonctionnalités de Java est disponible. Il faudra néanmoins faire attention à la compatibilité avec les JRE les plus anciens (API SystemTray permettant d'avoir une icône dans la barre des tâches par exemple).

Portabilité

Quels utilisateurs pourront faire fonctionner ma belle application RIA ? C'est la dernière question que nous posons ici, mais c'est probablement la première que vous vous poserez à l'heure de choisir votre technologie de développement… en particulier à l'ère des applications mobiles !

Silverlight

La portabilité reste en partie un point d'interrogation. Aucun problème dans la galaxie Windows (d'XP à 7), où le plug-in a fonctionné dans tous les navigateurs utilisés lors de nos tests. Sur MacOS X, bon fonctionnement également, même si à notre grande surprise les performances constatées sur Safari sont nettement plus faibles que celles observées sur Firefox. Pour ce qui est de Linux, Microsoft nous demande de nous en remettre à Moonlight… et c'est là que les problèmes commencent : Moonlight ne supporte de manière officielle que les fonctionnalités de Silverlight 1.0, quant à Silverlight 2.0, il existe une version bêta qui a fonctionné sur environ 50% des tests auxquels nous avons procédé… bref, ne comptez pas trop sur la compatibilité Linux, cela nous semble encore très aléatoire et sans véritable garantie d'aboutir. Silverlight est désormais supporté par un certain nombre de plates-formes mobiles : Windows Mobile -forcément- mais aussi le Nokia S60. Peu de plates-formes certes, mais là encore une grande simplicité de mise en oeuvre pour le développeur qui n'aura pas à acquérir de compétences spécifiques pour réaliser le portage de son application pour Silverlight Mobile. Globalement, la portabilité est loin d'être le point fort de Silverlight, et ce d'autant plus que même sur les plates-formes compatibles, le plug-in reste encore très nettement moins répandu sur les postes clients que ses concurrents.

Flex

Flash est présent de manière quasi-universelle sur le poste client, c'est à dire plus de 99% du marché d'après plusieurs études, ce qui en fait le plug-in le plus répandu aujourd'hui. Il fonctionne très correctement partout et les différences entre systèmes d'exploitation, dont ont par exemple souffert les distributions Linux à une époque, se sont estompées depuis Flash 9 si bien que Flex serait la technologie RIA portable par excellence… s'il n'y avait la question du mobile !
Flash Lite est la version de Flash qui permet d'exécuter ce dernier sur des machines à faibles ressources, et en particulier sur les mobiles. Malgré une présence sur le marché maintenant ancienne, Flash Lite n'a pas véritablement percé sur le mobile, phénomène renforcé par la volonté de Steve Jobs de ne pas l'installer sur iPhone. Mais les choses pourraient finir par changer maintenant que l'on annonce l'arrivée de Flash 10 (et non plus Flash Lite) sur plates-formes Android, Windows Mobile, Symbian et Palms OS durant la conférence Adobe MAX d'octobre (voir page 10).

JavaFX

Tout comme Flex bénéficie de la forte présence de Flash sur le poste client, JavaFx peut compter sur l'ubiquité des machines virtuelles Java (présentes sur environ 9 machines sur 10) pour percer. Lors de votre première utilisation d'une application JavaFX, la mise à jour de votre JVM permettant le support de JavaFX est suffisamment transparente pour ne pas qualifier cela de nouvelle installation.
Sur le mobile, JavaFX peut aussi faire la différence face à la concurrence grâce à JME qui est la plate-forme de loin la plus répandue sur les smartphones modernes : « des milliards de terminaux » affirme même Sun ! Bien qu'encore assez récente, les premiers retours d'expérience sur cette technologie sont assez positifs. JavaFX semble donc être le digne héritier de la philosophie Java : *« write once, run everywhere ».*

L'heure du choix

Difficile de trancher de manière universelle et définitive sur ces technologies. La vraie question est plutôt : quelle est la bonne techno RIA pour votre projet ? Au-delà de tous les critères que nous avons exposés plus haut, il est en un qui fera souvent la différence : la culture et le background des équipes qui vont devoir travailler sur le projet. On imagine difficilement des développeurs ASP.NET préférer JavaFX à Silverlight, ou des amateurs de Flash ne pas tenter l'expérience « Flex »… et ce serait un bien mauvais conseil de vous inciter à changer brusquement cette culture.
En effet, toutes ces technologies sont bonnes à prendre : Flex est largement éprouvé, Silverlight arrive à maturité, et si JavaFX est récent, il repose sur des JVM qui risquent d'être encore là très longtemps. Chacune d'entre elles a aussi ses points forts et ses points faibles qui peuvent la qualifier ou l'éliminer automatiquement. Le tableau suivant les résume rapidement :
Si vous êtes encore indécis après tout cela et que vous tenez absolument à avoir mon avis personnel et donc hautement subjectif sur le sujet… je choisirais Flex parce que c'est choisir la solution de loin la plus éprouvée, mais aussi parce que Flash, malgré ses défauts, est le plug-in navigateur qui me semble avoir le meilleur compromis entre performances, fonctionnalités et ubiquité. Mais en faisant un tel choix, je prêterai une attention toute particulière à l'architecture du projet et en particulier au lien entre l'interface et « back-end » afin que les gains de productivité initiaux ne soient pas noyés dans une soupe de code quelques mois plus tard.

Jean-Baptiste **Boisseau**
Eutech SSII

	Flex	Silverlight	JavaFX
Points forts	Maturité, productivité	Performances, fonctionnalités	Portabilité, utilisation des bibliothèques Java existantes
Points faibles	Technologie « cliente » à intégrer avec une technologie « serveur » tierce	Portabilité limitée	Pas d'outils de design d'interfaces, technologie encore émergente

Notre comparatif résumé en un tableau

Vous créez des sites Web ?

Créer des sites Web n'a jamais été aussi facile...

Des licences, du support technique, et de la visibilité pour les Pros du Web :

Vous êtes un développeur freelance, ou votre agence Web compte 10 employés ou moins ?
Ce programme est fait pour vous !

Devenez membre du programme et bénéficiez(*) pendant une durée de trois ans de licences gratuites(**) (Visual Studio, Expression, Windows Server, Microsoft SQL Server) et de support technique professionnel. Profitez également d'offres exclusives d'hébergement professionnels sous Windows à des prix jamais vus via nos partenaires.

Avec WebsiteSpark, vous rejoignez une communauté active de professionnels et vous gagnez en visibilité.

Inscription gratuite sur :
http://tinyurl.com/RejoignezNous
ou http://www.microsoft.com/web/WebsiteSpark/

Microsoft® WebsiteSpark™

Adressez-nous vos questions à : webfr@microsoft.com ou sur twitter via @WebsiteSparkFR

(*) Sous réserve d'éligibilité
(**) Sans coût d'entrée, valable pendant trois ans, un montant forfaitaire de USD $100 vous sera demandé si vous quittez le programme ou à l'issue des trois ans.

Google Wave : outil révolutionnaire de collaboration unifiée ?

Le 28 mai 2009, Google a annoncé Google Wave, son outil de communication et de collaboration. Il ne s'agit pas d'une simple application et il n'est pas encore accessible au public mais a déjà la réputation d'être un outil révolutionnaire.

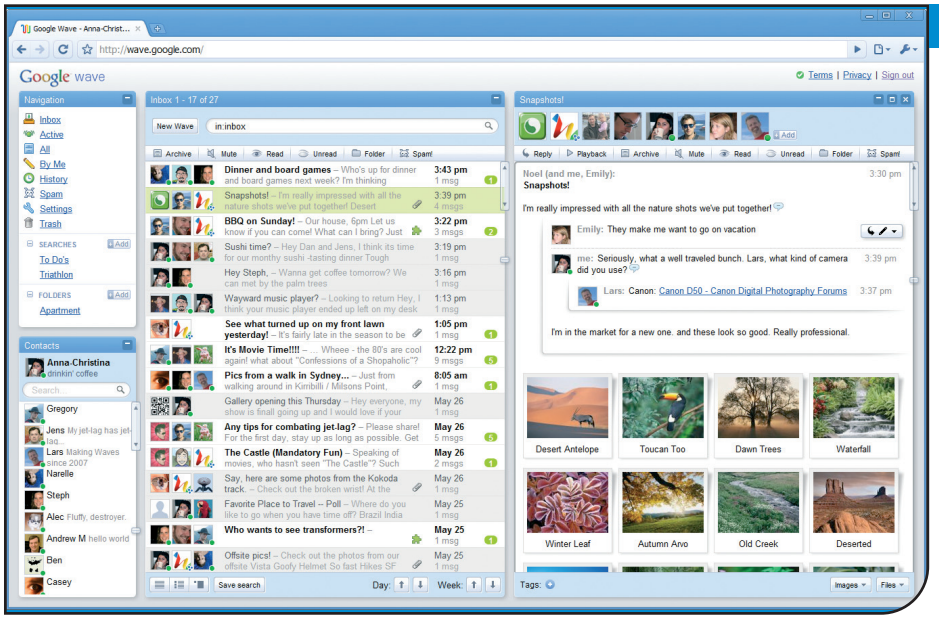


Fig.1

Beaucoup d'articles traitent de Google Wave et des concepts associés, et il est difficile de s'en faire une idée claire **car ses concepts sont réellement novateurs.** D'autant que seulement 25 000 personnes dans le monde ont été invitées à tester le Google Wave Developer Sandbox. Etant l'un de ces heureux élus, je profite de cet article pour vous présenter Google Wave : de quoi s'agit-il exactement, et est-ce réellement un outil révolutionnaire ?

Cet article s'inspire très fortement de deux posts que j'ai rédigés sur mon blog à la sortie de Google Wave : http://francois.goldgewicht.com/2009/06/03/google-wave-en-quelques-mots/ et http://francois.goldgewicht.com/2009/07/30/google-wave-premiers-tests/.

Les origines de Google Wave

Avant tout, il est essentiel de comprendre l'idée à l'origine de Google Wave. Tout a démarré il y a un peu moins de trois ans, dans l'esprit de Lars et Jens Rasmussen, les deux frères à l'origine de Google Maps. A l'époque ils dressent le constat suivant :

- l'e-mail et la messagerie instantanée (le *chat*), deux des succès les plus spectaculaires dans le domaine de la communication numérique, conçus dans les années 1960, ne sont que des adaptations électroniques de leurs équivalents traditionnels : respectivement, le courrier postal et le téléphone ;
- depuis, de nombreuses formes de communication ont été inventées, comme les blogs, wikis, documents collaboratifs. En outre, les ordinateurs et les réseaux ont grandement été améliorés.

Ils ont alors démarré un projet nommé *Walkabout,* dont l'objectif est de reposer sur un nouveau modèle de communication qui exploite tous ces progrès. La démarche sous-jacente peut être résumée en trois questions :

- Pourquoi devrions-nous passer notre temps à utiliser des types différents de communication et de collaboration (tantôt e-mail tantôt chat, tantôt conversations tantôt documents) ?
- Ne pourrait-on pas imaginer un modèle de communication qui unifierait la plupart des moyens utilisés actuellement, tout en restant simple ?
- Ne pourrait-on pas imaginer un système de communication qui exploite les capacités informatiques modernes, plutôt que d'imiter les moyens de communication non-électroniques ?

Ainsi, après plus de deux ans de travail, le projet Walkabout a été rebaptisé Google Wave, puis présenté au grand public le 28 mai 2009.

Google Wave au premier abord

Une application Web

Le site officiel présente Google Wave de la manière suivante : *« Google Wave is a new tool for communication and collaboration on the web ».* Google Wave propose en effet une application Web dont l'interface n'est pas sans rappeler GMail : [Fig.1]

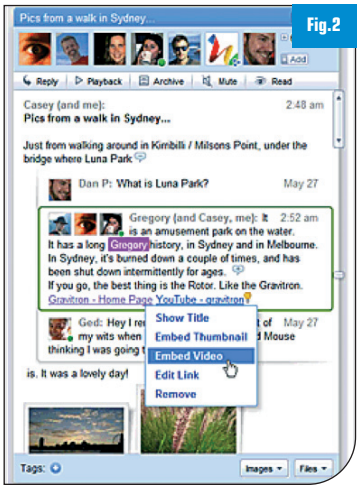


Fig.2

On retrouve effectivement une présentation en colonnes avec un menu, une liste de contacts, une boîte de réception et un cadre de visualisation de l'élément sélectionné dans la boîte de réception. Sauf que là, elle ne contient plus des e-mails mais des Waves.

Des Waves ?

Une Wave est une entité à mi-chemin entre une **conversation** et un **document**. Cette entité est **partagée** par un ou plusieurs **participants,** qui peuvent être des humains ou des robots. Vous créez une Wave, ajoutez des participants et cette Wave peut être ensuite modifiée par les participants en temps réel. [Fig.2]

Que signifie « modifier une Wave » ? Tout simplement taper du texte riche ou ajouter des photos, des vidéos ou même des gadgets (une carte par exemple). De quelle manière ? En insérant une réponse ou en éditant la Wave directement. Google Wave supporte les modifications concurrentes des Waves : vous les voyez évoluer en temps réel lorsque d'autres participants les modifient, caractère par caractère ! Ce nouveau mode de communication et de collaboration unifie donc les messageries instantanées (chat) et persistantes (e-mail) au sein d'un concept plus avancé.

Les Waves, une nouvelle façon unifiée de communiquer et de collaborer avec ses contacts

Comme l'explique Lars Rasmussen dans sa démonstration au Google IO, LA conférence Google, on passe donc d'un mode de communication « point à point » (e-mails) à un mode centralisé (à la façon d'un bus) : les participants manipulent les Waves dès leur création ou au cours de leur vie. Ainsi, les participants peuvent visualiser les modifications des Waves :

- en temps réel, dans le cas de modifications par d'autres participants connectés ;
- en playback, dans le cas de modifications effectuées par des participants connectés ou non.

La démonstration de cette fonctionnalité de **playback** est impressionnante. L'idée derrière cette fonctionnalité est simple, il s'agit d'indiquer aux participants : « Who said what and when ». Les participants peuvent donc visualiser l'historique des modifications, modification par modification, d'une manière animée ou par clics successifs.

Revenons à nos Waves

En fait, la notion de Wave repose sur un découpage fin : une Wave rassemble des **Wavelets.** Par exemple, lorsque vous entamez une discussion privée avec un participant particulier au sein d'une Wave rassemblant plusieurs participants, cela crée des Wavelets privées au sein de cette Wave (en plus des autres Wavelets partagées avec les autres participants). Celles-ci rassemblent elles-mêmes une hiérarchie de *blips*. Une *blip* consiste en l'unité de base des conversations et leur contenu est un document. [Fig.3]

Voilà, vous connaissez tout le vocabulaire ! En fait l'utilisateur n'a pas à connaître cette terminologie, mais elle est essentielle pour le développeur. Elle est donc clairement définie dans la documentation de l'API. Notons également que vous pouvez attacher des tags à vos Waves, ou les relier entre elles par du *drag and drop*.

Google Wave, une « simple » application Web ?

En fait, l'application Web décrite précédemment n'est qu'un des trois éléments qui constituent Google Wave, les trois "P" : il s'agit du *Produit*. Google Wave, c'est également une *Plate-forme* et un *Protocole*.

Un Produit

L'objectif de l'application Web Google Wave, c'est-à-dire du *Produit,* vous l'avez compris, est de permettre d'accéder aux Waves et de les éditer. Cette application a été développée en GWT et repose sur des apports de HTML 5.

Les intéressés pourront découvrir ces fameux apports de HTML 5 dans un article très clair de Tim O'Reilly (http://radar.oreilly.com/2009/05/google-bets-big-on-html-5.html), mais on peut en retenir au moins un : pour effectuer l'envoi d'un fichier, avec HTML 5, on n'est plus obligé de passer par l'habituel bouton "Parcourir".

On peut par exemple le faire via un *drag and drop* du fichier depuis l'explorateur vers une zone de la page HTML. Cet apport est largement exploité par Google Wave, notamment lors de l'attachement de fichiers à une Wave. La démonstration montre ainsi l'insertion de plusieurs images dans une Wave grâce à un seul *drag and drop.*

Une Plate-forme

Google Wave, c'est également une **Plate-forme** ouverte et extensible constituée de plusieurs API :

- **Embed** : API JavaScript permettant d'intégrer des Waves dans des pages Web, des blogs, etc.
- **Extensions** : API permettant de créer des extensions à l'application Web. Ces extensions sont faites pour travailler au sein des Waves.

Un Protocole

Google Wave, c'est enfin un **Protocole** ouvert spécifiant le stockage et l'échange des Waves, en gérant les accès concurrents.

Notons que le code de Google Wave est Open Source. L'objectif de Google est de le faire adopter par le plus de monde possible. Ainsi chaque entreprise pourra installer son propre Google Wave : on est loin de l'image de "Big Brother" si souvent attribuée à Google (néanmoins compréhensible concernant certaines applications)…

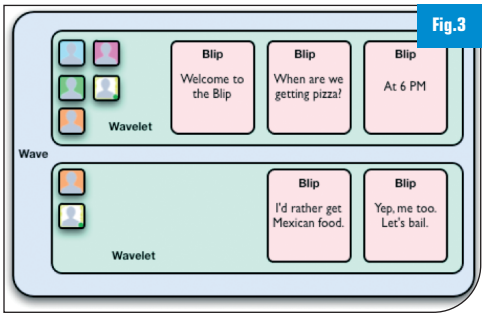
Le Google Wave Federation Protocol est un projet Open Source dont voici les principes :

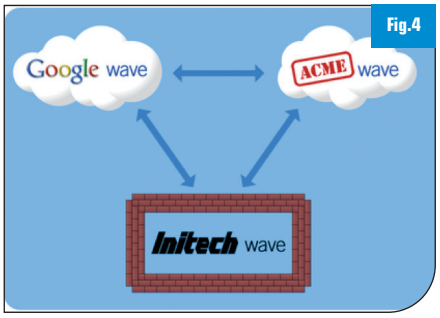
- tout le monde doit avoir la possibilité de devenir un Wave provider ;
- le modèle Wave repose sur un réseau distribué : les communications se font en peer-to-peer, et non via un serveur central ;
- tout le monde peut contribuer à l'amélioration du protocole et les décisions concernant ses spécifications sont publiques. [Fig.4]

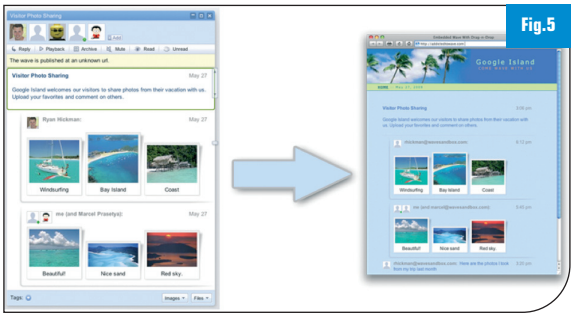
L'implémentation de ce protocole repose sur un modèle nommé « Transformation Opérationnelle », dont l'objectif est de permettre la gestion des accès concurrents à des documents en cours d'édition, tout en conservant leur intégrité.

Les concepts sous-jacents sont complexes et dépassent largement le cadre de cet article.

Pour plus d'informations à ce sujet, vous pouvez consulter le site officiel : http://www.waveprotocol.org/.







Quelques mots sur l'API Embed

L'API Embed est une API JavaScript qui permet d'intégrer des Waves dans des applications Web : [Fig.5]. Ainsi lorsqu'une Wave est modifiée dans l'application Google Wave, votre page Web ou votre blog qui embarque cette Wave sera mis à jour automatiquement. Et l'inverse devrait même être possible une fois que cette API aura été étoffée. Prenons un exemple simple. Considérons un *div* dans lequel on souhaite charger une Wave dont l'identifiant est *waveID* :

```
<div id="waveframe"></div>
```

Pour intégrer la Wave il suffit d'inclure l'API :

```
<script src="http://wave-api.appspot.com/public/embed.js"
        type="text/javascript"></script>
```

Puis le code suivant :

```
<script type="text/javascript">
   var wavePanel = new WavePanel(
      'http://wave.google.com/a/wavesandbox.com/'
   );
   wavePanel.loadWave('wavesandbox.com!w+waveID');
   wavePanel.init(document.getElementById('waveframe'));
</script>
```

Pour plus de détails, vous pouvez consulter le tutorial fourni par Google : http://code.google.com/apis/wave/embed/guide.html. Pour voir quelques exemples, vous pouvez consulter les samples des API Google Wave : http://wave-samples-gallery.appspot.com/. Même s'il vous faudra un accès au Google Wave Developer Sandbox pour les tester, vous pourrez en voir des copies d'écran ainsi que leur code source.

Quelques mots sur les API Extensions

Les Extensions peuvent être de deux natures : il y a les **Gadgets** et les **Robots**.

Gadgets

Les Gadgets sont des extensions côté client afin d'enrichir l'IHM des Waves. Elles peuvent bénéficier des interactions multi-utilisateurs (pour créer des jeux multi-joueurs par exemple). [Fig.6]
Notons que les Gadgets iGoogle et OpenSocial fonctionnent sur Google Wave, ce qui apporte des milliers d'applications d'entrée de jeu, même si ces derniers ne peuvent profiter des aspects multi-utilisateurs propres aux Waves. Les Gadgets ne peuvent pas modifier les Waves, et n'en ont qu'une visibilité restreinte : ils ne peuvent que détecter des changements dans la liste des participants.
Les Gadgets sont distribués sous la forme de fichiers XML, comme le montre l'exemple du Gadget « Hello World » (fichier hello.xml) :

```
<?xml version="1.0" encoding="UTF-8" ?>
<Module>
  <ModulePrefs title="Hello Wave">
    <Require feature="wave-preview" />
  </ModulePrefs>
  <Content type="html">
    <![CDATA[
      Hello, Wave!
    ]]>
  </Content>
</Module>
```

Ainsi un Gadget est défini par un élément Module. L'élément Require indique ici que ce Gadget utilise l'API Google Wave Gadgets, et l'élément Content permet de définir le code HTML, CSS et JavaScript du Gadget. Ce fichier doit être mis à disposition sur un serveur Web : pour ajouter un Gadget à une Wave, il suffit d'actionner le menu « Extensions > Add gadget » puis d'indiquer l'URL du fichier XML précédent. Ce n'est pas tout, cette API propose plusieurs fonctionnalités intéressantes : par exemple, de récupération de la liste des participants ou de manipulation de l'état du Gadget par les participants de la Wave. Voici par exemple le code d'un Gadget embarquant un compteur et permettant aux participants d'une Wave d'incrémenter ce compteur grâce à un bouton :

```
<?xml version="1.0" encoding="UTF-8" ?>
<Module>
   <ModulePrefs title="State Example" height="120">
      <Require feature="wave-preview" />
   </ModulePrefs>
   <Content type="html">
      <![CDATA[
         <div id="content_div" style="height: 50px;"></div>

         <script type="text/javascript">
           var div = document.getElementById('content_div');

            function buttonClicked() {
           var value = parseInt(wave.getState().get('count', '0'));
            wave.getState().submitDelta({'count': value + 1});
            }

            function stateUpdated() {
               if(!wave.getState().get('count')) {
                  div.innerHTML = "The count is 0."
               }
               else {
                  div.innerHTML = "The count is " +
                                  wave.getState().get('count');
               }
```

Fig.7

```
            }

            function init() {
               if (wave && wave.isInWaveContainer()) {
                  wave.setStateCallback(stateUpdated);
               }
            }

            gadgets.util.registerOnLoadHandler(init);

         </script>

         <input type=button value="Click Me!" id="butCount"
                                    onClick="buttonClicked()">
      ]]>
   </Content>
</Module>
```

Pour plus de détails, vous pouvez consulter le tutorial : http://code.google.com/apis/wave/extensions/gadgets/guide.html.
Quelques exemples de Gadgets réalisés :

- Are you coming? : permet de mettre en place un sondage dans une Wave ;
- Maps : permet une collaboration sur une carte Google Map au sein d'une Wave ;
- Play Chess : permet de jouer aux échecs dans une Wave.

Pour voir tous les exemples, vous pouvez là encore consulter les samples des API Google Wave et leur code source : http://wave-samples-gallery.appspot.com/.

Robots

Les Robots sont des extensions côté serveur afin d'automatiser des tâches. Il s'agit de participants à des Waves qui peuvent en modifier le contenu, ajouter ou retirer des participants, ou encore transmettre de l'information à d'autres Waves ou au monde extérieur. Elles ont donc les mêmes moyens que les participants humains. [Fig.7]
Les Robots peuvent modifier des Gadgets alors que ces derniers ne peuvent même pas détecter les Robots.
Les Robots sont des applications Web qui interagissent avec les Waves grâce au protocole Wave, qui repose sur HTTP. A l'heure actuelle, ces applications doivent obligatoirement être déployées sur Google App Engine, la plate-forme de Cloud Computing de Google, mais à terme, toute plate-forme implémentant le protocole Wave sera acceptée. Ces applications peuvent donc être écrites en Java ou en Python. Pour ajouter un Robot à une Wave, il suffit d'actionner le menu « Add a participant » et indiquer l'URL de déploiement du Robot (qui se termine donc par @appspot.com).
Voici un exemple simple de Robot écrit en Java. Un Robot est une servlet Java, il faut donc la déclarer de manière traditionnelle dans le descripteur de l'application Web (fichier web.xml) :

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee" version="2.5">
   <servlet>
      <servlet-name>Hello</servlet-name>
      <servlet-class>test.HelloServlet</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>Hello</servlet-name>
      <url-pattern>/_wave/robot/jsonrpc</url-pattern>
   </servlet-mapping>
</web-app>
```

Notez la valeur de l'élément url-pattern, propre au protocole Wave. Voici le code de cette servlet :

```
package test;

import com.google.wave.api.*;

public class HelloServlet extends AbstractRobotServlet {

   @Override
   public void processEvents(RobotMessageBundle bundle) {
      Wavelet wavelet = bundle.getWavelet();

      for (Event e: bundle.getEvents()) {
        if (e.getType() == EventType.WAVELET_PARTICIPANTS_CHANGED) {
            Blip blip = wavelet.appendBlip();
            TextView textView = blip.getDocument();
            textView.append("Hello !");
         }
      }
   }
}
```

Cet exemple simple permet de comprendre le fonctionnement des Robots :

- un Robot est une servlet qui étend une classe particulière, AbstractRobotServlet, fournie par la Robot Java Client Library, téléchargeable à l'URL http://code.google.com/p/wave-robot-java-client/ ;
- un Robot répond à des événements liés à la Wave : notre exemple de Robot est capable de répondre à un seul type d'événements, qui correspond à un changement de la liste des participants de la Wave, mais il pourrait tout aussi bien réagir aux événements suivants : création d'une Blip, changement de titre d'une Wavelet, suppression d'une Blip, etc.
- un Robot peut manipuler la Wave, ses Wavelets et ses Blips.

Il faut enfin fournir un fichier de configuration pour le Robot :

```
<?xml version="1.0" encoding="utf-8"?>
<w:robot xmlns:w="http://wave.google.com/extensions/robots/1.0">
   <w:capabilities>
     <w:capability name="WAVELET_PARTICIPANTS_CHANGED" content="true" />
   </w:capabilities>
   <w:version>1</w:version>
</w:robot>
```

Ce fichier indique la liste des événements auxquels le Robot est capable de réagir ainsi que la version du Robot. Attention, il est impératif de bien mettre à jour cette liste des événements.
Pour plus d'informations, vous pouvez consulter le tutorial Java : http://code.google.com/apis/wave/extensions/robots/java-tutorial.html. Vous trouverez l'équivalent Python sur la page http://code.google.com/apis/wave/extensions/robots/python-tutorial.html.

Quelques exemples de Robots réalisés :

- Complety : ce Robot complète vos phrases en remplaçant « ??? » par des réponses issues du moteur de recherche Google
- Dr. Weather : ce Robot ajoute dans une Wave des informations Météo sur une ville [Fig.9]
- Polly : un Robot permettant d'intégrer des sondages dans une Wave
- Rosy : un Robot qui traduit vos phrases lors de la saisie (je vous invite à jeter un œil à la fin de la démonstration Google IO)
- …
- Et Dr. Maps, un Robot que j'ai créé afin de tester cette plate-forme (voir http://code.google.com/p/golden-waves/). Il permet d'intégrer des cartes Google Map dans une Wave

Pour voir tous les exemples, vous pouvez là encore consulter les samples des API Google Wave et leur code source : http://wave-samples-gallery.appspot.com/.

Buzz ou révolution ?

Google Wave aujourd'hui

Lorsqu'on teste la plate-forme, on peut se rendre compte que l'application Web fonctionne bien… Dans l'ensemble ! En effet, l'application Web, la composante "Produit" de l'outil Google Wave, permet déjà de créer et manipuler des Waves avec des milliers d'autres participants du monde entier. Sa fluidité varie beaucoup : parfois très agréable à utiliser, elle l'est bien moins dans les périodes de grande activité. En effet, beaucoup de Waves atterrissent et évoluent dynamiquement dans notre boîte de réception car elles sont rattachées au groupe rassemblant tous les utilisateurs.

Il y a bien entendu des fonctionnalités non encore développées ainsi que de nombreux bugs, mais cela ne fait que confirmer que la phase de maturation de ce produit va être longue. Ce serait de toute façon le cas puisque ce produit apporte une nouvelle façon de communiquer et de collaborer. Point intéressant, l'équipe derrière Google Wave propose aux développeurs une heure chaque mercredi pour répondre en direct à leurs questions sur les API Google Wave. Le planning est disponible à l'adresse http://code.google.com/events/calendar/. Ceux n'ayant pas d'accès au sandbox peuvent consulter le groupe http://groups.google.com/group/google-wave-api/.

Vous souhaitez vous lancer dans l'aventure ? Google a ouvert depuis septembre des comptes supplémentaires afin de permettre à 120 000 nouveaux utilisateurs d'accéder au Google Wave Developer Sandbox. A vos marques, c'est par ici : https://services.google.com/fb/forms/wavesignupfordev/.

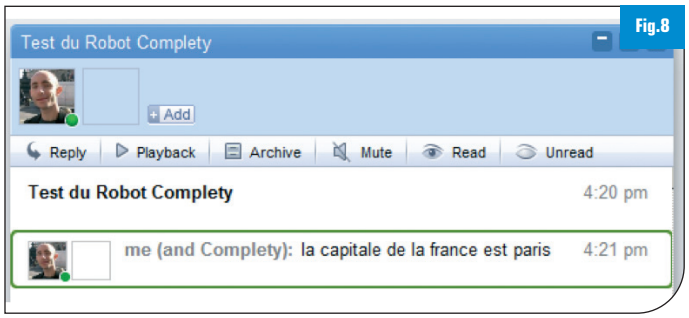


Fig.8

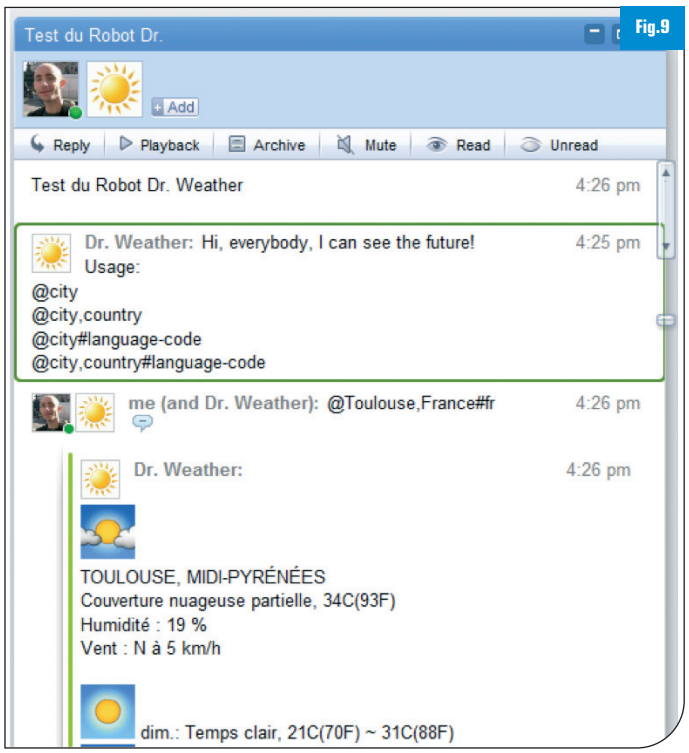


Fig.9

Un peu de recul…

On m'a parfois demandé s'il ne s'agissait pas d'un énième outil à la Twitter, Facebook, E-mail, IM, Wiki, etc. En fait, Google Wave est un outil centralisé pour toutes vos communications et rassemble d'une certaine manière tous ces outils en un seul. **Google Wave n'est pas révolutionnaire parce qu'il propose toutes ces fonctionnalités, mais parce qu'il les rassemble et les unifie de manière cohérente, autour de la notion de Wave.** Et, nous l'avons vu, c'est un outil extensible et ouvert.

Cette ouverture est essentielle : Google Wave et ses trois "P" est un outil révolutionnaire tant il change notre façon de communiquer et de collaborer. Tellement révolutionnaire que son succès n'aura lieu que s'il est largement adopté. C'est pour cela que son ouverture est essentielle. Sa flexibilité et son ouverture le rendent "facilement" intégrable au sein du SI. En effet, il semble destiné à remplacer tous nos outils habituels, mais d'ici là il peut s'intégrer avec chacun d'eux afin d'en être complémentaire.

Ce Google Wave ne laisse pas ses utilisateurs indifférents. Pour en être convaincu, il suffit de se perdre dans les milliers de Waves déjà créées dans le sandbox : beaucoup y voient des intérêts personnels (organisation de rendez-vous au restaurant ou au cinéma, partage de photos de vacances…) mais beaucoup d'autres y voient des intérêts professionnels (partage d'informations multi-sources, communication et collaboration homogènes…)

Il se passe quelque chose, donc. Mais concrètement, il est trop tôt pour prendre la pleine mesure de cet outil.

Quelques liens pour approfondir

http://googleblog.blogspot.com/2009/05/went-walkabout-brought-back-google-wave.html
http://wave.google.com/
http://wave-samples-gallery.appspot.com/
http://www.waveprotocol.org
http://francois.goldgewicht.com/2009/06/03/google-wave-en-quelques-mots/
http://francois.goldgewicht.com/2009/07/30/google-wave-premiers-tests/

François **Goldgewicht**, *Consultant Senior et Directeur Technique d'Aeon Consulting (francois.goldgewicht@aeon-consulting.fr)*

BLU AGE & magicdraw™
AGILE MODEL TRANSFORMATION

présentent

A PARTIR DE
1490 €
PAR UTILISATEUR !

M2Spring™ & M2Code FOR JAVA™

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

Boostez vos développements d'applications Spring & Java !

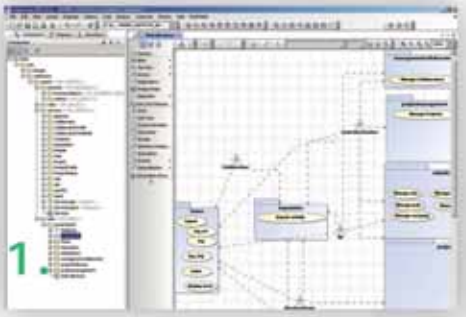
NOUVEAU!
- CRUD Booster
- Construit sur Eclipse 3.5 (Galileo)

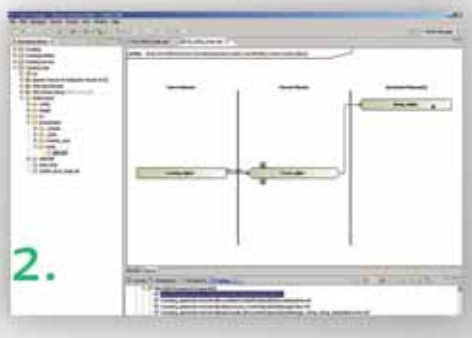
Egalement disponible :

M2Flex™

Pour la génération automatique de vos applications riches !

1. Dessinez facilement vos modèles UML,
2. Validez et débuggez vos modèles,
3. Générez en un clin d'oeil 100% de vos applications !





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

COMSOFT direct
Bechtle's Software Specialist
SOS DEVELOPERS

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

Choisir votre langage de programmation

Un langage de programmation est un moyen, pas une fin. Pourtant, le choix d'un langage peut avoir des conséquences non négligeables sur la vie d'un projet, aussi bien au niveau technique qu'humain. Souvent, ce choix n'est d'ailleurs pas discuté : le langage est imposé, que ce soit par le développeur, le chef de projet, le projet lui-même, le client ou tout autre raison.

Pourtant ce choix mérite qu'on s'y attarde un moment, étant donné la très grande diversité des langages existants. L'évolution des langages suit des modes à court terme et des tendances à plus long terme. On peut établir des classements selon de nombreux critères comme la syntaxe, le typage, le mode d'exécution ou le paradigme de programmation. Ces critères doivent vous guider lors de votre choix.

Paradigme de programmation

Les paradigmes de programmation font l'objet de très nombreuses variantes. Les trois grandes catégories les plus utilisées sont la programmation procédurale (typiquement le C), la programmation objet (Java, Python, Objective C) et la programmation fonctionnelle (Haskell, Erlang). Certains langages, qui n'étaient pas objets, ont évolué progressivement pour le devenir (Perl ou PHP), certains l'étaient dès la conception (Ruby), certains imposent même ce mode (Java), et d'autres peuvent être utilisés selon plusieurs paradigmes (Scala, Python). La programmation objet s'est aujourd'hui généralisée et permet de hauts niveaux d'abstraction. La programmation fonctionnelle revient progressivement à la mode avec des langages comme Erlang ou Haskell, du fait de la globalisation des applications et le cloud computing : ces langages sont idéaux pour créer des applications distribuées sur une grande quantité de nœuds et conserver la cohérence des données sans nécessiter de communication entre les nœuds.

La syntaxe

La syntaxe est le critère le plus visible pour le développeur et donne lieu à de nombreux débats. Ces débats ne sont pas stériles car la lisibilité d'un langage est cruciale pour la maintenance d'un programme et son évolution à moyen et long terme. De très nombreux langages, comme le PHP ou le Java, ont une syntaxe héritée de celle du C qui, si elle est familière à une majorité de développeurs, n'est pourtant pas la plus lisible ni la plus concise. D'autres langages comme Python sont, de ce point de vue, fortement expressifs et mènent à des programmes plus courts. Le Perl, par contre, est réputé pour son manque de lisibilité. Les langages fonctionnels ont généralement une syntaxe un peu déroutante si on vient des langages procéduraux ou objets, mais on s'y habitue assez vite.

Typage

Côté typage, on distingue le typage statique, ou les variables ont un type défini une fois pour toutes, du typage dynamique où les variables peuvent changer de type pendant l'exécution. Le typage peut également être fort ou faible. Un typage fort impose plus de contraintes sur les variables mais offre plus de garanties avant l'exécution. Les conversions implicites entre types y sont, par exemple, interdites.

Vitesse et mode d'exécution

Concernant le mode d'exécution, les deux grandes catégories sont les langages compilés (exécutés directement par le processeur), et les langages interprétés ou fonctionnant sur une machine virtuelle c'est-à-dire la plupart des langages dynamiques. Là, le combat se fait entre rapidité d'exécution et rapidité de développement. L'augmentation exponentielle de la vitesse des processeurs fait qu'il devient possible de tout développer avec un langage interprété. Comme le temps du processeur coûte beaucoup moins cher que le temps du développeur, il est beaucoup plus intéressant d'utiliser un langage hautement expressif comme Ruby, Python ou Scala plutôt que du C ou du C++. Si

certaines routines du programme ont besoin d'être très rapides, rien n'interdit de réécrire juste ces routines dans un langage compilé comme le C. Il faut aussi comprendre que les bibliothèques natives fournies avec les langages peuvent être extrêmement rapides car écrites elles-mêmes en C. En Python, l'utilisation de la bibliothèque NumPy permet par exemple de faire des calculs numériques complexes extrêmement rapides sur des matrices potentiellement énormes, avec très peu de lignes de code. Réimplémenter ces algorithmes de calcul soi-même en C est une très mauvaise idée et peut même être plus lent et plus gourmand en mémoire.

Gestion de la mémoire

La gestion de la mémoire est une tâche ingrate et difficile, source de nombreux problèmes. La majorité des langages modernes, y compris des langages comme l'Objective C, embarquent un « garbage collector », qui se charge de libérer automatiquement la mémoire correspondant aux variables ou objets qui ne sont plus utilisés. La gestion manuelle de la mémoire, nécessaire dans des langages comme le C ou le C++ à moins d'utiliser un garbage collector additionnel, est risquée, gourmande en temps de développement et ne se justifie aujourd'hui que pour les plates-formes embarquées ou l'écriture d'algorithmes hautement optimisés.

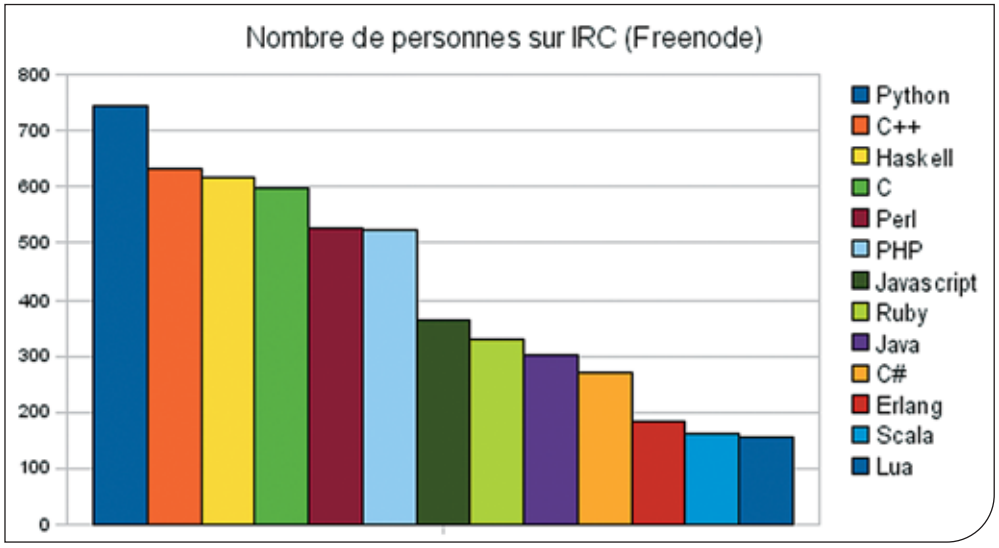
Multi-plate-forme

Aujourd'hui il n'est plus possible de ne raisonner que pour Windows. La croissance fulgurante des plates-formes mobiles de type Android ou iPhone, l'universalité des applications web et le développement des systèmes Mac OS X et Linux imposent de réfléchir au caractère multi-plate-forme des applications le plus tôt possible. Heureusement, la plupart des langages modernes dynamiques sont capables de fonctionner quasiment partout à condition de prendre un minimum de soin. Ainsi, le « *Write once, run everywhere* » n'est pas réservé à Java, mais s'applique à la plupart des langages dynamiques modernes, et quelques langages statiques.

Sécurité

Le choix du langage et du mode de développement peut influencer la qualité des programmes. Les langages compilés à typage statique fort sont beaucoup plus contraignants et vont avertir le développeur très tôt des failles potentielles, mais ne préviendront pas des erreurs de conception ou de logique.

Les langages fonctionnels comme le Haskell offrent des avantages intrinsèques supplémentaires grâce à la notion de pureté des fonctions (des fonctions dont le résultat ne dépend d'aucune variable globale ou environnementale). Avec les langages à typage dynamique faible, la stratégie utilisée est d'accorder la plus grande importance aux tests, au point



d'aboutir à du développement dirigé par les tests : on écrit d'abord les tests correspondant au résultat qu'on souhaite obtenir, puis seulement ensuite l'implémentation. Si les tests passent, le programme est considéré comme correct.

Si un problème est détecté, on écrit d'abord un test (unitaire ou fonctionnel) correspondant au problème, puis on corrige le programme. On peut étendre ce principe à la documentation, comme ce qui est pratiqué en Python avec la notion de « doctest » : un mélange de test et de documentation, qu'on peut écrire dès la phase de conception. Des outils de test et de sécurité en ligne de commande existent également en PHP (PHPUnit, PHP_CodeSniffer).

Licence

Un langage en soi n'est pas soumis à une licence, mais les outils le sont, c'est-à-dire les compilateurs, les interprètes, les bibliothèques et tout ce qui fait vivre un langage. Prenez toujours le temps de lire le petit texte juridique qui accompagne les outils que vous utilisez car il arrive qu'on ait des surprises.

Les outils et bibliothèques de base du langage ne devraient imposer aucune contrainte sur les programmes réalisés. C'est le cas pour la quasi-totalité des langages issus du milieu de l'open-source, qui vous permettent de créer aussi bien des logiciels libres que des logiciels propriétaires.

Les notions de copyleft applicables portent sur les outils eux-mêmes, pas sur les programmes que vous développez.

Type de projet

Chaque langage a ses domaines de prédilection. Ceux dans lesquels ils peuvent être utilisés dépendent en grande partie de la richesse de ses bibliothèques. Python est connu pour être « fourni avec les piles », c'est-à-dire que sa bibliothèque standard permet d'origine de tout faire. Des bibliothèques additionnelles autorisent le développement de grosses applications scientifiques, calcul numérique, visualisation 2D ou 3D, interfaces graphiques. Il est également utilisé depuis plus de dix ans pour le développement web orienté objet. Java est également très riche de ce point de vue et peut être utilisé pour tout projet. Le PHP n'est à l'aise que pour des applications web mais possède une visibilité et une communauté énormes. Perl et Ruby sont tout autant généralistes et utilisés également dans de nombreux domaines. Ruby, très utilisé au Japon, s'est fait surtout connaître en Europe via son framework web Ruby On Rails, dont les équivalents PHP et Python se nomment Symfony, Zend Framework, Pylons ou Django. Dans le monde mobile, on retrouve l'Objective C sur l'iPhone, et le Java sur Android. Des langages spécialisés existent aussi pour des domaines très précis, comme le « R » ou le « S » pour les calculs statistiques. Le C# est lui aussi généraliste, mais reste lié à Windows même s'il existe des implémentations libres (Mono) sous Linux.

Écosystème, communauté

Outre la qualité et la quantité des outils et des bibliothèques, il faut aussi veiller à la taille de la communauté. Il est intéressant de choisir un langage soutenu par une grosse communauté de personnes ou d'entreprises. Il peut aussi être intéressant de suivre les tendances. Plusieurs indicateurs existent, comme l'index TIOBE, qui donne dans l'ordre pour Octobre 2009 : Java, C, PHP, C++, Basic, C#, Python, Perl, Javascript et Ruby. Le graphique ci-contre correspond au nombre de personnes connectées aux canaux IRC correspondant à chacun des langages (un mardi vers 17h30). Ce n'est pas représentatif des parts de marché, mais c'est un indicateur comme un autre qui montre l'intérêt des développeurs pour les langages.

Conclusion

Le meilleur langage est en pratique celui que vous maîtrisez ou que l'équipe connaît. Car c'est avec celui-ci que le travail sera effectué le plus rapidement. Néanmoins, en tant que développeur, et même si certains langages permettent de tout faire et d'aller très vite, il est vital de maîtriser plusieurs technologies. En acquérant de bonnes notions de développement objet et de développement fonctionnel, vous serez à même d'aborder n'importe quel problème sous le meilleur angle possible.

En maîtrisant un langage dynamique interprété et un langage bas niveau compilé, vous bénéficierez à la fois de la vitesse de développement et de la vitesse d'exécution si besoin. Quant au choix final, si comme vous l'aurez compris ma préférence va à Python, mon meilleur conseil est de se forcer à apprendre un nouveau langage régulièrement, vous n'aurez que l'embarras du choix.

Et vive la diversité !

■ Christophe **Combelles**
Consultant, Responsable du centre de compétence Python - Alter Way Solution.

Paroles de développeur

Le choix du langage, il y a fort, fort longtemps...

par Richard Clark

« Quand je me suis lancé dans l'informatique en 1994, la première étape fut le choix de l'outil de développement.

L'enseignement que j'avais suivi (Louis-le-Grand et Centrale Lyon) des années auparavant ne m'avait donné aucune base, mais il faut bien reconnaître qu'à l'époque, développer sur un micro-ordinateur était plutôt considéré comme un hobby qu'un vrai métier. J'avais bien fait un peu de Pascal sur des gros systèmes mais le but était clairement l'apprentissage de l'algorithmie plutôt que le développement concret d'applications.

De plus, à l'époque, Internet ne faisait que poindre le bout de son nez (Compuserve venait à peine d'arriver en France) et seul mon entourage, aussi pointu en informatique que moi en broderie, pouvait m'aider dans ce choix. Comme Delphi et Java ne feront leur apparition que l'année suivante, j'ai donc considéré les solutions suivantes : C / C++, Turbo Pascal (déjà Anders Hejlsberg) ou Visual Basic 3.

Un an seulement après avoir choisi Visual Basic 3, je créais ma première vraie application, la gestion des documents techniques de la Cité Internationale de Lyon. Visual Basic prouvait que le développement RAD pouvait être productif. Par la suite, j'ai continué avec les différentes versions de Visual Basic (4-6) pour naturellement adopter Visual Basic .NET en 2002 avec l'arrivée du Framework .NET.

Si je vous ai présenté mon parcours d'autodidacte, c'est que j'ai eu la chance de ne pas avoir à me poser trop de questions. Apprendre Visual Basic ne m'a pris que quelques mois et j'ai rapidement pu créer mon application tout en continuant mon « vrai » travail.

De nos jours, la question peut paraître plus pointue car il existe de nombreux langages pour différentes plates-formes qui permettent d'être productif.

C, C++ forever

Mais avant tout, il ne faut surtout pas oublier les applications qui nécessitent une programmation de bas niveau, une optimisation des performances comme les jeux, le temps réel, l'embarqué et / ou la communication avec des appareils externes spécifiques. Dans ce cas, il n'y a pas le choix : C, C++ étant le passage obligé. J'ai rencontré il y a quelques années Michael Capps, le président de Epic Games (Unreal, Gears of War, etc.) et pour lui, le profil type est un pro du C++ avec de fortes connaissances en assembleur et une passion pour le multithreading. Certes, les jeux qu'il développe sont atypiques (ce sont plutôt des moteurs 3D) mais le temps du créateur de jeux isolé dans sa chambre est bien loin.

Et pour le Microsoft .NET ?

Quand Microsoft a révélé le Framework .NET en 2002, l'un des objectifs annoncé était l'unification de l'approche de la programmation, l'indépendance du langage. Et pour en faire la démonstration, nous avons eu droit à pas moins de 4 langages différents : C++ managé, Visual Basic .NET, C# et J#. Oublions tout de suite J# et C++ managé et concentrons-nous sur les deux principaux langages, VB .NET et C#.

Anders Hejlsberg, débauché de Borland par Microsoft dans les années 90, principal architecte du C#, a pu créer en partant d'une page vierge, un langage parfaitement adapté à la plate-forme .NET tout en bénéficiant de l'expérience Java.

Considéré à l'époque soit comme du C++++, soit du J--, c'est-à-dire une copie du Java, C# a connu un rapide succès auprès des développeurs de tout horizon. Il a séduit les développeurs C++ par une syntaxe proche sans sa complexité de mise en œuvre ainsi que les développeurs Java pour la même raison. C'est avec amusement que j'ai pu constater alors que les détracteurs de Visual Basic (VB est un T-RAD : Trop Rapid Development !) adoptaient à la vitesse grand V le C# tout en dénigrant Visual Basic .NET.

Visual Basic de son côté connaissait une véritable révolution. Certes on pouvait avoir une approche objet depuis VB4 comme le démontrait l'excellent livre de Bruce Mc Kinney, *"Hardcore With Visual Basic 4"* ou le

»

Anecdote

La première version de VB ne supportait pas les commentaires Xml pour générer la documentation de code. Un développeur de Microsoft a alors créé un petit utilitaire pour ajouter cette fonctionnalité, utilitaire développé en C# ! Alors qu'il allait le publier « officiellement », on lui a fait remarquer que ce serait bien qu'il soit écrit en VB et il a alors passé son week-end à le réécrire.

framework métier développé par Rockford Lothka, la CLSA, mais ce n'était pas dans la culture des VBistes. La nouvelle mouture de VB sauce .NET en a désorienté plus d'un au point que la communauté VB6 (1998 !) est toujours active.

Entre les deux langages on trouvera toujours des petites différences, des fonctionnalités qui existent en VB et pas en C# et inversement mais fondamentalement, il n'y a pas de différence. Alors la question est : lequel choisir et pourquoi ?

Bien qu'ayant été MVP Visual Basic .NET pendant 7 ans, il est clair que C# est LE véritable langage pour la plate-forme .NET. A cela il y a plusieurs raisons. Tout d'abord, C# est le langage de prédilection utilisé en interne chez Microsoft.

Vous pouvez le constater aisément en consultant les différents SDK et documentations publiées, ils sont tous avant tout en C# puis traduits (ou pas comme XNA) en VB .NET. A ce propos d'ailleurs, il faut reconnaître que Microsoft effectue un gros effort en ce moment pour assurer la diffusion de ses SDK, exemples et autres dans les deux langages et, bonne nouvelle, c'est une des priorités pour l'année qui vient aussi bien pour la documentation de base (MSDN Library) que pour des SDK dits de pointe comme Prism, MEF, Windows7 API Code Packs, Entity Framework, Silverlight, etc.

Ensuite, si vous recherchez des exemples de code, de l'aide sur Internet, vous constaterez que C# est nettement plus présent que VB .NET. Codeplex par exemple regorge de dix fois plus de projets en C# qu'en VB .NET ! Si l'on prend maintenant l'exemple des communautés CodesSources, bien que le taux de fréquentation annuel tous langages confondus soit en forte hausse, la partie VB a diminué de près de 25% en un an.

Ecrire un ouvrage sur C# est « beaucoup » plus lucratif que le même en VB .NET. Le meilleur exemple étant les ouvrages de Rockford Lothka qui existent dans les deux langages, qui traitent donc exactement du même sujet et qui se vendent nettement mieux dans leur version C# (mais je vous rassure, ce n'est pas en écrivant des livres de programmation que vous deviendrez riches ;-))

Côté enseignement, il en est de même et c'est surtout côté entreprises que l'exemple est le plus flagrant. Il peut arriver que la réécriture d'une ancienne application VB5/6 soit envisagée d'être réécrite en VB .NET mais cela reste l'exception. En 7 ans de consulting dans les entreprises, le nombre de projets VB .NET se comptent sur les doigts d'une main !

Il y a aussi un autre argument plus pernicieux qui prône l'utilisation de C# : sa ressemblance avec Java.

Le développement d'applications pour mobiles est en pleine expansion avec de nouveaux acteurs qui ont fait leur apparition ces dernières années comme l'IPhone d'Apple et Android de Google. Je ne dis pas que passer du C# au Java sera une partie de plaisir, mais vous serez moins dépaysé que de passer de Visual Basic à l'Objective C.

Approche VB ou .NET ?

Visual Basic .NET pour ne pas trop désorienter les anciens développeurs VB6 permet d'écrire des lignes « à la VB6 ». Exemple :

```
Dim s as String = Format(Now)
```

Je ne saurai que déconseiller cette approche. *Format* comme *CInt, Now, MsgBox* sont spécifiques au compilateur VB qui les transforme, in fine, par des appels à des méthodes du framework (souvent par un passage dans Microsoft.VisualBasic.dll).

Certes vous obtiendrez ce que vous voulez, mais vous n'appréhenderez pas toutes les subtilités du framework et donc toutes ses possibilités.

Par exemple dans cette ligne, il faut savoir que le formatage de la date prend en compte la Culture du Thread Courant. Il est donc préférable d'écrire :

```
Dim s as String = DateTime.Now.ToString(CultureInfo.
CurrentCulture)
```

Iron Python, F#, etc. ?

Depuis quelque temps, de nouveaux langages apparaissent issus généralement des recherches de Microsoft Research. On y retrouve des adaptations d'anciens langages à la sauce .NET ou de nouveaux (généralement suffixés par un #). Ils ont des utilités dans des approches vraiment spécifiques mais n'ont pas un caractère « généraliste » comme C# et VB .NET.

Pour conclure

Historiquement, le Basic (1975), QuickBasic (1985) puis Visual Basic (1991) ont été des piliers du succès de Microsoft. Force est de constater que depuis la naissance du .NET Framework, il est en déclin. Il n'y a pas à s'en réjouir ou à s'en lamenter, c'est un état de fait. C# est devenu de fait le langage de prédilection du .NET Framework et donc le langage à adopter si vous voulez programmer sur la plate-forme de Microsoft.

■ Richard **Clark**
Consultant et formateur .NET, gérant de Clark Ingénierie Informatique.
MVP .NET de 2002 à 2009, il anime le site www.c2i.fr *ou vous retrouverez de nombreux articles et actualités sur cette technologie.*

Paroles de développeur

l'embarrassante question du choix

par Frédéric Mazué

“ Programmer, écrire du bon et beau code, est difficile. Choisir le langage est plus difficile encore.

Pour enfoncer un clou, un marteau convient mieux qu'une paire de ciseaux, et pour dévisser une vis, le tournevis convient mieux que le cure-dent. C'est évident pour tout le monde, et de même c'est une évidence pour l'informaticien qu'il doit utiliser le langage approprié à un développement. Si tous les langages servent à développer, beaucoup sont aussi différents l'un de l'autre qu'un marteau l'est d'une paire de ciseaux, et beaucoup de langages qui prétendent être différents des autres n'en sont en fait que des clones déguisés. Les langages informatiques pullulent, choisir "le bon qui va bien" est sans doute la partie la plus délicate du travail de développement, et aussi celle qui a la plus grande répercussion sur ce travail. Pour corser encore la difficulté, si l'on connaît l'avenir du clou et du marteau, qui peut dire aujourd'hui comment évoluera une application, ou l'outil qui a servi à la développer ?

Recenser les points cruciaux

L'application doit elle est écrite rapidement ? Doit-elle être performante ? Livrer le code source (langages de script) au client est-il envisageable ? Est-ce une application "classique", genre répertoire téléphonique, ou "ultra-moderne", par exemple concurrente et distribuée ? Voici sans doute, avec les éventuels problèmes de déploiement, les premiers points à examiner. Pour une application à écrire vite, plus le langage est expressif et de haut niveau, mieux c'est. On préfèrera alors un Python à un C++. Mais si l'application doit être très performante, peu gourmande à l'exécution, ou si l'on veut qu'elle soit autonome (un simple exécutable) le choix sera contraire, avec pour C++ une très forte répercussion sur le temps d'écriture. Pour une application entre les deux, un langage entre les deux comme C#, à la fois productif à l'écriture et relativement bon à l'exécution, sera sans doute un bon choix.

Ne pas suivre la mode ni le marketing

Si une politique d'entreprise n'impose pas le choix du langage, on se méfiera de la mode et du marketing. On nous dit volontiers que des Java ou des C# sont simples, et bons à tout faire. Est-ce vrai ? Dans certains cas oui, mais pas nécessairement toujours. Une considération pertinente pour faire son choix est : qu'est-ce qui a motivé la création de ce langage ? Erlang par exemple a été créé pour écrire des applications concurrentes distribuées. Un domaine très concret et ciblé. Et il est sans égal pour cela, alors que Java et C# restent très "artificiels" dans ce domaine. Aux yeux de votre serviteur, un langage créé pour être plus propre, plus clair, plus facile que le voisin, est très suspect. Et puis cette simplicité est-elle au rendez vous ? Java, c'est 684 pages de spécifications. C++, et en dépit de sa réputation de complexité, c'est seulement 385 pages de spécifications, si l'on excepte la librairie standard. En outre, il est hélas trop facile d'utiliser de façon compliquée un langage réputé simple, alors qu'il n'est pas difficile d'utiliser de manière simple un langage réputé compliqué, paradoxalement. La facilité d'un langage est donc toute relative. En ce moment, le paradigme objet est à la mode. L'objet c'est très bien, mais ce n'est pas la panacée. Cela peut se révéler lourd à manier, coûteux à l'exécution, pas suffisamment expressif lors de la programmation. Le paradigme fonctionnel peut se révéler meilleur, car plus expressif. La tendance est aussi trop souvent de ne retenir qu'un langage pour un développement. Pourquoi pas plusieurs langages ? Par exemple F# (fonctionnel) et C# pour une application .Net. Chaque langage étant utilisé sur la partie de l'application qui lui convient le mieux. Ou pourquoi pas Python et C++, sachant que des outils tels que Swig permettent d'interfacer automatiquement les deux. Autre aspect : le bon fonctionnement est-il crucial, ou le bug tolérable ? Dans le premier cas, on se tournera vers un langage qui permet de prouver la correction du code écrit, comme avec Haskell ou Caml par exemple.

Faire confiance aux valeurs sûres

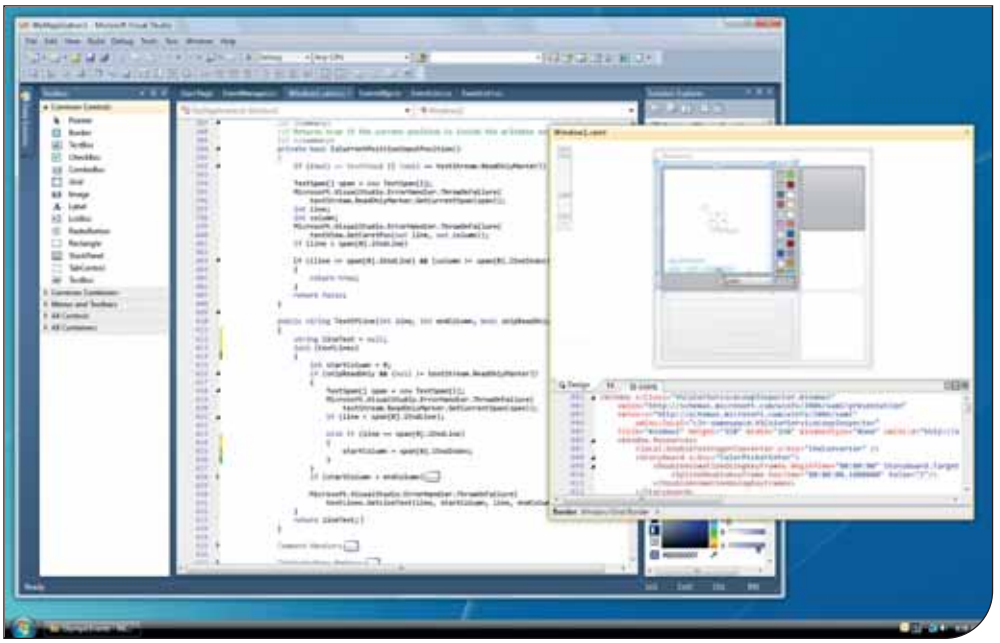
C'est sans doute une erreur de ne regarder que les langages "récents", alors que d'autres, même s'ils tombent dans l'oubli, sont de vraies valeurs sûres. Ainsi un Ada tombe en désuétude pour le grand public, alors que les entreprises qui l'utilisent encore le décrivent comme un avantage concurrentiel. Lisp lui aussi tombe dans l'oubli. Mais pourtant, pour des développements très complexes, avec des conceptions en bottom-up, c'est souvent lui qui est retenu. Ces "valeurs sûres", on pourrait encore citer COBOL, ne devraient jamais être négligées lors du choix, sous prétexte qu'elle sont anciennes. En outre, ces langages, et avec eux Pascal, ou C, ou C++, sont standardisés, ce qui est un point crucial.

Se former plutôt que galérer

Enfin, rejeter un langage parce qu'on ne le connaît pas bien, est probablement une erreur. Mieux vaut se former, plutôt que galérer avec un outil inadapté. Au final, cela coûtera moins cher. ■ ”

Le bon langage au bon moment, pour le bon développement

Vous aurez compris que choisir un langage est loin d'être une sinécure même si, dans certains cas, le choix sera évident, car imposé par la plate-forme ou le client final. Et finalement, chacun aura sa propre vision du langage même si des critères communs existent. Toutefois, entre préjugés et sectarismes, il s'avère difficile de toujours suivre le bon chemin…



L'outillage est aussi important que le langage lui-même.

Aujourd'hui, la profusion de langages peut faire dérailler son algorithme à plus d'un développeur. Mais en réalité, seule une poignée d'entre eux domine le développement serveur, desktop, mobile, embarqué et web. Le marché de l'emploi est lui aussi très largement dominé par un nombre limité de langages, pareillement en école, en SSII. On pourrait presque résumer ainsi : C++, Java, .Net (C#, VB), html, javascript, PHP. Selon **David Négrier** (The Coding Machine) :*« Il y a pléthore de langages en effet mais peu sont utilisés. Un grand nombre se rencontre plutôt en université comme D ou les langages fonctionnels. Par contre, des langages comme Ruby peuvent percer, cela montre que l'on n'est pas figé sur Java, .Net, PHP. Pour moi, un langage est utilisable quand il est outillé, avec un choix de librairies ».* Pour **Xavier Monnier** (responsable pôle développement & CMS Uperto) : *« Le choix est très variable selon le projet même si, finalement, on n'a pas beaucoup de choix. Il se fait selon l'expression des besoins, des critères objectifs. Il y a par exemple la contrainte client qui porte sur telle ou telle technologie ».* Ces contraintes peuvent peser très lourd sur la productivité du code et les performances finales de l'application, surtout quand il existe des contraintes de cohabitation ou d'intégration avec l'existant. *« Les jeunes diplômés, étudiants ont souvent un positionnement idéologique, le choix se fait avec les tripes. Mais dans le monde professionnel, il faut avoir une abstraction car le jeune diplômé peut avoir un rôle de conseil, et il est écouté. Il n'est pas là pour faire du prosélytisme »,* poursuit Xavier Monnier.

David Négrier

Xavier Monnier

« Il y a différents axes à cette question : les fonctions, les librairies, la communauté. Nous travaillons beaucoup sur le web essentiellement avec Java, PHP. Les langages sont très différents. Il en existe deux grandes catégories: langage typé et non typé. Java est typé avec une phase de compilation détectant les erreurs, contrairement à javascript. D'autre part, en Java, le développeur outille beaucoup, donc on peut réaliser beaucoup de contrôles, de vérifications », analyse à chaud David Négrier.

Valoriser et apprendre

Il ne s'agit pas non plus de changer sur un coup de tête de langage. Quand un développeur a acquis une compétence sur un langage, les outils, les librairies, il n'aura pas envie de les perdre. Il faut les valoriser, les exploiter au mieux dans son travail et surtout ne pas oublier une chose essentielle : garder l'esprit ouvert, faire de la veille technologique tout le temps. Et bien entendu suivre les évolutions du langage.

L'ouverture d'esprit est vitale pour tout développeur. Il ne faut pas être dogmatique pour le plaisir. Cela n'a aucun sens. Car il est possible qu'un jour, il puisse apprendre un langage du côté obscur !

C'est quoi choisir le bon langage au bon moment pour le bon développement ?

Cette question est plus complexe qu'il n'y paraît. Pour choisir le bon langage, il faut déjà savoir ce que vous

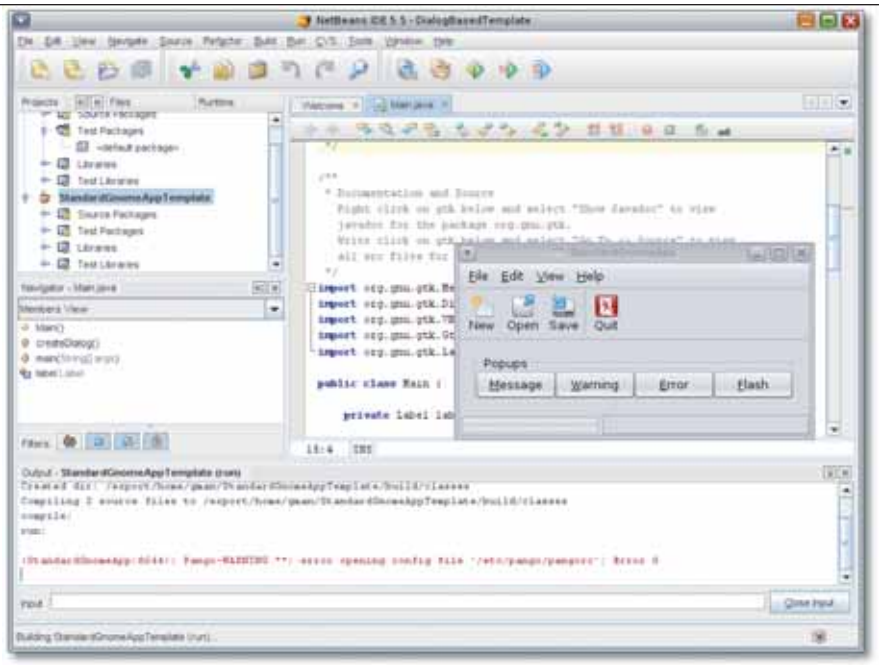
devez développer :
- application web
- application réseau
- application desktop
- application mobile

A partir de là, vous pouvez peaufiner vos besoins avec une matrice de type :
- quelle plate-forme matérielle (ex. : desktop, mobile Windows, etc.)
- quelle plate-forme logicielle / système (iPhone, Windows Phone, Windows 7, MacOS X, Linux Ubuntu, RIA, etc.)
- le type d'exécution : native (exécutable), via un runtime d'exécution (Flash, Flex, .Net, Java)

Bref, définir le système cible de votre future application. Plus vous maîtrisez votre plate-forme cible d'exécution, mieux c'est pour le projet. Par exemple, si vous ciblez une version unique de Linux ou encore un type de navigateur ou une plate-forme mobile. Car les impératifs de système cible peuvent conditionner le choix final de votre langage. Par exemple :
- iPhone : Objective-C et outils Apple, à la rigueur vous pouvez tester MonoTouch, pour les applications natives. Pour le site web, vous ciblerez les langages web classiques tout en prenant en compte les contraintes mobiles du téléphone (comme toujours pour un site mobile).
- Windows Phone : soit en C++, VB embarqué, voire Java.

Pour un développement web, vous aurez à votre disposition des plates-formes de type Flex, Silverlight, JavaFX ou un environnement site web « classique » : html, CSS, pages dynamiques avec PHP, Ajax, ASP, etc.

Vous pouvez aussi prendre en considération l'outillage. Car le langage n'est pas une fin en soi et vous serez peut être plus à l'aise avec un outil de type IDE (environnement graphique et intégré) qu'avec un éditeur de code textuel. Si pour le développement web, vous souhaitez bénéficier d'un environnement très complet, le choix sera vite fait et limité : Dreamweaver ou Expression Web. Même remarque



avec DirectX, OpenGL, en dehors de C++, point de salut !

Pour The Coding Machine, la checklist pour choisir ressemble à ceci :

1 : regarder les frameworks existants sur les langages, la communauté, l'écosystème

2 : langage typé ou non typé

3 : les outils

Norme et pérennité

Le développeur choisira (tout comme en SSII, entreprise) un langage dont la pérennité est assurée. On revient une fois de plus à la notion de communauté. Un langage open source pourrait donc avoir une meilleure image par son côté ouvert mais un langage open source n'est pas non plus un gage de pérennité. Des langages propriétaires gardent une grande importance, à l'image d'un VB, C# a été finalement ouvert (permettant des piles open source comme Mono).

L'usage d'un langage normé, c'est-à-dire ayant une norme, est un autre argument important pour l'implémentation, la portabilité du code. C'est le cas pour C++ (si on opte pour une version normée). L'utilisation des standards est aussi à privilégier. Le développement web en est un bon exemple malgré les problèmes de compatibilité. Sur ce cas, malheureusement, l'implémentation varie souvent d'un éditeur à un autre, donc il faudra vérifier la qualité de l'implémentation. Autre point délicat, l'interopérabilité. Par exemple, impossible de passer d'un code Flex à un code Silverlight sans réécrire. Et passer d'une librairie Ajax à une autre n'est pas une affaire de plaisir !

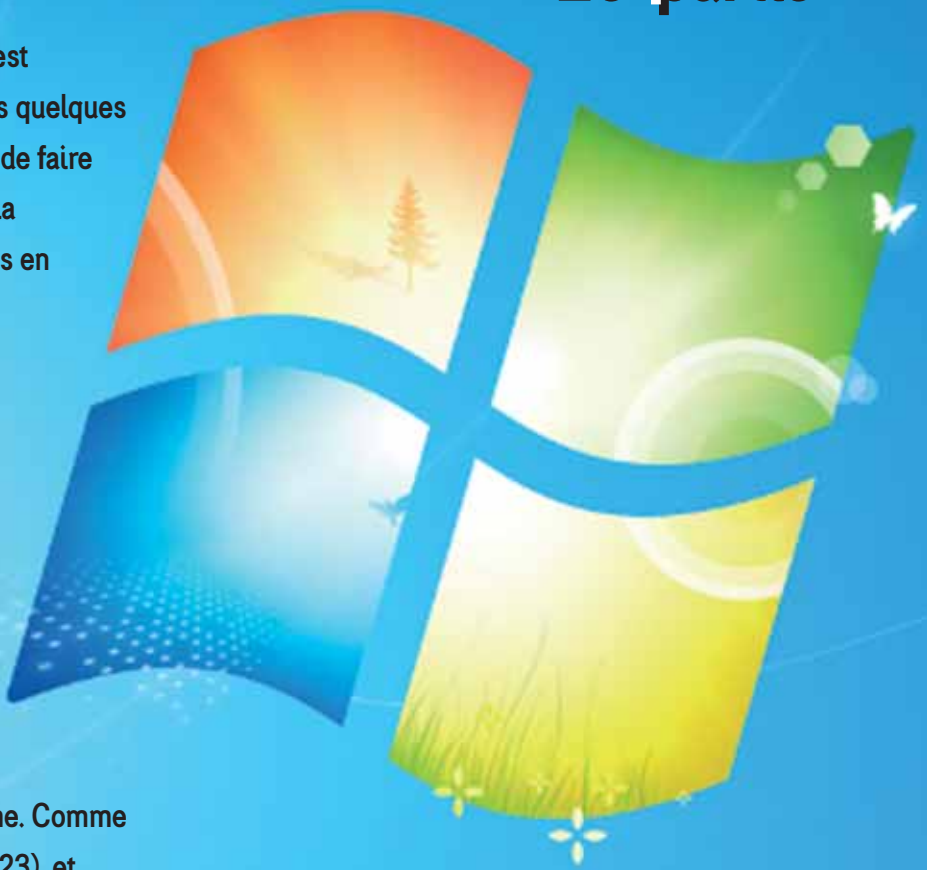
■ François **Tonic**

C++ : valeur sûre du développement

« Beaucoup d'étudiants se dirigent sur ce langage, surtout pour la finance, l'embarqué » commente Yann Lautredou (directeur laboratoire Supinfo). A l'Epitech, les étudiants de 2e année sont immergés dans le C++, dans une « piscine ». Comme nous l'a précisé **Salomon Brys,** les étudiants sont formés à être très rigoureux : présentation du code très stricte, s'il manque une règle ou qu'elle n'est pas respectée, c'est un point en moins. L'objectif étant que l'étudiant fournisse un code le plus indépendant possible de la plate-forme, tout en respectant les normes du langage.

Windows 7
Le guide du développeur
2e partie

Après des mois d'attente, Windows 7 est désormais disponible pour tous depuis quelques jours ! La mission de cette version est de faire oublier l'épisode Vista et de relancer la machine Windows sur de bonnes bases en attendant les vraies ruptures technologiques et systèmes dans les prochaines années.



Pour cette seconde partie, nous continuerons notre exploration des technologies Windows 7 pour le développeur : C++, VHD, le pack troubleshooting et Internet Explorer 8. Les nouveautés et améliorations se cachent un peu partout dans le système. Comme nous l'avions vu dans la 1re partie (n°123), et courant 2010, les possibilités de Windows 7 exploseront avec la disponibilité de Visual Studio 2010 et de .Net 4.0.

C'est désormais à vous, développeurs, d'optimiser, d'adapter, de porter vos codes, vos applications sur Windows 7 et de faire le feedback à Microsoft, aux éditeurs. Tout naturellement, nous suivrons de près la nouvelle aventure Windows 7 avec le déferlement des mises à jour dans les prochains mois.

Si vous avez déjà un retour, un commentaire, bon ou critique envers Windows 7, envoyez-le : rédaction@programmez.com

Bon Windows 7 !

■ François Tonic

Les nouveautés C++

Développer des logiciels, n'a jamais été aussi excitant qu'aujourd'hui. Nous sommes dans un monde de plus en plus connecté, interactif et mobile. Les ordinateurs n'ont jamais bénéficié d'autant de puissance, permettant le développement d'interfaces graphiques innovantes, naturelles et plus intuitives et ouvrant la voie ainsi à de nouveaux scénarios et opportunités.

Dans cet article, je me propose de faire un tour d'horizon de quelques nouveautés disponibles dans Windows 7, car par manque de place, nous ne pourrons malheureusement pas les couvrir toutes. Les exemples de cet article ne sont pas complets. Ils vous donnent simplement la démarche à suivre. Sachez néanmoins, que vous pourrez les trouver complets et détaillés dans le Kit de développement Windows 7.

La nouvelle barre des tâches Windows

La barre de tâches de Windows 7 est le résultat de plusieurs années d'évolution. Elle permet : le lancement rapide d'applications, la bascule rapide entre chaque application et fenêtre, la gestion récente et fréquente des destinations utilisateurs, l'accès à des tâches communes, le rapport de progression et la notification d'états au travers d'icônes, et enfin, le contrôle de l'application sans sortir de la barre de tâches, à partir de vignettes de visualisation. Elle devient le point central pour informer l'utilisateur sur la progression et la santé de l'application, remplaçant les raccourcis et le System Tray de Windows Vista. Prenons un exemple concret : l'intégration de la *Jump-list.* Lorsque nous cliquons sur le bouton droit de la souris au niveau de l'icône de l'application dans la barre des tâches, la liste des fichiers récents apparaît, ainsi qu'un certain nombre de tâches que l'on peut ajouter. Comme sur la figure suivante : [Fig.1]. Pour intégrer sa propre liste, il faut tout d'abord, indiquer à Windows l'extension de fichier que l'application utilise. Dans notre exemple nous utilisons l'extension *.evxaml.*
Vous devez définir un *APPID* pour l'application.

```
const LPCWSTR APPID =L"WindowsSept_HostingXAML";
```

Ensuite, indiquez à Windows l'APPID que vous allez utiliser.

```
hr= SetCurrentProcessExplicitAppUserModelID(APPID) ;
```

Puis associer l'extension de fichier à l'*APPID* et à l'application.
Voici un exemple de fichier de registre Windows.

```
[HKEY_CLASSES_ROOT\.evxaml]
 [HKEY_CLASSES_ROOT\.evxaml\OpenWithProgIds]
"WindowsSept_HostingXAML"=""
 [HKEY_CLASSES_ROOT\WindowsSept_HostingXAML]
"FriendlyTypeName"="@shell32.dll,-8975"
"DefaultIcon"="@shell32.dll,-47"
"CurVer"="WindowsSept_HostingXAML"
"AppUserModelID"="WindowsSept_HostingXAML"
 [HKEY_CLASSES_ROOT\WindowsSept_HostingXAML\shell]
@="Open"
 [HKEY_CLASSES_ROOT\WindowsSept_HostingXAML\shell\Open]
[HKEY_CLASSES_ROOT\WindowsSept_HostingXAML\shell\Open\Command]
@="c:\\HostingXAML.exe /doc:%1"
```

Maintenant, en double-cliquant sur un fichier ayant pour extension *.evxaml,* l'application se charge automatiquement avec comme paramètre le chemin d'accès au fichier. Il suffira alors d'indiquer à Windows de l'insérer dans la liste des fichiers récents à l'aide de l'API du Shell *SHAddToRecentDocs.*

```
::SHAddToRecentDocs (SHARD_PATHW,psz); (ou psz correspond au
chemin du fichier).
```

Si vous utilisez la boîte de dialogue standard d'ouverture de Windows (*IFileDialog2*), celle-ci l'ajoute automatiquement pour vous.

Les Rubans

Les Rubans Office 2007, sont désormais disponibles en standard dans Windows 7. Ils permettent aux développeurs de fournir à l'utilisateur une interface riche et beaucoup plus intuitive. Par exemple, vous pourrez donner un accès rapide à l'utilisateur, aux fonctionnalités les plus utilisées à l'aide de la barre d'outils d'accès rapide. Les rubans permettent également aux utilisateurs de retrouver facilement et rapidement des fonctionnalités qui seraient dans une application traditionnelle Win32, cachées au fin fond d'un menu. Les rubans sont constitués d'un menu « à la office », d'une barre d'outils d'accès rapide, d'onglets contextuels, et de contrôles (des commandes) regroupés par thème. [Fig.2]
Les API du Ruban se divisent en 2 parties :

- Des balises XML pour définir la structure et l'organisation des contrôles Ruban
- Des interfaces COM à initialiser, pour la gestion des évènements et des commandes.

Fig.1

Fig.2

Le fichier XML est divisé en deux parties. La partie définition des commandes *<Application.Commands>* et la partie Visualisation *<Application.Views>*, regroupant les commandes par onglet, puis par thème. Les commandes peuvent être représentées par plusieurs types de contrôles. Bouton, CheckBox, combobox, etc...

```
<Application  xmlns='http://schemas.microsoft.com/windows/
2009/Ribbon'>
  <Application.Commands>
    <Command Name="TabIcone" Symbol="cmdIcone" Id="41000">
      <Command.LabelTitle>
        <String Id ="410">Icônes</String>
      </Command.LabelTitle>
    </Command >
    <Command Name = "GroupProgression" Symbol="cmdGroupProg
ression" LabelTitle="Barre de Progression"/>
    <Command  Name="ProgStatusNormal"  Symbol="cmdProgStatus
Normal" Id="32787" LabelTitle="Etat Normal">
      <Command.LargeImages>
       <Image Source="round glass glassy bullet green 1.bmp"/>
      </Command.LargeImages>
    </Command>
<Command Name="ProgStatusErreur" Symbol="cmdProgStatusErreur"
 Id="32793" LabelTitle="Etat Erreur">
           <Command.LargeImages>
       <Image Source="round glass glassy bullet red 1.bmp"/>
      </Command.LargeImages>
    </Command>
  </Application.Commands>
  <Application.Views>
    <Ribbon>
      <Ribbon.Tabs>
        <Tab CommandName="TabIcone">
          <Group CommandName="GroupProgression">
            <Button CommandName="ProgStatusNormal" />
            <Button CommandName="ProgStatusErreur" />
          </Group>
        </Tab>
      </Ribbon.Tabs>
    </Ribbon>
  </Application.Views>
</Application>
```

Exemple de balises XML pour la définition d'un ruban.

Ce fichier est à compiler à l'aide de l'utilitaire *UICC.exe*

```
UICC.EXE BalisesRuban.xml BaliseRuban.bml  /res:Rubanres.rc
```

Puis il faut inclure le fichier de ressources créé dans le fichier de ressources de l'application.

```
#include "Rubanres.rc"
```

Ensuite, il faut créer une classe C++ qui implémente les interfaces *IUIApplication,* représentant les méthodes de rappels de la plate-forme de ruban, et *IUICommandHandler* pour la gestion des commandes et des évènements.
Enfin, ne pas oublier d'initialiser la plate-forme de ruban à l'aide de l'interface *IUIFramework* et de sa méthode *Initialize()* et de charger les rubans à l'aide de sa méthode *LoadUI()*. L'intégration des rubans dans votre application, est une manière simple et rapide de la "relooker" à moindre frais.
En effet, vous pourrez tout simplement faire correspondre vos anciennes commandes avec les commandes du ruban, sans être obligé de toucher au code originel. Les rubans vous permettent également un meilleur découpage entre la couche présentation et la couche d'implémentation.

Les Graphiques et Animations

Direct2D

Direct2D est une API pour la création de graphiques 2D. Basée sur Direct3D-10, elle offre aux développeurs Win32, une API indépendante de la résolution, et qui utilise la puissance des cartes graphiques de nouvelle génération. Vous pouvez, avec Direct2D, améliorer vos applications GDI existantes, sans être obligé de les réécrire. En effet Direct2D a été conçu pour coopérer avec le GDI et d'autres technologies graphiques.
Pour démarrer avec Direct2D, nous utilisons les interfaces *ID2D1Factory, ID2D1HwndRenderTarget* et *ID2D1Brush*
L'interface *ID2D1Factory* fournit le point départ. C'est une fabrique qui est indépendante des ressources de périphériques que vous utiliserez lors de l'instanciation d'autres ressources Direct2D. Pour créer et instancier une fabrique *ID2D1Factory,* vous utiliserez alors la fonction *D2D1CreateFactory*
Exemple : pour dessiner un simple rectangle, nous créons une fabrique Direct2D

```
CComPtr<ID2D1Factory> fabriqueD2D1;
hr=D2D1CreateFactory  (D2D1_FACTORY_TYPE_SINGLE_THREADED,&fa
briqueD2D1);
```

Puis à partir de cette fabrique, nous choisissons un contexte de rendu cible (ici la cible est l'accélération matérielle, mais vous pourriez choisir un contexte GDI à la place si vous aviez à mixer un nouveau rendu Direct2D et un rendu existant GDI)

```
CComPtr<ID2D1HwndRenderTarget> contexteRenduD2D1;
fabriqueD2D1->CreateHwndRenderTarget (proprietesRendu,proprietes
HwndRendu,&contexteRenduD2D1);
```

Pour dessiner notre rectangle il nous faut une ressource dépendante du périphérique, ici nous créons une ressource de type *ID2D1SolidColorBrush.*

```
CComPtr<ID2D1SolidColorBrush> brosseVerte;
hr=contexteRenduD2D1->CreateSolidColorBrush (D2D1::ColorF(D2D1:
:ColorF::Green),&brosseVerte);
```

Puis nous commençons le rendu

```
contexteRenduD2D1->BeginDraw ();
```

Nous remplissons un rectangle avec la ressource de type Brush

```
D2D1_RECT_F rectangleVert = D2D1::RectF( size.width/4 - 50.
0f,  size.height/2 - 50.0f,size.width/2 + 50.0f,size.height
/2 + 50.0f);
contexteRenduD2D1->FillRectangle (rectangleVert,brosseVerte);
```

Enfin nous indiquons à la plate-forme que le dessin est fini :

```
contexteRenduD2D1->EndDraw ();
```

[Fig.3]

DirectWrite

DirectWrite est une nouvelle API DirectX, qui permet aux applications de supporter un rendu texte de haute qualité indépendante du périphérique et qui améliore la lisibilité des documents. Pour de meilleures performances, elle utilise l'accélération matérielle au travers de Direct2D.

Exemple : pour écrire un simple texte à l'écran, après avoir initialisé une fabrique *ID2D1Factory,* un contexte de rendu cible *ID2D1HwndRenderTarget* et une ressource de type *ID2D1Brush,* nous instancions une fabrique de type *IDWriteFactory*

```
CComPtr<IDWriteFactory> _fabriqueDWrite;
DWriteCreateFactory  (DWRITE_FACTORY_TYPE_SHARED,__uuidof(ID
WriteFactory),
            reinterpret_cast<IUnknown**>(&_fabriqueDWrite));
```

A partir de cette fabrique, nous formatons le texte avec la fonction *CreateTextFormat()* en lui passant en paramètres les informations de typographies (taille, fonte, etc..) Enfin nous utilisons la méthode *DrawTextW()* de notre contexte de rendu cible *(ID2D1HwndRenderTarget)* pour écrire le texte à l'écran. Comme pour le rectangle, il faut entourer l'appel à *DrawtextW,* par les méthodes *BeginDraw()* et *EndDraw().* [Fig.4]

Animation

Sachez qu'avec Windows 7, il est également possible d'ajouter des effets d'animation à son application C++, fournissant ainsi à l'utilisateur des applications plus dynamiques et plus stylisées par rapport à l'application Win32 traditionnelle. Avec la plate-forme d'animation de Windows 7, une grande variété de caractéristiques peuvent être animées, telles que la position, la couleur, le contraste, la taille, la forme et bien d'autres encore.

Une animation débute à partir de l'interface *IUIAnimationManager*.

```
CComPtr<IUIAnimationManager> _animManager;
hr=_animManager.CoCreateInstance (CLSID_UIAnimationManager);
```

Avec ce gestionnaire, nous allons pouvoir créer des variables d'animation *IUIAnimationVariable* qui représentent des aspects à animer, comme la couleur, l'opacité, l'angle de rotation, etc..

```
CComPtr<IUIAnimationVariable> _animVariableRed;
_animManager->CreateAnimationVariable(INITIAL_RED,&_animVaria
bleRed);
```

Ensuite, nous allons planifier les transitions représentées par l'interface *IUIAnimationTransition* associées à nos variables grâce à un StoryBoard représenté par l'interface *IUIAnimationStoryboard.*

```
CComPtr<IUIAnimationStoryboard> pStoryboard = NULL;
    HRESULT hr = _animManager->CreateStoryboard(&pStoryboard);
```

A ce stade, il est possible de définir le type de transition que l'on souhaite, à l'aide de transitions standard définies dans la librarie *IUIAnimationTransitionLibrary.* (Ici nous définissons une transition de type Accéleration et Décélération)

```
CComPtr<IUIAnimationTransitionLibrary> _animTransitionLib;
hr = _animTransitionLib->CreateAccelerateDecelerateTransition
(<code omis>,&pTransitionRed)
```

Puis, nous ajoutons la transition à notre StoryBoard :

```
hr = pStoryboard->AddTransition(_animVariableRed ,pTransitionRed);
```

Ensuite, il faut planifier la transition à jouer

```
CComPtr<IUIAnimationTimer> _animTimer;
UI_ANIMATION_SECONDS secondsNow;
hr = _animTimer ->GetTime(&secondsNow);
hr = pStoryboard->Schedule(secondsNow);
```

En dernier lieu, il faut : Périodiquement mettre à jour l'état de l'animation. (peut se faire dans la procédure de fenêtre au niveau des messages WM_PAINT et WM_DISPLAYCHANGE) :

```
UI_ANIMATION_SECONDS secondsNow;
    HRESULT hr = _animTimer->GetTime(&secondsNow);
 hr=_animManager->Update(secondsNow)
```

Récupérer la ou les valeurs des variables d'animation :

```
DOUBLE red;
    HRESULT hr = _animVariableRed ->GetValue(&red)
```

Puis enfin, mettre à jour l'interface utilisateur avec les nouvelles valeurs des variables de transition.

```
_brosse->SetColor(D2D1::ColorF(static_cast<FLOAT>(red),static_
cast<FLOAT>(green),  static_cast<FLOAT>(blue),1.0));
 _contexteRenduD2D1->FillRectangle(rectPaint,_brosse);
```

Les Bibliothèques

Windows 7 introduit un nouveau concept de *Bibliothèques* en tant que nouveau point d'entrée pour les données des utilisateurs. C'est un lieu où les utilisateurs peuvent rechercher et organiser leurs données (par exemple par date, par type, par auteurs) comme des collections d'éléments qui peuvent être localisées n'importe où, en local ou au travers d'un réseau d'ordinateurs. Ce concept se substitue à la fonctionnalité *"répertoires connus"* (par exemple, Documents, Musique, Vidéo, etc.) des précédentes versions de Windows. Les API disponibles, donnent aux applications un accès simple pour interagir avec [Fig.5]. Une bibliothèque est définie dans Windows 7 par l'Interface *IShellLibrary,*

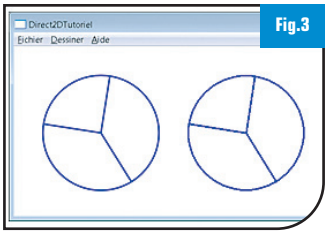


Fig.3

A gauche un tracé Direct2D, à droite un tracé GDI, notez la différence de rendu

Fig.4

```
CComPtr<IShellLibrary> Win7shellLibrary
```

Puis nous appelons sa méthode *SaveInKnownFolder()* pour sauvegarder la bibliothèque dont le nom est défini dans la variable *lpzNomLibrairie.*

```
CComPtr<IShellItem> Win7ShellItem;
hr=Win7shellLibrary->SaveInKnownFolder (FOLDERID_Libraries,lpz
```

```
NomLibrairie,
LSF_OVERRIDEEXISTING,&Win7ShellItem);
```

La particularité de cette méthode, c'est qu'elle permet de sauvegarder les bibliothèques directement dans les répertoires connus. Ici nous spécifions le répertoire *Bibliothèques* à l'aide de la constante *FOLDERID_Libraries.* A partir de là, il est possible d'inclure des répertoires à notre bibliothèque. Un répertoire est défini par l'interface *IShellItem.*

```
CComPtr<IShellItem> shellItem;
```

Que l'on crée directement à partir du chemin du répertoire

```
hr=SHCreateItemFromParsingName  (lpzNomRepertoire,NULL,IID_
IShellItem,(void**)&shellItem);
```

Puis on l'ajoute à notre bibliothèque

```
hr=win7Libraries->AddFolder (shellItem);
```

Enfin, nous validons les modifications en appelant la méthode *Commit()* de l'interface *IShellLibrary*

```
Win7shellLibrary->Commit ();
```

Service Web

Je terminerai cet article par une nouveauté de Windows 7 qui n'est pas forcément la plus spectaculaire, mais qu'il me semblait important d'aborder. Comme je le disais en introduction, nous sommes dans un monde de plus en plus connecté. Pour connecter deux applications entre-elles, à l'heure du "Cloud", rien n'est plus facile que d'utiliser le protocole SOAP au travers de services aussi divers que les services Web ou les services WCF de la plate-forme .NET. Et bien sachez qu'avec Windows 7, nous introduisons un jeu d'API nommé Windows Web Service API (WWSAPI) qui permet aux développeurs C++ de développer des services Web et des clients de services Web, sans pour cela être obligé de passer par une couche d'interopérabilité entre le monde natif et le monde .NET. Pour créer un client Win32 qui soit client d'un service Web, il faut créer un proxy à ce service Web. 2 outils vous permettent d'effectuer cette opération. L'outil *svcutil.exe* qui crée à partir de l'adresse du service Web le fichier de description WSDL et les schémas XSD associés.

```
svcutil /t:metadata http://localhost:10496/Service1.asmx
```

Puis l'outil *wsudil.exe* qui va créer le proxy proprement dit à partir des fichiers WSDL et XSD, sous la forme de deux fichiers de codes ".C" et ".H" que vous pourrez inclure directement dans votre application.

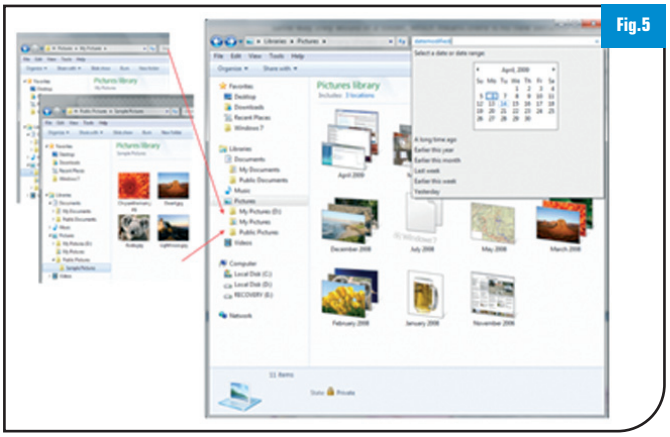


Fig.5

```
wsutil  *.wsdl *.xsd
```

Ensuite, il existe un jeu d'API disponible dans la librairie *WebServices.lib* que nous utiliserons pour instancier le proxy, et exécuter la méthode du service Web.

```
HRESULT hr;
  WS_ERROR* error;
  hr=WsCreateError (NULL,0,&error);
  WS_HEAP* heap;
  hr=WsCreateHeap (100,0,NULL,0,&heap,error);
  WS_SERVICE_PROXY* serviceProxy;
  WS_HTTP_BINDING_TEMPLATE templateValue={};
     hr= Service1Soap12_CreateServiceProxy(
     &templateValue,
     NULL,0,
     &serviceProxy,
     error);
WS_ENDPOINT_ADDRESS address={};
WS_STRING Url=WS_STRING_VALUE(L"http://localhost:20865/Servi
ce1.asmx");
address.url=Url;

hr=WsOpenServiceProxy (serviceProxy ,&address,NULL,error);
//Execute la méthode du service Web
WCHAR* resultat;
hr=Service1Soap_HelloWorld(
     serviceProxy,
     &resultat,
     heap,
     NULL,0,
     NULL,
     error);
WsCloseServiceProxy (serviceProxy,NULL,error);
WsFreeServiceProxy (serviceProxy);
WsFreeHeap (heap);
WsFreeError (error);
```

Conclusion

Comme je vous le disais en introduction, il existe beaucoup de nouvelles fonctionnalités dans Windows 7 que je ne peux couvrir ici. Comme le multi-touche qui permet aux utilisateurs de manipuler une application avec plusieurs doigts simultanément (pour autant qu'il possède un matériel qui le supporte). Comme la plate-forme de capteurs et localisation géopgraphique qui permettrait à une application de s'adapter par rapport à son environnement. Par exemple, lorsque l'on utilise un PC mobile dehors par un jour ensoleillé, l'application doit augmenter, la luminosité, le contraste, et les couleurs de saturation pour améliorer la lecture à l'écran. Une application pourrait fournir des informations spécifiques par rapport à leur localisation géographique, comme les restaurants les plus proches. Développer des applications aujourd'hui est de plus en plus passionnant, il n'y a de limites que celles de notre imagination à la mise en place de nouveaux scénarios pour une meilleure efficacité et expérience utilisateur.

■ Eric **Vernié** - *Microsoft France - Relation technique développeurs*

Introduction à l'assistant de résolution de problèmes de Windows 7 (TroubleShooting Pack)

Lorsque vous achetez un nouveau logiciel ou un nouveau périphérique, ce que vous souhaitez, c'est que cela fonctionne immédiatement, et il n'y a rien de plus frustrant lorsque ce n'est pas le cas.

Le problème est peut-être bénin, mais les heures à essayer de le résoudre peuvent s'enchaîner à un rythme effréné, surtout si vous n'êtes pas à la pointe de la technique (ce que je ne pense pas, puisque vous lisez Programmez !), ou que les utilisateurs de votre logiciel ne sont que des utilisateurs et non pas des informaticiens. Imaginez un système qui pourrait chercher, trouver et réparer automatiquement ce type d'erreurs bénignes ? Imaginez les heures de support et de maintenance gagnées si un tel système existait ?

C'est dans cette voie que s'engage Windows 7 avec la plate-forme de Résolution de problèmes (Windows Troubleshooting Pack).

Prenons un exemple concret.

Je souhaite naviguer avec Internet Explorer sur le site http://www.programmez.com, mais je reçois à la place une belle page d'erreur comme sur la figure suivante : [Fig.1].

Manifestement, j'ai un problème de connexion, mais en tant qu'utilisateur novice, je ne sais pas comment y remédier. Avant Windows 7, j'aurais tout simplement pris mon téléphone et contacté mon service informatique interne. Avec Windows 7, je clique sur le bouton ***Diagnostiquer les problèmes de connexion.***

Le système de résolution des problèmes se met en route [Fig.2].

Et trouve dans un 1er temps que l'accès physique à ma carte Wifi n'est pas possible et que pour résoudre mon problème il suffirait de basculer l'interrupteur pour la rendre active. En le faisant, cela résout le problème de connexion, mais je décide pour notre démonstration, d'ignorer cette étape et de continuer à essayer de résoudre le problème.

A cette étape, le système de résolution a trouvé que ma connexion réseau Ethernet n'est pas active, mais qu'il a besoin de privilèges administrateur pour tenter de le résoudre. (Je dis bien tenter car il n'y a en informatique pas de garantie à 100%) [Fig.3].

Je clique alors sur ***Essayez ces réparations en tant qu'administrateur.***

La vérification et la résolution automatique des problèmes démarrent [Fig.4].

Si tout se passe correctement le système de résolution des problèmes affiche le statut suivant : [Fig.5]

Il a détecté, mais pas corrigé, que la carte Wi-Fi n'est pas activée, et a résolu automatiquement le problème de connexion réseau.

En fermant, le système de résolution des problèmes, Internet Explorer peut alors atteindre cette fois-ci le site www.programmez.com

Si vous êtes curieux de nature et que vous possédez Windows 7, je ne peux que vous encourager à aller voir le centre de résolution des problèmes.

1. Activez le panneau de configuration
2. Dans la boîte de recherche en Haut à droite tapez **résolution**
3. Dans la liste des résultats choisissez Résolution des problèmes [Fig.6].
4. Vous aurez alors accès à différents Assistants de résolution de problèmes. [Fig.7]

Fig.1

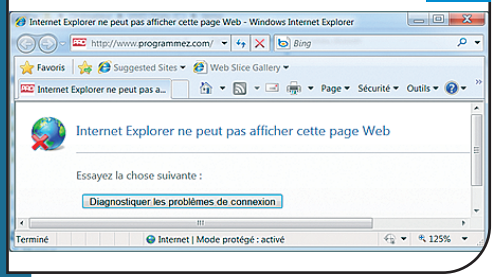


Fig.3

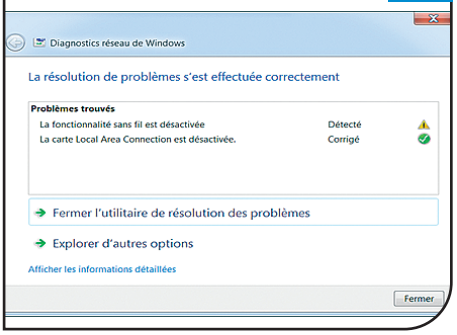


Fig.5

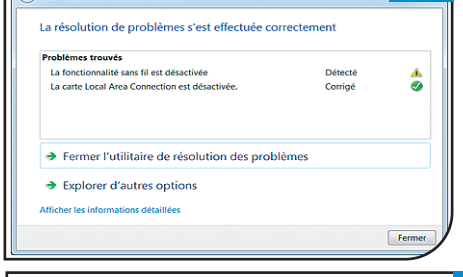


Fig.2

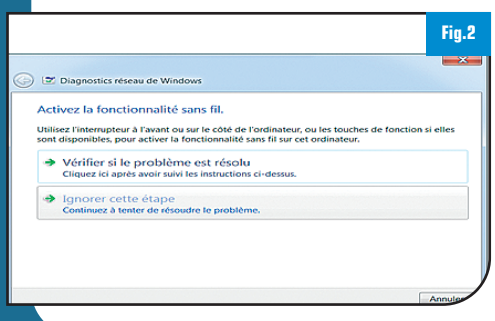


Fig.4

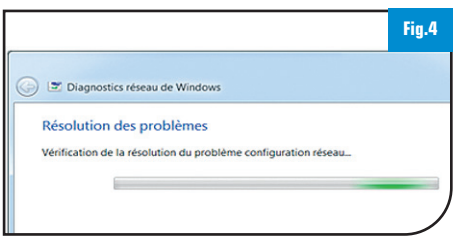
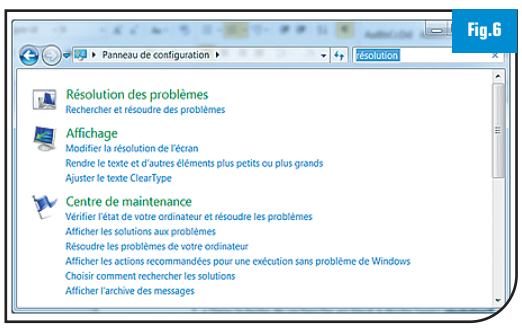


Fig.6

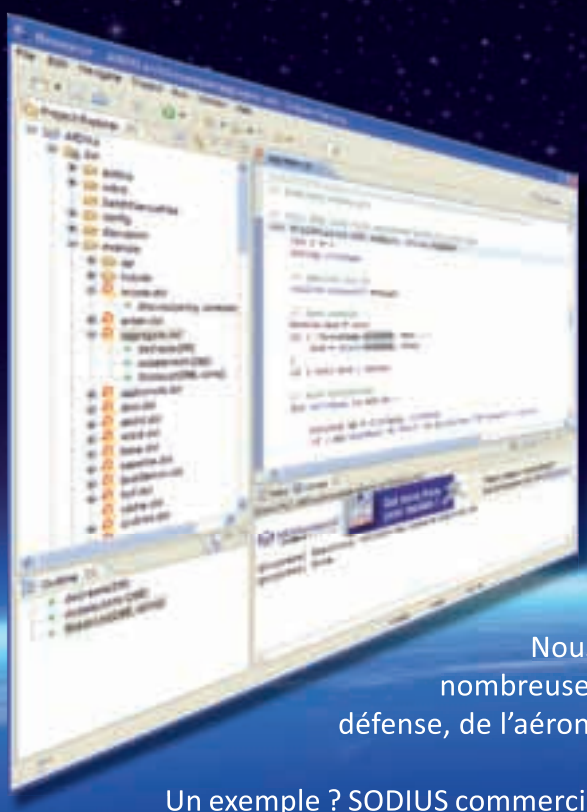


De nouveaux horizons pour vos développements

Il n'y a pas que l'Open Source pour réussir...

Vous avez développé un framework* logiciel performant (technologie ou métier) ?
Ce framework a été éprouvé par des clients professionnels exigeants ?
Vous avez identifié un marché transversal ou de niche ?
Vous souhaitez passer à la vitesse supérieure ?

Conjuguez vos talents avec SODIUS pour assurer l'industrialisation du framework et une mise sur le marché efficace !



SODIUS, Technology Partner de Telelogic et IBM depuis 7 ans, a développé un savoir-faire reconnu dans le domaine du génie logiciel, des ateliers de conception système et de l'interopérabilité sous Eclipse et Jazz.

Maintenant Business Partner de IBM Rational, SODIUS met a votre disposition son expertise pour la diffusion de solutions professionnelles sur des marchés exigeants.

Nous fournissons des produits OEM à de nombreuses entreprises dans les domaines de la défense, de l'aéronautique, de l'automobile, du tertiaire, ...

Un exemple ? SODIUS commercialise MDWorkbench for DOORS et DXL Editor (www.dxleditor.com), des framework pour l'application DOORS de IBM Rational.

Si vous avez un framework validé en main, nous proposons aux meilleurs une alliance pour profiter de notre savoir-faire et de notre réseau de distribution, et ouvrir de nouveaux horizons.

contactez nous !

Gérard Bouvet - Responsable des Partenariats
gbouvet@sodius.com

Sodius SAS
6 rue de la Cornouaille
44300 Nantes
Tél. : 02.28.23.60.60
www.sodius.com

*framework : atelier de développement.

Tout ce que je viens de vous montrer se trouve en standard dans Windows 7 pour des fonctionnalités de base liées au système d'exploitation. Une question se pose alors :

En tant que développeur de logiciel puis-je fournir ce type de fonctionnalité avec mes logiciels ?

La réponse est bien évidement oui, et surtout, Microsoft le préconise comme étant une bonne pratique, source d'une nouvelle efficacité. Je ne vais pas entrer trop dans le détail, mais voici ce qu'il faut retenir :

Un assistant de résolution de problèmes (*Troubleshooting Pack* en anglais) est constitué :

- D'un manifeste de métadonnées, sous forme de fichier XML, décrivant de quoi est constitué l'assistant
- D'un script de détection des problèmes
- D'un script de résolution des problèmes
- D'un script de vérification des problèmes
- Et optionnellement de ressources localisées dans la langue de votre choix.

Il est possible de créer le fichier manifeste XML avec l'outil **TSPDesigner.exe** que l'on trouve dans le Kit de développement Windows 7.

- La première chose à faire est de créer un nouveau projet, ici nous créons un assistant de dépannage pour une application WPF qui se connecte à une base de données SQL Serveur. Cet assistant nous permettra de résoudre les problèmes de connexion au Serveur SQL.
  Certain champs sont requis permettant d'identifier l'assistant. Le nom du projet, ainsi qu'une URL [Fig.8].
- Un problème est défini par sa cause (Root Cause). Ici, notre cause est la non disponibilité du Service SQL Server. Notez l'identification de la cause **DetectServiceSQL,** que nous réutiliserons par la suite [Fig.9].
- Nous allons définir le dépanneur (troubleshooter) qui aura pour tâche de lancer un script PowerShell qui devra detecter si le service SQL Server est démarré ou pas.
  A noter qu'il est possible d'indiquer au dépanneur de s'exécuter avec des privilèges administrateur, et que nous pouvons avoir une interaction avec l'utilisateur. C'est-à-dire qu'il est possible à ce stade de demander à l'utilisateur, soit de répondre à des questions avec une liste de choix, soit de lui demander des informations complémentaires, soit de lui afficher un message statique l'informant de l'état du dépanneur [Fig.10].
- Puis, nous allons configurer le résolveur indiquant dans notre exemple qu'il va tenter de démarrer le service SQL Serveur.
  Le démarrage du service , se fera à l'aide d'un Script PowerShell que nous définirons ensuite. A noter que nous indiquons d'exécuter le resolveur avec des privilèges administrateur, car le démarrage du service SQL Serveur ne peut se faire qu'avec des droits elevés et que nous avons également une interaction avec l'utilisateur qui pourra relancer l'application le cas écheant [Fig.11].
- Puis nous informons avec le vérificateur le système de dépannage, qu'il doit relancer le dépanneur pour vérifier que la cause du problème est résolue. [Fig.12].
- Enfin, l'élément **scripts** permet de déterminer quels scripts seront exécutés pour le dépanneur et le Résolveur de problèmes [Fig.12].
- Une fois nos métadonnées, configurées, il faut créer les scripts PowerShell de détection. En cliquant soit sur *Edit TroubleShooter Script et Edit Resover Script,* cela lance l'interface de programmation de scripts PowerShell, **Windows PowerShell ISE,** comme sur la figure suivante : [Fig.14 et 15]

C'est à vous de jouer maintenant et d'utiliser la puissance de PowerShell pour créer vos propres Scripts. Néanmoins ce qu'il faut savoir, c'est qu'il existe certaines *CmdLet* PowerShell qui permettent de dialoguer directement avec le système de résolution des problèmes comme ceux que nous utilisons ici.

Write-DiagProcess qui permet d'écrire un message dans la fenêtre du résolveur, et **update-diagrootcause,** qui lui indique si le problème est détecté.

- Ici nous testons avec une commande WMI si le service SQLExpress est lancé.

Fig.7

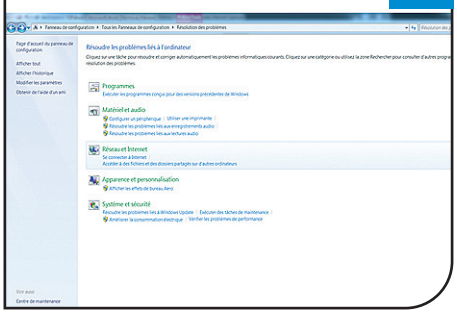


Fig.9

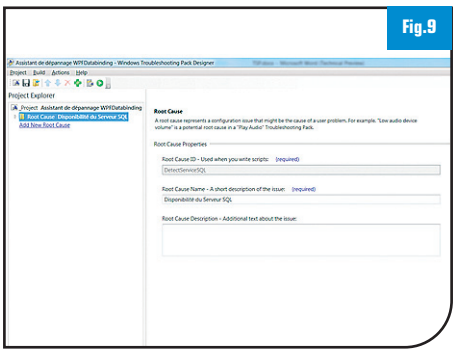


Fig.11

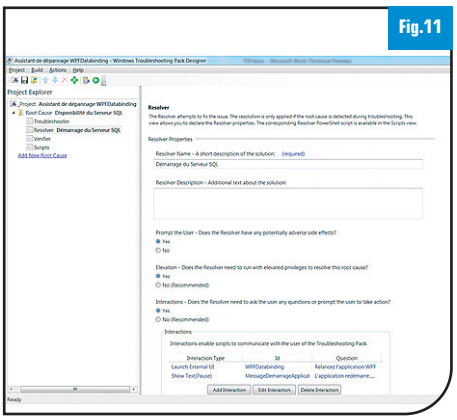


Fig.8

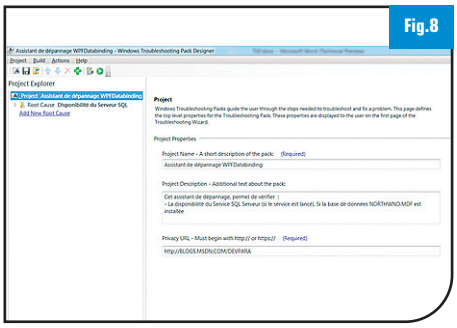


Fig.10

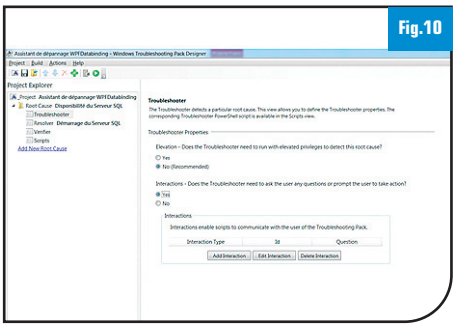
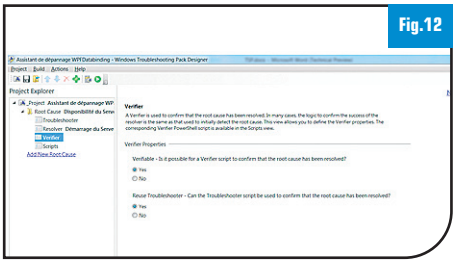


Fig.12



```
$SQlServeur=get-wmiobject win32_service | where-object {$_.Name
-like "*MSSQL*SQLEXPRESS*"}
```

- S'il n'est pas lancé, la cause du problème est détectée, donc on en informe le système à l'aide de la CmdLet :

```
update-diagrootcause -id $RootCauseId -detected $RootCauseDetected
```

A noter ici que cette CmdLet, a comme identifiant **DetectServiceSql,** le même identifiant que nous avons indiqué lors de la création de la cause du problème. La CmdLet **Get-DiagInput** qui permet de récupérer les entrées utilisateurs.

- Une fois les scripts définis, on va compiler et tester notre assistant à l'aide de l'outil TSPDesigner.exe [Fig.16].

Remarque

En mode débogage de scripts (par opposition à un déploiement en production), il faut permettre au système d'exécuter des scripts non signés. Pour cela, on ouvre une session PowerShell en mode administrateur et on exécute les commandes suivantes :

```
set-executionPolicy RemoteSigned
```

puis le script PowerShell suivant :

```
C:\Program Files\Microsoft SDKs\Windows\v7.0\Bin\TSPDesigner
\TestModeSetup.ps1
```

- Exécution de l'assistant de résolution de problèmes
  La compilation de l'assistant crée un répertoire *\output,* avec les scripts PowerShell associés, un fichier ayant pour extension *.DIAGPKG* et un fichier compressé de type cabinet *.DIAGCAB*. Il est alors possible pour lancer l'assistant de double-cliquer sur fichier *.DIAGPKG* ou *.DIAGCAB,* ou alors d'exécuter la commande Run de l'outil TSPDesigner.exe.
  Mais une fois que votre assistant est mis au point, l'une des possibilités est de l'activer directement dans votre application.

- Dans notre exemple, nous avons une application WPF qui se connecte à une base de données. Lors du chargement de l'application, si aucune connexion au serveur SQL n'est disponible, alors on invoque directement notre assistant de résolution de problèmes en utilisant le processus MSDT.EXE comme suit :

```
msdt /path <chemin d'accès au fichier ayant l'extension .diagpkg
```

ou alors

```
msdt /cab <chemin d'accès au fichier ayant l'extension .diagcab>
```

[Fig.17].

- En cliquant sur **Oui,** notre assistant démarre [Fig.18].
- Puis en cliquant sur Suivant, l'assistant détecte que le service SQL n'est pas démarré, et propose alors de résoudre automatiquement le problème [Fig.19].
- En appliquant la correction, il est alors possible d'interagir (optionnel) avec l'utilisateur en lui proposant de charger de nouveau l'application directement à partir de l'assistant [Fig.20].
- Enfin, l'assistant affiche un état de la résolution, c'est le vérificateur qui est alors exécuté [Fig.21]. Bien évidemment, il est possible d'enchaîner dans un seul assistant, plusieurs recherches et résolutions de problèmes.

CONCLUSION

L'assistant de résolution de problèmes dans Windows 7, n'est pas la fonctionnalité la plus en vue, mais elle permettra sans aucun doute si vous l'adoptez, d'ajouter de la valeur à vos applications, en proposant à l'utilisateur, une démarche simple et efficace de résolution de problèmes courants.
Non seulement la perception de l'utilisateur changera, mais vous gagnerez également en temps et en efficacité.

■ Eric **Vernié**
Microsoft France
Division DPE

Fig.13

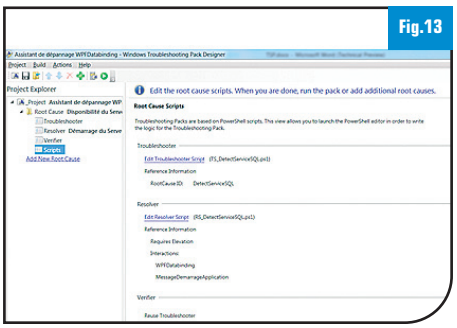
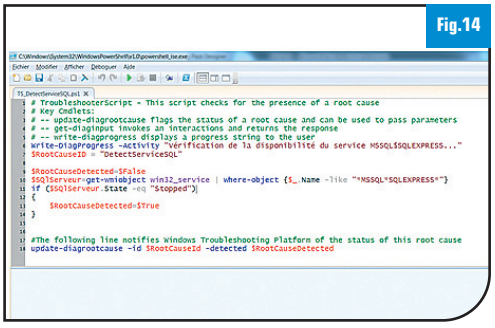
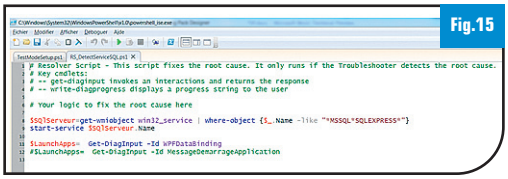


Fig.14



Exemple de dépanneur :
Détecte si le serveur SQL est lancé

Fig.15



Exemple de résolveur : Démarre le service SQLExpress

Fig.17

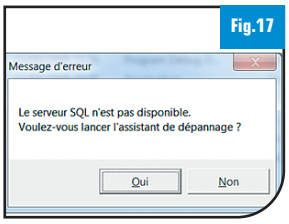


Fig.16

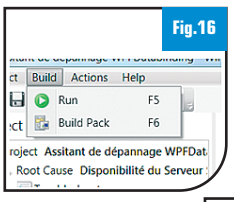


Fig.18

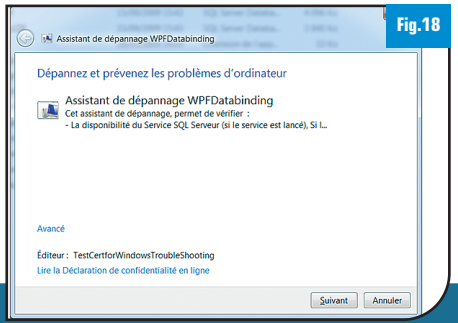


Fig.19

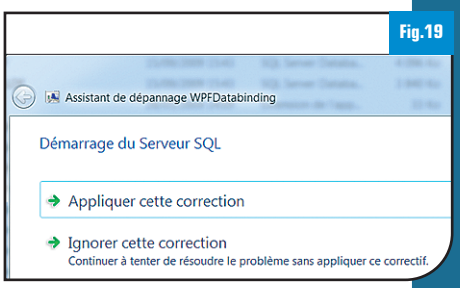


Fig.20

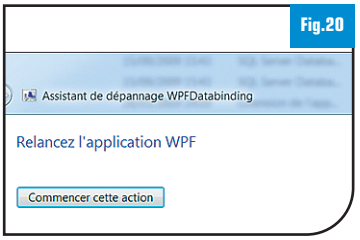
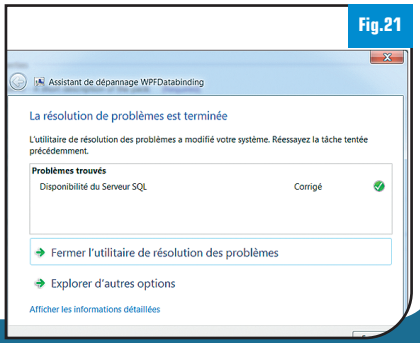


Fig.21



La flexibilité du VHDBoot

Une des grosses nouveautés de Windows 7 pour l'administrateur système, en termes de déploiement et de maintenance, est la capacité de démarrer son ordinateur ou celui d'un des utilisateurs à partir d'une image virtuelle VHD (Virtual Hard Disk). Bien évidemment cette capacité est très intéressante pour un développeur et c'est ce que nous allons examiner dans cet article.

Le « VHD Boot » est donc la capacité de démarrer à partir d'une image virtuelle Windows 7 ou Windows Server 2008 R2. Il y a différents scénarios qui peuvent être mis en place. Le premier peut être l'installation d'un Windows 7 ou 2008 R2 sur votre PC, directement en mode VHD Boot. Le deuxième scénario peut être la création d'une machine virtuelle Windows 7 ou 2008 R2 à partir d'un Hyper-V ou Virtual PC, de placer ce fichier VHD sur votre ordinateur et de modifier le secteur de démarrage afin de commencer sur la nouvelle image virtuelle VHD. Cela signifie que vous pourriez déjà avoir un système d'exploitation installé.

Il y a quelques inconvénients à utiliser cette méthode. Les performances vont diminuer d'environ 3%, l'hibernation ne fonctionnera pas, et BitLocker pourra bien évidemment être activé à l'intérieur du fichier VHD mais pas directement sur le disque. Le dernier désavantage est que l'index de Windows Expérience ne fonctionnera pas. Ces inconvénients ne sont pas si importants que cela lorsqu'un développeur apprendra qu'il peut démarrer sur plusieurs systèmes et exploiter WPF et la pleine puissance de traitement d'un GPU . Ce n'est pas tout car lorsque nous parlons de VHD, cela signifie que nous pouvons exploiter toutes les fonctionnalités d'un VHD, comme les disques différentiels.

Mise en place du VHD Boot

Nous allons commencer doucement par l'installation d'un Windows 7 sur un ordinateur équipé d'un disque qui est totalement vierge. Munissez-vous de votre DVD de Windows 7 puis démarrez dessus pour commencer l'installation du système d'exploitation.

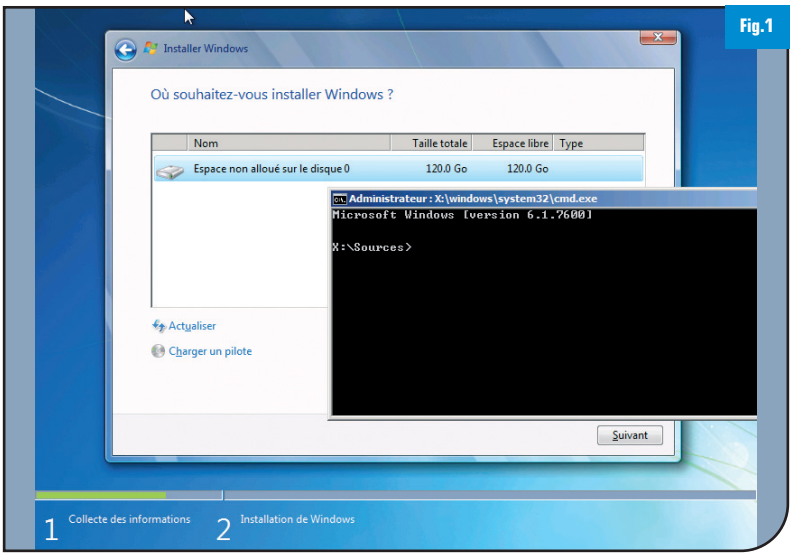


Fig.1

Il faut changer la séquence de démarrage dans le BIOS si vous ne démarrez pas directement sur votre DVD. Lorsque vous êtes sur l'écran « **Quel type d'installation voulez-vous effectuer ?** », cliquez sur « **Personnalisée (option avancée)** ».

Cette fenêtre va nous permettre de bien comprendre le fonctionnement du système VHD Boot par rapport à notre disque physique. Nous avons ici un ordinateur équipé d'un disque dur de 120 Go. Il s'agit donc de la taille de mon disque dur physique. Il n'y a pas de partition sur mon disque pour l'instant. Pressez les touches **Shift** et **F10** afin de lancer une invite de commandes. [Fig.1]

Maintenant, nous allons créer un fichier VHD sur lequel nous effectuerons l'installation de Windows 7. Dans l'invite de commandes, tapez **DISKPART** afin de lancer l'utilitaire de disques. Voici maintenant la liste des commandes qu'il faut exécuter :

- **LIST DISK** pour lister les disques présents. Nous retrouvons notre disque de 120 Go.
- **SELECT DISK 0** pour sélectionner le disque 0, soit le seul disque présent.
- **CREATE PARTITION PRIMARY** pour créer une partition primaire.
- **FORMAT QUICK FS=NTFS** pour formater le disque rapidement et en NTFS.
- **LIST VOL** afin de lister les volumes présents sur le disque. Nous pouvons voir notre volume de 120 Go et la lettre qui lui est attribuée, soit la lettre C dans notre cas.
- **CREATE VDISK FILE=C:\Windows 7RTM.vhd MAXIMUM=40000 TYPE=EXPANDABLE** pour créer notre fichier VHD sur lequel on va installer Windows 7. La variable MAXIMUM permet de spécifier la taille **virtuelle** maximum du fichier VHD et cela est très important car lorsque vous faites du VHD Boot il va « bufferiser » cette taille en mémoire. Si celle-ci est supérieure à la capacité physique (et disponible) de votre disque, vous aurez alors un bel écran bleu lors du démarrage. Enfin la variable **TYPE** permet de définir si la taille occupée sur le disque physique est tout de suite définie sur celui-ci (**FIXED**), soit 40 Go ici, ou si la taille sera occupée au fur et à

mesure (**EXPANDABLE**) jusqu'à la taille maximum définie.

- **SELECT VDISK FILE=C:\Windows 7RTM.vhd** permet de sélectionner le fichier VHD que nous venons de créer dans le gestionnaire de démarrage.
- **ATTACH VDISK** attache le fichier VHD dans le gestionnaire de démarrage et permet alors d'effectuer des opérations à l'intérieur du fichier VHD.
- **CREATE PARTITION PRIMARY** crée alors la partition principale dans notre fichier VHD.
- **FORMAT QUICK FS=NTFS** pour formater le disque (du fichier VHD).
- **ASSIGN LETTER=V:** définit une lettre de lecteur à notre fichier VHD.
- **LIST VOL** pour lister encore une fois les volumes présents sur le disque. Nous voyons maintenant le volume de 40 Go en plus de celui-ci de 120 Go.
- **LIST VDISK** afin de vous montrer que nous pouvons lister les disques VHD dans l'utilitaire de démarrage DISKPART.
- **EXIT** pour quitter l'utilitaire DISKPART.

Nous avons terminé de configurer nos partitions, nous pouvons maintenant actualiser la fenêtre où elles sont listées. Cliquez alors sur la nouvelle partition de 40 Go qui vient d'apparaître. [Fig.2]

Il est écrit « **Impossible d'installer Windows sur le disque 1 Partition 1. (Afficher les détails)** ». Ce n'est pas important car nous pouvons malgré tout cliquer sur Suivant et continuer l'installation de Windows 7 sur ce disque, soit continuer l'installation sur un fichier VHD. Il est bien évidemment possible d'installer un autre système, physiquement ou dans un fichier VHD, et de mettre en place du multi-boot. Il peut être très pratique de faire cela avec un Windows Server 2008 R2. Vous pourrez alors utiliser Hyper-V et utiliser votre fichier VHD Windows 7 ! N'oubliez pas qu'il faut faire attention de ne pas dépasser la taille physique de votre disque sinon vous aurez un écran bleu lors du démarrage.

Et si nous avions déjà un fichier VHD hébergeant un Windows 7 ou 2008 R2 créé avec Hyper-V par exemple ? Aucun souci ! Il faut une fois de plus démarrer sur le DVD de Windows 7 ou 2008 R2 et lancer une invite de commandes, soit en pressant les touches **SHIFT** et **F10**, soit en allant dans le menu « Réparer l'ordinateur » et en lançant une invite de commandes. N'oubliez pas que si vous déployez un VHD sur du matériel différent, le **SYSPREP** est essentiel !

Nous devons préparer notre disque dur, c'est-à-dire créer une partition primaire et formater le disque. Ensuite nous devons copier le fichier VHD sur le disque physique de la machine. Il n'est pas possible de faire un VHD Boot à partir d'un disque dur USB ou équivalent. Pour cela utilisez un simple « **copy X:\MonFichier.VHD C:** », sachant que **X:** correspond à votre disque dur USB et que **C:** correspond à notre partition que nous venons de créer. Il s'agit des commandes que nous avons décrites précédemment. Une fois que le fichier est créé, il nous faut exécuter différentes commandes :

- **DISKPART** afin de lancer l'utilitaire de disques.
- **SELECT VDISK FILE=C:\Windows 7RTM.vhd**
- **ATTACH VDISK**
- **EXIT**

A ce moment précis, nous avons notre fichier VHD qui est interprété par notre utilitaire de démarrage tel une partition, c'est-à-dire que nous pouvons lister le contenu du VHD. Ce qu'il faut comprendre c'est que nous avons physiquement un disque sur lequel un fichier VHD est placé. Lors du démarrage de notre ordinateur il ne sera pas capable de comprendre qu'il faut démarrer sur le fichier VHD car les fichiers de démarrage requis sont actuellement dans le fichier VHD

```
DISKPART> list vol

  N° volume   Ltr  Nom          Fs     Type        Taille   Statut     Info
  ----------  ---  -----------  -----  ----------  -------  ---------  --------
  Volume 0     D   GRMCULFREO_  UDF    DVD-ROM     2334 M   Sain
  Volume 1     C                NTFS   Partition    119 G   Sain
* Volume 2     V                NTFS   Partition     39 G   Sain

DISKPART> list vdisk

  VDisk ###  Disque ###  État                  Type        Fichier
  ---------  ----------  --------------------  ----------  -------
* VDisk 0    Disque 1    Ajouté                Extensible  c:\Windows7RTM.vhd
```

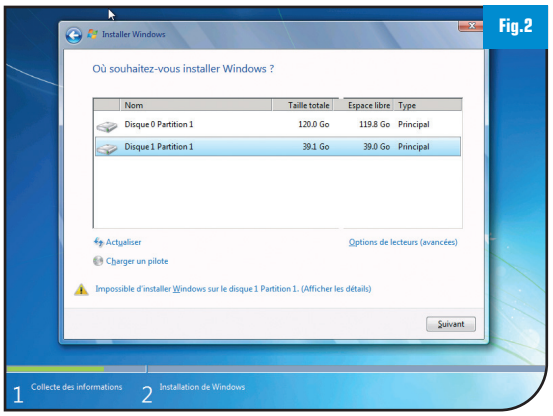


Fig.2

et non directement sur le disque physique. Il nous faut alors ajouter les bonnes entrées dans le gestionnaire de démarrage pour qu'il puisse démarrer sur le fichier VHD. Avant, nous étions obligés d'utiliser l'outil **BCDEdit,** ce qui était parfois compliqué. Vous pouvez d'ailleurs voir sur Internet que beaucoup de personnes expliquent comment modifier son secteur de démarrage en utilisant l'outil **BCDEdit.** Depuis la sortie finale de Windows 7, l'outil **BCDBoot** a vu le jour et c'est lui qui va nous rendre ce service. Cet outil est placé dans le dossier F:\Windows\System32\, F: étant la lettre attribuée à notre fichier VHD. Il faut donc s'y rendre par la commande « **cd F:\Windows\System32** ». Lorsque vous y êtes, tapez : « **BCDBoot f:\Windows /s C:** ». Cette commande copie les fichiers de démarrage et crée un nouveau magasin BCD (Boot Configuration Data) vers la partition système (soit notre disque physique dans le cas d'un VHD Boot). Il suffit maintenant de redémarrer pour utiliser son système, enjoy !

Conclusion

Le VHD Boot est une sacrée avancée pour les IT comme pour les développeurs. Il faut s'approprier la technologie afin d'en tirer parti au maximum. Ce qu'il ne faut pas oublier, c'est qu'il est possible d'utiliser un système de snapshots en créant des disques différentiels, c'est-à-dire basés sur la configuration d'un disque parent (**create vdisk file=C:\Windows7Parent01.vhd parent =C:\Windows7RTM.vhd**). Vous pourriez alors tester différentes versions de Visual Studio 2010 ou de vos applications sans impacter votre système !

■ William **Bories**

Internet Explorer 8 pour les développeurs

Au delà des nouveaux aspects les plus visibles comme les accélérateurs et les Web slices, Internet Explorer 8 fournit aux développeurs des fonctionnalités moins apparentes mais dont ils peuvent tirer parti afin d'améliorer les performances de leur site.

Tout d'abord, Internet Explorer 8 implémente les recommandations JSON de l'ECMAScript 3.1 à savoir le support des prototypes de DOM modifiables (mutable DOM prototypes) et la prise en charge native des API de sélecteurs, des fonctionnalités de pointe qui serviront de base pour la prochaine génération de sites Web ainsi que les URI de données (Data URI). Ensuite, c'est le développement d'applications AJAX qui est facilité à travers l'implémentation des spécifications HTML5 de stockage dans le DOM (DOM Storage), des évènements de connexion (Connectivity Events), des requêtes inter-documents (Cross-Document Request ou XDR) et de la messagerie inter-documents (Cross-Document Messaging ou XDM).

Enfin, n'oublions pas le profiler JScript intégré nativement aux autres outils de développement dans Internet Explorer 8 et qui permet aux développeurs de mieux analyser le fonctionnement de leur site et d'en améliorer les performances.

Prise en charge native de JSON

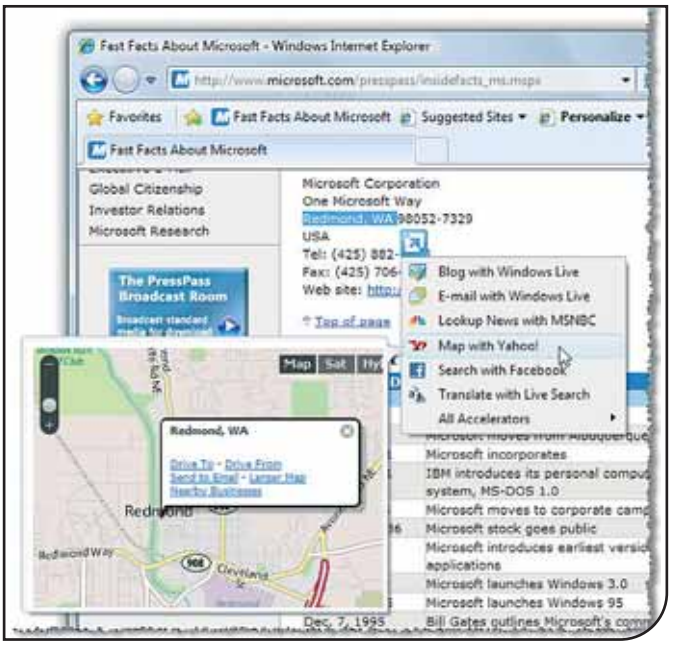
JSON (JavaScript Object Notation) est un format d'échange de données léger basé sur un sous-ensemble de la notation objet du langage JScript et est souvent utilisé pour transmettre des données entre les composants des sites Web AJAX. Malheureusement, de nombreux sites utilisent JSON au détriment de la sécurité et s'appuient sur la méthode **eval** JavaScript pour "réincarner" les chaînes JSON dans des objets JavaScript.

Les développeurs soucieux de sécurité utilisent plutôt un analyseur JSON plus sécurisé pour s'assurer que l'objet JSON ne contient pas un script exécutable, souvent au prix d'un impact significatif sur les performances.

Le moteur JScript, fourni avec Internet Explorer 8, implémente la proposition JSON du ECMAScript 3.1. La méthode JSON.**stringify** accepte un objet de script et renvoie une chaîne JSON, tandis que la méthode JSON.**parse** accepte une chaîne et en toute sécurité la transforme en un objet JavaScript. Cet ajout, en plus d'être sécurisé est également beaucoup plus rapide que d'autres implémentations.

Pour illustrer ce propos, voici un exemple de sérialisation et désérialisation des données à l'aide des méthodes JSON.**stringify** et JSON.**parse** :

```
var testObj = new Object();
testObj.numVal = 4;
testObj.strVal = "fool";
testObj.dateVal = new Date(2008, 3, 1);
testObj.arrVal = new Array(1,2,"test");
testObj.boolVal = true;
```




```
// Conversion d'objet en chaîne
var stringifiedValue = JSON.stringify(testObj) ;

// Création d'un nouvel objet à partir de la chaîne
var parsedValue = JSON.parse(stringifiedValue) ;
```

Recherche simplifiée dans le DOM

Les développeurs se trouvent souvent pris au piège d'affecter un identifiant à chaque élément dans leur code source afin de pouvoir utiliser l'incontournable API **getElementById** pour récupérer facilement les informations dont ils ont besoin. Peut-être êtes-vous, vous-même, un utilisateur chevronné de **getElementsByTagName** ? Quelle que soit la technique de recherche d'élément que vous employez, vous avez probablement été confronté à des scénarios où l'utilisation d'une de ces API ne convenait pas à vos besoins.

Quel que soit le scénario, vous avez pu être amené à vous tourner vers des frameworks implémentant une meilleure API de recherche d'éléments (prototype.js et jQuery, par exemple).

Avec Internet Explorer 8, il est possible maintenant d'utiliser les API **querySelector** et **querySelectorAll** qui devraient probablement devenir les API les plus fréquemment utilisées dans votre répertoire. Elles dérivent directement de la prise en charge du support amélioré des sélecteurs CSS 2.1 afin que vous puissiez tirer parti du moteur de mise en correspondance de sélecteur à tout moment depuis JavaScript. Pour expérimenter la syntaxe,

essayez de remplacer un appel à **getElementById**('id') avec **querySelector**('#id'): le résultat est le même. L'exemple suivant utilise **querySelectorAll** pour récupérer tous les éléments pertinents par classe et désactiver leur statut d'affichage collectif. Par comparaison, le Listing 1 en fin de cette partie illustre le code original non optimisé, démontrant que l'utilisation de l'API de sélecteurs nécessite beaucoup moins de code à la fois au téléchargement et pour l'exécution sur le client.

```
function useSelectors(clsName)
{
   // Tout masquer...
   var unset =
     document.querySelectorAll("h3, p, ul, code");

   for (var i = 0; i < unset.length; i++)
     unset[i].style.display = 'none';

   var set =
     document.querySelectorAll("." + clsName);

   for (var i = 0; i < set.length; i++)
      set[i].style.display = 'block';
}
```

Les API de sélecteurs sont utilisables à deux niveaux différents : document et élément. Si vous voulez trouver uniquement le premier résultat pour un sélecteur particulier, utilisez **querySelector** (qui renvoie l'élément trouvé directement). Si par contre vous souhaitez trouver tous les éléments correspondants, vous pouvez utiliser **querySelectorAll**. L'utilisation du niveau élément peut être importante lorsque vous souhaitez limiter vos recherches à un sous-répertoire particulier du DOM. Il est à noter que le paramètre de sélecteur chaîne fournie est comparé dans un premier temps à l'ensemble du document, mais les éléments retournés sont limités à la sous-arborescence appropriée fournie dans l'appel à l'API.

Les développeurs de WebKit ont écrit une série de tests qui vous permet de comparer la mise en œuvre native de l'API de sélecteurs à des implémentations classiques en Javascript démontrant le fait que vous pouvez exploiter cette mise en œuvre native afin d'accélérer considérablement votre code. Vous pouvez aller sur http://webkit.org/perf/slickspeed/ dans Internet Explorer 8 pour le vérifier par vous-même !

Prototypes du DOM modifiables (Mutable DOM prototypes)

Comme avec la prise en charge des API de sélecteurs, la prise en charge des prototypes de DOM modifiables offre aux développeurs les bases pour créer des abstractions compactes et efficaces.

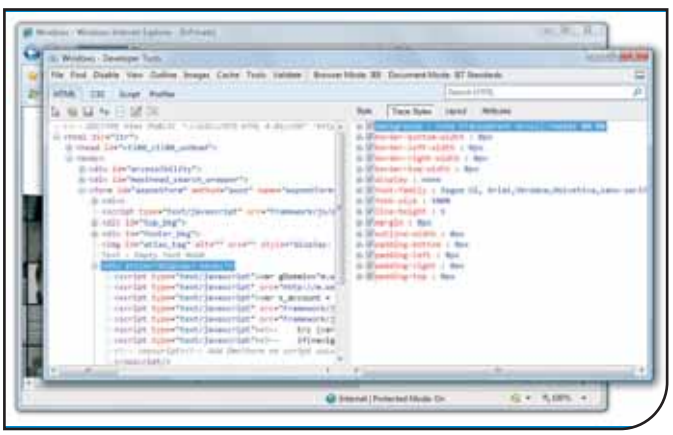
Comme de nombreux langages de programmation, les objets JavaScript dérivent de types bien définis et documentés qui peuvent être découverts par le biais de réflexion. Contrairement à de nombreux langages de programmation, toutefois, JavaScript rend ces types directement accessibles et modifiables par l'intermédiaire du support de l'héritage de prototype.

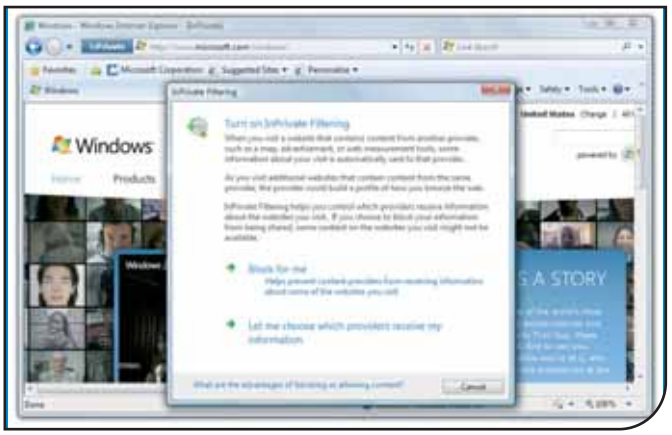
Pour illustrer ceci, prenons cet exemple :

```
var arrayInstance = new Array();

// Vérifions si il y a un "xyz"
// (cela ne devrait pas être le cas car il ne fait pas partie
// de l'objet ou de son prototype)
alert(arrayInstance.xyz);

// Ajoutons xyz au prototype
Array.prototype.xyz = "xyz string";

// Maintenant xyz est disponible
alert(arrayInstance.xyz);
```

Même avec cet exemple simplifié, vous pouvez voir que si xyz est un attribut qui améliore tous les éléments du tableau, alors son inclusion dans le prototype du tableau est la manière d'obtenir le meilleur impact par ligne de code. Suite aux commentaires de la communauté, Internet Explorer 8 apporte la puissance et la flexibilité des prototypes Javascript au DOM d'Internet Explorer. Vous vous demandez peut-être ce que cet apport peut bien avoir d'excitant mais j'espère pouvoir vous faire changer d'avis.

Par rapport aux frameworks fournis par les développeurs Web, les fonctionnalités fournies par un navigateur le sont à un rythme plus lent et répondent à un public plus général. Les développeurs Web ont besoin d'un moyen d'étendre le DOM avec de nouvelles fonctionnalités aussi rapidement que leurs besoins métiers l'exigent. Essentiellement, les développeurs doivent être en mesure de combler les lacunes d'un modèle bien avant qu'une implémentation native ne soit fournie dans n'importe quel navigateur. Les prototypes DOM fournissent aux développeurs Web les briques de base





qui leur permettent de combler les lacunes du DOM directement ou en remplaçant des implémentations existantes.
Si vous souhaitez modifier tous les éléments image dans le DOM à partir de l'exemple précédent, vous pourriez l'appliquer directement au prototype **HTMLImageElement** :

```
// Appliquer le changement au prototype.
HTMLImageElement.prototype.grow = function()
{
   this.width *= 1.5;
   this.height *= 1.5;
}

// Agrandir toutes les images dans le document.
for (var i= 0; i < document.images.length; i++)
{
   document.images[i].grow();
}
```

La méthode alternative consisterait à ajouter l'API manuellement à chaque élément image individuellement. Vous pouvez voir en quoi les prototypes modifiables offrent une approche beaucoup plus simple et efficace à ce problème classique.
En plus de fournir un socle de base, les prototypes DOM permettent de rendre JavaScript plus efficace en fournissant un endroit unique et logique pour des fonctions d'amélioration de l'élément.

Dans de nombreux scénarios, le prototype de l'élément est l'emplacement le plus raisonnable pour ces ajouts car c'est le parent de tous les éléments dans le DOM Internet Explorer. Par conséquent, les modifications apportées au prototype de l'élément s'appliquent à tous les éléments. Un bon exemple de la manière dont vous pouvez utiliser cette approche consiste à ajouter l'API **getElementsByClassName** à tous les éléments : les spécifications provisoires du projet HTML 5 ont défini récemment cette API et Internet Explorer 8 ne l'implémente pas encore nativement. Vous pouvez définir cette méthode pour traiter immédiatement les éléments comme s'ils le supportaient nativement. En plus, cette méthode devrait être plus rapide que les alternatives plus complexes offertes dans le cadre de frameworks génériques. Vous pouvez vous référer au **Listing 2** pour l'implémentation de _MSHTML5_getElementsByClassName_Elem.

```
<div id="div1" class="xyz">Text
  <div id="div2" class="abc xyz">More</div>
</div>
<div id="div3" class="xyz abc box">Other</div>

<script type="text/javascript">
   Element.prototype.getElementsByClassName =
         _MSHTML5_getElementsByClassName_Elem;

   // Invocation de getElementsByClassName.
   var div1 = document.getElementById('div1');
   div1.getElementsByClassName('abc xyz');
</script>
```

URI de données (Data URI)

Vous avez probablement été amené à créer un site Web nécessitant un certain nombre de petites images. Dans certains cas, vous pouvez utiliser les sprites CSS pour combiner des images et éviter des aller-retour réseau mais, dans d'autres situations, vous avez dû vous résigner à l'inévitable et supporter le coût d'un aller-retour supplémentaire.
Normalement, les URL font office de pointeurs vers des emplacements réseau qui contiennent des données. Par exemple, http://www.example.com/ie.gif indique au navigateur Web de télécharger le fichier image nommé "ie.gif" à partir du serveur HTTP www.example.com. Dans le cas d'une URI de données, que supporte Internet Explorer 8, au lieu d'un localisateur de ressources, l'URI contient directement les données de la ressource. Afin d'incorporer des données binaires au sein de la chaîne, les données sont tout d'abord encodées à l'aide du codage en base64.
Par exemple, voici une URI de données qui représente un point bleu de 10 x 10 pixels :

```

oAAAAKAQMAAAC3/F3+AAAACXBIWXMAAA7DAAAOwwHHb6hkAA
AAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAgY0hSTQ
AAeiYAAICEAAD6AAAAgOgAAHUwAADqYAAAOpgAABdwnLpRPA
AAAANQTFRFALfvPEv6TAAAAAtJREFUCB1jYMAHAAAeAAEBGN
laAAAAAElFTkSuQmCC
```

Internet Explorer 8 autorise des URI de données jusqu'à 32 Ko. L'utilisation avisée d'URI de données à la place d'URL peut vous

aider à éviter de coûteux aller-retour réseau et améliorer la performance globale de vos sites, mais gardez à l'esprit que l'encodage en base64 implique une surcharge de traitement et que les URI de données ne peuvent pas être mises en cache indépendamment de leur contenant (document, script ou feuille de style).

Listing 1 : Code original [non optimisé] pour masquer des éléments n'appartenant pas à une certaine classe. sans les API de sélecteur, ce code est environ 5 fois plus long.

```
function showHideContentAreaContent(clsName)
{
   // ********************************************
   // Début: cacher des éléments qui ne devraient pas être montrés

   // Cacher tous les H3's ...
   var h3Collection = document.getElementsByTagName('h3');
   for (var i = 0; i < h3Collection.length; i++)
   {
      // ...à moins qu'ils ne fassent partie d'une classe.
      if (h3Collection[i].className.indexOf(clsName) != -1)
        h3Collection[i].style.display = 'block';
      else
        h3Collection[i].style.display = 'none';
   }

   // Cacher les tag P qui ont un nom de classe...
   var pCollection = document.getElementsByTagName('p');
   for (var j = 0; j < pCollection.length; j++)
   {
      // ...sauf s'ils ont une classe.
      if (pCollection[j].className.indexOf(clsName) != -1)
         pCollection[j].style.display = 'block';
      else
         pCollection[j].style.display = 'none';
      }
   }

   // Cacher tous les éléments UL ..
   var ulCollection = document.getElementsByTagName('ul');
   for (var k = 0; k < ulCollection.length; k++)
   {
      // ......sauf s'ils ont une classe.
      if (ulCollection[k].className.indexOf(clsName) != -1)
        ulCollection[k].style.display = 'block';
      else
        ulCollection[k].style.display = 'none';
   }

  // Finalement , cacher l'élément CODE  sauf s'il ne doit pas l'être
   var codeColl = document.getElementsByTagName('code');
   for (var m = 0; m < codeColl.length; m++)
   {
      if (codeColl[m].className.indexOf(clsName) != -1)
     codeColl[m].style.display = 'block';
      else
        codeColl[m].style.display = 'none';
   }
   // Fin : cacher des éléments
   // ********************************************
}
```

Listing 2 : Implémentation de la fonction getElementsByClassName en JavaScript.

```
function _MSHTML5_getElementsByClassName_Elem(classList)
{
    var tokens = classList.split(" ");
    if (tokens.length == 0)
    {
        return null;
        // D'après HTML5: "S'il n'y a pas de token spécifié
        // en argument, alors la méthode doit retourner une
 NodeList vide
    }
    // Pré-remplir la liste avec les résultats du premier
 token recherché
   var staticNodeList = this.querySelectorAll("." + tokens[0]);
    // Démarrer l'itération à 1 parce que la première
correspondance a déjà été effectuée
    for (var i = 1; i < tokens.length; i++)
    {
        // recherche de chaque token indépendamment.
       var tempList = this.querySelectorAll("." + tokens[i]);
        // Collecter les "gardiens" entre chaque itération.
        var resultList = new Array();
       for (var finalIter = 0; finalIter < staticNodeList.length;
             finalIter++)
        {
            var found = false;
           for (var tempIter = 0; tempIter < tempList.length;
                 tempIter++)
            {
                if (staticNodeList[finalIter] ==
                    tempList[tempIter])
                {
                    found = true;
                    break; // Finir si trouvé.
                }
            }
            if (found)
            {
                // Cet élément est dans les deux listes;
                // il devrait rester dans le prochain tour
 de vérification de token
                resultList.push(staticNodeList[finalIter]);
            }
        }
        // copie du résultat AND pour le prochain token
        staticNodeList = resultList;
    }
    return staticNodeList;
}
```

■ Pierre-Louis **COLL**
Senior Escalation Engineer

Formation des développeurs : changez de spécialité !

Pour les développeurs, la formation est toujours nécessaire afin de suivre l'évolution continuelle des technologies. Mais en période de crise, qui touche désormais également l'informatique, il faut parfois changer de spécialité pour favoriser son « employabilité ».

La mobilité est à l'ordre du jour pour tout le monde, et pour les développeurs en particulier. En cette période de crise, c'est surtout le changement qui peut être porteur d'avenir. Valoriser son parcours professionnel, progresser dans son métier, dans la même société ou en changeant d'emploi, exigent une réactualisation régulière de ses connaissances et de ses compétences. Il s'agit de trouver une formation permettant de mieux rebondir.

Si, du fait de la réduction des budgets formation dans les entreprises, la formation doit de plus en plus souvent être prise en charge par les personnes elles-mêmes, certains ont toutefois la chance de bénéficier d'une formation au sein de leur entreprise, qu'il s'agisse d'approfondissement ou de reconversion. Ce qui n'est pas négligeable, étant donné le coût de ces formations qui peut atteindre plusieurs milliers d'euros pour une « master class ». *« Les sociétés qui vont plutôt bien traverser la crise sont celles qui n'auront pas arrêté d'investir dans la formation »,* souligne **François Salaun,** Président directeur général de Softeam. Quoi qu'il en soit, autant choisir une formation donnant le plus de chances possible.

François Salaun, PDG de Softeam

A quoi se former ?

Une fois que le développeur a décidé de se former – ce qui exige évidemment de dégager du temps, à défaut de disponibilité totale – , se pose ensuite la question de la nouvelle compétence à acquérir. Compétence complémentaire, par exemple ajouter une corde « métier » à une spécialité technologique, ou bien opter pour un domaine en plein développement. Nous avons déjà parlé dans ces colonnes du logiciel libre, qui connaît un développement important en tant que facteur favorisant la croissance économique. Autre compétence prisée : la conception et réalisation d'applications nomades sur poste mobile. Ces filières connaissent un succès grandissant depuis fin 2008 à l'Afpa (Association nationale pour la formation professionnelle des adultes). *« Le développement sur plates-formes mobiles est une des tendances actuelles du marché »,* confirme **Xavier Paradon,** directeur technique de Valtech Training. Cette société vient d'ailleurs de lancer des formations liées au marché du téléphone mobile et autres terminaux (iPhone et Google Android). Les aspects sécurité restent aussi d'actualité, notamment en relation avec les applications internet et les outils du web 2.0. Sans oublier les incontournables compétences télécoms et réseaux.

Côté métier, les développeurs peuvent acquérir une compétence dans un secteur industriel, commercial, télécom ou autre. Une autre possibilité est d'acquérir une compétence dans un progiciel applicatif, comme SAP. D'autant plus que les projets applicatifs sont en forte hausse : +16% entre fin 2008 et fin 2009, selon une enquête de Comm'Back, alors que l'évolution des autres projets dans la même période, notamment l'infrastructure, est au point mort, voire en baisse. Cette expertise dans un progiciel sera utilement complétée par une compétence métier correspondant à un module applicatif de ce progiciel de gestion intégrée : finance, gestion de production, gestion commerciale, gestion des ressources humaines, etc.

D'une façon générale, la polyvalence est souvent exigée, notamment dans les services informatiques de petite taille. Le développeur peut assurer la conception technique, la programmation, la mise au point (tests) de l'application, ainsi que la maintenance corrective (résolution des bogues) et évolutive (évolution des besoins). Pour s'adapter à ce type d'entreprises, il peut donc être utile de se former ou de se perfectionner à toutes ces spécialités du développement. On demande de plus en plus souvent aux développeurs une double compétence : technique et métier, ou bien plusieurs techniques. Les DSI évoquent l'amélioration de la communication entre informatique et métiers (91%), la création de postes et de processus d'interface avec les métiers (72%), le développement de la double compétence métier et technique des collaborateurs SI (69%).

Comment passe-t-on d'une compétence à une autre ?

Les informaticiens issus des grands systèmes, du client-serveur des années 1980 et autres technologies anciennes, ont intérêt à se former à UML, Java, SOA, etc. Chez Softeam, le passage de C++, objet et temps réel (les nouvelles technologies des années 1990) aux technologies d'au-

egilia®

LEARNING

LE SPÉCIALISTE DE LA
FORMATION CERTIFIANTE
EN INFORMATIQUE
ET MANAGEMENT

Faire de vos succès
notre réussite

www.egilia.com

CONTACTEZ NOS CONSEILLERS FORMATION

N° National 0 800 800 900

APPEL GRATUIT DEPUIS UN POSTE FIXE

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

jourd'hui (UML, méthodologies agiles, SOA, etc.) se fait naturellement, les plus anciens évoluant vers l'architecture, le consulting, la gestion des processus métiers, où ils associent le savoir-faire technique avec les expériences métiers.

Par ailleurs, pour développer de nouvelles compétences en finances, Softeam a monté une formation pour ses ingénieurs. *« C'est un secteur consommateur de ressources informatiques et très demandeur de l'association de compétences techniques et métier »,* souligne François Salaun. *« C'est un excellent passeport pour nos ingénieurs. »* Appuyée sur un cabinet externe, First Finance, cette formation est organisée en trois cursus, ou « master classes », d'une durée de 3 mois. Chaque master class est un parcours de formation certifiant, mixant e-learning, formation en classe et validation des compétences. *« Quelques ingénieurs ont déjà une formation finances, mais pour la plupart, ils viennent d'un cursus technologique, informatique ou télécoms »,* ajoute le PDG de Softeam.

Outre la remise à niveau dans une technique informatique ou un domaine d'activité, il peut être utile de se perfectionner en langues, notamment l'anglais qui devient quasiment incontournable pour tous les métiers de l'informatique. L'AFPA, par exemple, propose des formations d'anglais technique pour l'informatique, requérant toutefois des notions de base dans cette langue.

Formation certifiante ou diplômante ?

« La valeur ajoutée d'une formation est la certification professionnelle », affirme **Frédéric Vaugeois,** responsable formation chez Egilia. La certification améliore l'employabilité. *« Hier, c'était un facteur différenciateur entre deux candidats. Aujourd'hui, la certification est quasiment devenue obligatoire. C'est un élément de base préalable à l'entretien d'embauche, qui fait monter le taux d'employabilité à près de 100% à 6 mois. »* Par exemple, Egilia offre, entre autres, des certifications sur Microsoft (.net) et Sun (Java). *« La formation avec validation par un examen officiel est un réel plus »,* précise Frédéric Vaugeois. La plupart des grands organismes de formation proposent d'ailleurs des formations certifiantes (Adhara, Anaska, Demos, Egilia, Global Knowledge, Orsys, Valtech Training, etc.). *« Ces formations sont plutôt suivies par des personnes qui n'étaient pas certifiées, qui suivent un emploi et peuvent ainsi en retrouver rapidement »,* ajoute Frédéric Vaugeois.

Les certifications s'appliquent à des technologies (Microsoft, Java, etc.), mais aussi à des métiers (spécialiste des tests, chef de projet, responsable qualité informatique...). (cf. encadré). En effet, ces différentes spécialités se professionnalisent de plus en plus. Ainsi, le Fafiec fait état de plus de 1500 stagiaires dans ses formations aux métiers du test, en estimant la progression pour l'année 2009 d'environ +20% des demandes de prise en charge pour ces cursus.

Frédéric Vaugeois, responsable formation Egilia

L'AFPA propose des formations diplômantes (cf. encadré). Par exemple, la formation « développeur logiciel » débouche sur un Titre Professionnel de niveau III et / ou des Certificats de Compétences Professionnelles (CCP). Ce titre, composé de deux certificats de compétences professionnelles, peut également être obtenu par une démarche de Validation des Acquis de l'Expérience (VAE).

■ Claire **Rémy**

Certifications

Certains organismes de formation proposent des cursus spécialisés permettant aux stagiaires souhaitant valider leurs compétences techniques de se préparer aux certifications éditeurs. Ils disposent à cet effet de partenariats avec les éditeurs et de centres de tests. Les certifications sont relatives à des outils ou technologies : Java et JEE, Linux, technologies Microsoft, Cisco, Citrix, etc. Citons, par exemple, Cisco Certified Network Associate (CCNA). (Pour les technologies Microsoft, cf. Programmez! n°122 de septembre 2009, p.45.)

Il existe aussi des certifications sanctionnant une démarche qualité ou des bonnes pratiques, telles que ITIL (Information Technology Infrastructure Library) ou les formations à la démarche CMMI (Capability Maturity Model Integrated). Ces formations visent à optimiser la performance de l'entreprise, assurer un meilleur contrôle des délais et des coûts. Elles permettent également de valoriser l'expertise des informaticiens dans leur parcours professionnel.

Les certifications font souvent l'objet d'un complément optionnel à la formation. Par exemple, IB-Formation (Groupe Cegos) propose une journée optionnelle de certification ITIL, comprenant une journée de préparation et le passage du test.

Formations diplômantes

Outre les diplômes délivrés par les écoles et universités, il existe des formations diplômantes pour les adultes en formation continue. Ainsi, les formations dispensées par l'Afpa (Association nationale pour la formation professionnelle des adultes) ainsi que la VAE (Valorisation des acquis de l'expérience) sont sanctionnées par des titres professionnels du Ministère chargé de l'Emploi. Ces titres correspondent à différents niveaux de qualification professionnelle.

Niveaux I et II, équivalent au niveau bac + 4 : Personnel occupant des emplois exigeant normalement une formation de niveau égal ou supérieur à celui de la licence ou des écoles d'ingénieur.

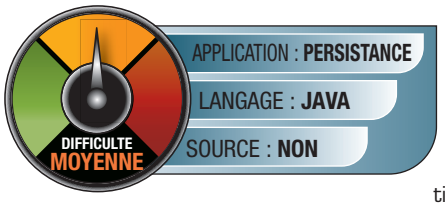
Niveau III, équivalent au niveau bac + 2 : Personnel occupant des emplois exigeant normalement une formation de niveau du brevet de technicien supérieur (BTS) ou du diplôme des Instituts Universitaires de Technologie (DUT) ou de fin de premier cycle de l'enseignement supérieur.

Niveau IV, équivalent au niveau bac : Personnel occupant des emplois de maîtrise ou possédant une qualification d'un niveau équivalent à celui du baccalauréat technique ou de technicien ou du brevet de technicien.

Hibernate Survival Guide

3e partie

Après avoir expliqué ce qu'était le cache de premier niveau et son rôle fondamental dans le fonctionnement d'Hibernate (HSG partie I) et la manière dont est gérée la persistance des relations (HSG partie II), nous allons nous attaquer, dans cet article au système de requêtage, aux pièges qu'il présente, ainsi qu'aux solutions pour les éviter.



Il ne s'agit pas ici de décrire le langage (HQL, JPA-QL) utilisé par Hibernate. De multiples ouvrages, à commencer par la documentation de référence Hibernate, le font longuement et avec talent. Le propos ici est plutôt de décrire comment Hibernate exploite ses requêtes. Et d'expliquer comment on peut éviter les deux écueils majeurs : le problème du N+1 et les jointures croisées insuffisamment réduites. C'est-à-dire comment utiliser de façon efficace le système de requêtage d'Hibernate.

LAZY OR NOT LAZY ?

Les relations entre deux entités peuvent être déclarées « paresseuses » (lazy en anglais). Une relation paresseuse qui part de l'entité A vers l'entité B, n'est pas résolue lorsque l'entité A est chargée. C'est uniquement lorsque l'on tente de passer de l'entité A pour atteindre l'entité B qu'Hibernate charge l'entité B.

Les relations paresseuses sont une nécessité vitale : si les relations étaient systématiquement résolues, le seul fait de charger une entité entraînerait automatiquement le chargement d'une grappe d'entités attachées – directement ou non – qui peut être potentiellement très large, voire englober l'ensemble de la base de données !

Pour éviter ce problème, peut-on – doit-on ? – déclarer toutes les relations comme étant paresseuses ? La réponse est complexe, évidemment. Le fait de déclarer qu'une relation est « souhaitée » (eager), c'est-à-dire non paresseuse, a de multiples avantages :

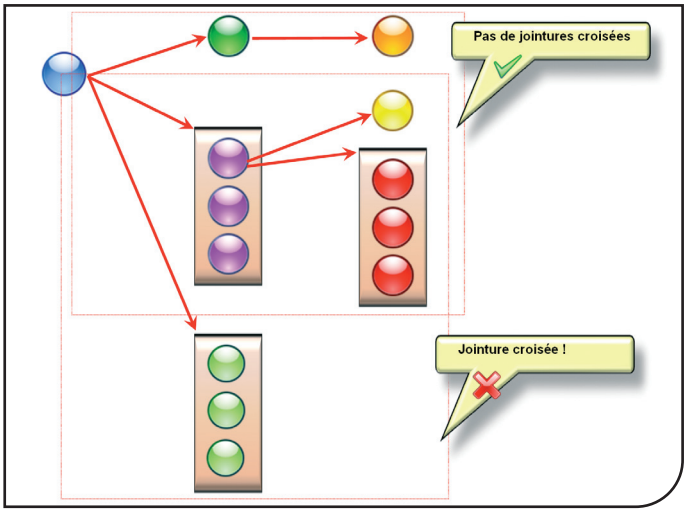
- Hibernate sait (la plupart du temps) charger l'entité A et en même temps l'entité B en une seule requête SQL lorsque l'on utilise la méthode « Session.get(EntityAClass, idA) ».
- Si l'entité A est détachée (retirée du cache de premier niveau), le fait de traverser le lien de A vers B n'entraînera pas la levée d'une exception de type LazyLoadingException.

La réponse est donc de ne pas utiliser systématiquement le chargement paresseux. En fait, le comportement par défaut d'Hibernate est souvent le plus approprié : les relations de type référence (OneToOne et ManyToOne) sont « souhaitées » et les relations de type collection de références (OneToMany et ManyToMany) sont « paresseuses ». En utilisant cette stratégie, il n'y a pas de risque de charger un arbre d'objets, mais seulement un fil. C'est un bon compromis.

Le problème est que les relations « souhaitées », même en nombre limité, peuvent provoquer le problème dit du « N+1 » lorsque l'on utilise le mécanisme de requêtage d'Hibernate.

LE PROBLÈME DU « N+1 »

Le problème du « N+1 » consiste à envoyer N+1 requêtes vers la base de données pour avoir la description étendue d'une liste de N



objets. La première requête fait l'acquisition des informations de base des N objets de la liste (une liste de clients par exemple). Et ensuite pour chaque client, une série de requêtes – une par client - permet d'obtenir la liste des produits attachés à chacun d'eux.

Un tel comportement a des conséquences souvent très fâcheuses sur les performances. Le « coût » associé à la prise en charge d'une requête par la base de données est largement supérieur à la restitution d'une ligne de la base de données. Dit plus simplement, une requête qui retourne mille lignes est beaucoup moins coûteuse que mille requêtes retournant une ligne ou même cent requêtes retournant dix lignes.

Ce phénomène est très courant sur Hibernate. Pour de multiple raisons que nous allons aborder dans la suite de l'article. Ce phénomène explique bien souvent les temps de réponse désastreux de projets utilisant Hibernate. Il est pourtant assez simple de diagnostiquer le mal : dans le fichier de configuration d'Hibernate ou JPA (hibernate.cfg.xml ou persistence.xml), il faut affecter la valeur « true » à la propriété « hibernate.show_sql ». Le phénomène se traduit alors par un flot de messages présentant des requêtes SQL, apparaissant sur le log ou la console, alors que peu de requêtes HQL ont été exécutées.

Premier conseil donc : ne pas laisser « traîner » les avalanches suspectes de messages SQL d'Hibernate !

En utilisant le requêtage Hibernate, les raisons de tomber dans ce piège sont multiples. La première raison, la plus surprenante est la présence d'au moins une relation en chargement « eager ». Surprenant, car justement, la méthode Session.get() utilise le fait que la relation soit marquée « eager » pour optimiser le chargement de l'entité en n'émettant qu'une seule requête ! Certes. Mais cette optimisation n'est pas utilisée lorsque vous utilisez HQL (ou l'interface Criteria) !

Ainsi, pour un client muni de son adresse (avec un lien « eager » entre Client et Adresse), il faudra une requête à Session.get pour ramener un objet client à partir de son identifiant :

```
Client c = sess.get(Client.class, 7) ;
```

Et 1+N requêtes pour ramener une liste de clients avec leurs adresses ! (une requête pour ramener la liste des N entités clientes et N requêtes pour ramener les N adresses de ces clients).

```
List<Client> clients = s.createQuery("select c form Client c").list();
```

La requête HQL ne demande pas les adresses. Certes. Mais comme le lien entre Client et Adresse est « eager », elles sont automatiquement chargées – d'où les N requêtes supplémentaires intempestives. La réflexion légitime que l'on peut alors se faire est : pourquoi Hibernate ne complète-t-il pas la requête SQL pour ramasser automatiquement les adresses liées aux clients recherchés ? Il fait bien ce travail pour les appels de type Session.get. Il sait faire donc. Alors ?
Le fait que le chargement des objets attachés par une relation « eager » ne soit pas pris en compte lorsque l'on utilise un mécanisme de requêtage (HQL, Criteria) est une décision délibérée de l'équipe Hibernate, motivée par le fait qu'il est alors très difficile pour le moteur Hibernate d'éviter l'autre grand problème : la jointure croisée insuffisamment réduite (dont nous parlons plus loin). Au développeur de le faire au coup par coup !
La première option que l'on pourrait prendre est de n'utiliser que des relations paresseuses. Cette solution est certes envisageable mais pas sans inconvénients : c'est se priver de toute possibilité d'optimisation du chargement d'un objet par « Session.get » (celles demandées explicitement et celles demandées par effet de bord, comme lors de certains appels à « Session.merge » par exemple). Et puis, on peut aussi avoir vraiment besoin des adresses chaque fois que l'on charge un client.

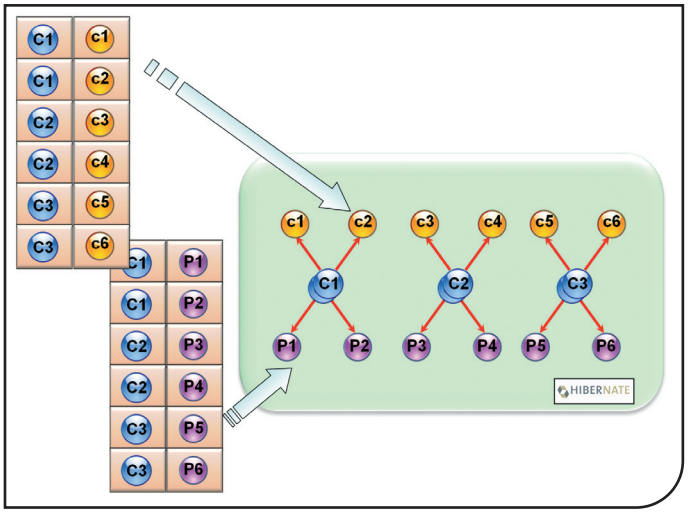
LE CHARGEMENT À LA VOLÉE

Heureusement, Hibernate nous propose un mécanisme qui permet de répondre au problème du N+1 : c'est le chargement à la volée (fetch). Pour cela, on ajoute la définition d'une jointure – généralement externe – à notre requête HQL. Et on indique que cette jointure doit être préchargée. Pour reprendre l'exemple donné plus haut :

```
List<Client> clients =
s.createQuery("select c form Client c left outer join fetch c.
adresse").list();
```

Le fait de précharger une jointure a deux vertus :
• Pré-charger les adresses (c'est l'effet recherché J)
• Initialiser le lien entre les entités clients et les entités adresses.
Le second effet n'est pas sans conséquence. Pour illustrer ce point, prenons le cas d'une relation Client-Produit. Un client peut disposer de plusieurs produits. Le code suivant :

```
List<Client> clients =
s.createQuery("select c form Client c left outer join fetch c.
produits "+
"where c.ville.name='Grenoble' ").list();
```



ramène la liste des clients Grenoblois avec leurs produits associés. Le lien entre les clients et les produits est établi.

Si on avait écrit :

```
List<Produit> produits =
  s.createQuery("select p form Produit p where p.client.ville
.name='Grenoble' ").list();
List<Client> clients =
  s.createQuery("select c form Client c where c.ville.name='Gre
noble' ").list();
```

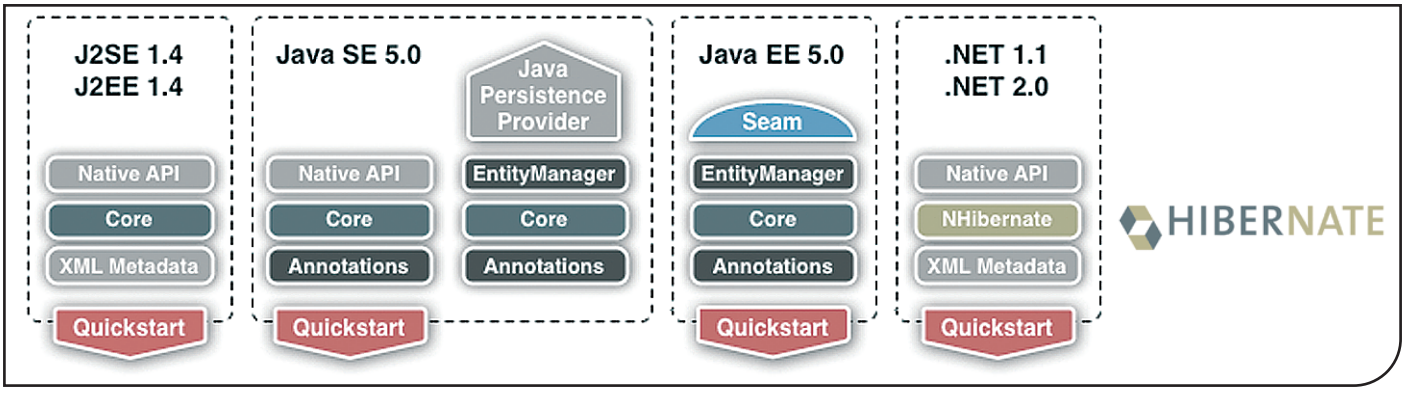
On aurait ramené les mêmes objets (clients et produits), mais les collections de produits dans les clients n'auraient pas été initialisées. Conséquence : si on tente d'accéder aux produits d'un client, Hibernate lance une requête SQL pour recenser ces produits. Pour s'apercevoir finalement qu'il en disposait déjà dans son cache de premier niveau et que c'était donc inutile !

CHARGEMENTS MULTIPLES

Peut-on pré-charger ainsi plusieurs relations ? La réponse est oui et non, tout dépend du contexte. En fait, le seul cas qui pose problème est celui qui se présente lorsqu'il faut pré-charger deux collections non imbriquées. C'est un cas, malheureusement très courant. Reprenons par exemple, le cas de notre client qui dispose d'une liste de produits et d'une liste de contrats. Le code suivant ramène, comme voulu, clients, produits et contrats en une seule requête SQL :

```
List<Client> clients =
  s.createQuery("select c form Client c"+
    " left outer join fetch c.produits "+
    " left outer join fetch c.contrats "+
    " where c.ville.name='Grenoble' ").list();
```

C'est super, n'est ce pas ? Et bien non, ce n'est pas super. Pourquoi ? Parce que la requête SQL générée contient deux jointures complètement indépendantes. Conséquence : le nombre de lignes retournées par SQL pour un client donné est le produit du nombre de produits dont le client dispose par le nombre de ses contrats. Si un client dispose de 10 produits et de 8 contrats, SQL retourne 80 lignes ! Evidement, le problème s'amplifie si on veut aussi ramener la



liste des contacts ... Notons que si les listes sont imbriquées, ce problème ne se pose pas. Par exemple, pour des livres constitués de chapitres, eux même divisés en paragraphes, le code suivant :

```
List<Livre> livres =
s.createQuery("select l form Livre l"+
" left outer join fetch l.chapitres c "+
" left outer join fetch c.paragraphes p"+
" where l.titre like 'A%' ").list();
```

retourne la liste des livres dont le titre commence par A, avec chapitres et paragraphes, pré-chargés et déjà reliés entre eux et aux livres. Le nombre de lignes retournées par SQL est égal au nombre de paragraphes définis en base de données. Et cela parce qu'il existe une restriction entre les chapitres pré-chargés et les paragraphes pré-chargés.
De même, les relations de type référence, ne posent pas non plus de problèmes : On peut multiplier un par un et cela à l'infini sans que cela fasse plus de un ! Ainsi, rien n'interdit de ramener un client avec la description de son adresse, la description de l'agence qui le gère, la catégorie à laquelle il appartient, etc.

```
List<Client> clients =
s.createQuery("select c form Client c"+
" left outer join fetch c.adresse "+
" left outer join fetch c.agence "+
" left outer join fetch c.categorie "+
" where c.ville.name='Grenoble' ").list();
```

Bien que ces trois jointures soient complètement indépendantes entre elles, elles ne posent aucun problème car elles ne peuvent retourner chacune, qu'une seule ligne.
La règle est simple : vous pouvez pré-charger autant de relations de type référence que vous voulez **plus une** (et une seule !) série de relations de type collections de références **imbriquées**.

L'ART DE PRÉ-CHARGER

Mais alors, comment ramener efficacement mes clients, avec leurs contrats et leurs produits ? En faisant appel à notre ami : le cache de premier niveau J. Si vous devez pré-charger plusieurs listes indépendantes, chargez la liste plusieurs fois ! La synthèse sera faite par Hibernate dans le cache de premier niveau. Ainsi, pour charger nos clients avec leurs produits et leurs contrats, on écrira :

```
List<Client> clients =
  s.createQuery("select c form Client c"+
    " left outer join fetch c.produits "+
    " where c.ville.name='Grenoble' ").list();
clients =
  s.createQuery("select c form Client c"+
    " left outer join fetch c.contrats "+
    " where c.ville.name='Grenoble' ").list();
```

La première ligne vous semble inutile ? (puisque les mêmes clients sont retournés par la seconde requête) ? Détrompez-vous ! Cette requête charge les clients demandés avec leurs produits. Le résultat n'est pas perdu car il est conservé dans le cache de premier niveau ! La seconde requête ramène aussi la description de ces mêmes clients avec leurs contrats. Le cache de premier niveau s'aperçoit à ce moment là, qu'il dispose déjà des clients demandés. Il se contente alors de les compléter, c'est-à-dire de leur associer les contrats. Et renvoie ensuite des objets clients disposant à la fois de leurs produits et de leurs contrats. Mais la base de données n'a renvoyé que 10+8 lignes pour chaque client et non 10x8 lignes. Nous ne sommes pas non plus tombés dans le problème du N+1 (mais plutôt dans le « 1+1 » qui est parfaitement acceptable J). Astucieux, non ?

On peut remarquer que cette méthode ramène « trop » d'informations puisque la description des clients est retournée inutilement par la seconde requête. On pourrait être tenté d'optimiser en écrivant :

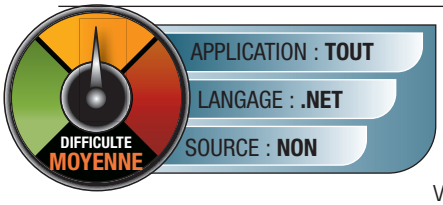
```
List<Produit> produits =
  s.createQuery("select p form Produit p where p.client.ville.
name='Grenoble' ").list();
  List<Client> clients =
s.createQuery("select c form Client c"+
" left outer join fetch c.contrats "+
" where c.ville.name='Grenoble' ").list();
```

C'est une mauvaise idée! Certes, les objets ramenés sont les mêmes. Mais le lien partant de client vers produit n'est pas établi (cf plus haut) ! Et toute tentative d'accéder à la liste des produits à partir d'un client se traduira pour Hibernate par une requête SQL intempestive.

■ Henri Darmet
Directeur Technique Objet Direct / Homsys Group
Objet Direct, filiale à 100 % de Homsys Group, est une société de conseil, de services et de formation, spécialisée sur les technologies objet et web. Conseil en méthodologie, en architecture et en urbanisation du SI, développement applicatif, édition et distribution de logiciels.
www.objetdirect.com

Migrez vos applications Delphi vers Mac et Linux avec PRISM

Delphi PRISM hérite du savoir-faire de Borland dans le domaine des outils de développement. Il est le successeur de CodeGear, une solution facilitant le développement d'applications .NET avec le langage Delphi au sein de Visual Studio.



PRISM permet aux développeurs Delphi de construire des applications .NET et de profiter de toute la richesse fonctionnelle de la plate-forme : WinForms, WPF, ASP .NET et LINQ. Delphi PRISM s'adresse donc aussi bien aux développeurs web qu'aux développeurs d'applications de bureau. Son grand intérêt, c'est avant tout le support de la technologie Mono et aussi la disponibilité des applications à la fois sous Mac et Windows, une première.

LES AVANTAGES

Delphi PRISM tire ses avantages du framework .NET 3.5 et profite du support de celui-ci sous Linux et Mac OS via Mono. L'aspect multi-plate-forme est une valeur essentielle de cet outil. Delphi PRISM inclut un IDE complet basé sur Visual Studio : le meilleur en termes de développement .NET. Cela permet d'éviter le surcoût de l'achat d'une licence de Visual Studio et de conserver le même outil pour les développeurs .NET.

Delphi PRISM inclut une compatibilité avec le code Delphi Win32. Cette fonctionnalité optionnelle peut être exploitée par projet ou par sous-projet. Ce qui facilitera la reprise de l'existant pour des applications anciennes qui ont besoin d'être conservées en l'état ou d'évoluer. Delphi PRISM peut travailler comme une solution qui mélange code managé et code natif Windows. Il permet ainsi de migrer progressivement depuis du code destiné à l'environnement Win32 vers .NET en utilisant notamment des solutions tierces à l'image de RemObjects Hydra. Hydra offre l'infrastructure nécessaire à la construction de portions de codes managées tout en ayant d'autres portions de codes natifs. Delphi PRISM propose différents types de projets, l'accent étant mis sur l'interopérabilité avec Mono et le langage Cocoa# pour Mac :

- Windows Application (WinForms)
- Windows Control Library
- Mono Console Application
- Cocoa# Application (Mac OS Tiger et Leopard)
- Mono Class Library
- WinForms Application (Mac OS X)
- Gtk# Application

Une base de données intégrée permet de s'affranchir de l'utilisation de SQL Server. Blackfish est une base de données SQL haute performance ayant une faible empreinte mémoire. Celle-ci est malgré tout compatible avec le standard SQL-92 et gère les transactions. Etant entièrement écrite en .NET, Blackfish pourra être déployée aussi bien en tant que serveur "stand alone" qu'intégrée dans une application comme on peut le faire avec SQLite. Blackfish a l'avantage de supporter les procédures stockées écrites en Delphi ou tout autre langage compatible .NET comme C# ou VB .NET.

LES LIMITATIONS

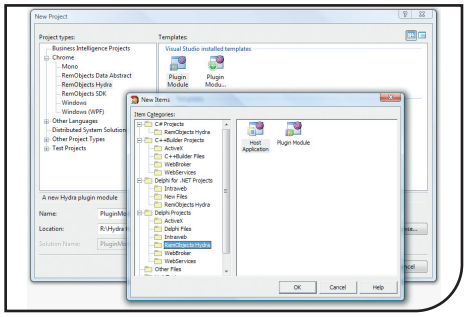
On pourra regretter l'absence de cette solution sous Linux et Mac OS en raison de la forte dépendance avec les outils Visual Studio. De plus, il ne sera pas possible d'écrire du code C# ou Visual Basic .NET au sein d'un projet Delphi PRISM.

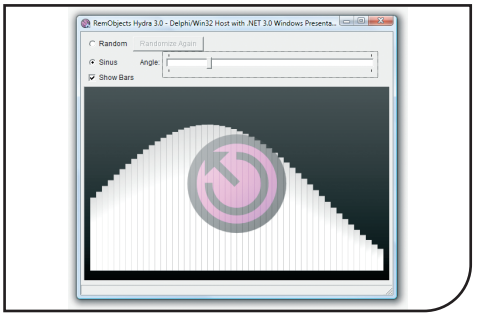
Un code source "Delphi PRISM" n'est pas du Delphi mais de l'Oxygene, un langage proche mais différent. Il est donc important de noter que la compatibilité avec les anciennes plates-formes d'exécution sera perdue. Il pourra donc être intéressant d'évaluer C# 4.0 et PRISM, tant en termes de fonctionnalités que de structure de vos applications. Une migration vers PRISM n'est pas nécessairement la meilleure approche, pensez à C# et VB .NET. Dans les deux cas un effort de réécriture de code est de toute façon nécessaire.

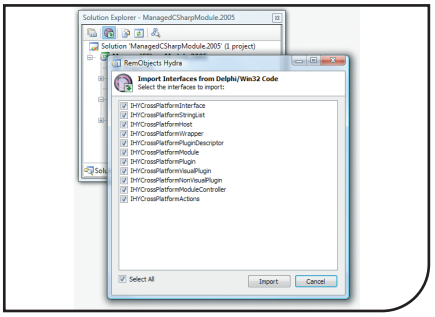
La différence la plus évidente entre Delphi PRISM et C# est bien sûr liée aux fonctionnalités complémentaires apportées par Delphi, dont Blackfish SQL fait partie, et la syntaxe objet du langage Pascal. Les développeurs ne maîtrisant pas ces aspects risquent de s'y perdre, au moins dans un premier temps. Pour cette raison, il sera plus difficile de trouver des compétences Delphi que C#, un critère à prendre en compte en tant que chef de projet.

CRÉONS UNE APPLICATION WINDOWS FORMS

La licence du produit a un coût de plusieurs centaines d'euro et inclut la maintenance ainsi que le support. Cependant, il sera possible d'évaluer ce logiciel gratuitement pendant une période d'essai.







Delphi PRISM est disponible en version française tout comme sa documentation. N'hésitez pas à télécharger la version d'évaluation pour réaliser ce tutoriel.
Delphi PRISM requiert la version 3.5 du framework .NET. Il sera également nécessaire d'installer Visual Studio Shell et le compilateur Oxygen. Afin de permettre la compatibilité avec les environnements Linux et Mac OS, on installera Mono. Vous pourrez ensuite installer PRISM en suivant les instructions.
Nous allons créer une application WinForms. Nous commençons par lancer l'environnement depuis le menu Démarrer. Puis nous créons un nouveau projet de type Windows.
Nous pouvons à présent créer/ajouter les composants graphiques à notre application, par exemple une ListBox. Nous y ajouterons des éléments par du code Delphi PRISM. On mettra le code suivant dans le fichier source MAIN.PAS :

```
// On déclare nos variables
var
  Contact: MyContact;
  MyContacts : List<Contact>;
begin

// On initialise une liste typée en utilisant les generics
MyContacts := new List<Contact>;

// On crée un album puis on l'insère dans la liste
MyContact := new Contact;
MyContact.firstName := 'Michel';
MyContact.lastName := 'Dupond';
MyContact.phoneNumber := '+33 1 23 45 67 89';
MyContact.country := 'France';
MyContacts.Add(MyContact);

// On exécute une requête LINQ
var Contacts := from Contact in MyContacts where Contact.country
.Equals('France');

// On parcourt le résultat et on l'ajoute dans une ListBox
for Contact in Contacts do
begin
   var item := String.Format('{0} {1}: {2}', Contact.Element
('firstName').Value, Contact.Element('LastName').Value , Contact
.Element('phoneNumber').Value);
    lbFoundContacts.Items.Add(item);
end;
end;
```

La migration facile avec Hydra 3.0

Hydra est un framework qui permet la création d'un pont applicatif entre Delphi Win32 et la plate-forme .NET de Microsoft. Aujourd'hui, de nombreux développeurs Delphi considèrent que la migration vers la plate-forme .NET est devenue un passage obligé. Cela s'explique notamment par le couplage historique entre Delphi et Windows qui est favorable à .NET. Les développeurs apprécieront les fonctionnalités avancées du framework .NET et notamment de Windows Presentation Foundation et de LINQ. Hydra est présent pour aider les développeurs à profiter de la richesse de leur code existant, tout en leur permettant de prolonger leur utilisation en étendant leur compatibilité au framework .NET. Hydra permet également de faciliter la migration des applications vers .NET.

RemObjects: Le compilateur

Le compilateur Delphi PRISM a été conçu par RemObjects. Il s'agit du même compilateur utilisé pour les produits Chrome et Oxygene qui sont maintenant devenus des sous-projets de Delphi PRISM. RemObjects Software et Embarcadero sont des partenaires particulièrement proches. Les licences Embarcadero de RemObjects Software sont incluses au sein de Delphi PRISM. Les deux entreprises travaillent ensemble sur des produits avancés dans l'objectif de progresser techniquement et d'offrir les meilleures solutions à leurs utilisateurs. RemObjects est le leader pour le langage Delphi.

Dans l'exemple qui suit, nous allons cette fois charger un document XML afin de remplir notre ListBox. La syntaxe est particulièrement concise grâce à la puissance de LINQ. La requête est à peine plus complexe que celle de l'exemple précédent. Il est particulièrement aisé de charger un document XML.

```
var
  ContactData: XDocument;
begin
   // Charge le document XML (objet de type System.Xml.Linq
.XDocument)
  ContactData := XDocument.Load('../../contacts.xml');

  // Cette requête récupérera les contacts pour la France
  var AlbumsLastYear := from Contact in ContactData.Descendants
('contact') where Contact.Element('country').Value = 'France'
select Contact;

  for Album in AlbumsLastYear do
  begin
    var item := String.Format('{0} {1}: {2}', Contact.Element
('firstName').Value, Contact.Element('LastName').Value , Contact
.Element('phoneNumber').Value);
    lbFoundContacts.Items.Add(item);
  end;
```

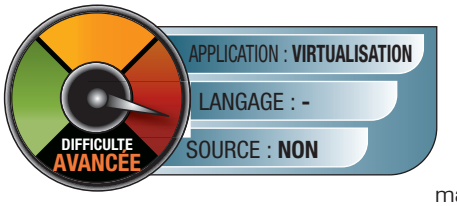
CONCLUSION

Comme le cristal découpe la lumière en de multiples couleurs, PRISM découpe le code de votre application en multiples modules permettant de s'affranchir des contraintes de plate-forme, de framework et de base de données. Tout en fusionnant le développement d'applications Web, RIA et de bureau grâce à la puissance de .NET, Delphi PRISM fournit un framework clef en main permettant de construire des applications à la fois pour Windows, Mac et Linux, avec une compatibilité au niveau du code source. Delphi PRISM permet de connecter des applications win32 existantes avec d'autres développées en .NET. La réécriture du code n'est pas un passage obligé grâce à ce framework. Les développeurs Delphi expérimentés préféreront cette solution plutôt que C#, bien que la migration ne soit pas aussi simple que le décrivent les plaquettes commerciales.

■ Loïc **Guillois**

Virtualisez vos Linux avec VirtualBox

De plus en plus utilisée dans les entreprises, la virtualisation peut rendre d'innombrables services au lecteur de Programmez!, qu'il soit professionnel ou passionné. Découvrons ensemble comment virtualiser des Linux au moyen de VirtualBox.



La virtualisation prend une place grandissante dans le monde de l'informatique. La virtualisation consiste à émuler, au sein d'une seule machine physique, une ou plusieurs machines dites virtuelles au moyen d'un logiciel (et éventuellement de matériel) spécifique. Ce logiciel de virtualisation devient le réceptacle pour un ou plusieurs systèmes d'exploitation, différents ou identiques. Les avantages de ce procédé sont nombreux. Pour les entreprises, avoir une machine qui devient l'équivalent de plusieurs PC représente une simplification évidente en ce qui concerne la gestion d'un parc serveur et une optimisation des ressources matérielles, pour ne citer que cela d'un très vaste sujet. Dans cet article, nous allons, quant à nous, adopter un point de vue plus 'geek'. En tant que tel, avoir à disposition de nombreux systèmes d'exploitation nous intéresse.

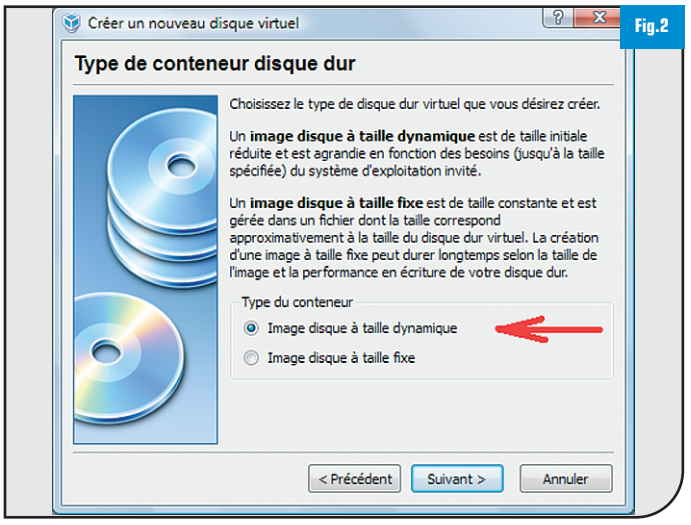
Mais pour disposer de plusieurs systèmes d'exploitation, il faut, optimalement, installer autant de disques durs que de systèmes, ou à la rigueur avoir des disques intelligemment partitionnés. Le nombre de disques qu'un PC peut accueillir étant restreint, nous sommes vite limités quand au nombre d'OS pouvant être installés sur notre PC. Et quand bien même aurions nous configuré une machine en quadruple ou quintuple boot, nous ne pourrions lancer qu'un seul OS à la fois. Ou alors nous devrions avoir de nombreuses machines physiques, ce qui n'est guère économique, question finance et encombrement. La virtualisation fait sauter ces limites. Nous allons découvrir cela en manipulant VirtualBox, un logiciel de virtualisation pour PC, initialement développé par l'entreprise Innotek. Celle-ci a finalement été rachetée par Sun Microsystem qui a repris et développé VirtualBox. VirtualBox est maintenant un logiciel très abouti, open source, et tournant sous Windows, Mac OS X, Linux, Solaris. VirtualBox est capable non seulement de virtualiser tous les systèmes pré-cités, mais aussi une quantité d'autres de MS-DOS à OpenBSD en passant par IBM OS/2. Vous pouvez obtenir VirtualBox à http://www.virtualbox.org. Nous allons nous amuser avec un VirtualBox tournant sous Windows Vista qui sera le système hôte, et nous allons virtualiser des Linux. Mais sur le fond, le contenu de cet article, destiné à initier le lecteur à la virtualisation et à la manipulation de VirtualBox, est valable pour toutes les combinaisons.

1 ANATOMIE DE VIRTUALBOX

VirtualBox est mal nommé car ce nom n'est pas à la hauteur de ses possibilités. VirtualBox devrait s'appeler VirtualBoxes. Car pour comprendre d'emblée les possibilités de ce logiciel plutôt bluffant, il ne faut pas voir en lui une seule machine virtuelle, mais un rack, dans lequel vous pouvez empiler un nombre illimité de machines virtuelles et faire tourner celles-ci simultanément, dans la limite, il en faut bien une quand même, de la mémoire de la machine physique hôte. Cela permet déjà de belles choses. Ainsi votre serviteur fait-il couramment tourner, sur un portable double coeur avec 3 Go de RAM, le système Vista Hôte et deux Linux virtualisés (ou invités) simultanément. Autrement dit trois machines en une avec des performances qui restent excellentes :) Commençons à manipuler. Installez VirtualBox. Au cours de la procédure d'installation, VirtualBox vous demandera d'accepter l'installation de pilotes qui lui sont absolument nécessaires pour émuler des cartes réseaux ou autres. Vous devez donc accepter ces installations.

Lancez VirtualBox et tout de suite allez visiter votre répertoire utilisateur. VirtualBox y aura créé un répertoire du nom de *.VirtualBox*. Dans celui-ci vous trouvez *VirtualBox.xml*, un fichier XML qui contiendra le nom de toutes les machines virtualisées, qui sont identifiées chacune par un GUID, un identificateur unique. Ce fichier renvoie à un sous-répertoire 'Machines", contenant autant de sous-répertoires que de systèmes virtualisés. Chacun de ces répertoires contient à

Spécification de l'OS invité dans une machine virtuelle.



Vive les disques dynamiques de VirtualBox.

son tour un fichier XML décrivant dans le détail chaque machine. Ainsi, pour chaque machine virtuelle, répétons-le, vous trouvez la mémoire disponible, les réglages du BIOS, les adresses MAC des cartes réseau, etc. Enfin le tout premier fichier XML, *VirtualBox.xml,* contient aussi des références sur les images des machines. Images qui sont rangées par défaut dans le sous-répertoire HardDisk. Qu'est-ce qu'une image ? C'est en ***un seul fichier,*** l'équivalent d'un disque dur complet: partitions, fichiers du système, fichiers de données, tout… Et déjà on constate un avantage énorme à cette virtualisation : la compartimentation. En effet, il n'est pas possible, en installant un OS virtuel, d'en écraser ou endommager un autre, chose qui est vite arrivée dans un environnement non virtuel. Autre avantage non moins énorme : la simplicité. Chaque OS consistant en un seul fichier, rien de plus simple que de stocker ceux-ci, par exemple sur un disque dur USB externe. Ainsi votre serviteur a-t-il, sur un tel disque à grande capacité, qui tient pourtant dans la poche, 5 Linux Gentoo et Ubuntu, un Open Solaris et même des Windows virtualisés. Rien de plus simple également que de sauvegarder entièrement un système virtualisé avant d'expérimenter avec lui. Pourquoi un disque USB externe ? Parce qu'il suffit de brancher celui-ci sur une autre machine hôte avec VirtualBox installé avec le même fichier *VirtualBox.xml* mentionné plus haut, pour retrouver le même environnement.

2 CRÉER UNE MACHINE VIRTUELLE

Configurer une nouvelle machine virtuelle et y installer un OS est très simple. On commence par cliquer, dans l'interface utilisateur de VirtualBox sur le bouton 'nouveau', comme le bon sens le suggère fortement. On traverse alors une succession de boîtes de dialogue. Passons directement au second dialogue, dans celui-ci, comme dans la [Fig.1], on spécifie l'OS invité. Ne pas trop s'inquiéter de ce que l'on dit à ce stade. En effet nous sommes en train de créer une machine virtuelle, donc un PC. Et non un OS. En fait ce que l'on donne dans ce dialogue ne se répercute que sur … l'icône de la VM. Et cela peut être changé ensuite à tout moment :) Dans le dialogue suivant vous donnerez la quantité de mémoire attribuée à la VM. 1028 Mo est une valeur qui va très bien. Le dialogue qui va suivre est plus important car il concerne la création du disque dur virtuel, ce fichier tout-en-un, que nous avons évoqué plus haut. Il est aussi possible, à ce stade de reprendre un disque existant. En choisissant la création, vous arrivez à un dialogue vous demandant si vous souhaitez un disque à taille dynamique ou à taille fixe [Fig.2]. Sans hésiter, choisissez un disque à taille dynamique. Concrètement, ce choix signifie que votre disque n'augmentera que selon les besoins, jusqu'à la taille maximum que vous spécifierez plus loin. Ceci présente l'inestimable avantage de pouvoir conserver une bonne collection de tels disques dans un encombrement réduit. Ne vous inquiétez pas du ou des systèmes de fichiers qui seront utilisés. Le disque dynamique n'a pas d'incompatibilité avec eux. Une fois arrivé au bout de la succession de dialogues, votre nouvelle machine virtuelle sera créée.

3 AFFINER LA CONFIGURATION DE LA MACHINE VIRTUELLE

Comme le montre la capture [Fig.3] l'interface de VirtualBox vous présente une vue de votre machine. Il est pertinent d'affiner dès à présent sa configuration, avant l'installation de l'OS invité. La première chose à faire est d'augmenter la taille de la mémoire vidéo qui est par défaut de 12 Mo. 128 Mo est une valeur qui va bien. Pour régler cela, cliquez sur 'Général'. De même vous pouvez activer le port série et l'USB de la machine. Autre réglage à faire immédiatement: celui de la carte réseau.

Le modèle de la carte émulée ne vous convient pas forcément. Son mode de fonctionnement non plus. Par défaut, le mode est NAT mais pouvez préférer *ponter* cette carte virtuelle avec la carte physique de la machine physique hôte. Il est important ici de bien avoir en tête que nous configurons une machine, fut-elle virtuelle. Ainsi si vous changez la carte réseau, vous changez son adresse MAC du même coup… Si un OS virtuel ne trouve plus le réseau d'un coup, c'est sans doute un problème d'adresse MAC. Nous verrons un exemple plus loin. Il est encore possible d'ajouter d'autres disques à la machine, tout comme on le fait avec une machine physique.

4 INSTALLER L'OS INVITÉ

Au premier lancement de votre machine virtuelle, il vous sera demandé d'insérer le Cd-Rom d'installation de l'OS dans le lecteur de la machine physique. Vous pouvez faire cela, et suivre la procédure d'installation tout à fait normalement. Qui plus est sans crainte de détruire quoi que ce soit sur la machine hôte. Mais il y a mieux à faire. En effet, comme le montre la capture [Fig.4] il est possible de démarrer directement sur une image ISO, ce qui évite la gravure d'un Cd-Rom. Et sur-

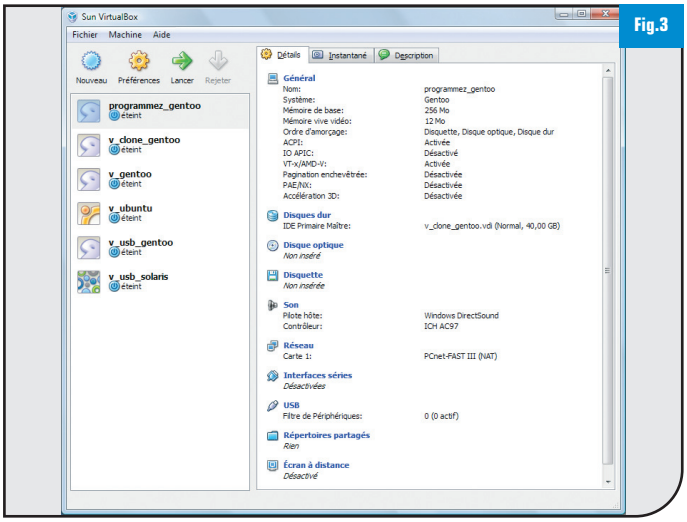


Fig.3

Vue d'ensemble des caractéristiques "matérielles" de votre machine virtuelle.

Fig.4

VirtualBox permet de démarrer directement depuis une image ISO, sans avoir à graver de Cd-Rom.

tout, cela peut se faire à tout moment, depuis le panneau de contrôle de la VM ou depuis son menu de gestion des périphériques [Fig.5]. Cette possibilité est idéale pour réparer un OS virtuel abimé à la suite d'une erreur de manipulation, même si vous êtes en voyage et que vous n'avez pas de CD sous la main :) Une autre possibilité pour réparer est de prendre le disque virtuel de l'OS cassé et de l'ajouter en second disque à une VM en bon état. On peut alors démarrer sur celle-ci, puis intervenir sur le disque de l'OS endommagé. Autant de manipulations qui demanderaient beaucoup de matériel mais qui se font en quelques clics de souris dans le monde de la virtualisation :) Finissez l'installation de votre système invité. N'oubliez pas, en ce qui concerne Linux que le noyau et les pilotes doivent être compilés en fonction du matériel, fut-il virtuel, exactement de la même façon que cela doit être avec une machine physique.

5 AFFICHAGE ET ADDITIONS INVITÉ

Si vous avez installé un Linux Ubuntu par exemple, vous vous retrouvez dans le bureau mais avec une fenêtre de taille ridicule. Pour avoir un affichage parfait il faut installer dans l'OS invité ce qui s'appelle des *Additions Invité*. Si vous avez installé un Linux en mode texte et que vous voulez le passer en mode graphique, vous devez ***d'abord*** installer les paquetages graphiques (Xorg, etc) puis ces fameuses *Additions invité*.

En effet, celles-ci vont installer des X-Windows virtuels et autres fonctionnalités, sur le système invité, à condition que les librairies X soient présentes. Installer les additions se fait en deux temps. D'abord, comme illustré [Fig.6], on monte l'ISO de ces Additions invité depuis le menu de périphériques de la VM. Cette ISO fait partie de la distribution de VirtualBox.

Ensuite on démarre la VM, puis après s'être logué en root, on saisira une suite de commandes dans cet esprit :

```
mkdir /mnt/invite
#montage de l'ISO du CD
# sur le système de fichiers
mount /dev/cdrom /mnt/invite
cd /mnt/invite
ls
./VBoxLinuxAdditions-x86.run
reboot
```

Le redémarrage de la VM est absolument requis pour que les *Additions invité* fonctionnent correctement. Moyennant quoi l'affichage sous X-Windows est désormais pleinement satisfaisant. [Fig.7]

6 CRÉER UN RÉPERTOIRE PARTAGÉ

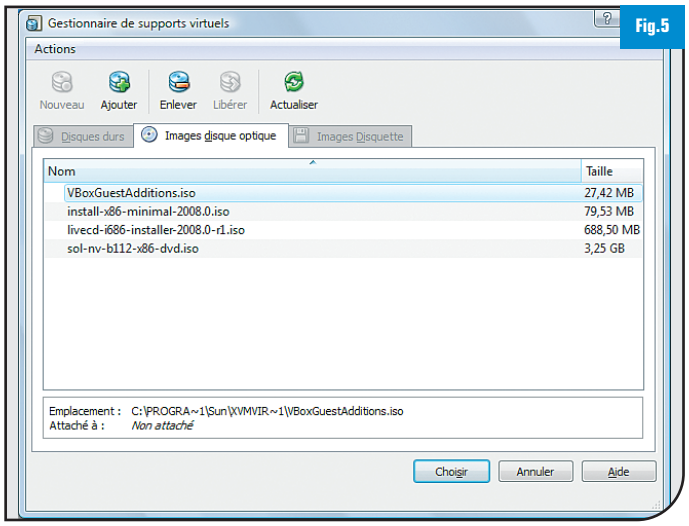
Comme le nom de VirtualBox le suggère, les VM sont autant de SandBox, de bacs à sable, et le problème de la communication entre elles et avec le monde extérieur se pose maintenant. Une première possibilité, offerte par les Additions est le répertoire partagé qui permet à une VM d'écrire dans un répertoire de la machine hôte. L'ajout d'un tel répertoire se fait depuis le menu *Périphériques* de la VM comme illustré [Fig.8]. Notre répertoire est baptisé vbox et se situe en C:\temp\vbox sur le Windows hôte. Ensuite sous Linux, ce répertoire qui est donc finalement un périphérique, doit être monté sur le système de fichier comme ceci :

```
# la première fois:
mkdir /mnt/vbox
# puis
mount -f vboxsf vbox /mnt/vbox
```

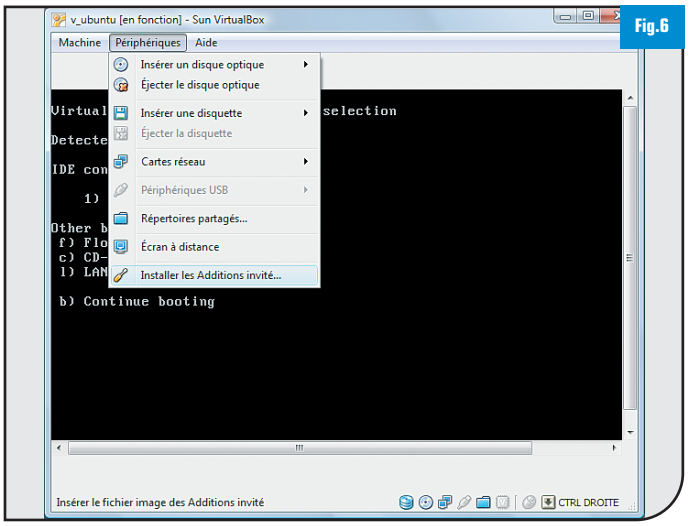
7 CONFIGURER LE NAT

Si ce répertoire partagé peut vous ôter une épine du pied dans bien des situations, un linuxien averti préférera SSH pour une communication avec la VM depuis l'extérieur. Pour cela, il faut que VirtualBox transfère les paquets réseau et donc que le NAT soit configuré. Ceci est valable pour SSH mais aussi pour tout outil réseau. Ainsi on peut souhaiter qu'une VM se comporte comme un serveur Web ou un serveur de courrier. Pour obtenir ce résultat nous allons utiliser VBoxManage, un outil en ligne de commande qui fait partie de la distribution de VirtualBox. Cet outil est plein de possibilités, toutes plus intéressantes les unes que les autres et nous invitons le lecteur à l'étudier de près. Utilisons cette commande pour que notre VM puisse se comporter comme un serveur Web :

```
VBoxManage setextradata "programmez_gentoo"
  "VBoxInternal/Devices/e1000/0/LUN#0/Config/natweb/Protocol" TCP
VBoxManage setextradata " programmez_gentoo "
  "VBoxInternal/Devices/e1000/0/LUN#0/Config/natweb/GuestPort" 80
VBoxManage setextradata " programmez_gentoo "
  "VBoxInternal/Devices/e1000/0/LUN#0/Config/natweb/HostPort" 8080
```



VirtualBox permet d'attacher des images ISO à un VM comme autant de périphériques.



Ajout de l'ISO des Additions Invité à une VM.

Nous redirigeons sur la première interface réseau de la VM (rang 0). *e1000* est le nom du driver le la carte émulée (ici Intel PRO/1000 MT). Le terme *natweb* est arbitraire. Si maintenant, Apache tournant dans l'invité vous saisissez :

```
http://localhost:8080
```

Dans le navigateur de l'hôte, vous verrez apparaître la page d'accueil de votre site virtualisé.

8 LE PROBLÈME DU DNS

Si la VM veut contacter l'extérieur, elle doit être renseignée sur l'IP d'un serveur DNS. Sous Windows hôte, il est fort probable que les configurations DHCP des cartes réseau de l'invité ne soient pas correctes en ce que concerne le DNS. Dans les FAQ sur le site de VirtualBox, on peut lire que ceci est dû au fait que Windows retourne le DNS de sa première interface configurée, qui peut ne pas être connectée. L'explication est peu convaincante car votre serviteur à rencontré le problème même avec une seule interface parfaitement opérationnelle sur l'hôte.
Pour contourner ce problème, on doit veiller à ce que le contenu du fichier /etc/resolv.conf des invités soit correct après la configuration des interfaces. Sous Ubuntu, j'ai obtenu cela en ajoutant quelques lignes dans /etc/rc.local

```
echo " " > /etc/resolv.conf
echo "#orange" >> /etc/resolv.conf
echo "nameserver 80.10.246.2" >> /etc/resolv.conf
echo "nameserver 80.10.246.129" >> /etc/resolv.conf
```

Sous Gentoo, le mieux est d'intervenir sur le script /lib/dhcpd/dhcpd-hooks/20-resolv.conf. Ou sinon reportez vous à la documentation de votre distribution préférée pour faire une manipulation équivalente.

9 CLONER DES LINUX

Installer un OS est toujours une opération très (trop) longue. Fort heureusement VirtualBox permet de les clôner :) Il n'est pas possible d'utiliser deux fois un même disque virtuel comme base à deux VM. VirtualBox vérifiera le GUID du disque et refusera l'opération. L'opération de clonage consiste alors tout simplement à demander à VBoxManage de dupliquer un disque virtuel, en lui attribuant un autre GUID. Par exemple :

```
VBoxManage clonehd C:\Users\Fred\.VirtualBox\HardDisks\v_gentoo.
vdi F:\virtual\images\v_clone_gentoo.vdi --format VDI
```

Et vous pouvez créer une nouvelle machine virtuelle sur la base de ce nouveau disque

10 LE PROBLÈME DES ADRESSES MAC SOUS LINUX

Mais l'interface réseau de votre VM sera alors totalement inopérante si le système cloné est un Linux. En effet qui dit nouvelle VM dit nouvelle adresse MAC de l'interface réseau. Or Linux va rechercher l'ancienne adresse MAC dont il garde une trace… quelque part. Deux solutions à ce problème. La première tient de la bidouille. Elle consiste à forcer l'ancienne adresse MAC dans le fichier XML décrivant la VM. Toutefois, cette bidouille qui est contre nature en regard de l'unicité des adresses MAC, ne doit être utilisée que dans des circonstances exceptionnelles. Sous Linux c'est udev qui garde une trace de l'ancienne interface, dans le fichier :

```
/etc/udev/rules.d/70-persistent-net.rules
```

Vous y voyez sans doute maintenant deux lignes: Une ligne pour une interface de nom *eth0* avec l'ancienne adresse MAC avant clônage, et une ligne avec une interface de nom *eth1* dont la nouvelle adresse MAC correspond à celle que vous trouverez dans le descriptif XML de la nouvelle VM. Vous pouvez alors configurer votre linux pour qu'il prenne en compte *eth1*, ou modifier ce fichier, en supprimant la première ligne et en donnant le nom *eth0* dans la deuxième ligne. Ou plus simplement, en détruisant carrément le fichier. Rebootez alors votre Linux virtuel. Le fichier sera de nouveau généré par udev et l'interface réseau sera opérationnelle.

11 CONCLUSION PROVISOIRE

Les manipulations décrites dans cet article vous permettront de vous amuser à l'infini avec VirtualBox et au-delà de l'amusement, vous aurez probablement l'occasion de constater que ce logiciel rend des services immenses. Mais il y a plus. VirtualBox est programmable. Pour une prochaine fois :)

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

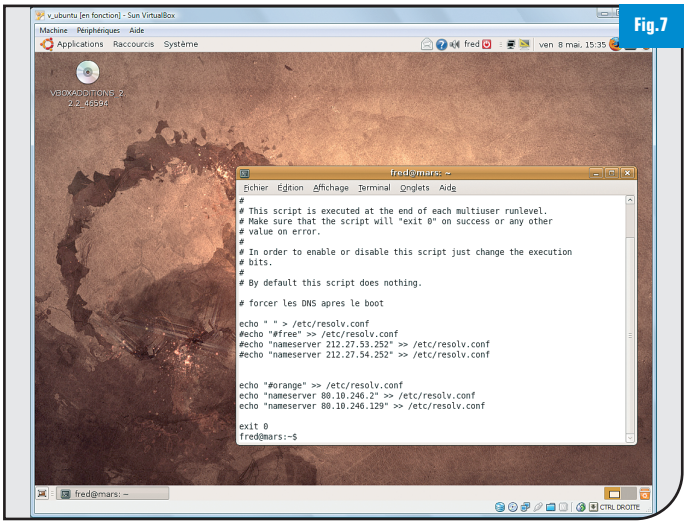


Fig.7

Ubuntu et le parfait affichage de son bureau, grâce aux Additions invité

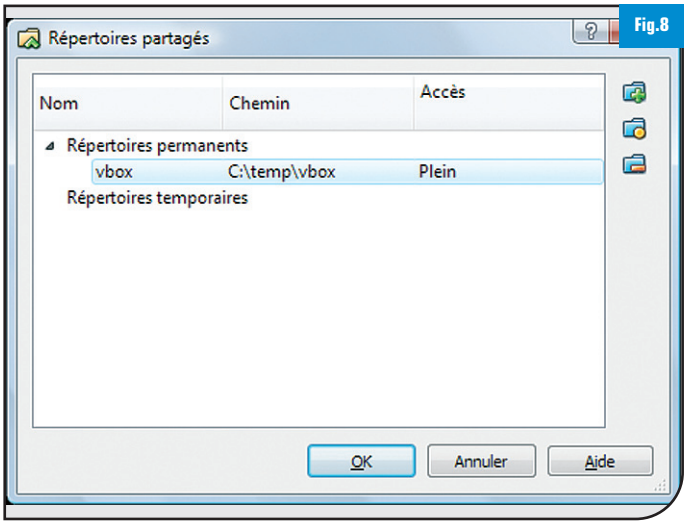
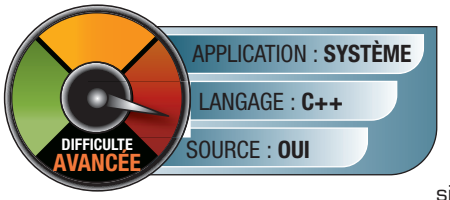


Fig.8

Définition d'un répertoire partagé entre une VM et le système hôte.

Les signaux et la programmation asynchrone sous Linux

Les signaux sous Linux sont en vaste sujet. C'est même en grande partie l'âme de la programmation UNIX. Jetons un regard simple sur ce vaste domaine et découvrons une fonctionnalité des noyaux Linux récents: **signalfd**.



La manipulation des signaux est un des aspects les plus passionnants de la programmation UNIX et donc Linux. Qu'est-ce qu'un signal ? Disons que c'est une sollicitation faite à un processus, et auquel celui-ci doit réagir immédiatement, en modifiant le cours normal de son flux d'exécution. Sur le principe, les signaux Linux c'est très simple. Dans la pratique, plein de subtilités surgissent. De plus, les sollicitations en question étant, dans l'immense majorité des cas, externes au processus, nous entrons dans le domaine de la programmation asynchrone, et du cortège d'ennuis et de bugs imprévisibles qui la suivent. Dans cet article nous allons nous intéresser à quelques bases de la manipulation des signaux, voir que ces manipulations peuvent vite devenir problématiques et comment la nouvelle fonctionnalité **signalfd**, propre au noyau Linux, facilite le travail.

1 LES BASES

Les signaux sont émis par des processus, mais le plus souvent par le noyau lui-même pour signaler, c'est le cas où jamais de le dire, des changements de conditions matérielles ou logicielles. Ainsi lorsqu'un *timer* arrive à terme, le noyau émet un signal lorsque des combinaisons de touches telles que Ctrl-C ou Ctrl-Z sont activées, le noyau émet un signal. Lorsqu'un terminal est arrêté, ses processus fils sont signalés. Lorsqu'un processus tente d'accéder à une zone mémoire illégale, il est signalé, etc. Concrètement, un signal n'est rien d'autre qu'un nombre, représenté pour le programmeur par une constante symbolique. Sous Linux il en existe 64 qu'il serait stérile de passer en revue ici. Nous renvoyons le lecteur aux nombreuses documentations existantes. Le processus destinataire d'un signal peut, avec les exceptions qui confirment la règle, soit capturer soit ignorer un signal émis à son intention. Si le signal est capturé, le flot d'exécution du processus saute dans une routine dite gestionnaire du signal, pour ensuite reprendre l'exécution au point où elle en était lors de la réception du signal. Là encore on rencontre des subtilités et cette règle comporte aussi des exceptions. Si le signal est ignoré, un traitement par défaut est appliqué.

2 TRAITEMENT CLASSIQUE

Ecrivons un programme (BasicSignalClassique.cpp sur notre site) qui capture la combinaison de touches ctrl-Z et donc qui capture le signal SIGTSTP, signal à ne pas confondre avec SIGSTOP qui, lui, ne peut être capturé.

```
#include <iostream>
using namespace std;

#include <unistd.h>
#include <signal.h>

void gestionnaire(int numero) {
  cout << "Signal recu: " << strsignal(numero) << endl;
}

int main() {
  // Mise en place du gestionnaire
  if(signal(SIGTSTP, gestionnaire) == SIG_ERR) {
    cerr << "Impossible d'installer le gestionnaire" << endl;
    return 1;
  }
  cout << "Gestionnaire en place" << endl;
  cout << "Ctrl-C pour arreter" << endl;
  // Mettre le processus en sommeil
  while(1)
    pause();

  return 0;
}
```

Sous Linux, le gestionnaire de signal est réinstallé par le système dès la fin de son exécution. Ce comportement est particulier à Linux. Avec d'autres UNIX il faut réinstaller le signal dans le gestionnaire. Avec Linux toujours, un signal est bloqué dans son gestionnaire, ce qui fait qu'il ne peut être appelé lors de sa propre exécution. Il n'en est pas forcément de même avec d'autres UNIX. Ceci dit pour pointer du doigt qu'écrire du code portable Linux/Unix est un vrai défi, le comportement des API variant sensiblement d'un système à un autre.

3 SIGNAUX ET APPELS SYSTÈMES LENTS

Que peut-on faire à l'intérieur d'un gestionnaire de signal, hormis, comme dans l'exemple précédent, imprimer sur la console l'identité du signal reçu ? Il y a sans doute beaucoup de possibilités, mais aussi avec de sérieuses limitations. Déjà un gestionnaire ne reçoit d'autres paramètres que le numéro du signal émis. On ne peut lui passer une structure de données dont le contenu serait préalablement initialisé. Un gestionnaire est donc contraint de travailler avec des variables globales et donc avec tous les problèmes de conflit que cela peut induire. Il y a d'autres subtilités. D'abord un signal peut faire échouer un appel système lent. Soit ce code :

```
#include <iostream>
using namespace std;

#include <unistd.h>
#include <signal.h>

void gestionnaire(int numero) {
 cout << "Signal recu: " << strsignal(numero) << endl;
}

int main() {
 // Mise en place du gestionnaire
 if(signal(SIGTSTP, gestionnaire) == SIG_ERR) {
   cerr << "Impossible d'installer le gestionnaire" << endl;
   return 1;
 }
 //siginterrupt(SIGTSTP, 0);

 cout << "Gestionnaire en place" << endl;
 cout << "Ctrl-C pour arreter" << endl;

 char c;
 int fd = 0 ; // stdin
 while(1) {
   read(fd, &c, sizeof(c));
   if(c == 'q') {
     cout << "Fin du programme" << endl;
     return 0;
   }
   Else {
     cout << "Echo: " << c << endl;
   }
 }

 return 0;
}
```

On nomme *appel système lent* un appel dont l'exécution est "longue". Cet appel est symbolisé dans le code ci-dessus par la fonction *read* qui lit un caractère sur le terminal. Sans doute cet exemple ne présentera pas de dysfonctionnement, mais si le même code lisait dans un socket on rencontrerait les problèmes à coup sûr. En effet, un signal peut faire échouer un appel système lent. Si un tel appel échoue, il est hors de question de le relancer dans le gestionnaire du signal. En effet, la plupart des appels systèmes Linux ne sont pas *ré-entrants* ce qui signifie qu'on ne peut pas les appeler si un appel précédent n'est pas correctement terminé. Il faut donc que ces appels soient relancés dans le corps principal du programme. Ceci est obtenu via un appel à la fonction *siginterrupt* immédiatement derrière l'installation d'un gestionnaire. (Voir la ligne de code en commentaire). Cette approche classique du travail avec les signaux n'est donc pas pleinement satisfaisante: à l'évidence, tout cela oblige à tester très méticuleusement les résultats des opérations d'entrées/sorties, ce qui ne simplifie pas la logique du code et son écriture, et nous commençons à voir qu'il peut s'avérer difficile de faire cohabiter signaux et entrées/sorties. Encore un mot avant de continuer: tout gestionnaire qui emploie un appel système, même dans de bonnes conditions, doit sauvegarder la variable *errno* et la restituer en sortie. En effet, un signal peut très bien se produire au moment où le programme s'apprête à analyser *errno* au retour d'un appel système.

4 SIGACTION, LES SIGNAUX À LA MANIÈRE POSIX

Jusqu'ici notre travail avec les signaux est assez brut de décoffrage. Les API conformes à la norme Posix permettent de travailler plus finement. Posix est un standard d'API pour système UNIX que Linux a le bon goût d'implémenter. Ce n'est pas le cas de tous les UNIXES malheureusement :(Si avec les API Posix les problèmes concernant ce qui se passe dans les gestionnaires restent entiers, du moins est-il possible de mettre ces gestionnaires en place plus finement, en spécifiant l'ensemble des signaux devant être bloqués pendant l'exécution du gestionnaire. Il est aussi possible de spécifier explicitement si les appels systèmes doivent être relancés (quoi que ce point précis ne soit pas standardisé Posix en fait) et si un gestionnaire doit être invoqué une seule fois ou réinstallé automatiquement. Voici à titre d'exemple un programme (DemoSigaction.cpp sur notre site) qui installe des gestionnaires pour les signaux SIGSTP et SIGQUIT correspondant aux combinaisons de touches Ctrl-Z et Ctrl-\ respectivement. Le gestionnaire pour SIGQUIT ne devra être exécuté qu'une fois. En outre, on demande que SIGINT (Ctrl-C) soit désactivé tant que l'on se situe dans le gestionnaire, et on veut encore qu'un signal soit masqué quand on est dans le gestionnaire de l'autre.

```
#include <iostream>
using namespace std;

#include <unistd.h>
#include <signal.h>


void gestionnaire_sigtstp(int sig) {
 cout << "Entree dans le gestionnaire SIGTSTP" << endl;
 cout << "Recu: " << strsignal(sig) << endl;
 sleep(2);
 cout << "Sortie gestionnaire SIGTSTP" << endl;
}

void gestionnaire_sigquit(int sig) {
 cout << "Entree dans le gestionnaire SIGTSTP" << endl;
 cout << "Recu: " << strsignal(sig) << endl;
 sleep(2);
 cout << "Sortie gestionnaire SIGQUIT" << endl;
}


int main() {
 struct sigaction action;

 // Configuration pour SIGTSTP
 action.sa_handler = gestionnaire_sigtstp;
 sigemptyset(&action.sa_mask);
 sigaddset(&action.sa_mask, SIGINT);
 sigaddset(&action.sa_mask, SIGQUIT);
 action.sa_flags = SA_RESTART;
```

```
  if(sigaction(SIGTSTP, &action, 0) != 0) {
    cerr << "Impossible installer gestionnaire SIGTSTP" << endl;
  }

  // Configuration pour SIGQUIT
  action.sa_handler = gestionnaire_sigquit;
  sigemptyset(&action.sa_mask);
  sigaddset(&action.sa_mask, SIGINT);
  sigaddset(&action.sa_mask, SIGTSTP);
  action.sa_flags = SA_RESTART|SA_ONESHOT;
  if(sigaction(SIGQUIT, &action, 0) != 0) {
    cerr << "Impossible installer gestionnaire SIGTSTP" << endl;
  }

  while(1)
    pause();
  return 0;
}
```

Ce code mérite d'être commenté. Nous utilisons donc dans cet exemple l'API *sigaction* en remplacement de l'API *signal* des exemples précédents. En plus du numéro de signal, *sigaction* attend un pointeur sur une structure dont le nom est également *sigaction*. Un membre de cette structure recevra le pointeur sur le gestionnaire du signal. Aucune difficulté ici. Le gestionnaire n'est pas plus évolué qu'avant. Il existe une alternative dont nous ne parlerons pas ici, mais qui de toute façon ne casse pas trois pattes à un pingouin ! Le membre *sa_mask* contient les signaux devant être masqués lors de l'exécution du gestionnaire. Attention, ce membre est une donnée opaque et l'emploi des API *sigemptyset* et *sigaddset* est incontournable. Enfin le membre *sa_flags* permet d'affiner les comportements. Ainsi c'est avec SA_RESTART que nous demandons la relance systématique des appels systèmes qui auraient pu être interrompus par un signal et c'est avec SA_ONESHOT que nous demandons que le gestionnaire de SIGQUIT ne soit exécuté qu'une fois. A la suite de quoi le comportement par défaut pour ce signal est rétabli.

5 SURVEILLER DE MULTIPLES ENTRÉES/SORTIES

Nous allons aborder la question encore en suspens d'un mariage plus harmonieux des traitements de signaux et des entrées/sorties. Avant cela, prenons le temps de voir comment il est possible d'attendre des opérations d'entrées/sorties sur plusieurs descripteurs de fichier simultanément, ce qui est un cas très fréquent avec des applications non triviales. En outre cette attente doit être limitée dans le temps. Tout ceci peut s'articuler autour de l'API select, comme ceci (DemoSelect.cpp sur le site) :

```
#include <iostream>
using namespace std;

#include <unistd.h>

int main() {
  fd_set readset;
  struct timeval temps_max;
  char c;
  int result;
  int fd = 0; // stdin

  temps_max.tv_sec = 5;
  temps_max.tv_usec = 0;

  FD_ZERO(&readset);
  FD_SET(fd, &readset);

  while(1) {
    result = select(1, &readset, 0, 0, &temps_max);
    if(result == 0) {
      cout << "Timeout -- Fin du programme" << endl;
      return 0;
    }
    if(FD_ISSET(fd, &readset)) {
      read(fd, &c, sizeof(c));
      if(c == 'q') {
        cout << c << endl;
        cout << "Fin du programme" << endl;
        return 0;
      }
      cout << c;
      temps_max.tv_sec = 5;
      temps_max.tv_usec = 0;
    }
  }
  return 0;
}
```

L'ensemble des descripteurs doit être contenu dans une structure de données *readset* (nous n'attendons que des lectures sur les descripteurs, et nous renvoyons le lecteur à la documentation de *select* pour tous les autres cas de figure). Attention, là encore cette structure de données est opaque et doit être manipulée avec les macros FD_XXX prévues à cet effet. Dans notre exemple nous ne mettons que le descripteur de *stdin* (soit la valeur 0) dans l'ensemble, mais rien n'empêche d'en mettre d'autres. Enfin, nous limitons le temps d'attente à 5 secondes. Bien remarquer que la structure *timeval* qui gère ce temps d'attente est réinitialisée à chaque lecture. Ceci est dû au comportement particulier de Linux qui met dans la structure le temps non encore écoulé à l'issue de l'opération de lecture. Les autres systèmes UNIX restaurent la valeur de départ automatiquement… Enfin, bien regarder le premier argument reçu par select. Ici nous passons 1. La valeur attendue est celle du plus grand descripteur de fichier de l'ensemble plus 1. Donc ici 0 + 1 = 1 :)

6 SIGNALFD, LE SIGNAL DEVIENT UN DESCRIPTEUR

Dans l'exemple précédent, si nous devions installer un gestionnaire celui-ci ne pourrait pas travailler avec, par exemple, des variables locales à la boucle *while*. Nous serions alors contraints à utiliser des variables globales pour aboutir à du code assez laid il faut bien le dire. Ce qui manquait à Linux jusqu'ici était la possibilité d'attendre des événements sur plusieurs objets de types différents. De la même façon que sous Windows, avec l'API *WaitForMultipleObjects*, il est possible d'attendre aussi bien le déclenchement d'un *timer* qu'une opération de lecture/écriture ou encore le basculement d'un

objet de synchronisation. La communauté Linux, toujours imaginative, a proposé plusieurs solutions pour traiter ce problème. Ce qui a parfois entraîné des discussions houleuses, événement fréquent quand des développeurs de talent défendent leurs idées. Passons sur les détails de toutes ces discussions :) Linus Torvalds, dans son rôle d'arbitre, a recommandé une solution "unixish" qui a débouché sur des API du nom de *signalfd, eventfd* et *timerfd*. Fondamentalement, avec cette approche, les signaux UNIX trouvent une correspondance avec un inode anonyme. Concrètement, cela veut dire que l'on peut attendre un signal en lisant dans un descripteur de fichier avec l'API standard *read*. Nous allons montrer un exemple avec *signalfd*, mais avant cela nous ne saurions trop encourager le lecteur à s'intéresser de près à *timerfd* qui simplifie elle aussi considérablement la programmation. Pour utiliser ces nouvelles fonctionnalités, vous devez avoir au moins un noyau 2.6.22, et une glibc 2.8 au moins. Et encore, avec celle-ci vous aurez sans doute un problème de compilation en C++ à cause d'un bug. Bug qui peut être contourné par l'infâme hack que vous verrez dans le code ci-dessous (DemoSignalfd.cpp sur le site), mais le mieux est encore d'avoir une glibc 2.9 dans laquelle le bug est normalement corrigé. Voici donc un exemple qui attend SIGTSTP en lisant dans un fichier, conjointement à l'attente d'une opération de lecture sur le canal d'entrée standard, le tout dans une limite de temps de 5 secondes.

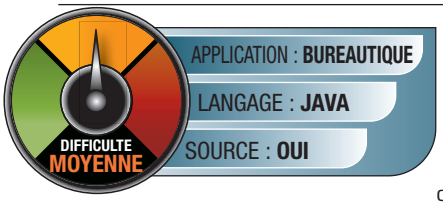
```
#include <signal.h>
#include <sys/time.h>
// horrible hack pour
// contourner un bug de
// la glic 2.8 qui empêche
// la compilation en C++
#undef __THROW
#define __THROW
#include <sys/signalfd.h>
#include <unistd.h>

#include <iostream>
using namespace std;


int main() {
 int fd_stdin = 0; // stdin vaut toujours 0
 int fd_signal;

 fd_set readset;
 sigset_t signal_mask;
 struct timeval temps_max;
 char c;
 int result;

 temps_max.tv_sec = 5;
 temps_max.tv_usec = 0;

 // Reset de l'ensemble read pour select
 FD_ZERO(&readset);
 // ajout du descripteur de stdin
 FD_SET(fd_stdin, &readset);
 // pour que signalfd fonctionne, le signal
 // classique correspondant doit être
 // masqué
 sigemptyset(&signal_mask);
 sigaddset(&signal_mask, SIGTSTP);
 if(sigprocmask(SIG_BLOCK, &signal_mask, NULL) == -1) {
   cout << "Echec de l'appel à sigprocmask -- Fin du programme"
<< endl;
   return 0;
 }

 // Obtenir un descripteur de fichier
 // réceptionner le signal
 fd_signal = signalfd(-1, &signal_mask, 0);
 if(fd_signal == -1) {
   cout << "Echec de l'appel à signalfd -- Fin du programme"
<< endl;
   return 0;
 }
 // ajout de ce descripteur à l'ensemble
 // read de select
 FD_SET(fd_signal, &readset);
 result = select(fd_signal+1, &readset, 0, 0, &temps_max);
 // Si timeout
 if(result == 0) {
   cout << "Timeout" << endl;
 }
 // Si caractère lu sur le terminal
 if(FD_ISSET(fd_stdin, &readset)) {
   read(fd_stdin, &c, sizeof(c));
   cout << "echo: " << c << endl;;
   cout << "Temps restant: " << temps_max.tv_sec << endl;
 }
 // Si ctrl-Z actionné
 if(FD_ISSET(fd_signal, &readset)) {
   struct signalfd_siginfo infos;
   read(fd_signal, &infos, sizeof(infos));
   cout << "Recu signal fd " << "Signal: "
     << strsignal(infos.ssi_signo) << endl;
   cout << "Temps restant: " << temps_max.tv_sec << endl;
 }
 cout << "Fin du programme" << endl;
 close(fd_signal);
}
```

Si tout ce qui précède a été compris, ce code ne présente pas de difficultés. Cependant, on notera que pour que le mécanisme de *signalfd* fonctionne correctement, il est impératif que le mécanisme classique des signaux soit inhibé pour les signaux avec lesquels on travaille. Nous obtenons cela avec l'API *sigprocmask*. Enfin, lorsqu'un événement est reçu par *select*, la macro FD_ISSET permet de tester duquel il s'agit. Et dans l'exemple précédent, lorsque Crtl-Z est actionné, nous pouvons y réagir dans le corps de l'application en ayant accès à d'éventuelles variables locales au lieu de nous livrer à des contorsions depuis un gestionnaire "excentré". Le revers de la médaille ? *signalfd, eventfd* et *timerfd* sont spécifiques à Linux et à lui seul. Impossible d'écrire du code UNIX portable avec eux, hélas !

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

Génération de documents OpenOffice.Org en Java

OpenOffice.Org est une suite bureautique libre qui ne cesse de monter en puissance depuis plusieurs années maintenant. Développée par Sun, elle s'accompagne d'un SDK fournissant aux développeurs l'ensemble des outils nécessaires pour interagir avec les composants. Cet article détaille sa prise en main dans un contexte Java.



Issue du rachat de la suite bureautique StarOffice par Sun à la fin des années 90, OpenOffice.org (OOo) est la version libre et équivalente de cette dernière. Utilisée par un nombre croissant de grands comptes, elle se pose en concurrent sérieux à la suite Microsoft Office. Désormais inscrite avec cette dernière dans une lutte pour devenir la suite bureautique internationale leader, elle présente l'avantage non négligeable de pouvoir tourner sur les principales plates-formes du marché (Windows, Linux, Mac OS X et Solaris).

OOo est divisé en plusieurs modules, répondant chacun à des besoins spécifiques des utilisateurs et pouvant interagir ensemble lors de la création de documents. Ces principaux modules sont les suivants :

- le traitement de texte *Writer*
- le tableur *Calc*
- le logiciel de présentations multimédias *Impress*
- l'outil de dessin vectoriel *Draw*

Grâce aux outils présents dans son kit de développement, OOo offre aux développeurs tous les outils nécessaires pour manipuler leurs documents, que ce soit en lecture ou en écriture. Ce SDK est basé sur la technologie orientée composant d'OOo qui a pour nom Universal Network Objects (UNO).

ARCHITECTURE DE UNO

UNO présente une architecture novatrice et puissante qui lui permet d'offrir une interopérabilité entre différents langages de programmation, différents modèles d'objets et différentes architectures machines. En outre, il permet l'accès à ses composants de manière locale ou bien depuis le réseau. Les composants du framework UNO sont regroupés sous forme de services qui exportent différentes interfaces. Ces dernières spécifient les fonctionnalités qu'elles proposent sans qu'on ait besoin de savoir comment elles seront implémentées. Elles sont définies dans le langage d'abstraction UNO IDL (Interface Definition Language) qui se rapproche grandement de CORBA.

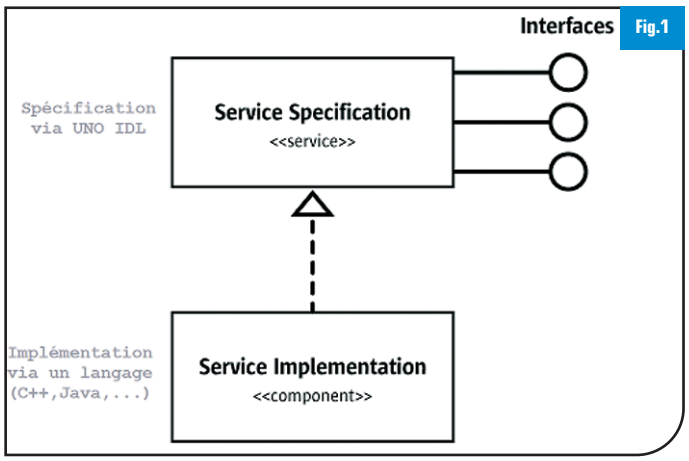


Diagramme détaillant la notion de Services.

Un gestionnaire de services permet d'instancier les différents services de UNO. La communication entre eux est ensuite rendue possible aux travers de bridges utilisant des interfaces spécifiques définies via UNO IDL. Une fois le service instancié, on doit alors demander une implémentation d'une des interfaces qu'il exporte [Fig.1]. Cette requête permet de s'abstraire du nom de l'implémentation de l'interface que l'on souhaite récupérer. Concrètement, on obtient ainsi une référence à un objet qui peut avoir été écrit dans d'autres langages et avoir été instancié sur une autre machine du réseau qui disposerait éventuellement d'une autre architecture ! C'est cet objet qui permet ensuite d'accéder aux fonctionnalités définies par l'interface demandée.

L'architecture de UNO permet donc de standardiser la communication entre différentes interfaces implémentées dans des langages différents. Les spécifications UNO détaillant le contenu de ces services sont réalisées principalement pour C++, Java et Python. Les implémentations réalisées pour ces langages permettent bien entendu d'accéder aux fonctionnalités et services UNO mais également de développer de nouveaux composants. Dans la suite de cet article, nous nous intéresserons à la mise en oeuvre de UNO via le langage Java.

INSTALLATION

La dernière mouture d'OOo est la version 3.1.1 (utilisée pour cet article). Elle est disponible à l'adresse suivante : http://fr.openoffice.org/. Une fois OOo téléchargé et installé, il nous faut récupérer un certain nombre d'archives jars nécessaires à la bonne compilation des applications Java utilisant UNO :

- *jut.jar et unoil.jar* se trouvent dans le répertoire <OOO_HOME>\Basis\program\classes
- *juh.jar, jurt.jar et ridl.jar* sont disponibles dans le répertoire <OOO_HOME>\URE\java

En outre, vous trouverez dans les sources accompagnant l'article l'archive *bootstrapconnector.jar* qui facilite le lancement d'une instance OOo depuis une application Java. Ces archives sont à mettre dans le CLASSPATH de votre application. Dans le cas qui nous concerne, à savoir la génération de documents OOo, l'installation du

SDK n'est pas nécessaire. En effet, elle le devient seulement si l'on souhaite développer des nouveaux composants OOo. Cependant, il est vivement conseillé de l'installer puisqu'elle donne accès à l'ensemble de la documentation des API OOo et qu'elle regorge d'exemples sur son utilisation avec les services UNO notamment. Elle est téléchargeable sur : http://download.openoffice.org/sdk/3.1.1_sdk.html .

CONNEXION À OOo

Le cadre de développement mis en place, il est temps de passer à la pratique. La première chose à faire dans une application utilisant UNO consiste à se connecter à une instance de OOo pour avoir accès à ses différents composants. Par défaut, il n'est pas possible de copier les jars OOo dans son projet sous peine de ne pouvoir accéder à l'exécutable OOo depuis son application, puisque l'API ne permet pas de préciser un chemin vers ce dernier. La classe *BootstrapSocketConnector* vient combler ce manque en permettant la récupération d'un contexte OOo via sa méthode statique *bootstrap* qui prend en entrée le chemin vers le répertoire des exécutables OOo. Le contexte obtenu, on accède à son gestionnaire de services afin de demander la création du service XDesktop. Ce service exporte l'interface XComponentLoader permettant la création de documents OOo que l'on va récupérer via le XDesktop.
Comme expliqué précédemment, OOo permet l'accès à ses composants en local ou bien via le réseau internet. L'obtention d'un contexte OOo et de son XComponentLoader en mode local est assez simple à réaliser comme on peut le constater :

```
public static XComponentLoader getLocalXComponentLoader() throws
 BootstrapException, Exception{
 // Récupération d'un contexte en précisant le chemin vers les
 exécutables OOo
 XComponentContext xContext = BootstrapSocketConnector.boot
strap(OOO_PROGRAM_PATH);
 // Accès au gestionnaire de services
 XMultiComponentFactory xmulticomponentfactory = xContext.get
ServiceManager();
 // Création du service XDesktop
 Object xDesktop = xmulticomponentfactory.createInstanceWith
Context("com.sun.star.frame.Desktop", xContext);
 // Récupération d'une implémentation de l'interface XComponent
Loader
 return OOoUtils.getInterface(XComponentLoader.class, xDesktop);
}
```

Une fois le service *XDesktop* obtenu, on va demander à récupérer une implémentation de l'interface *XComponentLoader* qu'il exporte via la méthode *getInterface* de la classe utilitaire *OOoUtils* créée dans le cadre de cet article :

```
public static <T> T getInterface(Class<T> clazz, Object obj){
 return (T)UnoRuntime.queryInterface(clazz, obj);
}
```

Cette méthode paramétrée utilise la classe d'environnement d'exécution UNO et simplifie la requête de demande d'implémentation d'interface. Compte tenu du nombre de requêtes de ce type qu'il est nécessaire de faire dans des applications UNO, cela constitue un gain de temps intéressant. La récupération d'un contexte OOo et de son XComponentLoader via internet est un peu plus fastidieuse puisque des problématiques réseau viennent s'ajouter et qu'il est nécessaire de préciser une URL UNO spécifique pour définir la machine distante sur laquelle on souhaite récupérer une instance OOo. Du côté de la machine distante, on a le choix entre 2 possibilités de configuration :

- Lancer OOo avec la commande *soffice.exe -accept=socket,host=<IP>,port=<Port>;urp; StarOffice.ServiceManager*
- ou Modifier la configuration de lancement de OOo qui se trouve dans le fichier *<Ooo_HOME>\Basis\share\registry\data\org\openoffice\Setup.xcu* en spécifiant cette URL UNO comme autorisée à accéder à OOo

Le code de la méthode *getRemoteXComponentLoader* vient détailler la connexion distante :

```
public static XComponentLoader getRemoteXComponentLoader(String
 url) throws BootstrapException, Exception{
 XComponentLoader xComponentLoader = null;

   if(url != null){
        // Récupération d'un contexte OOo
        XComponentContext xComponentContext = BootstrapSocket
Connector.bootstrap(OOO_PROGRAM_PATH);
        // Récupération du gestionnaire de service associé à
 ce contexte
       XMultiComponentFactory xMultiComponentFactory = xComponent
Context.getServiceManager();
        // Création du service d'accès distant
        Object objectUrlResolver = xMultiComponentFactory.create
InstanceWithContext("com.sun.star.bridge.UnoUrlResolver", xComponent
Context);
        // Récupération d'une implémentation de l'interface X
UnoUrlResolver
        XUnoUrlResolver xUrlResolver = OOoUtils.getInterface
(XUnoUrlResolver.class, objectUrlResolver);
     // Récupération d'une instance OOo sur le poste distant
 (processus OOo déjà lancé, ou création d'un nouveau)
    Object objectInitial = xUrlResolver.resolve(url);
        // Récupération du gestionnaire de composants
        xMultiComponentFactory = OOoUtils.getInterface(XMulti
ComponentFactory.class, objectInitial);
       // Récupération des propriétés du gestionnaire de composants
        XPropertySet xPropertySetMultiComponentFactory = OOo
Utils.getInterface(XPropertySet.class, xMultiComponentFactory);
        // Récupération du contexte par défaut de OOo distant
        Object objectDefaultContext = xPropertySetMultiComponent
Factory.getPropertyValue("DefaultContext");
     // Association à un contexte de composant
        xComponentContext = OOoUtils.getInterface(XComponent
Context.class, objectDefaultContext);
        // Association au service Desktop (service racine)
 avec le contexte configuré jusqu'ici
       Object xDesktop = xMultiComponentFactory.createInstance
WithContext("com.sun.star.frame.Desktop", xComponentContext);
       // Récupération d'une instance de l'implémentation de X
ComponentLoader
         xComponentLoader = OOoUtils.getInterface(XComponent
```

```
Loader.class, xDesktop);
      }
      return xComponentLoader;
}
```

Une fois l'instance du gestionnaire de services obtenue, on va utiliser le service d'accès distant *UnoUrlResolver* pour récupérer le gestionnaire de composants de l'instance OOo se trouvant sur la machine distante, précisée par l'url passée en entrée de la méthode. La suite est similaire à la méthode de récupération en mode local puisque l'on instancie le service *XDesktop* qui permet la récupération du *XComponentLoader*. Ce dernier objet permet de travailler avec des documents OOo présentés sous la forme d'objets *XComponent*.
La libération de la connexion à OOo se fait en fin de programme et utilise le service *XComponent*. A partir de ce dernier on récupère une instance de l'interface *XCloseable* sur laquelle on appelle la méthode close. Dans le cas où l'on n'arriverait pas à récupérer cette instance, on force la libération du *XComponent* en utilisant la méthode *dispose* sur une de ses instances.

CRÉATION D'UN DOCUMENT

Afin de tester nos fonctions de connexion à OOo, nous allons mettre en place un premier exemple simple qui se contente de créer un document OOo *Writer* et d'y écrire un classique Hello World ;). Pour cela, nous allons travailler avec l'instance de *XComponentLoader* que nous avons récupéré précédemment. Cet objet possède une méthode *loadComponentFromURL* qui permet de charger un document existant ou bien d'un créer un nouveau à la volée. Dans notre cas précis, l'appel à la méthode se réalise comme suit :

```
XComponent xComponent = xComponentLoader.loadComponentFromURL
("private:factory/swriter","_blank", 0, prop);
```

Les 2 premiers paramètres permettent d'indiquer que l'on souhaite créer une instance d'un nouveau document *Writer*. Quant au dernier, il s'agit d'un tableau de *PropertyValue* qui spécifie certaines propriétés de ce nouveau document. Ici, on l'utilise pour positionner la propriété *Hidden à true* afin de préciser que l'on souhaite que l'instance OOo lancée reste cachée sur le poste client. L'instance du *XComponent* renvoyée permet de travailler avec le document Writer nouvellement créé. L'écriture d'un texte tient alors en quelques lignes :

```
// Récupération de l'interface XTextDocument associée au XComponent
XTextDocument xTextDocument = OOoUtils.getInterface(XTextDocument
.class, xComponent);
// Accès à la partie texte du document
XText xText = xTextDocument.getText();
// Positionnement à la fin de la zone de texte
XTextRange xTextRange = xText.getEnd();
// Ecriture de notre Hello World ! ;)
xTextRange.setString("Hello World !\n");
```

SAUVEGARDE D'UN DOCUMENT

Le travail sur le document Writer terminé, il est temps de passer à sa sauvegarde. Pour cela, nous utilisons l'instance de XComponent qui le représente. Les possibilités offertes par UNO en termes de sauvegarde de documents sont identiques à celles proposées par la suite OOo. Ainsi, nous pouvons exporter notre document dans l'ensemble des formats d'écriture supportés par OOo, ceci incluant également le format PDF ou le format DOC de Microsoft. Le choix du format de sauvegarde n'a que peu d'impact sur le code produit puisqu'il suffit de modifier une simple propriété textuelle lors de l'appel à la méthode de sauvegarde pour passer d'un format à un autre ! Le code suivant détaille la sauvegarde d'un document :

```
// Récupération d'une instance de l'interface de stockage
XStorable xStorable = OOoUtils.getInterface(XStorable.class,
xComponent);
// Configuration des propriétés pour la sauvegarde
PropertyValue[] propertyValue = new PropertyValue[1];
propertyValue[0] = new PropertyValue();
propertyValue[0].Name = "FilterName";
propertyValue[0].Value = "StarOffice XML (Writer)";
// Sauvegarde à l'URL passée en entrée
xStorable.storeAsURL("file:///I:/helloWorld.odt", propertyValue);
```

Comme on peut le voir, la propriété *FilterName* permet d'indiquer que l'on souhaite enregistrer le document au format ODT. Pour un enregistrement au format PDF, on aurait du utiliser le filtre "writer_pdf_Export" et utiliser la méthode *storeToURL* de l'objet *XStorable* avec les mêmes arguments en entrée.

PROGRAMME COMPLET

Notre programme complet reprend les différents bouts de codes présentés précédemment et les assemble afin de générer notre document *Writer helloWorld.odt* :

```
public static void main(String[] args) {
  XComponentLoader xComponentLoader = null;
  XComponent xComponent = null;
  try {
    // Récupération d'une instance locale OOo
    xComponentLoader = OOoUtils.getLocalXComponentLoader();
    if(xComponentLoader != null){
      xComponent = OOoUtils.loadNewComponent(xComponentLoader,
DocumentType.WRITER, true);
    }

  }catch(BootstrapException bse) {
    System.out.println("Erreur à la récupération du contexte :
" + bse.getMessage());
  }catch(Exception exc){
    System.out.println("Erreur au chargement du composant :
" + exc.getMessage());
  }

  // Travail sur le document
  if(xComponent != null){
    printHelloWorld(xComponent);
    // Sauvegarde
    boolean save = OOoUtils.saveDocument("I:/", "helloWorld",
xComponent, SaveFormat.PDF, true);
    System.out.println("La sauvegarde a " + ((save) ? "réussi"
: "échoué"));
    // Fermeture
```

```
    OOoUtils.closeOrDisposeInstance(xComponent);
    System.exit(0);
  }
}
```

La création du programme complet générant notre document *Writer* se fait donc avec les étapes suivantes :

1. Connexion à OOo et récupération d'un *XComponentLoader*
2. Création d'un nouveau document à partir du *XComponentLoader*
3. Travail sur le document (via le *XComponent* qui le représente) en l'occurrence ici insertion du texte Hello World
4. Sauvegarde du document à une URL donnée et dans un format donné via le *XStorable* associé au *XComponent*
5. Fermeture de la connexion à OOo

Le lecteur aura noté que les différents codes ont été placés au sein de différentes méthodes utilitaires de la classe *OOoUtils*. L'exécution de ce programme permet d'obtenir le document présenté à la [Fig.2].

POUR ALLER PLUS LOIN ...

Cet exemple basique aura permis de mettre en avant les fondements de la programmation d'applications utilisant le framework UNO. Bien entendu, ces possibilités ne se limitent pas seulement à la simple création de documents *Writer* contenant du texte. Ainsi, il est possible de créér des documents *Calc, Impress* ou *Draw* mais dans tous les cas seule la partie de manipulation du document change alors que les parties de connexion et de sauvegarde restent sensiblement identiques. En outre, le lecteur trouvera dans les sources de l'article des exemples plus poussés avec notamment :

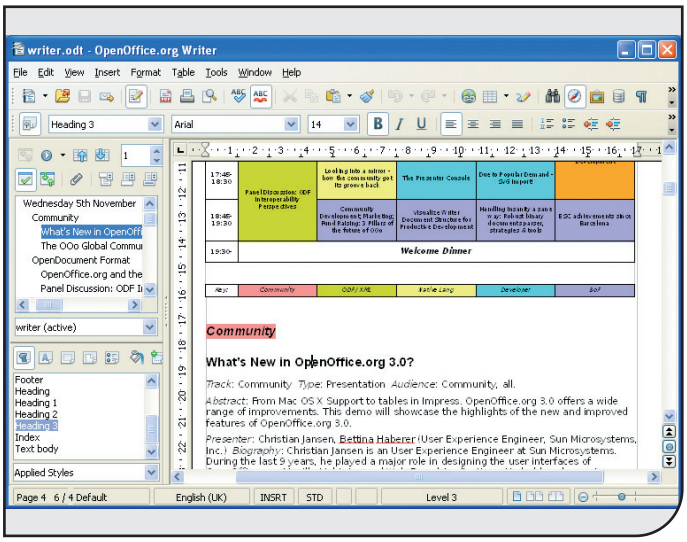
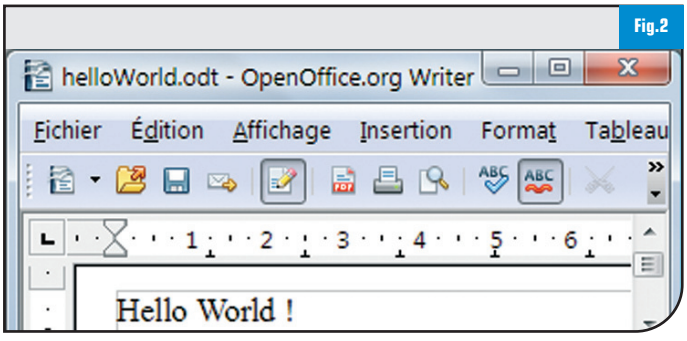


Fig.2



Document Writer helloWorld.odt

- la lecture d'un document *Writer* existant qu'il soit simplement textuel ou bien composé de tableaux
- la création d'un document *Writer* complexe contenant une image, des formes graphiques, un tableau mais également un graphique *Draw* [Fig.3]
- la création de documents *Calc* et *Draw*

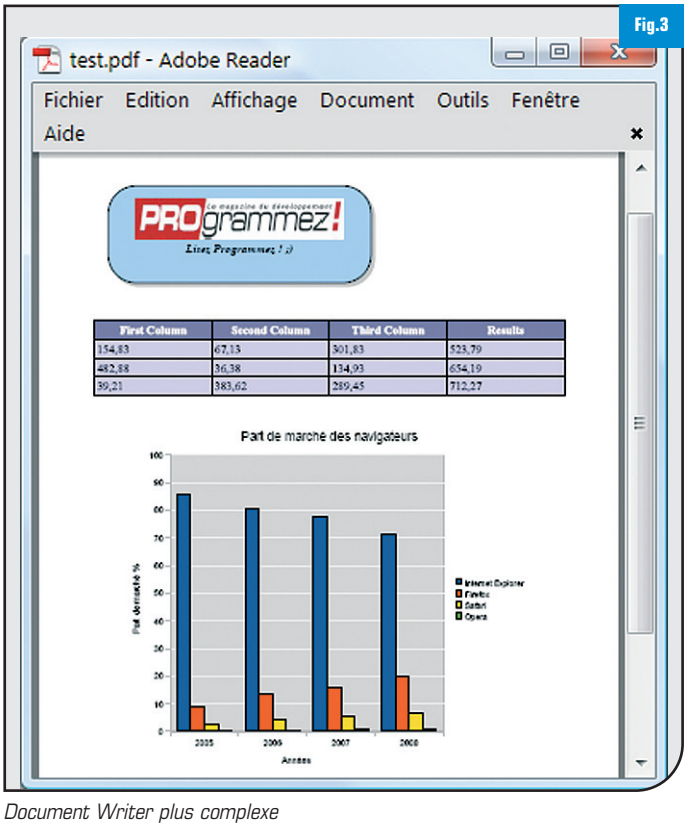
Tout ceci s'accompagnant bien entendu de l'ensemble des classes et méthodes créées pour faciliter l'utilisation d'OOo, dont certaines ont été présentées au cours de l'article. Enfin, il est à noter que la documentation fournie avec le SDK d'OOo se révèle vite indispensable. Outre le fait qu'elle contient la description de l'ensemble des composants UNO, elle regorge d'exemples de codes qui vous aideront à réaliser des documents OOo des plus poussés.

CONCLUSION

Au travers d'un exemple simple, cet article n'aura fait qu'effleurer les nombreuses possibilités offertes par le framework UNO dans la manipulation de documents OOo. Cependant, il aura permis de poser les bases de son utilisation pour démarrer plus facilement le développement d'applications plus complexes par la suite. Ce que la documentation amenée par le SDK, bien que très abondante, ne permet pas facilement puisque l'on a vite fait de s'y perdre. Heureusement, on arrive quand même assez souvent à y trouver son bonheur et dans le cas contraire, la communauté OOo est suffisamment importante et active, ce qui permet de trouver la solution à bon nombre de problèmes. Enfin, dans un contexte industriel où la production de documents détaillés à partir d'applications devient un enjeu majeur, le choix des outils d'OOo se révèle plus que pertinent.

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

Fig.3

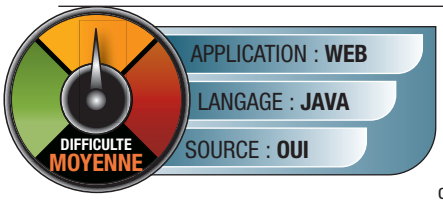


Document Writer plus complexe

GWT : comment ça marche ?

3e partie

Lors de précédents articles, nous avons pu découvrir GWT à travers une Application Internet Riche (RIA) qui nous permettait d'afficher des photos via le service internet de Picasa. Nous allons maintenant lui ajouter sa propre gestion de photos. Profitons de cette nouvelle fonctionnalité pour nous pencher plus précisément sur la partie Serveur avec Google App Engine. GWT fonctionne bien évidemment sur n'importe quel serveur Java, mais Google App Engine travaille particulièrement bien avec GWT grâce au plug-in de Google.



PETIT RAPPEL SUR GOOGLE APP ENGINE(GAE)

Auparavant exclusivement dédié au langage Python, **Google App Engine, depuis** avril 2009, permet l'hébergement d'applications Java/JEE. Cette solution de Cloud Computing est conçue comme *« platform as a service »* : Google fournit l'infrastructure complète, ainsi que l'environnement pour héberger l'application. Ces mêmes infrastructures qui hébergent Gmail, Google Calendar, Google Finance, GoogleEarth… et qui sont réputées pour leur qualité de service et leur capacité à monter en charge. App Engine propose ainsi de nombreux services, dont notamment un système de base de données, appelé « datastore », basé sur Google Big Table. La gestion de la persistance est réalisée par l'ORM *Datanucleus,* qui supporte les implémentations JDO et JPA.

GOOGLE APP ENGINE EST SIMPLE

En reprenant notre application GWT, il suffit d'indiquer que l'on souhaite utiliser Google App Engine et quelques clics après, on retrouve notre application déployée sur l'App Engine. Tout cela grâce au plug-in Eclipse [Fig.1 et 2].

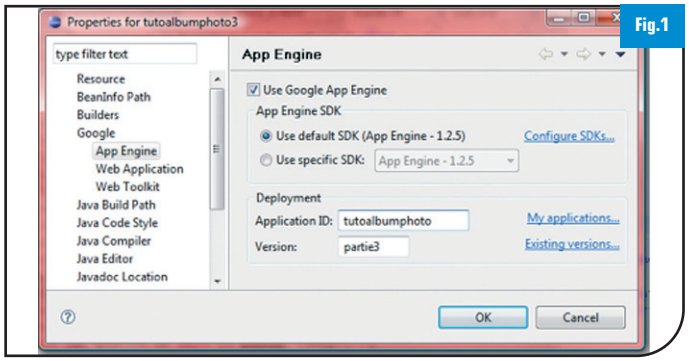


Fig.1

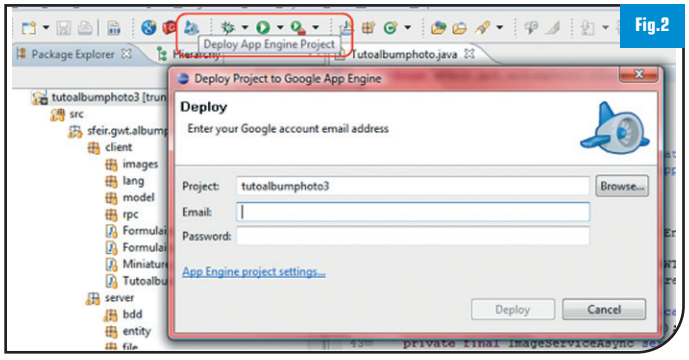


Fig.2

ET EN PLUS, L'UTILISATION DE BASE EST GRATUITE

Chaque application de Google App Engine dispose de suffisamment de processeurs, de bande passante et de stockage afin d'offrir gratuitement environ 5 millions de consultations de pages par mois. Au-delà, il est possible de louer des ressources supplémentaires à des prix compétitifs lorsque vous en avez besoin. Vous payez alors uniquement les ressources que vous utilisez.

Les limitations qu'apporte App Engine, permettent de garantir une montée en charge rapide de l'application. En d'autres termes, l'offre de Google pourrait se résumer à :

« Développez une application selon notre contrat, on vous garantit en retour qu'elle pourra être hébergée sur notre infrastructure que nous vous louerons à l'usage ».

PROGRAMMONS !

Jusqu'à présent, nous parsions un flux XML retourné par Picasa après une requête pour rechercher des photos. Ajoutons maintenant la possibilité à l'application de stocker nos photos directement sur l'App Engine. Pour cela, App Engine met à notre disposition l'API JDO qui permet de mapper des objets avec la base de données.

La base de données Big Table n'est absolument pas une base de données relationnelle comme nous avons l'habitude de voir, cela implique de nombreuses contraintes de conception. Big Table peut être considérée comme une « hashmap » d'objets (table de clés/valeurs). Pour revenir à l'application, créons notre objet persistant. Il suffit pour cela, dans un premier temps, de créer un bean java et d'y ajouter des annotations : @PersistentCapable sur la classe et @Persistent sur les propriétés à sauvegarder. Il nous faut également une clé primaire qui garantit l'unicité de l'objet lors de son enregistrement : la clé peut être soit un entier, soit un identifiant alphabétique ou alors le type « Key » d'App Engine, qui peut également servir à créer une sorte de clé étrangère.

```
@PersistenceCapable(identityType=IdentityType.APPLICATION)
public class Photo {
    @PrimaryKey
    @Persistent(valueStrategy = IdGeneratorStrategy.IDENTITY)
    private Key cle;
    @Persistent
    private String titre;
    @Persistent
    private Blob contenu;
//Accesseurs
}
```

Autre particularité de l'App Engine, le fait qu'il ne soit pas possible de créer des fichiers, pour des questions de performances, nous oblige à stocker les images dans la base de données.
Pour cela, Big Table introduit un type « Blob » permettant de stocker un tableau de bytes qui représente le contenu de l'image.
Procédons maintenant à la persistance des objets grâce au PersistenceManager.

```
public Photo creer(String titre, byte[] contenu){
  Photo images = new Photo();
  images.setTitre(titre);
  images.setContenu(new Blob(contenu));
  PersistenceManager persistenceManager = getPersistenceManager();
  return persistenceManager.makePersistent(images);
}
```

La création du PersistenceManager se fait comme ceci :

```
private static PersistenceManagerFactory pmf;
private static PersistenceManager pm;
public static synchronized PersistenceManager getPersistence
Manager(){
    if(pmf == null)
      pmf = JDOHelper.getPersistenceManagerFactory("nontransac
tional-datasource");
    if(pm == null || pm.isClosed()){
      pm = pmf.getPersistenceManager();
    }
    return pm;
}
```

Récupérons ensuite la liste des photos en exécutant une requête, en triant le résultat et en sélectionnant la fourchette :

```
public List<Photo> liste(int page){
    PersistenceManager persistenceManager = getPersistenceManager();
    int premiereImages = (page - 1) * PHOTOS_PAR_PAGE;
    Query requete = persistenceManager.newQuery(Photo.class);
    requete.setOrdering("titre ASC");
    requete.setRange(premiereImages, premiereImages + PHOTOS
_PAR_PAGE);
    return (List<Photo>) requete.execute();
}
```

Récupérons un élément grâce à son identifiant :

```
public Photo rechercher(Key cle){
    PersistenceManager persistenceManager = getPersistenceManager();
    return persistenceManager.getObjectById(Photo.class, cle);
}
```

IL faut maintenant créer une servlet qui affiche une image en lui donnant sa clé sous forme d'une chaîne de caractères. Les méthodes *KeyFactory.keyToString() et KeyFactory.stringToKey()* permettent de convertir une clé en chaîne de caractères.

```
protected void doGet(HttpServletRequest req, HttpServlet
Response resp) throws ServletException, IOException{
  ImageDbb service = new ImageDbb();
  String cle = req.getParameter("cle");
  if (cle == null) {
    resp.sendError(HttpServletResponse.SC_BAD_REQUEST, "La clé
de l'image est requise");
  } else {
   try {
    Key imageCle = KeyFactory.stringToKey(cle);
    Photo image = service.rechercher(imageCle);
    resp.setContentType("image/jpeg");
    resp.getOutputStream().write(image.getContenu().getBytes());
   } catch (Exception e) {
     resp.sendError(HttpServletResponse.SC_NOT_FOUND, e.get
Message());
   }
  }
}
```

Ajoutons une fonctionnalité qui prend en charge le paramètre « taille » dans l'URL, ainsi s'il est défini à « mini » cela créera une miniature de l'image. Pour cela, nous utilisons l'API Images de App Engine qui permet de réaliser des modifications de base sur les images.

```
String taille = req.getParameter("taille");
if (taille != null && taille.equals("mini")){
  ImagesService imagesService = ImagesServiceFactory.getImages
Service();
  Image imageOriginale = ImagesServiceFactory.makeImage(image
.getContenu().getBytes());
  Transform resize = ImagesServiceFactory.makeResize(200, 300);
  Image imageMiniature = imagesService.applyTransform(resize,
imageOriginale);
  resp.getOutputStream().write(imageMiniature.getImageData());
}
```

Réalisons maintenant un service RPC qui communique avec l'application pour renvoyer la liste des images. Il est nécessaire de convertir l'objet venant de la base de données en objet GWT. En effet à cause des métadonnées ajoutées par Datanucleus, il n'est pas possible d'envoyer cet objet à GWT, et, de même, l'envoi des données binaires représentant l'image ne pourra pas être affiché par GWT. Nous avions vu dans l'article précédent comment créer un service RPC, nous vous laissons donc le soin de le refaire. Voici juste le code de la partie serveur :

```
public List<Photographie> getImages(int page) {
  ImageDbb service = new ImageDbb();
  List<Photo> liste = service.liste(page);
  List<Photographie> listePhoto = new ArrayList<Photographie>();
  for (Photo image : liste) {
   Photographie photo = new Photographie();
   photo.setPhotoTitre(image.getTitre());
   String cle = KeyFactory.keyToString(image.getCle());
   photo.setPhotoMiniatureUrl("/image/?taille=mini&cle="+cle);
   photo.setPhotoUrl("/image/?cle="+cle);
   listePhoto.add(photo);
  }
  return listePhoto;
}
```

À ce stade, il suffit de modifier le client GWT pour qu'il puisse appeler ce service et non le service Picasa du précédent article.

Alors comment fait-on pour que notre service puisse envoyer des photos ? Il faut pour cela modifier la fenêtre d'ajout d'image pour y intégrer un FileUpload qui permettra de parcourir les fichiers de notre ordinateur pour choisir le fichier à envoyer. Seulement, il n'est plus possible de passer par un service RPC pour envoyer la photo, nous sommes obligés d'utiliser un formulaire HTML et d'envoyer comme une application web normale notre image vers une servlet. C'est sur ce point que GWT nous apporte une petite astuce, alors restez attentifs ! En effet, le formulaire envoie à travers une Iframe invisible, c'est pour cette raison que l'utilisateur ne voit pas sa page changée. Il faut pour cela mettre nos champs de saisies dans un FormPanel et changer le type d'encodage pour « encoding multipart » et utiliser la méthode post. Il faut aussi donner des noms à tous les champs HTML pour y avoir accès dans notre servlet :

```
public class FormulaireAjout extends DialogBox {
    private MesMessages mesMessages = GWT.create(MesMessages.
class);
    public FileUpload saisieImage = new FileUpload();
    public TextBox saisieTitre = new TextBox();
    private FormPanel form = new FormPanel();

    public FormulaireAjout() {
      setText(mesMessages.texteAjout());
      saisieTitre.setName("titre");
      saisieImage.setName("image");
      Grid grid = new Grid(3, 2);
      grid.setWidget(0, 0, new Label(mesMessages.url()));
      grid.setWidget(0, 1, saisieImage);
      grid.setWidget(1, 0, new Label(mesMessages.titre()));
      grid.setWidget(1, 1, saisieTitre);
     Button boutonAjouter = new Button(mesMessages.ajouter());
      grid.setWidget(2, 0, boutonAjouter);
      boutonAjouter.addClickHandler(new ClickHandler(){
        public void onClick(ClickEvent event){
          FormulaireAjout.this.form.submit();
        }
      });
      form.setAction("/image/");
      form.setMethod(FormPanel.METHOD_POST);
      form.setEncoding(FormPanel.ENCODING_MULTIPART);
      form.add(grid);
      form.addSubmitCompleteHandler(new SubmitCompleteHandler() {
        public void onSubmitComplete(SubmitCompleteEvent event) {
          FormulaireAjout.this.hide(true);
        }
      });
      add(form);
      center();
    }
}
```

Finalement, voyons comment ajouter notre image dans la Servlet à l'intérieur du *doPost()*

```
ImageDbb service = new ImageDbb();
if (ServletFileUpload.isMultipartContent(req)) {
   ServletFileUpload servletFileUpload = new ServletFileUpload
(new AppEngineFileItemFactory());
    try {
     List<FileItem> fileItems = servletFileUpload.parseRequest
(req);
      String titre = null;
      byte[] contenu = null;
      for (FileItem fileItem : fileItems) {
        if (!fileItem.isFormField()) {
        contenu = IOUtils.toByteArray(fileItem.getInputStream());
        } else {
          titre = fileItem.getString();
        }
      }
      if (titre == null || contenu == null) {
       resp.getOutputStream().print("Titre ou Fichier vide");
        return;
      }
      service.creer(titre, contenu);
      resp.getOutputStream().print("OK");
    } catch (Exception e) {
      System.out.println(e);
     resp.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
 e.getLocalizedMessage());
    }
}
```

Grâce à la bibliothèque *commons-fileupload,* on peut récupérer notre fichier et l'ajouter dans la base de données avec le titre donné dans le formulaire. Enfin, il suffit de rafraîchir la liste des images lorsque la fenêtre est fermée en téléchargeant à nouveau la nouvelle liste depuis le service RPC.

La réalisation d'un projet qui puisse monter en charge facilement, est simplifiée par Google App Engine. Seulement, il ne fait pas tout, c'est à nous d'optimiser notre application pour qu'elle n'atteigne pas tous les quotas, sinon on paye ! ;-) Pour cela, on doit se servir du « memcache » pour y mettre les informations souvent utilisées, faire des traitements en arrière-plan avec les « batchs », recourir à la « task queue » pour lancer nos tâches plus tard, etc. Cela demande un gros travail et peut-être de revoir tout son code, y compris la base de données, en diminuant le nombre d'entités et de ne charger que ce dont on a besoin.

DÉPLOYONS !

Comme rappelé au début, Google App Engine est simple, il n'est donc pas nécessaire que l'on vous montre comment faire, n'est-ce pas ? Toutefois, n'oubliez pas de demander un compte Google App Engine. Pour finir, nous vous laissons le plaisir d'ajouter la possibilité de supprimer les photos de la base de données, comme exercice.

Pour la dernière partie de cette série d'articles, nous verrons quelles sont les nouveautés de la prochaine version de GWT : GWT 2.0. Nous les mettrons ainsi en œuvre sur notre projet.

Démonstration : http://partie3.latest.tutoalbumphoto.appspot.com/

Site : http://code.google.com/appengine/

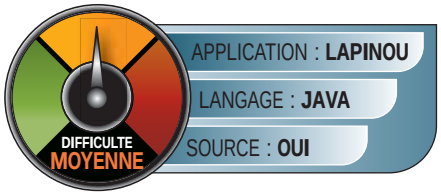
Sources : http://code.google.com/p/tutoalbumphoto/

Blog : http://www.insideit.fr

■ Christophe **Phu** et Patrice **de Saint Steban**
Ingénieurs d'étude et Développement - Direction Technique de SFEIR

Faites parler votre lapin avec Android

Le Nabaztag est un objet électronique décoratif d'utilité quotidienne. En utilisant votre connexion internet via son interface wifi intégré, il vous offre de multiples capacités telles que donner l'heure, indiquer la météo, les cours de la bourse, vous faire écouter la radio, interagir avec des objets réels et bien plus encore. Une présentation illustrée et beaucoup plus détaillée se trouve sur le site officiel http://www.nabaztag.com/.



Apres avoir déballé et convenablement installé votre lapin selon les instructions fournies dans le mode d'emploi, on peut s'attaquer à sa configuration. Rendez-vous sur le site http://my.violet.net/. Authentifiez-vous, sélectionnez le lapin dans le panel nommé « Mes Choses » puis cliquez sur l'onglet « Ecosystème ». Cochez la case « Recevoir les services et applications de l'écosystème » puis notez les numéros de « série » et « token » qui s'affichent juste en dessous, ils vous seront nécessaires pour utiliser l'API.

PRÉSENTATION DE L'API

En plus de toutes les fonctions disponibles depuis l'interface web du Nabaztag, vous avez la possibilité de programmer vos propres applications pour interagir avec votre lapin grâce à l'API. Cette bibliothèque de fonctions en ligne vous permet d'effectuer les opérations suivantes :

1 - Envoyer un message au lapin
2 - Envoyer une chorégraphie
3 - Récupérer et définir la position des oreilles
4 - Ecouter un mp3 ou une radio en streaming
5 - Récupérer des informations sur le statut du lapin.

La documentation officielle, bien que plutôt sommaire, fournit des exemples utiles qui vous aideront à forger vos requêtes et débugger votre code http://doc.nabaztag.com/api/home.html. Vous découvrirez que toutes les fonctions s'exécutent en appelant une url avec les paramètres adéquats. Il est théoriquement possible de contrôler le Nabaztag depuis n'importe quel appareil capable d'effectuer une requête HTTP.

PROGRAMMEZ VOTRE LAPINOU

Maintenant que nous savons de quoi il retourne, nous pouvons commencer à programmer. Ouvrez Eclipse et créez un nouveau projet Android. Créez ensuite une classe nommée « Nabaztag », ce sera notre objet principal pour la gestion des requêtes http et le traitement des réponses. Afin d'envoyer et recevoir des données depuis notre applications Android, nous allons devoir ajouter un attribut dans le « AndroidManifest.xml ». Ouvrez-le puis ajoutez le tag suivant :

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

Pour tester l'accès à internet, nous pouvons tout de suite lancer un navigateur avec une url de notre choix, si la page s'affiche sans erreur, alors notre application est convenablement configurée. Depuis l'activité principale j'exécute le code suivant :

```
Uri uri = Uri.parse("http://www.androlib.com/");
Intent intent = new Intent(Intent.ACTION_VIEW, uri);
context.startActivity(intent);
```

Puis dans notre classe « Nabaztag » ajoutons le constructeur et les variables appropriées :

```
public final class Nabaztag {

  // Constantes
  private static final String    APIURL =
     "http://api.nabaztag.com/vl/FR/api.jsp";

  // Variables privées
  private DefaultHttpClient    http;
  private DocumentBuilderFactory dbf;
  private DocumentBuilder      db;
  private Context        mContext;

  // Variables privées partagées
  private static String      mSerial;
  private static String      mToken;

  // Constructeur
  public Nabaztag(Context context){
    // Garde le contexte pour envoyer des Toasts
    mContext = context;
    // Récupère les paramètres Serial + Token
    nabUpdateSerialAndToken(context);
  }

}
```

Le compilateur devrait à présent vous demander de lui fournir les références relatives aux objets que nous venons de définir. Pour cela, depuis Eclipse appuyez sur « CTRL+SHIFT+O » ou ajoutez le code suivant :

```
import java.io.InputStream;
import java.net.URLEncoder;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

import android.content.Context;
import android.content.SharedPreferences;
import android.preference.PreferenceManager;
import android.widget.Toast;
```

Nous avons posé les bases mais nous n'avons pas encore défini la fonction pour récupérer les paramètres de « numéro de serie » et « token » indispensables à l'API. Le code de cette fonction tient en trois lignes :

```
// Met à jour les options
public static void nabUpdateSerialAndToken(Context context){
  // Récupère une instance du manager de préférence
  SharedPreferences pref =
    PreferenceManager.getDefaultSharedPreferences(context);
  // Récupère les préférences
  mSerial = pref.getString("serial", "");
  mToken  = pref.getString("token" , "");
}
```

Nous pouvons dès à présent attaquer le code permettant d'effectuer une requête http et récupérer son flux de réponse. Il existe plusieurs solutions pour envoyer des requêtes, dans notre exemple, nous allons profiter de la bibliothèque DOM contenue dans le framework Android. La fonction brute ressemblera à ceci :

```
// Envoi une requête HTTP
private Element sendXMLRequest(String uri) throws Exception{

  // Crée et envoie la requête Http
  if(http == null) http = new DefaultHttpClient();
  HttpGet request = new HttpGet(uri);
  response = http.execute(request);
  InputStream is = response.getEntity().getContent();

  // Crée le document Xml depuis le flux
  if(dbf == null) dbf = DocumentBuilderFactory.newInstance();
  if(db  == null) db = dbf.newDocumentBuilder();
  Document doc = db.parse(is);
  Element e  = doc.getDocumentElement();
  e.normalize();

  // Renvoi le document xml
  return e;
```


```
}
```

Notre objet « Nabaztag » contient maintenant toutes les bases, ajoutons-y les procédures relatives à l'API afin de contrôler notre lapin. La première fonction servira à réveiller le « Nabaztag ». Comme expliqué dans la documentation en ligne, l'url de l'API ressemblera a ceci :

```
http://api.nabaztag.com/vl/FR/api.jsp?sn=XXXXXXXXX&token=YYYY
Y&action=14
```

Le fichier de réponse xml renvoyé par le serveur quant à lui ressemblera à ceci :

```
<?xml version="1.0" encoding="UTF-8"?>
  <rsp>
    <message>COMMANDSENT</message>
    <comment>You rabbit will change status</comment>
  </rsp>
```

Il est donc assez facile d'en déduire le code de réponse :

```
// Réveil le Nabaztag
public void nabWakeup(){
  try{
    String  url  =  APIURL+"?sn="+mSerial+"&token="+mToken+"&
action=14";
    Element root = sendXMLRequest(url);
    String nabMessage  =  root.getElementsByTagName("message")
.item(0).getFirstChild().getNodeValue();
    String nabComment  =  root.getElementsByTagName("comment")
.item(0).getFirstChild().getNodeValue();
    Toast.makeText(mContext, nabMessage+"\n"+nabComment, Toast
.LENGTH_LONG).show();
  }catch(Exception e){
    e.printStackTrace();
```

DÉVELOPPEZ VOTRE SAVOIR-FAIRE

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

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

Choisissez votre formule

- **Abonnement 1 an au magazine : 49 €** (au lieu de 65,45 € tarif au numéro) *Tarif France métropolitaine*
- **Abonnement Intégral : 1 an au magazine + archives sur Internet et PDF : 59 €** *Tarif France métropolitaine*
- **Abonnement PDF / 1 an : 30 €** - *Tarif unique*
  Inscription et paiement exclusivement en ligne
  www.programmez.com
- **Abonnement Etudiant : 1 an au magazine : 39 €**
  (au lieu de 65,45 € tarif au numéro) *Offre France métropolitaine*

*11 numéros par an : 49 €**

Economisez 16,45 €*

**Tarif France métropolitaine*

+ Abonnement INTÉGRAL

ACCÈS ILLIMITÉ aux ARCHIVES du MAGAZINE pour 0,84€ par mois !

Cette option est réservée aux abonnés pour 1 an au magazine, quel que soit le type d'abonnement (Standard, Numérique, Etudiant). Le prix de leur abonnement normal est majoré de 10 € (prix identique pour toutes zones géographiques). Pendant la durée de leur abonnement, ils ont ainsi accès, en supplément, à tous les anciens numéros et articles/dossiers parus.

OUI, *je m'abonne* ***Vous pouvez vous abonner en ligne et trouver tous les tarifs*** www.programmez.com

PROGRAMMEZ

☐ **Abonnement 1 an au magazine : 49 €** (au lieu de 65,45 € tarif au numéro) *Tarif France métropolitaine*

☐ **Abonnement Intégral : 1 an au magazine + archives sur Internet et PDF : 59 €** *Tarif France métropolitaine*

☐ **Abonnement Etudiant : 1 an au magazine : 39 €** (joindre copie carte étudiant) *Offre France métropolitaine*

☐ M. ☐ Mme ☐ Mlle Entreprise : ..... Fonction : .....

Nom : ..... Prénom : .....

Adresse : .....

Code postal : ..... Ville : .....

Tél : ..... E-mail : .....

☐ Je joins mon règlement par chèque à l'ordre de Programmez ! ☐ Je souhaite régler à réception de facture

A remplir et retourner sous enveloppe affranchie à :
Programmez ! - Service Abonnements - 22 rue René Boulanger - 75472 Paris Cedex 10.

abonnements.programmez@groupe-gli.com

Offre limitée, valable jusqu'au 30 novembre 2009

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 124

```
    Toast.makeText(mContext, e.getMessage(), Toast.LENGTH_LONG);
  }
}
```

L'usage voudrait que l'on teste la validité de chaque objet avant d'en extraire les données, néanmoins dans un souci de clarté nous omettrons volontairement ces tests et ajouterons une clause « Try / Catch » globale.
Pour tester notre code, ajoutons un bouton à notre programme principal et dans le code de gestion ajoutons le code suivant :

```
Nabaztag nab = new Nabaztag(this);
nab.nabWakeup();
```

Si les paramètres de l'application « Numéro de série » et « Token » sont bien configurés et que le téléphone et le Nabaztag sont reliés à l'internet, le clic sur le bouton devrait afficher un message semblable à ceci :

```
COMMANDSENT
You rabbi twill change status
```

Une deuxième fonctionnalité importante à aborder avant de vous laisser programmer l'API est la récupération des listes. En effet, nous avons vu comment obtenir un message mais maintenant voyons comment obtenir une liste. Pour l'exemple nous récupérerons la liste des contacts ou amis du « Nabaztag ». Le code associé ressemblera à ceci :

```
// Récupère la liste des amis
public void nabGetFriendList(){
  try{
    String list  = "";
    String url    = APIURL+"?sn="+mSerial+"&token="+mToken+"&
action=2";
    Element  root  = sendXMLRequest(url);
    NodeList nodes = root.getElementsByTagName("friend");
    int    n     = nodes.getLength();
    for(int i=0;i<n;i++){
```

```
      Node node = nodes.item(i);
      NamedNodeMap attrs = node.getAttributes();
      list += attrs.getNamedItem("name").getNodeValue()+"\n";
    }
    Toast.makeText(mContext, list, Toast.LENGTH_LONG).show();
  }catch(Exception e){
    e.printStackTrace();
    Toast.makeText(mContext, e.getMessage(), Toast.LENGTH_LONG);
  }
}
```

Grâce à ces deux fonctions relativement simples « nabWakeup » et « nabGetFriendList » nous disposons de l'architecture de base pour la construction de notre objet « Nabaztag ». Il nous suffit désormais de suivre la documentation en ligne et de décliner chaque fonction avec ses paramètres. Les fichiers du projet sont disponibles sur CodeS-SourceS à l'adresse suivante :
http://www.javafr.com/codes/ANDROID-FAITES-PARLER-VOTRE-NABAZTAG_50668.aspx

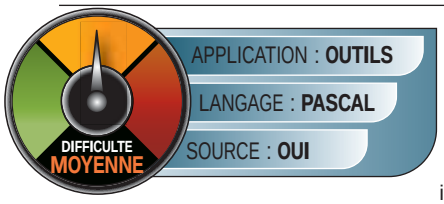
Ils vous permettront de lancer rapidement l'application afin de tester le code que nous venons de développer. Pour obtenir plus d'information sur Android et les applications qui composent son Market place, rendez-vous sur le site de référence http://www.androLib.com/
Si vous souhaitez vous former rapidement sur le développement d'application pour l'Android, utilisez notre formation en ligne http://www.codes-sources.com/video2brain.aspx?product_id=343

■ Emmanuel **ROBLES**
developpeur pour CodeS-SourceS et androLib

2e partie

Réalisation d'un compilateur de Pascal simplifié en C++, le lexing

Dans le précédent article nous avons brièvement vu les différentes étapes qui permettaient de créer un exécutable. Abordons maintenant la première phase plus en détail. Dans cette étape, il va falloir non seulement détecter et extraire les tokens du fichier source, mais aussi les vérifier. Enfin, l'ensemble de ces tokens seront sauvegardés puisqu'ils sont indispensables à la réalisation des étapes suivantes.



QUELQUES BONNES PRATIQUES

Avant d'aller plus loin, il est important lorsqu'on aborde un projet de cette taille, de se fixer quelques bonnes pratiques. Voici donc une liste, qui pourra sembler évidente pour certains, des habitudes à prendre :

- Fixez-vous une norme d'écriture, c'est-à-dire choisissez la manière dont vous indenterez votre code, et le style d'écriture que vous adopterez (Pascal Case, Caml case, C case, etc...).
- Essayez au maximum de rendre votre code le plus proche possible d'un langage parlé. Pour cela, utilisez des noms explicites d'identifiants et découpez votre code en une série de petites fonctions et classes. Une fonction ne devrait pas excéder en moyenne une cinquantaine de lignes.
- Évitez de mettre du code dans un fichier header. Pour les templates et fonctions "inliner" préférez l'utilisation d'un fichier externe inclus dans votre fichier header (l'extension est généralement .hxx sous Unix).
- Documentez votre code. Un commentaire devrait être écrit avant de commencer à écrire le comportement d'une fonction auquel il est rattaché.
- Essayez de ne faire qu'une classe par fichier.
- Évitez les "using namespace std;". Par exemple, préférez une simple "using std::cout;", si votre but est juste de ne pas réécrire "std::" devant un "cout".
- Utilisez des namespaces pour modulariser votre code.
- Usez et abusez des "assert" ! Assert est une macro fonction, qui permet de s'assurer qu'un prédicat est vrai. En mode release, les "assert" disparaissent purement et simplement, il n'y a donc aucun coût supplémentaire.

LES DESIGN PATTERNS

Un design pattern, ou patron de conception, est une solution élégante et éprouvée pour résoudre un problème récurrent. De nombreux design patterns existent et je ne les détaillerai pas tous. Ce projet en utilise trois: le *singleton*, le *flyweight* et le *visitor*. Chacun de ces patrons de conception sera expliqué au fur et à mesure.

LES SYMBOLES ET EXPRESSIONS

Dans notre langage, sont présents des mots et des symboles. De manière à aisément les modifier par la suite, nous allons commencer par créer une classe qui se chargera d'externaliser la liste des mots valides. Nous allons appeler cette classe "Configuration". Cette classe représentant une configuration unique, ne doit pas être dupliquée, et être accessible en tout endroit de notre code. Le design pattern singleton est donc tout à fait adapté.

EXPLICATION DU DESIGN PATTERN SINGLETON

Le but de ce design pattern est d'assurer l'unicité d'un objet cible. Le code nécessaire à sa réalisation n'est pas très compliqué. Nous allons créer une classe, la plus générique possible. Cette classe sera templatée par l'objet dont l'unicité doit être assurée. Tout d'abord, nous allons rendre impossible la construction d'un tel objet. Pour cela, nous allons établir le niveau de visibilité du constructeur par défaut, comme étant inaccessible. Il est maintenant impossible d'instancier la classe directement. Pour instancier une telle classe, nous allons créer une méthode publique, et surtout statique (c'est-à-dire qui peut être utilisée sans instancier la classe ou elle se trouve), et gérer la construction de cette classe à travers cette méthode. La méthode s'assurera que c'est bien toujours le même objet qui sera renvoyé, et non une duplication. Pour réaliser cela, il suffit d'utiliser une variable locale statique, qui par définition, ne sera initialisée qu'une et une seule fois. Ainsi, il est impossible de faire une copie de ce même objet. Voici à quoi ressemble notre classe (Ceci est une version simplifiée et compactée) :

```
template<typename T>
class Singleton
{
protected:
  Singleton() {}
public:
  static T& getInstance()
  {
    static T object;
    return object;
  }
};
```

La classe configuration contiendra un conteneur d'association clé-valeur (std::map), qui associera un type de mot à la représentation que l'on voudra en faire. Par exemple, je peux associer le mot "str" à "\"", pour indiquer le sigle qui permet l'ouverture et la fermeture d'une chaîne de caractères. Vous trouverez l'ensemble de ces sigles dans le fichier "Configuration.hxx". Maintenant, pour que notre clas-

se configuration soit un singleton, il suffit juste d'hériter de cette classe de la manière suivante :

```
class Configuration : public Singleton<Configuration>
{
  friend class Singleton<Configuration>;
private:
  Configuration();
  std::map<std::string, std::string> _keyWords;
};
```

Vous remarquez, que le constructeur de cette classe est mis en visibilité privée, afin d'empêcher la construction de cette dite-classe. De plus, la présence du mot clé "friend" peut sembler ici étrange. Bien que celui-ci soit facultatif, il va permettre d'alléger l'écriture lors de l'appel à une instance. En effet, pour pouvoir récupérer l'instance il faut écrire :

```
Configuration& cfg = Singleton<Configuration>::getInstance();
```

Ce qui n'est pas très élégant. En indiquant que configuration est ami avec le singleton, il est possible de simplifier l'écriture en :

```
Configuration& cfg = Configuration::getInstance();
```

Cette écriture étant tout de même plus intuitive.

LA CLASSE ERROR

Quelle que soit l'étape, en cas de code source invalide, il faudra signaler l'erreur à l'utilisateur. Pour l'aider dans cette démarche, il faut bien évidemment lui indiquer le type d'erreur, l'endroit ou se situe celle-ci et accompagner le tout d'un message explicatif. Toutes ces informations seront regroupées au sein de cette classe, dont voici le code simplifié :

```
class Error
{
public:
  Error(const Error::type type, const std::string msg, const int line)
    : _type(type), _msg(msg), _line(line)
  {
  }
private:
  const type _type;
  const std::string _msg;
  const int _line;
};
```

LA CLASSE ERROR HANDLER

Lorsqu'une erreur survient, on ne s'arrête généralement pas tout de suite. On cherche à en détecter le plus possible. C'est pour cela que même si un code source est faux, on continuera d'analyser le fichier afin de présenter à l'utilisateur le maximum d'erreurs en une fois. Cette classe n'est pas très compliquée à comprendre et contiendra essentiellement une liste d'erreurs (std::vector<Error*> _errors;);

LA CLASSE SYMBOL

A chaque fois que l'on récupérera un token, il faudra non seulement retenir le symbole textuel, mais aussi les informations qui lui sont associées, comme la ligne ou était présent celui-ci, ainsi que le type de token (par exemple, si c'est une expression simple, un opérateur, une chaîne de caractères, etc…). La classe Symbol se chargera, en fonction du symbole textuel donné au constructeur d'en déduire le type de token. En déléguant cette responsabilité à cette classe, nous allons grandement simplifier celle qui se chargera du lexing. Cette classe possèdera trois constructeurs, le premier demandera toutes les informations nécessaires à la création d'un symbole (token, ligne et type), le deuxième ne demandera que deux informations, et déduira à partir du token, le type de token, et enfin le troisième construira un nouveau symbole à partir d'un autre. Vous noterez aussi la présence d'une méthode "check()" qui permettra de vérifier, dans le cas ou le token est de type "identifiant" (c'est-à- dire si c'est un nom de variable ou de fonction), que ce nom est correct. Pour qu'un nom soit valide il doit être composé uniquement de lettres, de chiffres, et de caractères "_", et commencer par une lettre. Enfin l'opérateur << est redéfini afin de facilement afficher un symbole à l'écran. Notre classe Symbol à tout de même un défaut. Si un token apparaît plusieurs fois, alors dans la liste des symboles, nous aurons plusieurs fois en mémoire la chaîne de caractères représentant celui-ci. Dans un souci d'optimisation, nous allons faire en sorte d'assurer l'unicité de chaque chaîne rencontrée. Pour cela, nous allons maintenant étudier le design pattern *flyweight,* qui répond à ce besoin.

LE DESIGN PATTERN FLYWEIGHT

Imaginons la situation suivante: vous essayez de coder un logiciel de traitement de texte basique. Chaque lettre tapée possède une représentation visuelle en fonction de sa taille, de sa fonte et de sa couleur. Vous allez donc naturellement créer une classe "Lettre". Ainsi, pour chaque lettre tapée, une nouvelle instance de "Lettre" est créée. Le nombre de lettres tapées pouvant rapidement être important, la mémoire sera saturée très vite. Une approche intelligente, est de tout simplement vérifier lors de la création d'une lettre, que celle-ci n'est pas déjà présente avec les mêmes caractéristiques. Si c'est le cas, autant utiliser celle déjà existante. Cette astuce va permettre de gagner énormément d'espace mémoire. En C++, pour créer une classe *flyweight* générique, il suffit de peu de chose. Tout d'abord, utilisons un conteneur commun à tous les instances d'une même classe. Ce conteneur doit assurer l'unicité de chaque élément, nous utiliserons donc un *set.* Il suffit ensuite de vérifier, lors de la création d'un nouvel élément, si celui est présent ou non. Si ce n'est pas le cas, on ajoute cet élément. On renvoie ensuite l'adresse de l'élément, qui existera forcément. De plus, nous redéfinirons tous les opérateurs d'affectation et de comparaison afin de rendre notre classe facilement utilisable. Enfin, des méthodes statiques "clear()" et "size()" nous permettront respectivement de vider et d'obtenir la taille du conteneur partagé.
Voici un extrait de la classe simplifiée :

```
template <typename T>
class Flyweight
{
public:
  Flyweight(const T& elt);
  static void clear();
  static unsigned int size();
  const Flyweight& operator=(const T& s);
  bool operator==(const T& elt);
```

```
  bool operator!=(const T& elt);
protected:
  static std::set<T> set;
  const T* _elt;
};
```

LA CLASSE SHAREDSTRING

Maintenant que nous avons créé ce design pattern, nous allons l'appliquer à notre projet. Pour cela, nous allons créer la classe SharedString qui permettra de partager en mémoire les instances de chaîne de caractères similaires. La création de cette classe est triviale puisqu'il suffit d'hériter d'un *flyweight* spécialisé. Quelques méthodes supplémentaires sont présentes, comme la possibilité de récupérer un caractère à une position donnée, obtenir la taille de la chaîne, ainsi que des interactions définies avec les chaînes de caractère sous forme de "char*". Ceci permet à cette classe de s'utiliser exactement comme un std::string, ce qui rend son utilisation très intuitive. Voici un extrait de la classe simplifiée :

```
class SharedString : public Flyweight<std::string>
{
public:
  SharedString(const SharedString& s);
  SharedString(const std::string& s);
  SharedString(const char* s);
  const SharedString& operator=(const char* s);
  bool operator==(const char* s);
  bool operator!=(const char* s);
  char& operator[](const unsigned int i);
  unsigned int length() const;
};
```

LA CLASSE LEXER

Maintenant que nous avons créé tout le nécessaire, attaquons-nous à la partie principale: le lexing. Notre classe Lexer, n'est pas bien difficile à comprendre. Elle possède une liste d'erreurs (elle encapsule une classe Error Handler), et une liste de symboles. Il suffit juste de récupérer chacun des tokens et de mettre à jour la table des symboles au fur et à mesure. En cas d'erreur, on ignore le token en cours, on ajoute une nouvelle erreur à la liste d'erreurs, et on continue d'essayer d'extraire les tokens suivants.

Détaillons maintenant la méthode utilisée pour extraire les tokens. Tout d'abord, on vérifie que le token en cours n'est pas un token d'ouverture de commentaire, auquel cas, on passe directement à la ligne suivante, puisqu'un commentaire est par définition ignoré. On vérifie aussi que le token en cours n'est pas un token d'ouverture de chaîne de caractère. Si tel est le cas, alors on se contente de trouver le caractère de fermeture de chaînes de caractères et on enregistre la chaîne ainsi récupérée comme étant un token à part entière. Il est important de bien différencier les mots clés, des symboles atomiques. Un mot clé doit forcément être écrit de manière stricte, c'est-à-dire sans que celui-ci soit "collé" à un autre token, ce qui n'est pas le cas des symboles atomiques. Par exemple : "begin end" sont bien deux mots clés, mais "beginend", sera considéré comme une expression simple. En revanche, si on prend le symbole atomique ">=", on se rend bien compte que "4 >= 5" et "4>=5" sont deux expressions identiques.

Nous allons commencer par détecter les symboles atomiques ayant deux caractères. En effet, il faut toujours rechercher en priorité les tokens les plus grands pour éviter toute ambiguïté. Par exemple, il est important lorsque l'on rencontre la chaîne suivante : "<=", de bien détecter le symbole atomique "<=" et ne surtout pas détecter deux symboles atomiques qui se suivraient. Si l'expression en cours d'analyse, n'est ni un symbole atomique, ni un mot clé, alors il reste juste à vérifier que celle-ci soit une valeur ou une variable. Si l'expression en cours n'est composée que de chiffres, alors c'est une valeur. Dans le cas contraire, s'il ne s'agit d'aucun des types de token précédemment décrits, ce ne peut être qu'une variable.

MISE EN PLACE

Maintenant que notre Lexer est terminé, nous allons juste mettre en place un système d'option. Celui-ci est trivial, il suffit simplement de considérer que le premier argument de la ligne de commande (argv[1]), représente l'option, et les arguments suivants, les fichiers passés à notre binaire. Nous allons donc créer les deux options suivantes: *-l,* qui permettra de lancer le lexing sur les fichiers donnés en arguments, c'est-à-dire qui affichera les éventuelles erreurs de lexing, et *-L,* qui en plus de faire ce que fait la première option, affichera aussi des informations sur tous les tokens récupérés, c'est-à-dire les types de tokens, la ligne où ceux-ci ont été extraits, et, leurs symboles textuels. Cette mise en place est suffisamment simple pour que je ne la détaille pas.

Vous trouverez les sources de cette première phase sur le site. Dans le prochain article, nous aborderons le parsing, mais pour les plus impatients d'entre vous, le projet complet pourra être récupéré à cette adresse : https://svn.assembla.com/svn/cubs

■ Axel **Berardino** - *axel.berardino@gmail.com*
Étudiant en dernière année à l'EPITA. A souvent dispensé des cours et des formations en entreprise. A l'origine, ce projet de compilateur était dédié à une formation en C++.

COLLECTION

Collection « le guide complet » (Micro Application)

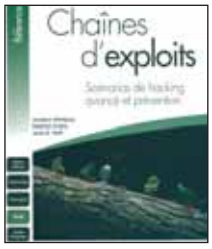
*Difficulté : ****
Prix : 15 €

Joomla : environnement web incontournable avec Drupal, Joomla s'impose aujourd'hui auprès de nombreux développeurs. Extensible et puissant, Joomla offre une flexibilité impressionnante et couplé avec VirtualMart, permet une mise en place rapide de son site de eCommerce. Dans ce guide pratique, les auteurs abordent l'installation, le paramétrage, les premiers contenus, l'administration et, bien entendu, la personnalisation. Rapide à lire, il permet d'acquérir les fondamentaux des outils.

Débuter en programmation : qu'est-ce que la programmation ? Comment débuter en douceur ? Nous nous sommes tous posés la question à un moment ou un autre. Ce livre se propose de vous accompagner dans les premiers pas de programmation en s'appuyant sur Visual Basic 2008 Express Edition, la programmation .Net et Windows, avec aussi un soupçon de Web. L'objectif est d'aborder les concepts objets, la sérialisation, les données, la gestion des événements, etc. De nombreux exemples ponctuent l'ouvrage.

SÉCURITÉ

Chaînes d'exploits



*Difficulté : *****
Editeur : Pearson
Auteur : collectif
Prix : 32 €

Une chaîne d'exploits est une méthode d'attaque consistant à combiner, à utiliser plusieurs méthodes d'attaques. Car comme le rappellent les auteurs, un hacker utilise souvent plusieurs techniques pour attaquer, accéder à un réseau, une machine, une application. Or, pour comprendre leur principe, il faut aborder la chaîne dans son ensemble et savoir la décortiquer. Ce livre se propose d'aborder les principales chaînes d'exploits. Basé sur des scénarios très concrets, l'ouvrage se veut compréhensible et pratique. De nombreux conseils ponctuent les chapitres. Bien entendu, les auteurs proposent aussi des mesures de prévention et les contre- mesures à appliquer.

LIVRE DU MOIS

Programmation Python 2e édition

*Difficulté : **** - Editeur : Eyrolles*
Auteur : Tarek Ziadé - Prix : 38 €

Incontournable, voilà en résumé ce qu'on peut dire de ce livre qui nous arrive en 2e édition. L'auteur revient sur l'intérêt du langage, son installation et les premiers pas dans le merveilleux monde du serpent. Cette seconde édition reprend le succès de la première en y ajoutant de nouveaux cas pratiques, de nouveaux exercices et surtout prend en compte Python 2.6, le script de migration 2to3, ctypes. Pour le reste, on retrouve les chapitres syntaxes, modules, conventions de codage, approche objet, tests, optimisation, etc. Très complet.

SI

Stratégie et pilotage des systèmes d'information

*Difficulté : ***
Editeur : Dunod
Auteur : collectif
Prix : 32 €

Le système d'information ou SI est comme un organisme vivant qu'il faut comprendre, faire grandir, gérer, voire punir. Ce que l'on appelle gouvernance du SI est souvent perçu comme une notion fumeuse et obscure. Dans cet ouvrage, les auteurs abordent deux notions importantes : la stratégie du SI et son pilotage. Avec des chapitres courts et très ciblés, l'ouvrage offre une vue d'ensemble de ces problématiques et surtout utilisent des cas concrets et réels pour appuyer sur les points sensibles. Instructif et clair, un ouvrage à lire, que l'on soit administrateur, DSI ou simple passionné. Vous comprendrez que le SI ressemble réellement à une jungle.

SÉCURITÉ

Sécurité informatique Ethical Hacking

*Difficulté : ****
Editeur : Eni
Auteur : collectif
Prix : 54 €

« La meilleure défense c'est l'attaque », l'adage vaut aussi en sécurité informatique. Il faut acquérir les techniques, penser comme un hacker. Le chapitre sur le Social Engineering, ou manipulation sociale, illustre pourquoi les failles humaines représentent plus de 60% des attaques réussies. Les failles physiques, qui permettent un accès direct aux ordinateurs visés ainsi que les failles réseaux et Wi-Fi sont présentées et illustrées avec à chaque fois des propositions de contre-mesures. Et chaque attaque a sa réponse. Il faut alors élaborer une stratégie de sécurité adaptée à chaque faille, chaque problème. A lire et à relire.

LANGAGE

Manuel de prise en main de XML

*Difficulté : ***
Editeur : Pearson
Auteur : Kevin Howard Goldberg
Prix : 29 €

Apprendre et comprendre XML c'est laborieux et peu passionnant ? Pas forcément, Kevin tente de rendre digeste XML, les concepts, son utilité, son fonctionnement, sa mise en œuvre.
On y trouve tous les fondamentaux : schéma, DTD, Xpath, XQuery, XSLT, etc. L'explication se veut claire, avec de nombreux exemples concis. Une approche, très appréciable, du XML.

Erratum
Une malencontreuse erreur s'est glissée concernant le livre « GreenIT » du n° précédent de Programmez (123). Il est édité chez DUNOD et non Eyrolles comme indiqué dans le texte.

Les outils des Décideurs Informatiques

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

L'INFORMATION SUR MESURE

L'INFORMATION EN CONTINU

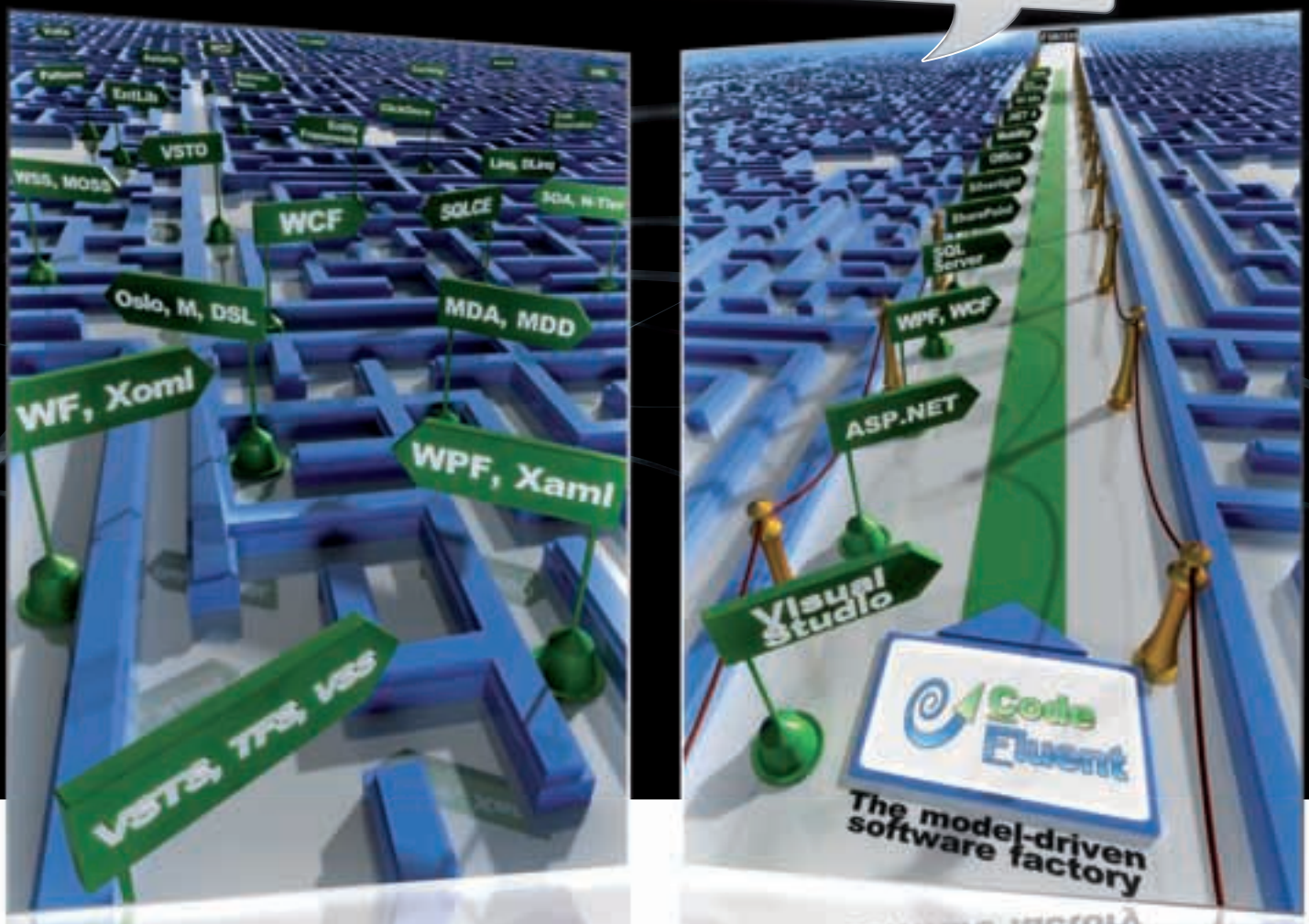
www.solutions-logiciels.com

CODEFLUENT

La première fabrique logicielle .NET entièrement pilotée par les modèles

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

sans **avec**



Le plus court chemin vers les technologies .NET

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

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

idEES