PROGRAMMEZ !
Le magazine du développement
PROgrammez !
mensuel n°142 - juin 2011

L'accélération matérielle booste le Web

Tests votre anti-bugs

Améliorer votre code avec les tests
Les bonnes pratiques, les outils, les nouvelles techniques de tests, JMeter, Serendipity Testing !

Carrière
On recrute ! L'open source explose

Smartphone
Windows Phone 7.5 toutes les nouveautés !

Open Source
Comment construire une communauté

Tout savoir sur JavaFX 2.0

Traduction
Traduire des textes avec les API Google et Bing

.Net
Découvrir Entity Framework 4.1

Drupal
Actions et déclencheurs sous Drupal 6 et 7

Outil
Utiliser Jenkins

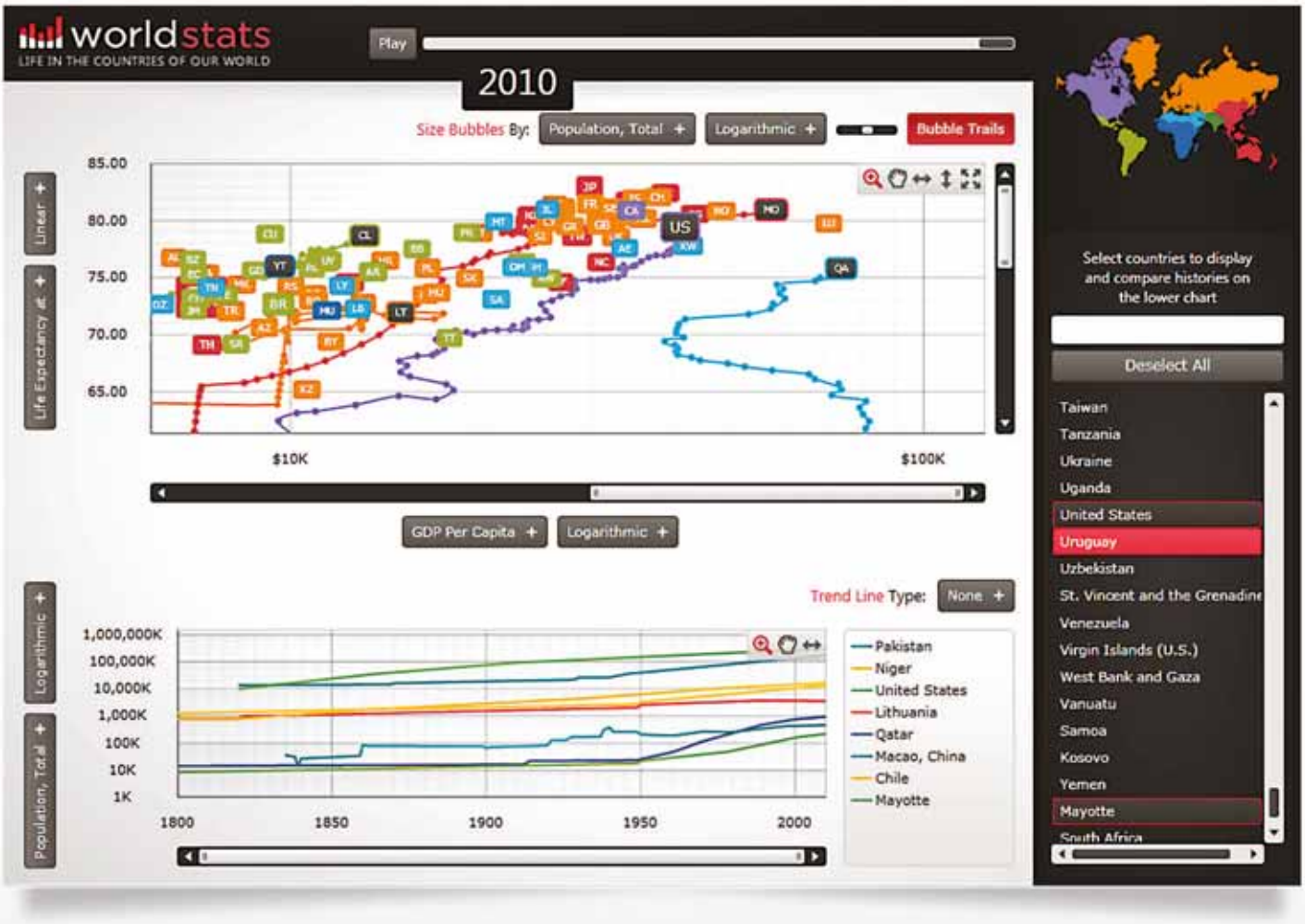
M 04319 - 142 - F: 5,95 €
Printed in France - Imprimé en France - BELGIQUE 6,45 €
SUISSE 12 FS - LUXEMBOURG 6,45 € - DOM Surf 6,90 €
Canada 8,95 $ CAN - TOM 940 XPF - MAROC 50 DH

ATELIER DE GÉNIE LOGICIEL PROFESSIONNEL
WINDEV®
16
CRÉEZ VOS APPLICATIONS POUR PC, MAC, LINUX, INTERNET, SMARTPHONES, TABLETTES,...
VOTRE CODE EST MULTI-PLATEFORMES
Windows, .Net, Java, PHP, Linux, Mac, J2EE, XML, Internet, SaaS, Pocket PC, Windows Phone 7, Android, ...
GÉREZ LE «CYCLE DE VIE» DE VOS APPLICATIONS (ALM)
N°1 EN FRANCE
• Crée des applications Windows, Linux, Mac,... compatibles Internet, Intranet, Android, Windows Phone 7 & Mobile
• Gestion complète du cycle de vie (ALM)
• Descripteur de données (UML,Merise...)
• RAD : créez une application d'un clic
• Lien avec toutes les bases de données (Lien natif, ODBC, OLE DB): Oracle, SQL Server, AS/400, Informix, DB2, MySQL, PostgreSQL, SQLite,...
• HyperFileSQL: puissante base de données Client/Serveur gratuite incluse
• Générateur d'états PDF, Codesbarres
• Accès natif SAP R/3, Lotus Notes, ...
• Gestion de planning, des Exigences
• Audit de vos applications
• Langage de 5e génération (L5G)
• SNMP, Bluetooth, TAPI, OPC, FTP, HTTP, Socket, Twain, API, DLL, XML...
• Domotique • Liaisons série et USB
• Débogage à distance, Profiler
• Multilangue (64 langues)
• Installateur 1-clic & Push
• Gestionnaire de versions
• Gestion de l'infrastructure
• Tout en français
• etc...
Support technique gratuit
10 FOIS PLUS VITE
DÉVELOPPEZ
CONCENTREZ-VOUS SUR LES BESOINS MÉTIER
NE RÉ-INVENTEZ PAS LA ROUE EN PERMANENCE
Elu «Langage le plus productif du marché»
VERSION EXPRESS GRATUITE
Téléchargez-la !
100 TÉMOIGNAGES SUR SIMPLE DEMANDE
01 INFORMATIQUE
100 RÉUSSITES RÉVÉLÉES
WINDEV WEBDEV L'AVENIR
996 NOUVEAUTÉS
WINDEV® 16
PLATEFORME INTÉGRÉE DE DÉVELOPPEMENT
Fournisseur Officiel de la Préparation Olympique
DEMANDEZ LE DOSSIER GRATUIT
Dossier gratuit 260 pages sur simple demande. Tél: 04.67.032.032 info@pcsoft.fr
www.pcsoft.fr
document non contractuel. Version 16 annoncée. *15 requêtes gratuites sur la version en cours de commercialisation, seule la

PROgrammez! Le magazine du développement
numéro 142 - juin 2011

sommaire\\

NetAdvantage®
ULTIMATE



RAPPORTS, VISUALISATION DES DONNÉES ET CONTRÔLES D'INTERFACE UTILISATEUR POUR ASP.NET, WINDOWS FORMS, JQUERY/HTML5, WPF,SILVERLIGHT ET WINDOWS PHONE 7

SCANNEZ ICI POUR DECOUVRIR ULTIMATE!
www.infragistics.com/ult

FAITES PASSER VOS APPLICATIONS AU NIVEAU SUPERIEUR
INFRAGISTICS.COM/ULTIMATE

INFRAGISTICS
DESIGN / DEVELOP / EXPERIENCE

Infragistics Ventes France N° Vert 0800 667 307 • Infragistics Ventes Europe +44 (0) 800 298 9055 • Infragistics India +91 80 4151 8042 • @infragistics

éditO


©V.T.

Pourquoi payer une fois quand tu peux payer deux fois

Décidément, notre univers informatique est véritablement impitoyable : Nokia qui se déleste de Symbian, Mono qui passe de vie à trépas et de trépas à la résurrection, ou encore Google qui fait le beau à I/O, Ubuntu qui vire Eucalyptus, Red Hat et VMware faisant la course du « mon PaaS est plus gros que le tien ». Difficile de faire mieux !

Après quelques mois d'incertitudes et de rumeurs, Google sortira finalement ChromeOS et les netbooks adaptés (Chromebook) mi-juin. L'idée est de ne rien stocker sur le PC qui devient un client léger, pour tout afficher dans son navigateur (Chrome bien entendu). Google est convaincu du concept et pense que les usages sont déjà là. Plusieurs problèmes, cependant, surgissent en regardant le modèle proposé : la nécessité d'une connexion Internet constante et de qualité avec un débit conséquent, l'achat du Chromebook et du forfait pour la connexion réseau (3G). Ce dernier point suscite une réaction car sur un an, il double le prix du Chromebook. Et aujourd'hui, il n'est pas possible de garantir une connectivité au réseau constante, même en wifi. Pour pallier, un peu le problème, le mode déconnecté sera pris en charge, c'est le minimum en effet. Mais le développeur devra encore adopter une nouvelle plateforme applicative car Android et ChromeOS ne sont pas compatibles. Au passage, n'est-ce pas de la vente forcée en France ? Et que fait l'Europe sur la pluralité des navigateurs, car ChromeOS n'est livré qu'avec Chrome !

Pourquoi ChromeOS ne prend-il pas plutôt exemple sur une plateforme cloud (PaaS) de type Cloud Foundry ou Windows Azure avec un modèle de développement plus proche du desktop et offrant des possibilités fonctionnelles particulièrement étendues ? Attendons de voir comment le marché va réagir. Mais pour notre part, le prix et le modèle applicatif nous incitent à garder nos netbook et iPad.

Ce mois-ci, nous avons voulu reprendre notre rôle d'évangéliste sur un sujet qui nous tient particulièrement à cœur : les tests. Personnellement, je faisais mes premiers protocoles de tests il y a 20 ans.
Alors, oui, le test c'est moche, pas sexy, et ennuyeux. Mais, vous n'êtes pas des bisounours tout de même ! Le test devrait être tatoué sur vos mains, dans votre cerveau. Il est indispensable, incontournable !
Même quand vous codez une petite application personnelle, le test doit être fait le plus tôt possible. Car on ne pense pas à tout quand on code. Une simple couverture de code peut déjà supprimer des défauts. La panoplie de tests est aujourd'hui impressionnante quel que soit le langage, l'IDE. Et c'est une compétence de plus en plus demandée. Alors pourquoi pas vous ?
Il y a des tests que vous pouvez réaliser rapidement : les tests unitaires car vous restez au plus près du code. Nous recommandons de faire au moins, notamment sur les sites web, des tests de performances, de montée en charge, de stress, d'interface. En SSII, en entreprise, on passe à un autre niveau, plus industrialisé.

Sur le plan des carrières, nous faisons le point sur le marché de l'emploi open source. Depuis début 2011, une belle dynamique existe, notamment avec des compétences pointues sur Drupal. Et de nombreux spécialistes du Libre cherchent à embaucher, notamment en province. Eh oui il n'y a pas que Paris pour faire carrière dans l'informatique.

François **Tonic**
Rédacteur en chef

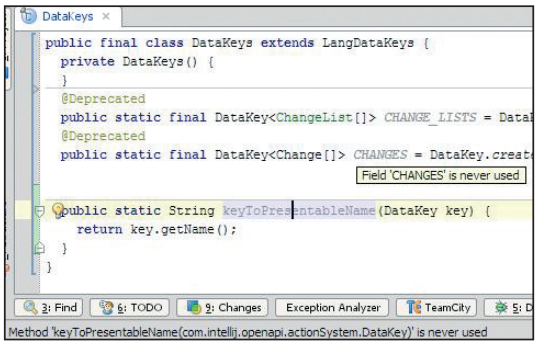
Le magazine du développement
PROgrammez!

Editeur : Go-02 sarl, 21 rue de Fécamp 75012 Paris - diff@programmez.com.
Rédaction : redaction@programmez.com
Directeur de la Rédaction : Jean Kaminsky.
Rédacteur en Chef : François Tonic - ftonic@programmez.com. **Ont collaboré à ce numéro :** F. Mazué, S. Saurel. **Experts :** D. Rousset, A. Auroux, T. Lebrun, A. Crepet, V. Pernin, P. Guillaud, B. Homes, J. Scher, M. Arnstamm, E. Lafkih, B. Pisson, C. Villeneuve, J.L. Letouzey, O. Gaudin, B. Michel, J. de Oliveira, F. Bellahcene, N. De Loof.
Illustrations couverture : © iStockPhoto/Malerapaso

Publicité : Régie publicitaire, K-Now sarl. Pour la publicité uniquement : Tél. : 01 41 77 16 03 - diff@programmez.com.
Dépôt légal : à parution - Commission paritaire : 0712K78366 ISSN : 1627-0908. Imprimeur : S.A. Corelio Nevada Printing, 30 allée de la recherche, 1070 Bruxelles Belgique. **Directeur de la publication :** J-C Vaudecrane
Ce numéro comporte un encart libre OVH.

Abonnement : Programmez 22, rue René Boulanger, 75472 Paris Cedex 10
Tél. : 01 55 56 70 55
abonnements.programmez @groupe-gli.com
Fax : 01 55 56 70 91 - 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

■ JetBrains dévoile **IntelliJ IDEA 10.5**. Il s'agit d'un excellent IDE Java. Cette version se concentre sur Java 7 et supporte le débug JavaScript dans



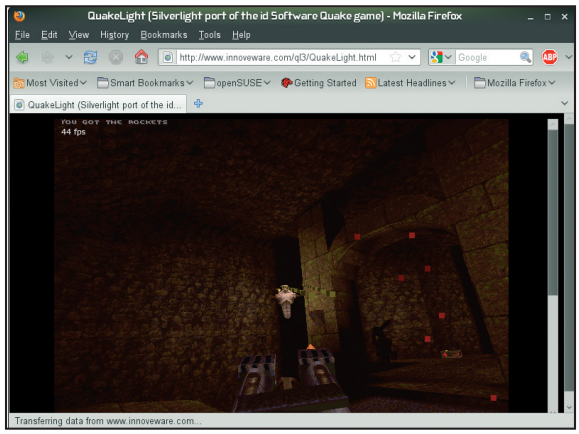
Chrome, Groovy 1.8, Spring 3.1, Jetty, XSLT2. Il améliore aussi le développement JavaScript, les performances, le refactoring. La version communautaire est disponible gratuitement.

■ EnterpriseDB a dévoilé Postgres Plus Advanced Server pour les serveurs HP-UX. EnterpriseDB fournira un support complet de HP-UX à compter de Juin 2011. **Postgres Plus Advanced Server** offre à ses utilisateurs l'opportunité de migrer facilement et en toute transparence leurs applications et leurs données depuis un environnement de production reposant sur une base de données Oracle vers Postgres Plus.

■ Compuware et Google travaillent ensemble sur l'optimisation des applications web autour de **Gomez.** L'outil de Compuware sera intégré au profit de Page Speed de Google. Ce projet doit fournir aux développeurs des recommandations, bonnes pratiques pour améliorer les performances des applications web.

■ Au même titre que le sont WordPress ou Drupal, Symfony2 est désormais disponible en version bêta sur WebPI. L'intégration de **Symfony2** à WebPI représente pour Symfony2 une belle opportunité d'accroître sa communauté et pour Sensio Labs, d'évangéliser autour du framework et d'atteindre de nouvelles cibles. L'éditeur de Redmond voit quant à lui sa plateforme enrichie par la dernière mouture du framework PHP, qui sera disponible fin juin.

Projet Mono : sauvetage in extremis



Le rachat de Novell, pour 2,2 milliards de dollars, par Attachmate a rapidement laissé place aux coupes et réorganisations. Une des cibles principales fut le ménage par le vide de la partie Mono, la pile .Net open source disponible sur Linux et MacOS X. La nouvelle avait fait le tour de la communauté en quelques jours. Mono n'était clairement pas une priorité pour le nouvel acquéreur et depuis qu'il existe, le projet a suscité beaucoup de rejets, mais à chaque fois, Miguel de Icaza a réussi à mobiliser développeurs et partenaires autour de Mono (à l'origine Mono provient de Ximian, acheté par Novell en 2003). Début mai, la situation était visiblement bloquée et l'avenir paraissait très sombre, voire fatal.

Mi-mai, Miguel de Icaza annonçait la création d'une nouvelle société pour Mono : Xamarin. Sa mission est de s'occuper du développement, du support, et de l'écosystème Mono. Les priorités sont : disposer d'une offre technique commerciale pour iOS, Android, continuer à contribuer, maintenir Mono et Moonlight et étendre les opportunités de marché de Moonlight sur mobiles et Mac App Store. Aujourd'hui, Miguel insiste sur la séparation stricte entre l'interface et le code afin de faciliter la disponibilité de Mono sur de nombreuses plateformes et donc des applications. D'ores et déjà, les développeurs travaillent sur la solution iPhone. Et le travail s'annonce chargé car il faut notamment que les futures versions mobiles soient pleinement compatibles avec MonoTouch et Mono pour Android. Il s'agit donc de réarchitecturer toute une partie de Mono pour avoir une compatibilité maximale quelle que soit la plateforme cible et le cœur de la pile .Net. Ce chantier prendra 3-4 mois. Cela est rendu nécessaire pour le fait que MonoTouch et Mono pour Android n'appartiennent pas à Xamarin… La page officielle de Mono n'est d'ailleurs plus maintenue et mise à jour depuis janvier dernier. Que va devenir ce qui reste de Mono chez Attachmate ? Pas grand-chose visiblement, aucun plan, à notre connaissance, n'a été annoncé, qui va alors assurer le support technique des MonoTouch et autres MonoDroid ? Surtout, ces mésaventures ne rassurent pas les développeurs, utilisateurs. Mono avait eu bien du mal à se faire une place. Il faut espérer que Xamarin n'aura pas trop à souffrir. Reste aussi à savoir comment Microsoft, qui a soutenu activement Mono, réagira à cette nouvelle.

La conférence Monospace à Boston (juillet) est maintenue. Occasion pour Miguel de dévoiler des plans plus concrets, des dates, des projets.

Sites : http://monospace.us/
http://www.xamarin.com/
http://www.go-mono.com/monologue/

agenda \\

JUIN

• Le 1er juin 2011, **Rencontre sur les langages de script.** Locaux de l'IRILL, 23 Avenue d'Italie, Paris, France - 3ème étage. https://www.irill.org/events/scripting-language-mini-conference-and-barcamp-registration

• Jeudi 09 juin 2011, **WygDay 2011** - l'édition Lilloise des Microsoft Days. EuraTechnologies 165, Avenue de la Bretagne 59000 Lille. http://wygday.wygwam.com/Inscription.aspx

• Le 14 juin 2011, Cambon-Capucines, 46 rue Cambon 75001 Paris. **Mythes et réalités de la sécurité** http://www.lecercle.biz/Portals/3/secured/Agenda.aspx

• Du 14 au 17 juin 2011, Centre de Congrès de Disneyland Paris **Hack in Paris.** permettra de découvrir la réalité concrète du hacking et ses conséquences pour le fonctionnement des entreprises. http://www.hackinparis.com

• Du 24 au 25 juin 2011, Cité des Sciences à Paris, **Les Journées Perl 2011.** http://journeesperl.fr/fpw2011/

ETRANGER

• Du 05 au 06 juin 2011, Irlande, Convention Centre de Dublin. **Red Hat EMEA Partner Summit 2011** http://www.europe.redhat.com/mktg/partnersummit/2011/

• Du 02 au 05 août 2011, USA, Hôtel Hilton San Diego Bayfront, **FileMaker Developer Conference 2011.** http://www.filemaker.fr/pr/devcon2011

Retrouvez toutes les applis Nokia sur **Ovi Store.**

- Plus de 40 000 applis
- Des utilisateurs dans plus de 190 pays
- Des opérateurs offrant une facturation intégrée dans 34 pays
- Des applis pour plus de 135 terminaux dont 80 Symbian

NOKIA
Connecting People

NOKIA
ovi

www.nokia.fr/develop

HÉBERGEMENT MUTUALISÉ REDONDÉ :

1&1 DUAL H

SÉCURITÉ REDOUBLÉE AVEC LA REDONDANCE GÉOGRAPHIQUE !

Un site indisponible peut avoir des conséquences désastreuses. Grâce à la redondance géographique, les nouveaux packs Dual Hosting 1&1 vous offrent désormais une sécurité maximale ! Votre site Web est dorénavant hébergé simultanément dans deux de nos centres de données ultramodernes. Si un incident se produit dans le premier centre, les serveurs du deuxième prennent le relais. Votre site reste de ce fait toujours opérationnel et vous avez la garantie de ne perdre aucune donnée.

&

MEMBRE DE united internet

OSTING

Des compétences, un savoir-faire et une qualité de service qu'aucun autre hébergeur que 1&1 ne peut vous proposer :

1&1 allie 20 ans d'expérience dans l'hébergement Web à une technologie de pointe mise en œuvre dans des centres de données européens haute performance. Notre avance technologique est assurée par la contribution en interne de plus de 1000 développeurs. **NOUVEAUTÉ :** 1&1 est maintenant le premier hébergeur au monde à vous fournir une sécurité redoublée grâce aux packs Dual Hosting et la redondance géographique. Et le tout à un prix défiant toute concurrence !

Sécurité redoublée :
Redondance géographique

Performance maximale :
Serveurs haut de gamme

Rapidité exceptionnelle :
Connectivité de 210 Gbit/s

Hébergement vert :
Energie renouvelable

Innovation permanente :
1000 développeurs en interne

NOUVEAU !

1&1 DUAL ILLIMITÉ

- 4 noms de domaine INCLUS
- Espace disque ILLIMITÉ
- Trafic ILLIMITÉ
- Bases de données MySQL ILLIMITÉES
- Comptes email ILLIMITÉS
- Accès FTP ILLIMITÉS
- Applications Click & Build ILLIMITÉES (au choix parmi 65 applications à installer en 1 clic)
- 1&1 WebStat
- PHP5, PHP Dev, Zend Framework, Ruby, SSI, Accès SSH, gestionnaire de version Git
- Disponibilité réseau de 99,99 %
- NOUVEAU : redondance géographique
- Et bien plus encore !

1&1 DUAL ILLIMITÉ

9,99€ HT/mois
11,95 € TTC/mois

~~19,99€~~ HT/mois (23,91 € TTC/mois) **-50% pendant 12 mois ***

Découvrez d'autres packs sur la page suivante.

Toutes nos solutions d'hébergement en détail sur notre site Internet.

1&1

Appelez le **0970 808 911** (appel non surtaxé) ou consultez notre site Web **www.1and1.fr**

* 1&1 Dual Illimité est à -50 % pendant 12 mois sous réserve d'un engagement de 12 mois. A l'issue des 12 premiers mois, ce pack est au prix habituel de 19,99 € HT/mois (23,91 € TTC/mois). Frais de mise en service de 11,95 € TTC. Conditions détaillées sur 1and1.fr. Offre sans engagement également disponible.

Google I/O : Android, ChromeOS, App Engine...

L'événement développeur de Google, Google I/O, est l'occasion pour l'éditeur de dévoiler la stratégie, les nouveaux outils. L'édition 2011 a fait le plein de nouveautés même si de nombreuses questions demeurent. Et Google attaque clairement Microsoft, Apple, Amazon...

Android : plus que jamais !

8 versions, plus de 310 modèles disponibles, mais une multiplication des versions dans la nature, une v3 « dédiée tablette » quelque peu bancale... Google avait besoin de remettre un peu d'ordre. Et ce sera chose faite. Tout d'abord, une mise à jour de la v3 sera disponible pour le Xoom de Motorala (et les autres). Cette version devrait corriger les principales lacunes. Mais la véritable nouveauté arrivera avec Ice Cream Sandwich. Cette version sera commune aux smartphones et tablettes, comme le fait Apple avec iOS. C'est une bonne chose pour le développeur et les constructeurs. D'autre part, un SDK pour les accessoires sera disponible. Le projet Android@Home vise tout simplement à intégrer Android à la maison... Et le projet Tungsten doit pouvoir contrôler la musique diffusée par le réseau Home...

Google App Engine

Google a dévoilé App Engine 1.5 et un aperçu de son futur. La 1.5 inclura les fonctions de backends (pour les longues exécutions), les tasks queues seront améliorées, notamment avec les API REST. D'autre part, on trouvera App Engine 1.5 avec 99,95 % de disponibilité et un nouveau module de facturation sera mis en place. Google App Engine deviendra un service Google en version "finale" cette année. Des fonctions de la version business seront disponibles dans la version standard. La version gratuite demeure mais sans SLA, ni support ou montée en charge illimitée. Et les trafics de bande passante sont limités à 1 Go / jour. Une version à 9 $ par application est disponible et une édition premium à 500 $. Attention aussi, comme pour Windows Azure, les API seront payantes au-delà d'un certain quota. Le langage Go sera intégré.

Quelques autres annonces techniques :

- Google Books API : API pour accéder, requêter le service livre de Google
- Google Storage est ouvert à tout le monde. Storage supporte OAuth 2, une localisation en Europe, nouvelle version des API, nouveaux quotas d'utilisation (pour la partie gratuite).
- Google Plugin for Eclipse arrive en v2.4. Il supporte le développement Android, App Engine ! Et il est possible de faire des projets App Engine connectés aux projets Android... De nouveaux assistants sont disponibles pour Android et GWT. On dispose aussi du service Cloud to device Messaging (C2DM), un système de notification à travers du cloud et Android
- Le P2P NFC sera disponible sur Android

ChromeOS : lancement 15 juin !

ChromeOS a fait une apparition assez remarquée. Google répète qu'il s'agit d'une révolution dans la conception du système et dans la manière d'utiliser un ordinateur. Tout est sur le web, Google refuse donc de modifier son approche. Le ChromeBook est un Netbook taillé pour ChromeOS avec une sécurité optimale, un démarrage ultra rapide, une autonomie d'une journée, mais il faudra bien rajouter les abonnements au réseau. Les premiers tests ont démarré il y a 6 mois. Google parle de bons retours permettant de corriger de nombreux bugs. Les partenaires comme Intel et Adobe sont importants pour la réussite du projet. L'aspect déconnecté est cependant pris en

compte par plusieurs applications Google, mais beaucoup d'autres annoncent cette fonction. Ouf ! On trouvera ces ordinateurs notamment chez Samsung et Acer. Les premiers modèles seront disponibles mi-juin.

Avec ChromeBook, Google est désormais partenaire de VMware et de Citrix pour faciliter le réseau en entreprise, simplifier l'administration et l'utilisation des applications. Un marché prometteur pour ChromeOS ? Pourquoi pas. Une des cibles est clairement les infrastructures utilisant toujours Windows XP. Le succès viendra-t-il du grand public ou des entreprises ? ChromeBook for business propose une formule de location à moins de 30 dollars, avec échange en cas de panne et à chaque nouvelle version. La version école sera à 20 dollars par utilisateur. Disponibilité mi-juin. ■

VoIP

Microsoft : pourquoi Skype ?

Google ou Microsoft, la bataille pour Skype a longtemps été affaire de rumeurs. L'annonce est tombée dans la journée du 10 mai : Microsoft rachète Skype, le service de téléphonie sur IP. Microsoft a mis sur la table un énorme chèque de 8,5 milliards de dollars, presque 3 fois le prix payé par ebay pour acheter Skype en 2005 ! L'intérêt de Skype se concentre sur la technologie de téléphonie sur IP (donc passant par Internet) et le nombre d'utilisateurs réguliers ou occasionnels, plus de 170 millions. Car Skype dégage peu de chiffre d'affaires, à peine 900 millions en 2010.

Pour Microsoft, Skype a l'avantage de compléter les offres Windows Live et Lync (communication unifiée). Et Microsoft a indiqué que Skype serait intégré dans ses services en ligne, ainsi que sur Xbox Live, voire Kinect, Windows Phone. Microsoft a voulu rassurer en réaffirmant que l'outil de communication resterait multiplateforme. Pour Microsoft, Skype constitue donc une technologie stratégique pour combler les retards de communication sur Internet de plusieurs de ses services et contrer Google et Apple sur la vidéoconférence par exemple.

Justice

Oracle vs Google : Apache surgit

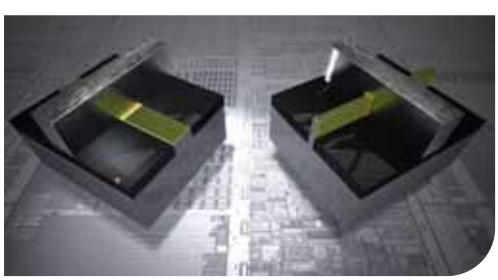
Le feuilleton juridique Oracle – Google a connu un rebondissement non négligeable : Oracle implique la fondation Apache. Le projet Harmony est au cœur de la demande d'Oracle. Ce projet n'avait pas obtenu la certification Sun, puis Oracle pour la licence d'utilisation Java (TCK). Cette licence valide la JVM selon les « normes » définies. Le blog de la fondation a été clair sur ce point : *« La fondation Apache a reçu une injonction de production de documents relatifs à l'utilisation du code Harmony d'Apache dans la plate-forme Android, ainsi que sur sa tentative infructueuse d'obtenir une licence du kit de compatibilité de Java SE. »*. L'objectif est de dévoiler tous les documents échangés entre Google et Apache vis-à-vis de l'utilisation de Harmony dans Android et plus précisément dans la machine virtuelle. Le nœud du problème de Java dans Android serait-il alors l'utilisation des portions de Harmony, lui-même non certifié ? Certes, mais dans ce cas, si Google était au courant du problème de licence et si le projet Android a été développé de zéro, pourquoi Harmony ?

En janvier dernier, Florian Mueller, un expert du juridique IT et anti-brevet avait soulevé plusieurs étrangetés dans des branches de code d'Android. Plusieurs fichiers seraient, selon l'expert, une copie de fichiers Java dans le projet Android (http://fosspatents.blogspot.com/2011/01/new-evidence-supports-oracles-case.html).

■ Un bracelet tag ? Oui c'est possible avec **addmefor**. Grâce à une application de décodage de QRcode sur mobile (i-nigma…), lorsque vous scannez le bracelet vous êtes automatiquement redirigé vers le profil facebook du porteur. Le produit est disponible sur le site de Addmefor en plusieurs coloris, proposé dans un coffret haut de gamme au prix de 25 euros TTC (frais de port compris).

■ **Linutop** dévoile Linutop 4, un mini-pc Linux. Le Linutop 4 est un PC miniature économique (<14 watts) et prêt à l'emploi. Compact, puissant avec un Processeur Intel ATOM à 1.6 GHz fanless (sans ventilateur), il est particulièrement adapté pour un poste de travail sans maintenance, un kiosque d'accès à Internet ou pour de l'affichage dynamique. Le système d'exploitation du Linutop 4 s'appuie sur la dernière version 'Lucid Lynx' d'Ubuntu et est fourni avec les logiciels préinstallés Linux : Firefox, Open Office et VLC Media player. A partir de 280 euros HT.

■ **Intel** a annoncé des transistors dotés d'une structure tridimensionnelle, baptisés Tri-Gate et présentés pour la première fois en 2002, dans un microprocesseur (nom de code Ivy Bridge) en production de grande série et gravé en 22 nanomètres. Ces transistors tridimensionnels représentent l'abandon de la structure planaire bidimensionnelle qui est à la base de tous les ordinateurs… Ils devraient permettre de nouvelles économies d'énergie et un empilement des composants.



Donner du design à son boîtier !

Vous voulez faire baver d'envie vos ami(e)s geeks ? Fractal Design propose toute une gamme de boîtiers PC, de la mini-tour à la full tour. Le dernier-né est le Core 1000, une mini-tour très sobre et assez esthétique, capable d'accueillir 3 ventilateurs, 2 ou 3 disques, 2 USB.

Le constructeur est une valeur sûre de la sobriété. Pour des boîtiers plus geek, plus fun, optez plutôt pour la marque Antec et son boîtier Skeleton.

Une approche originale : plateaux de composants et boîtier entièrement ouvert… avec une ventilation optimale. Antec propose aussi un boîtier totalement modulaire le LanBoy Air.

Il s'adapte réellement à vos besoins et vous pouvez parfaitement agencer l'intérieur du PC.

Le tuning PC demeure une vogue notamment chez les gamers, sans oublier le watercooling. Là, vous aurez l'embarras du choix, sur les formes et aussi les prix qui peuvent parfois dépasser les 4000 – 5000 euros pour des configurations complètes. Dans les boîtiers de watercooling (dépassant souvent les 1000 euros) le choix est impressionnant : LittleDevil Cooling, Corsair, aerocool. Ensuite c'est une question de goût et d'imagination…

RIM peut-il sauver le Blackberry ?

RIM, le constructeur du Blackberry, doit trouver des solutions pour conserver ses marchés et bloquer les coupes sombres que réalisent Apple et Google. RIM a dévoilé fin avril toute une série d'outils, de kit de développement pour intéresser les développeurs, et in fine, le marché. Voici ce qu'il faut retenir :

- Blackberry WebWorks SDK 2 pour TabletOS et smartphone : cette version introduit de nombreuses nouveautés comme la séparation des API Javascript vis-à-vis du système, une nouvelle architecture pour simplifier le développement, mise en open source des API (disponible sur github). Ce kit repose sur les standards du web (html, css, javascript).
- Blackberry TabletOS SDK pour Adobe Air : version dédiée à la technologie AIR d'Adobe. Il est possible de développer des applications pour le PlayBook directement depuis le Flash Builder.
- Lancement d'une solution entreprise autour des tablettes et smartphones. Il s'agit des serveurs BES et BES Express. Ces serveurs vont pouvoir gérer la sécurité d'Android, iOS notamment.
- RIM propose depuis peu de nouveaux terminaux (gamme Bold). Ils intègrent les technologies NFC, la réalité augmentée, la vidéo haute définition, Facebook.
- RIM intègrera le moteur de recherche Bing

La tablette de RIM, la Playbook n'est pas encore officiellement disponible en France et aucune date n'a été donnée. La concurrence va être dure pour cette tablette surtout avec l'arrivée de HP et de son offre WebOS et une nouvelle version de la tablette Android de Motorola dans les prochains mois.

GROUPE

MCNEXT

100% Microsoft
4 pôles pilotés par des experts
100 collaborateurs
40% de croissance par an

Formation
Consulting
Expertise
Développement
Intégration
Pilotage de projets
Design & ergonomie

Lancez-vous !
Rejoignez-nous

Projets .NET / C# / ASP.NET MVC
WPF / WCF / Silverlight / SCRUM

DotNET - SharePoint - Décisionnel - BizTalk

Un même langage pour réussir ensemble

Gold Partner - Paris - Genève
mcnext.com - 01 49 70 81 33

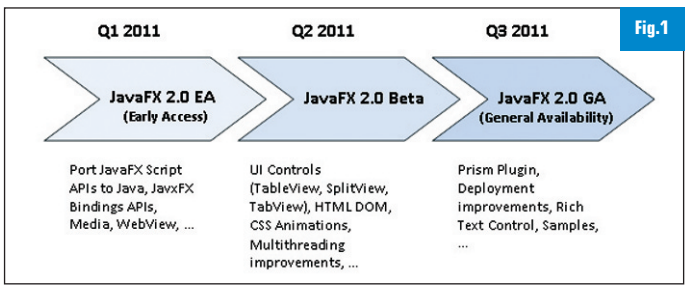
JavaFX 2.0 : Un changement de stratégie majeur

Annoncé en grande pompe par Sun dès 2007 lors de la grand-messe des développeurs Java, JavaFX devait être la plateforme qui permettrait à Sun de s'imposer dans le monde des RIA en venant concurrencer Adobe et Microsoft dans ce domaine. Las ! depuis lors, JavaFX accumule les déconvenues et ne convainc pas la communauté Java. Nouveau propriétaire de la plateforme, Oracle a repris les choses en main en annonçant un virage stratégique pour la future version 2.0 de JavaFX. Opération de la dernière chance pour une technologie qui s'annonce prometteuse pour le monde Java. Tour d'horizon.

Annoncée lors de JavaOne 2010, la version 2.0 de JavaFX marque un tournant pour la plateforme avec l'abandon du langage JavaFX Script et le retour au langage Java. Le but recherché étant avant tout de séduire à nouveau les développeurs Java afin de permettre à la plateforme JavaFX de décoller enfin. Conscient des erreurs de marketing et de timing commises par Sun lors du lancement de JavaFX, Oracle définit précisément une roadmap pour la release 2.0 [Fig.1].

Le projet de la 2.0 semble dirigé de manière agile avec une deadline connue et prévue pour le début du 3e trimestre de 2011 et un ensemble de fonctionnalités prévues mais pas forcément figées. Les équipes d'Oracle concentrent avant tout leurs efforts à fournir une plateforme JavaFX 2.0 100% Java et pleinement fonctionnelle sur le desktop et dans le navigateur [Fig.2]. Cette décision montre une lucidité certaine d'Oracle qui a pleinement conscience du retard pris par JavaFX et plus généralement par la plateforme Java en tant que telle dans l'univers mobile. Petit bémol, cette version ne sera disponible que sur Windows au moment de la sortie de la version stable, ce qui est encore preuve d'une volonté évidente d'éviter de se disperser et de concentrer pour le moment tous les efforts des équipes de développement sur la richesse fonctionnelle de la plateforme. Cet effort centré sur Windows met à mal le slogan historique de Java : « Write once, Run anywhere » !

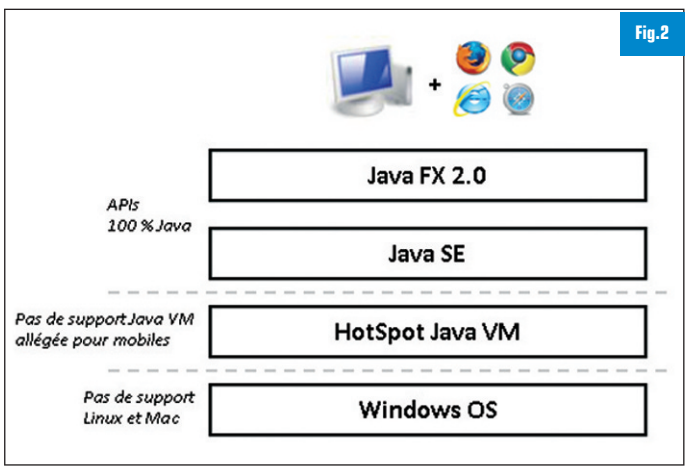
Enfin, dans cette volonté de pragmatisme et de transparence, on notera toutefois qu'Oracle peine à communiquer précisément quelle sera la licence de la plateforme JavaFX 2.0, même si la plupart des observateurs de la communauté Java se doutent que celle-ci ne sera pas de type open-source. Seule certitude sur ce point à l'heure actuelle : les contrôles graphiques seront bel et bien open-source. Ce n'est pas suffisant pour beaucoup mais cela permettra à minima aux développeurs tiers de venir enrichir les composants graphiques fournis en standard par l'API et d'améliorer ainsi la richesse fonctionnelle de la plateforme. Dernier point négatif, il concerne l'accès à l'Early Access de JavaFX 2.0. Dire que ce dernier se révèle compliqué est un euphémisme ! En effet, le JavaFX Partner Program lancé par Oracle pour donner accès à cette version EA est apparemment réservé aux entreprises souhaitant investir dans la plateforme en développant des applications réelles avec, ou à quelques rares têtes d'affiche de la communauté Java. Bref, difficile dans ces conditions de favoriser l'adoption de JavaFX 2.0 par le plus grand nombre dès maintenant.



Roadmap JavaFX 2.0

Du 100% Java

JavaFX 2.0 se présente sous la forme d'une simple bibliothèque jfxrt.jar qu'il suffit d'ajouter au classpath de son application Java comme on le fait classiquement. Le portage des différentes API de JavaFX 1.3 en langage Java va résoudre de facto un certain nombre de problèmes que les développeurs rencontraient avec le langage JavaFX Script, au premier rang desquels se trouve la gestion du multithreading. En outre, l'utilisation des génériques du langage Java sera également au rendez-vous avec la nouvelle API. Mieux encore, ce portage permet de conserver l'approche innovante amenée par l'API de JavaFX 1.3 pour la construction d'interfaces graphiques. Au centre de celle-ci se trouve la notion de Stage symbolisée par la classe éponyme contenant une Scene qui sera rendue graphiquement [Fig.3]. Cette dernière est composée d'objets Node



Architecture plateforme JavaFX 2.0

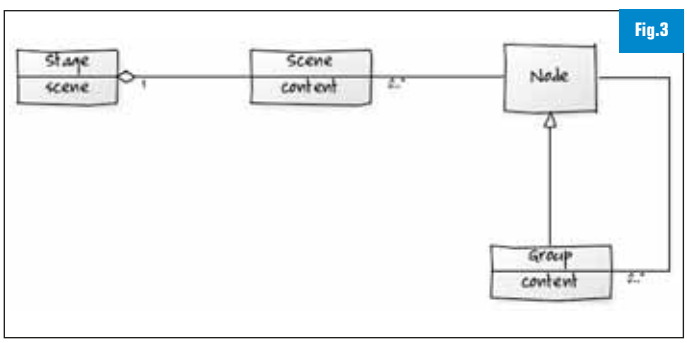
ou de regroupements d'objets de type Node qui correspondent au contenu de l'interface graphique rendue à l'écran.

Notre premier exemple de mise en œuvre de ces classes centrales de l'API de JavaFX 2.0 va consister en un classique « HelloWorld » affiché dans une interface graphique :

```
public class Main extends Application{

    @Override public void start() {
        // Création d'un objet Stage
        Stage stage = new Stage();
        stage.setTitle("Hello Java FX 2");
        stage.setResizable(false);
        // Création d'un regroupement de nodes
        Group root = new Group();
        // Création de la scène de notre stage
        Scene scene = new Scene(root,80,20);
        // Récupération du contenu du groupe via la classe Sequence
        Sequence<Node> children = root.getChildren();
        children.add(new Label("Hello Programmez !"));
        // On installe la scène au sein de notre stage
        stage.setScene(scene);
        // Affichage de notre stage
        stage.setVisible(true);
    }

    public static void main(String[] args) {
        // Utilisation du launcher JavaFX 2
        Launcher.launch(Main.class, args);
    }
}
```

La nouvelle API JavaFX 2.0 étant désormais packagée au sein d'un jar, le lancement de ce programme se fait de manière classique comme suit : java –cp jfxrt.jar fr.sample.jfx2.Main. Cet exemple montre également que les séquences introduites avec JavaFX Script sont portées en Java dans la nouvelle API. Pour ce faire, la classe Sequence a été introduite et se comporte comme un objet de type List dont on peut observer le contenu. De même, la release 2.0 se verra ajouter un type Map dont le contenu sera observable.
Innovation majeure du langage JavaFX Script, le binding sera porté dans la nouvelle API Java via une classe PropertyBinding. Son invocation sera paresseuse pour garantir un meilleur niveau de performance au sein des programmes. Pour le moment, l'utilisation de cette fonctionnalité n'est pas bien définie par l'équipe de JavaFX 2.0 mais on s'oriente vers l'utilisation d'une syntaxe de construction utilisant une méthode statique de type bindTo(<property>). Dans le même ordre d'idées, l'API proposera d'observer des pseudo-propriétés de certaines classes via l'utilisation d'inner-classes anonymes pour le moment mais en gardant à l'esprit que les closures de Java 8 devraient rendre très puissante cette fonctionnalité :

Fig.3



Vue simplifiée du coeur de l'API JavaFX 2.0

```
// Création d'un rectangle
Rectangle rectangle = new Rectangle();
rectangle.setX(10);
rectangle.setY(10);
rectangle.setWidth(60);
rectangle.setHeight(80);
// Mise en place de l'observation de la pseudo-propriété Hover
rectangle.addChangeListener(Recangle.HOVER, new ChangeListener() {
 public void handle(Bean bean, PropertyReference pr) {
    // On change la couleur de remplissage du rectangle
   rectangle.setFill(rectangle.isHover() ? Color.RED : Color.
BLUE);
}
});
```

Le passage à une API Java va ouvrir la possibilité aux développeurs de l'éco-système Java d'utiliser leur langage de script favori tournant sur la JVM pour la construction d'interfaces graphiques JavaFX 2.0. Ainsi, plutôt que de tenter d'imposer l'apprentissage d'un nouveau langage de script aux développeurs, Oracle mise sur la liberté offerte d'utiliser son langage favori afin d'utiliser la nouvelle API et également de tirer parti de la concision offerte par ces langages en regard de ce qu'il est possible de faire avec le langage Java actuellement. Enfin, l'avantage majeur du passage à une API 100% Java concerne le support au sein des IDE. Très faible avec JavaFX Script, il devient de facto excellent avec Java que ce soit sous Eclipse, NetBeans ou IntelliJ IDEA notamment. Plus de problèmes de complétion, de débuggage ou de tests unitaires avec cette nouvelle release. La productivité du développeur n'en sera que grandement améliorée !

Améliorations graphiques

A une époque où ses principaux concurrents sont depuis longtemps passés au rendu vectoriel pour leurs frameworks d'interfaces graphiques, la plateforme Java se contente du rendu bitmap qu'offrent les bibliothèques Swing et AWT. Conscients du problème, les développeurs de JavaFX ont planché sur une nouvelle pile graphique permettant un rendu vectoriel. Il faudra cependant prendre garde en cas de mélange de composants JavaFX et Swing. Ce dernier ayant un rendu bitmap, on se retrouverait alors avec une interface graphique vectorielle contenant des composants bitmaps. Pas vraiment l'idéal …
Nommée Prism, cette pile apporte une accélération graphique hardware aux interfaces graphiques créées à l'aide des API de JavaFX 2.0. Elle utilise DirectX sur les plateformes Windows et utilisera OpenGL sur les autres systèmes lorsque JavaFX 2.0 offrira un support pour ces derniers. Pour les machines ne disposant pas de cartes graphiques suffisamment puissantes, le moteur de rendu graphique supportera un rendu logiciel. Les principaux apports de Prism seront donc concentrés sur le rendu client et n'auront pas d'impact sur le développement de codes 3D. Cependant, un espoir

Les Rapports DevExpress

PRESENTATION CONTROLS | REPORTING CONTROLS
BUSINESS APP FRAMEWORKS | IDE PRODUCTIVITY TOOLS

XtraReports Suite

Cross-Platform .NET Support

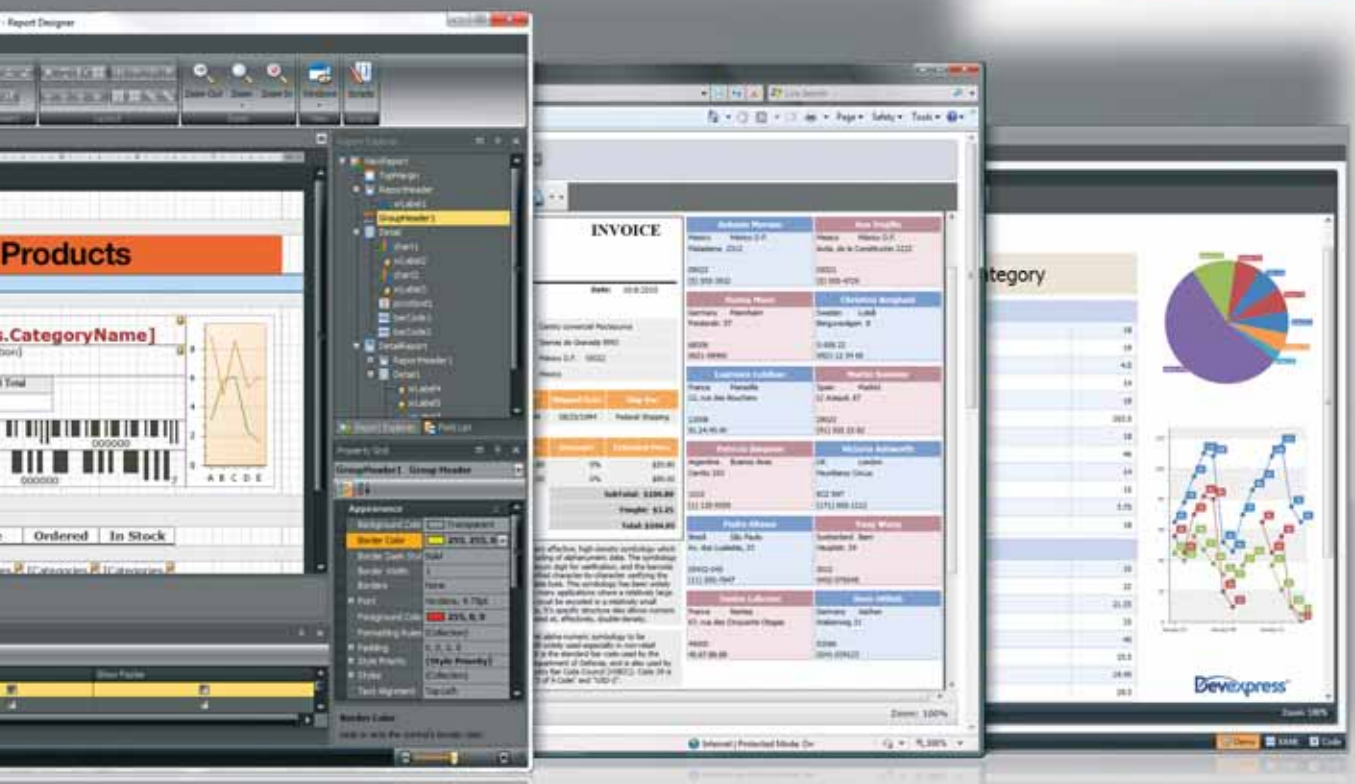
DevExpress crée des contrôles de présentation aux fonctionnalités complètes, des systèmes de rapports, des outils de productivité pour IDE et des frameworks d'applications Business pour Visual Studio .Net. Nos technologies vous aident à construire ce qu'il ya de meilleur, à avoir une vision plus claire des logiciels complexes, à améliorer votre productivité et à créer, dans le temps le plus court, des applications étonnantes pour Windows et pour le Web.

XtraReports Suite, de DevExpress, est une plateforme de Reporting de nouvelle génération pour Visual Studio. Elle vous permet de créer très rapidement des rapports professionnels, au pixel près, ciblant toutes les plateformes majeures Windows, incluant **WinForms**, **ASP.Net**, **Silverlight**, **WPF**. Au-delà de son ensemble de fonctionnalités de niveau professionnel, XtraReports Suite est livré avec un Designer de Rapports ergonomique pour l'utilisateur final, vous permettant de répondre avec la plus grande flexibilité aux exigences de Reporting.

Téléchargez aujourd'hui votre version gratuite d'évaluation et faites l'expérience de la Différence DevExpress.

www.DevExpress.com/Reporting

WIN ASP SL WPF



DevExpress™

WWW.DEVEXPRESS.COM

existe pour cette release 2.0 puisqu'à entendre certaines déclarations de l'architecte en charge du projet JavaFX chez Oracle, des fonctionnalités permettant la création de primitives 3D, la transformation de celles-ci mais également leur animation devraient être présentes.

Fonctionnalités medias avancées

Qui dit framework RIA, dit fonctionnalités médias évoluées. Au vu du niveau de maturité atteint par des concurrents comme Silverlight et Flex dans le domaine, les fonctionnalités proposées par JavaFX 2.0 peuvent être qualifiées de minimum vital pour exister dans le monde impitoyable des RIA. Ainsi, une nouvelle pile media est fournie dès la version EA de JavaFX 2.0.

Réécrite entièrement pour garantir une stabilité et des performances optimales, elle supporte la lecture média dans la plupart des formats de base en ayant une consommation CPU acceptable. La lecture de sons audio en latence très faible est améliorée et sera désormais utilisable pour des évènements graphiques, ou plus intéressant encore, dans des jeux. Au niveau vidéo, le mode plein écran est supporté. En outre, l'API media permettra d'interagir de manière très fine avec les contenus diffusés puisqu'il sera possible de leur associer des markers de manière programmatique. Ces derniers déclenchant des évènements lors de la lecture du flux vidéo, ce qui rend possible le déclenchement de certaines actions à un moment donné d'un contenu média. Enfin, l'API permet la synchronisation entre la lecture d'un flux média et les différentes animations réalisables au sein d'une application JavaFX 2.0 .

Rapprochement Web

Larry Ellison a bien vu vers où le vent tournait et si certaines de ses annonces, comme celle promettant un rendu des applications JavaFX entièrement en HTML 5, ne sont pour l'instant pas des réalités pour JavaFX 2.0, il n'en reste pas moins qu'Oracle a décidé d'effectuer un grand rapprochement entre la plateforme et le Web.

Le fameux JWebPane que les développeurs Swing ont espéré et attendu pendant des années semble enfin arriver avec la plateforme JavaFX et son WebEngine.

Ce dernier est une implémentation Java des technologies HTML permettant le parsing de contenu HTML et la production d'un arbre DOM correspondant. Ce WebEngine permet d'interagir avec les différents évènements pouvant intervenir lors du parsing d'une page HTML. En entrée, il peut accepter une URL, un fichier ou bien même une chaîne de caractères représentant du code HTML.

Afin de faciliter l'intégration de ce WebEngine avec les interfaces graphiques JavaFX, un objet Node spécifique a été développé. Nommé WebView, il rend en son sein le contenu obtenu via un WebEngine. Une implémentation de l'API DOM ayant été réalisée au sein de JavaFX, il est possible pour le développeur d'interagir programmatiquement avec le contenu de l'arbre DOM rendu au sein de la WebView.

Il est donc possible de modifier cet arbre mais également de mettre en place des listeners pour être averti des différentes actions faites sur le contenu web affiché dans l'interface graphique d'une application. L'affichage de la page web de Programmez! au sein d'une application JavaFX 2.0 peut se faire de la sorte :

```
public class Main extends Application {
    @Override public void start() {
        Stage stage = new Stage();
        stage.setTitle("Hello WebView");
        stage.setResizable(false);
        Group root = new Group();
        Scene scene = new Scene(root, 500, 500);
        Sequence<Node> children = root.getChildren();
        // Ajout de la web view à notre Scene
        children.add(getWebView());
        stage.setScene(scene);
        stage.setVisible(true);
    }

    private WebView getWebView() {
        URL url = null;
        try {
            url = new URL("http://www.programmez.com");
        } catch(Exception exc) {
            exc.printStackTrace();
        }
        WebEngine webEngine = new WebEngine(url);
        return new WebView(webEngine);
    }

    public static void main(String[] args) {
        Launcher.launch(Main.class, args);
    }
}
```

Le support CSS est également soigné au sein de JavaFX 2.0 avec le support de la mise en forme pour un layout basé sur une grille notamment. Les contrôles graphiques seront stylables via CSS également et il sera possible de définir des animations CSS sur ces derniers. Ainsi pour un bouton, le développeur pourra mettre en forme une transition entre ses différents états lors d'un évènement hover directement en CSS.

Contrôles standard

La forme d'une plateforme proposant un framework d'interfaces graphiques ne réside pas seulement dans ses possibilités techniques brutes mais également, et surtout serait-on tenté de rajouter, dans sa richesse fonctionnelle. En effet, comment espérer imposer et pérenniser en entreprise une technologie ne garantissant pas un niveau de richesse graphique minimum. La réponse est simple, c'est impossible ! Pour combler les manques remontés sur les précédentes versions de JavaFX, Oracle a réalisé un effort certain sur la présence en standard de plusieurs contrôles graphiques essentiels à la sortie de la nouvelle mouture. Parmi ces composants haut niveau essentiels on citera ainsi :

- TableView, un contrôle permettant l'affichage dynamique de larges ensembles de données hétérogènes. Il vient compléter les contrôles du même type ListView et TreeView portés des anciennes releases de JavaFX.
- SplitView, un contrôle s'adressant à des conteneurs et qui permet de diviser une vue au sein de laquelle on ajoute des conteneurs.
- TabView, qui est le pendant du JtabbedPane de Swing et est utilisé pour séparer une interface graphique en différents onglets.
- MediaPlayer, un contrôle customisable en CSS et fourni en standard qui offre aux applications un player média complet.

- WebBrowser, un contrôle permettant de bénéficier en standard d'un navigateur web au sein de ses applications.
- RichText, un contrôle amenant une zone de texte riche en standard à la plateforme.

L'utilisation du contrôle TableView combinée aux différentes nouveautés de JavaFX 2.0 comme les listes observables va permettre de rendre une table de données comme suit :

```
public class Main extends Application {
    @Override public void start() {
        Stage stage = new Stage();
        stage.setTitle("Hello TableView");
        Group root = new Group();
        Scene scene = new Scene(root);
        Sequence<Node> children = root.getChildren();
        // Ajout de notre TableView
        children.add(getTableView());
        stage.setScene(scene);
        stage.setVisible(true);
    }

    private TableView getTableView() {
        ObservableList<Person> data = FXCollections.<Person>
sequence(
                    new Person("Sylvain", "Saurel"),
                    new Person("Rachida", "Saurel"));
        // Création de notre TableView contenant des objets
Person
        TableView<Person> tableView = new TableView<Person>();
        // Ajout des données
        tableView.setItems(data);
        // Création des colonnes et des propriétés qu'elles
observent
       TableColumn firstNameCol = new TableColumn("FirstName");
        firstNameCol.setProperty(Person.FIRST_NAME);
        TableColumn lastNameCol = new TableColumn("LastName");
        lastNameCol.setProperty(Person.LAST_NAME);
        // Ajout des colonnes
        tableView.getColumns().addAll(firstNameCol, lastNameCol);
        return tableView;
    }

    public static void main(String[] args) {
        Launcher.launch(Main.class, args);
    }
}

public class Person implements Bean {
    private String firstName;
    private String lastName;

    public Person(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }
    // …
}
```

L'utilisation d'une liste observable comme source de données pour l'objet TableView va assurer qu'en cas de modification du contenu de la liste, le contenu de la table sera mis à jour automatiquement. En outre, on remarque ici l'utilisation de l'observation des pseudo-propriétés afin que le contenu de la table soit mis à jour de manière transparente lorsque la propriété désignée de l'objet correspondant est modifiée.
Enfin en parallèle à ces contrôles, des boîtes de dialogue standard seront ajoutées permettant d'afficher des messages à l'utilisateur mais également de proposer une boîte de dialogue interagissant avec le système de fichier client.

Déploiement

Le déploiement au sein du navigateur était le gros point noir de JavaFX dans les précédentes releases. En effet, l'utilisation de la technologie des applets pour réaliser un déploiement au sein du navigateur n'est pas pour convaincre les développeurs et les utilisateurs de la puissance et de la modernité de la plateforme. La comparaison du temps de chargement d'un programme Java lancé via une applet avec une application RIA basée sur Flex par exemple est clairement rédhibitoire pour JavaFX.
Le problème des applets vient en grande partie du plugin associé aux différents navigateurs pour leur lancement. Ce dernier utilise AWT, ce qui pénalise considérablement ses performances et son empreinte mémoire.
Le nouveau moteur de rendu Prism semble être la lumière au bout du tunnel qui permettra aux applications JavaFX d'être lancées via le navigateur de manière performante. Pour ce faire, le plugin associé au navigateur sera entièrement réécrit afin de ne plus avoir recours à Swing ou AWT mais uniquement à Prism. En outre, le support total de l'accélération graphique hardware par Prism laisse également entrevoir des possibilités intéressantes pour les applications JavaFX via le navigateur. En attendant de voir cette fonctionnalité disponible lors du lancement final de la version 2.0, il faudra se contenter de lancer les applications JavaFX en passant par des jars exécutables sur le poste client ou bien par des fichiers JNLP dans le navigateur.

Conclusion

La décision d'abandonner JavaFX Script est une bonne chose. L'ouverture aux multiples langages tournant sur la JVM qui en découle devrait permettre d'intéresser un plus grand nombre de développeurs à la plateforme. Pour les déçus de cet abandon, le projet open source Visage, qui reprend le développement du langage et tente de l'adapter à JavaFX 2.0, devrait être une solution efficace de remplacement.
Si la roadmap ambitieuse dévoilée par Oracle pour l'année 2011 est tenue, gageons que les plus critiques des observateurs sauront redonner une chance à la plateforme en dépit du fait qu'elle ne s'adresse qu'au poste client pour le moment.
Les premiers retours de la communauté Java sont plutôt positifs tout en restant mesurés et les apports au langage Java des futures versions du JDK devraient booster la plateforme JavaFX. En dépit de ces signes encourageants, force est de constater que le chantier s'annonce énorme…

■ Sylvain **Saurel** – *Ingénieur d'Etudes Java / JEE*
sylvain.saurel@gmail.com

Perdu dans le Cloud Computing ?...

Retrouvez notre brochure
dans ce numéro

Parlons-en au : 0972 104 746 (coût d'un appel local)
ou rendez-vous sur : ovh.com/cloudcomputing

Erratum : « Votre domaine à partir de : 4,99 € HT/an, soit 5,97 € TTC/an. »

ADSL | SDSL | Domaines | Emails | Hébergement | Private Cloud |

Navigateurs web modernes : vers une accélération matérielle

HTML5 et ses amis permettront dans le futur l'écriture d'une nouvelle génération d'applications très riches visuellement. Il faudra alors faire très attention à l'accessibilité de ces nouvelles applications. Mais il faudra aussi que les navigateurs modernes soient capables d'en rendre la complexité avec la plus grande fluidité possible.

Pour cela, les navigateurs devront être capables d'exploiter au mieux les ressources de la machine, que ce soit un PC ou un périphérique portable comme un téléphone ou une tablette. C'est avec cet objectif en tête qu'Internet Explorer 9 a été conçu depuis le début. Il est donc capable d'exploiter la puissance de traitement du GPU présent sur une machine tournant sous Windows Vista ou Windows 7 (et bientôt également de tous les GPU présents dans les téléphones Windows Phone 7).

Mettre en place ce que l'on appelle donc ***"l'accélération matérielle"*** semble donc indispensable si l'on souhaite pouvoir exploiter au mieux les merveilleuses promesses d'HTML5. L'ensemble de l'industrie semble d'ailleurs d'accord sur ce point. Cependant, la mise en œuvre technique de celle-ci n'est pas chose aisée. Ainsi, l'accélération matérielle proposée par les différents acteurs des navigateurs ne se valent pas. C'est ce que nous allons voir ensemble ici.

1 PROTOCOLE, MACHINE ET NAVIGATEURS UTILISÉS POUR LES TESTS

L'ensemble des tests ont été réalisés sous Windows 7 SP1 64 bits. La machine utilisée est un Sony Vaio Z (VPCZ13Z9E). Cette machine est équipée d'un Core i7 640m et de 2 GPU (nVidia GT330m et le GPU Intel HD Graphics intégré). La version des drivers est également importante pour bénéficier de l'accélération matérielle dans les derniers navigateurs. Voici donc la version des drivers nVidia : 263.14 (8.17.12.6314) et Intel : 2281 (8.17.12.6314) utilisés. Nous allons en effet nous servir des 2 GPU dans les tests pour que l'on puisse constater du bénéfice obtenu en fonction de la puissance sous-jacente. Voici les versions des navigateurs retenus pour ce test : Microsoft Internet Explorer 9 (version finale), Mozilla Firefox 4.0 (version finale), Google Chrome 12.0.712.0 canary build, Opera 11.50 labs (build 24661). Safari n'utilisant pas à ma connaissance d'accélération matérielle.

Note : nous avons tenté ici de comparer de manière la plus loyale possible les différents navigateurs. C'est donc avec cette démarche en tête que j'ai retenu les toutes dernières versions des navigateurs concurrents en activant ou forçant l'accélération matérielle lorsqu'elle n'était pas proposée par défaut.

Méthodologie pour analyser l'occupation GPU

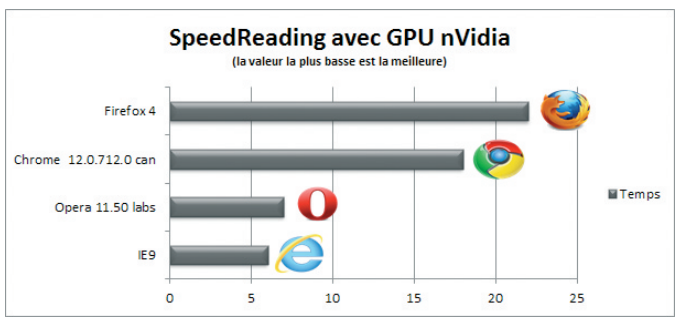
Pour finir sur le protocole retenu, sachez qu'à chaque fois, un seul navigateur était ouvert et aucune autre application ouverte en parallèle (à part les outils de mesure). Par ailleurs, l'utilitaire GPU Z fut utilisé pour permettre d'analyser en temps réel le taux d'occupation du GPU. Pour aller encore plus loin sur l'analyse de l'utilisation GPU, j'ai utilisé les outils du SDK Windows, notamment GPUView. On peut ainsi voir en détail la charge du GPU ainsi que le FPS (Frame Per Second) dans le temps associé à un processus particulier. Ce sont des outils en général utilisés pour débugger les problèmes de performances dans les applications 3D (DirectX ou XNA). Mais ils sont ici particulièrement intéressants pour mettre en lumière les différences de performance entre les navigateurs.

2 BENCHMARKS SUR MICROSOFT IE TEST DRIVE

Commençons par regarder quelques résultats sur le site maintenu par Microsof t: http://ie.microsoft.com/testdrive

Speed Reading avec GPU nVidia

Voici le tableau des résultats avec le GPU nVidia GT330m et le taux moyen d'utilisation pour chacun des navigateurs du GPU en cours d'exécution sur l'application Speed Reading. Le but du jeu est d'afficher, à travers l'utilisation de la balise <canvas>, l'ensemble des caractères le plus rapidement possible. La valeur la plus basse est donc bien évidement la meilleure :



Navigateur	IE9	Opera 11.50 labs	Chrome 12.0.712.0 can	Firefox 4
Temps	6s	7s	18s	22s
Taux d'utilisation moyen (GPU-Z)	36%	75%	31%	43%
FPS moyen (GPU View)	60	32	15	31

Cette première série de résultats est intéressante. On voit ainsi qu'Opera 11.50 et IE9 sont quasiment ex-aequo. IE9 et Opera 11.50 labs sont également du coup plus de 3 fois plus rapides que Firefox 4 et plus de 2 fois plus rapides que Chrome sur ce test. Chapeau bas donc à Opera pour cet excellent travail ! Cependant, on voit aussi qu'Opera peut saturer le GPU jusqu'à quasiment 100% pour atteindre le même niveau de performances qu'IE9.

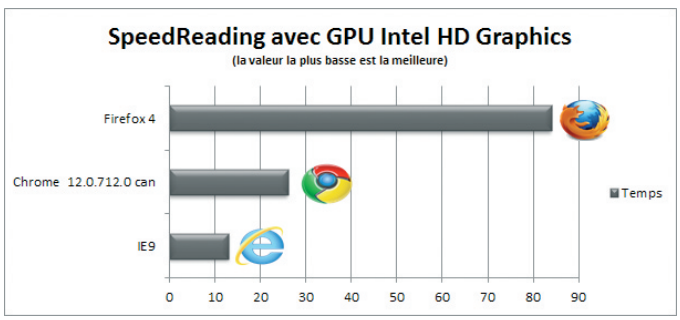
Cela veut donc dire que :

1 – IE9 en a potentiellement encore sous la pédale

2 – La version IE9 pour les périphériques mobiles permettra d'avoir d'excellentes performances sans pour autant forcément saturer le GPU… pour le bénéfice de l'autonomie ! Cela semble d'ailleurs déjà se confirmer sur les PC portables où l'on peut constater des autonomies différentes entre IE9 et la concurrence. Cette différence d'autonomie sera bien évidement plus ou moins visible en fonction du type de pages consultées. Cela veut dire aussi au passage qu'IE9 devrait être capable aussi de bénéficier de GPU moins puissants pour déjà vous apporter un confort inédit. C'est ce que nous allons justement voir après. On peut également noter sur les courbes GPU View des différences très importantes sur le taux FPS affiché par chacun des navigateurs sur ce test. IE9 est largement 1er avec une courbe moyenne oscillant dans les 60 FPS, Firefox 4 et Opera 11.50 sont au coude à coude aux alentours de 30 FPS et Chrome est loin derrière avec environ 15 FPS. Vous pourrez trouver la série de copies d'écran sur le blog IE France montrant les résultats présentés dans ce tableau avec le GPU nVidia. Vous y noterez également l'utilisation GPU-Z et la courbe de FPS de GPUView.

Speed Reading avec GPU Intel

Regardons maintenant le résultat avec le GPU intégré au Core i7 (Intel HD Graphics). Ce GPU dispose d'une architecture nettement moins puissante que celle du GPU nVidia sur le papier. Il est donc intéressant de voir si l'accélération matérielle est déjà intéressante à ce niveau. Voici les résultats :

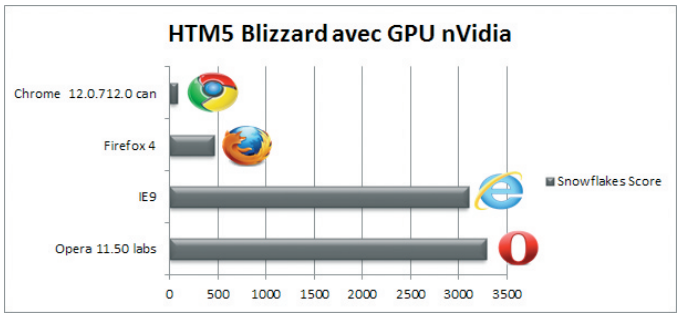


Navigateur	IE9	Opera 11.50 labs	Chrome 12.0.712.0 can	Firefox 4
Temps	13s	2029s	26s	84s

Opera passe malheureusement en rendu "software" avec la puce Intel intégrée. Pour ne pas avoir de problème d'échelle, je préfère donc ne pas intégrer le résultat au graphique car il dépasse alors les milliers de secondes. IE9 devient par contre presque 7 fois plus rapide que Firefox 4 et 2 fois plus rapide que Chrome avec le GPU Intel. Cela devrait donc intéresser tous les propriétaires de Netbook ou de portable d'entrée gamme. D'autant plus qu'IE9 s'occupe également de décompresser la balise vidéo de manière matérielle. Ce premier test semble donc indiquer que même un « petit » GPU peut être utile pour augmenter la vitesse de rendu d'une page web.

HTML5 Blizzard avec GPU nVidia

Regardons maintenant comment se comportent chacun des navigateurs avec la démo HTML5 Blizzard. Cette démo mélange plusieurs technologies comme Canvas, SVG, WOFF et un peu de CSS3. L'objectif de cette démo est de demander au navigateur de maintenir un taux FPS optimal de 60 FPS en adaptant dynamiquement le nombre d'objets affichés simultanément à l'écran. En résumé, meilleure l'accélération matérielle est, plus de flocons vous aurez à l'écran. J'ai donc lancé cette démo sur chacun des navigateurs et attendu suffisamment longtemps pour que le score "Snowflakes" se stabilise. J'ai également noté l'occupation moyenne du GPU ainsi que la mémoire vidéo utilisée par chacun des navigateurs. Sur ce graphique, la valeur la plus haute est donc cette fois-ci la meilleure :

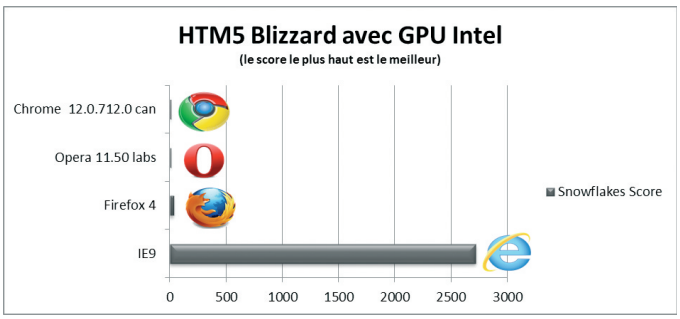


Navigateur	IE9	Opera 11.50 labs	Chrome 12.0.712.0 can	Firefox 4
Score	3101	3280	74	461
Occupation GPU moyenne	45%	72%	74%	83%
Mémoire vidéo utilisée	124MB	276MB	249MB	158MB

Opera 11.50 passe légèrement devant IE9 sur ce test ! Preuve s'il en est que les tests Microsoft ne sont pas spécialement optimisés pour IE 9. Il faut néanmoins tempérer les excellents résultats d'Opera en mettant en perspective son taux d'occupation GPU moyen 1,6 fois supérieur et sa consommation vidéo mémoire plus de 2 fois supérieure à celle d'IE9. Il est à noter aussi qu'Opera ne joue pas la musique en arrière-plan utilisant pourtant la balise <audio> (supportée par Opera) mais avec le codec MP3 (non supporté par Opera). Cela peut donc expliquer ce léger avantage. Malgré tout, IE9 et Opera 11.50 sont néanmoins au coude à coude en performance pure et restent tous les 2 nettement devant Firefox 4.0 et Chrome 12.0.712.0. En résumé, IE9 et Opera 11.50 sont plus de 40 fois plus rapide que Chrome 12.0.712 et plus de 6 fois plus rapide que Firefox 4.0 sur ce test spécifique !

HTML5 Blizzard avec GPU Intel

A nouveau, il est intéressant d'observer comment ces nouveaux navigateurs tirent parti de GPU moins puissants comme ceux d'Intel intégré. J'ai donc relancé le même test sur les 4 navigateurs en basculant sur ma puce Intel. J'avoue que je ne me serais jamais douté du résultat avant ! Regardons cela ensemble. La valeur la plus haute reste toujours ici la meilleure :



Navigateur	IE9	Opera 11.50 labs	Chrome 12.0.712.0 can	Firefox 4
Score	2716	3	3	32

Tout est chamboulé ! IE9 semble assez insensible au changement de GPU et obtient quasiment le même score qu'avec la puce nVidia. Firefox 4 semble capable d'utiliser en partie le GPU Intel mais voit ses performances dégringoler et divisées par presque 15 par rapport au GPU nVidia. Opera passe de la 1ère à la dernière place assez logiquement puisqu'il n'est pas capable d'activer l'accélération matérielle sur la puce Intel. Chrome semble souffrir également de l'incapacité d'exploiter le GPU Intel sur ce test. Pourtant, la version 11 était en meilleure forme. Preuve que c'est encore très expérimental du côté de chez Google.

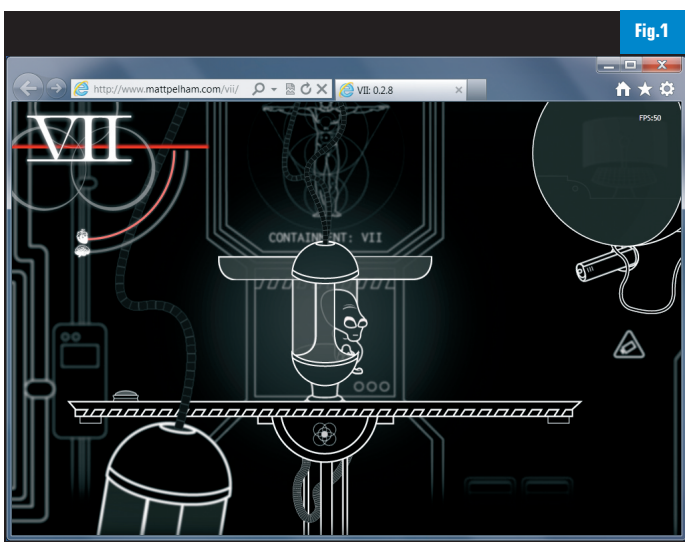
IE9 est donc très loin devant ses concurrents sur ce test lancé sur le GPU Intel. IE9 a en effet des performances respectivement 85 et 905 fois supérieures à Firefox et Chrome dans ces conditions.

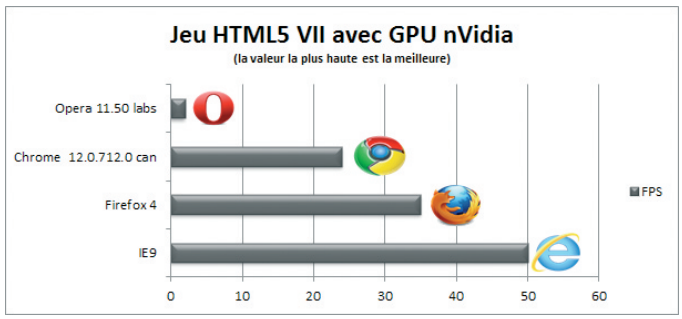
3 BENCHMARKS EXTÉRIEURS À MICROSOFT

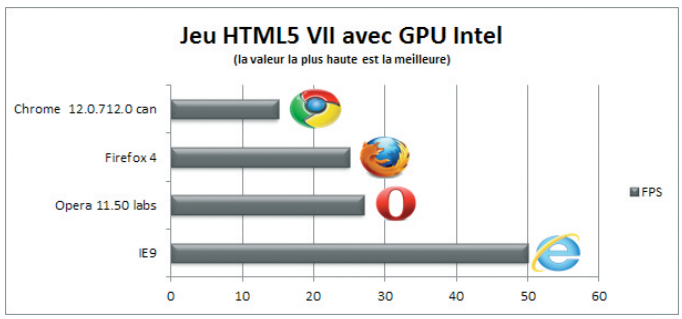
Nous avons vu des tests comparatifs effectués depuis un site maintenu par Microsoft. Même si d'autres navigateurs qu'IE9 s'en sont très bien sortis, il est légitime de se demander comment les mêmes navigateurs se comportent sur des sites non gérés par Microsoft.

Jeu HTML5 VII avec GPU nVidia [Fig.1]

Commençons avec le jeu HTML5 VII : http://www.mattpelham.com/vii . Ce jeu mélange de la gestion de physique et une sorte de jeu de plateforme. J'ai noté la valeur moyenne sur le 1er tableau. Regardons ainsi le FPS indiqué par le jeu avec les GPU nVidia et Intel :





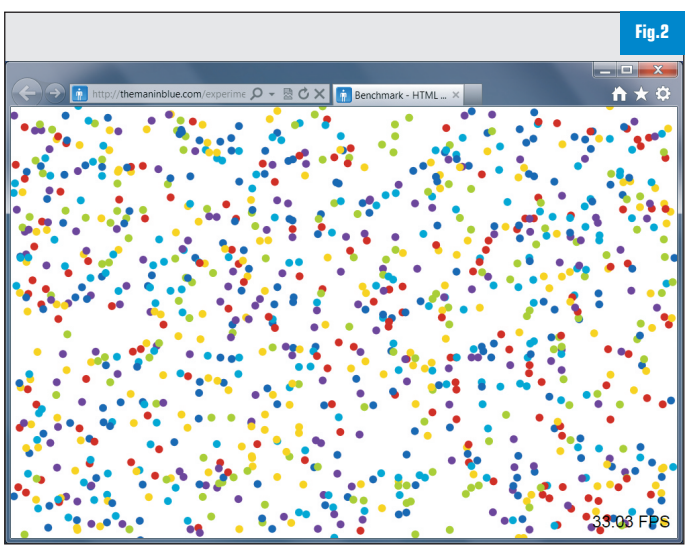


Navigateur	IE9	Opera 11.50 labs	Chrome 12.0.712.0 can	Firefox 4
FPS avec GPU nVidia	50	2	24	35
FPS avec GPU Intel	50	27	15	25

Avec le GPU nVidia, pour une raison obscure, Opera 11.50 rend le jeu totalement injouable avec un FPS ne dépassant pas les 3. IE9 de son côté finit 1er et propose ainsi une expérience de jeu agréable. Firefox plafonne dans les 35 fps, ce qui rend le jeu moins jouable. Chrome est trop limite pour rendre le jeu jouable.

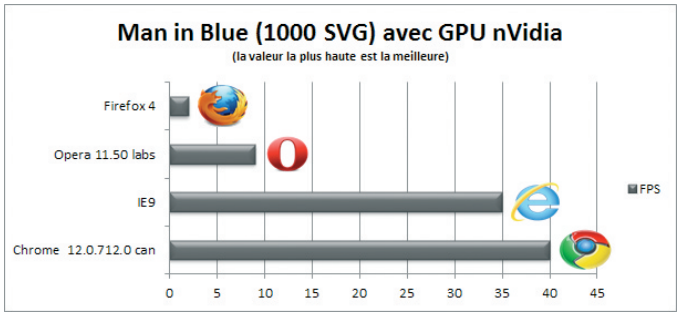
Avec le GPU Intel, IE9 finit toujours largement 1er avec exactement le même framerate qu'en nVidia. Opera 11.50 se comporte mieux en rendu software avec le GPU Intel qu'en mode hardware avec le GPU nVidia. Cela semble indiquer qu'il souffre d'un bug sur ce jeu avec l'accélération matérielle.

Firefox 4 et Chrome 12 descendent nettement d'un cran avec le GPU Intel, ce qui rend le jeu injouable dans les 2 cas. Tout comme l'ensemble des tests précédents, on observe donc à nouveau qu'IE9 exploite nettement mieux les capacités GPU du processeur Intel que ses petits camarades.



The Man in Blue Animation Benchmarking [Fig.2]

Jusqu'à présent, nous avons principalement regardé les performances des navigateurs sur la balise <canvas>. Dans son article "HTML5" versus Flash: Animation Benchmarking, Cameron Adams a créé des benchmarks intéressants utilisant SVG, canvas ou Flash pour voir quelle technologie était la plus adaptée à ses besoins. Vous pouvez les tester ici : http://themaninblue.com/writing/perspective/2010/03/22/. Comme jusqu'à présent nous avons joué avec la balise canvas, j'ai décidé de prendre un benchmark testant SVG. Je suis donc parti de celui utilisant 1000 particules avec les ombres portées activées.

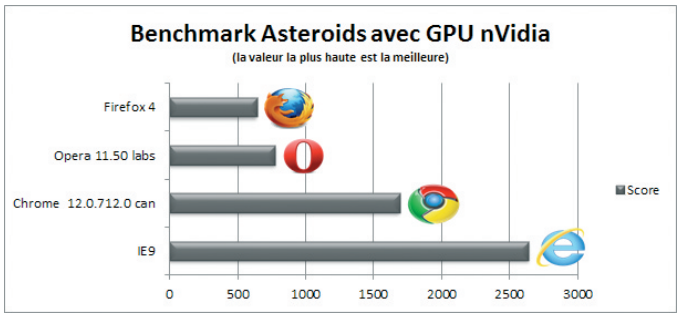


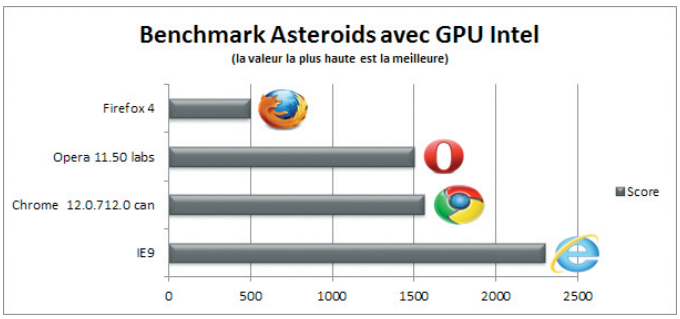
Navigateur	IE9	Opera 11.50 labs	Chrome 12.0.712.0 can	Firefox 4
FPS avec nVidia	35	9	40	2
FPS avec Intel	34	8	39	2

Sauf erreur de ma part, je n'ai pas l'impression que Firefox accélère aujourd'hui le rendu du SVG. Avec 1000 particules, son taux de rafraîchissement descend en effet à moins de 2fps sur ma machine. Chrome par contre passe 1er sur ce test ! J'avais d'abord testé avec 500 particules mais IE9 et Chrome étaient trop proches pour que j'arrive à voir qui gagnait ce match (tous les 2 autour de 75fps). En passant à 1000 particules, Chrome s'est détaché. IE9 reste par contre un excellent élève dans le domaine de l'accélération du SVG. C'est amusant de noter également que les scores sont quasiment identiques entre les 4 navigateurs que ce soit avec la puce Intel ou nVidia. Je n'ai donc publié qu'un seul graphique puisqu'ils sont quasiment identiques. Par ailleurs, il existe d'autres benchmarks SVG où IE9 est devant tout le monde, Chrome compris. Vous pourrez le tester par vous-même. Mais bon, ne faisons pas les mauvais perdants et reconnaissons le meilleur travail de la concurrence sur ce test !

Asteroids HTML5 Canvas 2D Rendering and JavaScript Benchmark [Fig.3]

Voici un petit dernier benchmark avant de conclure : Asteroids HTML5 Canvas 2D Rendering and JavaScript Benchmark. Vous pouvez le tester ici : http://www.kevs3d.co.uk/dev/asteroidsbench/. Celui-ci est dérivé d'un moteur de jeu d'astéroïdes utilisant canvas et le moteur JavaScript du navigateur. Regardons les résultats avec, comme d'habitude, l'usage de mes 2 GPU présents dans mon Vaio Z :





Navigateur	IE9	Opera 11.50 labs	Chrome 12.0.712.0 can	Firefox 4
Score avec nVidia	2638	777	1690	643
Score avec Intel	2301	1502	1562	494

IE9 finit encore une fois 1er à un benchmark mettant à mal les différentes parties de l'implémentation de l'accélération matérielle que ce soit en s'appuyant sur le GPU nVidia ou Intel. La différence de performances entre Firefox 4 et ses concurrents est vraiment très importante sur ce benchmark.

C'est encore plus flagrant lorsque l'on voit les animations à l'écran. Par ailleurs, Opera 11.50 se comporte à nouveau étrangement. Son score est en effet nettement meilleur avec la puce Intel qu'avec la puce nVidia ! Par ailleurs, je vous rappelle au passage que grâce à son nouveau moteur JavaScript nommé Chakra, IE9 est désormais 1er au benchmark JavaScript SunSpider maintenu par les équipes WebKit (moteur de Chrome et Safari) comme le montre ce graphique : [Fig.4].

Mais bon, les benchmarks JavaScript, c'est une toute autre histoire et il y aurait bien matière à écrire également… Mais ne boudons pas pour autant notre plaisir de voir IE9 1er à un test où longtemps Internet Explorer fut montré du doigt comme le vilain petit dernier !

4 CONCLUSION

Vous trouverez certainement beaucoup d'autres tests du même genre sur la toile. D'après ce que j'ai pu voir, IE9 finit souvent 1er ou très proche du 1er. IE9 est donc bien le navigateur actuellement le plus doué pour gérer l'accélération matérielle. Comme vous pouvez le voir avec les différents tests précédents, il propose en effet un résultat très homogène quel que soit le type d'éléments (HTML, canvas, SVG ou vidéo) et quel que soit le GPU qu'on lui propose. A vous maintenant de le vérifier par vous-même ! Vous aurez aussi sûrement remarqué que les différentes implémentations de l'accélération matérielle des navigateurs ne se valent pas.

Avec IE9, il paraît assez évident que Microsoft mène la danse à ce sujet en devançant clairement la concurrence. Nous avons d'ailleurs été les premiers à vous proposer l'accélération matérielle activée par défaut dans une version finale. Je pense aussi sincèrement que Microsoft n'est pas étranger à l'intérêt soudain de tous les éditeurs de navigateurs vers les performances de l'accélération matérielle. C'est un sujet clé pour les années qui arrivent.

Opera semble sur une excellente voie et m'a particulièrement impressionné lors de mes tests. Je garderai donc un œil attentif sur celui-ci ! Reste à voir si tout cela pourra être proposé par défaut et de manière stable. En effet, sur cette version alpha, il y a pour l'instant des comportements bien étranges. Par ailleurs, il ne semble pas capable aujourd'hui de supporter l'architecture d'Intel, pourtant très répandue sur les machines portables.

Mozilla avec Firefox propose aussi l'accélération matérielle activée par défaut mais semble clairement plus à la traîne en termes de performance. Il semblerait d'ailleurs que dans la roadmap qu'ils ont publié au sujet des futures versions de Firefox, l'accélération matérielle du canvas revienne dans le cahier des charges pour Firefox 5. Il reste en effet une très bonne marge de progression sur ce sujet, et c'est tant mieux. Je ne sais pas par contre si l'avenir sera à l'accélération matérielle du SVG ou à la décompression vidéo.

Pour finir, Google avec Chrome s'en sort souvent pas mal mais l'accélération matérielle semble pour l'instant trop instable pour vous la proposer par défaut, ce que j'ai d'ailleurs pu moi-même constater avec quelques « plantages » réguliers. Par contre, Google semble excellent sur SVG et détient également un niveau de stabilité de builds supérieur à Opera.

Internet Explorer 9 accélère donc complètement par défaut tous les éléments du Web y compris la vidéo, le canvas ou du SVG. N'hésitez donc pas à le tester pour vérifier par vous-même le confort qu'il peut vous apporter. Que votre GPU soit une bête de course ou non, IE9 devrait savoir quoi en faire !

David **Rousset**
http://blogs.msdn.com/iefrance

Fig.3

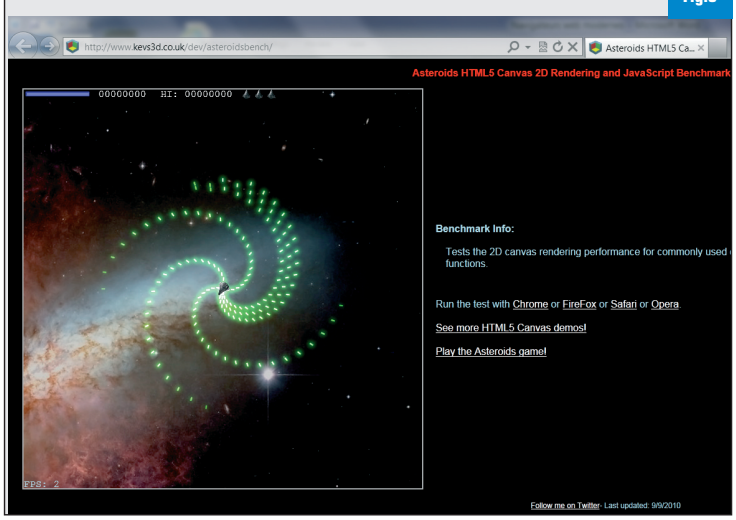
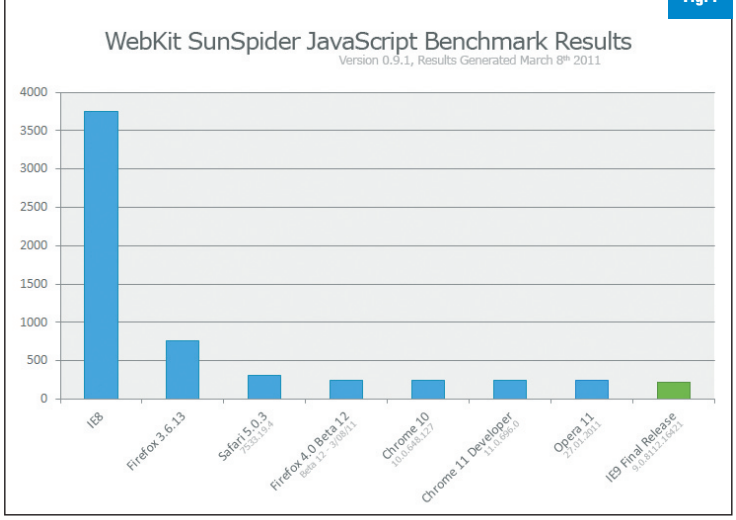
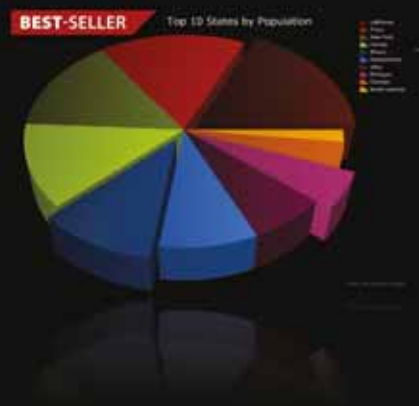


Fig.4





Codejock Xtreme Chart Pro ActiveX | à partir de € 153

CODEJOCK

Incluez des diagrammes dans vos applications ActiveX en quelques lignes de code.

- Affiche un ensemble riche de classes de personnalisation et d'amélioration
- Histogrammes, barres de dispersion, barres empilée, 100% barres empilées et barres horizontales
- Graphiques en secteurs 2D/3D, anneaux 2D/3D et tores 3D, ainsi que les secteurs éclatés
- Inclut les lignes standard, à dispersion, à traçage rapide, en escalier et spline
- Inclut aussi les diagrammes de points, de zones, entonnoir, financiers et de Gantt

Spread for Windows Forms | à partir de € 667

GrapeCity PowerTools

Feuille de calcul pour les applications Windows Forms, compatible avec Microsoft Excel.

- Accélérez le développement avec les concepteurs de feuilles de calcul, l'Assistant de prise en main et les concepteurs de graphiques
- Renseignement automatique : anticipation de la frappe dans la cellule
- Nouveau - outil intégré de création de diagrammes avec 85 styles
- Nouveau - préserve les .XLS et restaure les fonctions non supportées
- Inclut des apparences prédéfinies ainsi que la possibilité de créer des apparences personnalisées

FusionCharts | à partir de € 133

InfoSoft Global

Graphiques Flash & JavaScript (HTML5) interactifs pour les applications Web.

- Animez vos applications Web avec des graphiques interactifs et pilotés par les données
- Créez des graphiques AJAX avec des possibilités d'exploration en quelques minutes
- Exportez les graphiques au PDF et les données en CSV directement depuis les graphiques
- Créez des jauges, des tableaux de bord, des graphique financiers et plus de 550 types de carte
- Adopté par plus de 18 000 clients et 375 000 utilisateurs dans 110 pays

Janus WinForms Controls Suite V4.0 | à partir de € 609

Janus systems

Ajoutez des interfaces de style Outlook à vos applications .NET.

- Vues ruban, grille, calendrier, et barres chronologique/raccourcis
- Nouveau – Style visuel Office 2010 pour tous les contrôles
- Nouveau – Support des profils client Visual Studio 2010 et .NET Framework
- Janus Ribbon ajoute Backstage Menus et la fonctionnalité onglet comme dans Office 2010
- Prend désormais en charge la sélection de cellules multiples

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

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

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

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

Quoi de neuf dans Windows Phone 7.5 ?

Windows Phone 7, la nouvelle plateforme mobile de Microsoft a fait sa première apparition lors du Mix 2010 avec la bêta du SDK destinée aux développeurs. 1 an plus tard et quelques mois après la sortie officielle de WP7 pour le grand public, Microsoft dévoile lors du Mix 2011 les nouveautés apportées par la prochaine mise à jour majeure, intitulée Windows Phone 7.5 (nom de code : « Mango »). C'est parti pour une description des principales nouveautés !

Silverlight 4

Le SDK actuel de Windows Phone 7 est basé sur Silverlight 3 avec certaines fonctionnalités liées à l'environnement mobile. La prochaine version du SDK viendra avec l'apport de Silverlight 4 et les nombreuses améliorations que cette version fournit, comme un moteur de binding plus riche, le support des commandes (très utile pour l'implémentation du modèle MVVM), les styles implicites. En revanche, certaines autres fonctionnalités de Silverlight 4, peu adaptées à un contexte mobile sont absentes, comme la gestion de l'impression, le clic-droit, etc.

Multitasking

Avec la version actuelle du SDK Windows phone 7, il n'est pas possible de développer des applications capables de tourner en tâche de fond, une seule application peut être exécutée à la fois. Afin de supporter les multiples interruptions que peut subir une application (appel, mise en veille, etc.), le SDK propose un mécanisme de « tombstoning » permettant lorsqu'une application est quittée, de sauvegarder certaines informations en mémoire qui pourront être réutilisées lors du prochain lancement de cette même application. Avec Mango, il sera possible dans certains cas d'avoir plusieurs instances en parallèle.

Plus précisément lorsqu'une application sera désactivée, par défaut sa mémoire ne sera pas libérée par le système, ceci permettra une réactivation beaucoup plus rapide que ce qui est proposé à l'heure actuelle. A noter que jusqu'à 5 instances d'applications pourront tourner en même temps sachant que la mémoire occupée par toute application dite « dormante » pourra être libérée par le système si celle-ci venait à manquer, on rentre alors dans le processus de « tombstoning » expliqué précédemment.

Mais le système de multitasking ne s'arrête pas là. Pour aller plus loin, il sera possible d'exécuter du code en tâche de fond grâce à des « agents » qui ne seront pas dépendants de l'état de votre application et qui tourneront dans un processus différent de celle-ci. Ainsi, même si la mémoire de votre application est libérée par le système, un agent continuera à tourner, permettant par exemple à une application de flux RSS de maintenir l'état des flux à jour tout au long de la journée.

Le champ d'action de ces agents est très vaste, néanmoins il y aura certaines limitations comme l'impossibilité de mettre à jour l'UI d'une application ou d'accéder au flux de la caméra et du micro. Un agent ne pourra pas non plus exécuter du code ou télécharger des données en continu à n'importe quel moment, cela pourrait s'avérer très néfaste pour la batterie du téléphone ou votre forfait data. L'état du téléphone (branché ou non au secteur) et de sa connexion (cellulaire ou wifi) aura un impact sur la marge de manœuvre de l'agent.

Il sera également possible d'aller plus loin dans le système de notification. En effet, dans la version actuelle nous avons nativement la possibilité de déclencher de façon différée des alarmes ou de créer via, l'application calendrier, des rappels.

Des API permettront dans la prochaine version d'accéder à ces services. De plus, deux autres services seront à notre disposition pour la lecture de flux audio et le téléchargement de données en tâche de fond. Ces services seront, bien entendu, eux aussi indépendants de l'état de votre application.

Les ressources du téléphone

De nombreuses ressources du téléphone aujourd'hui réservées aux applications natives pourront être accessibles via le nouveau SDK. De nouveaux « launchers » et « choosers » (ces classes qui permettent d'interagir avec le téléphone) seront disponibles pour accéder aux contacts, au calendrier, etc. Il sera également possible de lancer, via l'une de ces classes, l'application Bing Maps pour afficher des itinéraires.

Avec la nouvelle gestion des tâches introduite par Windows Phone 7.5, il deviendra important que votre application puisse notifier l'utilisateur qu'une action ou qu'un évènement spécifique s'est déroulé, même si l'application est en arrière-plan. Pour cela, vous pourrez manipuler des alarmes et des reminders ! De nouveaux capteurs (boussole, gyroscope) seront également disponibles mais, en plus de ces capteurs, on disposera surtout d'un mini « framework » qui permettra aux développeurs de pouvoir les intégrer dans leurs applications le plus simplement possible. Enfin, et c'est sans doute l'une des choses qui faisait partie des demandes les plus fortes de la part des développeurs actuels, l'accès direct à la caméra sera désormais possible ! Libre à vous donc

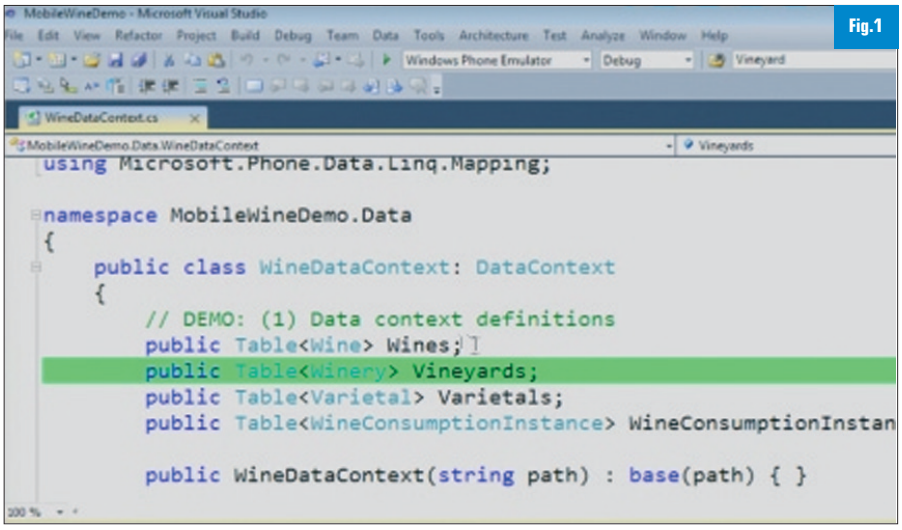


Fig.1

d'imaginer des scénarios de lecture de codes-barres, de réalité augmentée, etc.

Les sockets

Dans la première version du SDK, les développeurs ne pouvaient communiquer avec l'extérieur qu'en utilisant les protocoles HTTP ou HTTPS (pour des connexions sécurisées). Il sera maintenant possible d'utiliser des sockets, avec les protocoles TCP ou UDP, que ce soit en unicast ou en multicast ! Il s'agit là d'un point important car cette implémentation des sockets est plus riche que celle présente dans Silverlight 4, qui devient pourtant la nouvelle version de Silverlight dans Windows Phone 7.5. De plus, tout un nouveau monde de scénarios d'applications devient possible et, pour preuve, l'application Skype a été annoncée et sera donc disponible bientôt.

Les tuiles

Les tuiles sont les fameuses vignettes que l'on retrouve sur la page d'accueil des périphériques Windows Phone 7. Avec la première version du SDK, il était possible de les manipuler, par exemple au moyen de notifications de type « Push ». Cependant, la mise en place était quelque peu complexe et, surtout, on ne pouvait personnaliser que 3 champs bien spécifiques : le texte, un compteur (par exemple pour afficher le nombre de flux RSS non lus, le nombre de mails non lus, etc.) et l'image.
Dans sa nouvelle version, le SDK de Windows Phone dispose d'un nouveau modèle de programmation des tuiles, qui va vous permettre de les modifier directement depuis l'application, si celle-ci est au premier plan, ou bien à l'aide d'un agent (voir la partie précédente sur les nouveautés du multitasking) si celle-ci est en arrière-plan.
Vous allez également avoir la possibilité d'avoir plusieurs tuiles par application (là où, pour le moment, vous ne pouvez en avoir qu'une). Chacune de ces tuiles sera complètement indépendante des autres, ce qui fait que vous pourrez les utiliser pour pointer vers des parties spécifiques de votre application. De plus, vous pourrez à présent accéder aux 2 faces des tuiles, pour les animer à l'écran (et ainsi fournir plus d'informations aux utilisateurs).

Les notifications

Les notifications, que l'on reçoit en push sur le téléphone (grâce au service de notification) ont, elles aussi, été revues. Premièrement, les performances ont été améliorées (pour le bonheur des développeurs et des utilisateurs).
Ensuite, la quantité de données que l'on pouvait envoyer dans une notification a été augmentée : on passe maintenant de 1Ko à 4Ko. De plus, avant, on ne pouvait avoir que 15 applications qui utilisaient les notifications en push sur le téléphone : cette limite passe dorénavant à 50.
Enfin les « toasts », ces notifications que l'on aperçoit en haut de téléphone (comme par exemple lors de la réception d'un message texte), vont pouvoir être paramétrés pour rediriger l'utilisateur vers une page bien précise de l'application.

L'intégration de Silverlight avec XNA

Avec la première version du SDK de Windows Phone, vous n'aviez pas le choix : lors de la création d'un projet, vous étiez obligé de choisir entre une application XNA ou une application Silverlight. Certes, vous

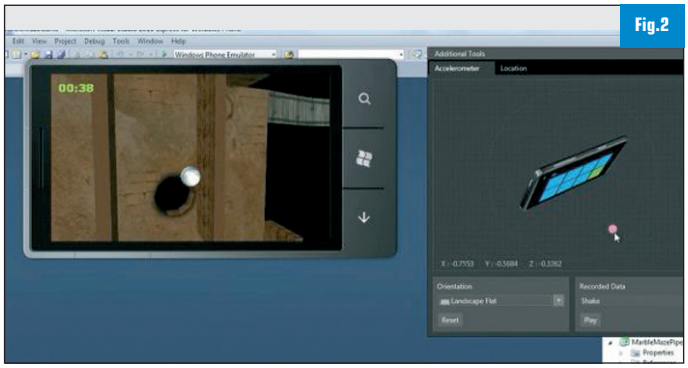


Fig.2

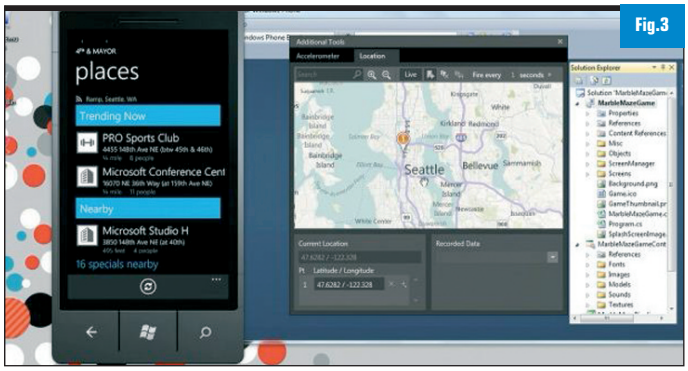


Fig.3

pouviez utiliser des fonctionnalités de XNA (comme par exemple la classe SoundEffect) dans une application Silverlight pour Windows Phone (et vice-versa) mais cela restait très limité. Dans sa nouvelle version, le SDK Windows Phone va beaucoup plus loin. Ainsi, vous avez la possibilité d'utiliser la boucle de jeu XNA (les fameuses méthodes Update et Draw) directement dans une application Silverlight ! De la même façon, vous pourrez utiliser des contrôles Silverlight au sein d'une scène XNA. Vous pourrez donc poser ces contrôles Silverlight mais surtout, vous pourrez interagir avec eux !

La base de données

La grande majorité des applications développées actuellement ont besoin, à un moment ou à un autre, de stocker des données. Avec la version actuelle du SDK, il y avait une quantité limitée de choix :

- Utiliser un Web Service pour stocker ses données sur un serveur distant (éventuellement dans une base de données)
- Utiliser l'IsolatedStorage pour stocker ses informations.

La première possibilité, bien que faisable, est très peu utilisée car elle repose sur l'état de la connexion Internet qui, comme on le sait tous, est assez compliqué à assurer/maintenir sur un périphérique mobile. L'autre possibilité, l'utilisation de l'IsolatedStorage, est alors celle que l'on trouve le plus souvent. L'IsolatedStorage, c'est un emplacement sur le périphérique dans lequel le développeur peut stocker toutes les données qu'il lui plaît. Le seul problème, c'est que les données sauvegardées le sont sous forme de fichiers texte : il est alors très dur (voir impossible) de les requêter. Certes, il existe des projets qui s'attachent à utiliser l'IsolatedStorage comme fournisseur pour une pseudo base de données (comme par exemple Sterling : http://sterling.codeplex.com), mais on remarque clairement que la notion de SQL, fortement importante, est absente de Windows Phone. C'est cette absence qui sera comblée par la prochaine version du SDK. En effet, une base de données locale, basée sur SQL CE et ADO.NET, sera mise à disposition des développeurs. Cette base de données sera utilisable avec LINQ To SQL, afin de pouvoir écrire des requêtes LINQ qui interagiront avec la base de données [Fig.1]. Au niveau du développement, il sera possible d'utiliser l'approche « Code First » (que l'on retrouve avec Entity Framework par exemple) à savoir l'utilisation d'attributs qui permettront de mapper les objets métier en une base de données (avec, tout de même, l'utilisation de quelques nouvelles classes du SDK).

Les outils de développement

Et oui, il n'y a pas que le système d'exploitation Windows Phone qui a évolué avec cette nouvelle mouture : les outils de développement, qui permettent justement de développer des applications pour Windows Phone, ont eux aussi subi un petit lifting de printemps ! Premier gros changement : la nouvelle version des outils de développement inclut maintenant des outils pour simuler l'accéléromètre et le GPS ! Il s'agit d'une grosse nouveauté car, jusqu'à présent, il était très compliqué de tester ces fonctionnalités sans avoir de périphériques (on ne pouvait pas le faire dans l'émulateur). A présent, les développeurs auront accès à 2 fenêtres supplémentaires :

- Une fenêtre permettant d'afficher le périphérique en 3D et de le manipuler (afin de tester l'accéléromètre)
- Une fenêtre affichant une carte permettant de simuler de nouvelles localisations GPS, des trajets, etc.

[Fig.2 et 3]

Un autre outil fait son apparition dans cette nouvelle version du SDK : il s'agit d'un outil capable d'instrumenter le code des applications Windows Phone pour faire de l'analyse de code et ainsi détecter les problèmes de fuites mémoire, comprendre quelles sont les méthodes qui consomment le plus de temps CPU, etc. Là encore, il s'agit d'un outil qui a longtemps été demandé par les développeurs, de par les problèmes de performances (comme sur le scrolling des ListBox par exemple) que l'on pouvait rencontrer dans la première version du SDK.

Le navigateur Internet

Lui aussi a connu, dans cette nouvelle version du SDK, un petit remaniement ou, devrait-on dire, une montée de version. En effet, le navigateur Internet du téléphone est à présent Internet Explorer 9, le nouveau navigateur Web de Microsoft. La raison de ce changement est simple : Internet Explorer 9 supporte HTML 5 ainsi que l'accélération hardware.

Le contrôle WebBrowser, utilisable en Silverlight pour naviguer vers du contenu HTML, se basera sur le navigateur du téléphone, autrement dit sur IE 9. L'objectif, à terme, est de permettre aux développeurs Web d'écrire leur code une seule fois et de faire en sorte que celui-ci soit multi système d'exploitation/multi périphérique.

Conclusion

La nouvelle version du SDK intègre un très grand nombre de fonctionnalités dont beaucoup ont été très largement demandées par la majorité des développeurs. La première version de Windows Phone souffrait de sa jeunesse mais Windows Phone 7.5 (« Mango ») rattrape une partie du retard pouvant exister par rapport aux autres plateformes. Reste à voir ce que proposera la prochaine version de Windows Phone…

■ Arnaud **Auroux** – *Infinite Square Consultant/Formateur*
aauroux@infinitesquare.com
■ Thomas **Lebrun** – *Infinite Square Consultant/Formateur*
tlebrun@infinitesquare.com
http://blogs.developpeur.org/thadeus/
http://blogs.developpeur.org/tom

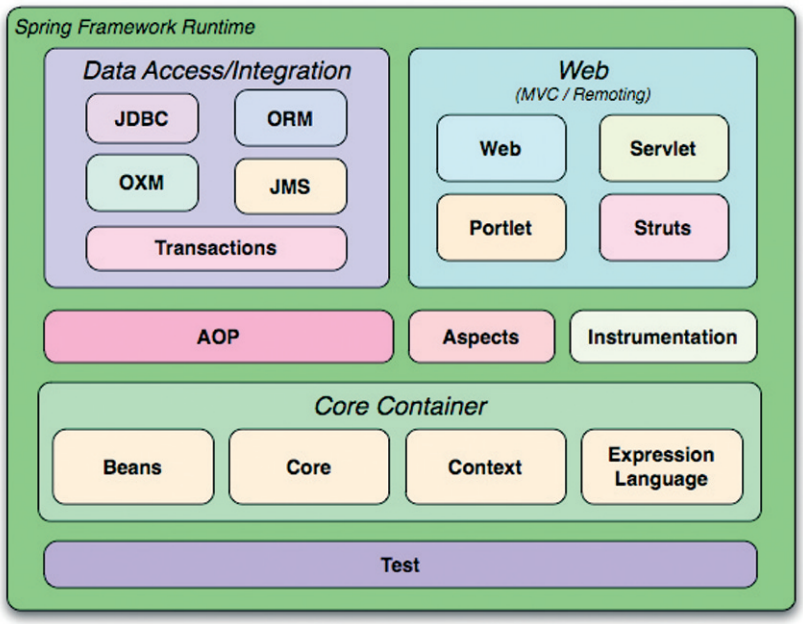
Les Nouveautés de Spring 3

Les facilités de configuration

La première version stable du framework Spring 3 a été la 3.0.0.RELEASE, sortie en décembre 2009. SpringSource, à cette occasion, a légèrement modifié sa politique de maintenance des versions communautaires en incitant les utilisateurs à migrer vers la dernière version. La version majeure précédente ne voyant en effet plus de correctif (il est toujours possible de compiler sa version à partir du SVN).

Nous allons, dans cet article, présenter les nouveautés des releases 3.x de Spring concernant la configuration de vos applications. Spring 3.1 est encore en milestone en avril 2011.

Le mois prochain, nous nous plongerons dans les autres changements apportés par Spring 3, notamment ceux touchant à l'abstraction des caches, ou encore à l'intégration des API de Java EE 6 (Servlet 3.0, JPA 2.0, etc.).



Spring EL (SpEL)

Un puissant « expression langage » compatible avec Unified EL a été proposé dans Spring 3.0. Il est utilisable dans la configuration XML et les annotations comme @Value.

L'intérêt principal est de pouvoir référencer dynamiquement des propriétés dont on ne connaîtra la valeur qu'à l'exécution. L'exemple d'utilisation ci-dessous illustre en outre l'injection d'une propriété de configuration sans passer par un bean en XML.

```
@Value("#{systemProperties.uneProp}")
```

On peut alors utiliser des valeurs de propriétés système ou des propriétés d'autres beans. Le SpEL sera vraisemblablement très utilisé dans les projets du portfolio Spring et a d'ailleurs été conçu en ce sens. Il existe en outre une API SpEL que l'on peut donc embarquer dans une application.

Profiles et API Environment

N'avez-vous jamais eu besoin de disposer dans le conteneur Spring d'un mécanisme permettant de variabiliser l'enregistrement des beans en fonction d'un environnement ?

La typologie cible de l'environnement d'exécution étant soit un environnement de recette, un environnement de pré-production, ou encore un ou plusieurs environnements de production, chacun dédié à des clients différents.

Avant les releases 3.x de Spring, des solutions satisfaisantes existaient déjà. Elles consistaient, par exemple, à lire des propriétés dans un fichier et à les utiliser pour substituer les valeurs de certaines propriétés de beans. Un exemple typique est celui de la déclaration d'une datasource, ciblant une source de données différente en fonction de l'environnement (recette ou production). Il suffisait alors soit de changer de fichier de propriétés ou de jouer avec la composition de contexte Spring via les imports pour valoriser différemment vos propriétés URL, driver ou username JDBC de votre bean datasource en fonction de l'environnement.

Mais comment changer facilement la définition même de votre bean en fonction de l'environnement d'exécution ? Spring 3.1 introduit la notion de *profiles* applicables à un groupe de définitions de beans. Reprenons l'exemple de la déclaration de datasource et introduisons au passage une autre facilité de configuration apportée cette fois par Spring 3.0, celle des bases de données embarquées. Pour exécuter vos tests unitaires, il peut s'avérer en effet fort utile de disposer d'une telle base. Pour ce faire, un namespace jdbc permet de définir votre datasource, correspondant à votre base de données embarquée préférée (HSQLDB, H2 ou Derby).

Avec la balise *script*, il est possible d'enchaîner des exécutions de scripts dès l'initialisation de la DataSource pour créer vos tables ou encore insérer un jeu de données. Voici comme vous définissez alors très simplement votre bean dataSource de type *javax.sql.DataSource*. Notez le fait qu'un « profile » est ajouté à la balise bean, ce qui permettra de n'enregistrer ce bean que dans le cas où le profile *dev* est activé à l'exécution.

```
<beans profile="dev">
  <jdbc:embedded-database id="dataSource">
    <jdbc:script location="classpath:conf/sql/schema.sql"/>
```

```
    <jdbc:script location="classpath:conf/sql/test-data.sql"/>
  </jdbc:embedded-database>
</beans>
```

Pour définir votre bean dataSource dédié à l'environnement de production :

```
<beans profile="production">
  <jee:jndi-lookup id="dataSource" jndiname="java:comp/env/jdbc/
datasource"/>
</beans>
```

Il est également possible d'annoter vos beans grâce à l'annotation @Profile.
Lors du chargement du contexte Spring, les déclarations de beans « profilés » ne seront pas parsées et donc les beans ne seront pas enregistrés si le profile n'est pas actif. Tous les beans non « profilés » seront chargés par défaut. Spring 3.1 apporte une nouvelle notion à travers l'API Environment qui contient, entre autres, des informations sur les profils actifs au moment de l'exécution. Il existe ainsi plusieurs moyens d'activer un *profile* :

- soit déclarativement, en valorisant la propriété *spring.profiles.active* classiquement à travers une propriété système JVM, une variable d'environnement, ou pour des applications web, un paramètre de contexte de servlet dans le fichier web.xml.
- soit programmatiquement avec les API ApplicationContext et Environment :

```
GenericXmlApplicationContext ctx = new GenericXmlApplication
Context();
ctx.getEnvironment().setActiveProfiles("dev");
ctx.load("classpath:/com/programmez/samples/config/xml/ticke
torder-service-config");
ctx.refresh();
```

L'API Environment permet également de personnaliser la résolution des placeholders. Dans les versions antérieures à la release 3.1 de Spring, il existait plusieurs mécanismes, dont les *PropertyPlaceholderConfigurer* pour résoudre ces placeholders dans la définition des beans de l'application *context*. On supposait qu'une propriété était définie soit dans un fichier properties, soit comme propriété système, soit comme variable d'environnement. Avec l'API Environment, on peut personnaliser cette résolution. Tout d'abord sur l'ordre de prédominance de la recherche des propriétés, la recherche effectuée par défaut étant hiérarchique (les propriétés système prédominent sur les variables d'environnement).
Mais également sur la manière même de les résoudre en définissant votre propre stratégie (appelée *PropertySource*), en la rendant prédominante sur les autres, ou encore en supprimant l'une des stratégies existantes. Imaginez alors les potentialités qui s'offrent à vous en définissant non seulement votre propre stratégie de résolution mais en la configurant également par Profile! Pour un environnement donné, vous pourriez avoir besoin de ne pas tenir compte des propriétés système :

```
ConfigurableApplicationContext ctx = new GenericApplication
Context();
MutablePropertySources propertySources = ctx.getEnvironment().
getPropertySources();
ropertySources.remove(DefaultEnvironment.SYSTEM_PROPERTIES_PRO
PERTY_SOURCE_NAME);
```

Spring Java Configuration

Les fonctionnalités principales du projet JavaConfig sont désormais intégrées dans Spring Core étendant ainsi le support de la configuration basée sur du code Java. Spring semble s'être inspiré de Google Guice qui, nativement, ne propose que le mode de configuration Java (un module annexe permet d'importer des fichiers XML).
Il est en outre possible d'injecter des valeurs issues de fichiers de configuration via @Value directement sur des membres d'une classe. La configuration Java (*Plain Old Configuration*, terme inventé par Erich Eichinger) représente une importante contribution de code. Elle n'était pas directement utilisable dans la version 3.0 (pas de component-scan, pas de feature specification, etc ...). La version 3.1 vient combler certains manques.

Exemple simple

Voyons comment mettre en place un exemple simple en se penchant d'abord sur la version XML de la déclaration d'un service singleton :

```
<bean  class="com.programmez.samples.gigreservation.service.
internal.DefaultTicketOrderService" />
```

L'équivalent en configuration Java donne :

```
@Configuration
public class OrderServiceConfig {

  @Bean
  public TicketOrderService orderService() {
    return new DefaultTicketOrderService();
  }
}
```

Il suffit alors d'instancier un *AnnotationConfigApplicationContext* en précisant la classe de configuration.

```
ApplicationContext ctx = new AnnotationConfigApplicationContext
(OrderServiceConfig.class);
TicketOrderService service = ctx.getBean(TicketOrderService.
class);
```

Au chargement, Spring étend, via CGLib, les classes annotées *@Configuration* pour leur ajouter des comportements. Ainsi, via l'annotation *@Bean*, on peut s'assurer que les singletons ne soient instanciés qu'une seule fois. Une demande de la communauté JIRA *SPR-7484* existe pour permettre d'utiliser *@Configuration* avec AspectJ plutôt que CGLib.

Component scan

Le component scanning est bien sûr possible à l'aide de l'annotation *@ComponentScan*.

```
@Configuration
@ComponentScan("com.programmez.samples.gigreservation")
public class OrderServiceConfig {
```

```
  // Classe de configuration vide
}
```

Cette option est identique au context:component-scan XML et à l'utilisation de *@Autowired* dans les beans.
Toutefois, se limiter à utiliser le *@ComponentScan* reste d'un intérêt très relatif si une partie du reste de la configuration n'est pas transposée en Java.

Configuration Java et XML

A première vue, ce mode de configuration impose de tout déclarer en Java. Comment traiter le cas des beans définis dans des fichiers XML ? Deux options possibles :

- Java centric : on inclut le ou les fichiers XML dans les classes de configuration Java.

```
@ImportResource("classpath:/ticketorder-service-config.xml")
```

- XML centric : On inclut la déclaration des beans de configuration Java dans le fichier XML en déclarant un *context:annotation-config* et en utilisant un "classique" ClassPathXmlApplicationContext pour le bootstrap.

```
<context:annotation-config/>
<bean class="com.programmez.samples.gigreservation.config.code.
OrderServiceConfig"/>
```

En dehors de toute considération d'architecture, on peut donc mélanger les deux approches de configuration en Java et en XML. En cas de doublon de définition sur un bean, Spring indique clairement la surcharge effectuée à partir de la configuration XML dans ses logs.

Feature Specification

La configuration Java est parfaitement valable mais reste très limitée du fait de l'absence d'équivalent aux namespaces XML tel que *tx:annotation-driven*.
Pour rappel, les namespaces XML sont un moyen de rendre sa configuration plus concise et lisible.
Voyons maintenant comment utiliser les *FeatureSpecifications* introduites dans la version 3.1 et ainsi bénéficier de la même facilité et concision de configuration en s'appuyant sur des appels de méthodes chaînées et de la complétion.
Le support de la configuration des transactions par annotations en XML peut être décrit par :

```
<tx:annotation-driven transaction-manager="txManager"/>
```

Voici maintenant l'équivalent en configuration Java :

```
@FeatureConfiguration
class TxFeature {
 @Feature
 public TxAnnotationDriven txAnnotationDriven(PlatformTransac
tionManager txManager) {
   return new TxAnnotationDriven(txManager);
 }
}
```

Beaucoup de *FeatureSpecifications* sont déjà disponibles et la version 3.1.0.M2 verra l'ajout du support de namespaces XML additionnels tels que *<context:mbean-export/>, <context:property-placeholder/>, <aop:aspectj-autoproxy/>, <jee:jndi-lookup /> ou <task:annotation-driven/>* via des feature specifications.
Spring Source explicite très bien le fait que tous les namespaces XML ne vont pas être transcrits en *FeatureSpecification* mais seuls ceux qui font sens dans une configuration Java.

Conclusion

Les principaux atouts de la configuration Java sont les suivants :

- typage fort et compilation (y compris lors d'un changement de configuration) ;
- refactoring facilité ;
- définition de bean complexe dynamiquement (à noter que les versions précédentes de Spring permettaient déjà de configurer un bean en Java en implémentant FactoryBean) ;
- détection de certaines erreurs de définition du contexte avant l'exécution ;
- puissance de la programmation objet (héritage et surcharge de classe de configuration possible).

Même si les dernières versions de Spring améliorent grandement la configuration à base de code, les Tech Leads de Spring mentionnent clairement le fait que la configuration Java n'a pas pour but de remplacer complètement la configuration XML, mais plutôt de venir en complément.
Pour l'instant la base de code supportant la configuration Java reste dans tous les cas moins éprouvée que l'approche classique en full XML ou en XML complétée d'annotations.
La configuration XML reste toujours populaire et peut être facilitée par des outils tels que SpringSource Tool Suite, elle est donc loin d'être en phase d'abandon.
Nous voyons plutôt dans la configuration Java une troisième voie intéressante pour la configuration Spring, qui cumule les bénéfices des deux premières, à savoir XML et les annotations, tout en évitant leurs problématiques inhérentes. Puisque la configuration se fait programmatiquement, elle reste type safe, tout comme avec les annotations et bénéficie de tous les avantages de Java comme nous venons de le décrire ci-dessus.
Etant donné que l'approche Java regroupe les éléments de configuration dans des classes dédiées (annotées par @Configuration, @FeatureConfiguration, etc.), elle garantit la séparation des préoccupations (*Separation of concerns*), ce qui était un des principaux arguments des défenseurs de l'approche XML et détracteurs des annotations. L'autre perspective intéressante que nous voyons dans la configuration Java est son utilisation conjointe aux Profiles, et la possibilité de variabiliser et de flexibiliser la configuration de vos applications en fonction d'un environnement ou d'un besoin métier particulier.

■ Agnès **Crepet**
Architecte JEE, JDuchess & Lyon JUG Leader
Twitter : http://twitter.com/agnes_crepet

■ Vladislav **Pernin**
Architecte JEE chez Zenika
Twitter : http://twitter.com/vladislavpernin

Construire et faire vivre une communauté

Une méthode appliquée aux projets de logiciel libre

Dans le domaine de la R&D, les technologies telles que le Web 2.0 ont favorisé le développement de nouvelles manières de travailler, souvent basées sur des relations entre pairs et évoluant vers la création d'une communauté organisée. Ces communautés ont montré leur efficacité et leur dynamisme dans de multiples domaines, c'est notamment le cas pour le développement de logiciels libres. Par ailleurs l'accroissement des connexions au niveau international appelle à une collaboration à distance entre des réseaux d'experts à travers le monde.

Développé à l'INRIA sur la base de l'expérience du chapitre local Européen de OW2, et sur la création de la communauté AspireRFID, cet article propose une méthodologie pour créer et faire vivre une communauté de pratique.

Qu'est-ce qu'une communauté ?

Une communauté peut être considérée comme un groupe de personnes qui partagent les mêmes intérêts, les mêmes préoccupations ou la même passion. Une communauté peut également être fondée sur des rôles ou sur des spécialités (administrateur réseau, utilisateurs Linux, utilisateurs de synthétiseurs Modulaires, etc.). Ces personnes approfondissent leur compréhension en partageant leurs connaissances, en aidant les autres a résoudre des problèmes, en interagissant régulièrement avec les autres membres de la communauté, en posant et en répondant a des questions, en réutilisant les bonnes idées. Lorsque la participation est active au sein du groupe une nouvelle identité virtuelle se crée sur la base de liens sociaux forts. Émerge alors un phénomène de production collective.

Une typologie des communautés

On distingue quatre grands types de communautés : les communautés d'apprentissage (ex. le projet Plume), les communautés d'intérêt (ex. Audiofanzine), les communautés de passion (ex. les administrateurs système participant aux conférences JRES), les communautés de pratique (ex. les groupes d'utilisateurs de Linux). Une communauté d'apprentissage est un groupe de personnes qui partagent des valeurs communes et qui sont activement engagées dans un processus d'apprentissage mutuel. De telles communautés sont devenues un modèle initial pour les autres types de communautés. Les membres d'une telle structure ont un sens fort de la loyauté envers le groupe qui alimente leur désir de continuer à travailler et à aider les autres, de produire un effet sur la communauté en étant non seulement réactifs mais actifs. Une communauté d'apprentissage laisse une liberté suffisante pour que les membres puissent exprimer des opinions personnelles, demander de l'aide ou des informations spécifiques et commenter des événements. Une communauté d'intérêt peut être vue comme un groupe de personnes partageant des thèmes qui ne nécessitent pas de structure formelle mais qui mènent des discussions tournées vers la collaboration et le partage des connaissances. Elles peuvent consister en des groupes de personnes faiblement connectées sans grande implication en termes de production commune. Elles restent cependant bien au fait de leurs thèmes et posent des questions.

Une communauté de passion est basée sur un groupe de personnes dont les activités, la gouvernance et la structure sont les plus riches et les plus formelles. Les membres y jouent un rôle particulier (ex. conseil en sécurité de réseau), ils aident activement les autres membres à coller à leur rôle et visent à maîtriser la discipline. La structure d'une communauté de pratique est moins formelle et basée sur des spécialités communes. Les membres ont un rôle ou une spécialité spécifique (ex. la sécurité, le développement JEE) et se focalisent sur le développement de l'expertise et des compétences qu'ils ont dans ce rôle ou cette spécialité. Un facteur de motivation essentiel est l'apprentissage de la spécialité et la résolution de problèmes. La méthodologie présentée dans cet article a été ébauchée à partir de, et appliquée à ce type de communauté.

Méthode

La méthode proposée est composée de 5 grandes étapes : **Analyse, se construire, promotion, entretenir, évaluation**.

Analyse

L'objectif de la phase d'analyse est d'initier le processus et d'identifier toutes les informations pertinentes pour comprendre ce que nous voulons faire, où nous voulons aller, pourquoi et comment. Comme nous l'avons déjà mentionné, une communauté peut émerger spontanément, à travers la seule passion partagée par un petit groupe de personnes enthousiastes, mais augmenter la probabilité de cette émergence suppose un peu d'organisation et de méthode.

Dans cette phase, il faut commencer par rêver : comment est-ce que, idéalement, la communauté devrait fonctionner ? Devrait-


© istockphoto/geopaul

elle être organisée ? Qui la guidera et fera se produire des choses ? Il s'agit de laisser libre cours à son imagination. Identifier les thèmes et les sous-thèmes permettra de clarifier les choses et facilitera la réflexion avec les collègues.

C'est à ce moment qu'il est important d'identifier l'équipe qui constituera le noyau dur de la communauté et de savoir qui est motivé et souhaite contribuer activement à l'aventure.

Une fois l'équipe définie, il s'agit de spécifier les bases opératoires, en fait de déterminer comment travailler ensemble. Cela suppose encore un peu d'imagination et peut-être quelques séances de brainstorming et des échanges d'information. Ces bases opératoires comprennent la gouvernance, la communication, le cycle de vie du développement, l'environnement collaboratif, la dissémination et la promotion.

A ce moment précis, il va falloir identifier et utiliser des outils/logiciels communautaires pour organiser la collaboration, les contenus structurés et pour faciliter les activités de dissémination (promotion et sensibilisation). Ces outils, organisés autour d'une forge logicielle permettront d'assurer les opérations quotidiennes et la communication en constituant le référentiel central de la communauté et du projet.

Ce processus doit permettre d'avoir une meilleure idée de ce que l'on veut construire. A ce stade il faut s'arrêter et voir s'il existe déjà des communautés ou des projets partageant les mêmes préoccupations et les mêmes thèmes. Le fait de rejoindre ces initiatives ou de décider de démarrer à partir de rien est une décision nécessaire, qui doit être prise en toute connaissance de cause, et qui implique deux démarches différentes.

A la fin de la phase d'analyse, la décision de rejoindre une communauté existante ou de créer de toute pièce notre structure doit être prise sur la base de l'ensemble des informations collectées et de la démarche retenue. La décision de rejoindre une communauté existante devra être soigneusement justifiée afin d'identifier les principales entraves en fonction du contexte et pour éviter de se reposer les mêmes questions plus tard, lorsque l'idée originelle sera réexaminée. Une décision de créer de toute pièce notre structure est encore plus difficile à prendre car le temps et les efforts passés jusque-là sont bien moindres que ce qui sera nécessaire pour lancer le projet et construire la communauté. Un engagement fort et une compréhension très claire du chemin à parcourir et des efforts à consentir sont indispensables et doivent être partagés par l'ensemble des membres du noyau dur. Dans la mesure où le choix s'est porté sur une construction à partir de rien, il est temps d'aborder la deuxième phase : la construction, ou «se construire».

Se construire

La construction est le moment où l'on va joindre les briques avec le ciment. On commence à développer, à créer la documentation (ex. guide d'utilisation, documentation d'API, cas d'utilisations, etc.), à mettre en place le portail (mission, licence, consignes d'installation du logiciel, page de téléchargement, état du développement, copies d'écrans, comment participer, etc.), à diffuser les paquets, publier les tâches et la feuille de route du projet, gérer les volontaires et les contributions extérieures. Les activités quotidiennes seront assujetties à la gouvernance de la communauté, la politique de propriété intellectuelle, le cycle de vie du projet et les règles de communication (simples et flexibles). Une fois ces tâches réalisées, il est temps de faire connaître votre existence à travers la promotion.

Promotion

Lorsque l'équipe est en place, qu'une base de code existe, qu'une documentation est disponible et qu'un portail permet à chacun de trouver toute l'information nécessaire soit pour contribuer soit pour utiliser le code, alors la communauté existe et il faut la promouvoir. A travers des articles dans les « newsletters » qui visent l'audience concernée, par une diffusion de messages via des listes de diffusion, en utilisant les réseaux des membres et en leur demandant de passer le relais. La promotion de la communauté peut aussi s'appuyer sur des revues et journaux nationaux et internationaux, sur des webzines et sur les réseaux sociaux. Une fois que la communauté a été créée, qu'elle fonctionne et qu'elle a atteint un certain niveau de reconnaissance, c'est le moment d'aborder la troisième phase: entretenir.

Entretenir

Une fois la communauté lancée et reconnue il faut maintenir la dynamique, pour cela son ou ses leaders devront consacrer du temps pour l'animer et la stimuler. Nous recommandons d'organiser périodiquement des événements (journées, assemblée générale), de tenir régulièrement des conférences téléphoniques, de publier les comptes-rendus, d'être actif sur les forums, et à l'occasion, d'alimenter les news et les forums de la communauté avec des informations provenant de l'extérieur. La transparence pour toutes les décisions et l'écoute des personnes, en laissant suffisamment de liberté pour traiter de tous les sujets et impliquer vos membres aux processus de décision, sont des points importants.

Il est également important d'atteindre une masse critique de membres qui vont permettre de répartir les opérations quotidiennes (administration système, « bug trackers », « bug fixers », webmaster, chef de projet, traducteurs, etc.) Il faudra néanmoins toujours essayer d'accroître la taille de la communauté, et ce,en proposant des initiatives (concours de programmation, « code camp », « project summit », club des utilisateurs, etc.), en utilisant les réseaux sociaux, en affichant les valeurs de la communauté sur le portail, en expliquant comment participer et devenir un membre, en incitant à rejoindre la communauté. Il ne faut pas hésiter à franchir les barrières et à aller chercher les bonnes idées dans les autres structures (par exemple les communautés de jeux vidéo).

Deux autres facteurs clés permettent d'inciter les utilisateurs externes à contribuer activement.

D'abord leur accorder de la reconnaissance, et partager la connaissance. Pour cela, il est possible d'afficher sur la page d'accueil les noms des membres récemment inscrits ou les meilleurs correcteurs de bugs. Ensuite de consacrer des ressources à l'organisation de la connaissance dans la communauté en proposant des outils de

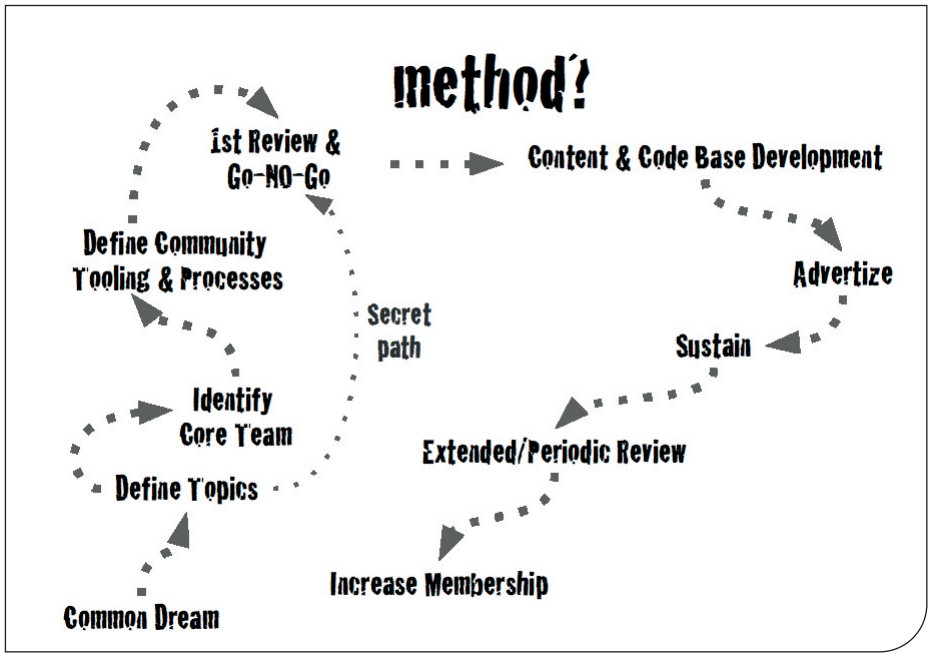
partage comme les webinars, des ateliers, des cas d'utilisation, etc.
Tout comme dans la phase d'analyse, il est également possible de voir si d'autres communautés ou projets existants pourraient être intéressés par l'utilisation de la base de code, ou si une coopération pourrait être intéressante. Dans le cas où de telles communautés seraient identifiées, contacter leurs leaders et leur proposer une collaboration peut être une excellente idée d'autant plus si votre budget est restreint.
La réalisation de cette phase nous amène à la question du contrôle ou disons de l'évaluation.

Evaluer

Pour pouvoir évaluer les progrès et prendre acte de l'effet des actions menées, il est souhaitable de définir son propre processus de contrôle plutôt que d'utiliser des modèles prédéfinis. Il faut pour cela identifier l'objectif de ma mesure (que mesurer et pourquoi), définir les métriques utilisées, collecter les données (analyse), identifier les problèmes et les tendances, définir les actions à engager pour les résoudre et les corriger, et réévaluer la santé du projet.
Il est également souhaitable de contrôler la croissance de la communauté et la santé du projet à travers des ordres de grandeur comme le nombre de bugs suivis, le nombre de téléchargements, le nombre de visites, le nombre de messages postés dans les forums et de mails échangés, le nombre d'événements organisés, etc. Enfin il faut aussi penser à surveiller l'évolution générale de la communauté : est-elle en phase de démarrage ? En phase de croissance ? Est-elle mature ? Ou bien commence-t-elle à décliner, ou à se relancer ? Il est important de savoir se situer dans cette grille car vous ne lancerez pas les mêmes actions si vous avez une communauté mature ou si vous êtes en cours de démarrage.

La communauté du projet AspireRFID

Le FP7 du projet ASPIRE, partiellement financé par EC, vise à développer et promouvoir un standard basé sur un middleware open source léger, conforme aux standards existants, évolutif en termes d'échelle, respectueux de la vie privée, et intégré pour faciliter le développement, le déploiement et le management d'applications basées sur des RFID, des senseurs , des capteurs. Le consortium ASPIRE s'est employé à créer une communauté dynamique autour des technologies RFID en veillant à amorcer une base de code capable de continuer à évoluer après la fin du financement du projet européen. C'est de là qu'a émergé l'idée de créer une communauté de pratique (et de passion) autour de cette base de code.



Méthode pour créer et faire vivre une communauté.

Nous avons saisi l'occasion de ce projet pour tester notre méthode, à la fois à travers des audits et des recommandations. Cela fut pour nous l'occasion d'évaluer la méthode et de faire des ajustements. Notre implication dans ASPIRE et dans la communauté OW2 nous a permis de constater une adéquation positive dans le domaine du RFID : d'un côté le projet ASPIRE visait à développer une plateforme logicielle RFID open source, et de l'autre une nouvelle initiative autour du développement logiciel autour des capteurs était soutenue par le consortium OW2. Le premier décollait alors que le second cherchait de nouvelles contributions. Il s'agissait d'un accord parfait qui bénéficiait mutuellement aux deux composantes. Enfin, les outils de collaboration (en deux mots, l'architecture de participation) convenait parfaitement au projet ASPIRE (wiki, webinar, mailing list, forge, bug tracker, serveur de téléchargement, participation et organisation d'événements autour du « middleware », etc.).
Nous allons maintenant décrire les trois principaux résultats qui ont été atteints :
Accélération de la phase de démarrage : la mise en œuvre de cette méthode nous a permis de découvrir que le consortium OW2 pouvait non seulement nous aider à créer la communauté AspireRFID, mais qu'il disposait de nombreux outils parfaitement adaptés aux besoins de la communauté AspireRFID. OW2 propose notamment une forge permettant aux développeurs de commencer à coder très rapidement. OW2 est également ouvert sur la sensibilisation et la promotion des outils.
Améliorer la dissémination et la promotion : la méthode nous a permis d'obtenir des résultats très rapidement grâce aux actions de promotions spécifiques comme des articles, des « code camps », concours de programmation, assemblées générales, club utilisateurs, etc. Nous avons découvert que OW2 était un excellent canal de valorisation et de promotion, notamment à travers sa liste de diffusion.
Enfin, le succès de la dissémination et de la promotion **renforça les produits de l'exploitation :** un benchmark RFID réalisé par le consortium industriel ICOM, la plateforme logicielle ASPIRE furent comparés à des produits commerciaux. En dépit du fait que le projet ne soit pas encore achevé et que les outils logiciels soient encore en cours d'élaboration, ASPIRE fut mieux classé que tous se compétiteurs. Cela conforta l'idée selon laquelle notre méthode avait permis de renforcer l'évaluation du produit, de diffuser sa feuille de route, d'accroître sa réputation et d'assurer la pérennité du projet.

Conclusion

Comme toujours avec une nouvelle méthode, des ajustements et des améliorations sont nécessaires. Nous avons travaillé à l'intégration des nouveautés qui nous sont apparues dans une nouvelle révision de version, cependant les principaux enseignements peuvent être résumés ainsi :

- Pour en obtenir les plus grands bénéfices, la méthode doit être appliquée dès l'origine du projet ; dans ce cas le projet avait démarré quelques mois avant que nous ne mettions en place cette méthode.
- L'implication et la motivation des personnes en sont la clé. Une communauté est fondée sur des personnes, et aucune méthode ou aucun outil ne peut remplacer la volonté de contribuer.
- Consacrer du temps au partage de connaissance est important, de même que trouver des personnes disposant d'un certain niveau de créativité et d'aptitude à exercer le leadership.
- Une vision commune doit être partagée entre les membres et elle doit être clairement affichée.
- Les différentes étapes décrites dans cette méthodologie ne sont pas nécessairement séquentielles : leur recouvrement partiel permet de sur-vitaminer le planning.
- AspireRFID est un projet FP7, il doit rendre des livrables basés sur une feuille de route définie qui peut parfois s'écarter des souhaits des membres de la communauté. Les conflits d'intérêt doivent être identifiés et traités au plus tôt.

Bien sûr, la version présentée ici peut être améliorée, à ce titre deux nouveaux articles ont déjà été publiés [6, 22]. Lancer une communauté et la développer avec succès n'est pas une action spontanée due à la chance ou au hasard mais s'appuie sur une approche structurée telle que présentée dans cet article. En présentant une telle méthode nous espérons inciter ceux qui souhaiteraient créer une communauté à se lancer dans cette aventure d'une manière aussi efficace que possible.

Références

[6] How to kill a community by Stéphane Ribas and Michel Cezon, OSW2009 – IEEE – 14-15 Oct 2009 Sweden, Skovde, ISBN 978-91-978513-3-6,

[22] Build a FLOSS community from scratch? Ways of building open source projects and pitfalls to avoid by Stéphane Ribas and Michel Cezon, TERENA 2010.

■ Stéphane Ribas

stephane.ribas@inria.fr

Equipe D2T *Direction du Développement Technologi*que, INRIA Rhone-Alpes

M.Sc, University of Surrey 1996, spécialiste en logiciel libre. Il est l'organisateur des conférences internationales fOSSa (free open source software for academia). Il enseigne à l'IAE de Grenoble. Il a rejoint l'INRIA en 2008 pour animer plusieurs projets Européen (QualiPSo, AspireRFID, etc.) et ObjectWeb2.

■ Patrick Guillaud

patrick.guillaud@inria.fr

Equipe D2T *Direction du Développement Technologique*, INRIA Rhone-Alpes

Responsable des relations entre STIC et société à la Direction du développement technologique de l'INRIA depuis 2010. Son thème de recherche est la conception de produits complexes par des équipes hétérogènes. Il enseigne à l'INSA de Lyon.

Tests votre anti-bugs

Ah, les tests ! Voilà bien un sujet dont de nombreux développeurs ne veulent pas entendre parler, ou tout du moins, pas trop. Même dans les entreprises, il existe toujours ce vilain réflexe de couper les tests pour aller plus vite, respecter (un peu) les délais ou tout simplement alléger le budget.

Or, sans test, impossible de trouver et corriger les bugs, impossible de qualifier son code et donc son application. Aujourd'hui, il existe pléthore de logiciels de tests, de méthodes. A propos de méthode, l'agilité impose la qualité et donc in fine, les tests. Et les principaux éditeurs de ce marché font des efforts considérables pour intégrer finement le développement et les tests, sans oublier le cycle de vie. Malgré tout, le test demande toujours un effort dans la mise en place, l'exécution et sa maintenance. Car il évolue selon les versions du code, les projets.

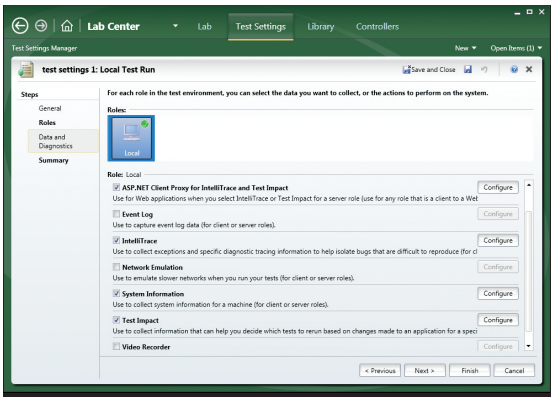
Dans ce dossier, nous allons aborder plusieurs thématiques, parfois peu connues, des tests, comme le Serendipity testing. Nous nous demanderons aussi pourquoi utiliser l'approche SQALE et quelles sont les nouvelles tendances du test…

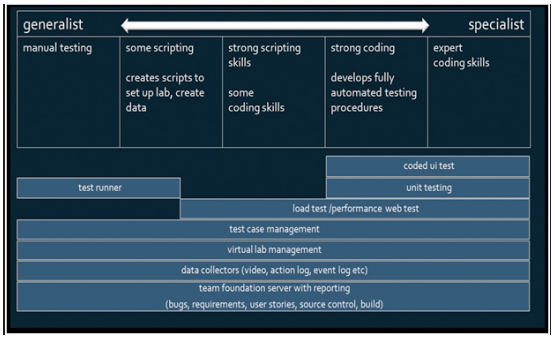
Nous dédions ce dossier à Adam Kolawa, expert reconnu des tests et des méthodes de développement. Co-fondateur de Parasoft, il nous a quitté brutalement le 26 avril dernier. (voir page 57)

■ François Tonic

Les tests, c'est moche mais ils peuvent sauver votre code !

Quel test choisir et pourquoi ? C'est une question qui revient souvent et que nous nous poserons dans les pages qui vont suivre. Mais d'abord, plantons le décor : les différents types de tests et quelques statistiques détonantes.





Une récente étude *« Software Integrity Risk Report »*, réalisée par Forrester et sponsorisée par Coverity, éditeur spécialisé dans les outils d'intégrité du code et des logiciels, dévoile des statistiques aussi étonnantes qu'inquiétantes :

- Plus de 90% des responsables interrogés confirment avoir recours à des codes tiers fournis par des éditeurs, des équipes sous-traitantes ou des fournisseurs d'open source
- Plus de 40% des responsables interrogés ont souligné le fait que les défauts des codes logiciels tiers provoquent des retards de mise sur le marché, des rappels de produits, des failles de sécurité, des allongements dans les délais de développement et des baisses de revenus. Autant de problèmes les incitant à vouloir obtenir une meilleure visibilité sur l'intégrité des codes.
- Environ 65% des entreprises déclarent que la satisfaction client est impactée par les défauts logiciels, et 47% pensent que le délai de mise sur le marché est également impacté par ces mêmes défauts logiciels
- Seules 44% des entreprises interrogées procèdent, pendant les phases de développement, à des tests de code automatisés sur les codes fournis par des partenaires. Tandis que 69% de ces mêmes entreprises procèdent à des tests automatisés pour les codes développés en interne
- Seules 35% des entreprises interrogées effectuent des évaluations des risques et des aspects de sécurité et de vulnérabilité pour les codes logiciels tiers qu'elles utilisent dans leurs développements. Tandis que 70% d'entre-elles appliquent ce type de méthodes aux logiciels développés en interne
- Seules 35% des entreprises interrogées procèdent à un contrôle de code manuel sur les logiciels fournis par leurs partenaires. Tandis que 68% le font pour les codes développés en interne
- Les écarts d'assurance de qualité sont également mis en exergue, avec 51% des entreprises interrogées déclarant effectuer des tests fonctionnels, de charge et d'unité automatisés pour les logiciels fournis par des partenaires tiers. Tandis que 75% appliquent ces mêmes méthodes d'assurance qualité aux logiciels développés en interne.

Quelques types de tests

- Test unitaire: le plus classique, qui permet de tester du code, avec un squelette généré.
- Test unitaire de base de données : équivalent du test unitaire mais pour les SGBD.
- Test ordonné : permet d'exécuter un ensemble de tests dans un ordre séquentiel, le test réussit si l'ensemble des sous-tests réussissent.
- Test Web : capture et rejoue un scénario de navigation Web.
- Test de charge : permet de rejouer en boucle un ou plusieurs tests en simulant de la charge pour vérifier les performances de votre application sous différents niveaux de stress.
- Test générique : test pouvant exécuter n'importe quelle application externe.
- Test d'interface utilisateur : tester la charge, la cohérence et la sécurité d'une interface
- Test personnalisé : il est possible de créer vos propres types de tests pour simplifier certaines tâches répétitives (par exemple, la génération d'un proxy lorsque l'on teste des services WCF).

A cela se rajoutent les tests en boîte noire et boîte blanche. Vous pouvez utiliser la couverture de code pour vérifier la présence de code mort, de codes dupliqués, etc. Ces tests sont importants pour structurer et épurer le code source. Les tests d'impact ou tests impactés permettent de mesurer l'impact d'une modification de code (le test connaît l'état d'origine du code pour mesurer l'impact). Très utile dans des projets complexes.

(En partie extrait de Microsoft, *« Les Tests logiciels, étude de cas avec Visual Studio 2010 »*)

Quel rapport avec les tests ? Une partie des problèmes cités plus haut peut être réduite par les tests, le plus en amont possible, dès les premières lignes de code. Mais il est vrai que le test représente entre 20 et 30 % du coût d'un projet informatique mais aujourd'hui, il est possible d'optimiser les tests, de réduire le coût par l'automatisation de ceux-ci.

■ François **Tonic**

Avis d'expert
Du test pour les développeurs ?

Les tests de logiciels prennent de plus en plus d'importance en France et dans le monde, pour preuve le succès de la certification CFTL-ISTQB et des Journées Françaises des Tests Logiciels organisées par le CFTL, avec plus de 400 personnes et près de 200 sociétés représentées lors de la dernière journée. Pour les testeurs c'est un succès, pour les développeurs, serait-ce un aveu d'échec ?

Dans les développements logiciels, la part des tests prend de plus en plus d'ampleur, le métier de testeur – fonctionnel ou technique – se développe, et pourtant la qualité des logiciels n'atteint toujours pas le niveau que les utilisateurs sont en droit d'attendre. Cependant, testeurs et développeurs font du mieux qu'ils peuvent pour livrer des logiciels avec un bon niveau de qualité.

Pourquoi cela ne fonctionne-t-il pas ?

- La complexité croissante des logiciels rend les tests de composants de plus en plus complexes. Ces tests unitaires devraient être effectués – en tout ou en partie – par les développeurs, mais ceux-ci ne maîtrisent pas les techniques à appliquer pour identifier rapidement les défauts. Ce manque de connaissances des développeurs induit un manque d'efficacité des tests unitaires et un accroissement des défauts livrés aux phases de tests subséquentes (tests d'intégration des composants, tests système et tests d'acceptation). L'accroissement des défauts dans le logiciel implique un accroissement des coûts et délais de réalisation dus à l'obligation d'identifier les défauts, de les corriger, de s'assurer ensuite de leur correction (retests) et de la non-introduction d'anomalies ailleurs dans le logiciel (tests de non-régression), tout cela avant la mise en production du logiciel.
- Les interactions des composants logiciels entre eux et l'explosion combinatoire des différentes configurations matérielles et logicielles rendent les tests d'intégration de composants plus longs et plus complexes. Lors de cette phase, la découverte de défauts qui auraient pu être décelés antérieurement ralentit le travail et nuit à sa rentabilité. Les développeurs peuvent, par leurs connaissances techniques et leurs compétences, contribuer à la réduction des anomalies découvertes dans les phases d'intégration.
- Si les tests de composants et les tests d'intégration sont effectués correctement, les défauts décelés lors des tests système et des tests d'acceptation proviennent d'une mauvaise compréhension des exigences et des spécifications par les équipes d'analyse et de réalisation. L'identification dans cette phase de spécifications ou d'exigences ambigües qui auraient pu (dû ?) être identifiées au cours de revues, démontre le besoin de ces activités pendant les phases de conception du logiciel. Une focalisation sur ces phases amont du développement démontre rentabilité et efficacité.

Ces constats ne sont pas une fatalité et peuvent être combattus activement. Développeurs et testeurs ont les mêmes objectifs : des logiciels sans défauts livrés aux utilisateurs dans le respect des délais et des budgets.

Comment atteindre ces objectifs ? En formant les développeurs aux techniques de test

Une solution, serait de former les développeurs aux techniques de base du test, afin qu'ils les utilisent dès la conception du logiciel. Ceci implique une refonte des cursus de formations de développeurs, pour y enseigner des techniques simples comme les Partitions d'Equivalences, l'Analyse des Valeurs Limites, les Tables des Décisions, et les Transitions d'Etats. Cette solution ne portera ses fruits qu'à moyen terme – après la fin des études de ces programmeurs – mais traite à la racine les problèmes les plus fréquents.

Une autre solution, à plus court terme, serait de sensibiliser les équipes de développement aux tests de logiciels, pour améliorer la complémentarité des équipes de réalisation et de tests, faciliter les échanges et mettre en place des modèles de développement identifiant les défauts au plus près du moment où ils sont introduits. De telles solutions sont déjà disponibles auprès des organismes de formation accrédités par le CFTL, et offrent des réponses méthodiques s'inscrivant dans le cadre de ce que les éditeurs proposent comme environnements de développement (p.ex. : Visual Studio de Microsoft, ALM 11 de HP, suite Rational d'IBM).

L'identification de défauts effectuée par les développeurs n'ôte rien au travail des testeurs et permet à ces derniers de se focaliser sur des défauts plus complexes à identifier. Cela permet aux développeurs, un meilleur respect des délais en évitant de revenir sur des développements passés pour les corriger. Pour les projets, cela assure un meilleur respect des budgets et des délais en prévenant des allers-retours inutiles. Ces solutions ne transformeront pas le paysage des tests et du développement du jour au lendemain, mais elles ont déjà fait leurs preuves et sont un premier pas sur le chemin menant à des développements de qualité. Les développeurs ont donc, comme les testeurs, les AMOA et AMOE, besoin de connaître les tests de logiciels s'ils souhaitent concevoir des logiciels de qualité.

■ Bernard **Homès**
Président du Comité Français des Tests Logiciels. Auteur de ***« Les tests logiciels – Fondamentaux »*** (éditions Lavoisier) voir page 82.
Testeur certifié ISTQB niveau Avancé complet (TM, TA, TTA)

Tester, tester et re-tester !

Il est difficile de se faire une idée précise de la proportion de telle ou telle catégorie de tests dans un développement. On distingue deux grandes familles : les tests fonctionnels et les tests techniques. La partie fonctionnelle occupera environ 70 % du temps global de test. Les nouvelles générations d'outils se focalisent sur la qualité logicielle au cœur des projets, le retour sur investissement et la productivité, l'intégration aux environnements de développement et enfin l'intégration au cycle de vie.

Les tests visent à améliorer la stabilité, la qualité du code et in fine, le logiciel. Ils éliminent les dysfonctionnements et valident les choix techniques, l'architecture et les spécifications. La qualité logicielle peut se faire si les tests existent et sont appliqués méthodiquement. Plus un bug, un problème structurel / technique est découvert tard dans le codage, plus il coûtera cher à corriger. A contrario, plus un bug est découvert tôt, moins il coûtera cher. Pourquoi ? Sa découverte se fera au plus près de son « introduction ».

Ce n'est pas un hasard si les méthodes agiles préconisent et imposent des tests d'intégration, l'intégration continue et des batteries de tests chaque jour.

Cette méthode impose de définir dès le départ les batteries de tests (tests unitaires particulièrement) pour valider le code, les fonctions, les procédures, etc. A chaque intégration de code, les tests sont réalisés. Et un rapport de bug est alors généré. La non-régression doit être testée à chaque modification, chaque code rajouté. Mais, attention, chaque type de test correspond à une démarche, un objectif. Ainsi, le test unitaire est celui que vous allez le plus utiliser au quotidien (voir page suivante).

Certains outils autorisent l'ordonnancement de différents tests selon un ordre défini (= test ordonné).

Cela est très pratique quand vous avez des tests unitaires, tests d'interface, de charge et de performance [Fig.1].

Une application desktop et un site web partagent les mêmes catégories de tests mais pas forcément dans une proportion identique.

En développement web, outre la sécurité, les tests de charge, de performance, de stress seront à privilégier.

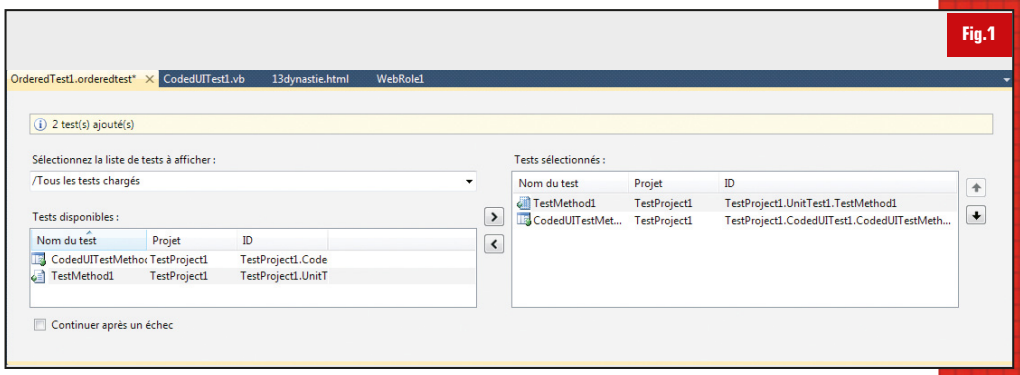


Fig.1

Conseil

Soyez ordonné, rigoureux. Un test ne peut réussir, ne peut être créé et organisé que si vous avez les idées claires. Un test mal calibré n'a aucun intérêt.

Ensuite, hiérarchisez les bugs découverts et revalidez chaque modification par de nouveaux tests. Test = cercle vertueux.

Testeur : un métier reconnu

Le Cigref définit le testeur ainsi : *« le « testeur », qui confirme l'importance apportée aux tests pour garantir la qualité des développements et préparer efficacement les déploiements. Il peut être pilote d'assurance produit, analyste test, homologateur ou qualifieur. On peut distinguer le testeur technique, développeur ou intégré à l'équipe de développement, et le testeur métier, en général dans les cellules qualité, maîtrise la partie métier et fonctionnelle ».*

Pour le CFTL (comité français des tests logiciels), il existe huit métiers du test logiciel : testeur, analyste de tests, analyste technicien du test, administrateur de plateformes de tests, consultant / consultant senior, responsable de méthode et de procédure de tests, chef de projets de tests. Il existe une certification pour les testeurs : l'ISTQB. Le CFTL le présente ainsi : « L'ISTQB (www.istqb.org) a mis en place un schéma de certification de testeurs composé de trois niveaux (Fondation, Avancé, Expert).

Ce schéma de certification est reconnu actuellement par 18 pays et a été validé par plus de 22 000 examens passés dans le monde.

Ceci fournit de nombreux avantages, tant pour les employeurs (niveau de compétences reconnu et consistant, avantage commercial par l'utilisation de personnel certifié, …) que pour les testeurs (preuve d'acquisition de connaissances, implication dans le métier, capacité à évoluer, …). ».

Pour aller plus loin

Emmanuel **Itié**, *Tester une application web*, éditions Eni

Bernard **Homès**, *Les tests logiciels*, Lavoisier

Benoît **Gantaume**, *JUnit*, éditions Eni

Collectif, *Pratique des tests logiciels*, Dunod

■ François **Tonic**

L'importance des tests unitaires

Les tests unitaires constituent les tests fondamentaux à effectuer sur le code. Ils vérifient si le code ne comporte pas de bug, d'erreur de programmation. Le test unitaire vérifie une partie du code, par exemple, en programmation objet, ce test ciblera la classe comme unité. Cela signifie que chaque classe est accompagnée d'une autre classe pour tester la première. Par exemple, un classe de test en groovy (http://groovy.codehaus.org/Unit+Testing) :

```
import groovy.util.GroovyTestCase

class MyTest extends GroovyTestCase {
  void testSomething() {
    assert 1 == 1
    assert 2 + 2 == 4 : «We're in trouble, arithmetic is broken»
  }
}
```

Le test unitaire est du code qu'il faut créer. Heureusement, des outils et frameworks sont là pour vous aider : JUnit, PyUnit, Unit Testing, etc. De nombreuses documentations sont disponibles en ligne, par exemple, autour de JUnit : http://junit.sourceforge.net/doc/cookbook/cookbook.htm et en Français l'excellent : http://www-igm.univ-mlv.fr/~dr/XPOSE2003/JUnit_tour/

Comme le montre le cycle, le test unitaire doit être votre compagnon du quotidien, vous permettant de qualifier, de valider votre code. Il vit avec votre code. Le test unitaire peut être généré automatiquement et vous pouvez l'adapter à votre besoin et surtout, à chaque changement important, le test doit être régénéré. N'oubliez jamais qu'un bon scénario de test est celui qui suit constamment le code. Bref, le test doit être mis à jour. Sinon, il ne servira à rien [Fig.2]. Dans l'exemple suivant, l'interface utilisation a été modifiée en mode design mais les tests unitaires et d'interface n'ont pas été modifiés / régénérés, d'ou les erreurs lors de la génération du projet [Fig.3].

Fig.2

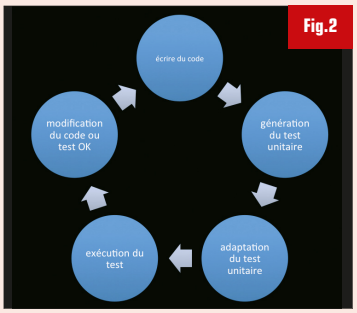
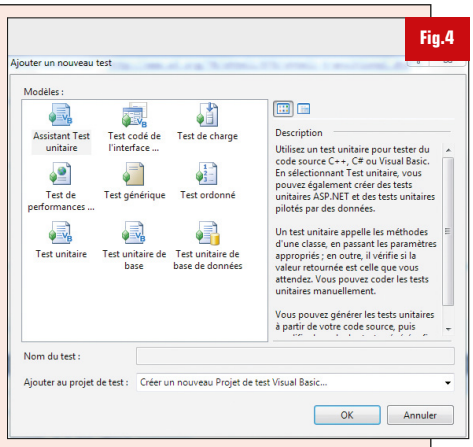


Fig.4



L'un des intérêts des tests unitaires est de procéder par *« assertions ». Il s'agit de mettre des expressions à vérifier. On utilise alors, comme en PHP, assert () et assert_options(). Le premier vérifie l'interface, le second, configure l'assertion. MSDN définit ainsi l'assertion : « Une assertion, ou instruction Assert, teste une condition, que vous spécifiez en tant qu'argument à l'instruction Assert. Si la condition a la valeur true, aucune action ne se produit. Si la condition a la valeur false, l'assertion échoue. S'il est exécuté avec une version Debug, votre programme passe en mode arrêt. ».* En C#, on peut utiliser la classe assert pour vérifier les conditions via true et false.

Les méthodes disponibles sont nombreuses. Par exemple, la possibilité de vérifier que la condition spécifiée est vraie. L'assertion échoue si la condition est fausse :

```
public static void IsTrue(
  bool condition,
  string message,
  params Object[] parameters
 )
```

Pour en savoir plus :
http://msdn.microsoft.com/en-us/library/ms244252.aspx

Fig.3

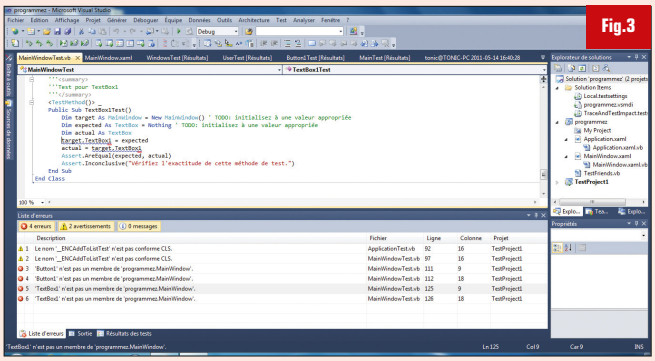
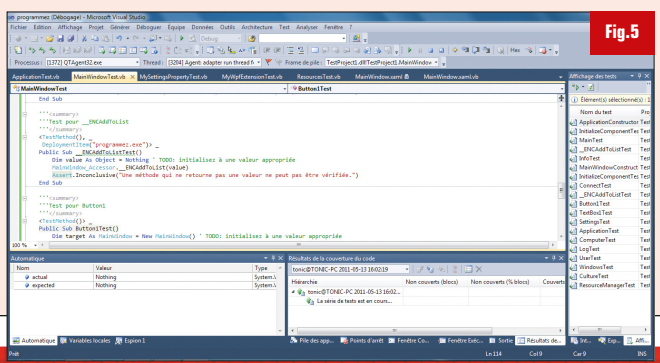


Fig.5



Une intégration de plus en plus poussée

Le test unitaire est ce que l'on pourrait appeler le métrique de base de la qualité pour le développeur. Prenons Visual Studio 2010. L'intégration des modules de tests directement dans l'IDE (nous avons utilisé l'édition Ultimate) est assez remarquable [Fig.4].

La démarche unitaire dans VS 2010 est assez simple :

1. Menu Test -> Nouveau Test
2. Assistant Test Unitaire : l'assistant crée le test unitaire à partir du code projet en cours. Il suffira ensuite de le personnaliser. Autre option : Test unitaire de base ou Test unitaire
3. Un nouveau projet est automatiquement créé (ex. : UnitTest1.vb) [Fig.5].

C'est tout. L'exécution du fichier de test dans VS lancera les tests unitaires. Exemple : http://blogs.dotnet-france.com/julienc/post/Introduction-aux-tests-unitaires-avec-Visual-Studio-2008.aspx

Les différents types de tests automatisés

La seule façon de prouver qu'un développement fonctionne comme attendu, c'est de le tester. Le code évolue avec le temps, il faut donc tester constamment. Que peut-on automatiser ?

Différents types de tests concernent différentes parties du code : [Fig.1]. La première étape est de tester chaque comportement technique en isolation. Ce sont les tests dits « unitaires ». Les interfaces web doivent être testées d'une autre manière, avec des robots qui navigueront automatiquement. Ce sont les tests IHM. Pour vérifier que le développement de l'application correspond effectivement aux demandes métier, la spécification peut être écrite sous forme de tests. Ceux-ci valideront également que les comportements unitaires fonctionnent bien ensemble. Les applications communiquent avec des systèmes externes comme des bases de données, des fichiers ou d'autres applications. Les tests d'intégration sont des tests qui valident ces interactions. Ils sont joués sur un environnement d'intégration où tous les systèmes sont déployés. Nous allons détailler sous forme de fiches techniques ces différents types de tests.

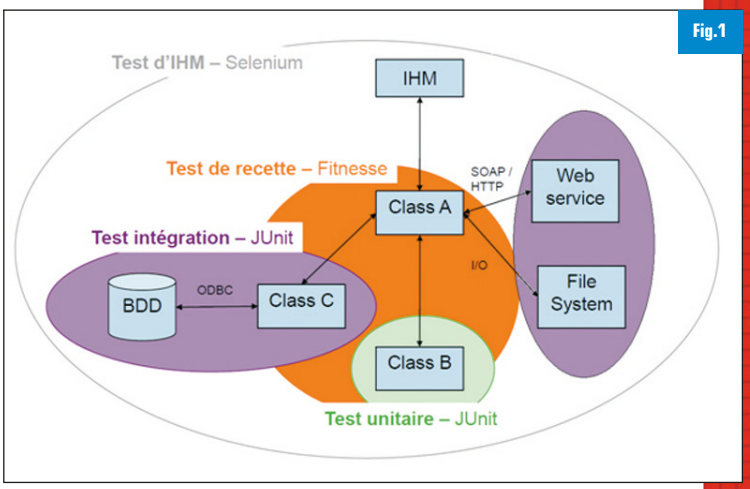


Fig.1

TESTS UNITAIRES

Pourquoi ?

Un test manuel permet de valider un ensemble de comportements à un instant T. L'idéal serait en fait de tester tous les comportements pour tous les instants.

Le test unitaire est un test technique automatisé qui permet de valider un unique comportement.

Le test unitaire doit être indépendant du reste du programme. Imaginons que vous devez parcourir 5 étapes dans votre application avant de tester votre comportement. Si le test échoue, la cause peut être cachée dans n'importe laquelle de ces 5 étapes.

Enfin, **le test unitaire doit être déterministe**. A chaque fois que la méthode testée est appelée avec les mêmes paramètres, le résultat doit être identique.

Qui ?

Les développeurs doivent apporter la preuve que ce qu'ils ont produit fonctionne. C'est donc eux qui ont la responsabilité des tests unitaires.

Quand ?

La façon la moins coûteuse de tester est d'écrire un test avant d'écrire le code. Ainsi, nous validons les points suivants :

- Tout code écrit est testable.
- Tout code écrit est testé.
- Le code est prouvé, non constaté.

Tout code écrit n'est pas forcément testable unitairement. Si plusieurs parties du code ont des interactions fortes, il sera difficile de les isoler pour les tester unitairement. Tester avant permet de veiller à ne pas créer de code fortement couplé.

Qu'est ce qu'on teste ?

Tout comportement doit être testé. S'il est possible de retirer un morceau du code sans casser les tests, alors vous n'avez pas complètement testé. Prenons l'exemple suivant :

```
public Character premierCaractere(String s) {
  Character c = null;

  if (s != null && !s.isEmpty()) {
    c = s.charAt(0);
  }

  return c;
}
```

Cette méthode a trois comportements :

1 - Si **s** est null.

2 - Si **s** est vide.

3 - Si **s** n'est ni null ni vide.

Il y a donc trois cas à tester, donc au moins trois tests à écrire.

Comment tester ?

Par convention, toutes les méthodes d'une classe sont testées dans une même classe. Chaque comportement est testé dans une méthode différente, dont le nom ou le commentaire permet de comprendre ce que fait le test.

Pour un test unitaire simple, en Java avec JUnit 5 :

```
@Test
public void premierCaractere_stringNull() {
  // Given
  String s = null;
```

```
  Character expected = null;

  TraitementString ts = new TraitementString();

  // When
  Character actual = ts.premierCaractere(s);

  // Then
  Assert.assertEquals(actual, expected);
}
```

Ici nous vérifions que la chaîne **s** est nulle.

- Soient (Given) : une chaîne d'entrée nulle et un résultat attendu nul
- Quand (When) : appel de la méthode testée
- Alors (Then) : vérification que le résultat obtenu est égal au résultat attendu.

Rappelez vous, unitaire veut dire : un et un seul comportement. Que faire quand la méthode que vous testez appelle une autre méthode ?

```
public Character premierCaractere(String s) {
  Character c = null;

  if (s != null && !s.isEmpty()) {
    c = service.uneAutreMethode(s);
  }

  return c;
}
```

Considérons que toutes les méthodes externes à notre classe fonctionnent. Il est inutile d'appeler le vrai service. Créons donc un « faux service » qui renvoie toujours la même valeur. On appelle cela un Mock. Voici un exemple avec le framework Mockito :

```
@Test
public void premierCaractere_stringNull() {
  // Given
  String s = null;
  Character expected = null;

  Service serviceMock = Mockito.mock(Service.class);
  Mockito.when(serviceMock.uneAutreMethode(s)).thenReturn(null);

  TraitementString ts = new TraitementString();
  ts.service = serviceMock;

  // When
  Character actual = ts.premierCaractere(s);

  // Then
  Assert.assertEquals(actual, expected);
}
```

Dans ce code, nous avons déclaré un mock de Service : serviceMock. Ensuite, nous avons indiqué que lorsque la méthode uneAutreMethode est invoquée avec notre chaîne **s** en argument, le mock retourne null.

TEST D'IHM

Pourquoi ?

Attention, tester l'ihm et tester en utilisant l'ihm, ce n'est pas la même chose ! Le test d'IHM valide les comportements graphiques, et en assure la non-régression. Affichages conditionnels, positions des éléments, navigation… autant d'éléments susceptibles d'être testés par ce test. Vous pouvez donc déconnecter complètement votre IHM du reste de votre application (à l'aide de mocks) et ces tests devraient encore fonctionner.
Si vous vous servez de l'interface pour faire des tests fonctionnels, alors ce sont des tests de recette.

Qui ?

Les experts fonctionnels rédigent les scénarios de tests. Le robot va simuler le comportement humain. Sélenium est un outil permettant de faire ces tests. Il comprend un enregistreur qui observe l'utilisateur afin d'être capable de rejouer le scénario automatiquement. Pour que tout fonctionne bien, une fois les scénarios enregistrés, les développeurs doivent retoucher un peu le code généré par l'enregistreur.

Quand ?

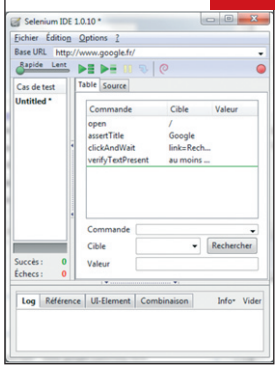
Le plus tôt possible, comme pour tous les tests (cf trucs de pros). Attention : les tests d'IHM sont très sensibles aux changements. Si votre IHM change fréquemment, vos tests devront évoluer. Ceci a un prix. A vous de trouver le meilleur compromis.

Qu'est ce qu'on teste ?

Ce sont les comportements graphiques uniquement qui sont validés par les tests d'ihm. Soit deux menus déroulants, un pour le pays, et un pour la région. Vous voulez faire apparaître le deuxième menu seulement quand un pays est sélectionné. Ce genre de comportement purement graphique peut être surveillé. Si en cours de développement de l'application le code qui fait apparaître le deuxième menu ne fonctionne plus, il faut pouvoir le repérer au plus vite.

Comment on teste ?

Fig.2



Deux approches possibles : par enregistrement ou par programmation. L'enregistrement fonctionne de manière très intuitive : appuyez sur «enregistrer», effectuez votre test et sauvegardez le code généré.
Avantage : c'est très facile.
Inconvénient : Certains comportements ne sont pas générables par cette méthode.
Exemple d'interface d'enregistrement : [Fig.2]. Développer ces tests directement sera la charge du programmeur.
Avantage : un test directement efficace et une page codée pour être facilement testable
Inconvénient : travail de développement supplémentaire.
Exemple de code du framework Sélénium :

```
public void testGoogle()  {
    selenium.open(«http://www.google.fr/»);
    assertEquals(«Google», selenium.getTitle());
```

```
    selenium.clickAndWait(«link=Recherche avancée»);
    verifyTrue(selenium.isTextPresent(«au moins un des mots suivants»));
}
```

LES TESTS DE RECETTE AUTOMATISÉS

Pourquoi ?

Avant une mise en production, le code est souvent gelé une semaine à un mois pour une campagne de tests de non-régression. Tout bug trouvé pendant cette période est extrêmement coûteux : il faudra vivre avec en production, ou bien recommencer entièrement la campagne. Les tests fonctionnels automatisés permettent de réduire drastiquement cette campagne. Ils testeront tous les cas répétitifs. Par exemple, pour une recherche sur plusieurs critères, il sera aisé de tester les combinaisons de tous les critères. Ces tests permettent donc d'**assurer l'application contre la régression**. L'étude détaillée de chaque point, soulignée par des exemples concrets, permet de lever bon nombre d'ambiguïtés. Cela permet également de vérifier la cohérence du cahier des charges. Enfin, cela donne aux développeurs un guide pour leur permettre de n'oublier aucune règle. Si ces tests sont bien organisés, une page récapitulera tous les critères de la recherche demandée. Elle explicitera les différentes combinaisons possibles.
Les tests fonctionnels en avance de phase permettent de **lever toute ambigüité sur le besoin**.
Ces tests peuvent être rédigés avant les développements. Dans ce cas, l'outil de tests indiquera le nombre de tests en succès. Cet affichage **donne une bonne vision de l'avancement des développements, en termes de fonctionnalités.** Sur notre recherche, chaque jour, lancer les tests automatisés permet de se rendre compte du nombre de critères implémentés dans le code.

Qui ?

Dans une entreprise classique, ce sont les experts fonctionnels (MOA) qui rédigent ces tests. Ils sont ensuite retravaillés par les développeurs (MOE), afin d'en faciliter l'implémentation.

Quand ?

Idéalement, ces tests sont rédigés avant de commencer à développer. S'ils sont rédigés après, l'unique bénéfice tiré est la non-régression.

Qu'est ce qu'on teste ?

Les tests fonctionnels automatisés permettent de tester des règles métiers complexes. Vous avez une recherche à plusieurs critères, dont les critères interagissent ? Vous avez un générateur de devis, qui en fonction des données entrées présente un prix ? Les tests fonctionnels sont alors incontournables.
Points importants :

- Tester la couche la plus haute possible, souvent le contrôleur.
- Chaque test doit être indépendant. Nettoyez consciencieusement vos données insérées en base.

Comment on teste ?

Différents outils proposent d'automatiser les tests fonctionnels. Ils sont le plus souvent basés sur Wiki. Les plus répandus sont Fitnesse et Greenpepper, qui permettent d'écrire les ensembles de règles de gestion (spécifications) et leurs tests dans un même endroit. Il faut utiliser la partie test de l'outil comme un remplacement de l'interface : *« si telle donnée est envoyée au serveur, il va renvoyer telle autre donnée »*.
Commencez par découper votre logiciel en fonctionnalités. Essayez de les rédiger sous la forme *« En tant que … je peux … pour … »*. Pour chaque fonctionnalité, créez une page où vous rappelez toutes les règles que les développeurs devront implémenter. Ensuite, décrivez votre jeu de données d'entrée sous la forme d'un tableau. Puis décrivez le comportement attendu.Les développeurs reprennent alors les tests, les mettent en forme, et écrivent une couche technique qui leur permet d'appeler leur application, d'insérer les données avant le test et de les supprimer ensuite.
Sur le wiki, vous aurez alors la possibilité de lancer vos tests.

Exemple de test :

En tant que télé-gestionnaire :
J'effectue une recherche avancée sur un ou plusieurs critères, et je vois une liste des tarifs déjà émis afin de pouvoir immédiatement sélectionner le plus rapidement possible le tarif en question –sachant que j'ai l'internaute au bout du fil.
Les différents critères étant :
- Type de tarif : choix dans une liste
- Date de création : entre 2 dates au format jj/mm/aaaa
- Nom : recherche texte de type contient (LIKE)
- Prénom : recherche texte de type contient (LIKE)
- Email : recherche texte de type contient (LIKE)
- Code postal : recherche commence par

Les critères sont combinatoires (critères liés par une condition ET).

• Insertion des données

script	Soient les tarifs		
ajouter un tarif avec l'id	1	le prenom	jean
ajouter un tarif avec l'id	2	le prenom	marc

• Test
On peut faire des recherches sur le prénom

Recherche par prenom	
prenom	nombre de resultat
jean	1
yves	0

• Suppression des données

script	Soient les tarifs
supprimer le tarif avec l'id	1
supprimer le tarif avec l'id	2

LES TESTS D'INTÉGRATION

Pourquoi ?

Les applications ont souvent besoin de communiquer avec d'autres systèmes : fichiers, base de données, autres applications… Ces communications sont coûteuses en temps et en argent : il faut éviter d'y recourir durant le développement. Pour développer, il faudra isoler l'application de ces systèmes, en les simulant. Ensuite, deux étapes sont nécessaires pour tester ces communications. Vérifier que le système externe fonctionne, puis valider la communication en elle-même. Cette validation, c'est le test d'intégration. Déployez votre application et les systèmes dans un environnement dit "d'intégration", dans des conditions identiques à l'environnement de production, et jouez ces séries de tests à chaque nouvelle version de l'application.

Qui ?

Les développeurs écrivent ces tests. Les responsables des systèmes externes seront sollicités pour la mise en place de l'environnement.

Quand ?

Au plus tôt. En effet, les problématiques d'intégration sont souvent très consommatrices en temps. Les tests vérifiant la configuration de votre application pourront être écrits au début de votre projet. Après les développements, vous vous êtes assuré que votre application fonctionne. Vous n'avez plus qu'à la connecter aux "vrais" services externes. C'est le moment de rédiger de nouveaux tests d'intégration.

Qu'est ce qu'on teste ?

Ces tests se font sur l'application complète, de bout en bout. Appelez les contrôleurs, voire même les vues, qui iront chercher ses données dans la vraie base et dans le vrai webservice.
Les mêmes outils que pour les tests d'IHM pourront être utilisés, en naviguant d'une application à l'autre.

Comment on teste ?

Pour tester la configuration d'un projet, ses accès aux fichiers ou aux bases de données, il est possible d'utiliser les frameworks de tests unitaires. Pour toute communication avec d'autres systèmes, déployez-les en condition réelle et utilisez les dans leurs ensembles.

■ Jonathan **Scher**,
Consultant Agile chez OCTO Technology
jscher@octo.com
Blog : http://slicesofit.com/

■ Maxime **Arnstamm**
Consultant Agile chez OCTO Technology
marnstamm@octo.com
Blog : http://slicesofit.com/

Les astuces des *pros*

Un bon test unitaire est un test simple. Et rédiger quelque chose de simple n'est pas facile. Vous ne savez pas par où commencer ? Vous vous demandez comment tirer un maximum de vos tests ? Voici quelques astuces pour vous y aider.

Given When Then

Comme indiqué dans la fiche pratique du test unitaire, cette structure vous permettra d'avoir des tests plus lisibles. Dans la partie "Given", vous initialisez toutes les variables nécessaires à votre test. Dans la partie "When", vous lancez l'action à tester. Enfin, en "Then" vous mettez votre assertion qui vérifie le résultat obtenu.
Pour chaque fonction, vous voudrez écrire plusieurs tests pour les cas nominaux, et pour les cas d'erreur (l'ensemble des comportements en fait). Pour chaque test, n'hésitez pas à rédiger une fonction différente, qui positionne (éventuellement de la même façon) un environnement, qui appelle une méthode, puis que vérifie le résultat. En évitant de repositionner l'environnement au sein de la même méthode de test, chaque assertion devient plus lisible. De plus, chaque test devient indépendant du précédent : une erreur non vue dans le premier test ne viendra pas tromper le résultat du second. Plus vos tests seront petits et focalisés, plus ils vous fourniront d'information lorsqu'ils passeront au rouge.

Ecrivez à l'envers !

Penser à toutes les conditions initiales pour votre test avant de l'écrire est parfois difficile. Ecrivez-le donc à l'envers ! Exemple en java avec Spring MVC et Mockito : *"Je veux vérifier que le contrôleur redirige vers la page de recherche"*

```
// Then
assertEquals(«redirect:search.html», result);
```

"Je veux vérifier… quand je lance une recherche"

```
// When
String result = controller.search(form);

// Then
assertEquals(«redirect:search.html», result);
```

"Je veux vérifier… quand je… sur l'Id 492 alors que le service associé ne retourne rien".

```
// Given
  int searchId = 492;

  Controller controller = new Controller();
  Service service = Mockito.mock(Service.class);
  controller.setService(service);
  Mockito.when(service.findById(searchId)).thenReturn(new Client
(searchId));

  Form form = new Form();
  form.setId(searchId);

  // When
  String result = controller.search(form);

  // Then
  assertEquals(«redirect:search.html», result);
```

Toutes ces conditions auraient été difficiles à écrire depuis une page blanche. En pensant à l'envers, je découvre au fur et à mesure tout juste ce dont j'ai besoin pour mon test.

Vos données sont fixées

Il est difficile d'attendre un résultat prédéfini si votre code testé repose sur un générateur de nombres aléatoires, ou sur une date qui varie par exemple. Dans ce cas, vous aurez besoin de découpler votre code du générateur aléatoire, puis d'injecter votre propre générateur fixe. Par exemple, si vous voulez tester le code suivant :

```
public long demain() {
      return new Date().getTime() *24*60*60*60*100;
  }
```

Commencez par le refactorer :

```
Date date = new Date();
  public long demain() {
      return date.getTime() *24*60*60*60*100;
  }
```

Entre la création de l'objet et l'appel de la méthode, vous pouvez désormais modifier le générateur de date.

Ne testez pas le framework.

Lorsque vous créez un test unitaire, vous supposez que tout ce qui est externe à la fonction testée fonctionne correctement. Les méthodes fournies par les frameworks que vous utilisez, ou le code qui est généré ne sont pas à tester : il faut partir du principe qu'ils fonctionnent. De même, ne faites pas d'appel aux webservices, mockez-les. Idem pour la base de données, ou pour la couche de service sous votre contrôleur. Mocker un service, c'est faire appel à un « faux » service qui a les mêmes interfaces que le vrai, mais qui répond toujours une valeur pré-définie. Utile, pour tester ! Ce sont les tests d'intégration qui se chargeront de vérifier que votre application une fois assemblée est bien fonctionnelle…

Stratégie pour tester du code existant.

Vous avez une application complète, mais non testée automatiquement ? Vous souhaitez vous mettre aux tests automatisés, mais vous ne savez pas comment et par quel bout commencer ?
Mettre tous les tests sur toute l'application serait très long et fastidieux. La stratégie que je vous propose permettra d'identifier les points critiques, et de se concentrer sur le test à ce niveau.
Commençons par le test unitaire. C'est le plus petit test, le plus facile à mettre en place. Imposez-vous la règle suivante : *"tout bug remonté doit être mis en évidence par un test avant d'être corrigé"*. L'idée est de se dire : il y a un bug, je fais un test unitaire qui provoque ce bug, puis je corrige ce bug. Et tant que ce test sera bon, vous saurez que le bug ne se reproduira pas.
Au début, la moindre petite correction va être très longue à corriger. Mais au fur et à mesure, les parties critiques de votre application seront couvertes de tests. Au bout d'un certain temps, vous pourrez proposer de mettre en place les tests unitaires également pour les nouvelles fonctionnalités.
Si vous suivez ce plan, votre code deviendra de plus en plus testable. Au bout d'un moment, vous pourrez ajouter des tests fonctionnels automatisés, pour chaque nouvelle fonctionnalité.

Test Driven Development

L'idée du TDD est de piloter votre développement par vos tests unitaires. C'est-à-dire que vous n'écrivez du code que lorsque le test associé a été écrit. Ainsi, vous aurez une couverture maximale de votre code par vos tests, et vous garantirez que votre code est testable. Un code bien testable est un code faiblement couplé - une grande qualité du point de vue de l'architecture du code.
Comment cela se pratique ? En trois mots : **test, code, refactor**.
Vous écriviez un test pour un comportement unitaire. Vous le lancez, et celui ci est rouge : vous avez prouvé que votre test demande une modification de votre code. Vous codez, de la manière la plus idiote possible pour faire passer votre test. Hop, il est vert. Enfin, vous passez un moment à refactorer votre code, afin d'enlever la duplication et de le simplifier au maximum. Et vous recommencez avec un autre cas.

Pourquoi tester au plus tôt ?

Combien coûte un bug ? C'est toujours difficile à estimer. Imaginons que le bug fait que les achats effectués sur votre site Internet ne sont pas enregistrés en base de données. Vous le trouvez deux jours après la mise en production. Catastrophe ! Vous avez perdu deux jours de vente. A cela, il faut ajouter les équipes que vous allez restaffer sur le bug en question. Ajoutez également une recette entière : il ne faudrait pas que la correction en urgence génère d'autres bugs. Bien. Si le bug est détecté en recette maintenant. Il va falloir qu'un développeur se penche sur le sujet. Si cela fait plusieurs mois qu'il a développé sa fonctionnalité, ce ne sera pas simple. Beaucoup de temps sera perdu.
Imaginons que vous ayez un test de recette, qui indique immédiatement à votre développeur que sa fonctionnalité a créé un bug, le soir même. Simple, il a juste à étudier le code qu'il vient d'écrire, et à trouver le bug en question. Et si, au moment où il introduit sa fonction, un test unitaire lui indique immédiatement le problème ? Hop ! En quelques minutes, c'est codé. Cette loi se nomme la loi de l'augmentation du coût du défaut. « Defect Cost Increase ». Plus un bug est découvert tard, plus le coût engendré est élevé. Plus vos tests sont effectués tôt, moins le coût est élevé.
Testez au plus tôt, cela vous évitera de payer cher plus tard !

Intégration continue

Votre code est testé ? Bien. Mais que de tests différents ! Lesquels devons nous passer, et à quelle fréquence ? L'intégration continue est là pour ça. Que diriez-vous d'une application qui, à chaque fois que vous annoncez que votre code a été modifié, repasse les tests qui vont bien ? Voilà, il s'agit de l'intégration continue. Hudson est une des applications d'intégration continue les plus connues. Il va surveiller votre gestionnaire de sources (svn, git, ou autres), et à chaque modification, il va télécharger et recompiler votre application. Si la compilation se passe mal, il vous enverra un email. Si la compilation réussit, il passera les tests unitaires. Pour les autres tests, vous avez le choix. Vous pouvez les lancer à chaque construction. Cependant, attention à la durée. Les tests unitaires sont rapides à passer : ils peuvent être lancés à chaque modification mineure. Les tests fonctionnels, d'intégration, ou d'IHM sont souvent plus longs, et demandent plus de ressources. Vous pourrez demander à votre serveur de ne les lancer qu'une fois par jour, et de vous en faire un rapport !

■ Jonathan **Scher**,
Consultant Agile chez OCTO Technology
jscher@octo.com
Blog : http://slicesofit.com/

■ Maxime **Arnstamm**
Consultant Agile chez OCTO Technology
marnstamm@octo.com
Blog : http://slicesofit.com/

Serendipity Testing : l'art d'identifier des bugs au hasard

Il vous est sûrement déjà arrivé de visiter un endroit que vous ne connaissiez pas, avec une carte à la main, et au bout d'un certain temps, vous vous êtes écarté de votre itinéraire initial, puis vous êtes tombé sur un endroit qui a profondément marqué votre esprit.

Combien de fois, lors d'une soirée chez un copain, vous avez pris la liberté de consulter sa bibliothèque ou ses CD et vous avez découvert un auteur ou un groupe de rock que vous n'avez pu depuis oublier. Un jour, au milieu d'une démo improvisée ou tout simplement en investiguant un problème applicatif, il vous est peut-être arrivé d'identifier un bug n'ayant rien à voir avec le problème initial, et vous vous êtes rendu compte que ce bug était là depuis le début du projet sans que personne ne s'en aperçoive. Dans les trois cas, vous avez été, sans le savoir, la victime heureuse de la **"Sérendipité"**.

La sérendipité (Serendipity) est le fait de *« Chercher quelque chose, puis trouver autre chose, et se rendre compte que ce que vous avez trouvé est plus adapté à vos besoins que ce que vous avez recherché »* (Lawrence Block)

D'autres définissent la sérendipité comme *« le fait de réaliser une découverte inattendue grâce au hasard et à l'intelligence, au cours d'une recherche dirigée initialement vers un objet différent de cette découverte »* (Ginzburg Carlo). Pour Robert King Merton, la sérendipité est *« l'observation surprenante suivie d'une induction correcte »*.

Daniel B. Klein définit la sérendipité comme *« une découverte majeure qu'une personne ne recherchait pas, qui modifie sa propre interprétation de ce qu'il était en train de faire, et qui se révèle évidente au découvreur »*. En d'autres termes, une solution meilleure que la précédente sans connaître la solution idéale au départ.

Le concept de la sérendipité est utilisé dans plusieurs domaines. On trouve ses applications en stratégie d'entreprise ou encore en psychologie, et surtout chez certains chercheurs qui se sont penchés sur l'analyse de l'impact de certains évènements accidentels (la sérendipité sociale et professionnelle) dans la carrière de comédiens, de footballeurs ou de chanteurs d'opéra. Toutefois, le domaine où la sérendipité a été largement théorisée puis appliquée est « la recherche documentaire ». Ainsi, *« la recherche par sérendipité permet de prendre conscience des itinéraires pas nécessairement linéaires pour trouver une solution »*.

> ***"Comment apprivoiser un tel phénomène dans le contexte du développement logiciel, qui par définition échappe au hasard et à toute notion de coïncidence?"***

Pourtant, avec l'émergence des réseaux sociaux, les grands acteurs du monde du digital et des plateformes sociales se sont emparés de la sérendipité et chacun essaie tant bien que mal de transformer les expériences de ses utilisateurs en rencontres émouvantes, engageantes, riches et agréables avec des approches diverses et variées, dont voici quelques exemples :

- iPod avec sa fonction de sélection aléatoire des morceaux de musiques.
- Les applications comme TweetRiver, Tweepular, TweetSum, TwitZap, TwitHive, Tweenky, Tweetree, Splitweet et CoTweet qui gravitent autour de la plateforme Twitter.
- L'application iPhone « UrbanSpoon » qui vous propose une sélection aléatoire des restaurants recommandés.
- StumbleUpon (un service Web) qui vous suggère un contenu susceptible de vous intéresser.

Quelle est la relation entre la sérendipité et le software testing ?

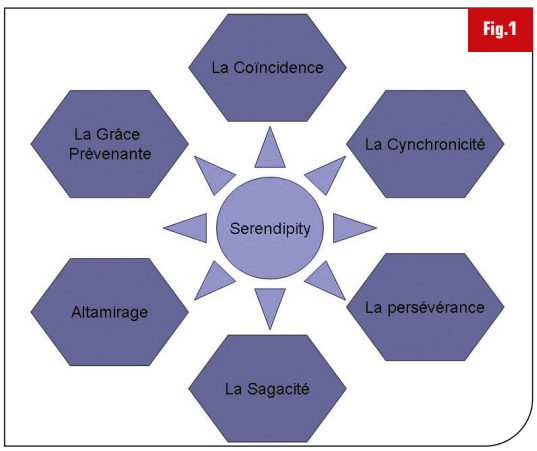
Et bien, si l'on regarde de très près l'évolution des méthodologies de test logiciel on peut se rendre compte que les tests « scénarisés » (ou scriptés) ont démontré leurs limites depuis quelques année déjà. C'est pour cette raison que les professionnels de testing ont introduit une nouvelle approche de testing dite « Exploratory Testing ». Nous ne traiterons pas dans cet article les techniques de tests exploratoires, mais nous allons juste souligner les deux notions fondamentales à savoir le caractère « exploratoire/découverte/freestyle » et « l'apprentissage continu par le test ». Le principe des tests exploratoires réside dans cette « liberté/improvisation » associée à une « responsabilité » qu'on accorde davantage au testeur. Désormais, on demande au testeur d'être « créatif » et non pas juste d'écrire et d'exécuter un scénario de test.

D'autre part, Cem Kaner décrit le testing comme une activité de *« recherche permanente d'informations »* un domaine où la sérendipité a trouvé plusieurs applications. Dans son livre *« Chance in the Midst of Design: Approaches to Library Research Serendipity »*, Daniel Liestman définit un ensemble de six approches différentes de la sérendipité dans le domaine de la recherche documentaire : [Fig.1].

La coïncidence

Rassurons tout de suite nos chers lec-

Fig.1



teurs sur le fait que nous ne recommandons pas des tests logiciels basés sur le « hasard », Daniel Liestman cite ici la coïncidence comme l'effet de « surprise » ou le fait de retrouver au bon endroit au bon moment, les Anglo-Saxons ont un terme précis « happenstance ». Dans notre contexte de testing logiciel, cette notion peut être remplacée par la multiplication des expériences les unes différentes des autres lors d'une campagne de Serendipity Testing. Ainsi, la reproduction rigoureuse d'une première observation d'un bug, qualifié initialement « d'aléatoire », permet d'établir qu'il s'agit bien d'un fait réel, puis d'en étudier les mécanismes. C'est aussi la capacité d'ouverture du testeur, à la nouveauté, aux idées originales, différentes et inhabituelles qui est mise en valeur et qui peut lui apporter le succès. Autrement dit, une démarche de Serendipity Testing favorise l'intuition et la curiosité constructive du testeur.

La grâce prévenante

Liestman appelle la *grâce prévenante*, l'aboutissement du travail d'un documentaliste grâce à l'exploration d'un corpus documentaire bien classifié. En d'autres termes, les découvertes fortuites sont le résultat d'une mise à disposition « experte, judicieuse et préalable » de l'information par d'autres personnes. Dans le cas d'une activité de testing, cela se traduit par la capacité d'une application à fournir des informations utiles et par conséquence, d'un testeur à exploiter toutes les données mises à sa disposition pour explorer efficacement un logiciel, cela englobe : la qualité et la pertinence des logs, l'utilisation des outils de profiling/debuging, etc.

La Synchronicité

Liestman emprunte le terme semi-mystique de la Synchronicité à Jung pour décrire l'occurrence simultanée d'au moins deux événements qui ne présentent pas de lien de causalité, mais dont l'association prend un sens pour la personne qui les perçoit. Dans notre contexte de testing, le concept de *synchronicité* peut être utilisé par le testeur pour l'aider à provoquer un parallélisme des événements qui n'ont pas un lien causal entre eux. Ainsi, le testeur peut découvrir un nouveau bug applicatif comme le résultat de l'enchaînement de deux processus (actions) synchrones.

La persévérance

Ici on souligne le caractère systématique du Serendipity Testing, plus le travail du testeur dure dans le temps, plus la probabilité sera en faveur des découvertes imprévues et des expériences fortuites. Le testeur doit être en permanence préparé et perceptif, dans le cas contraire il risque de passer à côté de certaines anomalies qui méritaient d'être remontées. Cette quatrième approche nous montre bien que l'effort et la ténacité peuvent certainement conduire à des résultats surprenants.

Puisque le processus peut s'avérer à la fois complexe et long, Liestman rappelle qu'un documentaliste doit reconnaître quand il a atteint le point du rendement décroissant. A un moment donné, le testeur doit se demander si le retour sur investissement est au rendez-vous.

Altamirage

L'*altamirage* suppose que les comportements « individuels » « peu orthodoxes » et « idiosyncrasique » conduisent inexorablement à des accidents heureux. C'est aussi la capacité de faire une découverte inattendue suite à une action hautement individualisée. *Altamirage* va bien au-delà des limites de la sérendipité en mettant l'accent sur le rôle de l'action personnelle du testeur. Le concept du « Boundary Testing » bien connu des professionnels des tests logiciels se rapproche légèrement de la notion d'*altamirage*.

Sagacité

Dernière méthode développée par Liestman, la sagacité exige *« l'intuition et l'habileté de la part du documentaliste, sans tomber dans la rigueur, la persévérance ou l'altamirage »*. Dans notre contexte, on peut dire que cette approche repose sur l'enthousiasme du testeur à l'égard de son sujet, la souplesse de sa pensée, la sensibilité au moindre indice fortuit et le désir débordant de faire des découvertes.

On peut bien sûr ne pas maîtriser la sérendipité, mais au moins on peut créer les conditions et les environnements favorables où elle a le plus de chances de se manifester. Dans la suite de cet article, nous allons détailler quelques actions concrètes qui peuvent être appliquées.

La lecture des rapports d'anomalie et des logs est une affaire de tous les testeurs

Je rajouterais même de tous les développeurs. Par les rapports d'anomalies on sous-entend, les notifications email envoyées par les systèmes de bug tracking dès qu'une anomalie a été ouverte, les incidents qui surviennent en production, ou tout simplement les tickets qui transitent par le Support Technique/Client. Je ne crois pas que tous les testeurs sont abonnés à cette mine d'information. L'objectif ici n'est pas de les résoudre, mais en lisant ces différents rapports, le testeur aura toute la chance d'identifier des combinaisons de cas/événements auxquelles il n'aurait jamais pensé. En lisant les rapports d'anomalie, Brian Marick recommande de garder dans l'esprit le quadruplet (menace, produit, problème, victime) : [Fig.2].

La **menace** représente les inputs ou l'environnement qui risque de ne pas être géré correctement par l'application testée. **Le produit** qui doit transformer ces inputs sur

Fig.2

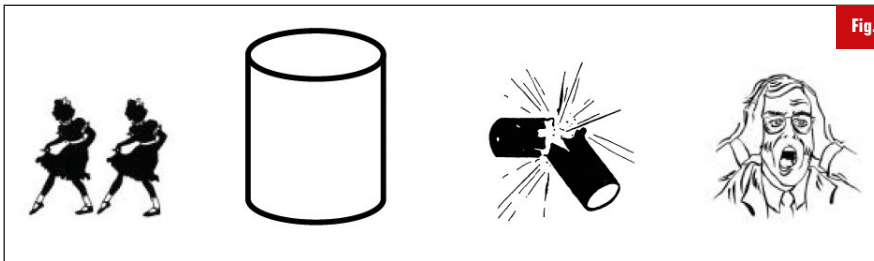
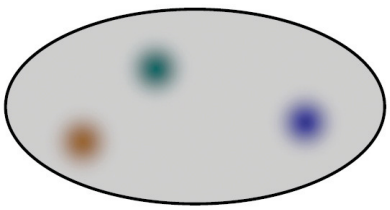


Fig.3



un environnement donné. Si le produit est incapable de gérer la menace, il provoque **un problème** sous la forme d'un mauvais résultat. Mais un problème n'a que peu de signification à moins qu'il n'existe une **victime** endommagée par ce mauvais résultat. En variant ces quatre éléments, le testeur peut trouver d'autres situations potentiellement source de bugs. Autre mine d'information à laquelle un testeur fonctionnel ne prête souvent pas attention : les logs applicatifs. On ne consulte souvent les logs que lorsqu'un problème applicatif nous a été remonté. En règle générale, le développeur scrute les logs pour comprendre l'origine du problème et voir comment il peut le corriger. Or, on peut faire le chemin inverse, à savoir, sans aucune anomalie apparente, en regardant les logs, on peut identifier par exemple des erreurs qui se succèdent d'une manière systématique à certains endroits, à un certain moment, ou alors avec un jeu de données bien particulier (car nous avons mis au préalable les traces en mode verbeux) avec un peu de reverse engineering, le testeur peut construire tout un scénario (fonctionnel cette fois-ci) qui reproduira les erreurs qu'il a observées initialement par « hasard ».

Comprendre le produit

Je me souviens bien que quand j'étais ingénieur de test & validation, je demandais souvent aux développeurs de m'expliquer le fonctionnement de certaines briques logicielles et l'architecture du produit dont j'avais la charge. A chaque fois le développeur me disait, *« mais pourquoi as-tu besoin de ces informations ? »*. Et bien, si vous connaissez la structure interne d'un système IT et les interdépendances entre tous les modules qui le composent, vous pouvez prendre les bonnes décisions et surtout les bonnes orientations dans votre démarche de Serendipity Testing. Cette connaissance vous permettra : [Fig.3 et 4].

- D'éviter les parcours redondants.
- Si une anomalie a impacté un module particulier, vous pouvez diriger vos efforts vers tous les autres sous-systèmes qui interagissaient avec le module en question pour voir comment ils se comportent (à condition évidemment que vous ayez bien lu le rapport d'anomalies)
- Souvent, les données sont partagées entre différents sous-systèmes applicatifs. Le testeur qui sait que les données sont partagées, peut exploiter cette connaissance au profit de nouveaux parcours d'utilisateurs. (une technique très répandue dans le domaine des tests de sécurité).

Connaître les utilisateurs finaux

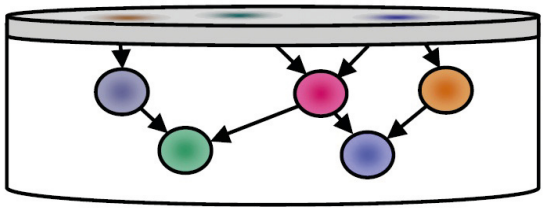
Dans le langage des testeurs on voit souvent le terme utilisateur (users) : l'utilisateur s'authentifie, l'utilisateur clique ici, l'utilisateur fait un RollOver sur ce menu, …. Pourtant, est-ce que le testeur connaît suffisamment bien son utilisateur? Rappelons aussi que l'utilisateur est aussi la Victime que nous avons évoquée un peu plus haut. Brian Marick va plus loin en affirmant *« Un bon testeur comprend comment les utilisateurs passent leurs journées »*.

Illustrons nos propos par deux exemples :

Un ingénieur est en charge de tester une application iPhone. La majorité de ses tests se porteront sur : *« je charge l'application, et je teste étape par étape tout ce que l'application est censée faire en se basant sur les spécifications »*. Est-ce que ce testeur aura le réflexe, en même temps qu'il est en train de cliquer partout sur son application, de passer (ou de recevoir) un coup de fil ou un SMS ? Alors que ce sont des situations réelles auxquelles un utilisateur final risque être confronté une fois que l'application est sur l'Apple Store.

Prenons un autre exemple, celui d'une application e-commerce avec un tunnel d'achat. Les testeurs professionnels développeront pour cela des scénarios de tests pour valider de bout en bout, tout le tunnel d'achat en parcourant tous les chemins possibles et imaginables. Or, un autre utilisateur, sur le même site e-commerce, va peut-être ouvrir plusieurs navigateurs, sélectionner plusieurs produits, comparer les articles sélectionnés depuis différentes fenêtres de son navigateur avant de passer sa commande, et tout cela en discutant avec ses amis sur Facebook. Entre deux actions, un temps non négligeable peut s'écouler. Il faut se rendre à l'évidence : un utilisateur a tendance à mener plusieurs actions simultanées. Regardez juste le nombre de sessions permanentes ouvertes du matin au soir sur votre poste de travail. Un utilisateur n'attendra pas la fin d'un programme quand ce dernier ne répond pas, il lancera une autre tâche, puis une autre, …

Fig.4



Comme vous pouvez le constater, et contrairement aux tests traditionnels «scénarisés » qui sont basés sur la notion (OK, KO), le Serendipity Testing porte sur le « feedback ». En adoptant une telle méthodologie, en plus de découvrir de nouvelles anomalies, nous obtiendrons un « feedback » utile, efficace et permanent.

En conclusion, cet article même est le résultat de la sérendipité car initialement mon sujet portait sur les tests exploratoires. C'est en lisant un article sur la manière dont certains acteurs du e-commerce implémentent les principes de la sérendipité pour améliorer leur taux de conversion/loyauté, que j'ai eu l'idée de faire le parallèle entre ce concept et celui des tests logiciels. Il est évident que d'autres études sont nécessaires pour mieux contextualiser cette approche, fournir une stratégie de mise en place d'une démarche de Serendipity Testing et voir de quelle manière elle complète les autres techniques et méthodologies de tests logiciels.

■ Elalami **Lafkih**
Consultant Valtech

Bibliographie

Jessica George, *"Socratic Inquiry and the Pedagogy of Reference: Serendipity in Information Seeking"*
Jennifer E. Nutefall and **Phyllis Mentzell Ryder**, *"The serendipitous research process"*
James Bach, *"Exploratory Testing Explained"*
Cem Kaner, *"A Tutorial in Exploratory Testing"*, April 2008
David F Horrobin, *"Quirks of fate?"*, British Medical Journal, 2 December 1978
Brian Marick, *"The Tester's Triad: Bug, Product, User"*
http://fr.wikipedia.org/wiki/Sérendipité

Tests de performance : préparez votre site web pour LE grand jour !

Le lancement officiel d'une application grand public est un moment extrêmement important : c'est là où une grande majorité des utilisateurs vont accéder au site pour la première fois, déterminer si le site est agréable et utile, et surtout, décider ou non d'y revenir.

Malheureusement, il arrive que des sites soient victimes de leur succès. Une affluence particulièrement forte la première journée peut mettre à genoux toute l'infrastructure et couper l'accès à tous les utilisateurs d'un seul coup. Les derniers exemples de europeana.eu en 2008 ou france.fr en 2010 sont des cas d'école. Nous allons voir dans la suite de cet article l'utilisation de l'application JMeter pour effectuer des tests de performance sur une application web, et les bonnes pratiques pour la tester efficacement.

Apache JMeter

JMeter est une application développée par la fondation Apache, entièrement écrite en Java, dédiée aux tests de charges et de performances et permettant d'effectuer des relevés de mesures et de tracer des courbes à partir des résultats de celles-ci.

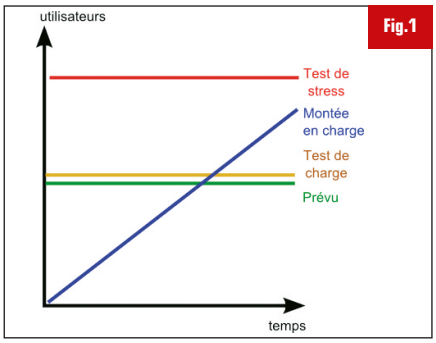


Fig.1

JMeter permet de simuler des utilisateurs qui vont travailler simultanément, en exécutant parallèlement un scénario de test défini par le développeur. Les actions que ces "utilisateurs" vont effectuer peuvent être simples (accès à une URL définie) ou complexes (extraire des données d'une page pour remplir un formulaire).
Mais avant de rentrer dans le détail de son fonctionnement, voyons les différents tests que l'on peut faire avec ce genre d'outil.

Les différents tests de performance

Un test de performance est basiquement la simulation d'un grand nombre d'utilisateurs effectuant des requêtes en même temps sur un site web, mais on peut tout de même distinguer plusieurs types de tests de performance : [Fig.1].

- Les tests de charge : Il s'agit de tests où l'on simule une quantité d'utilisateurs équivalente à celle attendue en production, sur une période plus ou moins longue. Ces tests permettent de savoir comment l'application se comporte dans le cas où tout se passe comme prévu et que le site a un succès "maîtrisé".
- Les tests de stress : Ces tests sont utilisés pour simuler une quantité largement supérieure d'utilisateurs que ce qui a pu être prévu, afin de voir le résultat sur l'application d'une forte affluence (comme le jour du lancement !).
- Les tests de montée en charge : dans cette situation, on augmente progressivement le nombre d'utilisateurs simulés pour déterminer le point critique au delà duquel l'application ne pourra plus répondre à tous les utilisateurs.

L'interface de JMeter

Les tests dans JMeter se configurent à travers une interface graphique. Cette interface se compose d'un panneau contenant deux arborescences (le plan de test, et le plan de travail), et d'un autre panneau affichant les paramètres de l'élément sélectionné dans une de ces arborescences [Fig.2]. Le plan de test contient le test tel qu'il va être exécuté. Le plan de travail va contenir les éléments que vous mettrez de côté (permettant de conserver des pans entiers de configuration sans avoir à les supprimer pour lancer les tests quand ils ne les utilisent pas). Les éléments de test s'ajoutent par le menu contextuel. Nous allons voir les composants principaux présents dans le plan de test : [Fig.3].

- **moteurs d'utilisateurs -> groupe d'unités :** C'est le composant principal d'un test de performance. Il permet de définir le nombre d'utilisateurs qui vont être simulés, le temps de chargement entre chaque utili-

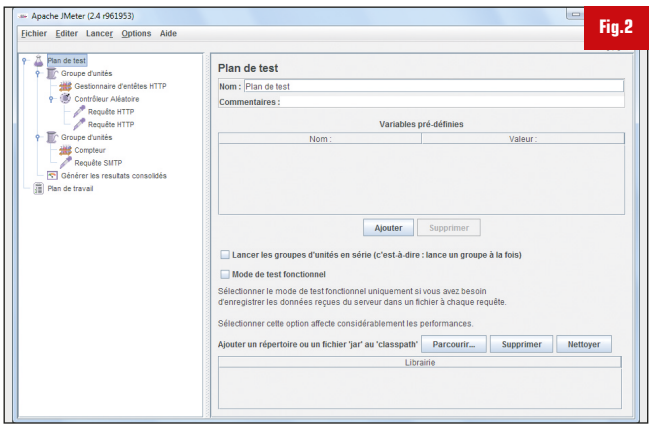


Fig.2

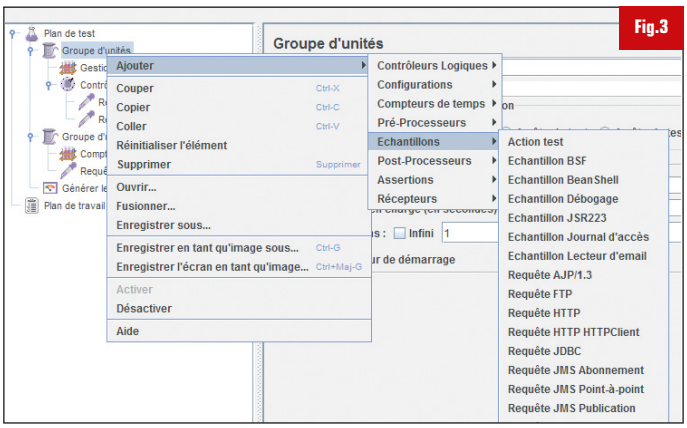


Fig.3

sateur (dans le cas d'une montée en charge progressive) et le nombre d'actions que les utilisateurs doivent effectuer. Les actions qu'ils vont effectuer sont définies dans la sous-arborescence de cet élément.
- **Echantillons -> Requête HTTP :** Action consistant à contacter une URL et récupérer le contenu renvoyé par le serveur.
- **Récepteurs -> Graphique des résultats :** Interface qui va présenter un graphique des temps de réponse des requêtes qui évoluent avec le test.

Exemple simple de test de performance

Dans l'exemple suivant, le test configuré va lancer un groupe d'utilisateurs pour parcourir un grand nombre de fois le site, le parcours étant défini comme l'appel de trois URL différentes. Le test est donc décrit comme un groupe d'unités (les utilisateurs simulés) contenant trois requêtes HTTP (Le parcours sur le site) et d'un graphique qui va permettre de voir l'évolution des temps de réponse du site [Fig.4].

Des tests plus complexes

JMeter permet des actions plus subtiles que l'accès direct à une URL. Il est par exemple possible de placer un "contrôleur aléatoire" qui va, au moment de lancer une action, choisir aléatoirement entre plusieurs URL, rendant les parcours des utilisateurs moins homogènes (et donc plus réalistes). Il est également possible de placer des temps de pause entre des actions, ou des "post-processeurs" qui vont analyser les pages renvoyées par le serveur et modifier la suite du test en conséquence.

Bonnes pratiques d'un test de performance

Un test de performance ne se prend pas à la légère, et il faut respecter quelques règles pour rendre l'opération efficace :
- **Testez de manière réaliste** : le test doit simuler de vrais utilisateurs, il faut donc faire en sorte que le test suive une certaines logique, et pas des accès aléatoires un peu partout sur votre site.
- **Testez ce qui est testable** : certains types de site web font l'essentiel de leur trafic autrement que par l'accès aux pages HTML (site exclusivement en Flash, site utilisant massivement AJAX). Il peut donc être plus utile de tester des services (Web-Services ou autres).
- **Testez à chaud** : le site peut réagir différemment si le serveur vient d'être démarré, ou si il a déjà derrière lui plusieurs heures de fonctionnement.
- **Testez les éléments sensibles** : Il y a principalement deux problèmes sur les sites web, les problèmes de calculs (si le serveur n'arrive pas à générer les pages assez vite) et les problèmes de bande passante (s'il y a trop d'informations qui circulent sur le réseau). Il faut essayer d'identifier intelligemment les parties du site qui peuvent générer ce genre de problème.

Tests de charge distribués

Un test comme celui présent ci-dessus fonctionne très bien, mais une fois la charge élevée, se pose un nouveau problème : le test est limité par les capacités de la machine qui lance les tests, et par son accès réseau [Fig.5].
Pour pallier ce problème, JMeter offre la possibilité d'exécuter des tests de manière distribuée, en coordonnant plusieurs machines pour effectuer les tests en même temps [Fig.6].
Pour effectuer un test distribué, il faut lancer un serveur JMeter sur toutes les machines qui devront exécuter le test, de déclarer les IP de ces différentes machines sur l'interface graphique du JMeter qui va piloter les tests (dans le fichier "bin/jmeter.properties"). Ainsi, il sera possible de lancer les tests sur tout ou partie du groupe de machines. Il est à noter que les machines ne se répartissent pas le test, mais l'exécutent toutes en même temps !

Des tests grandeur nature !

L'idéal pour un test réaliste est de pouvoir lancer des requêtes non depuis un seul endroit, mais depuis un ensemble d'endroits répartis un peu partout sur le territoire cible (la France, l'Europe ou le monde). Il est peu probable que vous disposiez personnellement d'un arsenal de machines aux quatre coins du globe, mais des grands groupes peuvent vous louer leurs machines : il s'agit des fournisseurs de Cloud Computing et plus précisément des fournisseurs de IaaS (Infrastructure as a Service - comme Amazon EC2), sur lequel vous pourrez vous-même déployer JMeter et effectuer vos tests distribués. Avec ce genre de tests, il n'y a pas que l'application qui est testée, mais également l'ensemble de l'infrastructure hébergeant votre site.

Conclusion

Grâce à JMeter, vous pourrez à moindre coût effectuer des tests de performance sur votre application Web, et vérifier qu'elle est prête pour LE grand jour ! Il faut toutefois être prudent lors de la description du test, car un test mal défini peut devenir complètement inutile (tester en charge l'accès à la page "conditions générales" d'un site web ne prouvera absolument rien)

■ Bertrand **Pisson**,
architecte java, Valtech
Blog : http://bertrand.pisson.eu

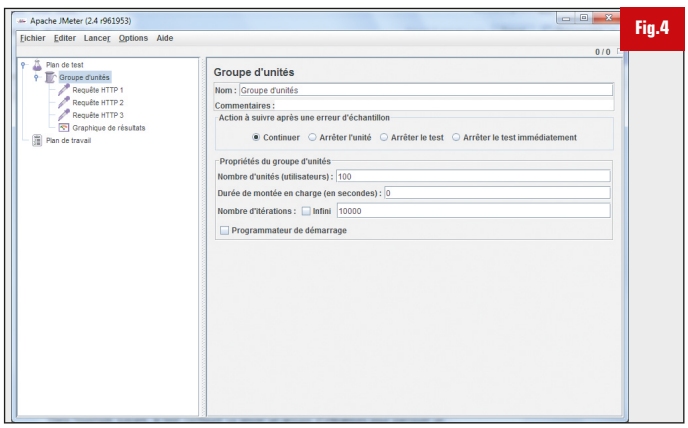


Fig.4

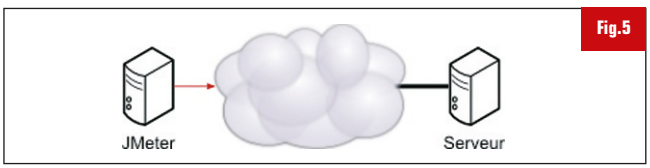


Fig.5

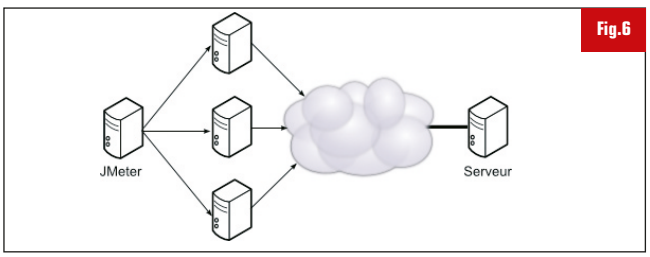


Fig.6

Comment tester le code PHP ?

Les tests en général font peur lors de la réalisation d'une application web. Ils sont souvent mis de côté pour réduire les coûts et les délais. Cependant, la vision à moyen et long terme démontre leur importance tant au niveau de l'industrialisation que de la fiabilité.

Les tests unitaires regroupent différents projets pour permettre aux développeurs de réaliser facilement ce type d'opération. Il existe des applications mais aussi des frameworks de test unitaires open source.

Démarrer avec les tests

Il n'est jamais évident de définir une date de lancement pour effectuer les tests. Cependant, si vous interrogez les différents experts PHP sur ce sujet, ils vous répondront *« le plutôt est le mieux »*, car plus vous attendez, plus vous risquez de perdre du temps.

Les points importants qui poussent les équipes à mettre ces tests en place concernent principalement la validation du code :

- Il est très complexe ou mal structuré pour repérer les erreurs et les problèmes
- Il ne peut pas être testé tout le temps en profondeur
- Il n'est pas parfait et, lors de l'ajout ou la modification d'une fonctionnalité, engendrer des régressions dans le projet.
- Il n'est pas ou difficilement réutilisable

Cependant les tests peuvent s'appliquer quelle que soit votre façon de programmer, c'est-à-dire par l'intermédiaire fonctionnel ou orienté objet.

Les tests avec SimpleTest

L'utilisation de SimpleTest pour effectuer des tests, consiste à vous les proposer très rapidement et facilement. Ils sont utilisés de différentes manières et vont vous permettre de soulager le côté validation d'un projet.

Tester vos pages web

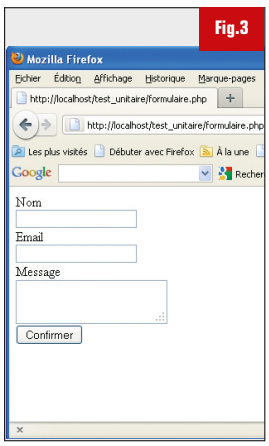
Le test des pages web vous sera utile si vous êtes amenés à vérifier l'existence d'une page qui est associée à un contenu, ou encore par l'intermédiaire d'alias. Un site peut très rapidement se retrouver avec des centaines de pages de contenu dynamique et si vous devez valider page par page après l'insertion d'une modification ou l'ajout d'une nouvelle fonctionnalité, vous allez passer de nombreuses heures de validation. Pour effectuer cette opération, vous pouvez utiliser un simple test à partir du listing1 :

Listing 1

```
<?php
require_once('simpletest/autorun.php');
require_once('simpletest/web_tester.php');
class SimpleFormTests extends WebTestCase
{
  function testPageExist()
  {
    $this->get('http://localhost/test_unitaire/formulaire.php');
     $this->assertResponse(200);
  }
}
?>
```

Si le test pose un problème quelconque, comme l'accessibilité, vous obtenez un message d'alerte, illustré par [Fig.1].

Si le test est concluant et donc une réussite, le résultat obtenu est représenté par [Fig.2].



Tester les formulaires

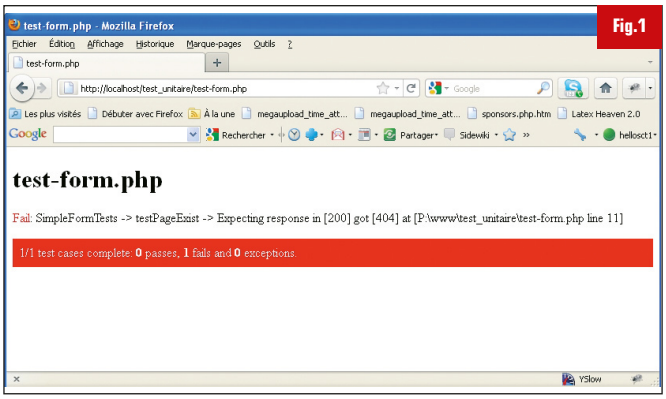
Lors de la réalisation de formulaires, les tests unitaires peuvent aussi être employés pour s'assurer du bon fonctionnement d'un formulaire de saisie. Il vous aideront, de façon automatisée, à la validation de celui-ci.

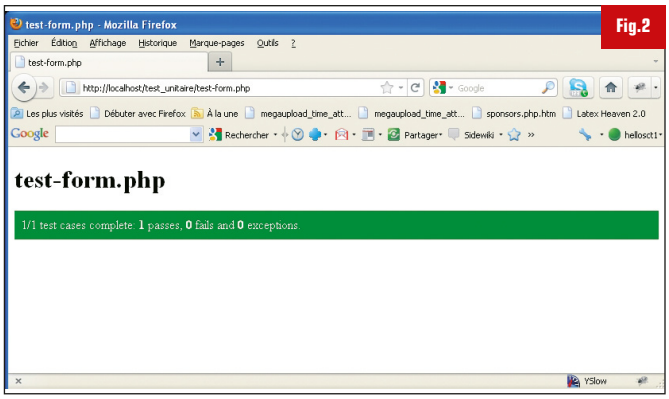
Par exemple, le formulaire représenté par [Fig.3], comprend 3 champs : Nom, email et une saisie libre.

Sa représentation est figurée par le listing 2 : « formulaire.php »

Listing 2 : formulaire.php

```
<html>
<body>
<form action=»formulaire.php» method=»post»>
```





```
Nom<br /><input type=»text» name=»lname» /><br />
Email <br /> <input type=»text» name=»name» /><br />
Message <br /> <textarea rows=»2» cols=»20»></textarea><br />
<input type=»submit» value=»Confirmer» />
</form>
</body>
</html>
```

Pour tester le formulaire, vous pouvez réutiliser l'exemple précédent comme ceci :

Listing 3 : test formulaire

```
<?php
require_once('simpletest/autorun.php');
require_once('simpletest/web_tester.php');

class SimpleFormTests extends WebTestCase
{
function testSubmissionSuccessful() {
    $this->get('http://localhost/test_unitaire/formulaire.php');
  $this->assertResponse(200);
   $this->setField("name", "programmez");
   $this->setField("email","votreemail@test.com" );
   $this->setField("message", "Message saisie");
   $this->clickSubmit("Envoyer");
   $this->assertResponse(200);
}
}
?>
```

Le but de ce test consiste à vous montrer que la page existe toujours, et que l'envoi du formulaire s'est bien déroulé. Par ailleurs, si vous modifiez les valeurs correspondant à une saisie différente, le résultat affiché pourra bien entendu être différent.

Tester les valeurs

L'autre point très utile, concerne la manipulation de valeurs. Celles-ci peuvent être des valeurs numériques, mais aussi des tests concernant différents cas d'utilisations, par exemple, la comparaison de 2 valeurs, comme le montre le listing 4.

Listing 4 : test de valeur

```
<?php
require_once('simpletest/autorun.php');
require_once('simpletest/web_tester.php');
class SimpleFormTests extends WebTestCase
{
 function testTableau()
 {
   $a = 1;
   $b = $a;
   $this->assertEqual($a, $b);
 }
}
?>
```

Cet exemple montre une comparaison de valeur entre 2 variables. Il vérifie si les 2 variables possèdent le même résultat. Si ce n'est pas le cas, un message d'alerte sera signalé, sinon le test sera validé. Il existe de nombreuses possibilités de définir si un résultat est bon ou pas, comme le montre le tableau suivant :

assertTrue($x)	Echoue si $x est faux
assertFalse($x)	Echoue si $x est vrai
assertNull($x)	Echoue si $x est initialisé
assertNotNull($x)	Echoue si $x n'est pas initialisé
assertIsA($x, $t)	Echoue si $x n'est pas de la classe ou du type $t
assertEqual($x, $y)	Echoue si $x == $y est faux
assertNotEqual($x, $y)	Echoue si $x == $y est vrai
assertIdentical($x, $y)	Echoue si $x === $y est faux
assertNotIdentical($x, $y)	Echoue si $x === $y est vrai
assertReference($x, $y)	Echoue sauf si $x et $y sont la même variable
assertCopy($x, $y)	Echoue si $x et $y sont la même variable
assertWantedPattern($p, $x)	Echoue sauf si l'expression rationnelle $p capture $x
assertNoUnwantedPattern($p, $x)	Echoue si l'expression rationnelle $p capture $x
assertNoErrors()	Echoue si une erreur PHP arrive
assertError($x)	Echoue si aucune erreur ou message incorrect de PHP n'arrive

Les tests avec PHP Unit

L'utilisation de PHP Unit est différente de la méthode précédente, mais offre aussi de nombreuses possibilités et des informations pour valider votre projet.
L'exemple montre l'utilisation de 2 tests. Le premier montre l'existence du tableau et est donc vide. L'autre test consiste à remplir ce même tableau en lui attribuant une valeur de 1.

```
<?php
require_once 'PHPUnit2/Framework/TestCase.php';
class ArrayTest extends PHPUnit2_Framework_TestCase
{
    public function testNewArrayIsEmpty()
    {
        $fixture = Array();
        $this->assertEquals(0, sizeof($fixture));
    }
    public function testArrayContainsAnElement()
    {
        $fixture = Array();
        $fixture[] = 'Element';
        $this->assertEquals(1, sizeof($fixture));
    }
}
?>
```

Conclusion

Quel que soit le framework de test unitaire que vous utiliserez, vous ne devez pas perdre de vue qu'il est important de définir des règles et de mettre en place des tests unitaires pour résoudre des problèmes d'analyse et de conception.
Par ailleurs, ces tests prennent encore plus d'importance lorsque vous possédez de nombreux écrans de saisie. Ainsi le résultat que vous obtenez sera de qualité.

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

Gérer sa dette technique avec SQALE dans Sonar

La plate-forme Sonar de suivi qualimétrique du code source propose désormais une implémentation de la méthode SQALE. Cette méthode permet une évaluation objective et quantitative de la qualité du code d'une application. Les indicateurs calculés et présentés par Sonar sont simples et proposent à la fois une vue haut niveau au décideur et des outils très opérationnels aux équipes de réalisation.

Le concept de dette technique est une métaphore qui a été utilisée pour la première fois en 1992 par Ward Cunningham (http://en.wikipedia.org/wiki/Ward_Cunningham). Il est aujourd'hui très utilisé dans la gestion des projets par les équipes agiles. Il permet de comparer les concessions architecturales acceptées lors d'un sprint de développement à une dette contractée sur le projet, qui aura permis de gagner du temps mais qui devra être remboursée à un moment donné.

Pour résumer le concept, on peut reprendre les propos de Ward Cunningham :

- Négliger la conception, c'est comme emprunter de l'argent
- Refactorer, c'est comme rembourser la dette principale
- Développer moins rapidement à cause de cette dette, c'est comme payer des intérêts sur l'emprunt.

Cette analogie financière rencontre un vif succès tant au sein des équipes afin d'expliquer le travail de refactoring qu'il faut financer et effectuer avant la livraison finale, qu'au sein du management car cela lui permet d'obtenir une vision nette du « Reste à faire projet », décomposé en un « Reste à Faire » fonctionnel qui sera couvert dans les sprints à venir et un « Reste à Faire » technique (la dette technique) qu'il faudra budgétiser et rembourser avant la clôture du projet.

De par sa puissance et sa facilité d'appréhension, ce concept ainsi que le mode de management associé s'étend au delà de la communauté agile. Que cette dette soit exprimée en dollars, en heures ou en unités d'œuvre, ces chiffres sont faciles à manipuler et permettent :

- De fournir un langage commun à toute l'équipe
- De suivre des tendances
- D'établir des tableaux de bord comparatifs
- De mettre en place une gestion proactive du patrimoine applicatif.

En connaissant la dette technique d'une application on peut planifier des opérations de maintenance préventive qui permettront de rendre celle-ci plus testable, plus évolutive donc moins coûteuse à maintenir dans le temps.

A la recherche d'un standard objectif

Il existe depuis longtemps des standards, tels que l'ISO 9126 ou l'ISO/IEC 15939 dont l'objectif est d'aider à évaluer la qualité du code source. Malheureusement ces standards sont incomplets et ne proposent pas de méthode concrète et fiable pour donner un score au code d'une application.

C'est pour répondre à ce besoin que la méthode SQALE (Software Quality Assessment based on Life cycle Expectations) a été conçue. Cette méthode puise sa source dans quelques principes simples comme le fait par exemple qu'avant d'évaluer la qualité d'une application il est nécessaire de préciser le référentiel qualité qui sera utilisé.

La méthode SQALE a été entièrement définie dans le respect des principes fondamentaux de la théorie de la mesure. Un des principes les plus importants est la clause de représentativité qui stipule que si, dans la réalité, une entité A est perçue strictement plus « grande » qu'une entité B, alors la mesure de « grandeur » associée à A doit être strictement supérieure à celle de B. Dans le cas présent cela induit que les mesures associées à A et B ne peuvent pas être égales. Sinon, cela compresserait l'information et ne serait plus une bonne représentation de la réalité.

Ce principe, qui parait évident, a par le passé été souvent ignoré par les méthodes d'évaluation de la qualité et les systèmes d'agrégation mis en œuvre par les outils d'audit de code. Chaque fois qu'un système de mesure de code viole la clause de représentativité, il génère des « faux positifs » qui décrédibilisent l'ensemble des mesures produites et agissent comme un frein à la généralisation de la mesure de la qualité du code au sein des entreprises et des projets.

SQALE, une méthode de qualimétrie exploitable.

SQALE (http://www.sqale.org) évite ces écueils et fournit des résultats précis, comparables et exploitables sur lesquels il est possible de s'appuyer pour prendre des décisions : évaluation de charge, comparaison de prestataires, planification d'un programme d'amélioration de la qualité du code.

Un modèle qualité

Le modèle qualité SQALE sert à formuler et à organiser les exigences non fonctionnelles relatives à la qualité du code. Il est organisé en 3 niveaux hiérarchisés. Le premier niveau est composé de caractéristiques, le deuxième niveau de sous-caractéristiques et le troisième niveau est composé d'exigences qui portent sur des attributs internes au code source. Ces exigences peuvent dépendre du contexte du logiciel et de son langage. On y trouvera des exigences concernant la complexité ou le couplage, ou bien encore des exigences concernant la bonne utilisation du langage ou sur la présentation du code… Le modèle qualité générique de

SQALE positionne chronologiquement les attentes (exigences) retenues en fonction de leur impact sur le cycle de vie du code, [Fig.1 et 2].

Puisque le modèle qualité SQALE est un modèle d'exigences (par opposition à ceux couramment utilisés qui sont des modèles d'observation), son contenu ne contient que des « must-have » caractérisant sans controverse ce qui sera considéré comme un code source de qualité au sein de l'entreprise. Il s'agit de définir des seuils de conformité sur certaines métriques, ainsi que des règles de codage reconnues pour leur bien fondé et n'acceptant pas d'exception.

Un modèle d'analyse

Il contient d'une part les règles qui servent à normaliser les mesures et contrôles relatifs au code, d'autre part les règles pour agréger les valeurs normalisées en indices. SQALE normalise les constats issus des outils d'analyse de code source en les transformant en indices représentant des charges de remédiation. Pour ce faire la méthode utilise, soit un facteur, soit une fonction de remédiation. L'agrégation des indices se fait par addition. Que ce soit dans l'arborescence du modèle qualité, ou dans l'arborescence de la hiérarchie des artéfacts du code source.

On retrouve bien une généralisation du concept de dette technique, mais au lieu de limiter le calcul à la charge induite par les violations des bonnes pratiques d'architecture, on étend ce calcul à des règles relatives à la fiabilité, à la présentation et à la compréhensibilité. On comptabilise ainsi, aussi bien la dette volontaire que la dette involontaire.

Pour une application donnée (ou un composant logiciel), on peut calculer une charge de remédiation pour l'ensemble des exigences qualité. Cette valeur s'appelle l'index SQI (SQALE Quality Index). Si on ne s'intéresse qu'à une caractéristique de la qualité du code et uniquement aux exigences qui lui sont associées, on obtient des index de testabilité, de fiabilité, de maintenabilité… (STI, SRI, SMI…).

Pour comparer des composants ou des applications différentes, on ramène ces dettes qualité à la taille de l'application, exprimée par exemple en KSLOC (milliers de ligne d'instruction). On obtient ainsi des densités d'index. C'est cette densité qui permet donner un score à l'application (le meilleur score étant A et le plus mauvais

Fig.1

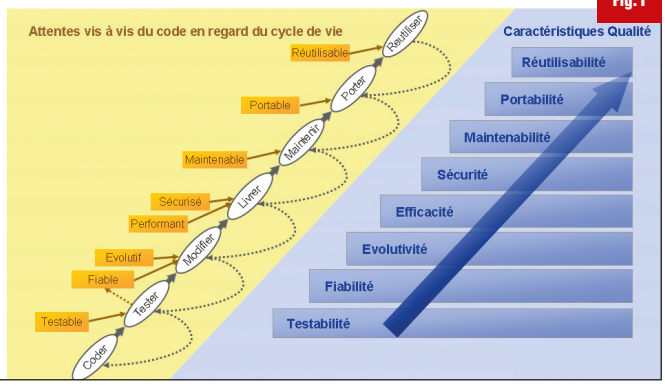


Fig.2

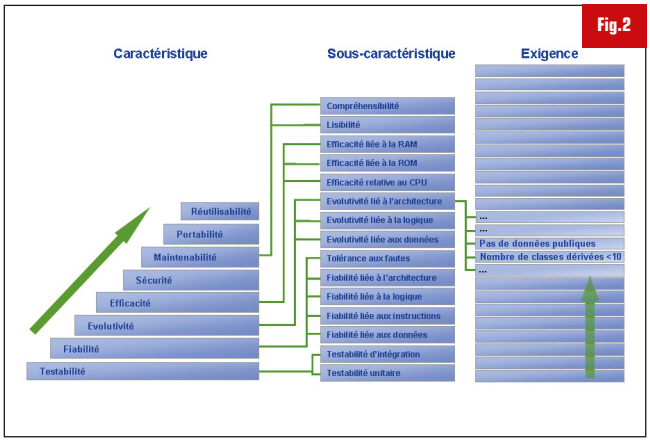


Fig.3

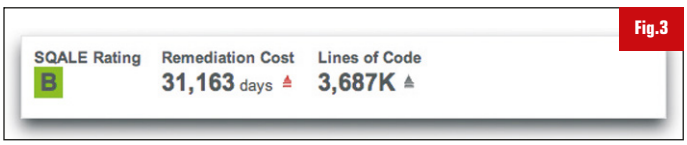


Fig.4

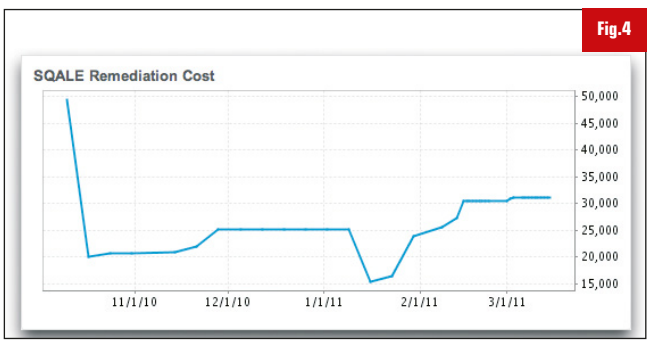


Fig.5

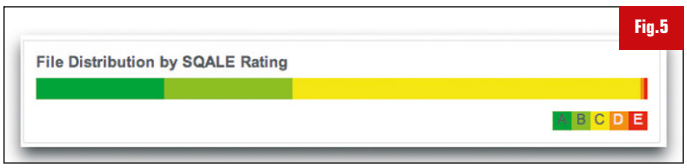


Fig.6

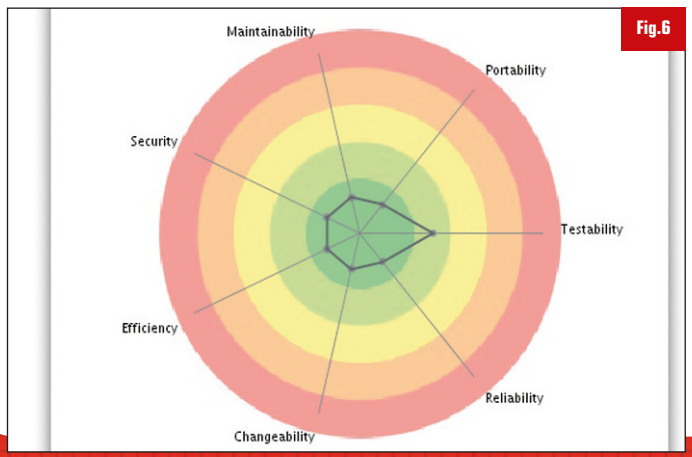
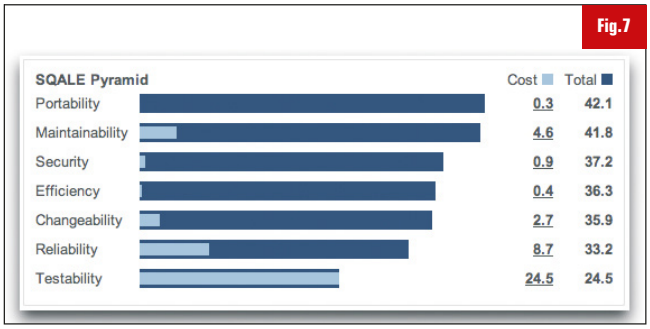


Fig.7



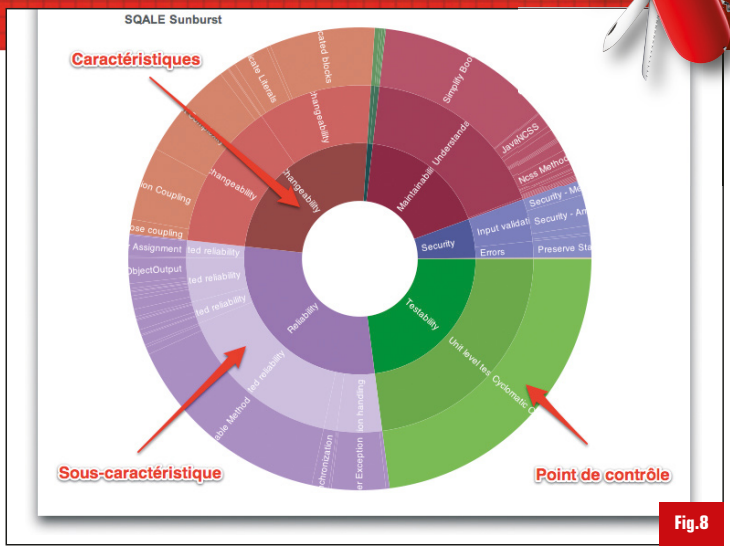


Fig.8

Fig.9

E). Les meilleurs codes ayant des densités très faibles.

L'implémentation de SQALE dans SONAR

SonarSource (http://www.sonarsource.com), distribue une implémentation de la méthode SQALE. Elle fournit tous les indicateurs SQALE sous forme de « widgets » qui viennent s'insérer dans les tableaux de bord standard à tous les niveaux du système d'information, du fichier jusqu'à l'agrégation de l'ensemble des applications. Ces indicateurs respectent l'esprit global de l'outil qui veut que lorsqu'une mesure est présentée il soit possible d'avoir un effet zoom pour comprendre sa composition.

Tout d'abord une vue globale permettant de connaître le score du composant, sa taille et le montant de la dette technique [Fig.3]. Une composante fondamentale de la gestion de la dette technique étant la dynamique de remboursement, Sonar propose un historique du coût de remédiation [Fig.4]. Puis une distribution de la densité d'index par fichier pour comprendre si la dette est bien répartie ou bien se concentre dans quelques composants : [Fig.5].

Ainsi que la densité d'index par caractéristique : [Fig.6].

La pyramide SQALE pour comprendre à quel niveau de maturité appartient la dette technique : [Fig.7].

Enfin, Sonar fournit un « sunburst » qui est une vue heuristique de la répartition de la dette, de la caractéristique à l'exigence [Fig.8]. Tous ces widgets permettent de former un tableau de bord modulable : [Fig.9].

Conclusion

Tous les ingrédients sont en place pour que le mouvement de démocratisation autour de la gestion qualimétrie du code source s'accélère : du côté des concepts, SQALE est une méthode simple d'accès qui fournit un langage commun au sein des équipes pour discuter de la dette technique. Du côté de l'outillage, l'implémentation qui est faite de SQALE au sein de Sonar est tout à fait abordable et très opérationnelle pour les équipes de réalisation.

■ Jean-Louis **Letouzey**
Expert Consultant chez DNV IT GS
Membre du CISQ
Auteur de la méthode SQALE

■ Olivier **Gaudin**
Fondateur et Directeur de la société SonarSource

Disparition d'Adam Kolawa

Adam Kolawa n'était pas un nom inconnu pour nos lecteurs. Il avait signé plusieurs articles et livres blancs dans le magazine et sur programmez.com. Nous avons appris sa brutale disparition fin avril. Adam était dirigeant et co-fondateur de Parasoft, éditeur d'outils de tests et de gestion du cycle de vie du développement. Avec 4 amis de Caltech University, il fonda Parasoft en 1987, spécialisé dans les outils de tests et l'optimisation pour java, C++, .Net, les architectures de services, etc. Même si Parasoft était peu connu en France, la société s'est imposée, sur un marché assez concurrentiel.

Adam Kolawa était l'auteur de plus d'une centaine d'articles dans la presse technique, mais aussi dans Forbes ou le Wall Street Journal, ainsi que de plusieurs livres *. Il était détenteur de plus de 20 brevets en technologie logicielle. eWeek le faisait figurer parmi les 100 « personnes les plus influentes de l'IT », et Ernst&Young l'avait élu en 2001 « Entrepreneur de l'année » (catégorie Software). Il travaillait beaucoup sur l'optimisation des processus de développement, la productivité ou encore sur les nouveaux marchés de l'informatique.

Adam Kolawa avait installé des filiales de Parasoft sur tous les continents. Cet homme brillant s'était donné la mission de faire partager l'excellence logicielle. Nous le considérions comme un ami et sommes particulièrement tristes de sa disparition prématurée.

■ Jean Kaminsky
et François Tonic

http://en.wikipedia.org/wiki/Adam_Kolawa
http://www.parasoft.com/jsp/products/article.jsp?articleId=2850

**Son dernier ouvrage souligne la nécessité pour le dirigeant d'entreprise de s'impliquer dans l'IT : « The Next Leap in Productivity: What Top Managers Really Need to Know About Information Technology » (Wiley, 2009) ». Il était aussi le co-auteur de : « Automated Defect Prevention: Best Practices in Software Management » (Wiley-IEEE, 2007) et de « Bulletproofing Web Applications » (Wiley, 2001)*

Tests unitaires en Java

Composante essentielle du développement logiciel, les tests unitaires demeurent l'un des outils les plus puissants pour améliorer la qualité du code d'une application en assurant sa fiabilité au cours du temps. Figure de proue des frameworks de tests de la famille xUnit, JUnit demeure la référence en Java pour mettre en place des tests unitaires automatisés au sein des applications d'entreprise depuis près de 10 ans maintenant. Cet article va permettre de présenter les concepts essentiels de JUnit et de donner, au travers d'un exemple, les clés pour mettre en place des tests unitaires efficaces en Java.

Créateur de la méthodologie XP, Kent Beck a mis au point les frameworks de tests de type xUnit et écrit la première implémentation en Java d'un tel framework avec Erich Gamma. Nommé JUnit, ce framework open source et gratuit est désormais la référence au sein du monde Java pour la réalisation de tests unitaires automatisés. Basé sur une philosophie assez simple, JUnit offre tout ce qu'un framework de tests unitaires moderne se doit d'offrir :

- Le résultat d'un test unitaire est binaire : soit le test passe, soit le test ne passe pas. Ceci évitant une validation humaine a posteriori sur ce résultat.
- Les tests sont présentés sous la forme de cas de tests regroupables ensuite sous la forme de suites de cas de tests permettant une hiérarchisation entre les différents tests unitaires d'une application.
- Un rapport de tests dans un format structuré vient présenter les différents résultats de l'exécution des tests unitaires JUnit. Ceci permettant à des outils tiers de venir s'interfacer avec JUnit.

Actuellement en version 4.8.2, le projet reste très actif et ne cesse de proposer de nouvelles fonctionnalités à chaque version. Profitant des avancées de Java 5 avec le support des annotations, la version 4 de JUnit permet de définir des tests uniquement via des annotations sur des méthodes Java. Ceci apportant au développeur plus de souplesse en l'affranchissant des contraintes d'héritage imposées par la version 3. Un test unitaire JUnit comporte 3 principales phases :

1. Mise en place d'un contexte commun à l'ensemble des tests d'un cas de test au sein d'une méthode annotée par @Before.
2. Cas de test à proprement parler, décrit au sein d'une méthode annotée par @Test.
3. Nettoyage du contexte commun à l'ensemble des tests au sein d'une méthode annotée par @After.

Hormis ces annotations essentielles, de nombreuses autres existent comme on peut le voir dans le tableau de la [Fig.1].

Chaque test unitaire doit être indépendant des autres et il ne faut pas présumer de l'ordre d'exécution du runner JUnit pour les tests d'un cas de test. La validité ou non d'un test unitaire est déterminée via l'utilisation d'assertions spécifiques JUnit. Originellement assez simples et basées sur des méthodes statiques de la classe Assert, elles permettent de tester notamment l'égalité entre objets via assertEquals, la nullité ou non via assertNull ou assertNotNull ou bien la différence entre objets via assertNotSame. Avec JUnit 4.4 sont arrivées les assertions basées sur des contrats utilisables via la méthode assertThat offrant aux développeurs la possibilité de réaliser des assertions beaucoup plus puissantes. Notre premier cas de test va tester différentes méthodes de la classe Stack du JDK 6 d'Oracle. Voici le code d'un petit cas de test pour cette dernière :

Fig.1

Annotation	Description
@RunWith	Permet de préciser à JUnit de lancer les tests d'une classe avec le runner spécifique passé en paramètre et non avec le runner par défaut.
@SuiteClasses	Crée une suite de test. Le paramètre value permet de lister les différents cas de tests contenus dans la suite. S'utilise avec le runner Suite de JUnit.
@Test	Définit une méthode d'une classe comme test unitaire JUnit
@BeforeClass / @AfterClass	Positionnées sur des méthodes statiques. Exécutées respectivement avant et après l'ensemble des tests d'un cas de test. Permet de mettre en place un contexte commun à l'ensemble des tests et de faire un nettoyage de contexte.
@Before / @After	Positionnées sur des méthodes de classe. Exécutées respectivement avant et après chaque test du cas de test concerné. Permet de mettre là encore un contexte commun à chaque test.
@Parameters	Utilisée sur une méthode statique, permet de définir une source de données pour des tests paramétrés. Chaque jeu de données défini sera exécuté pour le cas de test concerné. S'utilise avec le runner Parameterized de JUnit.
@Rule	Définit une sorte d'intercepteur qui sera exécuté avant et après chacun des tests d'un cas de test. Se rapproche des annotations @Before / @After mais en pouvant définir un traitement commun à plusieurs classes sans passer par une classe mère commune.
@Ignore	Désactive un test. Utile si celui-ci n'est pas encore au point et qu'on ne veut pas que son échec viennet faire échouer le résultat de l'ensemble des autres tests.

Principales annotations de JUnit

```
public class StackTest {
   private static Stack<Integer> stack;
   private static final int NUMBER = 5;

   @BeforeClass
   public static void setUp() {
      stack = new Stack<Integer>();
   }

   @AfterClass
   public static void tearDown() {
      stack = null;
   }
```

```
    @Before
    public void setUpTest() {
        stack.clear();
    }

    @Test
    public void pushTest() {
        stack.push(NUMBER);
        assertEquals(new Integer(NUMBER), stack.peek());
    }

    @Test
    public void popTest() {
        stack.push(NUMBER);
        assertEquals(new Integer(NUMBER),stack.pop());
        assertTrue(stack.isEmpty());
    }

    @Test
    public void isEmptyTest() {
        // Rajout d'un push ici afin de faire échouer le test pour
voir dans la figure 2
        // un test en échec
        stack.push(NUMBER);
       assertTrue("Test de la méthode isEmpty", stack.isEmpty());
    }
}
```

Ce cas de test très simple se compose de 3 tests qui vont respectivement tester les méthodes push, pop et isEmpty de la classe Stack. On remarque de plus que chacun d'entre eux est bien isolé des autres puisque la méthode setUpTest annotée avec @Before se charge de vider la pile entre chaque exécution de test.

Incontournable, JUnit l'est avant tout car il est parfaitement intégré au sein des principaux IDE du marché. Ainsi, pour exécuter ce cas de test au sein d'Eclipse, il n'y a rien de spécial à faire si ce n'est à ajouter la bibliothèque JUnit au projet contenant notre cas de test. Le résultat de l'exécution du cas de test est présenté à la figure 2 et permet de voir que les tests qui sont passés sont identifiés en vert et ceux qui ont échoué le sont en bleu en cas d'assertions non validées et en rouge en cas d'erreurs non prévues (levée d'une exception inattendue par exemple). Enfin, on remarque que le message précisé au sein de l'assertion apparaît sur le test qui n'est pas validé [**Fig.2**].

Pour couvrir plus de cas de tests, il serait intéressant de pouvoir jouer avec le contenu de la pile. Pour cela, il faudrait avoir différentes configurations de jeux de données que l'on va insérer au sein de la pile et ensuite jouer les tests avec chacun de ces jeux de données. La fonctionnalité de tests paramétrés proposée par JUnit répond clairement à ce besoin et permet de faire cela de manière rapide en utilisant le runner spécifique Parameterized et l'annotation @Parameters sur une méthode qui servira de source pour ce jeu de données. Nous pouvons réaliser un deuxième cas de test qui aura l'allure suivante :

```
@RunWith(Parameterized.class)
public class StackTest2 {
    private static Stack<Integer> stack;
    private List<Integer> values = new ArrayList<Integer>();
   private List<Integer> nbToPush = new ArrayList<Integer>();
    private int size;
    private int pop;
    private int eltToContain;
    private boolean contains;

    @Parameters
    public static Collection<Object[][]> data() {
        return Arrays.asList(new Object[][][]{
{ {1,6,7,8,12,2}, {6}, {2}, {1, 2, 10}, {12}, {true} },
{ {}, {0}, {0}, {}, {45}, {false} },
 { {1}, {1}, {1}, {15, 16}, {1}, {true} }
        });
    }

    public TestStack2(Object[] values, Object[] size,
 Object[] pop, Object[] pushes, Object[] eltToContain, Object[]
 contain) {
        this.size = (Integer) size[0];
        this.pop = (Integer) pop[0];
        this.eltToContain = (Integer) eltToContain[0];
        this.contains = (Boolean) contain[0];
        // on récupère les valeurs
      for (Object value : values) { this.values.add((Integer)
value); }
        // on récupère les valeurs à ajouter
      for(Object push : pushes) { this.nbToPush.add((Integer)
push); }
    }

    @BeforeClass
    public static void setUp() {
        stack = new Stack<Integer>();
    }

    @AfterClass
    public static void tearDown() {
        stack = null;
    }

    @Before
    public void setUpBeforeTest(){
        // ajout des valeurs à la stack
        stack.addAll(values);
    }

    @After
```

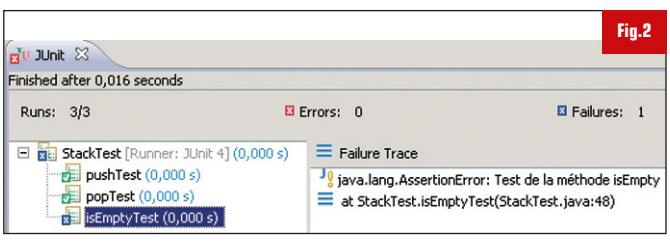


Fig.2

Résultat de l'exécution du cas de test StackTest

```
    public void tearDownAfterTest() {
        stack.clear();
    }

    @Test
    public void pushTest() {
        for(Integer v : nbToPush) {
            stack.push(v);
           assertThat("Push not correct", stack.peek(), equal
To(v));
        }
    }

    @Test
    public void sizeTest() {
      assertThat("Result size not correct", stack.size(), equal
To(size));
    }

    @Test
    public void containsTest() {
        assertThat("Result contain test not correct", stack.
contains(eltToContain), is(contains));
    }

    @Test
    public void popTest() {
        assumeTrue(!stack.isEmpty());
        assertThat("Value on pop not correct", stack.pop(),
equalTo(pop));
    }

    @Test
    public void isEmptyTest() {
        // on vide la pile
        for(int i = 0; i < size - 1; i++) {
            stack.pop();
        }
        assertTrue(stack.isEmpty());
    }

    @Test(expected = EmptyStackException.class)
    public void popTestException() {
        for(int i = 0; i < size - 1; i++) {
            stack.pop();
        }
        stack.pop();
    }
}
```

Ce cas de test va être exécuté à 3 reprises par le runner Parameterized de JUnit, ce qui correspond au nombre de jeux de données renvoyés par la méthode annotée par @Parameters. Ainsi, le constructeur du test sera appelé avec chacun d'entre eux donnant lieu à chaque fois à une exécution du test avec un jeu de données spécifique. Petit bémol toutefois, l'obligation de renvoyer une collection de tableau d'objets non typés ne rend pas le code très lisible. D'autre part, le test popTestException met en avant l'utilisation du paramètre expected de l'annotation @Test spécifiant que le test doit lever une exception précise pour être validé. Enfin, le test popTest démontre l'utilisation d'un autre concept intéressant de JUnit que sont les suppositions. Représentées par la classe Assume et les méthodes statiques de type assumeXXX, elles précisent à JUnit les pré-conditions à remplir pour que le résultat d'un test soit considéré dans le rapport final d'exécution. Ainsi, dans ce cas-là si la pile est vide le test sera bien exécuté et va lancer une exception de type EmptyStackException mais le test ne sera pas considéré en erreur puisque la pré-condition n'était pas vérifiée.
Maintenant que nous avons 2 cas de tests pour notre pile nous allons pouvoir créer une suite de tests grâce à l'utilisation du runner Suite de JUnit et à l'annotation @SuiteClasses :

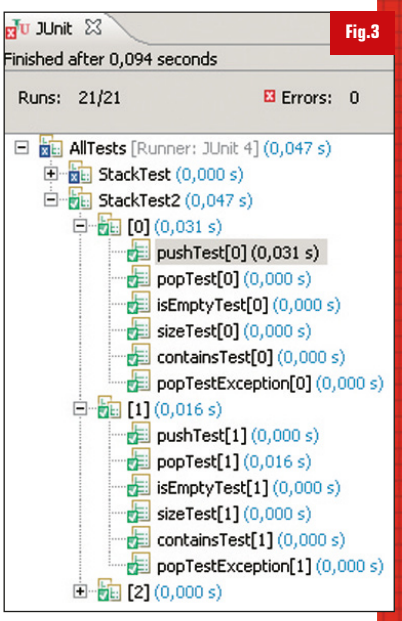
```
@RunWith(Suite.class)
@SuiteClasses(value = {
StackTest.class,
 StackTest2.class,
})
public class AllTests {
}
```

La création de la suite de tests se révèle très simple avec les annotations. Le rapport d'exécution obtenu suite au lancement de la suite de tests dans Eclipse est présenté à la [Fig.3].
Le rapport d'exécution de la suite de tests détaille clairement le découpage entre chacun des cas de tests qui la composent. En outre, on constate que le cas de tests StackTest2 a bien été exécuté à 3 reprises, ce qui correspond aux 3 jeux de données que l'on avait défini pour la configuration de ce test paramétré.

Conclusion

Cet article nous aura permis de faire un tour d'horizon des principales notions à connaître afin de mettre en place des tests unitaires automatisés en Java avec JUnit. Concurrencé par divers projets parmi lesquels on citera TestNG, il demeure malgré tout la référence dans le monde Java. Son évolution constante n'y est pas étrangère puisque les développeurs derrière le projet ne cessent de lui adjoindre de nouvelles fonctionnalités au fur et à mesure des releases. Enfin, son intégration parfaite aux différents outils du monde Java que ce soit sur la partie IDE ou bien build (Ant, Maven) fait de JUnit un framework de tests unitaires particulièrement adapté à l'intégration continue. Et c'est bien là que les tests unitaires automatisés prennent toute leur puissance !

Fig.3



Résultat de l'exécution de la suite de tests.

■ Sylvain **Saurel** – *Ingénieur d'Etudes Java / JEE*
sylvain.saurel@gmail.com

Faire carrière dans l'open source : le bon moment !

Pénurie, marché tendu, recherche de profils très spécialisés, salaires en hausse… tout et un peu n'importe quoi circule sur le marché de l'emploi informatique, car s'il existe effectivement un déficit de certaines compétences, la situation est loin d'être bonne pour de nombreux informaticiens (au sens large) qui cherchent toujours un travail.

On parle souvent de la difficulté à trouver de bons profils Sharepoint, experts Java ou base de données, etc. Il s'agit d'une réalité du marché. La même remarque peut se dupliquer pour l'open source et les logiciels libres. Même s'il faut nuancer l'analyse, car chaque, entreprise aura un besoin différent et les sociétés expertes en open source (SSII libres) peuvent avoir des visions divergentes du marché. Mais, un constat se dessine : les SSII libres embauchent, tout comme les SSII « classiques », voire les entreprises. Les compétences open source sont réellement recherchées.

« Les candidats n'ont pas trop de difficultés à trouver, qu'ils soient expérimentés ou un peu moins. Le marché est demandeur », résume Marc Hugon (directeur technique Sensio Labs). D'ailleurs la pression du marché est citée régulièrement par l'ensemble des acteurs de l'Open Source / Libre. Et cela se vérifie aussi bien chez les intégrateurs / SSII que chez les clients finaux. Smile ne dit pas autre chose : pénurie de certains profils, marché tendu. *« La tension est forte en Ile de France, le marché connaissant une croissance importante »* cadre Emmanuel Tormini (directeur de l'organisation et du recrutement Smile). C'est notamment le cas sur les profils eCommerce, Drupal. Et les grosses SSII « classiques » sont de réels concurrents sur le marché de l'emploi. La tension du marché, la concurrence influent directement sur le processus de recrutement. Chez Netapsys par exemple, sur les 40 recrutements prévus cette année, moins de 50 % sont effectifs. *« Nous sommes moyennement satisfaits. Il faut préparer les projets, la rentrée »,* analyse **Laetitia Aubras** (DRH de Netapsys). Et l'été n'est jamais une période propice à un recrutement soutenu !

Plutôt de jeunes diplômés que des seniors ?

Sur le profil, il est difficile d'établir un portrait robot du candidat idéal car les critères varient d'une SSII à une autre. Cependant, notre enquête a pu fournir quelques éléments de réponses :

- profil plutôt jeune, 25 – 35 ans, par exemple, à Sensio Labs, la moyenne tourne autour de 30 ans
- jeune diplômé, installé professionnellement avec une expérience, souvent bac + 5 ou pouvant justifier d'une expérience non négligeable
- passion et culture informatique, la participation à une communauté ouverte est appréciée
- La base salariale est très disparate, selon la société, l'expérience, etc. Au départ, le salaire annuel est compris dans une fourchette de 27-35 000 euros (moyenne que nous avons constaté). Cela peut atteindre 50 000 euros pour des profils « haut de gamme ». Chez Linagora, nous serions plutôt vers les 30 000 euros brut, voire, au-delà, selon les postes. Un profil architecte peut atteindre 50 000 euros.

> *« Les candidats n'ont pas trop de difficultés à trouver, qu'ils soient expérimentés ou un peu moins, le marché est demandeur. »*

Pour Marc Hugon, le salaire moyen est sur une tendance plate chez les SSII et autres éditeurs par contre, dans les startup, le salaire aura tendance à être plus élevé, les profils sont également différents. Chez Smile, le profil type est un candidat avec 2-3 d'expérience, et une bonne base technique. Côté recrutement, c'est environ un tiers de juniors, proportion que nous retrouvons fré-

Véronique Torner, *Alterway*

Reportage en ligne sur www.programmez.com

Quelques pistes d'embauches

Société	Nombre recrutement 2011	Commentaires
Alter-Way	Doublement des effectifs sur 3 ans	Développeurs, administrateurs…
Senseo Software	2 personnes par mois en moyenne depuis début 2011	développeurs
Uperto	Une dizaine d'offres open source (situation au début mai)	du développement à l'intégrateur CMS, chef de projet open source.
Linagora	Environ 70 en 2011	Paris et province
Smile	150 – 200	-
Netapsys	Environ 40	-

Avertissement : ces chiffres sont donnés à titre indicatif, ils peuvent changer à tout moment.

quemment dans les autres SSII du Libre.
Mais attention à ne pas retomber dans les travers des années 2000 où l'on voyait des profils de développeurs web surestimés, et un manque de maîtrise technique. *« Nous retombons dedans ! Des jeunes qui pensent être plus « bankable », surévaluent leurs compétences »*, avertit **Alexandre Zapolsky** (PDG de Linagora). Les risques apparaissent réels et peuvent peser sur le bon déroulement des projets et les échecs sont eux aussi réels. Ensuite, des SSII vont intervenir sur l'ensemble de la pile open source, donc en utilisant des profils très larges. Linagora décide au contraire d'être plus haut de gamme. *« Nous n'intervenons pas sur tout. Nos tarifs sont plus élevés mais la qualité se paie »*, poursuit Alexandre Zapolsky. Côté profil, Linagora ne déroge pourtant pas aux critères communs. *« Nous cherchons avant tout des profils techniquement bons, qu'ils soient juniors ou seniors »*. Mais il faut aussi que le nouveau venu ait de vraies qualités métier, sachant ce qu'est un projet, sa conduite, et qu'il ait aussi le sens du client et du service.

Le salaire : à négocier

Même si les principaux acteurs de l'open source français nous ont affirmé que le salaire n'était pas forcément le point central des candidats, il n'est pas oublié, loin de là, surtout quant le candidat possède une forte expérience et des compétences peu fréquentes. *« Il s'agit d'un sujet important, à cause du coût de la vie, même si ce n'est pas le premier point abordé par ceux qui débutent »*, analyse Alexandre. Nous rentrons dans le classique schéma de la négociation. Et les bases salariales varient souvent entre SSII, intégrateurs, même si durant notre enquête, le point de départ pour un débutant, jeune diplômé se situe aux alentours de 30 000 euros annuels (brut).

Des méthodes de recrutements diverses : formations internes et étranger

Le recrutement n'est pas une fin en soi. Si les entreprises sont prêtes à investir dans la formation et à proposer des parcours de carrière aux nouveaux arrivants, surtout si ceux-ci sont jeunes avec une expérience limi-

Pénurie et réseaux sociaux

Questions – réponses avec **Jacques Froissant**, *DRH de Valtech*

P ! : Dans le cadre des recrutements de profils open source, ressentez-vous une pénurie de candidats et y a-t-il une certaine tension concernant les salaires ?
JF : On assiste à une pénurie générale de développeurs. Le Syntec prévoit un delta de +10 000 créations de postes en 2011. L'APEC a vu à fin février une augmentation de +72 % du nombre d'offres dans le secteur informatique et + 272% pour les postes de développeurs !

P ! : Est-il vrai que les candidats ne sont pas stressés, parce que très demandés ?
JF : Complètement, ils sont sur-sollicités par les chasseurs de têtes et les recruteurs d'entreprises. La pression sur les salaires commence à être forte également.

P ! : Quelles en sont les conséquences sur vos recrutements, nombre de candidats, durée des processus ?
JF : Il est nécessaire de passer beaucoup plus de temps sur le sourcing de candidats pour un nombre beaucoup plus limité de retours. Chez Valtech nous avons développé une stratégie forte sur les réseaux sociaux, les circuits classiques (jobboard, bases de CV liées ne fonctionnant quasiment plus dans ce contexte). La présence de la marque employeur Valtech dans Linkedin, Viadeo, Twitter et Facebook permet aujourd'hui de générer près de 40% des candidats recrutés. De manière générale sur tous nos postes technos, les process se rallongent et il faut être extrêmement réactif dès que nous avons identifié un candidat potentiel.

P ! : Comment fidélisez-vous les profils opensource ? Comment évitez-vous le turnover ?
JF : Nous avons plusieurs secrets pour les garder mais on ne les donne pas ! Valtech fidélise par une formation permanente (assurée par notre filiale Valtech Training), des échanges techniques avec nos "gourous" technos sur notre intranet et lors d'événements conférences partage internes, une implication dans l'écosystème Open Source en poussant nos consultants à y intervenir, des soirées détentes, un Top Management à l'écoute … et les plus beaux bureaux de Paris ! Malgré tout cela il est clair que cela ne suffit pas toujours dans un marché pénurique où la surenchère est permanente.

P ! : Etes-vous confrontés au « débauchage sauvage » ?
JF : Oui, la pression sur nos équipes est forte. Valtech étant visible et réputé pour le niveau technique de ses équipes, nous faisons partie des cibles privilégiées. Mais l'évolution récente de Valtech vers un modèle où marketing, agilité, et technologie sont beaucoup plus liés donne des perspectives très intéressantes pour les équipes par la qualité et la visibilité des projets techniques sur lesquels nos consultants et développeurs travaillent. Ils sont ainsi moins enclins à écouter des sirènes extérieures.

P ! : Quid des profils féminin ?
JF : Là c'est plutôt aux écoles d'ingénieurs informatiques qu'il faudrait poser la question. Que font-ils pour changer l'image du développeur vis à vis des populations féminines ? Pour une jeune fille de 18 ans un développeur ou consultant informatique c'est plutôt "Rodolphe de Free" qu'autre chose ! Chez Valtech on a une proportion de femmes plutôt forte par rapport à la moyenne du marché. Et rappelons-le, notre DG Adjointe Lubomira Rochet est une femme dans un monde d'hommes d'ailleurs. ;-)

tée. Les recettes sont diverses : académie interne, formation intensive, missions en province ou à l'étranger…

Alter Way cherche aussi bien des juniors, ou seniors que des contrats de reconversion. La société investit beaucoup dans la formation autour du concept « Libre Académie ». La Libre Académie est à la fois dédiée à la formation magistrale et au tutorat professionnel, ouverte aux diplômés et aux reconversions professionnelles. *« La Libre Académie représente pour nous une approche nouvelle en matière de recrutement, de formation et d'emploi dans ce secteur des services numériques open source. Notre slogan: Faites vous plaisir, travaillez dans le Libre »,* explique **Philippe Montargès**, cofondateur d'Alter Way.

C'est un véritable investissement qui est réalisé sur les personnes. Sensio propose aussi des formations poussées à ces nouveaux arrivants.

Chez Linagora, la méthode est un peu différente.

« Nous faisons peu de sourcing même si nous changeons. Les candidatures sont surtout spontanées et par cooptation », intervient Alexandre Zapolsky de Linagora. Et pour lui, les bons experts Drupal sont des « Rolls du développement », difficiles à trouver ! Et la demande est très forte. Mais dans un contexte tendu, faut-il embaucher « tout » ce qui se présente ou au contraire, qualifier les candidats. Tout le monde s'accorde sur une chose : les délais de recrutement peuvent se rallonger de plusieurs semaines ! Et selon les exigences de la SSII, de l'intégrateur, le processus sera plus ou moins long. *« Nous cherchons des candidats ayant des qualités morales, une loyauté, un esprit sain et ouvert, bosseurs et rigoureux »*, précise avec force A. Zapolsky.

> *Drupal, la Rolls des compétences*

Il faut que les nouveaux arrivants, et les candidats, soient séduits par le cadre de travail, l'ambiance et l'esprit de l'entreprise. Ce qui n'est pas toujours simple. La SSII doit tout d'abord avoir une idée claire de sa vision, de l'esprit qu'elle veut inculquer. Et cela fait partie de la formation interne. La question du turn-over reste délicate pour les prestataires open source.

Les SSII open source cherchent, à garder les développeurs, l'enjeu est donc important. Il faut pour cela les impliquer dans des projets nouveaux et alterner les expériences internes.

La fidélité est un critère crucial pour éviter des départs au bout de 1 ou 2 ans. Car un recrutement coûte cher à l'entreprise et un départ trop rapide fait perdre de l'argent…

Chez Smile, la formation est très présente, avec un suivi régulier des collaborateurs par les RH et là aussi, mise sur la province et l'étranger. La province est une perspective professionnelle qui semble intéresser de plus en plus de nouveaux recrutés pour la qualité de la vie, le coût de la vie (particulièrement l'immobilier). Si la SSII possède des bureaux en province, cela peut être un atout et les candidats n'hésitent pas à demander !

■ François **Tonic**

ZOOM sur le marché de l'emploi

En avril, l'augmentation du nombre d'offres d'emplois confiées à l'APEC progresse de 78 % dans le secteur de l'informatique et de plus de 21 % dans les métiers du Web. S'il s'agit d'une bonne nouvelle pour le marché de l'emploi cela semble moins évident pour les entreprises qui doivent faire face à une réelle pénurie de candidats : *« Pour assurer la progression de covoiturage.fr nous avons ouvert quatre postes d'experts PHP en début d'année. A ce jour nous n'en avons pourvu que deux ! »* indique Frédéric Mazzella, directeur de Comuto, éditeur du site.

Dans ce contexte de plein emploi certains développeurs se sentent pousser des ailes et pratiquent un « zapping » de masse en sautant d'une entreprise à l'autre tous les six mois pour un meilleur salaire. Si la démarche est parfois justifiée, attention à ne pas en abuser : pour une entreprise, un employé commence à être vraiment utile au bout de six mois, parfois un an. Si vous avez plus d'une société par an dans votre expérience c'est un mauvais point sur un CV : le recruteur se méfiera car un mauvais recrutement coûte cher à l'entreprise. Un conseil aux informaticiens, privilégiez le confort de travail, c'est tellement mieux d'aller travailler avec le sourire.

Face à la raréfaction des profils expérimentés deux alternatives s'offrent aux entreprises : les écoles d'expertise informatique et l'alternance. Cette dernière présente deux avantages : des salaires raisonnables (pour l'entreprise) et des étudiants formés aux exigences et aux codes de l'entreprise. A noter qu'il s'agit également pour ces apprentis d'une excellente opportunité pour s'insérer efficacement sur le marché du travail et souvent obtenir un poste à la fin de leur formation.

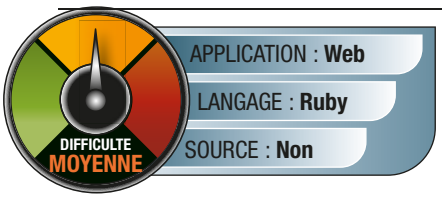
Parmi les solutions reconnues figure également le recrutement directement à la source. Considérée comme la principale école du secteur, EPITECH s'impose comme un vivier de talents, *« nos étudiants trouvent du travail avant même d'avoir terminé leurs études »*, confie Nicolas Sadirac, Directeur général d'EPITECH.

Seule fausse note, début 2011, malgré un discours volontariste du Gouvernement, la FAFIEC (organisme gestionnaire des budgets formation pour le secteur informatique) profite de l'aubaine pour diminuer la rémunération horaire des écoles d'alternance les forçant ainsi à augmenter le remplissage des classes pour atteindre leurs objectifs de rentabilité.

■ Cyril **Pierre de Geyer**
Consultant associé en stratégie Web et Innovation pour AGORATIC

La sécurité des applications Rails

La sécurité des applications web est souvent un sujet délicat : peu de temps à y consacrer, mais cela peut avoir des conséquences assez graves. Pas de panique ! Ruby on Rails est bien armé et avec un peu de rigueur, on peut se protéger sans trop de difficultés. Nous allons voir les principaux types d'attaques et comment les éviter.



INJECTIONS SQL

Commençons par un grand classique : les injections SQL. Une injection SQL consiste simplement à envoyer des données non prévues dans une requête SQL. Prenons comme exemple une application web où les utilisateurs sont authentifiés de la manière suivante :

```
    @current_user = User.where("login='#{params[:login]}' AND
 password='#{params[:password]}'").first
```

En temps normal, quand Joe s'authentifie, la requête SQL suivante est exécutée :

```
    SELECT * FROM users WHERE login='Joe' AND password='0521
bc575b0ff61daa62494c7ae9c5b6' LIMIT 1;
```

Mais supposons maintenant que Kevin, un Script Kiddie, passe dans le coin et décide de mettre "Joe' ; -" dans le champ login. La requête SQL va alors ressembler à :

```
    SELECT * FROM users WHERE login='Joe'; --' AND password=
'00000000000000000000000000000000' LIMIT 1;
```

Kevin a réussi à se faire passer pour Joe sans connaître son mot de passe ! Heureusement, Active Record permet de nous protéger assez facilement. Pour cela, il suffit d'utiliser les formes échappées ainsi :

```
    @current_user = User.where(:login => params[:login], :password
 => params[:password]).first
```

Active Record rajoutera un antislash devant chaque apostrophe de façon à éviter les injections SQL.

L'AUTHENTIFICATION ET LA GESTION DES DROITS

Pour la grande majorité des projets, l'authentification (et la gestion des droits qui vont avec) est un passage obligé. Pour cela, il existe un certain nombre de points importants à respecter comme le chiffrement des mots de passe stockés en base de données (ce que nous n'avons pas fait dans l'exemple précédent) ou réinitialiser la session après une authentification réussie. Les erreurs sont vite arrivées, aussi je vous recommande d'utiliser des plugins reconnus comme Devise et OmniAuth.

Il ne vous reste plus qu'à faire attention à un dernier détail : mettre en cache des pages nécessitant une authentification est une mauvaise idée. En effet, ces pages vont alors être servies par le serveur web sans passer par Rails, et donc sans vérification de l'authentification. Plus généralement, les caches demandent une attention toute particulière pour la sécurité.

SE PROTÉGER DES DONNÉES FORGÉES

L'étape suivante consiste à bien sécuriser l'accès aux données, aussi bien en lecture qu'en écriture. En effet, Rails possède quelques raccourcis très pratiques, mais qui peuvent poser problème quand ils sont mal maîtrisés. Le plus courant est l'affectation de masse, technique qui consiste à créer un objet Active Record directement depuis les paramètres de la requête HTTP. Par exemple, la création d'un compte utilisateur pourra s'effectuer de la façon suivante :

```
    @user = User.create(params[:user])
```

Supposons maintenant que la table users comporte un champ admin qui vaut 0 par défaut ou 1 pour les super-utilisateurs. Un utilisateur malveillant pourrait forger la requête HTTP pour ajouter un paramètre user[admin]=1 afin de gagner les pouvoirs réservés aux admins. La première solution pour se protéger de cette attaque consiste à écrire explicitement quels sont les paramètres autorisés :

```
    @user = User.create(:login => params[:user][:login],
                        :email => params[:user][:email],
                        :password => params[:user][:password],
                        :cgu => params[:user][:cgu])
```

Mais ceci peut vite devenir pénible quand on commence à avoir des formulaires un peu conséquents. C'est pourquoi on lui préfère généralement la deuxième solution : la déclaration dans le modèle de la liste des attributs qui ne peuvent pas être modifiés. Cette déclaration se fait à l'aide de la méthode attr_protected comme suit :

```
    class User < ActiveRecord::Base
      attr_protected :admin
      # ...
    end
```

Nous pouvons de nouveau utiliser l'affectation de masse sans craindre qu'un utilisateur se fasse passer pour un admin, Rails s'occupe de filtrer les paramètres.

Dans le même style, un attaquant peut essayer de forger des URL. Si, par exemple, l'utilisateur authentifié peut archiver l'item n°123 qui lui appartient, en appelant l'URL /items/archive/123, alors que se passera-t-il s'il appelle la même URL pour l'item n°456 qui ne

lui appartient pas ? La réponse dépend du code de la méthode archive. Une implémentation de base pourrait ressembler à :

```
class ItemsController < ApplicationController
  def archive
    Item.find(params[:id]).archive
    redirect_to items_url, :notice => "Item archivé"
  end
end
```

Pour se protéger des URL forgées, on pourrait la transformer en :

```
class ItemsController < ApplicationController
  def archive
    @item = @current_user.items.find(params[:id])
    if @item
      redirect_to items_url, :notice => "Item archivé"
    else
      redirect_to items_url, :alert => "Item introuvable"
    end
  end
end
```

Ce n'est pas parfait (on pourrait gérer les droits avec [CanCan] ou [Canable]), mais c'est déjà beaucoup mieux.
Un dernier petit truc pour la route avant de passer à autre chose. Si vous avez une API pour laquelle vous utilisez la serialisation XML ou JSON, faites attention à ce qu'elles ne renvoient pas des informations secrètes.

CROSS-SITE SCRIPTING

Jusqu'à maintenant, nous avons vu des attaques directes : un utilisateur essaye de s'en prendre à notre site. Il existe également des attaques plus pernicieuses que l'on classe sous le nom de Cross-Site Scripting (XSS en abrégé). Leur but est de s'en prendre aux utilisateurs de notre site en s'immisçant sur notre site. Ceci peut aller du spammeur qui mettra une balise <iframe> vers son site dans tous les formulaires qui lui passent sous la main à l'injection de javascript non maîtrisé. Par exemple, quelqu'un crée un item dont la description est la suivante :

```
<script>document.location='http://www.programmez.com/';
</script>
```

Si maintenant un visiteur affiche la description de cet item, il sera redirigé vers le site www.programmez.com. Vous vous dites que c'est ennuyeux mais pas bien méchant ? Oui, mais la même technique permet de voler les cookies et donc les sessions associées. Nous allons donc chercher à nous protéger de ces failles XSS.
En fait, ce n'est pas tout à fait vrai. Depuis Rails 3, les contenus dynamiques sont filtrés automatiquement avant d'être affichés. Si vous utilisez une version plus ancienne de Rails, pensez à échapper tous les contenus dynamiques avec le helper Rails h ou en utilisant un plugin comme XSS Terminate. Et si vous avez de bonnes raisons d'utiliser du HTML provenant de sources non-fiables bases de données, API, etc., vous pouvez toujours utiliser *sanitize* qui ne gardera que les balises non dangereuses et indiquera à Rails que cela peut être affiché sans filtrage.

REDIRECTIONS

Les attaques XSS ne portent pas uniquement sur le HTML. Certaines attaques peuvent aussi passer par des injections de CSS ou en utilisant les redirections.
Il existe une règle à respecter pour cela : ne faites jamais une redirection avec un élément qui provient de l'utilisateur. JAMAIS !
Mais voyons quelques ennuis qui pourraient vous arriver si vous ne respectez pas cette règle. Imaginons que vous ayez une méthode legacy :

```
def legacy
  redirect_to params.update(:action => 'main')
end
```

et qu'elle soit appelée depuis cette URL :

```
http://www.example.com/site/legacy?param1=xy&param2=23&
host=www.attacker.com
```

redirect_to va recevoir un argument :host => 'www.attacker.com' et va rediriger notre utilisateur vers le site de l'attaquant. Pas terrible, mais on peut faire pire en utilisant le protocole data qui permet, entre autres, de renvoyer du javascript qui sera interprété par le navigateur.

CROSS-SITE REQUEST FORGERIES

Pour finir, je voudrais juste dire un mot sur un dernier type d'attaques. Les CSRF, (abréviation de Cross-Site Request Forgery), sont des attaques complexes qui visent à forcer l'utilisateur à envoyer une requête HTTP vers notre site lorsque celui-ci visitera le site de l'attaquant. Je vous renvoie à wikipedia si vous voulez comprendre comment fonctionne ce type d'attaques. Sachez que Rails vous en protège. Pensez juste à utiliser une version à jour de Rails et à inclure cette ligne dans le <head> de vos _layouts_ :

```
<%= csrf_meta_tag %>
```

CONCLUSION

Nous avons pu voir qu'en prenant quelques bonnes habitudes, on pouvait développer des applications sûres en Rails. Il reste cependant des sujets que je n'ai pas abordés comme l'administration du serveur, la manipulation des fichiers ou encore le filtrage des informations sensibles dans les logs. Le guide Rails sur la sécurité est une lecture recommandée et pourra très certainement répondre à la plupart des questions que vous vous posez à ce sujet. Il est également important de se tenir au courant des mises à jour de sécurité de Rails et des plugins que vous utilisez.

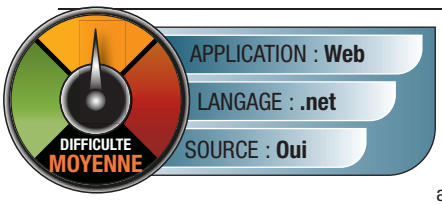
■ Bruno **Michel**
Ingénieur diplômé de l'Institut d'Informatique de l'Entreprise (IIE-CNAM). Il travaille actuellement comme développeur chez AF83 – http://www.af83.com/. Pendant son temps libre, il s'occupe du site web http://linuxfr.org et participe à la vie de deux associations dont il est membre : l'April http://www.april.org et Ruby France http://www.rubyfrance.org/.

Références

http://guides.rubyonrails.org/security.html
http://www.rorsecurity.info/
http://www.owasp.org/index.php/OWASP_Guide_Project
http://www.owasp.org/index.php/Top_10_2010-Main

Entity Framework 4.1 : Microsoft à la conquête du marché des ORM !

Suite à la première introduction d'Entity Framework (EF) dans .NET 3.5 SP1, les développeurs ont fait de nombreux retours afin de compléter et d'améliorer cette première version. Microsoft a entendu leurs desiderata et a apporté plusieurs améliorations avec EF 4.0. Pourtant les puristes du code n'étaient pas totalement satisfaits, en effet cette version nécessitait encore l'utilisation de modèles de design pour la création et la gestion des entités.



Avec la nouvelle version EF 4.1 on peut non seulement avoir une approche Database First et Model First mais aussi et surtout une approche Code First. Cette extension va faire d'Entity Framework un Data Layer complet et très performant.

APPROCHE DATABASE FIRST ET MODEL FIRST

L'approche Database First se fonde sur une base de données déjà existante dans laquelle on utilise des outils (comme EF Designer de Visual Studio) pour générer les classes C# ou VB. Les classes générées ainsi que les relations entre classes et base de données peuvent être modifiées par le Designer d'EF ou, pour les plus téméraires, via les fichiers XML … La priorité de cette approche est d'abord à la base de données, le code et le modèle viennent en second. L'approche Model First, quant à elle, consiste en un commencement « from scratch ». On débute par le modèle des entités, là encore via EF Designer. Ce modèle peut être utilisé pour générer la base de données ainsi que les classes C# ou VB.
La priorité de cette approche est donc donnée au modèle, le code et la base de données viennent ensuite.

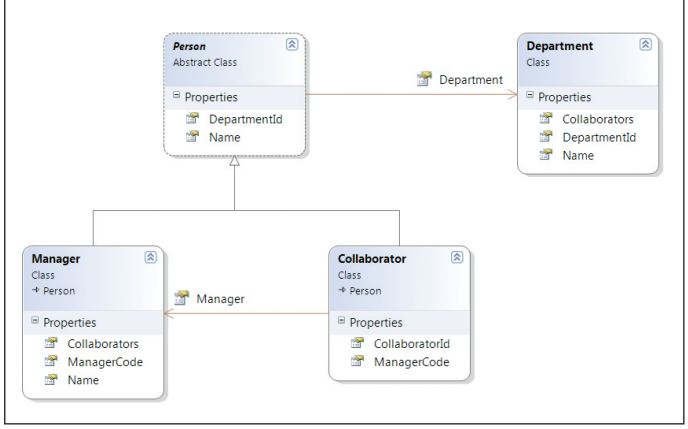
APPROCHE CODE FIRST

En plus de supporter le développement basé sur le modèle et le design, EF 4.1 offre une approche centrée en priorité sur le code. On appelle cela Code First : avec cette approche, on démarre par le code, il n'y a ni schéma, ni XML décrivant le modèle. On crée simplement les classes du domaine de l'application, Code First permettant leur utilisation avec EF. En utilisant un contexte, on peut écrire et exécuter des requêtes LINQ to Entities et tirer avantage du suivi de modification (Change Tracking) proposé par EF. Tout est géré de manière transparente, vous allez même oublier qu'EF est là !
EF 4.1 gère toutes les interactions avec la base de données pour vous. En arrière-plan, EF crée lors de l'exécution toutes les classes permettant de lier les objets avec la base de données : dans ce cas, le contexte n'est pas créé depuis les fichiers de configuration XML, mais calculé et généré en fonction de la configuration de la classe DbContext. Code First vous permet donc de :

- développer sans jamais avoir besoin d'utiliser le Designer ou définir des fichiers XML de mapping
- définir les objets du modèle selon la méthode «Plain Old CLR Objects » sans nécessiter des classes de base

Pour pouvoir utiliser ces nouvelles fonctionnalités, il faut installer la mise à jour d'EF 4.1. La Release Candidate est sorti mi-mars 2011, la version finale devrait déjà être disponible à la parution de cet article (en Release To Web et package NuGet). Cette mise à jour fonctionne bien évidemment avec VS 2010 et avec tout projet .NET 4 incluant ASP.NET et ASP.NET MVC. EF Code First permet d'utiliser des objets POCO (Plain Old CLR Objects) pour représenter les entités dans la base de données. Cela signifie qu'il n'est pas nécessaire de dériver les classes d'une classe de base spécifique à EF, ni d'implémenter des interfaces spécifiques ou des attributs de la persistance de données. Ceci permet aux classes de rester propres, facilement testables et réutilisables. Prenons le cas d'implémentation d'un domaine d'entreprise avec les entités *Managers*, *Collaborators* et *Departments*. [Fig.1]. Il suffit de créer les classes, comme vous le feriez de manière classique avec du code standard en utilisant les classes de base et les interfaces génériques. Par exemple, les classes Person et Collaborator ci-dessous sont définies comme des classes POCO standard, rien de spécifique n'a été rajouté. Elles peuvent être utilisées telles quelles avec EF 4.1 et Code First. A noter que pour que l'auto-mapping puisse fonctionner, il faut utiliser des propriétés nommées [Classname]Id pour les clés primaires.

Fig.1



```
public abstract class Person
{
    public string Name { get; set; }
    public int DepartmentId { get; set; }
    public virtual Department Department { get; set; }
}

public class Collaborator : Person
{
    public int CollaboratorId { get; set; }
    public string ManagerCode { get; set; }
    public virtual Manager Manager { get; set; }
}
```

EF vous permet de connecter très facilement les classes POCO à la base de données en utilisant la classe ObjectContext ou plus particulièrement la classe DbContext qui sert à relier les propriétés publiques de ces classes aux tables de la base de données. DbContext permet de travailler simplement avec les fonctionnalités les plus courantes d'ObjectContext. Ci-dessous, la classe CompanyContext hérite de la classe DbContext :

```
public class CompanyContext : DbContext
{
    public CompanyContext() : base(«CompanyDatabase») { }

    public DbSet<Collaborator> Collaborators { get; set; }
    public DbSet<Department> Departments { get; set; }
    public DbSet<Manager> Managers { get; set; }
}
```

Le code est maintenant prêt, la dernière étape consiste en la connexion avec la base de données. Il est possible par exemple d'utiliser le fichier de configuration (web.config ou app.config) pour définir la chaîne de connexion. EF Code First utilise la convention dans laquelle les classes DbContext cherchent par défaut la chaîne de connexion qui a le même nom que la classe du contexte. Il est aussi possible de définir un autre nom via un paramètre du constructeur (dans notre exemple : « CompanyDatabase »).

```
<?xml version=»1.0» encoding=»utf-8» ?>
<configuration>
  <connectionStrings>
    <add name=»CompanyDatabase» providerName=»System.Data.SqlClient» connectionString=»Server=.\SQLEXPRESS;Database=Products;Trusted_Connection=true;»/>
  </connectionStrings>
</configuration>
```

DATA ANNOTATIONS & CODE FIRST FLUENT API

La nouvelle version permet l'utilisation des Data Annotations. Les Data Annotations sont utilisées pour la validation ainsi que pour définir le mapping avec les colonnes de la base de données. Toutes les annotations sont implémentées comme attributs. EF 4.1 nous permet cela en s'appuyant sur des attributs se trouvant dans le namespace System.ComponentModel.DataAnnotations (ce sont les mêmes attributs utilisés pour la validation ASP.NET MVC).

Quelques attributs courants :

- [Key] – spécifie quelle(s) colonne(s) défini(ssen)t la clé primaire de la table.
- [StringLength] – spécifie la longueur maximale et la longueur minimale. La longueur minimale est utilisée uniquement pour la validation et non comme contrainte de la base de données.
- [MaxLength] – peut être utilisée à la place de [StringLength] pour spécifier la longueur maximale d'une colonne.
- [ConcurrencyCheck] – flag sur les colonnes qui active le check de concurrence.
- [Required] – spécifie quelle valeur est requise pour une propriété. La colonne est flaggée comme non nulle.
- [Timestamp] – flag sur les colonnes time stamp qui est utilisé pour le check de concurrence.
- [Column] – peut être utilisé pour spécifier le nom de colonne. Si vide, le nom de la propriété est repris dans le nom de la colonne. Cet attribut peut également être utilisé pour contrôler la position d'une colonne.
- [Table] – peut être utilisé pour spécifier le nom d'une table. Si vide, le nom de la classe est repris dans le nom de la table.
- [DatabaseGenerated] peut être utilisé pour signaler que la donnée est remplie via la base de données. Les valeurs par défaut sont « Computed », « Identity » ou « None ». « None » est renseigné par défaut.
- [NotMapped] Permet de définir une propriété dans une classe sans générer de colonne dans la base de données.
- [ForeignKey] et [InverseProperty] sont utilisés comme sur des colonnes avec des clés étrangères.

Voici un exemple d'utilisation de Data Annotations appliqué à notre domaine d'entreprise. La classe Manager ne contient pas de propriétés pouvant être identifiées automatiquement comme clé primaire. Par ajout de l'attribut [Key], on définit que la propriété Manager Code doit être interprétée comme clé primaire.

On ajoute aussi la vérification de concurrence ainsi que la longueur minimale et maximale (un message d'erreur est paramétré) de la chaîne de caractères pour la propriété Name. Ces attributs sont vus en détail plus loin.

```
public class Manager : Person
{
    [Key]
    public string ManagerCode { get; set; }
    [ConcurrencyCheck]
    [MinLength(5)]
    [MaxLength(20,ErrorMessage=»Le nom ne peut pas avoir plus de 20 caractères»)]
    public new string Name { get; set; }
    public virtual ICollection<Collaborator> Collaborators { get; set; }
}
```

Dans la classe Department, on rajoute aussi une Data Annotation [Required] pour que la propriété Name soit toujours renseignée.

```
public class Department
{
    public int DepartmentId { get; set; }
    [Required]
    public string Name { get; set; }
```

```
    public virtual ICollection<Collaborator> Collaborators { get;
 set; }
}
```

Les Data Annotations sont définitivement simples à utiliser, mais il est tout de même préférable d'utiliser de la programmation, afin que les classes n'aient aucune spécificité d'EF. Pour cela, il faut utiliser Code First Fluent API en implémentant la méthode OnModelCreating(…) de la classe dérivée CompanyContext et en définissant toutes les contraintes valables pour le domaine.

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Department>().Property(dp => dp.Name).
IsRequired();
   modelBuilder.Entity<Manager>().HasKey(ma => ma.ManagerCode);
    modelBuilder.Entity<Manager>().Property(ma => ma.Name)
        .IsConcurrencyToken(true)
        .HasMaxLength(20);
}
```

On peut même aller plus loin et décrire les opérations complexes en caractérisant les relations entre tables et colonnes. Le code ci-dessous montre que l'entité Manager contient un Department, la clé reliant les deux est DepartmentId. De plus, la suppression de Manager entraîne la suppression du Department. (CascadeOnDelete).

```
modelBuilder.Entity<Manager>()
    .HasRequired(d => d.Department)
    .WithMany()
    .HasForeignKey(d => d.DepartmentId)
    .WillCascadeOnDelete();
```

VALIDATION DES DONNÉES

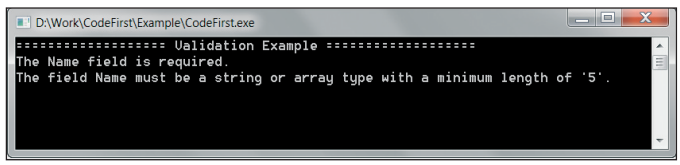
La validation des données est une problématique essentielle dans toute application. Il est important de s'assurer de la qualité et de l'intégrité des données afin de pouvoir les exploiter correctement et d'éviter les erreurs d'intégration dans la base.
Prenons l'exemple suivant : on s'assure que la propriété associée au champ Name est obligatoire et que sa valeur a une taille comprise entre 5 et 20 caractères via les Data Annotations.

```
[MinLength(5)]
[MaxLength(20,ErrorMessage=»Le nom ne peut pas avoir plus de 20
 caractères»)]
public new string Name { get; set; }
```

Par défaut, la validation va s'effectuer au moment de la sauvegarde des données (lors de l'appel de la méthode Save), mais il est également possible de valider les objets plus tôt en appelant la méthode GetValidationErrors(…).

```
public static void Validation()
{
    using (var context = new CompanyContext())
    {
        var manager = new Manager() { Name = string.Empty };
        context.Managers.Add(manager);

        var validationErrors = context.GetValidationErrors()
            .Where(vr => !vr.IsValid)
            .SelectMany(vr => vr.ValidationErrors);

        foreach (var error in validationErrors)
        {
            Console.WriteLine(error.ErrorMessage);
        }

        Console.ReadKey();
    }
}
```

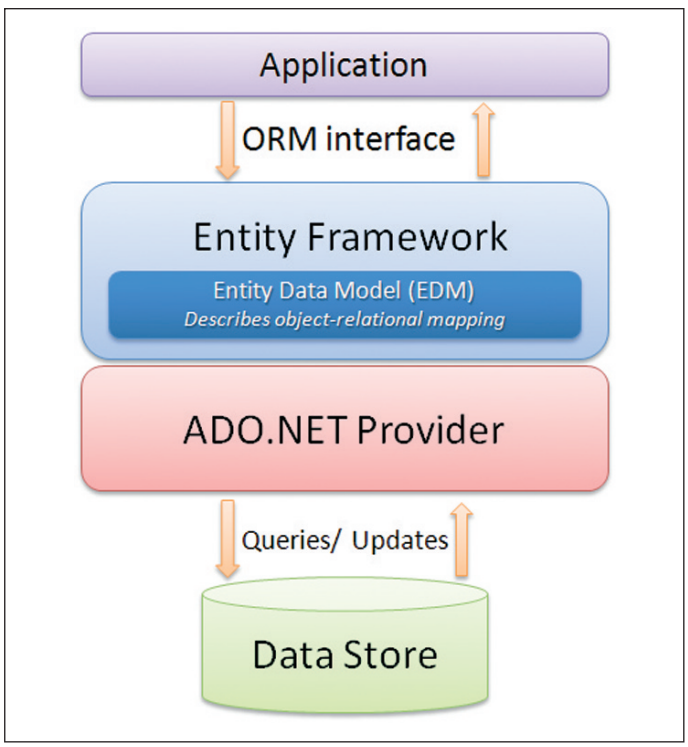
Quand le code s'exécute, la validation va échouer. Des messages d'erreurs apparaissent pour chaque contrainte non respectée. Dans notre cas, le minimum requis est 5 caractères, mais la propriété ne contient pas de valeur, ainsi un message générique est affiché [Fig.2].



Il existe bien évidemment d'autres Data Annotation permettant de valider les types, le format, les valeurs minimales et maximales autorisées, le respect d'une expression régulière…

AUDIT DES DONNÉES

Les propriétés de tout objet géré par EF peuvent être auditées. Cela est très utile lorsque l'on souhaite visualiser les changements effectués avant de sauvegarder, annuler les modifications en cours (on remplace la valeur actuelle par la valeur d'origine), ou encore anticiper un problème de concurrence lors de la sauvegarde.



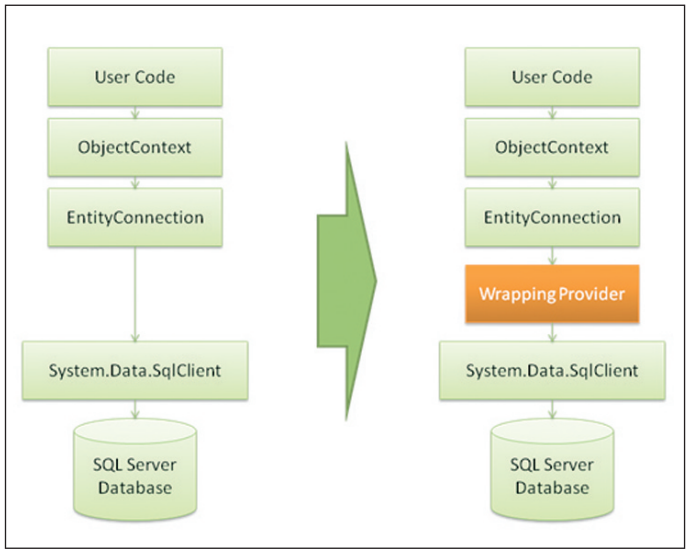
EF 4.1 vous permet d'accéder aux versions suivantes des propriétés
- d'une entité :
- la valeur en base de données (DataBase Value)
- la valeur initiale (Original Value)
- la valeur actuelle (Current Value)

```
public static void Audit()
{
    using (var context = new CompanyContext())
    {
        var manager = context.Managers.Find(«JDO»);

        using (var contextDB = new CompanyContext())
        {
            contextDB.Database.SqlCommand(
                «UPDATE managers SET Name = Name + '_DB' WHERE Manager
Code = 'JDO'»);
        }

        manager.Name = manager.Name + «_Memory»;

        string value = context.Entry(manager).Property(m => m.
Name).OriginalValue;
       Console.WriteLine(string.Format(«Original Value : {0}»,
 value));

        value = context.Entry(manager).Property(m => m.Name).
CurrentValue;
       Console.WriteLine(string.Format(«Current Value : {0}»,
 value));

       value = context.Entry(manager).GetDatabaseValues().Get
Value<string>(«Name»);
       Console.WriteLine(string.Format(«DB Value : {0}», value));

       Console.ReadKey();
    }
}
```

L'exécution de ce code permet de visualiser les différentes valeurs de la propriété Name de l'objet Manager [Fig.3]. Mettre en place l'audit est donc très simple, EF 4.1 contient déjà tout le nécessaire !

GESTION DE LA CONCURRENCE

L'un des problèmes fondamentaux liés à la gestion de base de données, est de pouvoir garantir la cohérence forte de celle-ci. Pour cela, on doit se munir d'un système de gestion de concurrence qui doit respecter les deux principes de base suivants :
- l'exécution simultanée des transactions produit les mêmes résultats que leur exécution séquentielle [Gardarin (1988)]
- l'exécution simultanée des transactions produit les mêmes résultats que l'exécution séquentielle des transactions dans l'ordre strict de leur arrivée [Rahgozar (1987)]



Nous allons voir dans cet exemple comment EF 4.1 nous permet, via la classe DbContext et ses multiples méthodes, de construire un gestionnaire de concurrence qui nous offre en cas de transactions simultanées l'application des deux solutions les plus courantes à cette problématique.

First Wins

First Wins est une stratégie de résolution de conflit dans laquelle la première modification est la gagnante. Les modifications apportées par la première transaction sont celles conservées, on notifie alors l'émetteur qui initie la seconde transaction qu'une modification des informations a déjà eu lieu (une exception peut être retournée).
Tout d'abord, nous devons déterminer quelles sont les propriétés sensibles à la concurrence, toutes les propriétés n'étant pas forcément critiques et ne nécessitant pas d'intégrer de processus de résolution de concurrence. Par exemple, si votre classe contient une propriété contenant la dernière date de mise à jour, il n'est pas nécessaire d'avoir de contrôle de concurrence sur cette propriété.
Pour réaliser ce check, il nous suffit d'ajouter l'attribut ConcurrencyCheck sur les propriétés de la classe de la manière suivante :

```
[ConcurrencyCheck]
public new string Name { get; set; }
```

On peut aussi utiliser la CodeFirst Fluent API sur les propriétés de la classe de la manière suivante :

```
modelBuilder.Entity<Manager>().Property(ma => ma.Name).IsConcur
rencyToken(true);
```

La question de la performance est à considérer. En effet, plus on aura de propriétés à contrôler, plus les performances se dégradent. Il est nécessaire d'utiliser cet attribut avec considération.

```
public static void FirstWins()
{
    using (var context = new CompanyContext())
    {
        var manager = context.Managers.Find(«JDO»);
        Console.WriteLine(«Initial Value: « + manager.Name);
```

```
        using (var contexDB = new CompanyContext())
        {
            contexDB.Database.SqlCommand(
                  «UPDATE managers SET Name = 'FirstWins' WHERE
 ManagerCode = 'JDO'»);
            Console.WriteLine(«DB Updated Value: FirstWins»);
        }

        manager.Name = «NewName»;
        try
        {
            context.SaveChanges();
        }
        catch (DbUpdateConcurrencyException ex)
        {
            ex.GetEntry(context).Reload();
        }

        manager = context.Managers.Find(«JDO»);
        Console.WriteLine(«Final Value: « + manager.Name);
    }
}
```

Premièrement, on récupère un manager depuis la base de données. Puis cet enregistrement est mis à jour directement en base afin de simuler un enregistrement concurrent. On modifie et on sauvegarde l'enregistrement avec la méthode SaveChanges(). Une exception de type DbUpdateConcurrencyException provenant du problème lié à la concurrence est retournée. C'est à ce niveau que le problème doit être géré en appliquant la règle choisie. Ici, on décide de conserver les premiers changements (First Wins) et on met à jour le contexte avec les valeurs issues de la base de données via la ligne de code ex.GetEntry(context).Reload().

Last Wins

C'est le mode de résolution par défaut appliqué par EF 4.1. C'est toujours celui qui écrit en dernier qui gagne. Les modifications liées à la première transaction sont écrasées. Si on souhaite être informé de la survenance de cas d'écrasement d'une transaction, on utilise l'attribut ConcurrencyCheck pour notifier l'utilisateur, écrire dans un fichier de log ou tout autre traitement.

```
public static void LastWins()
{
    using (var context = new CompanyContext())
    {
        bool savedChanges = false;

        var manager = context.Managers.Find(«JDO»);
        Console.WriteLine(«Initial Value: « + manager.Name);

        using (var contexDB = new CompanyContext())
        {
            contexDB.Database.SqlCommand(
               «UPDATE managers SET Name = 'LastWins' WHERE Mana
gerCode = 'JDO'»);
            Console.WriteLine(«DB Updated Value: LastWins»);
        }

        manager.Name = «NewName»;

        while (!savedChanges)
        {
            try
            {
                context.SaveChanges();
                savedChanges = true;
            }
            catch (DbUpdateConcurrencyException ex)
            {
                var entry = ex.GetEntry(context);
                entry.OriginalValues.SetValues(entry.GetDatabase
Values());
            }

        }

        manager = context.Managers.Find(«JDO»);
        Console.WriteLine(«Final Value: « + manager.Name);
    }
}
```

Contrairement au cas précédent, on relance la sauvegarde jusqu'au moment où celle-ci est réalisée sans exception. Dans le cas où une exception apparaît pendant cette opération, on synchronise le contexte (en remplaçant la valeur d'origine par celle remontée dans l'exception), puis on retente une sauvegarde. Ainsi, on est en mesure d'éviter la concurrence de transactions multiples tout en permettant aux développeurs de pouvoir ajouter des traitements (de log par exemple) lorsque l'exception DbUpdateConcurrencyException est levée.

CONCLUSION

Nous espérons que cette présentation de la nouvelle version d'EF de Microsoft vous a donné envie de jouer avec ! Même nous, qui sommes des puristes du code et du design, sommes conquis par la simplicité et la puissance du framework.
Grâce à ces nouveautés, EF devient l'un des produits les plus complet du marché, car il est le seul à supporter les 3 approches : Code First, Database First et Model First.
Evidemment, il reste encore des choses à améliorer : la possibilité de mettre à jour automatiquement la base de données lorsque la structure des classes est modifiée, ou encore certaines fonctionnalités manquantes comme le support des procédures stockées.
La bonne nouvelle, c'est que l'équipe en charge du développement d'EF est déjà retournée au travail. Vivement la prochaine version !

Retrouvez le **code** des exemples sur http://codefirst.codeplex.com !

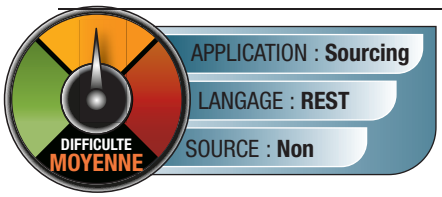
■ Jason **De Oliveira**
Practice Manager & Solutions Architect | MVP C#
.Net Rangers by Sogeti
Blog: http://jasondeoliveira.com

■ Fathi **Bellahcene**
Software Architect .Net Rangers by Sogeti
Blog: http://blogs.codes-sources.com/fathi

Jenkins en action

Jenkins

Jenkins – que vous connaissiez sans doute sous le nom de « Hudson », avant qu'Oracle ne fasse des siennes– est le serveur d'intégration continue dominant dans le microcosme Java. Sa principale force vient de la richesse de son écosystème de plugins. Je vous invite à me suivre dans cette visite guidée.



INSTALLATION

Peut on vraiment parler d'installation dans le cas de Jenkins ? Téléchargez le .war sur **jenkins-ci.org** et lancez java –jar jenkins.war. Voilà, c'est prêt, direction http://localhost:8080. La page d'administration de Jenkins nous permet de configurer tous les petits détails nécessaires, en particulier d'installer les plugins qui manquent pour adapter Jenkins à nos besoins.

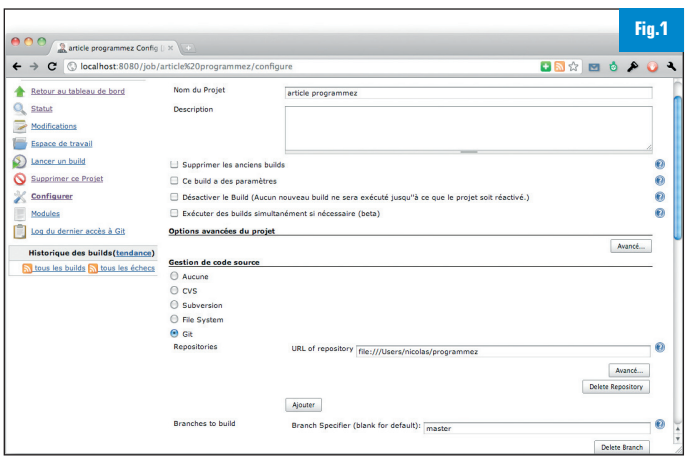
INTÉGRONS, INTÉGRONS

La base d'une intégration continue est le « gestionnaire de version » (VCS) qui gère l'historique de notre code source et toutes les misères que nous lui faisons subir. Jenkins propose un plugin pour tous les VCS courants et moins courants : Subversion, ClearCase, ou même PVCS, vous n'avez aucune excuse.

Depuis un moment déjà, les VCS distribués font parler d'eux ; nous n'allons pas déroger à la règle et donc jouer avec Git ! Notre projet est donc un projet Maven 3 sous Git, et après avoir installé le plugin Jenkins associé, nous configurons en quelques clics l'intégration continue de notre code [Fig.1]. Pour être efficace, l'intégration continue doit être capable de nous signaler les erreurs qu'elle détecte. La panoplie de plugins de notification nous offre un choix énorme dans lequel chacun piochera ce qui lui convient le mieux : mail, messagerie instantanée, son et lumière, écran spécifique, lampes à lave placées dans l'openspace, ou encore – pour les plus geeks – twitter.

BRANCHONS UN PEU

La liberté qu'apporte un DVCS nous permet d'inventer des modèles d'organisation du travail qui dépassent le bon vieux Subversion centralisé et ses multiples travers. Nous décidons par exemple de développer chaque fonctionnalité dans une branche dédiée (« feature-branch »). Nous la mettrons au point dans le moindre détail avant de l'intégrer au reste du logiciel.



Avantage de cette approche : tant que notre travail n'est pas terminé, le reste du logiciel n'est pas impacté. De même, nous ne sommes pas impactés par les aléas du travail sur d'autres fonctionnalités. Le travail dans chaque branche est beaucoup plus serein, et avance au rythme calme de quelques commits clairs par jour.

Inconvénient majeur : en retardant l'intégration de nos développements avec le reste du projet, nous allons à contre-courant de la logique d'intégration continue, à savoir identifier les bugs dès leur introduction.

La dynamique open-source qui anime Jenkins fait que nous ne sommes pas les premiers à rencontrer ce problème, et pouvons donc bénéficier de l'expérience de nos prédécesseurs. Le plugin Git pour Jenkins propose plusieurs modes de fonctionnement, pour savoir quelle(s) branche(s) il doit scruter, et que faire avec le code qu'il y trouve. Entre autres, nous pouvons lui demander d'effectuer un merge de notre code avec une autre branche…

A CHACUN SON JOB

Jenkins nous permet de créer une nouvelle tâche d'intégration continue à partir d'un job existant par recopie. Nous allons donc créer pour chaque feature-branch un job dédié. Pour les plus geeks, un script peut être utilisé comme *hook* Git pour créer ce job automatiquement à chaque nouvelle branche, via les API REST de Jenkins. Dans notre cas, nous créons deux petites sœurs à notre intégration continue :

- Une première qui va intégrer notre branche, de manière isolée. Une sorte d'intégration continue pour notre fonctionnalité prise isolément.
- Une seconde qui va intégrer notre branche en la fusionnant (*mergeant*) avec la branche *master*, vérifiant par anticipation que nos modifications s'intégreront sans souci avec le reste du logiciel.

La précaution n'aura pas été inutile, car Jenkins détecte rapidement un conflit entre notre branche et le master. Ne rêvons pas, cela va arriver plus d'une fois, mais l'intégration continue nous permet d'être notifié au plus tôt et de prendre en compte rapidement ce problème, sans pour autant perdre l'intérêt de travailler dans une branche isolée.

Quelques manipulations Git plus tard, après avoir identifié et fusionné le commit qui provoque le conflit, notre intégration continue voit de nouveau la vie en bleu, et nous pouvons repartir travailler le cœur léger [Fig.2 et 3].

SNAPSHOT

Quelques mois plus tard, notre projet a pris de l'embonpoint et est devenu un de ces monstres Maven multi-modules à 58 têtes ! Il devient hors de question pour nos développeurs sous Eclipse d'ouvrir tous les modules dans leur workspace – bien sûr, on pourrait leur conseiller d'utiliser IntelliJ Idea, mais c'est un autre débat.

Maven propose le mécanisme de SNAPSHOT pour ce cas de figure, avec la « dernière version » de chaque module. Une mise en œuvre « classique » consiste donc à faire tourner sur l'intégration continue une tâche Maven deploy pour mettre à disposition des SNAPSHOT à jour. Mais … (car il y a toujours un « mais ») il y a un effet de bord plutôt indésirable : si l'intégration tombe sur un bug sur le module N, tous les modules précédents auront été déployés. Nos SNAPSHOT risquent donc d'être incohérents.

Une nouvelle fois, c'est un plugin Jenkins qui va venir compléter notre forge logicielle. Le « Jenkins Maven Repository Server » va transformer notre intégration continue en exposant les résultats de build sous forme de dépôt Maven. Pas besoin d'un serveur Nexus dédié, de droits de déploiement à configurer. De nombreuses URL, dans une logique REST, nous permettent d'accéder aux résultats d'un build particulier, du dernier build réussi, pour un projet donné ou pour l'ensemble des jobs, voire pour un commit Git précis.

Pour notre cas, l'ajout d'un dépôt à notre configuration Maven suffit à obtenir les derniers SNAPSHOT du projet :

```
<repository>
  <id>jenkins</id>
  <url> http://jenkins/plugin/repository/project/article/Last
Successful/repository
  </url>
  <snapshots>
    <enabled>true</enabled>
  </snapshots>
</repository>
```

DANS LES NUAGES

Les choses se compliquent : notre projet devient de plus en plus ambitieux et dépasse les capacités de notre machine d'intégration continue. Malgré nos efforts pour rationaliser les tests, le build prend toujours plus de temps, et la qualité du feedback se détériore.

Fig.2

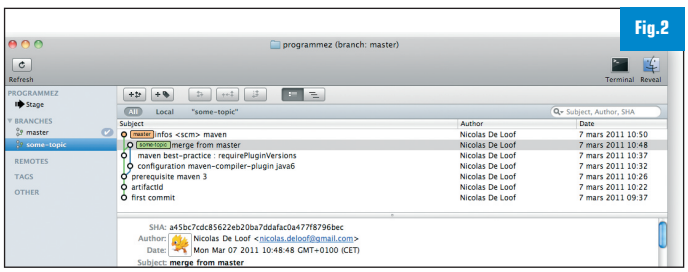


Fig.3

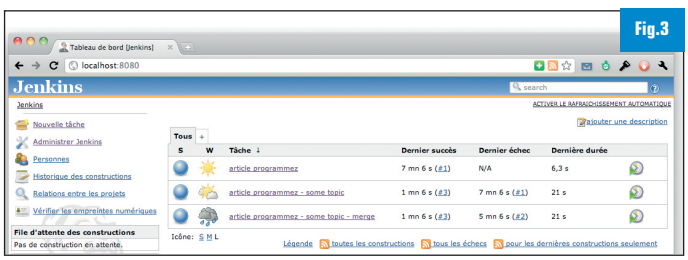
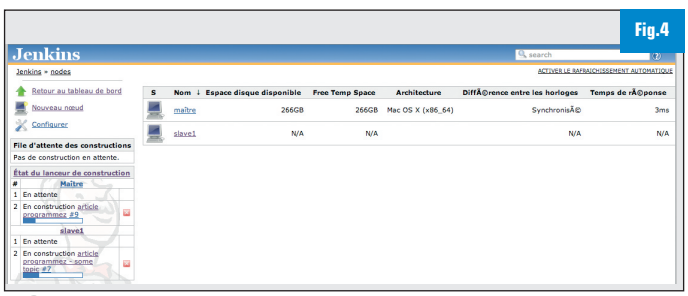


Fig.4



Il nous faudrait un plus gros serveur … ou plus de serveurs ?

Jenkins dispose d'un mécanisme de pilotage maître-esclave. Il nous permet d'étendre les ressources de notre intégration continue via d'autres machines. Jenkins pousse la facilité de mise en œuvre très loin en proposant une auto-installation des outils. Nous pouvons ajouter un nœud à notre « ferme d'intégration continue » en configurant simplement une connexion SSH à la machine cible, ou encore en exécutant dessus une commande Java Web Start. Le nouvel esclave mettra ses ressources à disposition de Jenkins, qui pourra répartir son travail [Fig.4].

RELEASE EARLY, RELEASE OFTEN

Etant une équipe agile et fière de l'être, nous effectuons une livraison tous les quinze jours. Le processus de *release*, utilisé à cette fréquence, ne peut plus être ce fameux moment de stress bien connu sur les projets marathon. Nous en avons fait un non-événement en confiant à notre intégration continue le soin de le porter dans son intégralité.

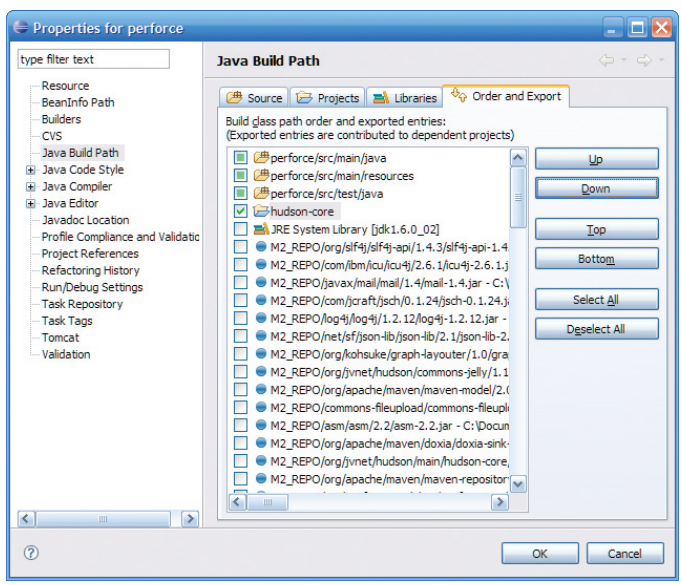
En installant le plugin Jenkins « Maven Release », un simple clic (et un peu de patience) suffit à lancer la procédure, totalement automatisée et configurée dans notre build Maven via le maven-release-plugin, et complétée côté Jenkins par une batterie de tests avancés : déploiement sur notre serveur de démonstration, tests fonctionnels Selenium, tests de performance jMeter… Ici encore, des plugins Jenkins viennent améliorer la présentation de notre intégration continue pour agréger le résultat de ces tests.

Notre objectif : pousser encore plus loin l'automatisation du processus et mettre en place un déploiement continu. Jenkins à tout le potentiel pour porter cette pratique, et déjà bon nombre de plugins pour nous y aider – il y a même un plugin BPM pour modéliser notre processus si besoin. Le dernier pas à franchir n'est plus technique mais organisationnel !

Comme nous l'avons vu, Jenkins n'est pas simplement un serveur d'intégration continue. C'est un écosystème en mouvement permanent, où bouillonnent des idées fracassantes qui poussent notre forge logicielle au-delà de nos espérances. Avec un outil aussi dynamique et bien inspiré, la seule limite, c'est notre imagination.

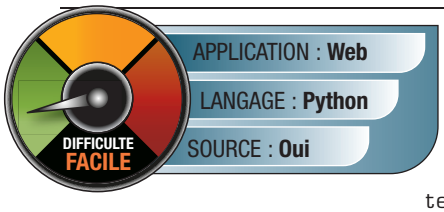
■ Nicolas **De Loof**

Senior Engineer chez CloudBees, Committer Apache Maven et JUG-Leader Rennais - http://blog.loof.fr



Traduire des textes avec les API Google et Bing

Google et Bing proposent un service de traduction des pages sur lesquelles vous êtes arrivés. Conçus à la base pour les navigateurs Web, ces services peuvent être utilisés très diversement et avec de multiples langages.



Les moteurs de recherche les plus populaires, Google, Bing, Yahoo,... proposent tous un service de traduction qui permet au navigateur d'afficher dans votre langue natale les pages visitées. Les API de ces services sont toutes des API REST. Il est aisé de les mettre en œuvre. Et si Javascript, en tant que langage embarqué des navigateurs, est le langage usuel, il est possible d'employer, hors du navigateur, n'importe quel langage capable de travailler en réseau. Nous allons utiliser les API de Google et Bing. Leurs documentations sont respectivement focalisées sur Javascript et C#. Pour varier les plaisirs et écrire du code clair et concis nous allons employer Python :-)

1 QUELQUES MOTS SUR REST

De plus en plus utilisé pour la mise en place d'architectures orientées services, REST est, à la base, l'architecture du Web. Son principe est de donner une représentation de l'état d'une ressource présente sur un serveur situé quelque part sur le réseau. Une ressource peut être une simple page HTML, un script PHP, ou quelque chose de plus complexe, comme un script PHP plus des données dans une base de données. Le client n'a pas à se préoccuper de l'organisation côté serveur, et dans tous les cas, on accède à la ressource au moyen d'une URL. Par exemple, soit une ressource sous la forme d'un fichier index.php auquel on accèderait en donnant son URL dans un navigateur : http://www.programmmez.com/index.php
Le navigateur affichera alors la page d'accueil du site de Programmez! qui est une représentation de cette ressource. Les URL peuvent être un petit peu plus compliquées. Par exemple :
http://www.monserveur.com/commande/executer?param1=10&param2=5
Ici nous accédons à la ressource 'exécuter' par le chemin commande. Ceci correspond peut-être à un répertoire nommé commande, ou peut-être que 'commande' sera reçu comme une action à exécuter côté serveur. En effet, en général rien n'indique comment une URL, pouvant être, qui plus est, un alias, sera exploitée côté serveur. Enfin l'URL passe des paramètres dont les valeurs sont respectivement 10 et 5.

2 LE PROTOCOLE HTTP

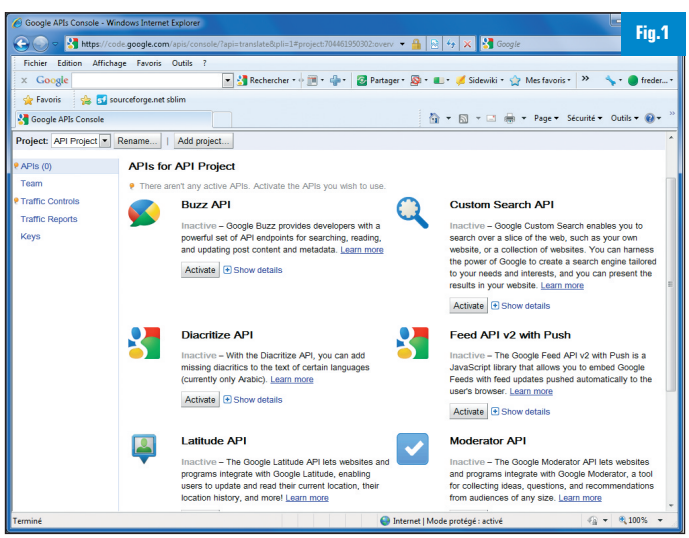
Dans tous les cas, nos URL travailleront avec les protocoles HTTP et HTTPS (http sécurisé). Ces protocoles définissent de quelle manière les données doivent être organisées pour envoi vers le serveur et aussi comment est organisée la réponse de celui-ci. Comme nous utilisons un langage de haut niveau qui manipule tout cela pour nous, nous ne nous attarderons pas sur ces questions aujourd'hui. Le lecteur intéressé par ce protocole peut s'il le souhaite, se reporter à l'article "Interroger Google Analytics" paru dans Programmez! N° 131

3 GOOGLE ET LA CONSOLE API

Pour utiliser les API Google il est nécessaire d'obtenir une clé, ce qui nécessite d'avoir un compte Gmail. En outre depuis quelque temps, Google regroupe les API dans ce qui est appelé une console, mais n'est en fait qu'une simple page Web située à cette adresse : [Fig.1].
http://code.google.com/apis/console/
Sur cette page, vous devrez créer une application, et alors une clé vous sera attribuée. Vous devrez ensuite activer la ou les API que vous souhaitez utiliser. Pour notre propos, vous activerez la *Translate* API. L'utilisation de l'API est gratuite, mais limitée à la traduction de 100 000 caractères par jour au moment de la rédaction de cette article. En outre, la taille du texte à traduire est limitée à 2 Ko pour une requête GET et à 5 Ko pour une requête POST. Enfin le résultat renvoyé par le serveur de Google est au format JSON.

4 HELLO WORLD 1.0

La tradition étant ce qu'elle est, nous allons demander à l'API de traduire en français la fameuse phrase Hello World. Nous commençons par écrire un premier jet. La documentation nous apprend que l'API se trouve sous le domaine www.googleapis.com. Le chemin language/translate/v2 nous permettra d'accéder à l'API dans sa version 2 et nous devons lui fournir des paramètres. L'ordre de ceux-ci est quelconque. Nous fournissons d'abord la fameuse clé en valeur



Désormais une 'console' regroupe les différentes API de Google

du paramètre *key*, puis le texte à traduire en valeur de paramètre *q*, puis le langage d'origine en valeur du paramètre *source*, le langage cible en valeur du paramètre *target* et finalement nous demandons un affichage agréable du résultat en donnant la valeur true au paramètre *prettyprint*. Au final voici cette URL :

```
https://www.googleapis.com/language/translate/v2?key=xxxxxxxx
xxxxxxxxxxxx6ZRQ&q= Hello+World&source=en&target=fr&prettyprint
=true
```

C'est parfaitement rébarbatif à écrire. En outre, votre oeil de lynx aura immédiatement remarqué que le texte passé est Hello+World et non Hello World. En effet certains caractères sont illégaux dans une URL, c'est le cas de l'espace. C'est pourquoi une URL doit être encodée en caractères légaux, ce qui est un autre travail rébarbatif. Heureusement Python va faire tout cela pour nous :

```
#! /usr/bin/env python
# RawTranslateClient.py

import httplib, urllib

params = urllib.urlencode(
 {
 'key' : 'AIzaSxxxxxxxxxxxxxxxxxJYiE6ZRQ'
 'q' : 'Hello World',
 'source' : 'en',
 'target' : 'fr',
 'prettyprint' : 'true'
   })

print "Params:", params

headers = {"Content-type" : "application/x-www-form-urlencoded"}
print "Headers", headers

command="/language/translate/v2?"
command += params
print "Commande: ", command
print "----------------------------"
```

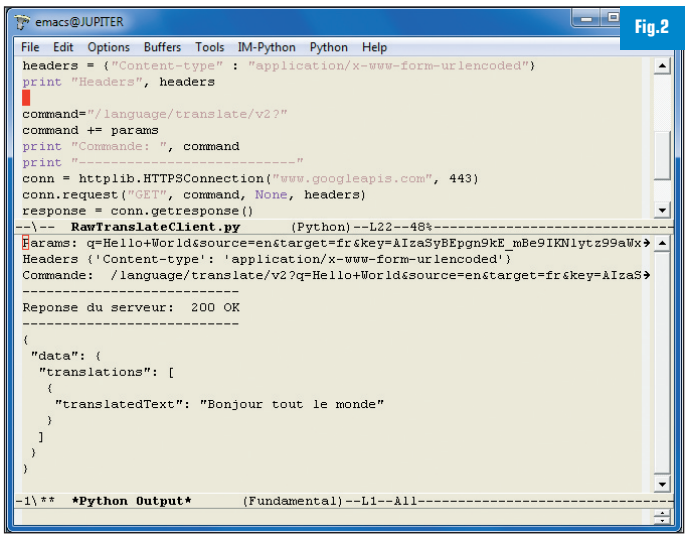


Fig.2

Notre premier script en action sous Emacs

```
conn = httplib.HTTPSConnection("www.googleapis.com", 443)
conn.request("GET", command, None, headers)
response = conn.getresponse()

print "Reponse du serveur: " , response.status, response.reason
print "----------------------------"

data = response.read()
print data
conn.close()
```

Ce code que nous voyons en action sous Emacs [Fig.2] encode les paramètres que nous fournissons dans un dictionnaire. Ensuite nous précisons dans l'en-tête de la requête (variable headers) que cette URL est encodée. Ceci est facultatif avec Google Translate et une requête GET, mais qui peut le plus peut le moins :-) Enfin nous ouvrons la connexion. On remarque qu'il s'agit d'une connexion sécurisée, protocole HTTPS. Enfin nous recevons la réponse du serveur. Une valeur de status de 200 indique que tout s'est bien passé. Enfin nous lisons le résultat :

```
{
 "data": {
   "translations": [
     {
       "translatedText": "Bonjour tout le monde"
     }
   ]
 }
}
```

4 NOTIONS DE JSON

Celui-ci est au format JSON, pour Javascript Object Notation. JSON est un format de données très intuitif. Un document JSON ne comprend que deux éléments structurels: des ensembles de paires nom-valeur et des listes ordonnées de valeurs. Comme son nom l'indique, il va comme un gant à Javascript. Il va également comme un gant au génial Python. Avec lui, les ensembles de paires nom-valeur trouvent une correspondance dans les dictionnaires et bien évidemment les listes sont des listes.... Python sait donc parser du JSON. Nous allons ainsi affiner notre code.

5 HELLO WORLD 2.0

```
#! /usr/bin/env python
# -*- coding: Latin_1 -*-
# TranslateClient.py
import httplib, urllib
import json

texte = 'Hello World'

params = urllib.urlencode(
    {
    'key' : 'AIzaxxxxxxxxxxxxxxxxxxxxYiE6ZRQ',
    'q' : texte,
    'source' : 'en',
    'target' : 'fr',
```

```
    })

# ici l'entête n'est pas absolument requis
# un dictionnaire vide, headers = {}
# pourrait convenir
headers = {"Content-type" : "application/x-www-form-urlencoded"}
command="/language/translate/v2?"
command += params

conn = httplib.HTTPSConnection("www.googleapis.com", 443)
conn.request("GET", command, None, headers)
response = conn.getresponse()
print response.status, response.reason
data = response.read()
conn.close()

print "--------------------------"

result = json.loads(data)
print "texte traduit: ",
print result['data']['translations'][0]['translatedText']
```

On remarque que le paramètre prettyprint a été supprimé de la requête. En effet, celui-ci provoque l'ajout de sauts de lignes de la part de l'API qui ne conviendrait pas au parser JSON de Python. A l'exécution, ce code affiche fièrement :

```
texte traduit: Bonjour tout le monde
```

7 DÉTECTER LE LANGAGE SOURCE

L'API Google est capable de détecter seule en quel langage est écrit le texte à traduire. Dans ce cas, la structure de données du résultat se verra enrichie d'une paire dont le nom sera detectedSourceLanguage et la valeur sera celle du langage détecté. Ce qui revient à dire avec Python que le dictionnaire fourni en guise de résultat par le serveur sera enrichi d'une clé. Le code ci-dessous est modifié en conséquence pour bien se comporter dans tous les cas: si le langage source est spécifié normalement, s'il ne l'est pas et que la détection réussit, s'il ne l'est pas et que la détection échoue. Dans ce dernier cas, l'API retourne le texte d'origine.

```
#! /usr/bin/env python
# -*- coding: Latin_1 -*-
# TranslateClientDetect^.py
import httplib, urllib
import StringIO
import json

texte = 'Hello World'
# ci-dessous langage indétectable
#texte ="1a3bh"

params = urllib.urlencode(
    {
    'key' : 'AIzaXXXXXXXXXXXXXXXXXXJYiE6ZRQ',
    'q' : texte,
    # on ne specifie plus le langage source
    # 'source' : 'en',
    'target' : 'fr',
    })

headers = {"Content-type" : "application/x-www-form-urlencoded"}
command="/language/translate/v2?"
command += params

conn = httplib.HTTPSConnection("www.googleapis.com", 443)
conn.request("GET", command, None, headers)
response = conn.getresponse()
print response.status, response.reason
data = response.read()
conn.close()

print "--------------------------"

result = json.loads(data)
resultdict = result['data']['translations'][0]
if resultdict.has_key('detectedSourceLanguage'):
    print "langage détecté: ",
    print resultdict['detectedSourceLanguage']
else:
    print "le langage n'a pas été, ou n'a pas pu être détecté"
print "texte traduit: ",
print result['data']['translations'][0]['translatedText']
```

8 AVEC UNE REQUÊTE POST

Si nous voulons traduire un texte d'une taille jusqu'à 5 Ko, nous devons émettre une requête POST au lieu d'une requête GET. Voici comment :

```
#! /usr/bin/env python
# -*- coding: Latin_1 -*-
# TranslateClientPost.py
import httplib, urllib
import StringIO
import json

texte = 'Hello World'

params = urllib.urlencode(
 {
   'key' : 'AIzaSyBEpXXXXXXXXXXXXXXXXXYiE6ZRQ',
   'q' : texte,
   'source' : 'en',
   'target' : 'fr',
 })

# Attention, pour une requête POST
# l'en-tête "Content-type" : "application/x-www-form-urlencoded"
# est requis, contrairement aux requêtes GET
headers = {"Content-type" : "application/x-www-form-urlencoded",
    "X-HTTP-Method-Override" : "GET"}

# Remarquer la disparition (facultative) du point d'interrogation ?
command="/language/translate/v2"

conn = httplib.HTTPSConnection("www.googleapis.com", 443)
```


```
conn.request("POST", command, params, headers)
response = conn.getresponse()
print response.status, response.reason
data = response.read()
conn.close()

print "--------------------------"
result = json.loads(data)
print "texte traduit: ",
print result['data']['translations'][0]['translatedText']
```

Ce code présente quelques subtilités. On remarquera que les paramètres de la requête ne sont plus concaténés à l'URL. Pour une requête POST ces paramètres doivent être dans l'en-tête de la requête. Python se charge de les y placer lorsqu'ils sont transmis en argument à la méthode *request* de l'objet HTTPSConnection. Ensuite, toujours en ce qui concerne l'en-tête, il est cette fois absolument requis de préciser que les paramètres sont encodés, et enfin, comme l'explique la documentation, "X-HTTP-Method-Override" : "GET" indiquera au serveur qu'il doit traiter la requête POST comme une requête GET. Oui, la logique derrière tout cela échappe un peu…

9 AVEC BING

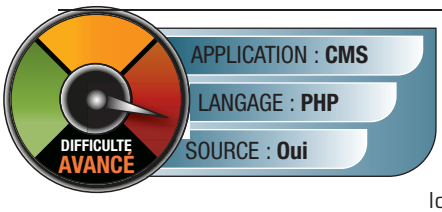
Bing fournit une API semblable à celle de Google, avec toutefois des différences qui la rendront plus ou moins intéressante, selon les goûts. D'abord le résultat de la traduction est différent. A chacun de choisir ce qui lui semble le plus pertinent. Bing est peut-être meilleur. La taille maximale du texte à traduire par requête est plus petite, mais il n'y a pas de limite de quota, et le protocole est HTTP. En revanche Bing ne sait pas détecter une langue source. Le résultat peut être fourni en JSON, mais aussi en XML ou SOAP. Il est simple de transposer à Bing ce que nous avons vu avec Google. Là encore il faut une clé appelée ici AppId, que l'on se procurera à http://www.bing.com/developers. Toutes les API Bing sont classées en 'Sources' (Video, Search, etc.). Pour la traduction, on spécifie 'Translation' au paramètre 'Sources'. Enfin on spécifie le format souhaité pour le résultat dans le chemin de la requête. Voici ce que cela donne :

```
#! /usr/bin/env python
# -*- coding: Latin_1 -*-

# TranslateClient.py
import httplib, urllib
import StringIO
import json


texte = 'Hello World'

params = urllib.urlencode(
 {
   'AppId' : 'E15xxxxxxxxxxxxxxxxxx358595F1',
   'Sources' : 'Translation',
   'Version' : '2.2',
   'Market' : 'fr-fr',
   'Query' : texte,
   'Translation.SourceLanguage' : 'en',
   'Translation.TargetLanguage' : 'fr',
 })


# header facultatif
headers = {"Content-type" : "application/x-www-form-urlencoded"}
command="/json.aspx?"
command += params

conn = httplib.HTTPConnection("api.bing.net")
conn.request("GET", command, None, headers)
response = conn.getresponse()
print response.status, response.reason
data = response.read()
conn.close()

print "--------------------------"

result = json.loads(data)
querydict = result['SearchResponse']['Query']
resultdict = result['SearchResponse']['Translation']
print "Texte à traduire: ", querydict['SearchTerms']
print "Texte traduit: ", resultdict['Results'][0]['Translated
Term']
```

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

Actions et déclencheurs sous Drupal 6 et 7

Drupal, CMS extrêmement souple, permet de définir des actions en réponse aux événements internes signalés par des déclencheurs. Voyons comment mettre ceci en pratique sous Drupal 6 et 7



Drupal est un CMS hautement extensible, au moyen de modules. Classiquement, un module définit des traitements effectués lors de situations bien précises, par exemple lorsque l'utilisateur valide un formulaire. Mais le besoin peut se faire sentir d'effectuer des traitements dans des situations peu ou pas définies à l'avance. Par exemple un mécanisme d'expédition de mails à l'administrateur ne saura pas d'avance de quoi celui-ci désire être notifié.

Dans un tel cas, Drupal prévoit la mise en place d'actions qui seront lancées, en réaction à des événements, par des déclencheurs associés à ces actions. Drupal vient bien sûr avec des actions et des déclencheurs par défaut, mais il est possible d'enrichir tout cela par programmation, ce que nous faisons maintenant. Pour cet article, nous supposons que le lecteur connaît PHP, et qu'il connaît au moins l'interface d'administration de Drupal.

La programmation action/déclencheur présente des similitudes avec la programmation 'simple' de module, et nous invitons le lecteur à (re)lire l'article «*Ecrire un module d'extension pour Drupal*» paru dans Programmez 123. Cet article se focalise sur Drupal 6, mais nous signalerons au passage les particularités relatives à Drupal 7 sorti récemment.

1 PETIT RAPPEL AU SUJET DES HOOKS

Sous Drupal, tout se passe avec un mécanisme de fonctions de rappel, ou hooks, que le cœur du système se charge d'invoquer aux moments opportuns. Ces hooks sont définis à l'intérieur de modules et obéissent à une convention de nommage précise. Nous verrons plus loin que les actions et les déclencheurs obéissent à des règles similaires, un petit peu compliquées par le fait que le contexte d'utilisation n'est pas connu à l'avance. Commençons par nous rafraîchir la mémoire en écrivant un module rudimentaire. D'abord son fichier .info (progaction.info sur notre site)

```
; $Id$

name = Progaction
description = Demo de Programmation Action/Declencheur
core = 6.x
php = 5.1
```

Puis le code (fichier progaction.module sur notre site). Le code est délimité par les commentaires 'Etape 1' dans le source d'exemple.

```
function writelog($msg)
{
 watchdog('progaction', $msg);
}

function progaction_user($op, &$edit, &$account, $category = NULL)
{
 if($op ='login')
 {
   writelog(«user login»);
 }
 if($op ='loginout')
 {
   writelog(«user logout»);
 }
}
```

Nous avons ici un hook, qui implémente hook_user. Donc, selon la convention de nommage, notre fonction s'appelle progaction_user. Le système invoque automatiquement cette fonction de notre module chaque fois que quelque chose se passe, qui est relatif à un utilisateur du site. Le quelque chose étant mieux défini via l'argument $op. Ainsi nous pouvons réagir selon, par exemple, que l'utilisateur se connecte ou déconnecte. Sous Drupal 7 les choses fonctionnent selon le même principe du hook. Cependant, l'opération ($op) n'est plus passée en argument du hook. A la place de cela, les hooks sont plus nombreux: un par opération. Lorsque ce hook est invoqué, notre fonction écrit un message dans le journal du système. Ceci est simple et clair, avec le problème que notre hook est par définition, ***irrémédiablement couplé*** avec les événements concernant l'utilisateur. Si nous voulons traiter des événements relatifs aux nœuds (insertion, effacement, vue, modification…) nous devons coder un deuxième hook, dont le nom par convention sera progaction_nodapi. Si nous voulons encore traiter des événements relatifs aux commentaires, nous devrons écrire une troisième fonction, ou hook, et ainsi de suite… Enfin, si notre hook d'exemple est irrémédiable couplé avec les événements utilisateur, il l'est aussi ***forcément***. L'administrateur du site ne peut décider de l'exécution ou non du hook. Fort heureusement vient le mécanisme action/déclencheur qui remédie à ces deux soucis.

2 UNE PREMIÈRE ACTION SIMPLE

Nous travaillons toujours avec le même module (mêmes fichiers .info et .module. Nous remettons le code précédent en commentaire, et nous nous intéressons maintenant au code de l'étape 2. A

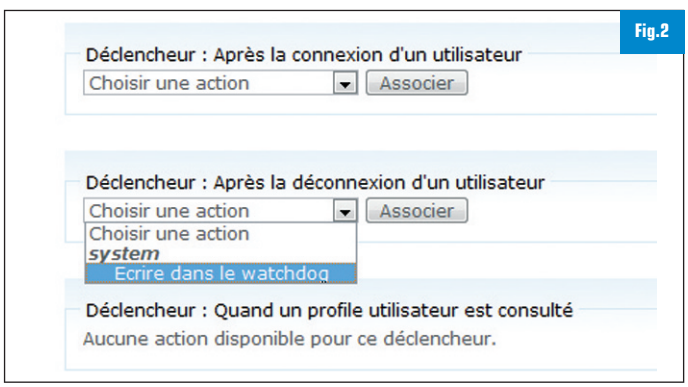
partir de maintenant, il est évident que le module 'Trigger' de votre Drupal doit être activé.

```
function writelog()
{
 watchdog('progaction', «Une action programmez»);
}

function progaction_action()
{
 writelog();
}

function progaction_action_info()
{
 $info['progaction_action'] = array(
   'type' => 'system',
   'description' => t('Ecrire dans le watchdog'),
   'configurable' => FALSE,
   'hooks' => array(
     'nodeapi' => array('view'),
     'user' => array('login', 'logout'),
   ),
 );
 return $info;
}
```

Actions disponibles pour Drupal : **Fig.1**

Type d'action	Description
comment	Retirer le commentaire
node	Mettre en lumière le contenu
node	Retirer le contenu de la mise en lumière
node	Promouvoir le contenu en page d'accueil
node	Publier le contenu
node	Enregistrer le contenu
node	Retirer le contenu de la page d'accueil
node	Retirer le contenu de la publication
system	Ecrire dans le watchdog
user	Bannir l'adresse IP de l'utilisateur courant
user	Bloquer cet utilisateur

Notre action simple apparaît désormais parmi les actions disponibles.



Association de notre action simple avec l'événement de déconnexion d'un utilisateur.

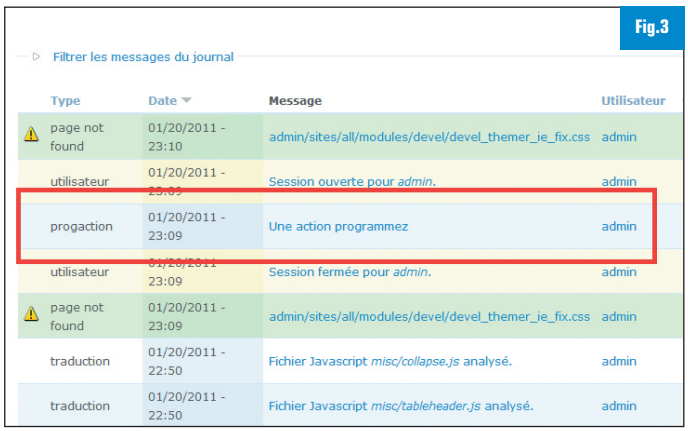
On remarque que pour l'instant notre fonction writelog ne reçoit plus de paramètre. En effet nous serions bien en peine de lui en passer, car les actions simples ne permettent pas de travailler avec des paramètres. Vient ensuite l'action proprement dite, nommée progaction_action, mais ceci n'obéit, pour une fois, à aucune convention. La dernière fonction est un hook :-) C'est l'implémentation de hook_action_info, que le système lancera lors de l'installation du module. Ici nous retournons au système un tableau d'actions (une seule dans cet exemple). Notre action est de type 'system', et sa description est à l'attention de l'administrateur du site. Notre action n'est pas configurable par définition car il s'agit d'une action simple. Enfin, notre action concerne deux hooks, nœuds et utilisateurs. Et seulement dans le cas des opérations 'view' pour les nœuds et 'login' et 'logout' pour l'utilisateur. Sous Drupal 7 on écrira ceci en lieu et place :

```
'triggers' => array('node_view', 'user_login', 'user_logout');
```

Lors de l'installation du module, le système va ajouter une entrée correspondant à notre action dans la table 'actions' de la base de données. Une désinstallation du module ne supprimera pas l'entrée dans la table. La désinstallation propre des modules sort du cadre de cet article. Notre action est donc désormais listée sur la page 'Actions' de l'interface d'administration (AdministrerlConfiguration du sitelAction) comme illustré [Fig.1]. En se rendant sur la page d'administration des déclencheurs (AdministrerlConstruction du sitelDéclencheurs) il est possible d'associer notre action. L'illustration [Fig.2] montre notre action sur le point d'être associée avec l'événement de déconnexion d'un utilisateur. A la suite de quoi, on peut vérifier que notre action écrit effectivement dans le journal en se rendant à AdministrerlRapportslEntrées récentes du journal [Fig.6]. Les geeks incorrigibles peuvent aussi consulter la table watchdog directement en base de données. Même si nous avons atteint notre premier but, le résultat reste peu satisfaisant: notre action ne peut recevoir de paramètres, et elle ne peut pas agir en fonction de l'événement, par exemple en consignant le nom de l'utilisateur dans le journal. Il est temps d'aborder maintenant les actions avancées.

3 LES ACTIONS AVANCÉES

Les actions avancées se caractérisent d'abord par le fait qu'elles sont configurables, ensuite qu'elles peuvent recevoir des paramètres de ladite configuration. Cette configuration se fait dans l'interface d'administration, ce qui implique d'exposer un formulaire, d'en valider les valeurs saisies et de stocker ces valeurs en base de



Notre action a écrit dans le journal.

données. Ceci nécessite trois fonctions supplémentaires. Ces fonctions obéissent cette fois à une convention de nommage avec les suffixes _form, _validate et _submit respectivement. Voici le code (étape 3 dans le source) qui réalise tout cela :

```
function writelog($msg)
{
 watchdog('progaction', $msg);
}

function progaction_action($object, $context)
{
 writelog($context['texte']);
}

function progaction_action_info()
{
 $info['progaction_action'] = array(
   'type' => 'system',
   'description' => t('Ecrire dans le watchdog'),
   'configurable' => TRUE,
   'hooks' => array(
     'nodeapi' => array('view'),
     'user' => array('login', 'logout'),
   ),
 );
 return $info;
}

function progaction_action_form($context)
{
 $form['texte'] = array(
 '#type' => 'textfield',
 '#title' => t('Donnez un texte'),
 '#description' => t(«Ce texte sera ecrit dans le watchdog quand
 l'action s'execute»),
 '#default_value' => isset($context['texte']) ? $context['texte'] :
 'Programmez!',
 '#required' => TRUE,
 );
 return $form;
}
```

```
function progaction_action_validate($form, $form_state)
{
 $texte = $form_state['values']['texte'];
 if(strlen($texte) > 25)
 {
   form_set_error('texte', t('Ce texte est trop long (25 char max)'));
 }
}

function progaction_action_submit($form, $form_state)
{
 return array(
   'texte' => $form_state['values']['texte'],
 );
}
```

On note que notre action reçoit désormais 2 arguments $object et $context. Le premier est l'objet concerné par ce qui a déclenché l'action. Par exemple un nœud, un commentaire, etc. Le second est le contexte dans lequel l'action est déclenchée. Nous savons qu'une action peut-être assignée à différents événements, comme par exemple ceux relatifs aux contenus ou aux utilisateurs. C'est ce que le contexte renseigne. En outre, comme le montre le code, le contexte contient les valeurs configurées depuis l'interface qui passe donc ainsi indirectement des arguments à l'action. Bien remarquer, dans le hook action_info, que nous avons positionné configurable à TRUE. Enfin, il est important de savoir que si le système stocke les noms des actions simples en base de données (table 'actions') ceci ne fait plus sens pour une action avancée, car l'administrateur peut définir plusieurs instances de celle-ci. C'est pourquoi les actions avancées reçoivent un identificateur numérique dans la table 'actions'. Une fois notre module (ré)installé, l'action avancée est à notre disposition dans l'interface [Fig.4] et si nous cliquons sur le bouton 'Créer' nous arrivons à une page permettant de définir le texte, ici Programmez!, qui sera passée à l'instance de notre action via le contexte [Fig.5].

4 MÉCANISME DE DÉCLENCHEMENT ET NORMALISATION DES CONTEXTES

Nous arrivons à un stade où il est pertinent de se poser quelques questions existentielles. Nous savons mettre en œuvre une action avancée, mais on peut se demander par quelle magie le système invoque les fonctions correspondant à nos actions, et par quelle

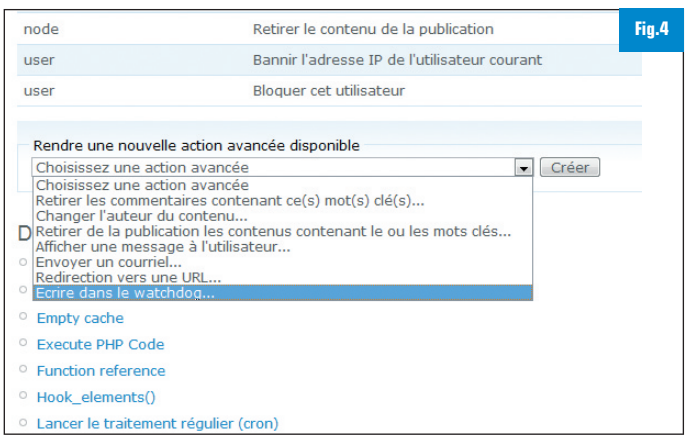


Fig.4

Notre action avancée est désormais disponible dans l'interface d'administration.

Fig.5

Configuration d'une instance d'une action avancée.

autre magie c'est le bon objet qui est passé à une action ? En outre, si nous voulons créer nos propres déclencheurs, nous devons savoir répondre à ces questions :-) Tout se passe dans le module Trigger (pour Déclencheur en français). Ce module fait partie des modules systèmes, donc il fait toujours partie de Drupal, mais le module doit être activé pour que le mécanisme action/déclencheur fonctionne, comme nous l'avons dit. Ce module n'est rien d'autre qu'un module, donc à la base il implémente des hooks, comme tous les modules. Comme par défaut les déclencheurs peuvent concerner 4 types d'événements: *user*, *content*, *comment* et *cron*, le module implémente 4 hooks: hook_user, hook_nodeapi (pour content), hook_comment et hook_cron, tout simplement. Nous savons que le cœur de Drupal invoque automatiquement les hooks dans les modules. Les hooks des modules traitent essentiellement 3 points. D'abord déterminer si une action est assignée à l'un de ces quatre événements. Ensuite créer un contexte et le normaliser. Enfin invoquer l'action en lui passant les arguments convenables. Le premier point se traite tout simplement en interrogeant la table 'trigger_assignements' en base de données. Ici, on fera attention à la façon dont est stockée l'action dans la colonne 'hook' de la table. Si son identificateur est une chaîne (son nom) alors il s'agit d'une action simple qu'il suffit d'invoquer. Si son identificateur est un nombre entier, nous avons affaire à une action avancée, qui attend des arguments. Ceux-ci sont stockés sous forme sérialisée dans la colonne 'parameters' de la table 'actions' [Fig.6]. Nous arrivons maintenant au deuxième point, le contexte. Celui-ci n'est rien d'autre qu'un tableau. Dans le cas d'une action avancée, il doit au moins contenir des clés correspondant aux noms des arguments attendus, associées à leur valeur. Ces données proviennent de la table action que nous venons de lire. Ensuite il faut une clé 'hook' qui contiendra le nom du hook de traitement dans lequel nous nous trouvons actuellement. Ainsi si nous sommes dans une implémentation de hook_nodeapi, on écrira :

```
$context['hook'] = 'nodeapi'
```

Et enfin, nous devons normaliser le contexte, c'est-à-dire fournir à l'action un $object en rapport avec le contexte. Par exemple, si nous avons une action dont le type est 'user', son code va s'attendre à recevoir un objet de type 'user'. Mais si l'administrateur du site a associé l'action à un événement de type nœud, nous rentrerons dans le hook nodeapi qui reçoit un objet de type 'node' en argument. Nous ne pouvons transmettre cet objet à l'action, mais au contraire lui fournir un objet encapsulant l'utilisateur courant. Faire ceci, c'est normaliser le contexte. Voici à titre d'exemple un extrait du module trigger qui est appelé par l'implémentation de hook_comment :

```
function _trigger_normalize_comment_context($type, $comment) {
  switch ($type) {
```

Fig.6

aid	type	callback	parameters	description
4	system	progaction_action	a:1:{s:5:"texte";s:11:"Programmez!";}	Ecrire dans le watchdog
comment_unpublish_action	comment	comment_unpublish_action		Retirer le commentaire
node_make_sticky_action	node	node_make_sticky_action		Mettre en lumière le contenu
node_make_unsticky_action	node	node_make_unsticky_action		Retirer le contenu de la mise en lumière
node_promote_action	node	node_promote_action		Promouvoir le contenu en page d'accueil
node_publish_action	node	node_publish_action		Publier le contenu
node_save_action	node	node_save_action		Enregistrer le contenu
node_unpromote_action	node	node_unpromote_action		Retirer le contenu de la page d'accueil
node_unpublish_action	node	node_unpublish_action		Retirer le contenu de la publication
user_block_ip_action	user	user_block_ip_action		Bannir l'adresse IP de l'utilisateur courant
user_block_user_action	user	user_block_user_action		Bloquer cet utilisateur

La table actions conserve les arguments des actions avancées sous forme sérialisée.

```
    // An action that works with nodes
    // is being called in a comment context.
    case 'node':
      return node_load(is_array($comment) ?
        $comment['nid'] : $comment->nid);

    // An action that works on users
    // is being called in a comment context.
    case 'user':
      return user_load(array('uid' => is_array($comment) ?
        $comment['uid'] : $comment->uid));
  }
}
```

Enfin quand tout ce travail de préparation est terminé, il ne reste plus qu'à invoquer l'action, ou éventuellement un groupe d'actions rangées dans un tableau, au moyen de la fonction actions_do. Et bien sûr, une action peut ainsi profiter de toutes ces informations. Ainsi (étape 4 dans l'exemple) voici une action qui écrit un message différent dans le journal, selon qu'il s'agit d'un contexte 'nodeapi' ou 'user' :

```
function progaction_action($object, $context)
{
  $msg = $context['texte'];

  switch($context['hook'])
  {
    case 'nodeapi':
      $msg .= ' Bonne lecture';
      break;

    case 'user':
      $msg .= 'Abonnez-vous ! :-)';
      break;

    default:
      break;
  }
  writelog($msg);
}
```

5 METTRE EN PLACE SES PROPRES DÉCLENCHEURS

A partir de ce que nous savons maintenant, nous pouvons mettre en place nos propres déclencheurs. C'est ce que fait le code de l'étape 5 dans notre exemple dont voici l'extrait important

```
function progaction_action_info()
{
  $info['progaction_action'] = array(
    'type' => 'system',
    'description' => t('Ecrire dans le watchdog'),
    'configurable' => TRUE,
    'hooks' => array(
      'progaction' => array('suivi_node',
      'suivi_connexion_user',
      'suivi_deconnexion_user'),
    ),
```

```
  );
  return $info;
}

function progaction_hook_info()
{
  return array(
    'progaction' => array(
      'progaction' => array(
        'suivi_node' => array(
          'runs when' => t('Un node est vu'),
        ),
        'suivi_connexion_user' => array(
          'runs when' => t('Un utilisateur se connecte')
        ),
        'suivi_deconnexion_user' => array(
          'runs when' => t('Un utilisateur se deconnecte')
        ),
      ),
    ),
  );
}

function progaction_nodeapi(&$node, $op, $a3 = NULL, $a4 = NULL)
{
  switch($op)
  {
    case 'view':
      $context = array();
      $context['hook'] = 'nodeapi';
      // Ici pseudo code de simplification
      // On doit interroger la table trigger_assignments
      // pour connaître la ou les actions
      // associés à suivi_node
      $id_action = 4;
      // Puis interroger la table action
      // pour connaître les paramètres des actions
      $context['texte'] = 'Programmez!';
      // enfin on doit normaliser le contexte
      $object = $node; // ou autre chose
      actions_do($id_action, $objetc, $context);
      break;
    default:
      break;
  }
}
```

La deuxième fonction est l'implémentation de hook_hook_info (hook_trigger_info sous Drupal 7). Nous définissons ici 3 déclencheurs. Les clés 'runs when' du tableau vont correspondre à l'affichage des déclencheurs dans l'interface. Dans :

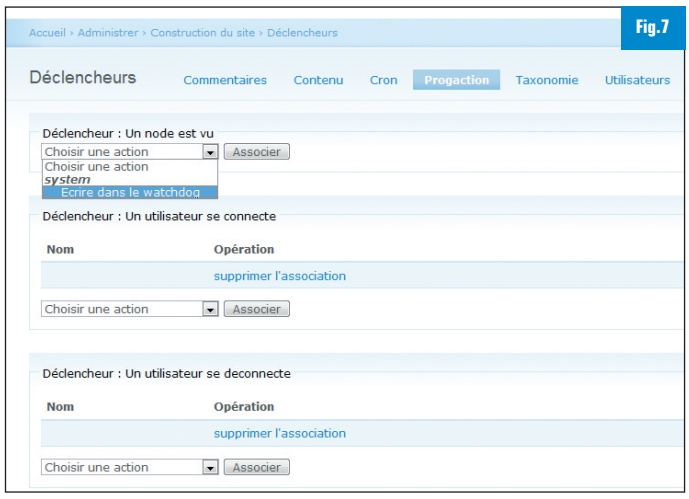
```
'progaction' => array(
  'progaction' => array(
    'suivi_node' => array( // etc...
```

le premier terme progaction définit un onglet 'Progaction' dans la page des déclencheurs, le second demande à ce que l'affichage des déclencheurs soit fait sous cet onglet [Fig.7]. Si nous avions donné :

```
'progaction' => array(
  'user' => array(
    'suivi_node' => array( // etc...
```

L'onglet 'Progaction' existerait toujours mais les déclencheurs seraient affichés sous l'onglet Utilisateur. Bien entendu, il est possible de panacher: afficher tous nos déclencheurs sous l'onglet 'Progaction', ainsi que quelques-uns seulement sous d'autres onglets. Enfin remarquer les noms d'opérations des déclencheurs ('suivi_node, etc.). Ces noms ne doivent pas comporter d'espaces et ils réapparaissent dans la première fonction, progaction_hook_info, qui déclare les actions. Vient enfin progaction_nodeapi en guise d'implémentation de hook_nodeapi. Ici ce n'est que du pseudo code, qui reprend la démarche expliquée au paragraphe 4. Pour transformer ce code en un code complet et fonctionnel, il suffira d'interroger la base de données au moyen des API de base de données Drupal … pour une prochaine fois :-)

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



Nos déclencheurs apparaissent dans l'interface d'administration sous un onglet dédié.

COMMERCE

Magento

*Difficulté : ***
Editeur : Pearson
Auteur : Christophe le Bot
Prix : 36 €.

Magento est une solution e-commerce très populaire depuis quelques mois. Grâce à sa version open source, Magento bénéficie de nombreuses fonctions pour monter rapidement une boutique en ligne de qualité professionnelle, à condition de maîtriser les mécaniques. C'est la mission de cet ouvrage, destiné à tous les utilisateurs de Magento, quel que soit leur rôle - e-commerçants, concepteurs de sites, développeurs... Complet et pratique, il vous aidera aussi bien à concevoir votre projet de boutique qu'à la créer de A à Z et à l'administrer au quotidien. Vous découvrirez les capacités de Magento et toutes ses fonctionnalités en détail, et approfondirez son utilisation pour améliorer votre site et y ajouter des extensions.

CLOUD

Cloud Computing
2e édition

*Difficulté : ***
Editeur : Dunod
Auteur : Guillaume Plouin
Prix : NC €.

Le cloud computing est une vaste nébuleuse où il n'est pas toujours simple de savoir « qui fait quoi ». Dans cette seconde édition, l'auteur replonge dans le cloud : définition, approches, acteurs, stratégie, et tente de répondre à la question : *le cloud,* c'est quoi et ça sert à quoi ? L'un des intérêts du livre est de proposer un discours clair s'appuyant sur de nombreuses sources. A lire si vous souhaitez comprendre les fondamentaux.

BUREAUTIQUE

Programmation OpenOffice.org et LibreOffice

*Difficulté : ****
Editeur : Eyrolles
Auteur : collectif
Prix : 45 €.

Écrit par deux contributeurs majeurs de la communauté francophone, ce livre est une référence incontournable sur le puissant langage de macros OOoBASIC et sur l'API d'OpenOffice.org et de LibreOffice (pratique pour le développeur de vérifier la bonne compatibilité). Il explique comment utiliser l'interface utilisateur liée aux macros et aux scripts afin d'automatiser des tâches répétitives, mais aussi comment tirer parti du langage de OOoBasic pour manipuler des documents, créer des boîtes de dialogue et des formulaires, exploiter des bases de données externes ou intégrées, intercepter des événements. Chaque point de l'API est décrit grâce à de nombreux exemples de macros et de routines réutilisables.

SPÉCIAL TESTS

Les tests logiciels

*Difficulté : *** - Editeur : Lavoisier - Auteur : Bernard Homès - Prix : 79 €.*

Les tests devraient être une préoccupation des développeurs et on devrait les réaliser tout au long du développement. Mais encore aujourd'hui, le test est souvent la portion congrue des projets informatique et parfois même, réduits à peu de chose. Oui le test coûte cher en temps et en argent. Bernard Homès, spécialiste reconnu du test logiciel en France, revient aux bases du test. Tout d'abord, l'auteur présente clairement les fondamentaux à connaître et particulièrement les processus organisant et structurant les tests ainsi que la dimension éthique qu'il faut avoir. Puis nous rentrons dans le cycle de vie et les techniques pour définir, exécuter les batteries de tests. Car le test vit et évolue avec le projet, les versions. L'ouvrage se conclut sur deux chapitres importants : les outils et l'automatisation et les modèles de tests. Un livre indispensable à lire et à relire.

Tester une application web

*Difficulté : ** - Editeur : Eni - Auteur : Emmanuel Itié - Prix : 39 €.*

Dans un développement web, ne pas faire de tests relève de la faute professionnelle ! Il faut savoir comment tester, connaître les points sensibles, avoir les bons réflexes. Vous trouverez ici une présentation des concepts de base de la qualité logicielle, des préconisations de développement d'une application Web, les tests unitaires et/ou d'homologation avec les anomalies les plus souvent observées, ainsi que les risques induits. Les exemples proposés sont en PHP mais ils s'appliquent à tout projet web.

JUnit

*Difficulté : ** - Editeur : Eni - Auteur : Benoît Gantaume - Prix : 39 €.*

Les tests unitaires sont l'Alpha et l'Oméga du test pour les développeurs. Parmi les frameworks / outils les plus connus, il y a JUnit pour le code Java. Quelle que soit la façon dont l'équipe de développement travaille, que vous soyez débutant ou expert, manager, développeur, architecte ou chef de projet, ce livre vous permettra d'appréhender les tests automatiques, de les insérer dans une logique de fabrication de logiciels et de les mettre en oeuvre efficacement. L'auteur propose une prise en main complète de Junit et de la manière de l'utiliser, de créer les scripts de tests. Une excellente initiation !

FUTUR

Quand la vie remplace le silicium

*Difficulté : ** - Editeur : Dunod*
Auteur : collectif - Prix : 19,90 €.

L'informatique, est-ce uniquement de la silice ? Les recherches fondamentales tendent à faire disparaître cette idée dans la notion de bio-informatique ou comment le vivant peut aider à changer l'informatique. Les auteurs expliquent d'abord comment des chercheurs s'efforcent de comprendre les mécanismes naturels de l'apprentissage et de l'évolution pour concevoir des vaisseaux spatiaux plus autonomes et des systèmes industriels plus sûrs. Les travaux sont assez nombreux et la 2e partie de l'ouvrage décrit quelques-unes de ces recherches et comment les chercheurs utilisent des entités d'ADN, de virus... Enfin, nous abordons le futur des architectures : le massivement parallèle, l'ordinateur quantique, etc.

KIMSUFI.COM

Le serveur qui me suffit !

	Kimsufi 2G	Kimsufi 16G	Kimsufi 24G
	14.99 € HT /M SOIT 17 93 € TTC /M	**39.99 € HT /M** SOIT 47.83 € TTC /M	**49.99 € HT /M** SOIT 59.79 € TTC /M
Frais de mise en service	**Offerts***	**Offerts***	**Offerts***
Disponibilités	1 heure	1 heure	1 heure
	Commander	Commander	Commander
Processeur			
Marque	Intel®	Intel®	Intel®
Modèle	**Celeron / Atom**	**i5**	**i7**
Fréquence	1.20+ GHz	4x 2.66+ GHz	4x 2(HT)x 2.66+
Architecture	64 bits	64 bits	64 bits
NIC	FastEthernet	GigaEthernet	GigaEthernet
Mémoire vive	2 Go	16 Go	24 Go
Disque dur	1 To	2 To	2 To
Backup FTP	100 Go	100 Go	100 Go
Connexion réseau			
Connexion	**100 Mbps**	**100 Mbps**	**100 Mbps**
Bande passante	Illimitée	Illimitée	Illimitée

* : les frais de mise en service sont offerts pour un paiement annuel. Sinon : 49.99€ HT (59.79€ TTC).

Systèmes d'exploitation

Linux/Webmin, Plesk ou brut
Windows®/TSE ou Plesk
BSD brut, pour les fans de BSD
Solaris brut, avec ZFS !

Utilisations

Grâce à un large choix de systèmes d'exploitations, vous avez le choix d'utiliser votre Kimsufi comme bon vous semble, sans pour autant avoir des compétences avancées en administration.

Hébergement

Votre serveur avec LAMP ou WAMP et Plesk à la demande !

Bureau à distance

Faites de votre serveur un PC de bureau avec vos documents, musiques, vidéos… sur Internet !

VPS

Partagez votre serveur physique en une multitude de machines virtuelles.

Distribution brute

Idéale pour les experts, les distributions brutes offrent une liberté totale.

Solution écologique

Les serveurs dédiés Kimsufi sont conçus dans le respect de l'environnement, et sont hébergés avec le soucis permanent d'économiser de l'énergie.

Hardware

Ils sont équipés de pièces dont la consommation électrique est réduite sans compromettre la puissance pour autant.

Datacentre

La principale difficulté d'un datacentre est d'évacuer la chaleur des milliers de serveurs en fonctionnement.Kimsufi a résolu ce problème en utilisant un système exclusif de refroidissement liquide/air qui consomme 90% d'énergie en moins qu'une climatisation classique.

Windows Azure

Microsoft

.NET, PHP, Java…
Nous avons chacun notre langage, mais pensons tous Windows Azure.

.Net, PHP, Java… Grâce à la plateforme applicative Windows Azure, vous déployez et exécutez vos applications dans le Cloud quel que soit le langage ou l'outil de développement. Et disposez d'un environnement de développement et de production à la demande, en quelques minutes.

Avec Windows Azure, le code vous appartient, pour le reste comptez sur nous.
C'est ça la puissance du Cloud.

Pensez Windows Azure.
Essayez la plateforme Windows Azure gratuitement*
sur www.windowsazure.fr

*1 mois d'utilisation offert. Offre valable jusqu'au 30 juin 2011 minuit.