PROGRAMMEZ

PROGRAMMEZ!

le magazine du développeur

www.programmez.com

Un code pour les gouverner tous !

100% mobile avec :

Xamarin, Nokia X, Windows Phone 8.1, Firefox OS

10-13-10 © mstay (tornade)

Java

L'essentiel de Devoxx France 2014

Déjeunez intelligent !

Osez le Brown Bag Lunch

DITES-LE AVEC DES API

Deezer, Skype, Lync, Yammer

L'informatique quantique

L'ultime frontière ?

Créer et gérer un projet Open Source

08-06-13 © whanwhanai (usine)

Coding4Fun

Raspberry Pi : plus fort que jamais

Mensuel n°175 - Juin 2014

M 04319 - 175 - F: 5,95 € - RD

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

PAESSLER
the network monitoring company

PRTG Network Monitor vous offre un serveur Syslog et d'interruption (Trap) SNMP

Un outil de supervision réseau complet

Connaissez-vous PRTG Network Monitor ? Ce logiciel de Paessler vous permet d'avoir une vue d'ensemble sur votre infrastructure informatique. Grâce à cette solution, rien ne vous échappe de l'historique d'activité des connexions, pas même des interruptions (Trap) SNMP. Vous gardez un œil sur vos messages Syslog et Traps et maitrisez votre réseau sans ajouts coûteux ou configurations ultérieures. **Cet outil est inclus dans la licence PRTG sans aucun frais supplémentaire.**

Même dans la version gratuite Freeware de 10 capteurs de PRTG, vous bénéficiez d'un puissant serveur Syslog et d'interruption (Trap) SNMP pour l'ensemble de votre réseau. Vous pouvez analyser en profondeur les connexions et les Traps et même surveiller votre serveur de messagerie ou votre site Web.

Gestionnaire Syslog/ Trap de PRTG : mode d'emploi

Les équipements de votre réseau envoient à PRTG des messages relatifs aux Syslogs ou aux interruptions SNMP. Les Syslogs sont néanmoins connus pour être plus descriptifs que les C3PO et cela peut engendrer un nombre important de messages. La solution PRTG est en mesure de recevoir plus de 10 000 messages par seconde et même de les filtrer en fonction de paramètres multiples, parmi lesquels les équipements, l'importance, les services ou les composants.

Vous configurez les paramètres de filtrage vous-même et choisissez le type de messages que PRTG devra surveiller et enregistrer pour analyse ultérieure. Les données reçues seront stockées dans la base de données interne de haute performance à PRTG et ainsi accessible à tout moment pour fournir des comptes rendus et rapports historiques d'activités.

Le système d'alerte et de notification complet est bien sûr mis par PRTG à votre disposition, ainsi que les différentes applications gratuites pour Android, iOS et Windows Phone.

Autres avantages de la supervision réseau avec PRTG

- Installation rapide et configuration intuitive
- Interface Web rapide et performante
- Affichage clair et détaillé de l'ensemble du réseau
- Notifications par email, SMS, Traps SNMP, exécution de programme...
- Mise à jour automatique et gestion de maintenance
- Assistance professionnelle
- Environ 200 types de capteurs pour la surveillance VoIP, des sites Web, des emails, des applications, des environnements virtuels ou des bases de données
- Supporte SNMP, WMI, Flow, Packet Sniffing...
- Propre base de données intégrée, optimisée pour les données de surveillance
- Réduction des coûts : votre réseau n'a plus de secret pour vous et vous n'achèterez que ce dont vous avez vraiment besoin.

Découvrez toutes les possibilités offertes par PRTG !

Téléchargez la version d'essai de PRTG Network Monitor de Paessler valable pendant 30 jours avec un nombre illimité de capteurs. Ce test entièrement gratuit ne vous engage en rien, mais vous apportera une expérience inoubliable de supervision réseau.

Découvrez par vous-même comment PRTG vous facilitera votre travail au quotidien !

EN SAVOIR PLUS:
www.paessler.fr/gratuit

Paessler AG
T : +49 (911) 9 37 75 - 0
F : +49 (911) 9 37 75 - 409
info@paessler.com
www.paessler.fr
Contact : Corinne Portenschlager

420626/FR/20140515

édito

Comment faire ?

Il y a certains mois où je me demande comment faire rentrer 100 pages dans 84 pages ! On peut réduire la taille des caractères et fournir la fonction zoom... Il faut faire des choix, décaler des articles d'un mois (toutes mes excuses aux auteurs). Simple et rapide.

Mais rapidement, on tombe dans le dur. C'est à ce moment-là que je passe des heures à regarder le sommaire, les articles qui ne rentrent pas. Je retourne dans tous les sens le problème. Parfois, c'est pour aérer mes neurones que je regarde un épisode de Game of Thrones ou de Battlestar Galactica. Et l'on se remet au travail...

Comme vous le constaterez, plusieurs rubriques habituelles ne sont pas présentes ou très peu (hacking, développeur du mois, matériel). Nous avons voulu privilégier l'actualité chaude : un grand retour sur Devoxx France 2014 et sur le dernier rebondissement dans le procès Google – Oracle. Sur ce dernier point, il est encore difficile de dire si les API vont être des victimes du dernier jugement...

Autre très gros dossier du mois, le développement mobile est à l'honneur (eh oui encore !). Nous avons voulu privilégier l'actualité de ces dernières semaines : Windows Phone 8.1, Nokia X (vous savez le téléphone Nokia — maintenant Microsoft — fonctionnant sous Android), les dernières versions de Firefox OS et un gros focus sur le développement multi-plateformes à partir d'un code unique, avec en ligne de mire les outils Xamarin qui agitent beaucoup la communauté depuis le mois d'avril !

Autre dossier, créer et gérer un projet open source. Car au-delà de l'intérêt d'un projet open source, il ne faut jamais oublier sa maintenance, son développement, sa disponibilité. Comme vous le verrez, ce n'est pas toujours simple, surtout quand on commence à chercher une licence ouverte.

Dans ce numéro, il sera beaucoup question d'API. Deezer, Skype, Yammer, vous n'aurez que l'embarra du choix ! Sans oublier, la nouvelle tendance déjeuner : les BBL. Fini le déjeuner bête et stupide : mangez et formez-vous !

Bon code !

François Tonic
Directeur de la publication & rédacteur en chef
ftonic@programmez.com

sommaire

À LIRE DANS LE PROCHAIN NUMÉRO
n° 176 en kiosque le 28 juin 2014

Cahier vacances **100 % Drupal**

MariaDB de A à Z

Coding à la plage :
Raspberry Pi, Gadgeteer, JavaScript, Google Maps, Nokia X, iOS, Windows Phone, Xamarin

Attention : numéro exceptionnel de 100 pages

8,5 %
le chiffre du système KitKat...

Vers un visa développeur en France ?

Gmail
bientôt une nouvelle interface ?

Miroir, miroir, quel est le langage le plus utilisé ?

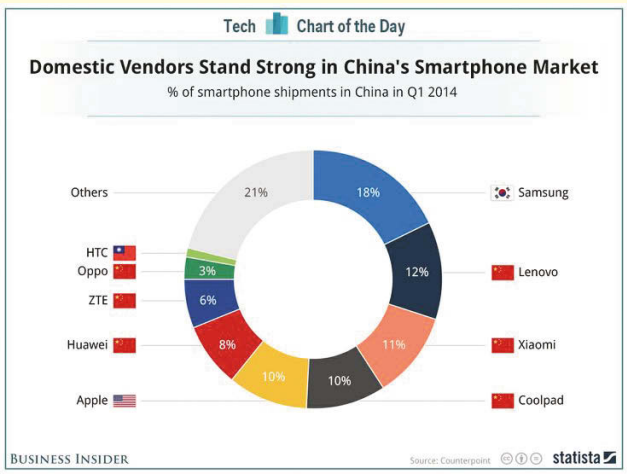
L'index TIOBE donne chaque mois les langages les plus utilisés par les développeurs. Il s'agit d'un indicateur basé sur les recherches web. Des % à manipuler avec précaution.

Mai 2014	Mai 2013	Tendance	Langage	%	Evolution (en %)
1	1	↓	C	16,926	-1,8
2	2	↓	Java	16,907	-0,01
3	3	↑	Objective-C	11,791	+1,36
4	4	↓	C++	5,986	-3,21
5	7	↓	(visual) basic	4,197	-0,46
6	5	↓	C#	3,745	-2,37
7	6	↓	PHP	3,386	-2,40
8	8	↓	Python	3,057	-1,26
9	11	↑	JavaScript	1,788	+0,25
10	9	↓	Perl	1,470	-0,81

Pas de changement pour le trio de tête : C, Java, Objective-C. Tous les langages, ou presque, subissent des baisses. Le marché est très segmenté. F# fait un bond de 25 places pour se stabiliser à 13e place.

Le marché chinois des smartphones

L'immense marché chinois est un enjeu pour les constructeurs de smartphones. Les acteurs locaux pèsent lourd : 50 % au 1er trimestre 2014. Samsung demeure 1er avec 18 %. Apple est 4e (10 %) avec Coolpad.



Les salaires 2014

Le cabinet Robert Half a dévoilé son étude sur les salaires dans le monde informatique. Le marché évolue beaucoup. Le cabinet estime une croissance de 5 % (postes à pourvoir). Parmi les postes clés à surveiller en 2014 : le lead développeur / chef de projet technique.
Le cabinet estime que les tendances sur les salaires sont plutôt à la stabilisation (idem pour les primes), des hausses sont toutefois possibles pour environ 20 % des entreprises.

poste	expérience	salaire brut annuel
architecte logiciel	5-15 ans	60 – 85 000 €
ingénieur développement	2 – 7 ans	40 – 55 000 €
	7-15 ans	55 – 65 000 €
ingénieur développement mobile	2 – 10 ans	40 – 55 000 €
DBA	0-3 ans	30 – 35 000 €
Expert BI	5-10 ans	50 – 80 000 €
chef de projet qualité logiciel / test manager	3 – 15 ans	48 – 70 000 €
intégrateur web / html	0-3 ans	25 – 35 000 €
	2-5 ans	35 – 45 000 €
	+ 5 ans	45 – 50 000 €
designer web	2-5 ans	35 – 45 000 €

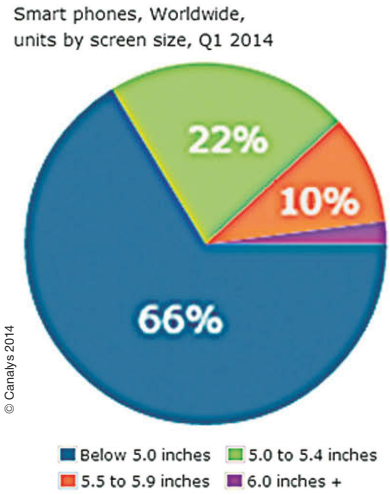
Les salaires des développeurs (niveau ingénieur) sont des fourchettes relativement hautes. Pour un développeur débutant, pas de surprises, nous serons toujours vers 30 – 33 000 € (île de France).

PANIQUE À BORD ?

La cotation de 37 entreprises du Cloud baisse (OK, – 30 %) et on commence déjà à paniquer, et à parler d'une bulle Cloud. Respirez !

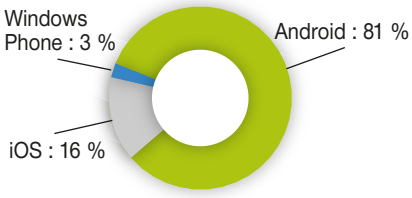
Marché smartphone : et les gagnants sont ?

Le cabinet d'analyse Canalys a sorti son étude sur le marché mondial des smartphones pour le 1er trimestre 2014.



Premier constat, les modèles ayant un écran inférieur à 5 pouces continuent de dominer le marché : 66 %. Les modèles entre 5 et 5,4 pouces suivent à 22 %. Les énormes modèles (6 pouces et +) sont anecdotiques : 2 %. Après, certains marchés peuvent être plus dynamiques sur les grandes tailles.

Côté système, c'est sans appel :



Si Windows Phone est dynamique sur certains marchés, au niveau mondial, le chemin à parcourir est encore énorme. Microsoft paie ici ses médiocres ventes sur plusieurs marchés clés tels que les États-Unis et le Japon.

Côté constructeur, Samsung est toujours le 1er vendeur avec 31 %, Apple se tient sans problème sur la 2e marche du podium avec 16 %.

MATÉRIAUX :

un nouveau composite résistant jusqu'à 1 200° ! C'est la trouvaille du français Herakles avec le Prophete.

WEBKIT

Un nouveau moteur JavaScript devrait s'installer dans les prochains mois dans le projet WebKit. FTJIT s'appuie sur LLVM pour le compilateur JIT. Mozilla et Google explorent aussi la piste LLVM.

24 juin

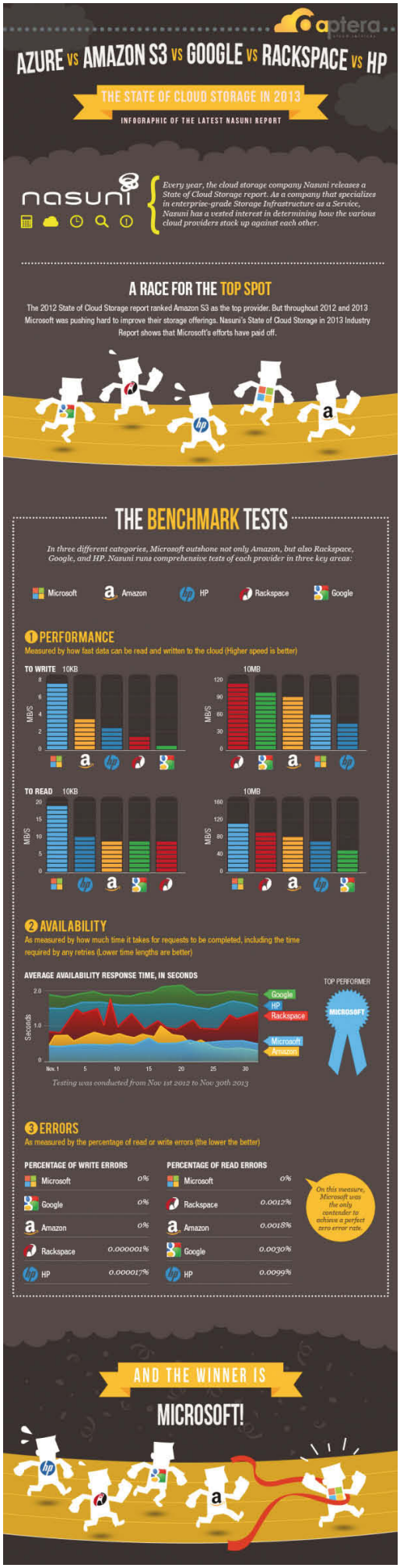
disponibilité de Windows Phone 8.1

XBOX ONE – *CHER :*

oui, mais dans sa version sans Kinect...

Basic a 50 ans !

Eh oui, le langage basic a 50 ans. Il fut créé en mai 1964 grâce aux travaux de Kemeny et Kurtz. Le but était de créer un langage pour pouvoir utiliser un ordinateur et faciliter l'accès aux étudiants. Toute l'idéologie du BASIC tient dans son nom : Beginner's All-purpose Symbolic Instruction Code ! Le Basic est devenu une référence dans les années 1980 avec Microsoft. Il était une fondation du Z80, du CPC 464, des premiers Mac ! Souvenez-vous du GW Basic (que de souvenirs) ou encore du GFA Basic (encore plus de souvenirs) ! Le basic avait aussi des détracteurs pour dire : c'est un langage qui contribue à écrire de mauvais programmes ! Les lieux de sa naissance, à l'université Dartmouth (USA), ont célébré l'événement (http://www.dartmouth.edu/basicfifty/).



Hackathon Fhacktory : l'application BlenderBQ gagne

24 heures ! Pas une heure de plus. 24 heures pour concevoir et développer une application. C'est le défi du hackaton Fhacktory qui s'est déroulé en mars dernier. Découverte du gagnant : BBQ.

BlenderBQ : interface vocale et gestuelle pour Blender

Notre application est divisée en deux parties : un serveur qui récupère et traite les commandes utilisateurs, et un client qui se charge de les traiter dans Blender.

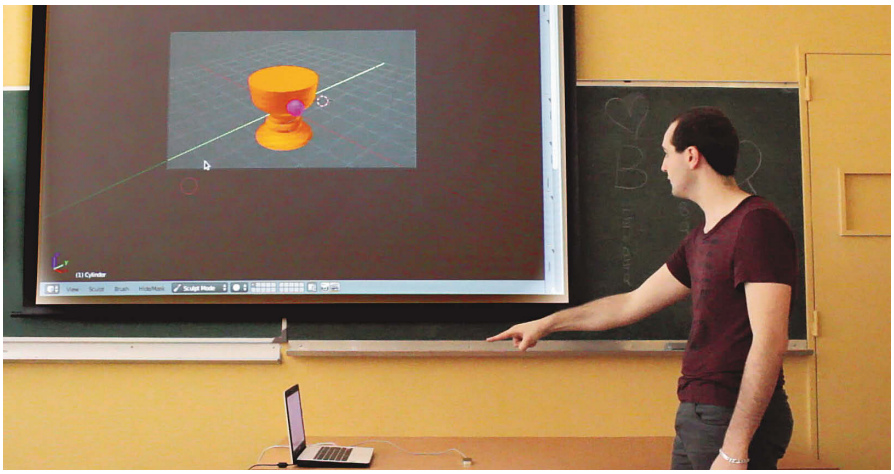
Le serveur

Dans un premier temps, le serveur va démarrer l'écoute d'événements du LeapMotion, ainsi que la pipeline de reconnaissance vocale (dont les fonctionnements sont détaillés plus loin). Celui-ci tourne sur Python 2.7, car l'API LeapMotion n'était pas compatible avec les versions suivantes de Python (3.3 par exemple). L'architecture client-serveur est tout ce qu'il y a de plus standard : le serveur va envoyer des commandes en fonction des entrées LeapMotion / reconnaissance vocale, via une simple connexion socket. Nous avons tout d'abord utilisé des socket UNIX par souci de performance des sockets TCP, mais il s'avère que la différence n'est plus très notable.

Le client

Le scripting de Blender est également fait en Python, mais contrairement au serveur, celui-ci utilise une version modifiée de Python 3, adaptée à son usage. C'est pourquoi nous avons utilisé des sockets relativement bas-niveau, plutôt que des systèmes de communication RPC ou avec un Manager, qui utilise Pickle, donc incompatible entre versions de Python. En ce qui concerne le contrôle de Blender, beaucoup d'obstacles ont dû être surmontés. Blender bloque dès le lancement d'un script, et ce jusqu'à ce que celui-ci se termine. Cependant, nous devions absolument maintenir l'écoute sur le serveur d'entrées. Nous avons donc utilisé le concept d'opérateurs modaux mis à disposition par Blender. Ces opérateurs fonctionnent en effectuant certains callbacks particuliers au fur et à mesure de l'utilisation de l'opérateur. Ceux-ci bloquent le fonctionnement de Blender, et donc nécessitaient un parallélisme pour récupérer les commandes du serveur. Malheureusement, Blender n'est pas thread-safe, ce qui nous a obligé à effectuer uniquement des opérations non-bloquantes lors de la récupération des commandes, ainsi que la gestion des paquets expirés ou redondants.

Ainsi, à chaque réception de commande, un ordre



est activé et un affichage se met en place pour changer les objets (« mesh ») de Blender. Notez que le système de coordonnées de Blender et celui du LeapMotion sont très différents, ce qui induit une conversion obligatoire des valeurs de coordonnées. D'abord à cause de l'échelle, mais aussi à cause de l'orientation des axes et de la différence entre coordonnées absolues et relatives.

Commande vocale

La chaîne de traitement

La commande vocale est basée sur l'utilisation de PocketSphinx (CMUSphinx), intégré via une pipeline GStreamer, et disponible à travers une librairie Python. Nous avons passé beaucoup de temps à comprendre le fonctionnement de la pipeline, et où nous nous placions dans la chaîne d'information vocale. Finalement, nous avons défini un corpus de mots simples correspondant à des commandes, et avons construit le dictionnaire et le modèle acoustique à partir de l'outil lmtool de CMUSphinx.

Commandes implémentées

Le dictionnaire est constitué de mots courts et très différents les uns des autres afin de rendre leur reconnaissance plus fiable. Un premier groupe de commandes ('above', 'right', 'camera', etc) permet de contrôler la vue, tandis que le second groupe ('object', 'pottery', 'paint') sert à passer d'un mode à un autre. Il est facile d'ajouter des commandes supplémentaires puisqu'il suffit de les insérer dans le dictionnaire puis de les faire correspondre à l'appel correspondant dans l'API Blender.

La grammaire gestuelle

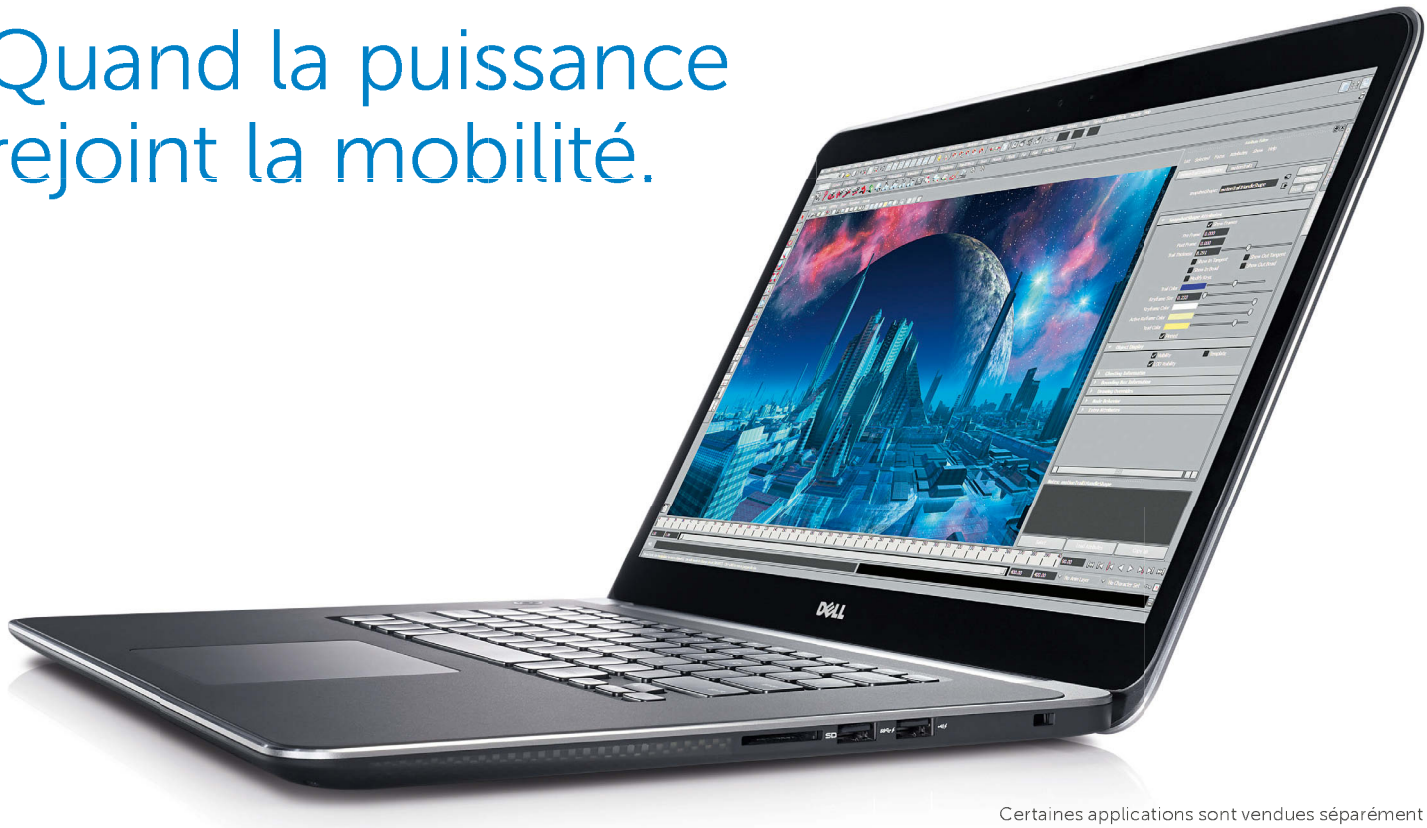
Au cours du temps, le Leap Motion envoie régulièrement des "frames", qui contiennent tout ce qu'il détecte. Ces frames comprennent le nombre de mains détectées et leur position en 3 dimensions, ainsi que le nombre de doigts visibles pour chaque main, ainsi que leur position. La fréquence d'envoi des frames est élevée : environ 20 frames sont envoyées par seconde. Cependant, cette fréquence peut varier selon les ressources disponibles, l'activité générale, et d'autres facteurs. À chaque réception d'une de ces frames, notre serveur analyse le mouvement des mains de l'utilisateur. Ceci est réalisé principalement par mémorisation des informations clé à chaque étape, afin d'en analyser l'évolution au cours du temps. En 24 heures, nous avons implémenté quatre gestes :

- Le geste 'grab' permet d'attraper l'objet, de le déplacer dans l'espace et éventuellement de le faire pivoter selon chaque axe.
- De manière complémentaire, le geste 'scale' permet d'agrandir ou rétrécir l'objet en mimant un étirement avec les deux mains.
- Un balayement de la main (geste 'swipe') permet, à la manière d'un potier, de mettre en rotation un objet.
- Enfin, le geste 'touch' permet, en pointant le doigt dans la direction souhaitée, de sculpter l'objet de manière très intuitive.

Merlin **Nimier-David**,
Jean-Marie **Commets**, Pierre **Turpin**,
Arthur **Gustave Monod**, Alin **Dicu** (Fhacktory)

Dell recommande Windows.

Nouvelle station de travail Dell Precision M3800. Quand la puissance rejoint la mobilité.



Certaines applications sont vendues séparément et peuvent varier d'un pays à l'autre

Découvrez la station de travail 39 cm (15,6") Dell Precision M3800.

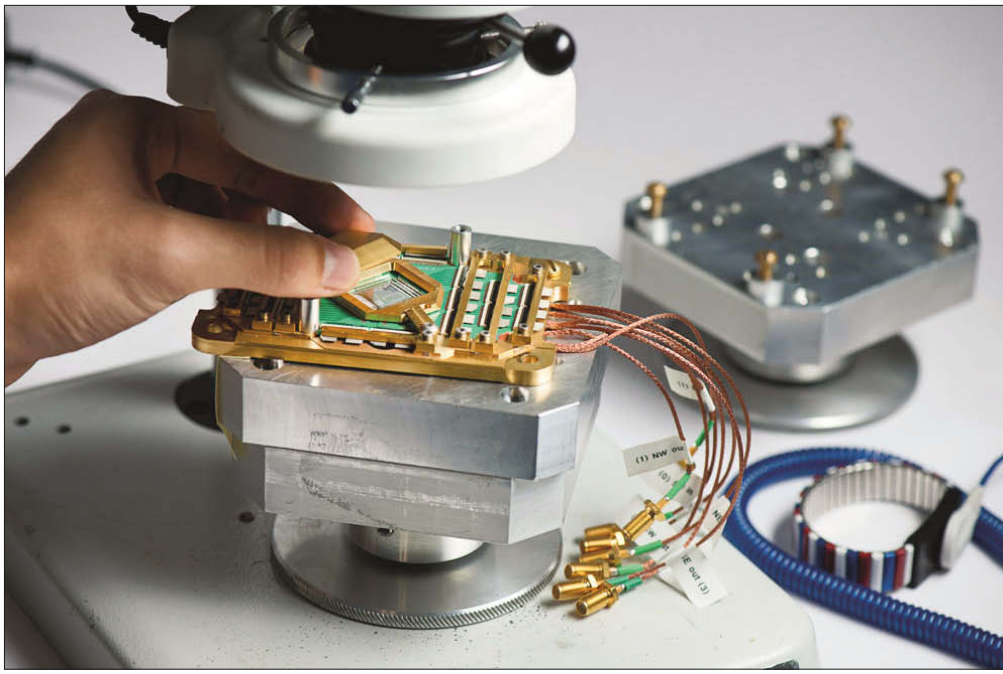
Une conception fine (18 mm) et légère (1,88 kg)[1] qui répond à vos envies et vous offre des performances adaptées à vos besoins. Exécutez vos logiciels professionnels les plus exigeants. Profitez de la puissance des processeurs Intel® core™ et des cartes graphiques NVIDIA® Quadro®. Repoussez les limites de votre imagination avec la station de travail Dell Precision M3800. Écran tactile QHD+ (3 200 x 1 800) disponible en option.

Rendez-vous sur **Dell.fr/pme** ou trouvez le partenaire Dell le plus proche sur **Dell.fr/findapartner**

La station de travail Dell Precision M3800 est disponible avec les processeurs Intel® Core™ i7. La station de travail Dell Precision M3800 est une marque commerciale de Dell Inc. Intel, le Logo Intel, Intel Inside, Intel Core et Core Inside sont des marques de commerce d'Intel Corporation aux États-Unis et dans d'autres pays. Dell S.A. Capital: 1 782 769 €, 1 Rond Point Benjamin Franklin - 34938 Montpellier Cedex 9 France. RCS Montpellier N° 351 528 229 - APE 4651 Z. [1] Poids de départ. Le poids varie en fonction des paramètres de configuration et de fabrication.

L'informatique quantique : l'ultime frontière ?

Un ordinateur, tout comme une casserole, est un système physique. Les physiciens diront que vous préparez ce système dans un état initial lorsque, dans une poêle, vous mettez de l'huile, cassez un oeuf, et la placez sur le feu. Il convient ensuite d'attendre quelques minutes que l'évolution physique s'opère, avant d'observer le système dans un état final intéressant: celui d'oeuf frit. Évidemment, tout ici est dicté par les lois de la physique; et même si le four à micro-ondes n'est pas la meilleure façon de cuire un oeuf, cette invention montre que d'autres principes physiques sont utilisables pour parvenir à ses fins.



Montage du bloc processeur de D-Wave Systems

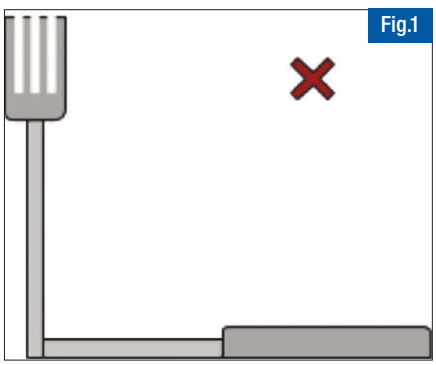
L'ordinateur comme un système physique

Ces banalités nous poussent à nous interroger. D'une part, nous savons qu'un ordinateur est un système physique qui, préparé dans un état initial (lorsque vous exécutez un programme *f* sur une entrée *x*), permet, après une courte évolution physique, d'observer un état final intéressant (le résultat du calcul, *f(x),* qui apparaît à l'écran). Mais d'autre part, nous savons que les lois de la physique sont parfois contre-intuitives, à la limite du magique: notamment celles de la physique quantique, découverte au début du 20ème siècle. Dès lors, n'est-il pas naturel de s'interroger: un système quantique ne permettrait-il pas d'effectuer un calcul plus rapidement? Pourrions-nous utiliser une partie de la "bizarrerie" de la physique quantique afin d'inventer des ordinateurs plus efficaces, ou plus sûrs? L'ordinateur quantique est-il le four à micro-ondes de l'informatique?
Pour explorer ces questions, tentons de comprendre en quoi la physique classique et la physique quantique sont fondamentalement différentes.

Le principe de superposition

En physique classique (Newtonnienne, celle qu'on apprend au lycée, celle que l'on observe aux échelles de la vie de tous les jours), les états des objets sont mutuellement exclusifs. Exemple: un couvert peut-être une fourchette, ou un couteau. Mais ça ne peut pas être à la fois une fourchette et un couteau.

Armés de ces même couverts, partons à l'assaut de l'un des phénomènes les plus étranges de la physique. Posez votre fourchette et votre couteau de façon à ce qu'ils se touchent à l'extrémité du manche et soient orthogonaux **(figure 1)**. En physique quantique, l'espace des états possibles est toute la table, c'est à dire tout le plan décrit par l'abscisse "être un couteau" et l'ordonnée "être une fourchette". Par exemple, prenons le point de coordonnées *(couteau=0.5, fourchette=0.5)*. Ce point est entre la fourchette et le couteau: il correspond à l'état d'un seul et unique couvert qui serait, curieusement, moitié fourchette, moitié couteau.

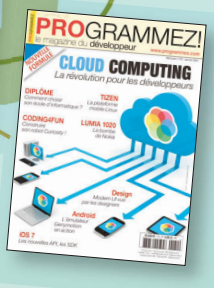


D'une façon générale, si, en physique classique, un système pouvait être dans un état *A* ou dans un état *B*, alors en physique quantique cet objet peut aussi se trouver dans un état superposé *a.A+b.B*, c'est-à-dire à la fois dans l'état *A*, avec une amplitude *a*, et dans l'état *B*, avec une amplitude *b*. C'est le *principe de superposition*. Bien que fort simple à expliquer (je viens de le faire en quatre lignes), le principe de superposition est terriblement difficile à accepter. L'esprit humain a besoin de se rattacher à ce qu'il connaît, à son expérience de tous les jours. Votre premier réflexe sera donc de songer à ces amplitudes comme à des probabilités. Autrement dit, vous aurez tendance à penser qu'un couvert qui serait dans l'état superposé *a.couteau+b.fourchette* est un couvert qui a une probabilité *a* d'être un couteau, et une probabilité *b* d'être une fourchette. Or ce n'est pas tout-à-fait le cas, pour deux raisons.
Le première de ces raisons, c'est que les amplitudes *a* et *b* ont le droit d'être des nombres négatifs. Autrement dit l'espace des états possibles est vraiment toute la table et non pas juste le quart de plan pointé par le bout de la fourchette et du couteau. Ce n'est clairement pas le cas pour les probabilités: a-t-on déjà vu un météorologue vous annoncer une "chance" *-0.8*

Une année pleine de technologies et de codes avec PROGRAMMEZ!

le magazine du développeur

www.programmez.com



réservée à la France Métropolitaine

Spécial étudiant 39€

1 an 11 numéros

1 an 11 numéros

49€

seulement (*)

2 ans 22 numéros

79€

seulement (*)

(*) Tarifs France métropolitaine

09-18-12 © DrAfter123 / istock

Toutes nos offres sur www.programmez.com

Oui, je m'abonne

ABONNEMENT retourner avec votre règlement à Programmez, 17, route des Boulangers 78926 Yvelines cedex 9

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

☐ ***Abonnement 2 ans au magazine :*** 79 € (au lieu de 130,9 €, prix au numéro)

☐ ***Abonnement spécial étudiant 1 ans au magazine :*** 39 €

Photocopie de la carte d'étudiant à joindre

Tarifs France métropolitaine

Offre spéciale : abonnement + clé USB Programmez!

☐ 1 an (11 numéros) + clé USB : 60 €

☐ 2 ans (22 numéros) + clé USB : 90 €

Clé USB contenant tous les numéros de Programmez! depuis le n°100, valeur : 29,90 €

Tarifs France métropolitaine

☐ 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 175

***Offre limitée**, valable jusqu'au 30 juin 2014*

qu'il va faire beau demain? Or, précisément, pour comprendre la vraie nature de ces amplitudes quantiques, il faut faire l'effort de s'imaginer un monde où une telle annonce serait monnaie courante. Dans un tel monde, imaginons qu'un ivrogne titube entre un point *A* et un point *B* en utilisant la règle suivante **(figure 2)**. S'il est en *A*, alors il se déplace en *B* avec "chance" *0.5*, et reste en *A* avec chance *0.5*. S'il est en B, par contre, alors il se déplace en *A* avec chance *0.5*, et reste en *B* avec chance *-0.5*.
Question: A supposer qu'il part de *A*, où se trouve l'ivrogne au bout de deux étapes? Pour y répondre, examinons les chances pour qu'il soit en *B*. Il y a pour cela deux scénarios possibles: 1/ avec chance *0.5*0.5* il est resté en *A* puis allé en *B*. 2/ avec chance *0.5*-0.5* il est allé en *B* puis est resté en *B*. Au total, ses chances d'être en *B* au bout de deux étapes sont de *0.5*0.5-0.5*0.5*, soit zéro. Autrement dit, l'ivrogne se trouve forcément en *A*. Moralité, là où les probabilités s'additionnent toujours, car elles sont forcément positives, les amplitudes quantiques, elles, parfois se soustraient, car elles peuvent être négatives. Lorsqu'elles se soustraient, les physiciens parlent de phénomène d'*interférence destructive*. Ils aiment aussi à parler, pour décrire ce phénomène, de *dualité onde-corpuscule*, car les ondes, comme des vagues, peuvent parfois additionner leur forces, mais aussi les soustraire: c'est le cas quand le haut d'une vague rencontre le creux d'une vague, laissant le niveau d'eau inchangé.

La deuxième de ces raisons, qui est liée à la première, c'est que l'état *a.A+b.B* est un état précis, entièrement déterminé, et non pas une distribution de probabilités. Autrement dit, le processus que suit notre ivrogne quantique n'est en fait pas une marche aléatoire.
D'ailleurs, cela avait été le cas, si à chaque étape l'ivrogne lançait une pièce de monnaie au ciel pour savoir si se déplacer ou non, alors au bout de deux étapes, nous l'aurions trouvé en *A* avec probabilité *0.5*, ou en *B* avec probabilité *0.5*, ce qui n'est pas le cas ici. Par conséquent, l'état au bout d'une étape, *0.5.A+0.5.B* n'est pas interprétable comme une distribution de probabilités. Il doit être compris comme: "l'ivrogne se trouve en *A* avec amplitude *0.5* et simultanément en *B* avec amplitude *0.5*".

Encore une fois, le principe de superposition est terriblement difficile à accepter. Votre deuxième réflexe sera donc de penser qu'il y a quelque chose de faux dans la physique quantique; et qu'on a jamais pu observer un tel ivrogne, ou encore qu'il doit exister une meilleure explication que celle-ci. Ce scepticisme est la réaction saine d'une personne rationnelle. Seulement voilà, les physiciens sont des personnes de nature

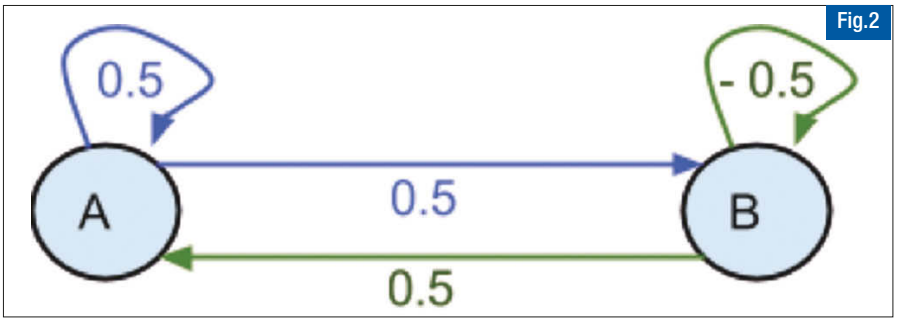


Fig.2

sceptique et rationnelle; cela fait 100 ans que la physique quantique existe, 100 ans que l'on observe dans les laboratoires des particules se comporter de manière étrange, à la manière de l'ivrogne quantique, 100 ans que l'on cherche une meilleure explication des phénomènes observés, et finalement 100 ans que le principe de superposition s'impose comme étant l'explication la plus simple, la plus claire, et en définitive la plus réaliste. Pour l'heure il nous faut donc admettre qu'à l'échelle des particules, le principe de superposition s'applique.

Le software quantique

Le principe de superposition est certes choquant, mais pour l'informatique, c'est une aubaine, car il nous offre un parallélisme gratuit et inespéré. Prenons l'exemple d'un informaticien qui souhaiterait, étant donné un nombre *x*, calculer son plus petit facteur premier *f(x)*. Autrement dit, l'ordinateur en tant que système physique doit emmener l'état initial *15* vers l'état final *3,* ou encore *77* vers *11*, etc. Un algorithme naïf consisterait à tenter de diviser *x* par *2*, puis par *3*, etc. jusqu'à lui trouver un diviseur. Cet algorithme est très lent: son temps de calcul moyen est proportionnel à *x*, et donc exponentiel en la taille (le nombre de chiffres) de *x*.
A l'heure actuelle, on ne connaît pas d'algorithme efficace pour résoudre ce problème, c'est-à-dire dont le temps de calcul serait polynomial en la taille de *x.* Les méthodes connues à ce jour reposent toutes, in fine, sur la force brute: je propose une solution y_1, elle ne marche pas, j'en propose une autre y_2... et ainsi de suite jusqu'à trouver la bonne, y_j égal f(x). Or ici le principe de superposition nous donne un nouvel outil algorithmique: il nous offre la possibilité de préparer, très rapidement, une variable dont l'état est la superposition de toutes les solutions possibles: $a_1.y_1+a_2.y_2+...+a_N.y_N$. Cet état ne prend pas plus de place en mémoire que n'en prenait $max(y_1,...,y_N)$: c'est une unique variable. L'ordinateur quantique a donc le pouvoir d'explorer toutes les solutions possibles d'un problème... en même temps!

Voilà qui est digne de faire bondir d'enthousiasme toute personne un tant soit peu sensibilisée aux difficultés de l'algorithmique. Cet enthousiasme est à tempérer, cependant. L'algorithmique quantique, hélas, n'est pas une discipline facile. Car certes, on peut préparer l'état initial $a_1.y_1+a_2.y_2+...+a_N.y_N$ à moindre coût. On peut même effectuer en temps constant une évolution quantique sur cette superposition, qui à chaque y_i concatènera *1* si y_i est le plus petit facteur de *x*, et *0* sinon. De cette façon, on obtient un pas de temps $a_1.y_10+a_2.y_20+...+a_j.y_j1+...+a_N.y_N0$. La solution $y_j=f(x)$ est donc présente dans la superposition, à portée de main. Seulement voilà, comment l'observer?

Car enfin, a-t-on jamais observé qu'un couvert en superposition pouvait être un couteau et une fourchette en même temps? A-t-ton jamais observé un ivrogne être en position *A* et en position *B*, en superposition? Non. En fait le principe de superposition n'est pas observable directement. Il est simplement la meilleure explication de ce qu'il se produit lors des étapes intermédiaires, d'évolution physique, avant l'observation finale. Pour clarifier ce point, retournons à notre ivrogne quantique. Le principe de superposition est la meilleure explication disponible du fait que, parti de *A*, si on l'observe après deux étapes de temps, l'ivrogne se trouvera forcément en position *A*.
Si, maintenant, nous modifions notre protocole expérimental et tentons d'observer l'ivrogne après une seule étape de temps, lorsqu'il est encore dans l'état *0.5.A+0.5.B*... nous ne pourrons guère observer la superposition elle-même, nous constateront simplement avec probabilité *0.5* que l'ivrogne est en *A*, ou avec probabilité *0.5* qu'il est en *B.*
Autrement dit, à l'heure de l'observation, les amplitudes quantiques redeviennent de simples probabilités, la distinction entre amplitudes positives et amplitudes négatives est perdue à tout jamais, l'état du système est modifié irrémédiablement. La physique quantique fournit, d'ailleurs, de très bons générateurs de nombres aléatoires, par le biais de l'observation de systèmes superposés.

Cette bonne nouvelle pour les salons de Poker en ligne, en est une mauvaise pour l'algorithmique quantique. Car si nous tentons d'observer l'état $a_1.y_10+a_2.y_20+...+a_j.y_j1+...+a_N.y_N0$, nous

Quelle interopérabilité entre mes différents fournisseurs Cloud ?

Avec Aruba Cloud,

vous avez l'assurance de ne pas être prisonnier d'un fournisseur. Nos services sont intégrés au **driver DeltaCloud** et compatibles **S3**. De plus, vous pouvez utiliser des formats standards d'images de machines virtuelles, **avec VHD et VMDK**, ainsi que des modèles personnalisés provenant éventuellement d'autres sources.

3 hyperviseurs

6 datacenters en Europe

APIs et connecteurs

70+ templates

Contrôle des coûts

Nous avons choisi Aruba Cloud car nous bénéficions d'un haut niveau de performance, à des coûts contrôlés et surtout car ils sont à dimension humaine, comme nous. Xavier Dufour - Directeur R&D - ITMP

Contactez-nous! 0810 710 300 www.arubacloud.fr

aruba CLOUD

Cloud Public | Cloud Privé | Cloud Hybride | Cloud Storage | Infogérance

MY COUNTRY. MY CLOUD.*

*MON PAYS, MON CLOUD.

n'obtiendrons y_j *1* qu'avec probabilité a_j. Et comme nous n'avions pas de raison de favoriser y_j à l'heure de la préparation initiale, cette probabilité sera exponentiellement faible.

Tout l'art de l'algorithmique quantique, donc, se résume à trouver des moyens de faire interférer destructivement les mauvaises solutions y_i pour *i* différent de *j*. De même que l'ivrogne quantique, au bout de deux étapes, ne se trouve jamais en position *B*, le programmeur quantique doit s'arranger pour que, au bout d'un certain nombre d'étapes, le système ne se trouve jamais dans l'état y_i pour *i* différent de *j*. C'est possible pour le problème de la factorisation. L'algorithme de factorisation quantique, découvert par Shor en 1994, permet de factoriser *x* en un temps polynomial en la taille de *x*.
C'est une application extraordinaire: toute la cryptographie à clé publique moderne repose sur la supposée difficulté du problème de la factorisation. Autrement dit, l'une des conséquences de l'avènement de l'ordinateur quantique sera de briser les protocoles de sécurité `https`, `ssl`, etc.

D'une façon plus générale, tout problème qui se résout par force brute en *n* coups classiquement, peut-être résolu par un algorithme quantique en racine carrée de *n* coups, via l'utilisation d'un autre algorithme quantique découvert par Grover en 1996. Encore une fois, il s'agit là d'une autre application extraordinaire, qui permet une accélération de la résolution d'une quantité énorme de problèmes.
On connaît en définitive très peu d'algorithmes quantiques. Mais à eux seuls, ces deux-là justifient déjà pleinement la course actuelle à la fabrication des ordinateurs quantiques.

Toutefois, si vous souhaitez vous essayer à programmer un ordinateur quantique, des langages et des simulateurs dédiés existent: téléchargez par exemple `Quipper`: http://www.mathstat.dal.ca/~selinger/quipper/.

La simulation quantique

Attendez une minute. Puisqu'un logiciel comme `Quipper` permet de simuler un ordinateur quantique, sur un ordinateur classique, pourquoi chercher à en construire un véritablement? Ne pourrait-on pas programmer l'algorithme de Shor sous `Quipper`, l'exécuter sur un ordinateur de bureau, et espérer ainsi casser la cryptographie à clé publique moderne?
Oui et non. Oui, car le simulateur effectuerait en effet des factorisations. Non, car il le ferait de façon inefficace (après tout, le contraire reviendrait à dire que nous connaissons un algorithme classique de factorisation efficace, ce qui n'est pas le cas). En fait, c'est `Quipper` lui-même qui est inefficace, au même titre que tout autre simulateur de physique quantique basé sur de la physique classique, d'ailleurs.
En effet, en physique classique le principe de superposition n'a pas cours: de même qu'un unique couvert ne peut pas être couteau et fourchette à la fois, une unique variable ne peut pas être y_1, y_2, ... y_N à la fois. Pour simuler classiquement la superposition quantique de toutes les solutions possibles, il ne suffit pas d'une variable, il en faut *N*.

Penchons-nous sur cette question de l'inefficacité des ordinateurs classiques lorsqu'il s'agit de simuler un système quantique, pour tenter d'évaluer l'ampleur des dégâts. Un bit, comme chacun le sait, est un système physique (classique) ayant deux états possibles: *"0"* ou *"1"*. De même un registre de *n* bits est un système physique ayant $N=2^n$ états possibles : *"0..00"*, *"0..01"*, ..., *"1..11"*. Un qubit, ou quantum bit, est un système physique quantique pour états possibles tous les : a_0.*"0"*+a_1.*"1"*. Et un registre de *n* qubits est un système physique quantique ayant pour états possibles tous les : $a_{0..00}$.*"0..00"*+$a_{0..1}$.*"0..01"*+...+$a_{1..1}$.*"1..11"*. Il en résulte que, si l'on peut donner une description classique complète d'un état de *n* qubits, il nous faut donner les 2^n nombres $a_{0..00}$, $a_{0..01}$, ..., $a_{1..11}$. Par conséquent, la simple description classique d'un système quantique de taille *n* prend déjà une taille exponentielle en *n*.
D'une façon générale, la difficulté à simuler un système quantique de taille *n* d'un ordinateur classique est exponentielle en *n*. D'ailleurs, l'une des prochaines grandes applications de l'ordinateur quantique sera de nous permettre de simuler efficacement la physique quantique.
Après tout, quoi de plus efficace, pour simuler un système quantique, qu'un autre système quantique? Initialement, c'est précisément cette application qui avait poussé Feynman, prix Nobel de physique, à proposer le concept d'ordinateur quantique en 1982.

Le Hardware quantique

A l'heure actuelle, les physiciens parviennent à construire de "petits" ordinateurs quantiques, constitués d'une douzaine de qubits. Ces ordinateurs permettent de valider la théorie: en factorisant 15 aujourd'hui, nous nous assurons que l'algorithme de Shor fonctionne bel et bien, et nous permettra de craquer les codes cryptographiques, demain.
Cependant, ces petits ordinateurs quantiques expérimentaux restent pour le moment parfaitement simulables classiquement. Pour faire la différence il faudra que les ordinateurs quantiques "deviennent grands": qu'ils aient davantage de qubits. Mais, puisque nous savons déjà fabriquer une douzaine de qubits, pourquoi ne pas en fabriquer une autre douzaine?

Si, au marché, vous achetez une barquette de douze oeufs, vous passerez relativement inaperçu. Si, par contre, vous lui achetez dix douzaines d'oeufs, alors il y a fort à parier qu'elle, ou bien le maraîcher débonnaire du stand d'en face, commenteront vos exploits. Cette observation amusée, ne brisera pas vos oeufs pour autant: pour peu que vous restiez concentré, ils arriveront à bon port.

A l'inverse, en physique quantique, l'observation, comme nous l'avons vu, modifie irréversiblement le système. Sous son influence les amplitudes quantiques redeviennent de simples probabilités. L'observation "classicise" le système. C'est pour cela, d'ailleurs, que la physique quantique est généralement associée aux petites échelles (particules, atomes).
En effet, tant que le système est petit (Ex. une douzaine de qubits), ses chances de passer inaperçu (c'est-à-dire de ne pas interagir avec un photon de passage, ou autre), sont fortes, et il se comporte alors quantiquement.
Réciproquement dès qu'un système est grand (Ex. dix douzaines de qubits), ses chances de passer inaperçu s'amoindrissent drastiquement: le système est "observé" par l'environnement avec lequel il interagit, et il se comporte alors classiquement.

Pour parvenir à réaliser un ordinateur quantique de grande taille, la physique expérimentale est donc confrontée à de multiples défis. Il lui faut parvenir à: 1/ isoler un grand nombre de qubits du reste du monde, en les ayant préparés dans un état initial, 2/ sans les observer, pouvoir gouverner leur dynamique, afin de leur appliquer un algorithme quantique, 3/ les observer au moment souhaité. Chacun de ces critères est difficile, mais réalisable, le problème étant d'obtenir un système physique les réunissant.

Transporter des centaines d'oeufs que chaque regard pourrait briser, et parvenir à les faire frire dans un four micro-onde en temps polynomial, c'est aussi ça, les frontières de l'informatique !

Pablo **Arrighi**
Maître de Conférences en Informatique à l'Université de Grenoble-Alpes. Spécialiste de l'Informatique quantique. Ses travaux de recherche, qui portent sur les limites physiques du calcul, et prônent une approche informaticienne de la physique théorique, sont reconnus internationalement.

Montée en charge linéaire et extrêmement performante

NCache™
100% .NET

Pour applications .NET et Java
(supporté sur Windows Azure et Amazon AWS)

TayzGrid™
100% Java

Les données en cache (via NCache), réduisent les accès coûteux en base, et permettent à vos applications de monter en puissance vers “extreme transaction processing” (XTP). TayzGrid est une implémentation native 100% Java de NCache.

Cache distribué en mémoire

- Extrêmement rapide et montée en charge linéaire avec 100% uptime
- Topologie en Miroir, Répliquée, Partitionnée et Cache Client
- NHibernate et Entity Framework cache niveau 2

Optimisation ASP.NET de Web Farms

- ASP.NET Session State cache
- ASP.NET View State cache
- ASP.NET Output Cache provider

Memcached Wrapper Fourni

Partage de données en mode Runtime

- Notifications d'événements puissants pour le partage de pub / sub données

Alachisoft
sales@alachisoft.com
US: +1 (925) 236 3830
Siège de l'entreprise

Télécharger un essai GRATUIT!
www.alachisoft.com

RedFabriQ
info@redfabriq.com
Tel: +33 1 40 16 07 89
Distributeur et intégrateur en France

Challenge hacking Europe – Afrique : HNC

L'association ACISSI a lancé son 6e challenge Hacknowledge Contest 2014-2015. Ce challenge de sécurité informatique se déroulera sur deux années. La première étape a eu lieu fin avril dernier en Belgique, d'autres suivront. Chaque « rencontre » se déroule sur 12 heures et met à défi plusieurs équipes. Plusieurs challenges sont proposés, sur différents thèmes. En tout, + 60 épreuves sont déjà prévues (forensic, matériel, réseau, systèmes industriels, etc.).
Pour en savoir plus : http://www.hacknowledge-contest.org

juin

CMSDay : une journée dédiée aux CMS

Pour cette édition 2014 se tenant le 17 juin, 32 conférences sont au programme du 1er événement européen dédié aux CMS open source et aux stratégies digitales.
20 CMS, 4 coorganisateurs (Smile, Cybercité, Meanings et SmartFocus) et des grandes entreprises utilisatrices interviendront sur les grandes tendances de la gestion de contenus grâce à :

- **Des tables rondes « Solutions »**, où débattront les CMS sur des sujets comme : l'omnicanal, le Time To Market, le secteur public, les business models open source, la personnalisation, l'interaction et la production de contenu.
- **Des tables rondes et ateliers « Stratégies digitales »**, animés par les co-organisateurs avec des témoignages d'utilisateurs (Air France, Bouygues, Carrefour, XL Airways,...) : design, référencement, multisites, rich média, réservation en ligne...
- **Des ateliers,** où les CMS présenteront leurs solutions sous différents angles : cloud, big data, portail, expérience utilisateurs...

EclipseCon France 2014

Les 18 et 19 juin, la conférence Eclipse France se déroulera à Toulouse. L'appel aux contributions s'est déroulé en avril et l'agenda sera dévoilé courant mai. Plusieurs sessions sont déjà prévues : CoffeeScript, Mathus, débug et tracing tool. EclipseCon France est la conférence officielle de la fondation.
Site : https://www.eclipsecon.org

PHP Tour Lyon 2014

Cette année, l'AFUP arrête la caravane du PHP Tour à Lyon les 23 et 24 juin prochains. Sur 2 jours, vous pourrez rencontrer les acteurs de communautés PHP en France et quelques invités surprises. De nombreuses sessions techniques seront proposées. Le programme sera disponible très bientôt. Site :
http://afup.org/pages/phptourlyon2014/index.php

Xamarin Turns 3 ! Vive la pizza party !

Vous voulez en savoir plus sur Xamarin ? Savoir comment développer sur iOS et Android avec C# ? L'événement Xamarin Turns revient le 24 juin à partir de 19 h 30 ! Pour en savoir plus : http://www.meetup.com/Paris-XAMARIN-Cross-platform-Mobile-Meetup/

Salesforce1 World Tour Paris : 26 juin

Salesforce organise son événement de l'année : Salesforce1 World Tour. Le grand patron ouvrira l'événement. De nombreux ateliers, des sessions techniques, de démos autour des outils et solutions Salesforce, notamment pour les développeurs, se dérouleront toute la journée !

juillet

Forum Terratec : tout pour le HPC

Le monde du HPC a son événement : le forum Terratec se déroulant à l'école Polytechnique près de Paris les 1er et 2 juillet. Plus de 1000 participants, des partenaires et sponsors de prestige (Bull, HP, Intel, CEA, Dell, AMD...), de nombreuses sessions techniques seront proposés toute la journée.
Site : http://www.teratec.eu/index.html

bientôt

DroidCon 2014 : initialement prévue en juin, la conférence DroidCon Paris se tiendra en automne prochain. Informations à venir !

Green Code Lab Challenge 2014 : l'événement de l'éco-informatique aura lieu du 26 au 28 novembre. Comment faire un code et un logiciel écoresponsable ? Toutes les réponses au Green Code Lab ! site : http://www.greencodelab-challenge.org/GCL2014/

conférences des JUG

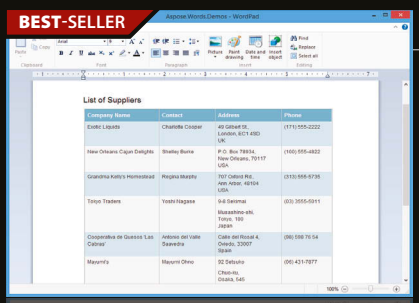
Toulouse Jug : soirée JUG sur Java EE avec Antonio Goncalves, dans les locaux de l'Epitech http://toulousejug.org
Jug Lyon : high scalable backend avec Thrift, MongoDB et non blocking IO – 17 juin. Site : http://www.lyonjug.org
Riviera JUG – soirée Java 8 – 18 juin Site : http://rivierajug.org
Paris JUG : soirée cloud computing le 10 juin !

Casual Game Cup 2014

Grand concours ouvert à tous les développeurs. En participant à la Casual Games Cup, vous avez 5 mois (de mai à septembre 2014) pour créer votre jeu web, Facebook ou mobile et le poster sur www.casualgamescup.com. Ce concours vise à encourager la création de jeux casual et la découverte de nouveaux talents. Prêt(e) à relever le défi ? Alors venez montrer votre savoir-faire et votre créativité et tentez de remporter des dotations attractives !
Pour en savoir plus : www.casualgamescup.com

Une conférence, un événement pour développeur ?
Envoyez-nous votre agenda : redaction@programmez.com

BEST-SELLER



Aspose.Words for .NET | à partir de € 718

ASPOSE Your File Format APIs

Lisez, modifiez et écrivez des documents Word sans Microsoft Word.

- Création de documents, manipulation du contenu/formatage, puissante capacité de fusion de courrier et exportation en DOC/HTML
- Accès détaillé à tous les éléments d'un document par programmation
- Support les formats de fichiers: DOC, DOCX, WordprocessingML, RTF, HTML, OOXML, OpenDocument, PDF, XPS, EMF et EPUB

BEST-SELLER

DevExpress DXperience | à partir de € 1 077

DevExpress

Tous les outils DevExpress ASP.NET, WinForms, Silverlight, WPF et IDE Productivity en un.

- Abonnement de 12 mois pour tous les produits et mises à jour DevExpress et accès aux versions bêta en développement actif
- Modèles et thèmes d'application intégrés exceptionnels
- Support de nouvelle vue Windows 8 UI et panneaux ancrables tactiles
- Support interface codée pour test environnement utilisateur

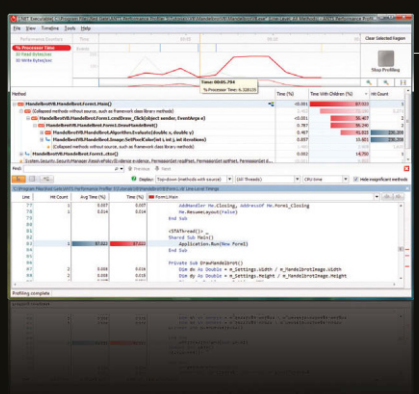
BEST-SELLER

oXygen XML Editor Professional | à partir de € 350

<oXygen/> XML Editor

Éditeur XML multiplateforme supportant la plupart des technologies XML.

- Distribuez/actualisez facilement des plug-in/frameworks pour Oxygen
- Simplifiez la configuration des projets XML avec Master Files
- Affichage des modifications et commentaires de révision dans légendes
- Support d'édition visuelle convivial pour DocBook, DITA, TEI, XHTML
- Validez les documents XML avec les schémas XML, Relax NG, DTD, NVDL et Schematron



Red Gate .NET Developer Bundle | à partir de € 760

redgate®

Éradiquez et corrigez le code lent, identifiez le code .NET bogué et comprenez les raisons.

- Bénéficiez d'une vue d'ensemble des performances de vos applications et identifiez les goulots d'étranglement, dans le code ou la base de données
- Trouvez rapidement les fuites de mémoire et optimisez la mémoire de vos codes C# et VB.NET
- Comprenez et déboguez le code de tiers, frameworks, composants et bibliothèques inclus
- Standardisez la gestion des performances de votre équipe de développement

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

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

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

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

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

VISA MasterCard AMERICAN EXPRESS

Les API de la discorde : Oracle gagne en appel contre Google !

Souvenez-vous, Oracle avait attaqué Google sur la présence de quelques lignes de code dans le cœur d'Android. Oracle avait finalement perdu. Mais en appel, Oracle a obtenu une précieuse victoire. Le dernier jugement renverse totalement la première décision...

Le premier jugement n'avait pas tranché le fond du débat. L'appel l'a fait. Le fait le plus notable à retenir est que la cour d'appel fédérale américaine reconnaît à Oracle que les interfaces de programmation Java (= les API) sont soumises à la loi de protection du droit d'auteur (loi valable aux Etats-Unis). Deux faits sont à considérer immédiatement : les API sont-elles en danger ? Oracle peut continuer la procédure judiciaire et demander quelques milliards à Google ! L'affaire sera sans doute débattue prochainement au tribunal de San Francisco sur l'usage « fair use » (ou non) des API par Google. Pour Google, les API ne sont pas concernées par le copyright américain. Il faut préserver l'accès libre car ce sont des éléments vitaux pour les développeurs.

L'API est-elle un livre ?

Reconnaître la soumission des API au droit d'auteur poserait alors un problème pour l'innovation, l'interopérabilité... D'autres se positionnent pour la reconnaissance du droit d'auteur. Mais les juges estiment que l'organisation, la structure, le code des API ont droit à la protection par le droit d'auteur, d'où la nouvelle décision et la reconnaissance du droit d'auteur pour les API et la violation de ce droit dans 37 packages Java... La plainte d'Oracle concerne 3 % du code des API d'Android ! Oracle a découvert que Google avait utilisé 9 lignes de code copiées de Java. Sans surprise, Google s'est dit déçu et préoccupé par les conséquences que cette décision pourrait avoir sur l'innovation, le travail des développeurs, avec de possibles domages pour l'industrie. Le logiciel est déjà soumis aux droits d'auteurs et à diverses protections.

Comme l'écrivait Ars Technica, le fait de pouvoir copyrighter des méthodes d'accès, le code d'API limitera la créativité et donnera surtout trop de contrôle aux détenteurs de ces droits d'auteur, même sur des services et applications déjà disponibles et installés. On imagine les risques de procès et de royalties demandés... Avec raison, Ars Technica évoque le livre. Celui-ci est protégé par le droit d'auteur mais le livre peut être lu où on veut et par n'importe quelle personne. Cela ne viendrait à personne d'interdire ou de limiter la lecture à des endroits précis ou d'autoriser uniquement telle personne.

> *« Soumettre les API au droit d'auteur remet en cause l'intérêt même de l'API : faciliter l'intégration, l'interopérabilité entre applications »*

L'autre enjeu est de savoir si Google a utilisé le code Java selon la pratique du fair use (autorisée aux Etats-Unis selon des pratiques précises) ou si Google a outrepassé la pratique en toute connaissance. Ensuite, où s'arrête le fair use et l'infraction ?

Florian Mueller (FOSS Patents), spécialiste des brevets et des problèmes légaux dans les technologies, est revenu sur l'affaire Google – Oracle. Il n'est pas surpris et dit qu'il ne faut pas être surpris par cette nouvelle décision. Pour Florian, les développeurs et utilisateurs ne doivent pas s'inquiéter. Le jugement rappelle les pratiques de la loi. Le fait nouveau est que les API peuvent être soumisses au droit d'auteur. Il pense aussi que Oracle et Google pourront s'entendre si l'appel se confirme. Cette affaire n'est pas comparable à celle opposant Apple et Samsung qui ne concerne pas le code proprement dit.

Florian rappelle, avec raison, que la syntaxe et le langage Java ne sont pas concernés. Ils sont libres d'usage. Le conflit porte uniquement sur les API et non le langage.

Suites et conséquences

Comme dit plus haut, un nouveau procès aura certainement lieu (2015 ?). Google peut potentiellement demander à la cour suprême américaine de se prononcer sur cette affaire. Mais un accord peut aussi être trouvé entre Google et Oracle.

Quelle conséquence pour les API en général ? C'est sans doute la véritable conséquence de cette décision, si elle est confirmée. Car, reconnaître le droit d'auteur sur l'API pose la question de leur utilisation, des conditions d'accès. Avec l'explosion des API et de leur usage, cette décision peut potentiellement freiner l'extension de l'API, sans oublier les possibles procès.

Pas de panique pour le moment, notamment en Europe, où la législation sur le droit d'auteur dans les technologies n'est pas identique. Cependant, il faudra suivre de près cette affaire.

Nous reviendrons très prochainement sur cette affaire.

François **Tonic**

ikoula
HÉBERGEUR CLOUD

LE CLOUD GAULOIS, UNE RÉALITÉ ! VENEZ TESTER SA PUISSANCE

EXPRESS HOSTING

Cloud Public
Serveur Virtuel
Serveur Dédié
Nom de domaine
Hébergement Web

sales@ikoula.com
01 84 01 02 50
express.ikoula.com

ENTERPRISE SERVICES

Cloud Privé
Infogérance
PRA/PCA
Haute disponibilité
Datacenter

sales-ies@ikoula.com
01 78 76 35 50
ies.ikoula.com

EX10

Cloud Hybride
Exchange
Lync
Sharepoint
Plateforme Collaborative

sales@ex10.biz
01 84 01 02 53
www.ex10.biz

Devoxx 2014 : du Java, de la technologie et du code

Les 16, 17 et 18 avril derniers a eu lieu à l'hôtel Marriott la 3e édition de Devoxx France, organisée par le Paris JUG ! Devoxx, c'est bien sûr le rendez-vous incontournable de tous les passionnés de développement et de technologie, javaneros, et geeks dans l'âme... On y retrouve bien sûr de nombreux experts en la matière, java champions et autres stars. C'est également l'occasion de revoir ses (anciens) collègues et confrères, au détour d'un couloir et de faire une pause dans son quotidien de développeur ; sortir le nez du code et prendre un peu de recul sur notre métier, sur l'ère numérique qui s'ouvre à nous et entre-apercevoir le futur qui nous attend...

D.R.

Du très lourd !

Au programme de Devoxx cette année, des conférences bien sûr, des university, labs, quickies, du live coding... Les grands sujets du moment ont été couverts, notamment avec le Chinese Democracy du monde java, le Duke Nukem Forever de chez Oracle : je veux bien sûr parler de la sortie très attendue de la version 8 de Java SE !

Mais également, pas mal de Devops (avec de l'outillage : Docker, Chef & Puppet...), du Big data toujours (Hadoop encore une fois à l'honneur), un peu de NoSQL avec Cassandra, et également de nombreuses conférences cette année sur le sujet des objets connectés (Google glass, S.A.R.A.H., NAO et autres drones volants). Mais j'aimerais revenir sur une autre facette de l'événement : mettons de côté la technique, les frameworks et les robots, pour prendre un peu de hauteur sur nous-même et réfléchir à ce que devient notre métier, celui de développeur, à notre condition bien sûr, à la passion qui nous anime, mais aussi au secteur du numérique qui nous fait vivre et qui paie notre steak quotidien...

La révolution numérique

Soyez heureux ! Notre génération est et sera témoin du passage à une nouvelle ère : l'ère numérique. Il s'agit bien là d'un tournant dans notre histoire, au même titre que l'ont été les révolutions agricoles et la révolution industrielle en leur temps. Pour Gilles Babinet, "la révolution numérique aura [d'ailleurs] plus d'impact que la première révolution industrielle" !

L'informatique et l'avènement d'Internet sont en train de changer radicalement nos modes de vies : les services en ligne, les objets connectés, les réseaux sociaux, les mobiles ont bouleversé nos habitudes, offrant de nouvelles perspectives pour notre société. Mais comment allons-nous appréhender cette révolution ?

L'industrie du numérique en France

L'industrie du numérique est un point fort pour la France ; c'est le constat dont nous pouvons nous vanter et qu'a fait Guy Mamou-Mani lors de sa keynote :

Les exemples d'initiatives françaises innovantes (ou à l'origine d'un de nos compatriotes) ne manquent pas (netvibes, deezer, daily motion, pour n'en citer que quelques-unes)

· Les Ecoles/Universités françaises constituent un vivier de talents pour les entreprises étrangères, qui s'arrachent les étudiants.

Nous disposons des meilleurs talents dans le domaine du numérique et pourtant, la France donne l'impression de surfer avec beaucoup de difficultés sur la vague du numérique, et peine à en tirer de réels bénéficies pour notre économie.

C'est le constat dont nous a fait part Tariq Krim, vice-président écosystème et innovation du Conseil national du numérique (CNN) lors de son intervention : la France est un pays visionnaire qui, tout au long de son histoire, a su faire preuve d'inventivité et d'innovation. Pour autant, l'excellence technique n'y est pas suffisamment reconnue et valorisée.

Espérons que le message de Tariq, qui a remis en mars au gouvernement un rapport comprenant 6 grandes mesures pour valoriser le talent des développeurs français, ne restera pas lettre morte et sera mis en œuvre le plus rapidement et fidèlement possible.


D.R.


D.R.

La formation 3.0

Paradoxe à la française : les entreprises manquent cruellement de personnel qualifié dans le secteur informatique, et parallèlement à cela, on compte près de 40000 "informaticiens" qui ne trouvent pas d'emploi !

Quel est le problème ? Selon Guy Mamou-Mani, il s'agit d'une inadéquation entre les compétences des postulants et les besoins des entreprises. La formation continue des salariés serait en cause, et les torts partagés. D'une part, la grande majorité des entreprises gèrent assez mal l'évolution de leurs développeurs au sein de l'entreprise. D'autre part, en tant que développeurs, nous ne pouvons rester "à jour" sans un minimum d'auto-formation, tout au long de notre carrière. Sans cet effort partagé de développement des compétences, un développeur devient très vite "obsolète" et perd de sa valeur pour l'entreprise et plus globalement, sur le marché de l'emploi.

Comme l'ont souligné Dominique Van Deth et Danny Gorris durant leur keynote "900 000 emplois IT à pourvoir d'ici 2015 en Europe. Comment susciter les vocations", le volume de diplômés en France reste en dessous des besoins des entreprises.

Le système éducatif actuel n'est clairement pas favorable à la découverte de vocation dans le développement logiciel chez les jeunes et ces vocations ne peuvent être que tardives.

Les enseignements en la matière sont totalement absents des programmes, seuls quelques lycéens de terminal S suivant l'enseignement de spécialité "Informatique et Sciences du Numérique" (ISN) pourront l'aborder.

Comment dans ces conditions se découvrir et développer un intérêt pour les métiers du numérique ?

Heureusement, parallèlement au système éducatif traditionnel, de nouvelles initiatives de formations ont vu le jour, tentant de combler les lacunes du système. Pour n'en citer que deux : l'Ecole 42 (enseignement gratuit et ouvert à "tous") représentée par Kwam Yamgnane, ainsi que simplon.co, la "Fabrique de codeurs entrepreneurs à Montreuil", présenté par Henri Fournet et Frédéric Bardeau (une formation orientée pratique sur une durée de 6 mois, également ouverte à tous).

Quoi que puissent en penser leurs détracteurs, ces initiatives ont le mérite d'exister ; leurs objectifs sont louables et bien fondés dans le contexte actuel. Celles-ci proposent un modèle alternatif venant tant bien que mal soutenir un marché de l'emploi sous-alimenté.

D.R.

« La meilleure façon de prévoir l'avenir, c'est de l'inventer »
(Alan Kay)

L'apprentissage par la musique

Une autre initiative mérite elle aussi d'être saluée : celle de Geert Bevin, pour son intervention intitulée "Programmer avec Emotion". Accompagné de son instrument à la fois intriguant et futuriste (une Eigenharp !), il nous a expliqué dans quelle mesure la créativité qui est en nous, a toute sa place dans le développement logiciel et dans l'utilisation qu'on en fait.

« Salut l'harpiste ! » et merci pour cet harmonieux mélange d'émotion, de technologie et de réflexions sur nos métiers.

Bruno **Doolaeghe**
Ingénieur-Concepteur chez Soat

Docker : virtualisation, DevOps, Cloud et micro-services

Avant d'introduire Docker (docker.io), je voudrais commencer par citer Hadi Hariri de la société JetBrains. Pendant sa présentation au sujet de REST et des API s'appuyant sur HTTP, lors de Devoxx France 2014, il a parlé des effets de mode dans l'univers des développeurs.

Alors qu'il soulignait que REST était un terme relativement récent, il rappelait que le protocole HTTP tel que nous l'utilisons pour REST a été décrit dans des RFC qui datent de quinze ans. Pour nous taquiner en conclusion de sa présentation, il nous a dit à peu près ceci : «Vous aimez les buzzwords, aujourd'hui vous avez RESTful à la bouche, l'an prochain vous parlerez de micro-services, mais n'oubliez pas que ça reste tout simplement du HTTP. »
Docker est intéressant sur ce point, car c'est précisément un projet à la croisée des mondes et des modes. Il reflète bien le contenu de ce Devoxx France mouture 2014, et la meilleure définition que l'on puisse lui donner, c'est qu'il s'agit d'un outil de virtualisation DevOps qui nous vient du Cloud, et qui facilite grandement la mise en place de micro-services standardisés.
Devant autant de buzzwords, quelques explications ne seront pas de trop pour se remettre dans le contexte.

Les gros mots du moment

DevOps

Le DevOps concerne aussi bien les développeurs que les opérationnels. Ce courant très à la mode en ce moment, bâtit un véritable pont entre les équipes de réalisation et les équipes d'exploitation.
Dans le cadre de Devoxx France 2014, beaucoup de conférences ont été données sur le thème du DevOps, notamment au sujet des tests de performance et au sujet de Docker. Mais le DevOps ne s'arrête pas aux aspects techniques, il concerne également tout ce qui a trait aux méthodes de travail, en particulier sur les concepts de livraison continue, et de pipelines de déploiement que l'on retrouve régulièrement dans les méthodes agiles.

Cloud

Le sujet chaud de 2013 était de retour cette année. Dans la révolution as a Service, il semble que la partie infrastructure (IaaS) ait été bien digérée puisque les orateurs ont surtout insisté cette année sur le modèle des plateformes (PaaS). Lors des conférences, on s'est donc éloigné de l'infrastructure, de la location de temps de calcul, de RAM et d'I/O, avec l'archi-connu Amazon AWS, pour aborder plus en profondeur l'achat d'une ferme de conteneurs Java EE ou d'une base de données, et ainsi de suite.

Micro-services

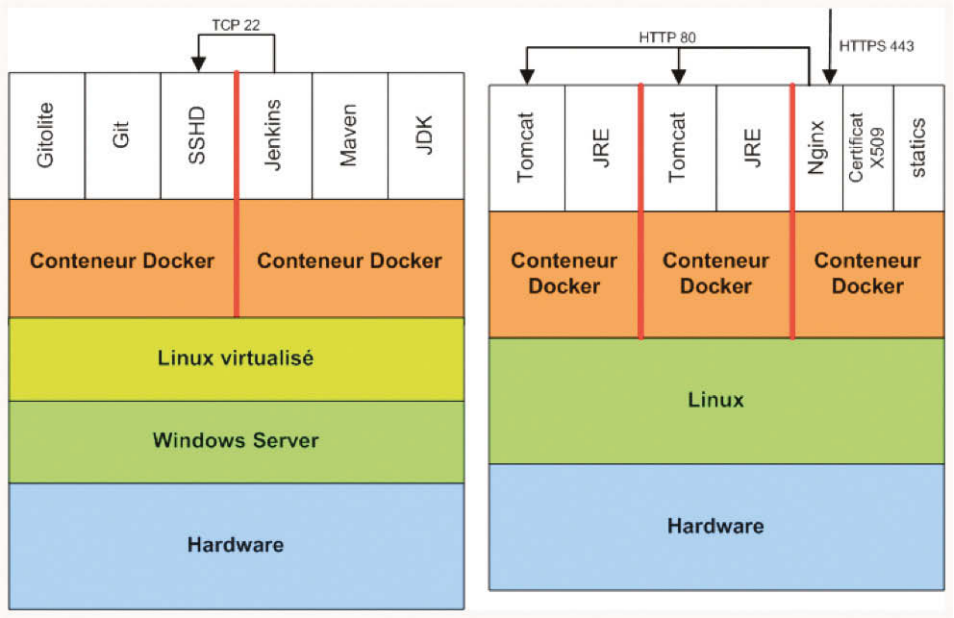
Prenez HTTP, appliquez les bonnes pratiques REST, vous êtes sûrs de n'avoir rien oublié ? Le SOA revient à la charge avec son nouveau nom de code : les micro-services. Plutôt que d'avoir un tutti frutti d'URLs pointant vers une API REST monolithique, la notion de micro-service veut que chaque fonctionnalité soit hébergée de manière indépendante. Une fonctionnalité, une URL, une API, un livrable : un micro-service.

Trois problèmes, une solution : Docker

Changeons de rôle maintenant, et au lieu de penser comme un développeur, mettons-nous à la place d'un exploitant, dont le rôle est de fournir tout ce qu'il faut pour héberger nos applications, aussi bien en environnement d'intégration que de qualification ou de production. Trois problèmes se posent principalement à nous, et ce sont justement les trois problèmes que Docker cherche à résoudre en s'appuyant sur des technologies ou des pratiques existantes.

Problème n°1 : gérer les ressources

C'est un véritable problème et pas des moindres. Car les ressources, ce sont des coûts qui peuvent très vite s'envoler dès lors que l'on commence à déployer des architectures complexes avec de la redondance et de la répartition de charge. Il faut y ajouter un impératif de sécurité qui empêche d'avoir une approche approximative du partage des ressources.
L'une des façons de faire consiste à appliquer la virtualisation, la gestion par quota et le cloisonnement. Ces techniques ne sont pas nouvelles. On les retrouve à bas niveau dans la gestion de ressources réseau (la QoS, le NAT...), des ressources disque (chroot), etc.
Et si plus récemment les machines sont devenues suffisamment puissantes pour qu'on applique ces principes à l'échelle de l'OS entier, avec plusieurs machines virtuelles par machine physique, cette méthode s'avère souvent inadaptée et trop coûteuse.
En effet, d'un côté toutes les ressources consommées par l'OS sont démultipliées par le nombre de machines virtuelles, de l'autre, si l'on met deux serveurs sur la même machine virtuelle, on perd une partie du cloisonnement. Docker change ce paradigme en appliquant la virtualisation plus finement. Au lieu de virtuali-



Exemple de mise en oeuvre de conteneurs Docker

ser l'OS, ce sont des groupes de processus qui sont exécutés de manière isolée du reste du système. Cette fonctionnalité d'isolation est fournie directement par le système, comme c'est le cas sous Linux.
Quand le système ne fournit pas directement cette fonctionnalité, Docker peut alors s'installer sur un OS virtuel qui la propose : typiquement un Linux virtualisé, sur un serveur Windows. Au pire le surcoût reste limité à un OS virtuel, et au mieux il est inexistant.

Problème n°2 : déployer rapidement à l'identique

Le second problème auquel Docker s'attaque est celui de la qualité et de la complexité de fabrication d'un environnement d'exécution. Partant des bonnes pratiques existantes, Docker propose un système de recettes, à l'image de ce que font Puppet ou encore Chef et une gestion de l'historique des changements, à l'image de Git. Il faut cependant noter que Docker fait tout ceci de manière agnostique en matière de langage. Là où il faut connaître un peu de Ruby pour écrire des recettes Chef ou Puppet, le système de Dockerfiles (les recettes) peut s'appuyer sur n'importe quel mécanisme sous-jacent, par exemple de simples lignes de commande en Bash ou du Python. Dans le Cloud, cela signifie qu'un fournisseur n'est plus obligé d'imposer un langage à son client ou de multiplier les langages supportés sous forme de plugins ou de variantes de machines virtuelles. L'autre avantage non négligeable de Docker, est que tout est beaucoup plus rapide à déployer. Il n'est plus nécessaire de préparer une recette de fabrication d'une machine virtuelle. L'OS est déployé une bonne fois pour toute et il est possible d'y implémenter autant de conteneurs Docker que souhaités. Or créer un nouveau conteneur est un processus bien plus léger que l'installation d'une image d'OS complète.

Problème n°3 : créer des capsules standardisées

Une fois que l'on a résolu les deux premiers problèmes, reste encore à permettre à tout un chacun de ne pas réinventer la roue, à savoir : le conteneur applicatif, car malgré le mécanisme de recettes réutilisables, il faut bien réaliser une toute première recette.
Là encore Docker fournit une solution qui permet de gagner du temps. Il s'agit d'un magasin de recettes mis à disposition par la communauté, à l'image des systèmes de paquets que l'on connaît déjà sous Linux ou BSD. Il devient donc possible de standardiser les conteneurs les plus utilisés.
Disposant d'ores et déjà de très nombreux outils Libres ou Open Source, on peut s'attendre rapidement à ce que le Docker Index (https://index.docker.io/) fournisse un grand nombre de recettes très utiles, et les bénéfices du fonctionnement communautaire sont immédiats :

- une meilleure propagation de l'information,
- une entraide en cas de problème,
- une participation d'acteurs importants qui promeuvent déjà les outils Libres ou Open Source.

dotCloud mais pas seulement...

La startup qui est derrière Docker, dotCloud, fondée par Solomon Hykes, est bien sûr le premier hébergeur utilisateur de Docker. Son succès ne fait plus aucun doute lorsque l'on sait que ses propres concurrents l'utilisent, et la preuve en est : l'une des présentations de Docker faite à Devoxx France 2014 a été co-animée par Nicolas De Loof de CloudBees, et Ludovic Champenois de Google.
Après un an d'existence à peine et à deux doigts de passer en version 1, l'outil semble d'ores et déjà s'être fait une place dans le Cloud. De là à dire qu'il va devenir incontournable aussi bien pour les équipes de développement que pour les équipes d'exploitation dans nos missions, il est encore trop tôt pour le dire. En revanche il n'est jamais trop tôt pour commencer à s'y intéresser...

Noël **Bardelot**
Ingénieur Concepteur chez Soat

SARAH : connecter et interagir avec l'Internet des Objets au quotidien

Jean-Philippe Encausse est venu nous présenter une conférence sur les objets connectés, intitulée « S.A.R.A.H », nom emprunté à la série de Science-Fiction Eureka dans laquelle une "maison intelligente" est mise en scène.

L'objectif du projet S.A.R.A.H., dont Jean-Philippe est à l'initiative, est de reproduire un environnement similaire à cette maison connectée. Pour entrer dans le vif du sujet, Jean-Philippe nous en fait une petite démonstration en direct : il n'utilise ni télécommande ni clavier pour passer à sa slide suivante, mais fait un simple geste de la droite vers la gauche en face d'un Kinect connecté. A cet instant-là, la slide suivante apparaît ponctuée d'un "SUIVANT" prononcé par S.A.R.A.H.

L'Internet d'aujourd'hui

Lors de son introduction, Jean-Philippe met en avant les trois grands piliers du web, qui, selon lui, forment une base solide : HTML5, CSS3 et Javascript. D'après les données présentes sur github, Javascript serait actuellement le langage le plus utilisé. Vient ensuite la présentation des 3 révolutions actuelles :

- La miniaturisation (raspberry Pi, ...),
- L'internet mobile,
- Et le crowdfunding.

Ces révolutions amènent un certain nombre de problématiques, à commencer par le responsive design. L'interface utilisateur ne peut fatalement pas être la même sur un téléphone, un ordinateur, une table surface ou encore un écran de réfrigérateur. Le responsive design doit donc être appliqué pour offrir la meilleure expérience utilisateur possible, quel que soit le support ou l'objet connecté. Se pose également la question de l'accès au monde physique via les caméras, nfc, etc., et avec une simple connexion Internet, on sait bien que ce n'est pas toujours évident. Enfin, Jean-Philippe fait le constat suivant : si les nouveaux objets connectés comportent leurs fonctionnalités, et parfois une API, ils ne savent pas toujours communiquer entre eux. Et c'est de ce constat- là qu'est né le projet S.A.R.A.H.

S.A.R.A.H.

L'approche de S.A.R.A.H. est plutôt simple : elle repose sur un serveur node.js qui va permettre de construire une couche d'abstraction à tous les objets connectés que l'on souhaite utiliser. Plusieurs plugins (~150) existent aujourd'hui pour interagir avec ces différents objets ou effectuer des actions particulières. Ces plugins concernent un tas de fonctionnalités

Fig.A

variées, cela va de la recherche météo en passant par le contrôle sonore d'un ordinateur ou encore le fonctionnement autonome d'un aspirateur ! On peut cependant distinguer 4 grandes familles de plugins :

- les plugins http, service rest, etc...
- les usages périodiques (cron,...)
- l'interrogation d'objets virtuels en faisant du scraping (via phantom.js)
- les usages un peu plus "custom" : TV, Zoé, Powerpoint sur lequel on va par exemple simuler des appuis touches, dans la mesure où il n'y a évidemment pas d'API pour le contrôler.

Jean-Philippe lance alors une vidéo nous présentant l'étendue des fonctionnalités de S.A.R.A.H. et de l'utilisation que l'on peut en faire. Pour résumer, un client C# dans SARAH permet de créer les différents usages comme le matching d'une commande à une requête http, la détection des commandes vocales, des mouvements et des QR codes. La reconnaissance vocale est faite via xml. Voici par exemple à quoi cela ressemble avec l'API Speech de Microsoft, dont vous trouverez plus de détails sur l'article suivant :
http://encausse.wordpress.com/2012/10/04/s-a-r-a-h-un-grammaire-enrichie/
La reconnaissance gestuelle est faite de la même façon, tandis que la reconnaissance faciale se fait avec OpenCV.
L'application S.A.R.A.H. présente un portail d'applications de la forme suivante : Fig.A.
Ce portail permet de télécharger et configurer les différents plugins que l'on souhaite utiliser. Un moteur de règles donne également la possibilité de déclencher des actions les unes à la suite des autres, de choisir les actions à appliquer en fonction de la situation ou du contexte (si vous êtes chez vous ou si vous y êtes absent, par exemple). Enfin, on trouve une fonctionnalité d'apprentissage des usages de S.A.R.A.H. permettant à l'application de mémoriser les actions qui lui sont régulièrement demandées, pour être capable de vous les proposer d'elle-même en fonction du contexte.

Une communauté d'ampleur pour un large éventail d'usages

Les plugins disponibles sont développés par la communauté, qui compte environ 2000 personnes actuellement. Ce chiffre s'explique facilement par le nombre d'usages possibles, avec entre autres :

- assistant personnel (agenda, téléphone, confort, multimédia)
- écologie et domotique pour économiser de l'énergie
- santé et accessibilité (coach sportif, allumage de lampes à la détection de certains sons [comme la sonnette du portail] pour des personnes malentendantes par exemple, etc...)
- jeux et réalité augmentée
- urbanisme : véhicules intelligents, panneaux intelligents
- agence et commerce pour la simplification de l'expérience utilisateur (gestion de stock, etc)
- etc.

Jean-Philippe conclut son intervention avec des démonstrations assez futuristes sur le potentiel de S.A.R.A.H. : reconnaissance faciale, vocale, gestuelle, du cinémagraphe et même un début d'hologramme ! Cette conférence a donc été l'occasion de faire un état des lieux sur la domotique et ses usages d'aujourd'hui. Le seul petit bémol que l'on pourrait émettre, porterait sur l'environnement "full Microsoft" utilisé pour S.A.R.A.H. : Windows, Kinect (API en C#), etc... Ce choix d'architecture est lié au fait que la Kinect est actuellement le meilleur dispositif en termes de rapport qualité/prix, pour effectuer de la détection audio et vidéo. Reste que S.A.R.A.H s'avère un projet très prometteur qui, on l'espère, continuera de se développer pour un jour ou l'autre peut-être, entrer dans notre maison...

Maxime **Schneider-Dufeutrelle**
Florent **Lagrede**
Ingénieurs d'études chez Soat.

Complétez votre collection

PROGRAMMEZ! le magazine du développeur www.programmez.com

Prix unitaire : 6 € (Frais postaux inclus) **France métropolitaine uniquement.**

nouveau

Tout Programmez! sur une clé USB

Tous les numéros de Programmez! depuis le n°100.

29,90 €*

Clé USB 2 Go. Livrée dans sa boite. Testé sur Linux, OS X, Windows. Les magazines sont au format PDF.

* *tarif pour l'Europe uniquement. Pour les autres pays, voir la boutique en ligne*

vous pouvez aussi commander directement sur notre site internet : www.programmez.com

☐ ***168*** : |__| exemplaire(s)
☐ ***169*** : |__| exemplaire(s)
☐ ***170*** : |__| exemplaire(s)
☐ ***171*** : |__| exemplaire(s)
☐ ***172*** : |__| exemplaire(s)
☐ ***173*** : |__| exemplaire(s)
☐ ***174*** : |__| exemplaire(s) soit |__|__| exemplaires x 6 € = |__|__| € + ☐ 29,90 € soit au **TOTAL** = |__|__|__|__|__| €

☐ ***La Clé USB avec les numéros de Programmez depuis le n°100***
29,90 € (Tarif pour Europe uniquement)

Commande à envoyer à :
Programmez!
7, avenue Roger Chambonnet
91220 Brétigny sur Orge

Prix unitaire : 6 € (Frais postaux inclus), France métropolitaine uniquement.

PROG 175

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

Prénom : |__|__|__|__|__|__|__|__|__|__|__|__|__| Nom : |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|

Adresse : |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|

Code postal : |__|__|__|__|__| Ville : |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|

Tél : |__|__|__|__|__|__|__|__|__|__| ***(Attention, e-mail indispensable)***

E-mail : |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__| @ |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|

Règlement par chèque à l'ordre de Programmez !

Valable jusqu'au 30 juin 2014
Quantités limitées

50 nouvelles choses que l'on peut faire avec Java 8...

... autres que les lambdas et les streams ! C'était le sujet de la présentation de José Paumard en ce dernier jour de Devoxx France. En 50 minutes, José Paumard nous a listé 50 évolutions apportées par Java 8 par rapport à Java 7. Morceaux choisis.
Nouvelle API «Date», très fortement inspirée de JodaTime (plus précisément, il s'agit d'une réécriture complète du framework par son créateur, afin de pallier certains défauts d'origine), avec ses concepts jusqu'alors inexistants dans l'API Java :

- Un Instant, représentant un point précis sur la ligne de temps,
- La durée entre 2 instants : Duration,
- L'espace entre 2 dates : Period,
- La gestion des fuseaux horaires,

Une interface déclarant des opérations facilitant l'arithmétique entre ces différents objets,
Et enfin bien sûr, le lien avec l'ancienne version de l'API.
De nouvelles fonctionnalités sur les **String**, qui n'étaient autrefois possibles qu'à la main ou via des librairies externes (StringUtils d'Apache ou Guava de Google par exemple) sont incluses dans la nouvelle version :

- Streamer sur les lettres de la chaîne pour y apporter des modifications grâce aux lambdas expressions,
- Concaténation facilitée (StringJoiner).

De même dans la catégorie **I/O**, pouvoir facilement streamer les lignes d'un fichier, les fichiers d'un répertoire, récursivement ou non, ...
Concernant les **Collections**, l'API s'est considérablement enrichie avec de nouvelles méthodes d'itération (`forEach()`) et de modification (`replace`, `merge`, `putIfAbsent`, ...) ; ainsi que de nouveaux Comparator permettant de trier spécifiquement les `null`, chaîner les Comparator, ...
Un point intéressant concerne la classe ConcurrentHashMap, qui a été complètement réécrite par rapport à la version de Java 7 : elle est maintenant thread-safe sans lock, et supporte un plus grand nombre d'éléments (l'index est passé de `int` en `long`).
Le dernier point que j'évoquerais ici parmi tous ceux cités, la nouvelle classe ComputableFuture qui permet de faciliter la **programmation asynchrone**.
Cette présentation a permis de mettre en évidence les nouveautés de l'API qui ont été masquées par l'arrivée des lambdas, et qui vont grandement faciliter la vie des développeurs. Pour voir si le code que vous écriviez n'est pas maintenant devenu «inutile», un seul chemin : la Javadoc !

Benoît **Cotinat** – *Netapsys*

Au revoir Maven... et place à Gradle !!

Le titre se veut volontairement provocateur tout comme la conférence à Devoxx France de Cédric Champeau : «***Gradle ne fait pas que remplacer Maven***».
Je vous propose un retour sur cette conférence pour voir les différences entre ces deux solutions et les apports de Gradle

Maven : un structurateur de build.

Apache Maven est aujourd'hui utilisé par la majorité des développeurs pour automatiser la construction (build) et le déploiement (deploy) des applications. Le concept de «**Convention over configuration**» impose de respecter un certain nombre de règles :

- Nommage et structure des répertoires projet,
- Définition du ou des pom (fichiers xml décrivant les règles de construction du projet : nom, version, dépendances, plugins...).

Maven se veut être un véritable structurateur de build laissant peu de place aux spécificités.
Pour builder avec Maven, il faut donc respecter les conventions, minimiser les configurations autant que possible et se plugger au cycle de vie à travers les différents goals (compile, test, package, install deploy...). En cas de spécificités, la solution est bien souvent d'écrire un plugin Maven spécialisé ou d'en chercher un pouvant répondre au besoin.

Et Gradle dans tout ça ?

En mélangeant et en reprenant les meilleurs concepts d'**Ant**, **Maven**, **Ivy** et **Groovy**, Gradle apporte de la flexibilité et de la souplesse pour répondre aux problématiques de build.
Contrairement à Maven, Gradle permet :

- De générer **plusieurs artéfacts pour un même build,**
- De **dé-corréler le build et le déploiement.**

Gradle repose sur le concept de **tâches** (tasks) qu'il est possible de créer **dynamiquement** notamment grâce à la puissance du langage **Groovy**.
L'autre point fort de Gradle est la notion de **build incrémental** qui permet de gagner du temps dans la construction des livrables grâce à la définition de points d'entrée et de sortie permettant à Gradle de savoir que le build est ou non **UP-TO-DATE**.
L'initialisation d'un projet par la commande «`gradle init`» génère un **wrapper dédié** au projet contenant toutes les informations de versions y compris celle de Gradle, ce qui fait que le build du projet est portable sur n'importe quel environnement.
Un fichier de build Gradle (**build.gradle**) pour un projet Java tient en une ligne :
apply plugin:'java'
Il suffit ensuite de lancer `gradle build` pour construire le projet.
Gradle permet aussi de la souplesse dans la définition des repositories (adresse d'un repository officiel, adresse d'un proxy, adresse d'un serveur partagé et toute autre ressource au sens large) pour faciliter la récupération des librairies ou autres.
Les temps d'exécution des builds sont proches et équivalents à ceux de Maven si «**—daemon**» est ajouté en option dans les commandes Gradle.
Pour terminer, quelques mots sur le futur de Gradle. Actuellement en version 1.11, le cycle de release aboutit en moyenne à une nouvelle version toutes les 6 semaines. Une version 2 est d'ores et déjà prévue avec la prise en compte du parallélisme pour optimiser les builds dans des environnements distribués.
De toute évidence, Gradle est une solution à envisager dans la construction de projet embarquant des architectures / technologies différentes entre front (Angular, Dart...) et back (Java, .NET...).

Céline **Gilet** – *Netapsys*

Au secours, mon code AngularJS est pourri !

Durant cette édition de Devoxx, je tenais particulièrement à assister au talk de Thierry Chatel **Au secours, mon code AngularJS est pourri !** Derrière ce nom un peu barbare, Thierry nous y présentait les bonnes et mauvaises pratiques.

Je pratique AngularJS à plein temps depuis plus d'un an, et son retour sur les bonnes pratiques m'intéressait pour m'aider à éventuellement resituer «nos pratiques». Pour faire court, je partage la même vision que Thierry, sauf sur un point mineur que je détaillerai en fin d'article.

Bonnes pratiques

Sans plus attendre, voici quelques points clé de la réussite d'une grosse (ou petite) application AngularJS:

- Pas d'optimisation prématurée (écrire du code pour gagner 3ms, cela rendrait le code moins lisible pour un gain...),
- Effectuer le «binding» sur une fonction et non sur le résultat (ne pas hésiter à rejouer plusieurs fois

un traitement, cela allège le code et ce n'est souvent pas coûteux en ressources),

Ne pas hésiter à manipuler le JSON pour reconstruire un vrai modele objet :

- Si on a besoin de beaucoup de méthodes d'un service dans un écran (ne pas hésiter à mettre ce service sur le scope),

Une arborescence fichier fonctionnelle et non technique :

- Définir les routes dans les différents modules, à la place d'un gros fichier de routes (cela facilite la lisibilité, et participe à isoler au maximum les modules),

- Le contrôleur ne sert qu'à initialiser le scope et faire le passe-plat vers les services (cela est vrai pour presque tous les langages),

- Ne pas hésiter à mettre plusieurs contrôleurs sur un écran (un contrôleur sur un ng-repeat permet d'effectuer un traitement sur chaque item de la liste (par exemple pour un $watch),

Pas de métier dans les contrôleurs, ni dans les templates :

- Ceci est du code «métier» : ng-class=»{'foo': bar >= 100}»
- Ceci n'en est pas :D : ng-class=»{'foo': bar.isFoo()}»

Stocker des promises au lieu du résultat (cela évite les traitements conditionnels pour savoir si la donnée est présente. Le code sera exécuté quand la promise sera résolue).

Conclusion

- Faire au plus simple
- Partir du html
- Profiter un maximum de la «souplesse» du JavaScript
- Coder en «Objet»

Petite divergence d'avis

Thierry nous préconise «1 fichier = 1 module» pour éviter d'avoir à gérer l'ordre de chargement des fichiers, et je suis d'accord avec lui pour faciliter le chargement.
Je suis plus pour utiliser des conventions de nommage (ex: monModule.mdl.js), ce qui me permet ensuite de charger les modules en premier avec les tâches Grunt ou Gulp.

Medy **BELMOKHTAR** – *Netapsys*

Les Object Calisthenics

J'ai assisté à la présentation rapide de Guillaume Duquesnay. Le sujet est la programmation des Object Calisthenics qui définit une manière d'écrire du code.
Voici une liste non exhaustive des contraintes définissant cette méthode :

- Ne pas utiliser de `ELSE` : son utilisation pourrait être révélatrice d'une méthode ayant plusieurs intentions,
- Pas plus d'un niveau d'indentation : cela améliore la lisibilité du code, ainsi que sa testabilité et sa maintenabilité,
- Encapsuler chaque type primitif (`int`, `long` ...) dans une classe dédiée : cela définit clairement le sens de la donnée et permet une programmation plus orientée objet (en Java, un type primitif n'est pas un objet).
  Exemple :

```
class NombreDeJoueurs {
      int valeur;
  }
```

- Un seul point par ligne : cela facilite la lecture du code,
- Pas d'abréviation : cela donne du sens à la variable et diminue les ambiguïtés survenant surtout lorsque le nom de la variable n'est composé que d'une seule lettre,
- Pas de collection de premier ordre (`Collection`, `List` ...) mais des classes de collection.
  Exemple :

```
class PaquetDeCartes {
    List<Carte> cartes;
}
```

- Utiliser au maximum 2 attributs de classe/d'instance. Au delà, la classe a de bonnes chances de faire plusieurs choses distinctes => il faut la découper en plusieurs classes,
- Ne pas utiliser de getter : cela évite de briser l'encapsulation des données par leur exposition en dehors de la classe.

Le bilan dressé par Guillaume est que ces contraintes donnent plus de sémantique aux classes et mènent à une programmation fonctionnelle, expressive et réactive. Cela lui a permis de redécouvrir les design patterns, notamment le pattern Command. Pour conclure, il conseille de tester cette méthode en appliquant strictement les règles pendant au moins 20 heures, afin de pouvoir vraiment en comprendre les avantages et les inconvénients.

Fabien **Duminy** - *Netapsys*

Retour sur le BOF avec les JDuchess

Quand la journée des conférences se termine à Devoxx France, c'est que les BOFs ne sont plus très loin ! Les BOFs (Birds Of a Feather) sont des sessions informelles qui donnent l'occasion de rencontrer et d'échanger avec des groupes d'utilisateurs sur les technologies Java, Scala, Groovy et autres.
J'ai choisi de vous faire un retour sur le BOF relatif au Duchess France qui est un JUG (Java User Group) 100% féminin.

Trois temps forts ont rythmé cette session :

- Le premier a permis de rappeler l'ensemble des **actions réalisées** au cours de l'année passée (conférences, hackathons...) et les JDuchess sont désormais présentes dans **8 villes** en France dont notamment Paris, Nantes, Lyon ou Tours,
- Le second a permis de **donner la place à 5 témoignages de femmes** ayant des carrières à forte visibilité dans le monde de l'IT (Amira Lakhal, Mathilde Lemée, Agnès Crepet, Laure Nemée et Kathryn Rotondo). Que ce soit en freelance, salariée d'une SSII, commiteur open-source ou créatrice d'entreprise, chacune a **partagé son expérience et ses difficultés** en toute transparence (barrières, investissement dans des projets parallèles, gestion vie professionnelle / vie privée),
- Le troisième a permis d'échanger sur ce qui pourrait **inciter et encourager les femmes à s'investir dans des métiers techniques** et notamment celui de développeur.

Sur ce dernier point, il ressort un manque de communication auprès des jeunes, principalement dans les collèges et les lycées. Ce sera donc le challenge pour cette nouvelle année : parler et expliquer ce qu'est le quotidien d'un développeur.
Retrouvez toute l'actualité et les interviews réalisées par les JDuchess sur leur site à l'adresse suivante : http://www.duchess-france.org/

Céline **Gilet** - *Netapsys*

REACT.JS : La partie 'V' du MVC repensée

J'ai assisté au B.O.F du ParisJS, ma grande première dans cette communauté et au Devoxx France. Le sujet concernait React.js, il était présenté par Khalib Jebbari. La philosophie du Framework a piqué ma curiosité au vif. Je l'ai testé sur des cas basiques et vous propose une brève présentation.

React.JS : est une librairie JS dédiée aux interfaces graphiques.

React.js (Git Hub : https://github.com/facebook/react,) est un Framework dédié à la création d'interface utilisateurs des applications web; il a été mis à la disposition de la communauté courant 2013. Propulsé par Facebook et Instagram, il se range dans la catégorie V voire C du 'MVC'
React.js nous amène à remettre en cause l'interaction entre les views et les templates pour la construction des vues. En fait, le principe de couplage du View-Model et des templates Handlebar par exemple n'existe plus. **Il n'y en a que pour les composants!**

Quelques notions clés :

Les composants (component)

Notion centrale du Framework, les composants (component) sont des blocks HTML réutilisables et modulables.
Ce sont les **briques de HTML** et leurs **événements associés** que l'on veut pouvoir rajouter au DOM.

Les propriétés : props et state

Les props sont des paramètres passés au composant. Une fois créés, il n'est pas possible de les modifier sauf si l'on a implémenté un handler modifiant ces propriétés via la propriété **state.**
Le type des props peut être spécifié ainsi que la présence ou non du paramètre lors de la création du composant.

DOM Virtuel

React.js utilise un DOM virtuel pour les mises à jour du DOM. Ce mécanisme lui permet d'être très rapide. En effet, il maintient une copie du DOM initial et un autre portant les modifications. Dès qu'il détecte un changement sur un nœud, il reconstruit entièrement l'arbre. Cette opération optimise la régénération du DOM et rend l'affichage du rendu visuel plus rapide.

JSX

Librairie **optionnelle** qui permet juste de représenter le contenu du composant sous forme d'arbre dans le DOM virtuel.
En l'absence de ce dernier, on doit gérer la logique d'affichage des composants en Javascript.
Le code ci-dessous montre la différence entre les deux syntaxes.
//Création d'un composant hello world

- Avec JXS :

{Lien externe pour tester le code}
http://jsfiddle.net/2UQKF/1/
{Pour intégrer le code directement dans le billet}

```
<iframe width=»100%» height=»300» src=»http://jsfiddle.net/2UQKF/1/embedded/» allowfullscreen=»allowfullscreen» frameborder=»0»>
</iframe>
```

- Avec le framework REACT.DOM

{Lien externe pour tester le code}
http://jsfiddle.net/RU4GA/2/
{Pour intégrer le code directement dans le billet}

```
<iframe width=»100%» height=»300» src=»http://jsfiddle.net/RU4GA/2/embedded/» allowfullscreen=»allowfullscreen» frameborder=»0»></iframe>
```

La méthode render

Elle contient la logique d'affichage de la vue, l'imbrication des composants sous forme d'arbre à l'intérieur du DOM. On peut y déclarer du HTML, des composants ou un mix des deux.
Cette méthode est appelée quasiment à chaque changement d'état du composant.

Exemple de composant « mixte»

Ce composant mixe la création d'un composant qui contient des composants et du HTML.
Exemple de composant :
{Lien externe pour tester le code}
http://jsfiddle.net/e9dFB/17/
{Pour intégrer le code directement dans le billet}

```
<iframe width=»100%» height=»300» src=»http://jsfiddle.net/e9dFB/17/embedded/» allowfullscreen=»allowfullscreen» frameborder=»0»></iframe>
```

Note : *La fonction* **render** *retourne à chaque fois l'ensemble de nos composants React sous forme d'arbre qui peuvent ou non être affichés. Dans le cas où l'on utilise JSX ne pouvant renvoyer qu'un seul nœud, il convient, quand cela est nécessaire, de regrouper le HTML des composants sous un node principal.*
Dans le cas de JSX, le commentaire au début du JS est obligatoire.

Conclusion

Un Framework orienté vue propulsé par Facebook, on se doute qu'il a et aura ses adeptes.
La prise en main est rapide et le dossier d'installation fournit des exemples exhaustifs incluant les cas d'utilisations avec requireJS.
Cependant, j'ai trouvé que React permet au développeur que je suis de mettre beaucoup de logique dans la vue. Etant habituée à séparer la View-model et ses templates, pouvoir mixer l'ergonomie de l'application `(la création des composants et la manipulation du DOM)` et la logique de fonctionnement `(Gestion évènements utilisateurs et mise à jour de l'interface utilisateur voire du model)` au même endroit me semble un peu ... propice à l'écriture de code spaghetti.
Il serait néanmoins intéressant de l'intégrer dans une vraie application MVC pour un retour plus complet.
Dans ce billet j'ai voulu, comme lors de la session du BOF, partager sa philosophie qui reste intéressante, et son utilisation sur des cas basiques de construction de vues pour les applications web.

Des bonnes pratiques ?

Sur le site officiel du projet, les développeurs du Framework préconisent entre autres de :

- Découper l'ergonomie en composants les plus petits possibles pour faciliter les tests unitaires,
- Connaître le workflow minimal de l'application,
- Identifier les composants sujets à des modifications,
- Implémenter les fonctions de mises à jour sur ces composants.

La présentation sur React.js de Khalib Jebbari est disponible sur son github :
https://github.com/DjebbZ/react-paris-js/blob/master/NOTES.txt

Qui l'utilise ?

-> Facebook/Instagram
-> Khan Academy

Source

Présentation du Framework :
http://facebook.github.io/react/index.html
http://code.tutsplus.com/tutorials/intro-to-the-react-framework—net-35660
http://facebook.github.io/react/docs/thinking-in-react.html
http://webdesignporto.com/react-js-in-pure-javascript-facebook-library/

Performance du DOM Virtuel :
http://calendar.perfplanet.com/2013/diff/

Michelle **Avomo**
Netapsys

Réussir son projet Open Source : les bonnes pratiques

Il est désormais loin le temps où lancer un projet Open Source relevait uniquement de l'engagement philosophique pour un partage des connaissances ouvert et libre. Bien entendu, cette motivation reste bien présente au sein des communautés et des associations de soutien au Logiciel Libre. Mais créer un projet Open Source est dorénavant une décision mûrement réfléchie en accord avec des motivations personnelles et économiques aux multiples reflets.

Dès lors, plus question de lancer un simple message du style « Hello everybody (...), I'm doing a Free OS... » et d'espérer qu'une communauté s'empare de votre code pour en faire le noyau le plus utilisé au monde. Vous ***devez*** maîtriser la réussite de votre projet !

Réussir son projet ?

Mais au fait : réussir son projet, ça signifie quoi exactement? **C'est bien évidemment la question qu'il faut se poser avant toutes les autres**.
Tout au long de votre projet, vous allez devoir prendre des décisions structurantes sur de nombreux sujets : hébergement collaboratif, outils, licence, communication, documentation...et pour cela, vous devez être très clair avec vos objectifs :

- Concrétiser une idée originale et la partager avec les autres pour l'enrichir ?
- Etre reconnu par les gourous du domaine ? Devenir vous-même une référence ?
- Démontrer votre savoir-faire pour alimenter une activité freelance (ou décrocher un job de rêve au bout du monde) ?
- Partager le socle logiciel du produit phare de votre startup pour en favoriser sa diffusion ?
- Mutualiser vos efforts de R&D avec des partenaires pour réduire vos coûts ?
- Vous libérer de la maintenance des millions de lignes de code de votre framework pour vous concentrer sur les plugins lucratifs ?

Vous allez vite constater qu'il n'est pas toujours simple de faire les bons choix dans la masse des solutions disponibles. Voici donc quelques points de repère pour répondre aux interrogations les plus courantes :

- Quand faut-il publier son projet ?
- Quels sont les ingrédients pour attirer des contributeurs ?
- Quid des licences ? Comment peuvent-elles m'aider à atteindre mes objectifs ?
- Comment publier mon projet ? Quels outils mettre à disposition des développeurs ?
- Comment faire vivre la communauté qui va se créer autour de mon projet ?

Quand publier ?

Plutôt que de répondre directement à cette question, rappelons deux règles simples mais toujours vérifiées en pratique :

- Règle n°1 : quand une idée est originale, elle l'est dès le début du projet. Inutile de fignoler la Nième interface d'administration avant de publier : les utilisateurs vont s'emparer du projet très vite si ce dernier répond à un vrai besoin, même s'il est incomplet.
- Règle n°2 : la qualité est primordiale. Un projet incomplet mais avec une base très bien structurée et codée mobilisera rapidement une communauté de développeurs prêts à prendre en charge les fonctionnalités manquantes. Dans le cas contraire, la sanction sera immédiate. Autant être fixé rapidement et pouvoir corriger le tir sur une quantité de code limitée, sinon vous risquez d'être découragé devant l'ampleur de la tâche et vos efforts auront été inutiles.

Notre conseil : *concentrez-vous sur la qualité et validez votre projet le plus vite possible.*

Même si les deux règles précédentes visent plutôt des nouveaux projets, elles restent applicables au cas des logiciels propriétaires qui sont « libérés » par leurs éditeurs. En particulier, rendre Open Source une application en perte de vitesse en espérant que les contributions lui permettront de retrouver une seconde jeunesse est voué à l'échec.
Par contre, publier en Open Source un logiciel reconnu pour sa qualité et en plein succès – en adaptant naturellement le modèle économique – peut avoir un effet de levier réel.

Nul n'est prophète en son pays...

Il y a de fortes chances pour que les premières contributions à votre projet viennent de l'autre bout du monde...à condition d'avoir créé les conditions favorables !
L'utilisation de l'anglais est donc particulièrement requise. Pour la documentation et le site Web qui hébergera le projet bien entendu, mais pas seulement : penser aussi aux commentaires et au code lui-même. Cela semble évident, mais le *franglais* est à proscrire absolument !
Quel développeur anglophone aura envie de contribuer à un code source qui ressemble à celui ci-dessous ?

```
…
int table[nbLigne];
/* Tableau initialization */
for(int i=0 ; i <nbLigne ; i++)
{
 if(maVariableGlobale[i] == TRUE)
   table[i] = i ;
…
```

Penser aux contributeurs

Un projet Open Source, c'est d'abord une communauté de contributeurs qui le fait vivre. Sans elle, il sera comme une coquille vide. Dans ce paragraphe, nous ne nous intéressons pas à la manière de faire vivre cette communauté, mais aux quelques règles de bon sens qui permettront à cette dernière d'exister.
Si la publication en anglais est une première règle *sine qua non*, il faut aussi penser à la **lisibilité** de votre publication. Comprenez : la qualité structurelle du code et les commentaires.
Respecter les règles de codage aura un double impact :

- Favoriser la compréhension de votre projet par autrui, et ainsi faciliter les contributions. Un code mal construit impose un effort supplémentaire qui peut éloigner les développeurs les plus motivés.
- Améliorer la qualité perçue de votre projet, et donc susciter l'intérêt.

Les règles de codage relèvent des bonnes pratiques propres à chaque langage. Il est donc difficile de les expliciter ici. Pour les projets écrits en langage C, le GNU coding standard est toutefois une bonne référence. On peut aussi s'appuyer sur les règles édictées par quelques organisations de référence. Par exemple, s'inspirer des règles de codage de Google en C++ ou de DRUPAL en PHP est un bon point de départ.
Voici quelques règles qu'il est utile de bien respecter dans tous les cas :

- Adopter une convention de nommage claire et consistante
- Eviter les raccourcis d'écriture et ne pas hésiter

à réécrire une commande sur plusieurs lignes si besoin

- Par exemple écrivez :

```
if(my_pointer == NULL)
{
  my_pointer = malloc(sizeof(my_struct));
}
```

Plutôt que :

```
if(!my_pointer) my_pointer = malloc(size
of(my_struct));
```

- Soigner l'indentation du code

Notre conseil : *a minima, portez une attention particulière aux warning de votre compilateur.*
Par exemple avec GCC, activez un maximum de warnings et traitez les comme des erreurs : gcc -Wall -Werror

N'hésitez pas à utiliser des outils permettant de tester la qualité de votre code. Même si ces derniers ne sont pas infaillibles, publier les résultats obtenus sera un gage de sérieux.
Complément : Si votre projet vise le secteur industriel, il est sans doute judicieux de prévoir de respecter des normes plus rigoureuses, comme par exemple la norme MISRA (C et C++). La norme MISRA, créée en 1998 par la *MotorIndustry Software Reliability Association* a pour vocation de définir un ensemble de règles permettant d'améliorer la robustesse des logiciels.
Un exemple pour mieux comprendre la philosophie de cette norme : pour être conforme MISRA C, il est **obligatoire** de prévoir un branchement *default* dans un bloc *switch* ou un *else* dans un bloc *if*.
Des outils existent pour valider la conformité d'un code source à la norme MISRA : par exemple les outils d'analyse statique de Coverity ou LDRA Testbed de Liverpool Data Research Associates. Ces derniers sont toutefois propriétaires et assez onéreux, donc peu accessibles aux particuliers. Mais si votre projet intéresse vraiment un industriel, ce dernier pourra sans doute vous aider à tester la conformité de votre code... Un autre élément de qualité du code concerne naturellement la documentation de ce dernier, sur laquelle il faut porter une attention toute particulière.
Adopter Doxygen est une bonne habitude plus que recommandable. Son utilisation permet de générer automatiquement la documentation de votre projet à partir de commentaires structurés avec des tags de référence : le gain de temps est remarquable et la lisibilité de votre projet garantie. Pour s'en convaincre Il suffit de demander à un développeur d'IHM pourquoi il privilégie la librairie Qt plutôt que GTK...réponse immédiate : la qualité de la documentation ! Un très bel exemple de documentation générée automatiquement (avec qdoc, un outil Qt équivalent à Doxygen).
Voici un exemple de commentaire intégré dans du code pour être exploité ensuite par Doxygen :

```
/**
 * This computes the addition of two
integer values
 * @param a is the first integer
 * @param b is the second integer
 * @return the calculated value
 */
int myAddition(int a, int b);
```

Notre conseil : *commentez votre code source en Doxygen dès la première ligne de code.*

Last but not least : penser à porter une attention particulière aux outils de production.
Ils sont trop souvent négligés en début de projet, lorsque quelques lignes dans un Makefile suffisent pour compiler le code source. **Souvenez-vous que vous devez anticiper le succès de votre projet** et faciliter la vie aux contributeurs.
Optez plutôt pour un outil spécialisé comme CMake (adopté par KDE, MySQL, OpenCV, etc.). Deux avantages majeurs à utiliser CMake :

- Une prise en charge facilitée de la complexité croissante de votre projet
- L'outil de production est multi-plateformes, permettant aux contributeurs de travailler dans leur environnement sans adaptation spécifique.

Signalons aussi l'outil historique du projet GNU : les autotools, qui sont encore largement utilisés mais plus complexes à mettre en œuvre.

Penser (aussi) aux utilisateurs !

Pour un projet Open Source, il n'est pas toujours évident de distinguer les contributeurs des utilisateurs. Autrement dit : pas de contributeurs sans utilisateurs.
Il ne faut donc pas négliger ce qui va rendre un projet attrayant : des visuels pour donner envie d'essayer votre application, une documentation claire et complète, des packages d'installation et, pour les bibliothèques logicielles, des exemples.
En ce qui concerne les visuels, est-il vraiment utile de rappeler l'importance des incontournables screenshots pour une application ? Mais ce n'est pas toujours très applicable. Il existe aujourd'hui un media frappant : les vidéos. Dans ce cas, la moindre démonstration dans une fenêtre console devient vivante et aura un impact redoublé.
Il ne faut pas se voiler la face : les développeurs rechignent souvent à rédiger la documentation utilisateur sous prétexte qu'il s'agit de temps perdu sur la production de code. Pourtant, c'est un élément important pour attirer des utilisateurs néophytes et favoriser le bouche à oreille.
Notre conseil sera double :

- Publier tôt avec une documentation utilisateur complète, car nécessitant un moindre effort.
- Indiquer clairement que l'on recherche des volontaires pour aider à rédiger la documentation, plutôt que de la négliger purement et simplement.

Les packages d'installation vont favoriser le test de votre projet. Même un développeur chevronné l'utilisera lorsqu'il découvrira votre projet et voudra se faire son idée en quelques minutes. Après, bien entendu, s'il est convaincu, il téléchargera les sources et recompilera l'ensemble dans son environnement. Soigner le packaging est donc important pour développer la communauté des utilisateurs.
Enfin : prendre soin de fournir des programmes d'exemple dans le cas des bibliothèques logicielles. Le fameux 'Hello World' est bien sûr incontournable, mais il ne faut pas oublier que les programmes d'exemple seront le premier contact d'un utilisateur avec votre projet. Ils doivent donc être didactiques mais démonstratifs, donner envie d'aller plus loin.
Vous l'aurez compris : créer un projet Open Source, c'est aussi du marketing !

Choisir sa licence

Le choix de la licence apparaît comme une question très technique, souvent un peu obscure. Il est vrai que pas moins de 70 licences libres sont recensées !
Pourtant, les premiers (et principaux) critères de sélection n'ont rien de technique. Ils relèvent surtout de choix de principe effectués au lancement du projet :

- Le projet est-il associé à un objectif économique ?
- Faut-il se prémunir contre toute récupération ou appropriation industrielle du projet ?
- Souhaite-t-on se rapprocher d'une fondation existante et bénéficier de sa notoriété ?
- La motivation profonde est-elle de partager un savoir-faire ou une connaissance scientifique ?

Il existe en effet deux grands types de licences : les licences copyleft et les licences non-copyleft. Dans un premier temps, bien comprendre la distinction entre les deux modèles est nécessaire et – presque – suffisant pour faire le bon choix....
Les licences copyleft relèvent de la démarche initiale du mouvement Free Software lancé par Richard Stallman. L'accent est mis sur la liberté et la préservation de cette liberté. Liberté de télécharger, d'utiliser, d'étudier, de diffuser à une double condition :

- Une fois la technologie en votre possession, vous ne pouvez pas vous l'approprier (elle garde sa licence originale)
- Toute évolution introduite dans cette technologie est-elle même soumise à la même licence.

Le deuxième point explique que l'on parle alors de **licence contaminante**.
L'exemple de licence copyleft la plus répandue est la licence GNU GPL (GNU General Public Licence). C'est par exemple la licence du noyau Linux. Les conséquences sont immédiates : tout développement destiné à être intégré dans le kernel Linux sera sous licence GPL (effet contaminant). C'est le cas – en théorie – de tous les drivers. En théorie car certains industriels n'hésitent pas à diffuser des drivers qui ne sont pas sous licence GPL...

Quand choisir une licence copyleft ?
Première réponse simple : vous n'aurez peut-être pas le choix ! Si vous choisissez de vous appuyer sur des composants logiciels sous licence copyleft, ou de développer une fonctionnalité complémentaire à un logiciel sous licence copyleft (projet dérivé), l'effet contaminant s'applique.
Un autre critère clé est le suivant : vous souhaitez vous protéger contre toute appropriation de votre technologie par un concurrent (économique ou universitaire).
Il n'en reste pas moins qu'une licence copyleft pourra être considérée comme trop restrictive si on souhaite favoriser une diffusion large de notre technologie. La notion de contamination freinera un industriel qui souhaite s'appuyer sur des technologies libres pour développer un produit, tout en protégeant un savoir-faire spécifique.
Il existe donc des licences non-copyleft beaucoup plus permissives. Les plus connues sont les licences BSD, MIT et Apache. Il est tout à fait possible d'utiliser une technologie libre sous licence non-copyleft dans des produits propriétaires.
Signalons enfin deux points remarquables :

- Si le kernel Linux est sous licence GPL, il y a une petite phrase qui précise que les environnements fonctionnant au-dessus de ce kernel ne sont pas tenus d'être sous licence GPL. C'est ce qui permet à quiconque de faire fonctionner une application propriétaire dans un OS GNU Linux.
- La licence LGPL est un bon compromis souvent adopté pour les bibliothèques logicielles. Tout en protégeant la bibliothèque elle-même par une licence copyleft, elle permet de linker cette dernière à tout type de projet, libre ou non.

Le gestionnaire de versions

A minima, votre projet doit être géré par un logiciel de gestion de versions.
Git, initialement développé pour le kernel Linux (par Linus Torvalds lui-même), s'impose alors nettement comme le gestionnaire de configuration des projets libres. Il a supplanté Svn (Subversion), autre outil régulièrement rencontré dans le monde Open Source.
Sans entrer dans le détail des avantages de git sur svn, cette suprématie s'explique sans aucun doute par sa capacité à gérer un très grand nombre de développeurs sur un même projet, lorsque svn trouvera ses limites avec une dizaine de contributeurs...
Son mode de fonctionnement est particulièrement adapté aux projets collaboratifs. En effet, chaque contributeur :

- télécharge une copie locale de l'ensemble du repository
- travaille à partir de son repository local
- peut facilement créer des branches pour tester de nouvelles idées, sans mettre en danger le reste du projet
- dispose d'une gamme d'outils complète pour communiquer ses évolutions aux autres participants

Ce mode de fonctionnement rend le gestionnaire de version Git beaucoup plus rapide et tolérant aux erreurs du développeur (impossible d'annuler un commit malheureux avec svn...).
Svn est basé de son côté sur un modèle classique de serveur centralisé sur lequel chacun contribue après avoir effectué un checkout.
Chaque commit est alors appliqué directement sur le repository centralisé, à condition de disposer des droits pour cela. Pour contribuer, il faut donc déjà avoir été accepté par la communauté.
La sélection est très sévère car vous l'aurez compris : chacun de vos commits peut déstabiliser la branche principale du projet.

Héberger et diffuser son projet

Voilà : vous avez suivi tous nos conseils et vous êtes prêts à faire connaître votre œuvre au plus grand nombre. Mais au fait, comment diffuser efficacement un projet Open Source et le faire connaître ?
Réponse : en l'hébergeant sur une forge, car vous disposerez alors d'une palette d'outils bien utiles et deviendrez visible. Si un développeur cherche un projet Open Source sur lequel contribuer, nul doute en effet qu'il commencera par chercher sur les forges les plus utilisées.
Impossible de ne pas connaître Sourceforge, forge historique sur laquelle vous avez nécessairement téléchargé un logiciel au moins une fois.
De la même manière que Git est devenu le gestionnaire de version privilégié, github est aujourd'hui la forge qui regroupe à lui seul environ 55% des projets Open Source. Quelques chiffres pour se sentir moins seul : en 2013, github représentait 3 millions d'utilisateurs, 4,6 millions de projets et environ 2,5 millions de commits annuels.
En hébergeant votre projet sur github, vous disposerez :

- D'une page de description du projet automatique à l'aide du README.md,
- D'un système de suivi des bugs,
- D'un wiki pour l'échange d'informations,
- Du système de fork/pull_request qui favorise les contributions,
- Et enfin d'un véritable réseau social dédié.

Bien entendu, il est tout à fait possible d'héberger soi-même son projet sur un site Web dédié. Nous le déconseillons toutefois. Tout d'abord parce que cela va donner une image 'suspecte' de votre démarche...le partage est-il vraiment votre objectif ? Les contributeurs n'apprécient pas toujours un tel comportement. Ensuite parce que votre projet doit monopoliser votre énergie : quel intérêt de la dépenser dans la mise en place et la maintenance d'outils alors que tout est à votre disposition sur une forge communautaire ? Il est enfin évident qu'une telle démarche réduira fortement la visibilité du projet, et sera donc un handicap pour créer une communauté active.

Une communauté s'est créée...il faut la faire vivre !

Enfin...votre projet est solide techniquement, bien lancé, protégé par une licence adaptée et hébergé dans une forge communautaire visible. Une petite communauté s'est créée et produit déjà les premières évolutions. Rassurez-vous : tout ne fait que commencer !
Il s'agit en effet maintenant de faire vivre cette communauté. C'est à dire de :

- Définir une gouvernance claire : ne pas oublier qu'un projet Open Source doit maîtriser en permanence sa qualité. Définir clairement qui peut accepter des contributions et les intégrer dans le tronc principa,
- Donner de la visibilité sur la roadmap : un projet retiendra des contributeurs s'ils ont le sentiment que vous savez exactement où vous voulez aller,
- Communiquer en externe : participer à des conférences, publier des articles, avoir un site Web à jour avec des news récentes, répondre aux demandes sur les forums,
- Communiquer en interne : être disponible, mettre en place des outils de communication,
- Créer des liens avec les intégrateurs : ces derniers sont au contact direct des projets, ils pourront promouvoir votre technologie si elle répond à un besoin exprimé par leurs clients.

Conclusion

Arrivé à ce point, que pourrait-il manquer d'important pour être complet ? Sans aucun doute un enthousiasme à toute épreuve et une bonne dose d'énergie. Ce sont les deux ingrédients absolument nécessaires pour s'engager dans l'aventure de l'Open Source. Pour tous les autres sujets plus techniques, nous espérons que les éléments présentés dans cet article constitueront des repères solides. Enjoy !

Olivier **Vine**
Directeur d'OpenWide Ingénierie

Créer son projet en open source, un modèle de développement d'avenir

Pourquoi faire le choix de développer son projet selon les règles de l'open source ?

Le fait de contribuer à l'enrichissement collectif du monde de l'informatique n'est pas l'unique motivation d'un projet open source. Là où le logiciel libre met en avant une certaine éthique du développement, l'open source privilégie un procédé plus pragmatique, basé sur le libre accès au code. Cela va au delà de l'aspect enthousiasmant que peut avoir le fait de participer à un projet basé sur le partage, la coopération et l'enrichissement collectif; la mise à disposition du code source de son projet, afin qu'il puisse être librement accessible, modifiable et redistribuable offre de nombreux avantages. En effet, c'est avant tout la garantie d'une plus grande qualité et d'une meilleure sécurité du code, celui-ci étant analysé et testé par tous les contributeurs. Ceci permet également une détection et une correction plus rapide et efficace des bugs; les communautés permettant une grande réactivité. Par ailleurs l'apport d'une communauté nombreuse et active dynamise le projet et stimule son évolution constante. Celle-ci peut en permanence proposer des améliorations, développer des extensions... En bref, la mise en commun des ressources des contributeurs permet une capacité de travail accrue et un recul plus grand sur le projet.

Le point fort de l'open source : la communauté

Dans un projet open source, le code est visible afin qu'il puisse être commenté, testé, amélioré par tous. Pour qu'un tel projet prenne tout son sens, il est donc logiquement essentiel qu'il dispose d'une communauté de contributeurs. Il sera d'autant plus pérenne et fiable si cette communauté est importante en nombre et en implication. Elle peut être le meilleur gage de qualité de votre projet. La division du travail permet évidemment une progression plus rapide et dynamique, et la confrontation de points de vue différents enrichit la réflexion à tous les stades du développement du projet. Par ailleurs, la communauté est un élément essentiel du bouche à oreille, qui représente souvent le meilleur moyen de faire connaître son projet et de le promouvoir. Aussi il est essentiel d'assurer une communication de qualité auprès et au sein de celle-ci, la tenir régulièrement à jour des évolutions, ambitions (roadmap) et éventuelles difficultés du projet. Réciproquement il faut être à l'écoute des contributions qui sont l'atout majeur de l'open source. Ce travail de liaison permanent est primordial afin de maintenir la cohérence du projet et l'harmonie des contributions.

Différents moyens au service du projet

Quels outils pour vous accompagner dans le développement du projet ? Afin de mettre en commun les travaux et ressources au sein de la communauté, il est nécessaire de disposer d'une plateforme d'hébergement et de partage pour le projet : la forge. De nombreuses solutions existent en la matière, au nombre desquelles le classique sourceforge.net, GitHub, Google Code et bien d'autres. Cette forge représente l'outil fondamental du travail en open source ; elle permet aux contributeurs d'accéder au code source du projet, et d'y soumettre le fruit de leurs travaux, leurs réflexions ou rapports de bugs. Ces plateformes présentent également le grand avantage d'être un vivier naturel de contributeurs potentiels, et offrent au projet une visibilité auprès de nombreux développeurs susceptibles d'y participer. Il peut être également utile de disposer d'un site internet afin de communiquer autour du projet, de son avancement, et de lui offrir une plus grande visibilité; de manière plus globale, il s'agit de le promouvoir auprès de sites spécialisés, blogs ou forums. Par ailleurs, une documentation fournie et de qualité est une aide à la compréhension et à la clarté de votre projet, et donc à sa diffusion.

Une question épineuse, celle de la licence

L'une des questions les plus importantes et complexes qui se pose au développeur qui veut initier un projet open source est celle du choix de la licence. La licence est ce qui définit les traits d'utilisation et d'accessibilité au projet open-source. Il existe deux grandes catégories parmi les licences libres qui sont les licences libres dites copyleft et non copyleft. Le copyleft, par opposition au copyright, indique que quiconque redistribue le logiciel avec ou sans modifications doit aussi transmettre la liberté de copier et de modifier ce logiciel. Autrement dit, toute redistribution ou réutilisation du code hérite des contraintes de la licence initiale. Parmi les licences copyleft, la GPL (General Public License) est la plus utilisée. Elle donne la liberté sur un logiciel de l'exécuter, de l'étudier, de redistribuer, de le copier et de faire bénéficier à la communauté des versions modifiées. Parmi les licences non copyleft, on peut trouver des licences connues comme la licence MIT, ou encore BSD. Leur particularité est que chaque personne recevant le logiciel a tous les droits sur le logiciel mais qu'elle a aussi l'obligation de mettre le nom des auteurs avec la notice de copyright. Ainsi du code sous ce type de licence peut tout à fait être utilisé dans un logiciel propriétaire. Il est primordial de bien choisir sa licence, pour définir quels droits aura le public sur le code, en fonction de la portée et de l'utilisation, notamment commerciale, que l'on veut lui donner.

Un modèle économique viable ?

La satisfaction de participer à une œuvre collective qui aboutit à la création d'une solution performante peut être une récompense en soi. Cependant, on peut légitimement se poser la question de la rémunération dans le cas d'un projet pour lequel le code est libre d'accès et de redistribution. En d'autres termes, le modèle open source est-il viable économiquement ? La réponse à cette question réside essentiellement dans les activités de service. En effet la mise en place d'un logiciel open source auprès d'une entreprise par exemple, entraîne un besoin en termes d'intégration et de support. On peut citer l'exemple de Red Hat, qui base son activité sur le service autour de distributions et logiciels open source. L'équipe de développement est naturellement la plus à même de remplir cette mission. Par ailleurs certains projets possèdent plusieurs licences: une destinée à une diffusion large et gratuite, une autre pour la commercialisation du produit sous des versions alternatives, notamment auprès des entreprises. D'autre part, les dons des contributeurs peuvent représenter une part non négligeable de l'effort financier soutenant le projet, ainsi que le sponsoring de grands groupes qui n'hésitent pas à investir dans ce type de projets, à l'instar de Sun, Google ou IBM. L'open source est une méthode de développement qui offre de nombreux avantages, et qui a de plus en plus la faveur des développeurs, mais aussi des entreprises qui n'hésitent plus à recourir à ce type de produit. S'il soulève encore des interrogations en termes de licence et de modèle économique, il apparaît comme une solution d'avenir, et son utilisation ainsi que son intégration dans le paysage informatique sont amenées à croître encore.

Faouzi **Chaachoua** et Ibrahim **Fofana**, *développeurs web à l'ETNA*

Le projet open source MVVM Light Toolkit

Le MVVM Light Toolkit est un projet open source très utilisé dans le monde .NET (XAML/C#). Il rencontre une grande popularité due en grande partie à sa simplicité d'utilisation et le fait qu'il n'essaie pas de résoudre tous les problèmes mais qu'il se concentre seulement sur certains d'entre eux.

Cet article n'est pas une présentation de MVVM Light en soi, mais plutôt une discussion sur la création et sur certains choix faits dans le cadre de ce projet.

La genèse

MVVM Light n'a pas été créé au début comme un projet open source, mais plutôt comme une librairie pour construire des exemples pour mon blog (http://blog.galasoft.ch). En effet je souhaitais avoir une fondation sur laquelle je puisse construire des exemples progressivement plus complexes. Apres avoir montré le code à plusieurs amis de la communauté Windows Presentation Foundation (une plateforme pour construire des applications desktop pour Windows en 2009), ils ont tous été intéressés à voir une version de production. J'ai donc remis l'ouvrage sur le métier avec l'aide de quelques-uns d'entre eux, et ai publié la version 2 quelques semaines plus tard. C'était donc un effort collectif depuis le début.

Peu de temps après, j'ai publié une version pour Silverlight, qui a été très bien reçue par la communauté. De même, quand Windows Phone a été créé par Microsoft, MVVM Light a très vite été disponible, et plus tard pour les applications Windows 8 « Store ». Aujourd'hui, MVVM Light supporte toutes les plateformes qui utilisent XAML/C# (y compris XBOX) et je prépare une version pour Android et iOS via Xamarin. C'est donc vraiment une librairie « cross-platform ».

Pourquoi CodePlex ?

Le site CodePlex est un lieu habituel et bien connu pour abriter des projets de la communauté Microsoft, et donc c'était un choix logique pour MVVM Light. En 2009, GitHub en était à son enfance et donc encore peu utilisé. De plus CodePlex offre la possibilité d'utiliser Mercurial pour le source control, ce qui me convient mieux (Mercurial offre aussi le source control déconnecté, mais a l'avantage d'être plus facile à utiliser que Git, surtout quand il n'y a que peu de développeurs travaillant sur un projet).

En plus du site public sur CodePlex, j'utilise un repo prive hébergé par BitBucket. C'est en fait le repository sur lequel je travaille à de nouvelles fonctions pour les versions futures de MVVM Light. Quand je publie une version, je merge le repo de BitBucket et le repo de CodePlex. Cela peut paraître compliqué, mais étant un perfectionniste, j'apprécie la possibilité de travailler à de nouvelles fonctions de manière un peu « cachée » jusqu'à ce que je me sente prêt à ouvrir le code au public.

Les deux repositories (CodePlex et BitBucket) sont gratuits et j'utilise TortoiseHg pour l'accès, qui est un excellent programme également gratuit.

Les outils

Pour développer MVVM Light, j'utilise Visual Studio pour lequel j'ai la chance d'avoir une licence gratuite, étant un « Most Valuable Professional » Microsoft. Cela dit, la plupart des fonctions de Visual Studio que j'utilise pour MVVM Light sont disponibles dans Visual Studio Express, qui est disponible gratuitement pour tout développeur. Donc a priori il est parfaitement possible de développer une plateforme open source pour l'écosystème Microsoft sans frais.

La licence

J'ai décidé de distribuer MVVM Light sous la licence MIT. Il s'agit d'une des licences les plus permissives qui soit. Je n'ai jamais été un fan des licences restrictives populaires et compliquées telles que GPL. Pour moi, ce type de licence contribue à expliquer pourquoi nous n'avons pas plus de projets open source, et pourquoi beaucoup préfèrent éviter d'utiliser ces librairies dans leurs applications. Le simple fait que la licence GPL prenne plusieurs pages montre bien qu'elle est tout simplement inutilisable par des programmeurs comme vous et moi qui n'avons pas des armées d'avocats pour nous défendre. En fait, j'ai même considéré presque de manière sérieuse la possibilité d'utiliser la licence WTFPL (http://www.wtfpl.net/) mais ai quand même opté pour une licence moins controversée...

En gros, la licence MIT reflète ce qui est pour moi l'essence de l'open source : partager son savoir et son travail en encourageant les autres à utiliser et modifier le code suivant leurs besoins spécifiques. La licence MIT demande l'attribution, ce qui me parait être la moindre des choses. Cela dit, n'ayant pas de moyens juridiques, même cette clause est impossible à vérifier, et donc je n'attache pas une importance monumentale à ce fait.

Les fruits du travail

Bien que je ne gagne pas d'argent de manière directe avec MVVM Light, j'ai grandement profité du travail effectué. Tout d'abord, en termes de réputation dans la communauté et d'expérience, les 5 dernières années ont été fantastiques. J'ai eu la chance d'être invité à de nombreuses conférences internationales, de contribuer à des publications telles que MSDN Magazine, et je publie en Mai 2014 un cours chez Pluralsight dédié à MVVM Light. Ces quelques activités sont en partie rémunérées. J'ai aussi un bouton « donations » sur le site mvvmlight.net, mais il faut bien reconnaitre que presque personne n'utilise cette option.

Cela dit, il va de soi que le plus grand plaisir est de partager le savoir, d'étendre ce savoir en communiquant avec la communauté, et de rencontrer des utilisateurs heureux dans le monde entier.

J'espère que cet aperçu des coulisses de MVVM Light vous aura éclairé sur les motivations de ce projet et vous encouragera d'une part à visiter sa page, et peut être à l'utiliser dans un projet, mais surtout à contribuer aussi à un ou plusieurs projets open source dans votre langage préféré.

Laurent **Bugnion**

Rendre son projet open source avec Github

Souvent le passage à l'acte de rendre son projet open source se heurte à une difficulté : comment faire réellement. Pour cela plusieurs solutions existent dont fait partie l'utilisation d'un gestionnaire de source centralisé comme GitHub. L'intérêt majeur de ce genre de solution est qu'elle permet d'héberger gratuitement l'intégralité du code source de son projet, mais aussi d'utiliser un ensemble d'autres outils disponibles comme les pages de wiki, pratiques pour la documentation du projet, ou les systèmes de gestion de tickets qui permettent de gérer les remontées de bug. GitHub étant un système basé sur la gestion versionnée de source Git, il faudra commencer par installer Git sur le poste des développeurs. Pour cela GitHub fournit un mode d'installation pour les 3 systèmes d'exploitation : https://help.github.com/articles/set-up-git

Une fois Git installé, il est nécessaire de se créer un compte sur GitHub en se rendant à l'adresse : https://github.com/login. Sur GitHub, chaque personne peut avoir un rôle différent :

- Owner : propriétaire du projet, c'est lui qui initialise le projet et qui attribue les rôles à d'autres membres de l'équipe,
- Mainteners and collaborateurs : personnes maintenant le projet, et qui le gèrent,
- Contributors : développeurs qui proposent du code,
- Community members : utilisateurs actifs du projet comme ceux qui vous remontent régulièrement des bugs ou qui écrivent la documentation

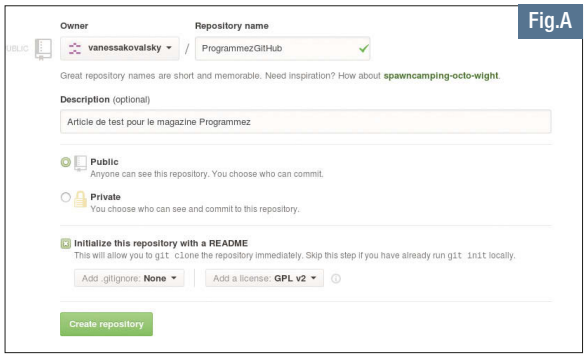


Illustration 1 : Création d'un repository sur GitHub

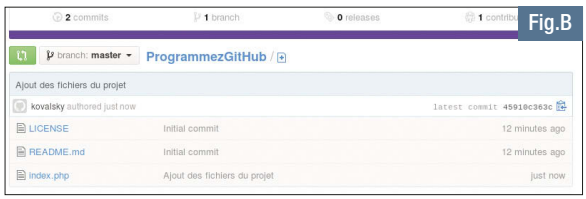


Illustration 2 : Ajout du fichier du projet

Le code et les fichiers nécessaires

Une fois ces rôles identifiés au sein de l'équipe, le propriétaire peut donc commencer à publier sur GitHub. Toutefois, 3 fichiers sont très utiles pour rendre votre projet open source :

- Readme.md : explique l'objectif et les fonctionnalités de votre projet, peut aussi servir de document du projet, c'est par là que l'on conseille aux utilisateurs de commencer lorsqu'ils téléchargent un projet
- Licence : fichier qui précise la licence d'utilisation du projet
- Contributing : fichier expliquant la manière de contribuer au projet par exemple : où commencer, les tags à utiliser dans la liste des tickets pour les débutants, les conventions de codage, etc.

Une fois ces trois fichiers mis en place, il reste à créer le dépôt sur Github et à l'utiliser pour stocker le code du projet. Une fois identifié sur Github, voici à quoi ressemble la page de création d'un repository (espace de stockage pour un projet) : Fig.A.

Une fois le repository créé, nous pouvons faire un git clone en local :

```
git clone https://github.com/vanessakovalsky/ProgrammezGitHub.git/home/ProgrammezGitHub/
```

La copie effectuée en local, les fichiers du projet sont ajoutés dans le répertoire nouvellement cloné, puis les fichiers de notre projet sont ajoutés au repository de GitHub :

```
git add *
git commit -m "Ajout des fichiers du projet"
git push origin master
```

Et nous voyons bien apparaître nos fichiers sur le répertoire distant de GitHub : Fig.B.

Maintenant que le code est en ligne, il reste plusieurs choses à faire : documenter le projet, et surtout le faire connaître et le maintenir.

Communiquer sur son projet : un enjeu important dans l'open source

Une fois le code mis en ligne avec la licence, l'objectif principal va être de faire connaître son projet pour attirer des utilisateurs et des développeurs, et pouvoir créer une communauté autour de ce projet. Pour cela plusieurs choses peuvent faciliter l'adhésion des utilisateurs et des développeurs à notre projet :

- La prise en compte des retours et des besoins utilisateurs, avec la mise en place d'un « release early, release often », qui, en résumé veut dire qu'il vaut mieux publier tôt et souvent pour que les utilisateurs voient que le projet évolue, mais aussi puissent vous faire rapidement des retours, et orienter les décisions à prendre pour que le projet continue à être utilisé. La liste des Issues de GitHub doit donc être revue régulièrement, et des réponses doivent être apportées à chaque demande, même si c'est pour dire que cela n'est pas possible pour l'instant, mais que cela pourra être fait plus tard, ou que vous soyez prêt à accepter un patch ajoutant cette fonctionnalité.
- La rédaction de la documentation est à deux niveaux : pour les utilisateurs, car quoi de plus frustrant qu'un logiciel qui pourrait répondre à un besoin, mais qu'on ne sait pas utiliser et qui n'est pas documenté ? En effet, la documentation à destination des développeurs permettra également d'attirer des contributeurs pour votre projet. Pour cela le wiki de GitHub est un outil formidable, et n'hésitez pas à demander à vos utilisateurs s'ils peuvent contribuer (ne serait-ce que pour corriger les fautes d'orthographe des pages), cela permet qu'ils soient mis en valeur et en fera des community members actifs.
- Les présentations du projet lors de conférences, la publication de fiche dans des annuaires logiciels ou d'articles de blog, par exemple. Celles-ci peuvent être faites par le propriétaire ou un des mainteneurs du projet, mais aussi plus simplement par ses utilisateurs, quoi de mieux qu'une communication d'utilisateur déjà convaincus à d'autres potentiels utilisateurs ? Le projet est maintenant prêt à se développer, bon courage à tous et n'oubliez pas comme le dit si bien Framasoft : « La route est longue, mais la voie est libre ! ».

Vanessa **Kovalsky David**,
Lead développeuse Drupal /PHP chez Webnet
http://vanessakovalsky.net
Membre actif de DrupalFr, et de la promotion du logiciel libre avec l'ALDIL
Lead dev sur le projet Esecouristes : https://github.com/vanessakovalsky/esecouristes

Un code pour les gouverner tous !

Aujourd'hui, il existe clairement 3 plates-formes : Android, iOS, Windows Phone. D'autres tentent l'aventure, telle que Firefox OS. Dans ce dossier, nous allons vous faire découvrir à partir de ce numéro le Nokia X qui a beaucoup agité la communauté des développeurs Windows Phone, nous explorerons les dernières évolutions de Firefox OS, sans oublier les nouveautés de Windows Phone 8.1. Le développeur est un des enjeux des constructeurs et éditeurs : il faut alimenter les App Store.

Dans ce dossier, nous nous focaliserons sur le développement multi-plateformes, c'est-à-dire sur les différentes plates-formes mobiles, à partir d'un code source unique. Nous avons opté pour les outils Xamarin qui permettent de générer du code pour les différents systèmes mobiles, à partir d'un code C# unique. L'avantage est d'être proche du code natif spécifique à chaque plate-forme. Bien entendu, vous n'allez pas générer 100 % du code Objective-C ou Android, mais vous irez bien plus vite dans vos développements.

Bon code.

La rédaction.

10-13-10 © mstav (tornade)

08-06-13 © whanwhanai (usine)

Natif vs HTML 5, un marché élargi : le casse-tête du développeur Web

Aujourd'hui, le marché mobile est très éclaté, mais au niveau mondial, la domination d'Android et de iOS est sans partage. Windows Phone est clairement la 3e plateforme mobile avec une forte poussée dans plusieurs pays. Se posent alors pour le développeur plusieurs questions : comment supporter les 3 principaux systèmes, choisir entre code natif et HTML 5, smartphone et/ou tablette ?

Les rapports de l'éditeur Appcelerator, avec le cabinet IDC, fournissent des éléments intéressants à ne pas négliger.

Faut-il réellement opposer code natif et code non natif ?

Ce débat n'est pas nouveau. Même si le code non natif (typiquement HTML 5 + JavaScript + CSS) a fait des progrès sur les performances et l'accès aux API natives, et donc aux différents composants matériels d'un terminal mobile, il reste encore en retrait sur ces deux points, car tout n'est pas accessible. Le W3C met en avant l'Open Web Platform avec un ensemble dédié aux mobiles. Et cela impose aussi des tests de compatibilité pour les développeurs, notamment à chaque itération système et matérielle.

HTML 5 n'est pas supporté de manière homogène par les navigateurs mobiles, ce qui posera aux développeurs une grande rigueur dans le développement (sur les fonctions à utiliser) et dans les tests. Pour en savoir plus : http://mobilehtml5.org

Cependant, les wrappers se multiplient pour permettre un accès de plus en plus large aux API natives. Et le modèle hybride HTML 5 – natif peut être une solution intéressante à condition d'avoir un accès suffisant aux API, ce qui est parfois loin d'être le cas, notamment sur Android.

La portabilité du code est souvent mise en avant, mais encore une fois si vous cherchez la performance et l'accès complet aux API, le développement natif reste incontournable.

Considérer les évolutions et des tendances, mais se méfier des prédictions

Les smartphones et tablettes évoluent régulièrement ainsi que les systèmes mobiles. Régulièrement, on voit apparaître de nouvelles tendances, de nouvelles technologies, certaines disparaîtront en quelques mois, d'autres resteront. Ainsi, si nous connaissons le dernier rapport Appcelerator /IDC, les prédictions (d'après les remontées des développeurs interrogés) pour 2014 :

- iOS dans la voiture : ça tombe bien, Apple a sorti une édition spécifique,
- Google Glass,
- Windows Phone : même si 50,2 % estiment que ce n'est pas une priorité.

Mais finalement, l'intérêt technologique va aussi dépendre de vos apps et si elles doivent utiliser des fonctions matérielles très récentes. On abeaucoup parlé de NFC, de technologies vocales, mais sont-ce des fonctions utilisées dans la vraie vie ?

Par contre que cela ne vous empêche pas d'être curieux et d'expertiser les nouveautés de toutes les plates-formes.

Une tendance qui s'affirme : l'utilisation d'un back-end Cloud pour les applications mobiles. De plus en plus de services Cloud (stockage, sécurité, vidéo, authentification) proposent des fonctions spécifiques aux mobiles.

Un développeur mobile doit être capable de coder partout !

Soyons clairs : un développeur mobile qui se limite à une plateforme devrait se remettre en question immédiatement.

Sauf à être un dieu du code et avoir une parfaite maîtrise de sa plateforme, le développeur mobile a vocation à passer d'Android, à iOS ou à Windows Phone sur le même projet et dans la même journée.

Il ne doit surtout pas être monosystème, mais très ouvert. Les éditeurs de jeux par exemple cherchent souvent des développeurs multi-plateformes.

Bien plus que le développeur orienté desktop, le développeur mobile exige cette polyvalence par nature, ou doit être capable d'être opérationnel sur une nouvelle plateforme en quelques jours. Il y a 2 ans, il pouvait se permettre d'être iOS ou Android ou les deux. La maturité de Windows Phone impose une 3e plate-forme.

Cette approche multi-systèmes vous permettra aussi d'être agile et de vous adapter à toutes les situations. Par contre, cela impose aussi une grande variété de matériels pour les tests et plusieurs postes de développement.

Côté salaire, une étude d'Urban Linker indique :

- débutant (— 1 an expérience) : 35-38 000 €
- jusqu'à 2 ans d'expérience : 38 – 42 000 €
- au-delà : de 42 à 60 000 € selon la compétence et l'expérience

Nous estimons que le salaire de départ est surcoté ce qui ne reflète pas forcément la réalité du terrain où vous serez plus proche de la rémunération standard du développeur 28-33 000 €. Il n'y a pas forcément de surcote à cause de la spécialisation mobile sauf à avoir un profil pointu et expert.

Lien à lire :

HTML 5 vs Native Code : http://www.developereconomics.com/downloads/can-html5-compete-native/

François **Tonic**

Windows Phone 8.1 : what's new ?

Lors de la Build 2014, Microsoft a dévoilé en grande pompe la nouvelle version de Windows Phone 8.1, très attendue par ses utilisateurs. Les nouveautés sont nombreuses et touchent la quasi-totalité des fonctions du téléphone. Il serait trop long de tout détailler en un seul article, nous allons donc vous montrer, selon nous, les plus significatives : le centre de notifications, l'assistant personnel Cortana, IE11 ... Et nous finirons par un aperçu des nouveautés côté développement.

Principales nouveautés

Avec cette nouvelle mouture, Microsoft a avant tout écouté ses utilisateurs et a rajouté des fonctionnalités issues de ses concurrents comme le « **centre de notifications** » (Fig.1) qui permet d'accéder à différents réglages comme le wifi, le bluetooth ou encore le mode avion et bien évidemment aux notifications mails, sms, facebook. Le plus intéressant est que ce centre de notifications est complétement paramétrable par l'utilisateur conformément à la stratégie de Windows Phone.

Côté look & feel, on notera la possibilité d'afficher trois colonnes sur l'écran Start (Fig.2), même à partir d'un Nokia Lumia 920, téléphone utilisé pour tester WP8.1. Cette option était auparavant réservée aux téléphones à grand écran comme le Nokia Lumia 1320. Les tuiles sont à présent transparentes et l'utilisateur peut définir une image en fond d'écran à partir de ses photos.

Internet Explorer a également droit à son lot de nouveautés avec la possibilité de lire les vidéos directement sur la page web sans passer par le mode plein écran.

Le mode Lecture apporte du confort à l'utilisateur, principalement pour les sites non optimisés, accompagné du *swype* pour revenir à la page précédente en déplaçant le doigt vers la droite. On peut désormais télécharger un fichier, mémoriser ses mots de passe, ouvrir une page en mode navigation privée. Pour finir, on peut épingler son site sur l'écran d'accueil avec une tuile dynamique (http://www.buildmypinnedsite.com/).

Je soulignerai également le nouveau mode de saisie du clavier baptisé « word flow » qui permet d'écrire des messages en glissant le doigt sur le clavier. Après un peu d'apprentissage, on accélère sensiblement le temps de saisie d'un message ou d'un mail.

Côté Entreprise, le support du VPN arrive enfin, et permet de se connecter aux serveurs de son entreprise de manière sécurisée.

Cortana : l'assistant personnel

La nouveauté qui a fait le plus parler d'elle est Cortana ! Cet assistant personnel (Fig.3) vient concurrencer Siri d'Apple et Google Now. Son nom est issu de l'intelligence artificielle du jeu Halo.

Cortana est capable d'effectuer des recherches sur l'intégralité de votre smartphone mais également d'agir, comme par exemple de créer une nouvelle tâche. Cortana utilise en tâche de fond Bing pour vous fournir du contenu adapté à votre demande mais surtout Cortana va apprendre en fonction de vos habitudes ! Les premiers tests sont plutôt concluants, même si vous devez avoir pour l'instant un bon accent anglais, elle (Joe Belfiore utiliser « her » pour parler de Cortana) ne sera disponible en français qu'en 2015.

Fig.1

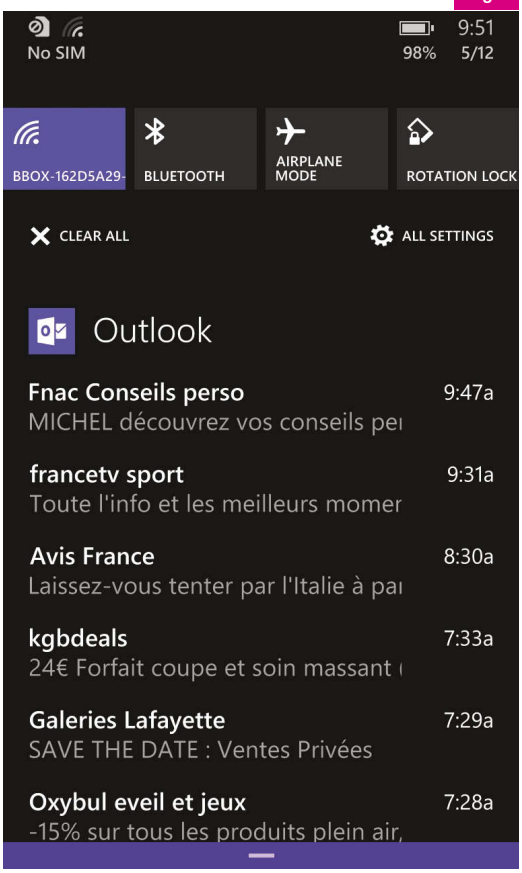
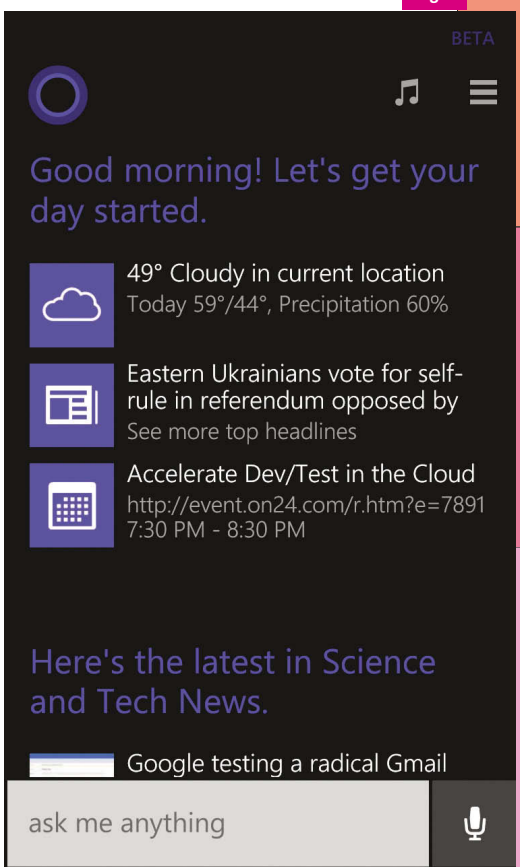


Fig.2

Fig.3



Le point le plus différenciant de ses concurrents est la possibilité de connecter vos propres applications à Cortana pour enrichir ses réponses grâce à la mise à disposition d'API pour les développeurs.
Dès maintenant Microsoft a mis en ligne un exemple d'applications utilisant Cortana, il s'agit de « MSDN Voice Search for Windows Phone 8.1 », téléchargeable à l'adresse : http://bit.ly/1olgoZz.

Comment fonctionnent les API Cortana ?

D'un point de vue « high level », pour intégrer les API Cortana à son application, le développeur doit définir trois éléments (Fig.4) :
Définir les commandes vocales « voice commands » que l'application doit supporter dans un fichier XML. Ces commandes doivent inclure des patterns de phrases comme « Skype, Call [contact_name] ». Ces commandes contiennent des informations complémentaires notamment comment l'application doit être lancée, et ce que l'utilisateur doit voir pendant le chargement de ladite application.

```
<?xml version=»1.0» encoding=»utf-8»?>
<VoiceCommands xmlns=»http://schemas.microsoft.com/voicecommands/1.1»>
  <CommandSet xml:lang=»en-us» Name=»englishCommands»>

    <CommandPrefix>MSDN</CommandPrefix>
    <Example>How do I add Voice Commands to my application</Example>

    <Command Name=»FindText»>
      <Example>Find Install Voice Command Sets</Example>
      <ListenFor>Search</ListenFor>
      <ListenFor>Search for {dictatedSearchTerms}</ListenFor>
      <ListenFor>Find</ListenFor>
      <ListenFor>Find {dictatedSearchTerms}</ListenFor>
      <Feedback>Search on MSDN</Feedback>
      <Navigate Target=»MainPage.xaml» />
    </Command>
  …
```

Enregistrer le fichier XML contenant les commandes vocales lors du démarrage de l'application. Ces commandes peuvent bien sûr être mises à jour ensuite, mais vous devez « installer » les commandes au premier démarrage de l'application.

```
private async void RegisterVoiceCommands()
{
  // SHOULD BE PERFORMED UNDER TRY/CATCH
  Uri uriVoiceCommands = new Uri(«ms-appx:///vcd.xml», UriKind.
  await VoiceCommandService.InstallCommandSetsFromFileAsync
(uriVoiceCommands);
}
```

Gérer l'activation de la voix. Dès que Cortana a reconnu votre application dans les propos de l'utilisateur accompagnée d'une commande préalablement enregistrée, les paramètres sont passés à une page de votre application indiquant la commande vocale reconnue et l'ensemble du texte.

```
protected override void OnNavigatedTo(System.Windows.Navigation.
NavigationEventArgs e)
{
  base.OnNavigatedTo(e);
  if (e.NavigationMode == System.Windows.Navigation.Navigation
Mode.New)
  {
    string recoText = null; // What did the user say? e.g. MSDN,
 «Find Windows Phone Voice Commands»
    NavigationContext.QueryString.TryGetValue(«reco», out reco
Text);
    string voiceCommandName = null; // Which command was recognized
in the VCD.XML file? e.g. «FindText»
    NavigationContext.QueryString.TryGetValue(«voiceCommand
Name», out voiceCommandName);
    string searchTerms = null; // What did the user say, for
named phrase topic or list «slots»? e.g. «Windows Phone Voice
 Commands»
    NavigationContext.QueryString.TryGetValue(«dictatedSearch
Terms», out searchTerms);
    switch (voiceCommandName) // What command launched the app?
    {
      case «FindText»:
        HandleFindText(searchTerms);
        break;
      case «nlpCommand»:
        HandleNlpCommand(recoText);
      break;
    }
  }
}
```

Dans le prochain numéro de programmez!, nous vous montrerons en détail ces API.
Cortana devrait arriver également sur les autres plateformes de Microsoft, à savoir Windows 8 et Xbox.

Application universelle et Windows Runtime

L'un des changements majeurs de cette nouvelle version de Windows Phone est le support des applications universelles. Avec cette nouveauté, il devient possible de faire tourner son application sur Windows via une Windows App et bientôt sur Xbox One. Et l'impact sur la partie développement est important puisque ce changement se base sur le support d'un nouveau Framework : le Windows Runtime.
Sachant que les applications tournant sur la version 8.0 de l'OS mobile utilisent une version adaptée de Silverlight, la question qui vient immédiatement à l'esprit concerne la migration vers cette nouvelle platefor-

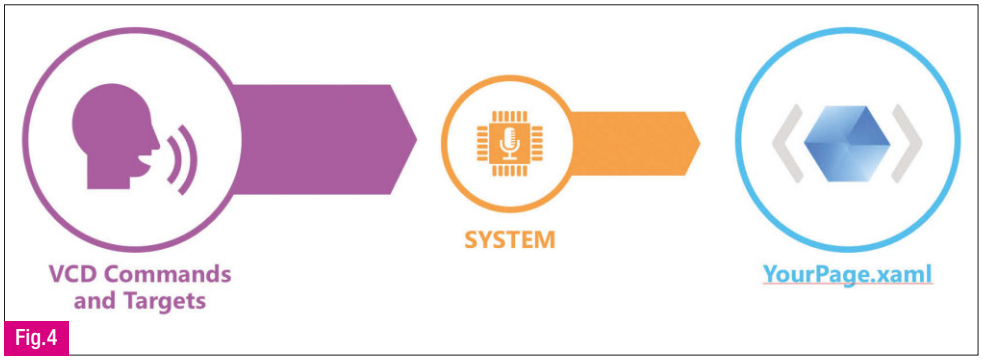


Fig.4

me. Microsoft n'abandonne pas pour autant Silverlight et permet une transition en douceur en proposant une version 8.1 de ce Framework. Il est donc très facile de mettre à jour une application existante. De plus Silverlight 8.1 supporte quelques fonctionnalités (copier/coller, personnalisation du fond d'écran de l'écran de verrouillage, alarmes et rappels, etc.) qui ne sont pas encore présentes dans le Windows Runtime.
Lors de la mise à jour ou création de votre application il faudra donc faire un choix : opter soit pour Silverlight 8.1, soit pour Windows Runtime, chacun ayant ses avantages et ses inconvénients.
En ce qui concerne Silverlight, l'utilisation de ce Framework permet d'exploiter toutes les capacités de l'OS et facilite la mise à jour d'une application existante.
De son coté, Windows Runtime permet de créer des applications universelles facilement. Il convient donc d'étudier attentivement ces deux options afin de prendre la bonne décision.

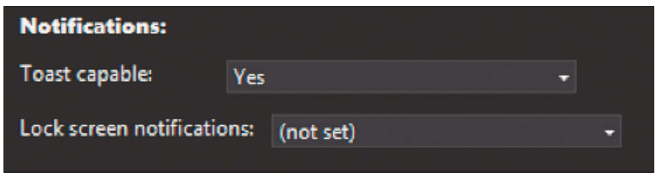
Nouveauté sur le développement d'application

Nous allons maintenant étudier quelques concepts existants dont l'implémentation change avec l'utilisation de Windows Runtime. L'ensemble des exemples qui vont suivre sont issus d'une application disponible à l'adresse suivante :

https://github.com/melcom/WindowsPhone81

Les notifications Toasts

La première étape avant de pouvoir utiliser les notifications et de modifier le Package.appxmanifest du projet en sélectionnant oui dans la rubrique « Toast capable » :



Voici ci-dessous le code qui permet de générer une notification. Comme on peut le constater, la création d'un toast se base sur l'utilisation d'une structure XML qui sert à décrire ses différentes propriétés. Pour changer le texte par exemple, il suffit de modifier le contenu XML pour y insérer son message :

```
XmlDocument xml = ToastNotificationManager.GetTemplateContent
(ToastTemplateType.ToastImageAndText01);

XmlNodeList texts = xml.GetElementsByTagName(«text»);
texts[0].AppendChild(xml.CreateTextNode(«Description»));

var toast = new ToastNotification(xml);

ToastNotificationManager.CreateToastNotifier().Show(toast);
```

Il y a quelques nouveautés liées à la présence du centre de notification. La première est la gestion de l'historique. Il est désormais possible de gérer ce dernier en effaçant toutes les notifications en attente :

```
ToastNotificationHistory history = ToastNotificationManager
.History;
history.Clear();
```

Il est également possible d'envoyer une notification sans afficher de pop-up, grâce à la propriété SuppressPopup. La notification sera alors uniquement consultable dans le centre de notification. On peut également lui donner une date de péremption. Lorsque cette date sera dépassée, elle disparaitra du centre de notification.

```
toast.SuppressPopup = true;
toast.ExpirationTime = DateTimeOffset.UtcNow.AddSeconds(30);
```

Password Vault

Une autre nouveauté disponible est la possibilité de stocker de façon sécurisée les différents noms d'utilisateur et mots de passe utilisés par l'application. En effet plutôt que de stocker ces informations dans les paramètres, vous pouvez utiliser la classe PasswordVault comme le montre l'exemple ci-dessous :

```
PasswordVault vault = new PasswordVault();
PasswordCredential credential = new PasswordCredential
(resource, User.Text, Password.Password);
vault.Add(credential);
```

Les données sont alors stockées de façon permanente et cryptée. On peut par la suite les récupérer pour les utiliser. Il faudra néanmoins bien veiller à gérer les exceptions lors de la récupération, car si aucun mot de passe n'est trouvé, la méthode « Retrieve » lèvera une exception pour le notifier.

```
try
{
 IReadOnlyList<PasswordCredential> credential = vault.Find
AllByResource(resource);

 foreach (PasswordCredential pc in credential)
 {
   RetPassword.Text = vault.Retrieve(resource, pc.UserName)
.Password;
 }
}
catch (Exception)
{
 RetPassword.Text = «Aucun mot de passe trouvé.»;
}
```

Geofencing, Tiles & Badges

L'application de démonstration comporte également d'autres nouveautés qui ne sont pas décrites dans cet article. Je vous laisse donc les découvrir en parcourant le code disponible sur Git Hub.

Next...

Pour conclure, Windows Phone 8.1 enrichit dans le bon sens l'OS mobile de Microsoft qui essaie de rattraper son retard commercial sur ses concurrents. En tout cas, les développeurs peuvent d'ores et déjà s'en donner à cœur joie et, grâce à la version preview, commencer à développer les applications « nouvelles générations ».

Michel **Hubert**
Manager | MVP Azure chez Cellenza
Cellenza - Software Development Done Right

Pierre-Henri **Gache**
Consultant Senior chez Cellenza
Cellenza - Software Development Done Right

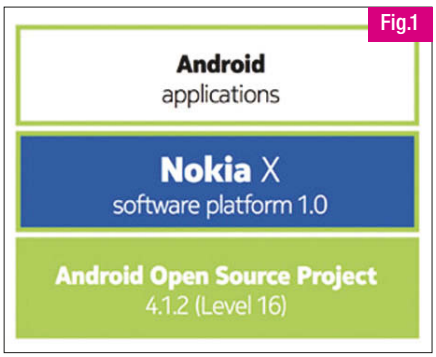
Porter une application Android existante sous Nokia X

Dévoilée en ce début d'année, la plateforme Nokia X est la première tentative de Nokia dans l'univers Android. Destinée à se positionner sur l'entrée de gamme, la plateforme Nokia X se base sur le projet Open Source Android (AOSP) que Nokia a customisé pour l'adapter à ses fins. Le résultat est une plateforme qui propose une compatibilité forte avec les applications Android existantes afin de limiter les efforts de portage des développeurs Android et de favoriser l'essor de son store d'applications Nokia X. Dans cet article, nous nous proposons de détailler les différentes étapes à suivre pour porter une application Android existante sous Nokia X.

L'annonce de Nokia X aura pu paraître assez surprenante de prime abord. En effet, le rachat récent de Nokia par Microsoft semblait lier le destin de Nokia uniquement à l'OS Windows Phone. Néanmoins, le projet Nokia X était en route depuis de longs mois et son positionnement a du sens puisqu'il vise avant tout un marché d'entrée de gamme là où la gamme Lumia se positionne sur le milieu et le haut de gamme.

Architecture Nokia X

Le résultat du travail de Nokia est une plateforme Android adaptée à l'univers du géant Finlandais. Nokia X se base sur la version 16 de l'AOSP, ce qui implique que la plateforme supporte la version 4.1.2 d'Android (figure 1).
Le projet Open Source Android correspond à la version Android utilisée massivement par les smartphones Android et développée par Google notamment à cela près qu'elle ne comporte aucun des services propriétaires de Google. Sur cette base, Nokia a ajouté ses services propriétaires qui sont le pendant des services Google présents sur les smartphones Android. On retrouve ainsi Nokia In-App Payment pour la gestion des achats in-app, l'API HERE Maps permettant d'utiliser la solution de cartographie HERE Maps, et Nokia Notifications qui autorise l'envoi de notifications Push au sein des applications Nokia X. Les développeurs Android habitués à l'OS de Google auront tout de suite fait le rapprochement avec les services de Google suivants : Google In-App Billing, Google Maps et Google Cloud Messaging (GCM).

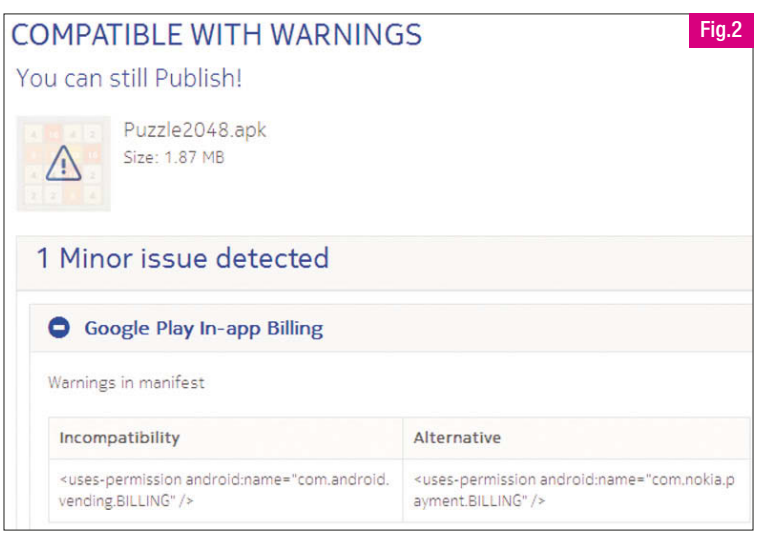


Architecture Nokia X

Compatibilité

Du fait de son architecture, la plateforme Nokia X est compatible avec près de 75% des applications Android existantes. Pour ces dernières, aucune modification ne sera à effectuer et elles pourront être publiées en l'état sur le store Nokia X. Les principales sources d'incompatibilité étant liées aux applications reposant sur des services propriétaires de Google tels que Google In-App Billing, Google Maps et GCM. Pour celles-ci, il est nécessaire de modifier leur code en remplaçant les APIs Google par les APIs Nokia équivalentes. Il est bon de souligner toutefois que ces applications pourraient être publiées sur le Nokia Store mais bien évidemment ne seraient pas pleinement fonctionnelles ce qui serait très pénalisant pour l'utilisateur et la réputation du développeur.
Le processus d'analyse de la compatibilité d'une application Android existante sur la plateforme Nokia X est facilité par la mise à disposition par Nokia d'un utilitaire en ligne analysant un APK de manière statique, permettant d'obtenir un aperçu de la compatibilité d'une application existante. Cet outil est accessible ici : http://developer.nokia.com/nokia-x/analyse/ . Cependant, il convient de rester prudent car cet outil semble se baser principalement sur le contenu du manifest Android et des permissions déclarées pour vérifier si une application utilise des services Google. Ainsi, une application proposant un Live Wallpaper passe ce test mais ne pourra pas être compatible étant donné que le smartphone Nokia X ne supporte pas cette fonctionnalité.
En considérant une implémentation du jeu phare du moment 2048 disponible sur le Google Play Store (https://play.google.com/store/apps/details?id=com.ssaurel.puzzle2048) et recourant aux achats in-app via la solution de Google, l'analyse de la compatibilité avec Nokia X rend le résultat présenté à la figure 2.

L'utilitaire remarque la déclaration de la permission com.android.vending.BILLING et propose donc d'utiliser Nokia In-App Payment à la place. Dans tous les cas, il est préférable de tester



Analyse de la compatibilité

ensuite son application de manière dynamique soit sur un terminal Nokia X physique soit via le service de Remote Device Access (RDA) proposé gratuitement par Nokia sur son site : http://developer.nokia.com/resources/remote-device-access .
Enfin, le développeur Android découvrira avec plaisir que les APIs spécifiques proposées par Nokia se calent très précisément sur les APIs exposées par les services de Google ce qui doit réduire le travail de portage spécifique à un maximum de 8 heures selon le géant Finlandais.

Portage pratique

Les fondements techniques et les spécificités de la plateforme Nokia X passés en revue, nous rentrons dans le vif du sujet en portant une application Android existante vers la plateforme Nokia X. Disponible ici : https://play.google.com/store/apps/details?id=com.app.liveweatherlite, l'application Live Weather propose la météo du jour ainsi que les prévisions météos des jours prochains.
Elle ne recourt à aucun service spécifique de Google et rentre de fait dans la catégorie des 75% d'applications existantes compatibles sans modifications.
Néanmoins, nous suivons le processus classique de portage en commençant par analyser la compatibilité via l'utilitaire de Nokia.
Ce dernier confirme notre sentiment premier puisqu'il ne révèle aucune incompatibilité à priori. La phase de remplacement des services Google par les services Nokia X équivalents peut donc être évitée ici. Une fois le lien au store de Google retiré, pour la présentation d'une popup d'évaluation de l'application redirigeant vers le Play Store, on peut générer l'APK de Live Weather à destination de la plateforme Nokia X. La troisième étape consiste ensuite à tester l'application en condition réelle sur Nokia X que ce soit via le service RDA de Nokia ou sur un périphérique physique. Réalisés sur un smartphone Nokia X, les tests valident le fonctionnement de Live Weather (figure 3).

Publication

L'application validée sous Nokia X, il reste enfin à la publier sur le store de Nokia dédié aux appareils Nokia X. Pour ce faire, il est nécessaire de créer un compte développeur gratuit à l'adresse suivante : http://developer.nokia.com/publish . Le compte créé, on démarre l'ajout de Live Weather en allant dans l'onglet Publish et en cliquant sur la zone «Publish app». Comme cela se fait habituellement, le développeur doit alors saisir un certain nombre d'informations concernant son application telles que son titre, sa description, son mode de distribution (gratuit ou payant), sa catégorie ou encore des screenshots permettant aux utilisateurs d'en avoir une vision rapide et précise.
Une fois ce travail effectué, nous pouvons lancer le processus de soumission sur le store Nokia. L'application va alors être validée a priori et rentrée en phase de QA où des équipes dédiées vont effectuer un certain nombre de tests. Ce point est très important puisqu'il permet, à l'instar de ce que propose Samsung sur son store, d'avoir une vraie phase de qualification gratuite et renvoyant un rapport de test précis sur les problèmes éventuellement rencontrés. Idéal pour corriger d'éventuels problèmes grossiers avant que les utilisateurs ne les découvrent.
Ceci constituant un avantage des stores Nokia et Samsung pour Android, en comparaison du Play Store de Google qui se contente d'éventuelles revues a posteriori.
L'application peut alors être téléchargée directement depuis le Nokia Store (figure 4).

Conclusion

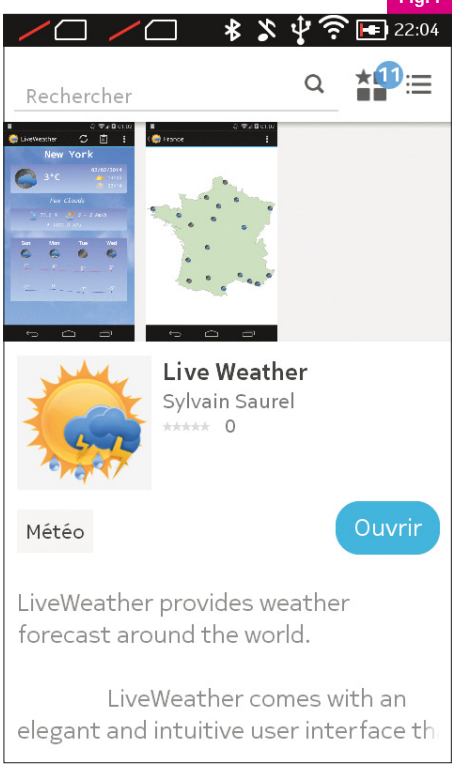
Cet article aura mis en exergue la qualité du travail réalisé par les équipes de Nokia pour créer une plateforme basée sur Android intégrant ses services propriétaires, tout en simplifiant au maximum le travail de portage des applications Android existantes.
En effet, l'enjeu se situe bel et bien là pour Nokia et sa plateforme Nokia X puisque l'on connaît l'importance d'avoir un store applicatif bien fourni à proposer à ses utilisateurs pour qu'une gamme d'appareils puisse s'imposer.
Notre exemple, correspondant au cas le plus répandu, aura montré qu'il est possible de porter une application Android existante en tout juste 1 heure sur Nokia X si celle-ci n'utilise aucun service Google propriétaire. Pour les 25% d'applications restantes qui utilisent notamment des services Google spécifiques, il conviendra d'effectuer un travail de portage facilité par la correspondance volontaire des APIs Nokia équivalentes. Là encore, les équipes de Nokia ont fait preuve de pragmatisme en mettant à disposition du développeur des APIs se calquant sur celles de Google pour faciliter au maximum le portage et la prise en main. Devant la facilité de portage des applications existantes vers Nokia X et le succès que semblent connaître les smartphones de la gamme depuis son lancement, la plateforme représente désormais une opportunité de plus pour des développeurs Android toujours en recherche de stores alternatifs pour se démarquer et ne plus dépendre uniquement du Google Play Store.
Le mois prochain, nous irons plus loin sur Nokia X.

Sylvain **SAUREL**
Ingénieur d'Etudes Java / Java EE
sylvain.saurel@gmail.com

Fig.3

Live Weather sous Nokia X

Fig.4



Live Weather sur le Nokia Store

XAMARIN, l'extension multiplateforme de .Net

1ere partie

XAMARIN est une technologie qui a le vent en poupe ! Sur un marché ultra dynamique tel que celui du mobile, disposer d'une solution multi-plateformes performante est devenu incontournable. Ce tour d'horizon synthétique à 360 degrés, avec un focus particulier sur l'implémentation des UI représentant le premier challenge pour le développeur, vous permettra d'en mesurer tout le potentiel.

A la question « Qu'est-ce que XAMARIN » ? J'ai souvent tendance à apporter trois réponses : la première fait référence à la genèse de XAMARIN issue du fruit de plusieurs années de développement au travers du projet Mono qui permettait de créer des applications DotNet sous Unix. Ce projet a été étendu par son fondateur Miguel de Icaza aux environnements OSX, iOS et Android. Nous parlons donc d'une solution éprouvée par une communauté existante ayant plusieurs années d'expériences dans le portage de .Net sur d'autres environnements que Windows.

La seconde réponse consiste à positionner XAMARIN dans l'écosystème Microsoft. XAMARIN est une **extension** à l'écosystème Microsoft .Net que nous connaissons tous et qui comprend le Framework .Net, Visual Studio et Team Foundation Server. Vous pouvez vous passer de Visual Studio et Team Foundation en utilisant le studio de développement de XAMARIN mais vous perdez toute la puissance qu'apportent ses produits dans la productivité de vos applications. Par **extension**, j'entends aussi surtout souligner que XAMARIN ne couvre pas le développement Windows Phone et Windows Modern UI car ces derniers sont bels et bien gérés par Visual Studio.

La troisième réponse permet de définir en trois points clés ce qu'est techniquement XAMARIN : un SDK fournissant une API qui couvre en C# ou F# quasiment 100% de l'API iOS et Android ; un compilateur qui permet de traduire votre code .Net vers les cibles iOS et Android et un outil proposant des extensions à Visual Studio pour vous permettre de designer, debugger, tester et compiler vos applications mobiles.... Fig.1.

Globalement, l'intégralité de votre développement peut se dérouler sous Visual Studio et Team Foundation. Vous allez toutefois avoir besoin d'un poste sous OS X pour compiler vos applications iOS. Sur le Web, de nombreuses offres de Cloud pour MAC sont à votre disposition (http://cloudmac.net/ ou http://macminicloud.net/). Avec ces offres, vous pourriez connecter l'agent de build MAC de votre ALM Team Foundation Server. Au passage, l'ALM dans le cadre du développement mobile est un sujet sérieux que je vous proposerai d'aborder dans un prochain billet.

Les wrapper d'API mobile et les compilateurs

Quand on parle de WRAPPER aux API iOS et Android, la première question qui vient à l'esprit, c'est naturellement la capacité de XAMARIN à actualiser son API dès la sortie des nouvelles versions d'OS. C'est très rapide. L'équipe des « binder » est organisée et elle profite des versions alpha/beta des nouveaux OS pour actualiser les SDK. Concernant les compilateurs, le sujet est beaucoup plus attrayant techniquement parlant. Il y a en effet deux fonctionnements différents pour les compilateurs iOS et Android. Pour iOS, le compilateur traduit le code .Net en code binaire ARM en faisant abstraction de l'utilisation et des atouts que pourraient apporter une machine virtuelle. L'usage d'une machine virtuelle est strictement interdit par APPLE. Toutefois, dans cette configuration, XAMARIN apporte des outils dans son Framework pour gérer des fonctions telles que le « garbage collector ».

Pour Android, le compilateur traduit le code .Net en code IL. Vous comprenez dès lors que sous Android le code XAMARIN est exécuté dans une machine

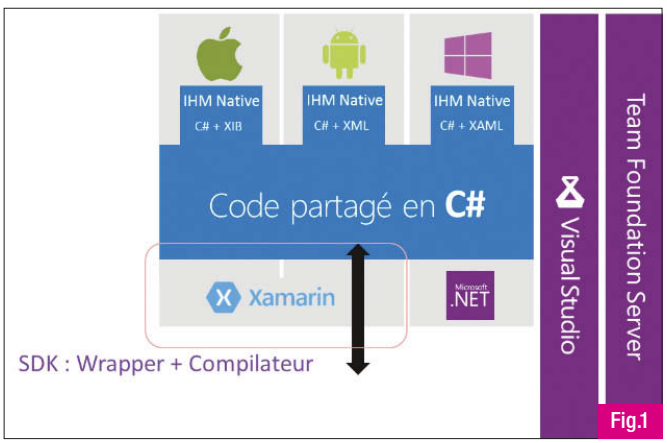


Fig.1

virtuelle .Net. Android étant développé en Java, ce dernier propose sa propre machine virtuelle. En conséquence de quoi, la machine virtuelle de XAMARIN est exécutée sous la machine virtuelle d'Android. Quid des performances ? Force est de constater que le fil d'exécution du code .Net est plus rapide que le fil d'exécution du code Java. L'overload s'opère dans l'intersection des deux machines virtuelles, lorsque la machine virtuelle XAMARIN/.Net dialogue avec la machine virtuelle Java. Plus votre fil d'exécution est long, plus vos performances seront meilleures en XAMARIN vs NATIF ANDROID Fig.2.

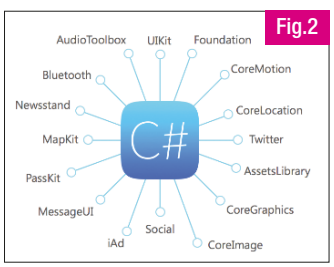


Fig.2

Au regard des deux modes de compilation, chacune des plateformes mobiles induit par conséquence un périmètre de contraintes différent. Pour bien comprendre les diverses contraintes et les anticiper dans vos implémentations, une documentation complète est disponible :

Android :
http://docs.xamarin.com/guides/android/advanced_topics/limitations/offline.pdf
iOS : http://docs.xamarin.com/guides/ios/advanced_topics/limitations/offline.pdf

Implémentation et réutilisation du code

Après toutes ces précisions nous pouvons commencer à entrer dans le vif du sujet pour parler d'implémentation. Pour ce faire il faut disposer d'une licence XAMARIN. La licence est attribuée par développeur et par cible (iOS / Android). Pour couvrir les plateformes mobiles vous devrez disposer de deux licences. Vos applications sont « runtime free », sans licences supplémentaires et sans limite sur le nombre d'applications.

Pour débuter, nous vous conseillons d'utiliser les exemples proposés par XAMARIN nommés « Pre-Build ». Il s'agit d'exemples de code que vous pouvez directement télécharger sur GitHub à l'adresse suivante : https://github.com/xamarin/prebuilt-apps.

Pour cet article, nous utiliserons l'exemple FieldService pour illustrer la structuration d'un projet mobile. L'intérêt de cette solution c'est qu'elle

comprend une Apps iOS, Android et WinRT.
Il y a plusieurs manières de construire son code mobile. Comme expliqué plus haut, XAMARIN est une extension. Il est fort possible que vous ayez déjà développé des applications Windows Modern UI ou Windows Phone. Il est donc intéressant dans cette situation d'isoler des parties de cette implémentation dans une librairie (une PCL) pour rendre cette partie du code réutilisable pour les autres projets mobiles. Voici comment est constitué un projet multiplateformes mobile : Fig.3.
Sous Visual Studio voici le rendu de FieldService issu du pré-build XAMARIN :

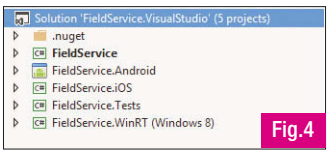


Fig.4

Fig.4. Dans un premier temps, nous pouvons constater trois projets mobiles (figure 4) : iOS, Android et Windows Phone. Il s'agit de projets contenant toute la logique UI des applications mobiles. La logique métier est quant à elle isolée dans une librairie commune, notre quatrième projet. Il y a plusieurs manières de partager du code. La figure 3 montre l'approche PCL (Portable Class Library). Alors que dans l'exemple du pré-build FieldService de XAMARIN c'est l'approche « fichiers partagés » qui est mise en avant. Personnellement l'approche PCL me semble particulièrement intéressante car elle « isole » mieux nos implémentations. Elle impose toutefois de-facto le partage d'un subset .Net compatible avec toutes vos plateformes. Le choix de l'approche PCL est donc le choix du plus petit dénominateur commun .Net compatible avec les plateformes mobiles ciblées ! Voici les différentes approches pour partager le code entre tous vos projets mobile :

Méthode	Avantages	Inconvénients
Pragma	Partage d'un seul fichier, mutualisation d'un « atome » de code.	Code illisible sur des volumes de code.
File Links	Partage d'un seul fichier	Une modification pour une plateforme peut causer des erreurs sur un autre projet qui référence le même fichier. L'erreur sera visible uniquement à la compilation ou à l'exécution.
PCL	Une seule assembly partagée. Code bien structuré	Périmètre réduit de l'API .NET (subset DotNet pour les cibles mobiles).
Partial class	Possibilité d'avoir une partie d'une classe partagée et une autre partie spécifique.	Lisibilité et maintenance du code du fait de disposer d'une classe atomisée sur plusieurs projets.
Inheritance	Partage d'une partie du code entre différentes classes	Vous douteriez de mon objectivité ;-)

Pour continuer à avancer dans la compréhension du développement multiplateforme mobile nous devons répondre à trois points essentiels :
- Les patterns du développement mobile,
- Outillage pour créer des UI / IHM,
- Patterns UI de développement sur les plateformes cibles (iOS, Android).

Les patterns mobiles

Le développement en C# d'application mobile XAMARIN, du fait que le Framework couvre 100% de l'API mobile ciblée, signifie également que vous devez maitriser les patterns d'implémentations iOS et Android. Si vous disposez déjà d'une expérience mobile iOS et/ou Android, c'est un excellent avantage. Il ne suffit pas de disposer d'une expérience C# mais véritablement de comprendre et de maîtriser les plateformes mobiles. Les enjeux sont véritablement nombreux et l'usage d'une seule filière de développement permet de réduire considérablement les contraintes d'ingénierie pour vous concentrer sur les enjeux « métiers et mobile ». Voici les 4 Piliers sur lequel il est fondamental de s'attarder :
- **Expérience utilisateur** de la plateforme mobile cible,
- **Batterie** : on l'oublie souvent, votre architecture et vos implémentations doivent prendre en compte cet aspect. Imaginez une application mobile qui boucle en tâche de fond (ping, synchro, etc...), et, par conséquent fait descendre drastiquement la batterie à 2h d'autonomie ! C'est juste impossible. D'autant plus que nous voyons apparaître sur les mobiles des outils de reporting de consommation d'énergie permettant à l'utilisateur d'arbitrer et de « killer » ou réduire l'usage d'une application énergivore.
- **Sécurité** : c'est un vaste sujet mais vos projets doivent impérativement refléter la sécurité sur toutes les couches. On parle de persistance, mémoire, flux, authentification et autorisation. L'enjeu est tellement majeur que chez RedFabriQ nous avons développé un centre d'expertise dans ce domaine avec notre partenaire VigiTrust pour intégrer cette dimension dans nos développements.
- **Performance** : il s'agit d'un sujet d'architecture, de conception et de développement. Au même titre que la sécurité, la performance est déterminante. D'anciennes pratiques de développement refont surface. La médiocrité de la performance dégrade l'expérience utilisateur.

La création des IHM : outillage

La création des IHM mobiles iOS et Android sous XAMARIN est possible via les outils de développement de XAMARIN ou Visual Studio. A ce jour, le plug-

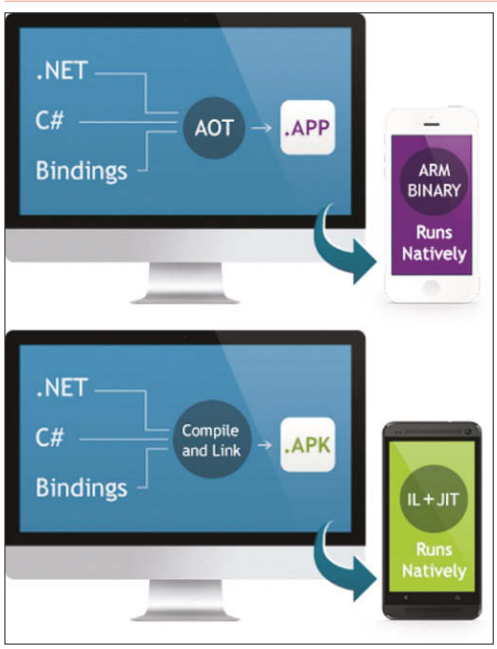
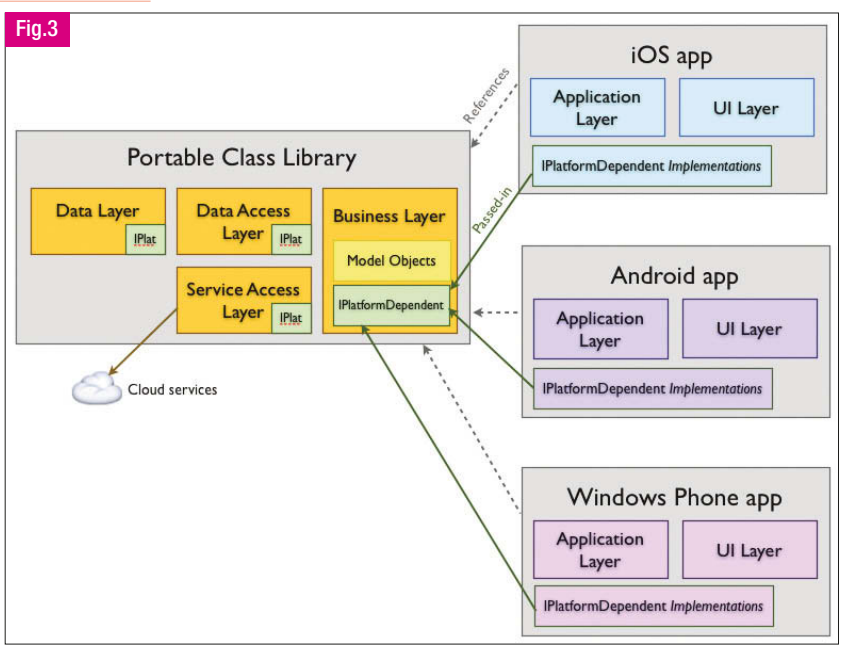


Fig.3



in XAMARIN pour Visual Studio couvre uniquement ANDROID. Toutefois, lors de la dernière BUILD 2014, Miguel de Icaza, a présenté une pré-version de l'éditeur d'IHM pour iOS sous Visual Studio. Seule la compilation pour iOS nécessite une machine OSX.
Voici un exemple d'édition d'IHM Android sous Visual Studio. Le plug-in XAMARIN vous permettra par conséquent de créer vos vues (ficher XIB pour iOS, Fichier AXML pour Android). Une fois les vues créées, vous allez devoir implémenter les traitements dans des « contrôleurs » qui devront hériter des classes dédiées à cet effet et exposées par XAMARIN.
Nous y reviendrons Fig.5.
L'environnement Visual Studio ci-dessus et le plug-in XAMARIN pour Android proposent un designer et des outils dans la TOOLBOX avec une collection de contrôles UI pour bâtir notre écran.

Les patterns UI

XAMARIN couvrant les API iOS et Android, cela nous conduit à devoir respecter les règles d'implémentation des différentes plateformes comme le ferait une implémentation native. Certains Framework comme QT Mobile proposent des abstractions aux UI mobiles.
Cette approche permet de développer une seule fois l'interface graphique pour tous les mobiles. En contrepartie, votre IHM ne respecte pas les standards de chaque plateforme car elle fagote l'expérience utilisateur. Le graal serait que XAMARIN propose également un complément de Framework UI universel pour offrir le choix aux développeurs.
Aujourd'hui en tant que développeur, vous devez impérativement comprendre les patterns UI des plateformes mobiles que vous allez adresser. Pour faire simple, voici la liste non exhaustive des modèles d'implémentation qui sont sous-jacents aux VIEW :

- IOS : UIViewController, UITableViewControleur, UITabViewControleur...
- Android : Actitity + Fragment + Animation etc...
- Windows : code Behind

Pour améliorer votre productivité et la réutilisation de votre code, je vous recommande d'utiliser un Framework MVVM tel que MVVMCross.
L'implémentation d'un modèle MVVM vous permettra, de partager une grande partie du code de vos IHM.
Au regard de la figure 6, les vues créées plus haut sont représentées par le carré bleu. Il nous reste maintenant à implémenter les VIEW MODEL et le BINDING entre la VIEW et la VIEW MODEL et ensuite associer votre VIEW MODEL à votre service. Voici un tableau qui, pour chaque implémentation UI, révèle les classes qu'il faut surcharger dans les projets correspondant à chacune des cibles mobiles :

Layer du MVVM	Sous iOS	Sous Android	Sous Windows Universal Apps
VIEW	Fichier XIB	Fichier AXML	Fichier XAML
VIEWMODEL	INotifyPropertyChanged	INotifyPropertyChanged	INotifyPropertyChanged
MODELES	Classe Object	Classe Object	Classe Object
CONTROLER / Implémentations tiers	UIViewController, UITableViewControleur, UITabViewControleur...	Actitity, Fragment, Animation...	Code Behind

Créer une IHM est finalement la partie la plus « consommatrice » en effort de développement. Alors n'hésitez pas à investir de votre temps pour en maitriser les rouages, les outils et les solutions. Je vous propose de passer au reste de l'implémentation. Pour ce faire nous allons utiliser MVVMCross qui vous fera gagner énormément de temps. Pour rappel, MVVMCross est un Framework MVVM multiplateformes C#. Les plateformes ciblées sont : IOS, Android, Windows Phone8, Windows Store et WPF
Les atouts de l'utilisation de MVVMCross

- **Portabilité** : il est important d'utiliser des librairies portables autant que possible, qu'il s'agisse des modèles de vues, des modèles, des services et de la logique métier.
- **Maintenabilité** : utilisation d'un Framework d'inversion de contrôle. Avec MVVMCross on peut utiliser l'injection de dépendance pour avoir des applications robustes et facilement maintenables.
- **Facilite les tests unitaires** : l'apport du couplage faible proposé par MVVMCross permet de faciliter le développement de tests unitaires.
- **Utilisation du pattern d'architecture MVVM** : il est utile de découpler les vues des modèles en utilisant le binding, ce qui permet d'avoir des applications lisibles et flexibles.
- **Utilisation des plugins** : MVVMCross fournit un ensemble de plugins multi-plateformes comme le plugin File ou Network qui offre un certain nombre de fonctionnalités.

Une solution XAMARIN utilisant le Framework MVVMCross consiste en :
Un projet Core qui contient tous les modèles de vues, les modèles et les services partagés,

- Un projet UI pour chaque plateforme ciblée,
- Optionnellement on peut avoir des plugins ou une partie du code est défini dans une librairie portable et l'autre partie dans une librairie spécifique à la plateforme.
- Classes d'une librairie Core :
  - App : classe responsable du démarrage de l'application, elle charge les modèles de vues et les services.
  - Start : Classe responsable de la définition de modèle de vue. C'est la première classe à s'exécuter.
  - ViewModels : les modèles de vue créés pour l'application
  - Services : tous les services créés pour l'application
- UI
  - Classe Application de chaque plateforme responsable du cycle de vie de chaque application,
  - Setup MVVMCross responsable du lancement de l'application,
  - Views : les vues de l'application spécifique à chaque plateforme,
  - Presenter : code personnalisé des vues pour gérer les interactions des vues

Dans le prochain numéro, nous passerons au code !

Zaak **Chalal**
CEO de Red Fabriq

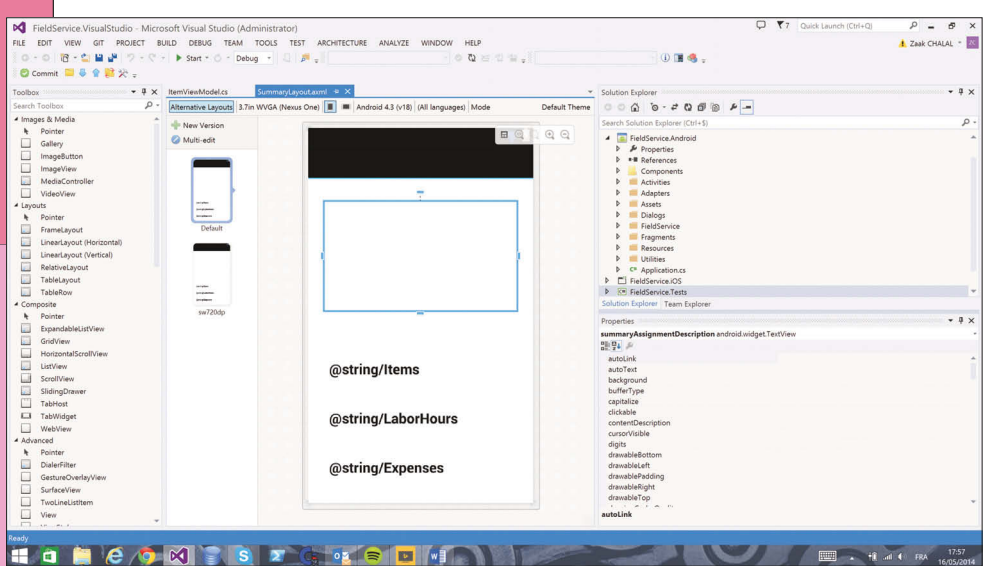
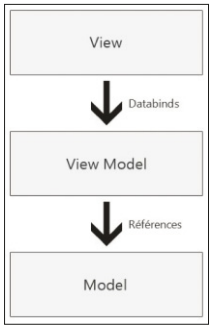


Fig.5

1 application, 1 code partagé, 5 plateformes !



La maitrise du développement mobile sur différentes plateformes est plus ou moins complexe dans la mesure où, pour offrir la meilleure expérience utilisateur possible, tout en ayant des performances natives, il faut connaître différentes technologies et leurs outils, tels que les couples Objective-C et Xcode, Java et Eclipse, C#/XAML et Visual Studio...

Chez Happly-Sfeir , nous avons relevé le défi. Nous avons cherché à maîtriser le sujet. **Xamarin** nous a aidé à étoffer nos compétences.

OUTILS ET TECHNOLOGIES

Pour commencer, nous avons dû nous assurer que l'on disposait des ressources nécessaires pour mener à bien notre mission...

Les prérequis techniques

Qui dit développement avec Xamarin dit connaissance de la programmation objet, du Framework .NET et du langage C#. Au niveau des patterns, je vous recommande fortement un minimum de maîtrise des patterns MVC (pour Android et iOS) et MVVM (pour Windows RT et Windows Phone).

Note : *Si le pattern MVVM vous est inconnu, je vous recommande la lecture de «MVVM. De la découverte à la maîtrise: WPF, Silverlight, WP7» de Jonathan Antoine et Thomas Lebrun.*

Niveau ergonomie de vos futures applications, il est essentiel de connaître les best-practices pour chacune des plateformes, ainsi que la maîtrise des outils pour faire de belles applications.
Grâce à la possiblité que nous offre Xamarin et le Framework MvvmCross, nous allons donc architecturer notre solution autour du pattern MVVM sur les différentes plateformes !
MvvmCross est un petit Framework qui permet d'architecturer une solution Xamarin autour du pattern MVVM. Il facilite la portabilité et le développement par interfaces grâce aux classes portables et à l'inversion de controle (IoC). Cela va également faciliter nos interactions avec les différents éléments des téléphones (géolocalisation, caméra, ...) - en passant par des plugins.

Préparer le terrain

Au niveau des logiciels et autres SDK, vous aurez besoin des éléments suivants, en fonction de l'installation matérielle dont vous disposez :

	PC	Mac	Mac + PC
SDK Android	Oui	Oui	Sur l'un ou l'autre
JDK	Oui	Oui	Sur l'un ou l'autre
SDK iOS	-	Oui	Sur le Mac
SDK Mac OS X	-	Oui	Sur le Mac
SDK Windows Phone	Oui	-	Sur le PC
SDK Windows 8	Oui	-	Sur le PC

Pour tester vos applications, je vous recommanderais de vous procurer des appareils relatifs à chacune des plateformes sur lesquelles vous souhaitez développer.

Note : *Un outil d'archivage tel que Git (inclus dans TFS 2013 et Visual Studio 2013) va vous permettre de synchroniser votre travail entre vos différents espaces de travail sur PC et sur Mac.*

PREMIERS PAS

Toutes les installations terminées, on peut commencer par :

- Créer une nouvelle solution que nous allons appeler "Happly" (à tout hasard :p)
- Créer un premier projet : une librairie de classes portables que nous allons nommer «Happly.Core» (*).

  Ce projet va accueillir la totalité de notre code métier : référence vers les services web, nos services, nos convertisseurs, nos ressources texte, etc.
- Pour englober un grand nombre de fonctions et faciliter les échanges avec les services Web ainsi que la lecture et l'écriture de données en base et/ou dans des fichiers XML, il nous sera utile d'installer les packages NuGet suivants :
  - Microsoft BCL Portability Pack,
  - Microsoft BCL Build Components,
  - Microsoft Async,
  - Microsoft Http Client Libraries
- Créer les différents projets pour nos différentes plateformes :
  - Happly.Phone* pour Windows Phone,
  - Happly.Store* pour Windows RT,
  - Happly.Touch* pour iOS,
  - Happly.Droid* pour Android.
- Installer le package NuGet "MvvmCross" dans l'ensemble de nos projets (.Core, .Phone, .Store, .Touch, .Droid). Ce package va créer des classes de démarrage, des ViewModel et leurs vues prêts à être utilisés.
- Référencer le projet PCL .Core dans chaque projet (.Phone, .Store, .Touch, .Droid)

(*) Il est recommandé de garder la nomination des projets telle quelle car certains composants de MvvmCross fonctionnent avec de la réflexion et les classes par défaut (App, Setup, FirstViewModel, ...) ont comme namespace .Phone, .Store, .Touch et .Droid.

LES FONDAMENTAUX

Tout en place, il faut demander aux différents systèmes de remplacer leur système de navigation initial par celui surchargé par MvvmCross.
Pour le projet Windows Phone, il faut par exemple, initialiser le système MvvmCross en ajoutant le code suivant pendant la construction de la classe App.xaml.cs :

```
var setup = new Setup(RootFrame);
setup.Initialize();
```

Puis, modifier la première navigation :

```
private void Application_Launching(object sender, Launching
EventArgs e)
{
  RootFrame.Navigating += RootFrameOnNavigating;
}
private void RootFrameOnNavigating(object sender, Navigating
CancelEventArgs args)
{
  args.Cancel = true;
  RootFrame.Navigating -= RootFrameOnNavigating;
  RootFrame.Dispatcher.BeginInvoke(() => Mvx.Resolve<IMvxApp
Start>().Start());
}
```

Pour le projet Windows Store, il faudra modifier la méthode OnLaunched de manière à instancier la classe Setup au niveau où on vérifie que notre pile n'est pas vide :

```
if (rootFrame.Content == null)
{
  // When the navigation stack isn't restored navigate to the
first page,
  // configuring the new page by passing required information
as a navigation
  // parameter
  var setup = new Setup(rootFrame);
  setup.Initialize();
  var start = Mvx.Resolve<IMvxAppStart>();
  start.Start();
}
```

En ce qui concerne iOS, il faut remplacer le contenu de la méthode Finihed-Launching par le code suivant :

```
public override bool FinishedLaunching (UIApplication app,
NSDictionary options)
{
  _window = new UIWindow (UIScreen.MainScreen.Bounds);
  var setup = new Setup(this, _window);
  setup.Initialize();
  var startup = Mvx.Resolve<IMvxAppStart>();
  startup.Start();
  _window.MakeKeyAndVisible ();
  return true;
}
```

Enfin, pour Android, pas besoin de modifier le système d'amorçage. Vous devrez simplement ajouter un SplashScreen, soit la classe suivante à la racine :

```
[Activity(Label = «Hello World», MainLauncher = true, Icon =
 «@drawable/icon»)]
public class SplashScreen : MvxSplashScreenActivity
{
  // instancier un splashscreen ayant comme vue «Resource\
Layout\SplashScreen.axml»
  public SplashScreen() : base(Resource.Layout.SplashScreen)
  {
  }
}
```

...Et une vue AXML dans le dossier "Resources/Layout" () :

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

Puis, ajouter le fichier MvxBindingAttributes dans le dossier "Resources/Values":

```
<?xml version=»1.0» encoding=»utf-8»?>
<resources>
  <declare-styleable name=»MvxBinding»>
    <attr name=»MvxBind» format=»string»/>
    <attr name=»MvxLang» format=»string»/>
  </declare-styleable>
  <declare-styleable name=»MvxControl»>
    <attr name=»MvxTemplate» format=»string»/>
  </declare-styleable>
  <declare-styleable name=»MvxListView»>
    <attr name=»MvxItemTemplate» format=»string»/>
    <attr name=»MvxDropDownItemTemplate» format=»string»/>
  </declare-styleable>
  <item type=»id» name=»MvxBindingTagUnique»/>
  <declare-styleable name=»MvxImageView»>
    <attr name=»MvxSource» format=»string»/>
  </declare-styleable>
</resources>
```

A partir de ce moment, on peut commencer à coder ! Presque... Je vais m'étaler sur quelques explications quant aux classes de démarrage et aux différents héritages obligatoires.

Les classes App et Setup

Dans le projet cœur, il sera nécessaire d'instancier les plugins éventuels et d'informer le ViewModel de démarrage de notre application dans une classe «App» (héritant de MvxApplication).

```
namespace Happly.Core
{
  public class App : MvxApplication
  {
    public override void Initialize()
    {
      RegisterAppStart<FirstViewModel>();
    }
  }
}
```

Dans chaque projet, il est impératif de rajouter une classe Setup à la racine de chacun des projets et d'appeler la classe Happly.Core.App.

```
// La classe Setup doit être à la racine de chaque projet
namespace Happly.Phone|Happly.Store|Happly.Touch|Happly.Droid
{
  public class Setup : MvxPhoneSetup|MvxStoreSetup|MvxTouchSetup|Mvx
AndroidSetup
  {
    public Setup (Context context) : base (context) { … }

    public override IMvxApplication CreateApp()
    {
      return new Core.App();
    }

    public override IMvxTrace CreateDebugTrace()

    {

      return new DebugTrace();
    }
  }
}
```

Héritages obligatoires

Les différents éléments de l'application devront hériter de certaines classes issues de MvvmCross. Par exemple, un ViewModel devra toujours hériter soit d'un enfant de MvxViewModel soit de cette dernière. Pour des raisons de confort (internationalisation par exemple), nous avons fait le choix de créer une classe intermédiaire, BaseView :

```
namespace Happly.Phone|Happly.Store|Happly.Touch|Happly.Droid
{
  public class BaseView : MvxPhonePage|MvxStorePage|MvxView
Controller|MvxActivity
  {
    public BaseView () { … }
    public void DoAnything () { … }
  }
}

namespace Happly.Phone|Happly.Store|Happly.Touch|Happly.Droid
{
  public class FirstView : BaseView
  {
    public FirstView () : base () { … }

    public override void DoAnything ()
    {
      base.DoAnything();
      …
    }
  }
}
```

Spécificités iOS et Android

Dans le système d'exploitation mobile d'Apple, les onglets, les "points", les tableaux de données ou les collections de données nécessitent qu'on hérite de controleurs dédiés à ce type de pages... Vous pourrez ainsi créer un `ViewController` dans lequel vous dessinerez vos onglets où vous pourrez hériter de `UITabBarController`. Ainsi, en fonction de vos besoins, vous pourrez être amené à faire hériter vos pages de MvxTabBarViewController

Spécificités Android

Hello World !

Projets créés, les classes de démarrage App et Setup présentes et présentées, héritage évoqués, on peut commencer à coder un premier Hello World !

Core

Dans le dossier ViewModels du projet Happly.Core, on va créer une classe `FirstViewModel` qu'on va faire hériter de `MvxViewModel` ; y créer une propriété `Hello` et une commande `SayHello`.

```
public class FirstViewModel : MvxViewModel
{
  // propriété Hello en guise d'exemple
  private string _hello;
  public string Hello
  {
    get { return _hello; }
    set
    {
      _hello = value;
      // notifie le système MvvmCross que la propriété a été
modifiée
      RaisePropertyChanged(() => Hello);
    }
  }

  // Une commande
  public IMvxCommand SayHello
  {
    get
    {
      return new MvxCommand(() =>
      {
        Hello = «Bonjour, on est le « + DateTime.Now;
      });
    }
  }
}
```

Windows Phone et Windows 8

Sous Windows Phone / Windows 8, la page qui doit hériter respectivement de `MvxPhonePage` ou de `MvxStorePage` donne le ViewModel en tant que contexte. Il suffit de binder les propriétés qui nous intéressent : Les commandes et le contenu.

```
<!– Windows Store : –>
<views:MvxStorePage x:Class=»Happly.Store.Views.FirstView» xmlns:
views=»clr-namespace:Cirrious.MvvmCross.WindowsStore.Views;
assembly=Cirrious.MvvmCross.WindowsStore» ...>
  <!– Le contenu ici –>
</views:MvxStorePage>

<!– Windows Phone : –>
<views:MvxPhonePage x:Class=»Happly.Phone.Views.FirstView» xmlns:
views=»clr-namespace:Cirrious.MvvmCross.WindowsPhone.Views;
assembly=Cirrious.MvvmCross.WindowsPhone» ...>
  <!– Le contenu ici –>
</views:MvxPhonePage>

<!– Le contenu pour Windows Store et Windows Phone : –>
<Grid x:Name=»ContentPanel» Grid.Row=»1» Margin=»12,0,12,0»>
  <StackPanel>
    <TextBox Text=»{Binding Hello, Mode=TwoWay}» />
    <TextBlock Text=»{Binding Hello}» />
    <Button Command=»{Binding SayHello}»/>
  </StackPanel>
</Grid>
```

iOS et Android

Sous iOS, pas une ligne de code pour designer notre application, il faudra utiliser XCode (Xamarin.iOS dispose également d'un designer bêta), pour glisser-déposer vos éléments à partir de la bibliothèque d'objets.

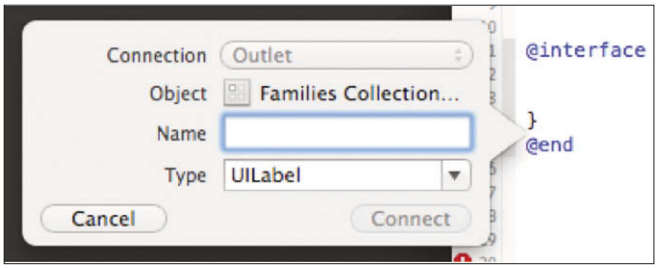
Pour nommer vos objets, il suffit :

1 - D'afficher "Assistant Editor" ,

2 - Afficher le "Connections inspector" ,

3 - Faire un glisser-déposer à partir de la commande "New Referencing Outlet" jusqu'à l'assistant, en dehors du constructeur de l'interface de votre ViewController

4 - Taper le nom que vous souhaitez dans la bulle qui vient de faire son apparition :



5 - Une propriété a été créée : `@property (nonatomic, retain) IBOutlet TYPE *VotreObjet;` et le constructeur de votre interface contient maintenant : `TYPE *_VotreObjet`

6 - Lorsque vous retournerez sur Xamarin, ce dernier va synchroniser l'interface en référençant vos modifications dans le fichier .designer.cs des Vues.

```
public class FirstView : MvxViewController
{
  public FirstView()
    : base (UserInterfaceIdiomIsPhone ? «FirstView_iPhone» : «First
View_iPad», null) { … }

  public override void ViewDidLoad()
  {
    base.ViewDidLoad();
    // méthode 1 pour appliquer les bindings sur iOS :
    this.CreateBinding(SayHelloUIButton).To((FirstViewModel
vm) => vm.SayHello).Apply();
    this.CreateBinding(SayHelloUILabel).To((FirstViewModel
vm) => vm.Hello).Apply();
    // méthode 2 pour appliquer les bindings sur iOS :
    var bindings = this.CreateBindingSet<FirstView, FirstView
Model>();
    bindings.Bind(SayHelloUIButton).To(vm => vm.SayHello);
    bindings.Bind(SayHelloUILabel).To(vm => vm.Hello);
    bindings.Apply();
  }
}
```

Sous Android :

```
[Activity(Label = «Accueil», MainLauncher = true)]
public class FirstView : MvxActivity
{
  protected override void OnCreate(Bundle bundle)
  {
    base.OnCreate(bundle);
    // afficher FirstView.axml présent dans le dossier Resources\Layout
    SetContentView(Resource.Layout.FirstView);
  }
}
```

```
<?xml version=»1.0» encoding=»utf-8»?>
<LinearLayout  xmlns:android=»http://schemas.android.com/apk
/res/android»
    xmlns:local=»http://schemas.android.com/apk/res-auto»
    android:orientation=»vertical»
    android:layout_width=»fill_parent»
    android:layout_height=»wrap_content»
    android:id=»@+id/FirstView»>
  <Button android:layout_width=»fill_parent»
    android:layout_height=»wrap_content»
    android:text=»Dire bonjour !»
    local:MvxBind=»Click SayHello»/>
  <TextView local:MvxBind=»Text Hello»
    android:layout_width=»fill_parent»
    android:layout_height=»wrap_content»/>
</LinearLayout>
```

Navigation

La navigation dans chaque plateforme est différente, mais pour chacune d'entre elles, la navigation est définie comme la capacité qu'a l'utilisateur d'avancer ou de revenir en arrière dans différentes pages de contenu.
Pour ce faire, vous devrez faire en sorte que chaque page aie un ViewModel qui lui soit propre ; et appeler une méthode qui appelle le nouveau ViewModel : `ShowViewModel<TViewModel>()`.
Lorsque vous faites appel à cette méthode, le framework MvvmCross va :
1 - Trouver la vue à utiliser pour le ViewModel appelé (par réflexion),
2 - Créer une instance de la vue trouvée,
3 - Créer une instance de votre ViewModel,
4 - Fournir le ViewModel en tant que DataContext à la vue,
5 - Demander au système d'afficher la nouvelle page (grâce au pattern IoC),
6 - Chercher à exécuter respectivement les méthodes Init() et Start().

Les méthodes Init et Start

Dans les ViewModels (héritant soit d'un enfant, soit directement de IMvxViewModel), deux méthodes nous permettent d'initialiser le contenu de notre page, hors du constructeur :
La méthode virtuelle `Start()`, à overrider si nécessaire, est appelée lors du chargement de la page après l'avoir construite.

```
// Command in FirstViewModel for navigation
public IMvxCommand SimpleNavigation
{
  get
  {
    return new MvxCommand(() => ShowViewModel<SecondViewModel>());
  }
}

public SecondViewModel() { /* do anything */ }
...
public void Start()
{
  if (_loginService.IsLoggedIn()) { /* show content for this user */ }
  else { /* show content for all users */ }
}
```

La méthode `Init(object obj1, object obj2, …)`, à créer est l'équivalent de la méthode virtuelle `Start()`, mais permet de rajouter un nombre (infini?) de paramètres, ce qui peut s'avérer utile pour passer des paramètres lors de la navigation entre différentes pages :

```
public IMvxCommand ParamNavigation
{
get
  {
    return new MvxCommand(() =>
      ShowViewModel<SecondViewModel>(new
        {
          parameter1 = value1,
          parameter2 = value2
        });
```

```
    );
  }
}

public void Init(string parameter1, string parameter2)
{
  _parameter1 = parameter1;
  _parameter2 = parameter2;
  /* do anything with parameters */
}
```

Internationalisation

Il existe plus d'une solution pour internationaliser son application. L'une des solutions est d'utiliser les fichiers de ressources dans le projet PCL et de binder les textes dans les vues. Pour cela, on base chacun de nos `ViewModel` sur une classe faite-maison qui hérite de `MvxViewModel` et qui nous donne un accès statique aux ressources.

Plugins MvvmCross

Certaines fonctionnalités nécessitent des développements différents et particuliers (envoi de messages, appels, ...).
MvvmCross n'inclut pas tout mais grâce à l'architecture basée sur le pattern IoC de MvvmCross, on peut notamment :

- Greffer différents plugins déjà existants et téléchargeables via NuGet par exemple,
- Créer quelques-uns très facilement (sous formes de librairies réutilisables dans n'importe quelle autre solution),
- Ecrire des fonctionnalités spécifiques à chaque plateforme (non réutilisables).

Ainsi, on peut avec le simple appel d'un plugin, passer un coup de fil, interroger le téléphone sur sa position ou encore savoir si l'utilisateur a un accès à Internet ou pas.

Note : *Les plugins MvvmCross doivent être instanciés lors du démarrage de l'application.*

Un plugin ?

Les plugins MvvmCross sont une couche supplémentaire au dessus de l'IoC. Chacun d'entre eux contient :

- Un projet "librairies de classes portable"
  - Une interface qui définit les propriétés, les méthodes, ...
  - Un point d'entrée à PluginLoader.Instance
- Accessoirement, un projet par plateforme
  - Les méthodes codées de l'interface définies dans la PCL pour la plateforme spécifiée.
  - Une entrée à "Plugin"

Utiliser les plugins

Dans un premier temps, à partir de notre "Hello World", on va appeler le standard du pôle formations de chez Happly-Sfeir. Pour commencer, on va ajouter le package NuGet "MvvmCross PhoneCall Plugin".
Dans la classe de démarrage Happly.Core.App, on va ajouter la ligne suivante pour instancier le plugin, juste avant de spécifier avec quel ViewModel on va démarrer notre application :

```
Cirrious.MvvmCross.Plugins.PhoneCall.Instance.EnsureLoaded();
```

Pour passer nos coups de fil, on va dessiner un bouton avec nos différents designers et binder une commande dans notre ViewModel :

```
public IMvxCommand CallCommand
{
  get
  {
    return new MvxCommand(() =>
    {
      Mvx.Resolve<IMvxPhoneCallTask>().MakePhoneCall(«Happly Formations», «+33 3 89 35 19 19»);
    });
  }
}
```

XAML Windows Phone / Windows Store :

```
<Button … Command=»{Binding CallCommand}» />
```

AXML Android :

```
<Button … local:MvxBind=»Click CallCommand» />
```

iOS :

```
this.CreateBinding(CallUIButton).To((FirstViewModel vm) => vm.CallCommand).Apply();
```

Écrire son plugin MvvmCross

Admettons maintenant que nous cherchions à afficher une barre de progression sur les trois plateformes... On pourrait glisser-déposer les éléments graphiques via Visual Studio et Xcode, puis binder sur une propriété sur chaque projet (pour l'afficher/masquer par exemple). Ou, on pourrait envisager de développer un plugin qui, lorsqu'il sera appelé affichera une barre de progression (indéterminée pour simplifier le code).
On va donc créer une nouvelle solution (Happly.Plugin.ShowProgressBar) dans laquelle on va créer les projets suivants :

Plateforme	Type de projet	Nom du projet
Code partagé	Bibliothèque de classes portable	Happly.Plugin.Calendar.Core
Windows Phone	Bibliothèque de classes Windows Phone	Happly.Plugin.Calendar.Phone
Windows Store	Bibliothèque de classes (applications Windows Store)	Happly.Plugin.Calendar.Store
iOS	Xamarin.iOS Library Project	Happly.Plugin.Calendar.Touch
Android	Android Class Library	Happly.Plugin.Calendar.Droid

Pour chacun de ces projets, on va Installer le package NuGet "MvvmCross - CrossCore". Dans le projet "Happly.Plugin.Busy.Core", on va ajouter deux nouveaux éléments :

- une interface qui va définir les méthodes dont on aura besoin dans toutes les plateformes

```
public interface ICalendar
{
  void Show(DateTime start, DateTime end, string name);
}
```

- une implémentation de IMvxPluginLoader, qu'on nommera "PluginLoader" :

```
public class PluginLoader : IMvxPluginLoader
{
  public static readonly PluginLoader Instance = new PluginLoader();
  public void EnsureLoaded()
  {
    var manager = Mvx.Resolve<IMvxPluginManager>();
    manager.EnsurePlatformAdaptionLoaded<PluginLoader>();
  }
}
```

Dans les différents projets, on va créer une classe Plugin et ajouter une méthode Load pour :

```
public class Plugin : IMvxPlugin
{
  public void Load()
  {
    Mvx.RegisterType<ICalendar,  MvxPhoneCalendar|MvxStore
Calendar|MvxTouchCalendar|MvxDroidCalendar>();
  }
}
```

...Et implémenter notre Interface dans chaque projet, en fonction de sa plateforme :

```
public class MvxPhoneCalendar : ICalendar
{
  public void Show(DateTime start, DateTime end, string name)
  {
    /* do calendar on Windows Phone - eg. SaveAppointmentTask */
  }
}

public class MvxStoreCalendar : ICalendar
{
  public void Show(DateTime start, DateTime end, string name)
  {
    /* use Windows 8 Calendar API - eg. AppointmentManager */
  }
}

public class MvxTouchCalendar : ICalendar
{
  public void Show(DateTime start, DateTime end, string name)
  {
    /* use iOS Calendar API - eg. EventKit */
  }
}

public class MvxDroidCalendar : ICalendar
{
  public void Show(DateTime start, DateTime end, string name)
  {
    /* use Android Calendar API eg. intents, CalendarContract, … */
  }
}
```

Pour l'utiliser, il suffira de compiler, de récupérer les librairies, et de référencer le coeur, ainsi que les différentes librairies dans les projets adéquats. Dans notre petit exemple :

Plateforme	Références à ajouter dans notre Solution Happly
Core (PCL)	Happly.MvvmCross.Plugins.Calendar.Core
Windows Phone	Happly.MvvmCross.Plugins.Calendar.Core Happly.MvvmCross.Plugins.Calendar.WindowsPhone
Windows Store	Happly.MvvmCross.Plugins.Calendar.Core Happly.MvvmCross.Plugins.Calendar.WindowsStore
iOS	Happly.MvvmCross.Plugins.Calendar.Core Happly.MvvmCross.Plugins.Calendar.Touch
Android	Happly.MvvmCross.Plugins.Calendar.Core Happly.MvvmCross.Plugins.Calendar.Droid

Avant d'appeler une méthode du plugin, il vaudrait mieux s'assurer que le mécanisme des plugins est instancié et que le plugin est bien chargé par le système d'exploitation hôte. Ensuite, on peut par exemple dessiner un bouton et le binder à une Commande :

```
public IMvxCommand SaveOnCalendarCommand
{
  get
  {
    return new MvxCommand<FirstViewModel> (save =>
      {
        Happly.MvvmCross.Plugins.Calendar. PluginLoader.Instance.
EnsureLoaded();
        Mvx.Resolve<ICalendar>().Show(DateTime.Now, DateTime.
Now.AddHours(2), string.empty);
      }
  }
}
```

Vous aurez sans doute deviné qu'il s'agit du pattern IoC ! Si vous ne souhaitez pas passer par un plugin pour n'importe quelle raison, il est possible d'utiliser le pattern au sein même de la solution principale. On aurait donc une interface dans notre projet PCL et des classes héritant de cette interface dans nos différents projets. Pour l'utiliser, il suffira de "résoudre" l'interface dans une variable et d'utiliser cette dernière :

```
var calendar = Mvx.Resolve<ICalendar>().Show(DateTime.Now,
DateTime.Now.AddHours(2), string.empty);
```

Design

Cela ne vous a sans doute pas échappé : l'Interface Utilisateur de chaque plateforme doit être faite séparément avec les designers et/ou les langages spécifiques aux différentes plateformes, soit les couples suivants :

Platefome	Type de fichier / Langage	Designer
Windows Phone / Windows Store	XAML	Visual Studio / Blend
iOS	XIB / Storyboard	XCode / Xamarin on Mac
Android	AXML	Xamarin, Eclipse ou autre

En dehors de l'aspect réalisation, n'oublions pas que chacun des éditeurs que sont Microsoft, Apple et Google ont des préconisations quant au design de vos applications sur leurs plateformes. Vous pouvez les retrouver ici :
Windows Phone : http://dev.windowsphone.com/en-us/design
Windows Store : http://msdn.microsoft.com/en-US/windows/apps/hh779072
iOS : https://developer.apple.com/design
Android : http://developer.android.com/design

CONCLUSION

Le regard qu'apporte Xamarin est prometteur. Couplé à MvvmCross, c'est une merveille : gain de temps (3x pour 3 plateformes), code partagé dans les 80%, presque aucune limite technique et j'en passe ! Les seules limites qu'on a pu trouver jusqu'ici s'arrêtent au décalage entre la sortie des dernières versions des SDK et la mise à jour sur Xamarin.

Remerciements

MVVM (livre) : Jonhathan Antoine, Infinite Square
Conseils. MvvmCross : Arnaud Maichac, Happly-Sfeir
Vérification : Arnaud Maichac (Consultant .NET Happly-Sfeir)

Andrés **Talavera**
Développeur .NET Happly-Sfeir
Microsoft Student Partner France

Firefox OS : découverte et développement

Nous vous en parlions dès le numéro 161 : l'arrivée du nouveau système mobile Firefox OS, développé, soutenu et maintenu par la fondation Mozilla. C'est pourquoi il est important de connaître aujourd'hui la situation de ce système, de son avancée et de sa position mondiale sur le marché de la téléphonie.

Le projet de Firefox OS a été lancé en 2011 et 2 ans après, il était déjà disponible et utilisable. Ainsi, depuis son lancement les versions disponibles s'appuient sur le cycle de réalisations du navigateur Firefox :

- 1.0 Février 2013
- 1.2 Septembre 2013 basé sur Firefox 24
- 1.3 Janvier 2014 basé sur Firefox 28
- 1.4 Version en cours

En février 2014, s'est déroulé le Mobile World Congress en Espagne, et Mozilla y a dévoilé et montré les nombreux avancements de l'existant ainsi que des nouveautés intéressantes. Par exemple les nouveaux téléphones et l'orientation de la nouvelle version majeure 2.0.

ARCHITECTURE

Firefox OS (nom de code "Boot to Gecko" ou "B2G") est un système d'exploitation Open Source pour les téléphones mobiles. Il possède un cœur Linux, associé à un moteur de rendu Gecko. La différence par rapport aux autres fournisseurs touche les logiciels qui s'exécutent aussi bien sur le téléphone, que sur les ordinateurs, grâce à la technologie (web) utilisée. Elle doit permettre aux développeurs Web de programmer en HTML5 et en JavaScript, et en même temps de bénéficier de la puissance technique qu'offrent ces deux langages. La partie la plus complexe pour eux est de comprendre l'intégrité, car l'interface affichée pour les utilisateurs est une application Web.

Les couches

Un téléphone fonctionne comme un ordinateur de bureau, c'est-à-dire sur 3 niveaux : tout d'abord un système d'amorçage, ensuite un système exploitation (OS), et pour terminer le niveau des applications. Ces 3 niveaux indépendants, tous aussi importants les uns par rapport aux autres, sont reliés entre eux pour pouvoir communiquer. Ainsi, vous trouverez :

GONK Fig.1

Il s'agit d'un noyau Linux, basé sur Android Open Source Project (AOSP). Il fonctionne comme un système d'exploitation. Il contrôle l'ensemble des composants du téléphone de la plateforme de FirefoxOS. Ces composants sont GPS, Bluetooth, USB, appareil photo, caméra, périphériques, etc. De plus, il a la charge de démarrer le téléphone, mais aussi de permettre d'appeler une personne ou encore de recevoir la radio.

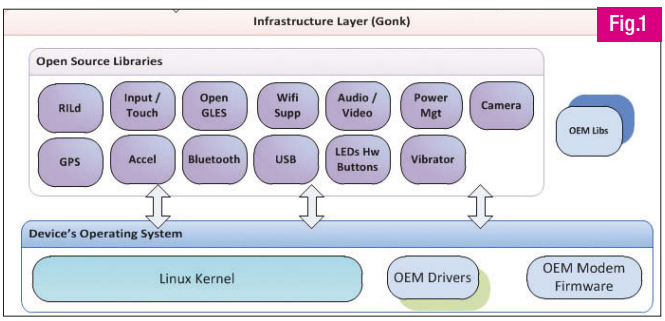


Fig.1

GECKO Fig.2

Gecko est le 'runtime', c'est-à-dire le moteur de rendu, ce qui permet d'exécuter des applications pour Firefox OS. Le but de Gecko est de lire le contenu Web, et par conséquent les langages ouverts tels que HTML, CSS, JavaScript. Il garantit le bon fonctionnement des API avec par exemple la prise en charge de la partie graphique, la gestion des contacts, les alarmes, les droits d'accès, les SMS, etc.

GAIA Fig.3

Il s'agit de l'interface utilisateur de la plate-forme utilisateur de Firefox OS. Son rôle permet de gérer toutes les applications standard dont vous avez besoin lors de l'utilisation d'un smartphone : l'accès aux réseaux sociaux, la messagerie... C'est-à-dire qu'il s'occupe de la partie que vous visualisez et par conséquent, vous contrôler le verrouillage de l'écran, l'écran d'accueil, etc. Ce niveau va permettre d'utiliser les différents périphériques gérés par la couche Gecko à travers les API Web ouvertes.

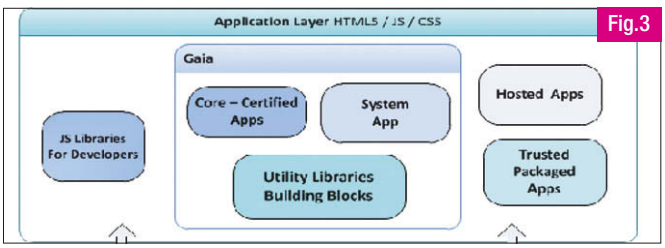


Fig.3

La sécurité

La sécurité est très importante, car chaque niveau d'architecture peut subir une attaque à cause d'une faille de sécurité. De nombreux documents liés à la sécurité touchent différents niveaux :

- Le modèle,
- Le système,
- Des applications,
- À l'installation et la mise à jour des applications,
- Au niveau des tests et du débogage.

Les Fonctions

Comme vous avez pu le voir un peu plus haut, la Roadmap des versions est importante. Cependant les fonctionnalités nécessaires qu'attendent les utilisateurs au niveau d'un smartphone, sont disponibles et peuvent être contrôlées à travers des WebApps.

Les fonctionnalités indispensables concernent :

- La téléphonie, c'est-à-dire téléphoner, gérer ses contacts, envoyer et recevoir des SMS, des emails,
- Le multimédia qui se traduit par la possibilité d'utiliser la caméra, d'écouter de la radio ou de la musique,
- La configuration de base qui se traduit par gérer un calendrier, l'heure,

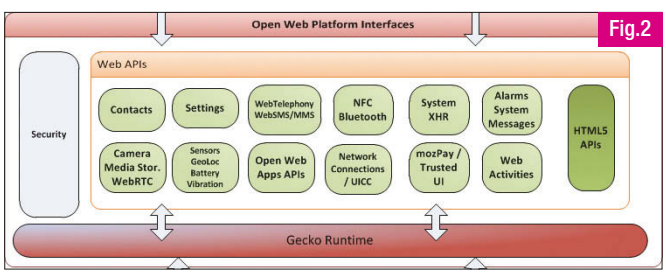


Fig.2

l'écran d'accueil, un clavier, des périphériques (USB, Bluetooth...), l'autonomie (batterie). Mais aussi un Marketplace pour obtenir de nouvelles applications.

Les API Web

Les API Web sont des logiciels indispensables pour un bon fonctionnement du téléphone. À partir de la version 1.2, Firefox OS compte plus de 30 API Web natives pour améliorer les fonctionnalités au niveau du développement des applications Web. Certaines API étaient déjà présentes dès le lancement de l'OS comme WebNFC et Data Store API, mais aussi Webkit, les API de vibration et de l'état de la batterie, Apache Cordova et PhoneGap.

LES TERMINAUX

En 2014, une nouvelle vague de smartphones et de tablettes (7 modèles en tout) a été dévoilée bénéficiant de la dernière version de l'OS. L'ensemble de ces modèles sera disponible dans l'année.

L'entrée de gamme

Il s'agit d'un téléphone à destination des pays émergeants, car la particularité concerne le prix de 25 dollars.
Au niveau des caractéristiques, le modèle présenté est un Spredtrum avec un processeur SC8621. Il se présente avec 1 processeur Cortex A5 @ 1 GHz, 2 Gb de mémoire extensible et un petit écran 3.5-inch HVGA, ce qui correspond à une résolution de 320x240 pixels. Pour les composants externes, l'appareil bénéficie de WiFi, Bluetooth, FM, Camera... et de nombreuses fonctionnalités basiques : SMS, Video Player , navigateur, email, etc.

Au-dessus

ZTE : le fabricant propose différents modèles ZTE Open avec Firefox OS. Le modèle ZTE OpenII disponible en février 2014, se compose d'un CPU Qualcomm MSM7225A 1.0Ghz et GSM (850/900/1800/1900) en dual core 1,2 GHz, d'un écran 3.5" HVGA (320x480 pixels), 256 de mémoire, un disque SSD de 32 Gb. Au niveau des composants externes : une batterie 1200 mAh, WiFi, Radio, microUSB, WiFi. Alcatel One Touch Fire est un autre téléphone. Il se compose d'un écran 3.5" HVGA (320x480 pixels) , d'un Qualcomm MSM7227A et GSM (850/900/1800/1900), de 160 à 256 Mo de mémoire suivant le modèle, un disque SSD de 32 Gb.

Les tablettes

La tablette Foxconn InFocus est équipée d'un écran 10 pouces (1280x800 pixels) avec un processeur quadruple cœur de la gamme ARM Cortex A7 (A31) de fréquence 1,0 GHz, d'une mémoire de stockage 16 Go, de 2 Go de RAM, d'un appareil photo arrière de 5 Mégapixels et d'une caméra frontale de 2 Mégapixels. Au niveau des composants externes : A-CPG, une batterie 7000 mAh, Wifi.
La tablette Vixen propose une tablette différente, comprenant un biprocesseur Cortex-A9 de fréquence 1,2 GHz, dotée d'un écran LCD HD 7 pouces, de 8 Go d'espace de stockage, d'une RAM de 1 Go, d'une caméra arrière de 2,0 Mégapixels et avant de 0,3 Mégapixel. Elle dispose de la technologie WiFi.

POUR LES CONTRIBUTEURS

La fondation Mozilla propose aux contributeurs un téléphone appelé 'Firefox OS Flame' pour aider les développeurs dans la réalisation de leurs projets. Il s'agit d'un téléphone modulable à tous les niveaux, mais surtout permettant de tester leurs jeux, applications... Ce téléphone bénéficie des technologies suivantes : un Qualcomm Snapdragon MSM8210 avec une fréquence de 1,2 GHz, équipé d'un écran 4.5 pouces et embarque un processeur double cœur (Dual Core).
Une des particularités de ce smartphone est de permettre aux développeurs de configurer la mémoire RAM en quelques clics pour choisir un téléphone qui pourrait posséder de 256 Mo à 1 Go. Ainsi leur permettre de vérifier si le projet Web fonctionne sur les différents modèles distribués. Il est équipé d'une batterie de 1800 mAh, avec un support Dual SIM. Au niveau des composants externes, l'appareil bénéficie des technologies WiFi, Bluetooth et la 3G/UMTS quadri-bande (850/900/1900/2100).

LES AUTRES MODÈLES

La fondation Mozilla propose une compatibilité de Firefox OS pour des smartphones non partenaires. Grâce à cette ouverture, les appareils équipés de Android 4.0 (minimum) peuvent prétendre supporter Firefox OS à la place du système qu'ils ont. Mais vous pouvez vérifier la configuration exacte de votre appareil avec l'aide en ligne
https://developer.mozilla.org/en-US/Firefox_OS/Firefox_OS_build_prerequisites
Cette page vous aide à savoir si le socle de l'appareil est basé sur un Linux 64 bits, et propose une suite d'étapes afin de valider si votre appareil mobile est compatible.

DÉVELOPPEMENT

Comme Firefox OS est composé d'une architecture à 3 niveaux, le développement classique que vous pouvez utiliser, concerne le niveau GAIA, car les autres niveaux sont gérés par l'OS de Firefox.

Attention il est tout à fait possible de réaliser des développements sur les niveaux inférieurs, cependant vous devez solliciter la fondation Mozilla pour connaître tous les détails.

Pour développer au niveau de la couche GAIA, il faut connaître la technologie HTML 5, et aussi avoir des connaissances dans les modèles MVC pour respecter les bonnes pratiques de développement si vous travaillez à plusieurs.
Tout d'abord, un environnement de travail est nécessaire. Pour cela, vous avez besoin de posséder votre espace virtuel comme Foxbox, VirtualBox... et avoir installé GIT/GitHub.
Ce dernier vous permet de voir les évolutions de votre code, les mises à jour et les modifications de toutes les personnes qui contribuent au projet.
Dans cet environnement, pensez à installer 1 navigateur Firefox, l'émulateur Firefox OS et l'outil APP Manager (détails plus loin dans l'article). De plus, il existe d'autres extensions pour déboguer ce que vous devez installer si votre application a besoin d'utiliser les différents périphériques d'un téléphone.
Cette configuration vous garantit un espace se rapprochant d'un téléphone Firefox.

Les outils de tests

La réalisation des tests s'effectue en semi-automatique. Tout d'abord, vous devez passer votre application Web aux validateurs W3C pour être certains de respecter les balises HTML. Cependant, si vous souhaitez tester les fonctions spécifiques WebAPPs, vous utiliserez les outils proposés par Mozilla, comme le simulateur.

Simulateur

Le simulateur FirefoxOS est un add-on, disponible pour le navigateur Firefox, sous licence GNU et fonctionne sur Linux, Mac et Windows. Grâce à ce simulateur, vous pouvez tester et suivre l'évolution de l'OS à partir de votre ordinateur. Ce simulateur est à destination des développeurs, bricoleurs et à toutes les personnes qui veulent en savoir plus sur cet outil.

Fonctionnalités

Au niveau des fonctionnalités de celui-ci, la version actuelle (4.0) propose de nombreuses optimisations et corrections de bugs avec des évolutions au niveau MarketPlace, qui est utile pour simuler l'achat de nouvelles applications comme Achat validé, non validé et remboursement. Bien entendu, la simulation du touchpad a été amélioré pour rendre votre souris plus proche d'une utilisation tactile.

Utilisation

Pour bénéficier de l'outil dans votre navigateur, vous devez effectuer certaines étapes :

- Tout d'abord, vous téléchargez l'extension comme un add-on classique dans le navigateur Firefox c'est-à-dire à partir de la barre de menu « outils > modules complémentaires ». Vous effectuez une recherche de 'Firefox os' et vous installez l'extension.
- Ensuite, pour exécuter le simulateur, vous vous rendez dans le menu déroulant « outils > développeur Web > Firefox OS simulator »

L'écran de configuration se lance automatiquement. Il se découpe en 2 parties :

Une partie qui est représentée par un bouton, permettant de lancer le simulateur et de l'arrêter. L'autre partie est le journal des applications Web que vous avez installé. Vous trouverez la possibilité de télécharger les vôtres pour vérifier la compatibilité ou à partir d'un lien internet.

Cette partie est importante, car elle est liée au fichier manifest.webapp et si le simulateur détecte une erreur, vous serez alerté Fig.4.

Le message signale la ligne qui pose problème. Ici, il s'agit d'un problème de quote, mal fermé.

Si votre fichier manifest.webapp est correctement configuré, il sera reconnu par le simulateur et peut être exécuté par la suite Fig.5.

Lorsque vous démarrez le simulateur, celui-ci effectue les mêmes étapes, comme si vous possédiez un téléphone Firefox dans les mains. Il amorce les périphériques, charge l'OS et les applications déjà installées et celles de votre journal (Dashboard).

L'interface propose une taille simulée de 320x480 pixels, et pour naviguer entre les écrans, vous cliquez sur le bouton de la souris et faites glisser tout en le maintenant enfoncé. Ainsi, vous verrez les applications intégrées et les applications que vous avez ajoutées.

Les autres outils

Tests unitaires

Comme le langage de développement est HTML5/CSS /JavaScript, vous pouvez utiliser n'importe quels outils génériques disponibles pour effectuer des tests unitaires. Bien entendu, à l'heure d'aujourd'hui Qunit est un des outils les plus avancés au niveau des tests.

APP Manager

Il facilite la programmation d'applications grâce au prototypage et dépannage en temps réel. Il se compose de plusieurs outils pour vous aider dans les développements Web. Grâce à cela, vous pouvez tester, déployer et dépanner les applications Web sur des téléphones Firefox OS directement à partir d'un environnement de travail. Il améliore le processus de développement et le débogage des applications Firefox OS ou lors de l'utilisation du simulateur, ou avec un appareil connecté.

Comme ceci, vous n'avez pas besoin d'utiliser d'outils ou de SDK supplémentaires. En effet, AppManager va aider à déployer directement sur un appareil, à valider les applications et à les déboguer à partir du périphérique Fig.6.

Au niveau de l'utilisation de AppManager, il améliore les outils existants pour fournir un environnement intégré de débogage et de déploiement, et par conséquent remplace l'actuel tableau de bord du simulateur. Il permet d'installer des applications hébergées ou empaquetées dans le simulateur

Ce gestionnaire d'applications affiche des informations supplémentaires, qui sont utiles pour le développeur telles que : des informations sur l'appareil connecté, effectuer des captures d'écrans, visualiser un meilleur rendu du journal des applications déjà installées, etc.

Par ailleurs, l'utilisation de la console est accessible pour consulter les avertissements et les erreurs à l'intérieur de l'application. De plus, la modification du code HTML et CSS est toujours possible.

Les outils sont très nombreux : vous pouvez consulter tous les détails à partir de la page dédiée.

LE PARTAGE SUR LE MARKETPLACE

Le partage des applications passe par un market ouvert, c'est-à-dire une plate-forme de téléchargement, où vous trouverez différentes applications et jeux pour votre téléphone.

La préparation

Une fois que votre application est terminée, vous devez la déployer sur le Marketplace Firefox. Deux options s'offrent à vous pour la déployer.

Applications hébergées

Les applications hébergées permettent de stocker vos applications sur un serveur Web de votre choix, qui est accessible publiquement, comme n'importe quel autre site Web. Vous devez quand même créer le fichier 'manifest.webapp' valide. Si votre application ne comporte que des pages statiques (HTML/CSS /JavaScript) et sans aucun traitement du côté serveur, vous pouvez vous appuyer sur GitHub. Cependant, si les pages sont dynamiques, il est préférable d'utiliser un hébergeur générique qui possède de bonnes capacités, ou un héber-

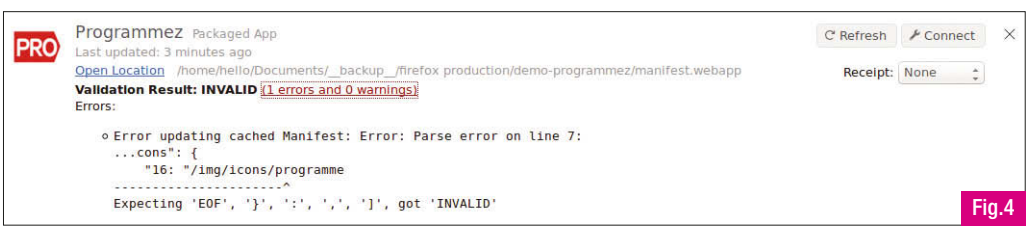


Fig.4

Fig.5

gement qui correspondra parfaitement à votre application, car l'espace est dédié aussi bien pour les internautes que pour les possesseurs de smartphones Firefox OS.
Au niveau du code, vous devez créer un fichier d'installation qui se traduit par un kit d'installation appelé pakage.manifest.

```
{
  "name": "Nom de l'application",
  "package_path" : "http://votreSite.com/api/mon-application.zip",
  "version": "1",
  "developer": {
    "name": "Auteur",
    "url": "http://votreSite.com"
  }
}
```

Ce fichier est composé de différentes informations au format JSON que vous devez compléter.
Ensuite, vous devez prévoir une page d'installation HTML avec le script suivant :

```
<html>
  <body>
    <p>Page installation</p>
    <script>
    var manifestUrl = 'http://VotreSite.com/api/package.manifest';
    var req = navigator.mozApps.installPackage(manifestUrl);
    req.onsuccess = function() {
      alert(this.result.origin);
    };
    req.onerror = function() {
      alert(this.error.name);
    };
    </script>
  </body>
</html>
```

Il s'agit d'un script javascript, dont vous devez spécifier le chemin complet du 'package.manifest'.
L'installation se fait automatiquement grâce à la fonction InstallPackage.
Bien entendu, il faut prévoir une réponse pour signaler un succès ou un problème.
Ces 2 fichiers doivent se placer dans le même dossier pour permettre l'installation de l'application.

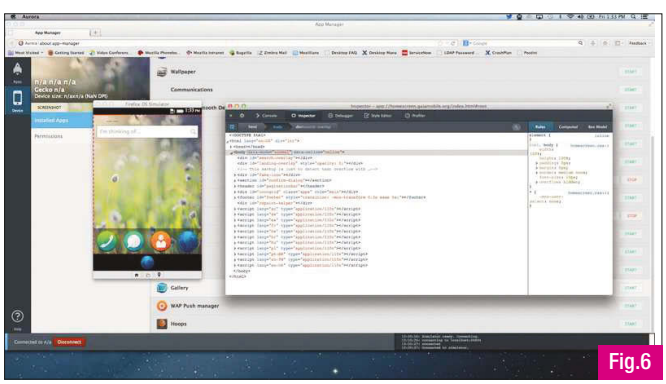


Fig.6

Applications empaquetées

Les applications empaquetées seront hébergées par le marketplace de Firefox, si vous ne souhaitez pas héberger chez vous. Pour cela, vous devez créer le fichier 'manifest.webapp' pour rendre votre application Web compatible et ensuite le transformer en fichier ZIP pour qu'il soit reconnu dans le marketplace.
Le but d'une application empaquetée est d'avoir un moyen pratique de fournir des applications sur le marketPlace. Cela donne aux utilisateurs de l'application plus d'assurance que l'application a été soigneusement étudiée pour la sécurité potentielle , la vie privée et les questions de capacité.

La publication

La dernière étape concerne l'envoi de votre application dans le marketplace. Pour cela, vous devez vous rendre à l'adresse de la marketplace pour vous faire connaître. https://marketplace.firefox.com/developers/.
Vous arrivez sur l'écran de configuration qui se découpe en plusieurs parties. La partie supérieure propose de sélectionner le système d'exploitation App Firefox. Vous pouvez aussi sélectionner de proposer une application payante.
La partie inférieure affiche deux onglets. Un pour hébergé et l'autre pour emballé.

- Si vous avez choisi d'héberger votre application, vous devez vous reporter au paragraphe au-dessus et suivre la procédure.
- Si vous choisissez le mode emballé, c'est à dire que vous voulez déposer votre application sur le MarketPlace de Firefox, vous devez alors construire votre fichier ZIP

Bien entendu, si une erreur est repérée, une ou plusieurs informations seront affichées.
L'étape suivante concerne la configuration de l'écran de présentation, c'est-à-dire que vous renseignez le nom, la description, une capture-écran, la catégorie où elle sera rangée, etc.
Enfin, pour apparaître dans le marketplace, un petit peu de patience est nécessaire, car quelques vérifications et contrôles sont effectués pour certifier votre application comme conforme.

Où l'acheter

Actuellement les points de vente en France sont limités, voire inexistants, car même si les pays émergeants et de nombreux pays européens le commercialisent, les Français souhaitant en obtenir un, doivent l'acheter sur internet comme sur Ebay ou Amazon. On pourrait imaginer qu'un opérateur français les proposent dans son catalogue hors forfait puisqu'ils sont compatibles avec l'ensemble des opérateurs que nous connaissons. Cependant, différents communiqués ont dévoilé qu'à partir du premier semestre (été) 2014, certains modèles pourraient être disponibles en France équipés de Firefox OS.

L'avenir

L'avenir de Firefox OS passe par une nouvelle version majeure, avec une refonte du moteur principal. Cette version va répondre aux différentes attentes et absences. L'avantage de ce changement permettra de réaliser de nouvelles applications qui manquaient et de suivre l'évolution du Web. Une des technologies prometteuses concerne la Technologie WebRTC, qui est une technologie phare. Elle apporte des fonctionnalités majeures au niveau de la voix, de l'audio et des échanges de types peer-to-peer.

Christophe **Villeneuve**
consultant IT pour Neuros, auteur du livre « PHP & MySQL-MySQLi-PDO, construisez votre application », aux Éditions ENI. Rédacteur pour WebRIVER, membre des Teams DrupalFR, AFUP,

Votre application WEB sur Firefox OS

Firefox OS vous offre la possibilité de réaliser des applications mobiles HTML 5 sans connaissances supplémentaires. Cette compatibilité vous permet de développer vos projets en une seule fois, mais utilisables sur l'ensemble des systèmes du marché : ordinateurs de bureau, portables, tablettes. Pour rendre votre application web portable, vous devez respecter les standards W3C. Ainsi elle peut être transformée en une application mobile pour Firefox OS en un minimum de temps fonctionnel sur Firefox OS.

La migration d'un projet web ou d'une application (API) s'effectue en plusieurs étapes. Ces différentes étapes nécessitent très peu d'apprentissage, car elles s'appuient sur les normes HTML5. Une application web doit être autonome et par conséquent pouvoir utiliser le cache de l'appareil pour être toujours disponible même si vous ne possédez pas de connexion internet.

ÉTAPE 1 : LE MANIFEST

Le 'Manifest' se présente sous la forme d'un fichier, appelé 'manifest.webapp'. Il se compose d'informations qui interagissent entre le navigateur (qui est votre OS) et l'application Web Open Web App.
Il s'agit de l'étape la plus importante et la plus compliquée, car ce manifeste est structuré au format JSON. Comme tous fichiers, certains champs sont obligatoires comme le nom, la description, l'affichage des icônes et les auteurs. Mais de nombreuses options sont disponibles pour améliorer l'autonomie et la possibilité d'avoir une meilleure compatibilité comme les permissions, le cache, les langues...

fichier manifest.webapp

Ce fichier 'manifest.webapp' se positionne au niveau de la racine de votre site web et se décompose de la manière suivante :
Les informations indispensables concernent la version, le nom de l'application, la description :

```
«version» : «1.0»,
«name» : «Votre application »,
«description» : «La description de votre application web»,
```

Pour exécuter l'application web, c'est-à-dire l'exécution du fichier index.html, vous spécifiez le chemin complet pour lancer celle-ci. Ici le fichier index.html se trouve à la racine de votre site web.

```
«launch_path» : «/index.html»,
```

Pour afficher votre application dans la galerie des lanceurs de votre téléphone, il faut définir des icônes avec des tailles différentes, car tous les téléphones ne sont pas fabriqués de la même façon, avec des tailles différentes. Ici nous définissons 3 tailles des 6 dimensions possibles (max 256x256), en spécifiant le chemin absolu du fichier icône. Celle-ci est une image PNG, qui peut être réalisée avec n'importe quel logiciel de DAO que vous pouvez utiliser.

```
«icons» : {
  «16» : «/images/icons/appicon-16.png»,
  «48» : «/images/icons/appicon-48.png»,
  «128» : «/images/icons/appicon-128.png»
},
```

Vous pouvez spécifier un nom, qui correspond au nom de la personne ou de la société qui a réalisé l'application. Cette information est utile pour le market, lorsque les utilisateurs voudront connaître les auteurs du projet.

```
«developer»: {
  «name» : «Votre nom»,
  «url» : «http://votreSite.com»
},
```

Lorsque vous créez une application web (web App), il est utile que celle-ci puisse toujours fonctionner même si vous n'êtes plus connecté, c'est à dire, si vous vous retrouvez en mode hors connexion. Pour activer le cache, vous devez spécifier le chemin absolu vers le fichier 'cache.manifest', qui se présente de la manière suivante :

```
«appcache_path» : «/cache.manifest»,
```

Au moment du déploiement, vous devez pouvoir identifier vos partenaires ou les sites qui pourront proposer votre application. Ici, nous allons permettre à n'importe qui d'installer l'application et vous devez ajouter la ligne suivante :

```
«installs_allowed_from» : [«*»],
```

La langue est très importante, surtout si vous désirez être utilisé par le monde entier. Pour cela, la déclaration s'effectue par 'locales'. Il s'agit d'une surcouche 'custom' des informations renseignées un peu plus haut dans le fichier. Ainsi, dans une langue, vous pouvez fournir une description dans une autre langue, spécifier une URL spécifique pour le pays, etc.. Par conséquent, nous ajoutons une nouvelle langue, « l'Anglais »

```
«locales» : {
  «en» : {
    «description» : «Description of your web application in the
  proposed language»,
    «developer» : {
      «url» : «http://votreapplication.com»
    }
  },
},
```

Comme vous vous trouvez avec 2 langues : le français et l'anglais, vous choisissez une des deux qui servira de langue par défaut, par exemple l'anglais :

```
«default_locale» : «en»,
```

Pour faire fonctionner une application, celle-ci doit identifier les API qu'el-

le compte utiliser et qui nécessitent l'autorisation de l'utilisateur. Cette opération se charge de définir les droits d'utilisation. Il existe de nombreuses options pour contrôler les droits de votre application.
Pour notre exemple, il est nécessaire d'activer l'option 'systemXHR' qui permet de créer des requêtes HTTP sans aucune restriction sur l'origine.
La configuration se présente de la manière suivante :

```
«permissions» : {
  «systemXHR» : {}
}
```

Configuration du serveur web

Le fichier 'manifest.webapp' est considéré comme un type (appelé mime-type). Cela permet au navigateur de pouvoir traiter correctement ce fichier en fonction de son contenu. Pour effectuer cela, vous devez modifier la configuration de votre serveur (Apache, Nginx, IIS...) ou dans le fichier .htaccess. Nous privilégions la création d'un fichier .htaccess pour que ce 'mime-type' soit pris en compte. Pour cela, nous ajoutons dans ce fichier la ligne suivante :

```
AddType application/x-web-app-manifest+json .webapp
```

ÉTAPE 2 : ACTIVER LE CACHE

Déclaration

Pour activer le cache et rendre votre application web utilisable hors connexion, vous devez ajouter un tag supplémentaire dans la balise HTML de votre page de la manière suivante :

```
<html manifest=»offline.appcache»>
  …
</html>
```

Il s'agit d'un pointeur qui sera en place lors de l'utilisation de votre application. Ainsi, vous pouvez créer votre propre offline.appcache dans les fichiers à télécharger.

```
AddType text/cache-manifest .manifest
```

Fichier offline.appcache

Ce fichier 'offline.appcache' se positionne à la racine de votre site web. Il s'agit d'un fichier texte contenant, sur une ligne, le nom du fichier de votre application. Le fichier 'offline.appcache estnécessaire pour le bon fonctionnement de l'application car vous y enregistrez votre fichier index.html, les fichiers CSS, images, JavaScript...
Bien entendu avec chaque fichier, il ne faut pas oublier le chemin, et la représentation qui se présente de la manière suivante :

```
CACHE MANIFEST
# version 0.01

/index.html
/css/data.css
/images/logo.png
/images/bandeau.png
/javascript/librairie.js
/javascript/cache.js
/javascript/api.js
```

Le contenu de ce fichier comporte plusieurs particularités, qui sont des standards du format HTML 5.

Tout d'abord, la ligne qui commence par #, correspond à la version des fichiers cachés. C'est à dire que si vous modifiez le contenu d'un de ces fichiers, il faut changer le numéro de version comme ceci, le cache sera automatiquement régénéré.
L'autre point à connaître, concerne les fichiers « .htaccess », « cache.manifest », « webapp.manifest » qu'il n'est pas nécessaire de cacher, car ils sont déjà utilisés au niveau général de l'environnement.
L'opération qui vient d'être présentée dans cette étape va télécharger les fichiers sur l'appareil. Ainsi, l'application fonctionnera même lorsque celui-ci n'est plus connecté à Internet.

ÉTAPE 3 : LES TESTS

Il s'agit d'une étape importante dans le processus, car si vous ne testez pas votre application web, des effets de bords non souhaités peuvent apparaître.
C'est pourquoi la vérification est nécessaire.
Ces tests s'effectuent avec le simulateur de FirefoxOS qui est disponible comme plug-in pour le navigateur Firefox. Nous vous conseillons de vous rendre sur l'article précédent, qui explique les différentes étapes pour utiliser ce simulateur.

ÉTAPE 4 : DÉPLOIEMENT

Lorsque les 2 précédentes étapes ont été réalisées, il est important de signaler que votre application est disponible. Pour cela, vous devez l'envoyer au 'Market Place' de Firefox OS, qui permet que votre projet soit disponible auprès des millions d'utilisateurs à travers le monde. Pour se faire référencer dans le 'Market Place' de Firefox OS, vous vous rendez à l'adresse Internet suivante : https://marketplace.firefox.com/

Nous vous conseillons de vous rendre sur l'article précédent, qui montre les différentes possibilités qu'offre le Market Place.

Ressources

Manifest
https://developer.mozilla.org/fr/docs/Applications/Manifeste

Documentation du cache
https://developer.mozilla.org/fr/docs/Utiliser_Application_Cache

Simulateur Firefox OS :
http://people.mozilla.org/~myk/r2d2b2g/

Article source original de David Walsh
http://davidwalsh.name/firefox-app

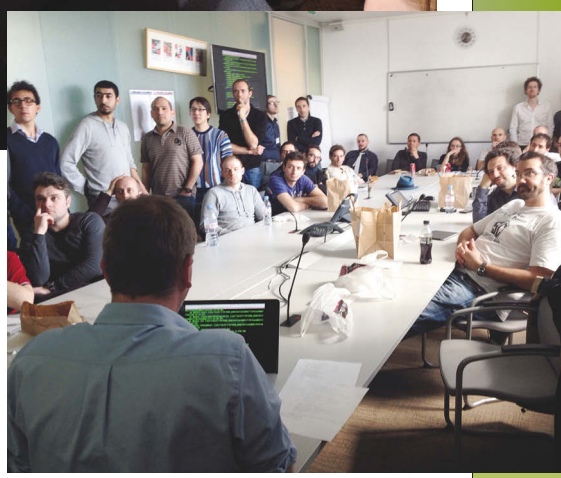
L'ensemble des sources et la structure du dossier sont disponibles en téléchargement sur le site Programmez

Conclusion

Il est très facile de rendre une application disponible sans être connecté. L'avantage de s'appuyer sur le cache permet d'étendre les possibilités des applications de type WebAPP pour vos jeux, animations, applications.... Ainsi, il n'est pas nécessaire d'effectuer de développement supplémentaire ou d'utiliser un SDK spécifique. Enfin, l'ensemble du projet utilise toutes les fonctionnalités HTML 5, qui évoluent au fil des versions de l'OS de Firefox.

Christophe **Villeneuve**
Consultant IT pour Neuros, auteur du livre « PHP & MySQL-MySQLi-PDO, construisez votre application », aux Éditions ENI. Rédacteur pour WebRIVER, membre des Teams DrupalFR, AFUP, LeMug.fr, Drupagora, PHPTV.

Brown Bag Lunch : la pause déjeuner technique



Connaissez-vous les BBL ou Brown Bag Lunch, littéralement déjeuner avec un sac brun ? Le principe est simple : durant le déjeuner, une mini conférence technique est organisée autour du panier-repas brun, les fameux brown bag lunch. L'événement dure environ 1 heure. Vous trouvez un speaker, vous lui achetez son déjeuner, vous trouvez une salle de réunion. C'est tout. Devant 10-20 personnes, le speaker parle d'un sujet défini à l'avance. Une autre manière de déjeuner. En France, les BBL se développent un peu partout. Retours d'expériences !

Importateur du BBL dans le nord de la France

Cyril Lakech, développeur Web et Java chez Adeo, créateur et organisateur du Ch'ti JUG et de Devoxx4Kids Lille, speaker à Devoxx France sur Git++, importateur du concept BBL dans le nord de la France. Il revient sur son expérience.

« J'ai vu se développer les BBL via twitter et ça m'a tout de suite donné envie d'en organiser à Lille. J'ai donc référencé mon entreprise sur la liste des lieux se proposant d'en accueillir sur le site : http://www.brownbaglunch.fr/locations.html#groupeadeo

Mais je n'ai pas reçu de propositions immédiatement, alors je suis parti à la pêche aux sujets qui pourraient intéresser la communauté locale des développeurs.

Un camarade de classe m'a proposé de venir présenter un sujet autour des services proposés par Amazon avec AWS. C'était notre premier BBL chez Adeo, et on a tout de suite eu 25 personnes. Mathieu nous a présenté comment créer une plateforme de crawling de site en utilisant les différents services d'AWS comme EC2, S3, SQS, IAM... c'était très technique et ça a beaucoup plu. Certaines personnes ne connaissaient presque pas les services d'Amazon et grâce à ce BBL, ils pouvaient lister plusieurs services majeurs de la plateforme. Cool !

Puis on s'est aperçu que pas mal de monde semblait intéressé par NodeJS sur le réseau collaborateur interne de l'entreprise. Alors on a organisé un BBL sur le format openspace conférence. Toutes les personnes intéressées par le sujet NodeJS pouvaient venir pour participer à un atelier d'échange pour comprendre, débattre ou poser des questions. Cette édition a eu un franc succès avec plus de 30 participants. C'était désorganisé, mais c'était assez riche dans le contenu. Pas moins de 20 personnes ont pris la parole lors de ce BBL et une dizaine de discrets ont simplement écouté ce qui se disait sur le sujet. Encore une fois, le concept a plu et les participants demandaient déjà quand serait la prochaine édition !

Du coup, avec mon équipe de développement, on a voulu préparer une présentation d'AngularJS; ce framework intéresse beaucoup de monde dans notre entourage, et il y avait une forte demande pour qu'on adresse ce sujet lors d'un BBL. C'est donc une présentation avec 4 speakers qu'on a organisée. Une fois encore une trentaine de personnes sont venues assister à ce BBL pour découvrir AngularJS. C'était très enrichissant de préparer et de présenter cette session, car même si le contenu de la pré-

sentation était assez simple, cela nous a aidés à comprendre comment expliquer AngularJS à d'autres développeurs. On en a profité pour dire tout le bien qu'on pense de ce framework, et au passage, on a quand même appris quelques astuces sur le sujet en préparant notre présentation.

Avec Hubert Sablonnière, nous avions été retenus pour être speakers à Devoxx France avec un sujet autour de l'amélioration de la qualité des projets avec l'utilisation de Git. Quoi de mieux qu'un BBL pour tester notre présentation avant Devoxx France pour récolter des retours et pour répéter nos blagues devant un public moins conséquent qu'une salle de 200 personnes ? On a présenté notre sujet devant une vingtaine de personnes, ce qui nous a aidés à améliorer notre présentation en récoltant de nombreux retours très intéressants auprès du public. À la suite de ce BBL, nous en avons profité pour échanger autour de l'organisation d'un hackathon chez Adeo en juin prochain.

On a donné ce même BBL sur Git avec Hubert dans une autre société, Ikomobi, un peu plus tard pour encore mieux se préparer et partager notre vision sur la qualité de l'historique du code source avec Git.

On ne connaît pas encore les dates des prochains sujets, mais on va très certainement reparler de NodeJS avec une présentation de sa mise en application sur un projet avec le framework KrakenJS. Le sujet MongoDB intéresse aussi beaucoup de monde, à suivre.

En résumé, les BBL sont devenus une excuse pour la communauté des développeurs pour se réunir et échanger autour d'un verre après les présentations. C'est l'occasion d'apprendre des choses pendant le temps du midi. C'est également un formidable laboratoire pour se préparer à donner une conférence pour tester une présentation. Les BBLs permettent à ceux qui souhaitent partager leur savoir avec les autres de faire connaître les bonnes pratiques et évangéliser sur des sujets variés. Bref, les BBLs c'est un moyen incontournable pour animer les communautés de développeurs ! »

Romain Linsolas,

DÉVELOPPEUR ET LEADER TECHNIQUE SPÉCIALISÉ JAVA, WEB ET USINE LOGICIELLE POUR SOCIÉTÉ GÉNÉRALE.

« Je suis également speaker occasionnel, en particulier pour Devoxx et Devoxx France. Je porte les deux casquettes des brown bag lunches : d'une part je suis bagger, c'est-à-dire que je présente des sujets auprès d'entreprises qui en font la demande, et d'autre part je joue le rôle d'hôte en accueillant des présentateurs à la Société Générale.

Mon aventure a commencé en novembre 2012 par un billet de David Gageot sur son blog (http://blog.javabien.net/2012/11/05/bbl/). En passant indépendant, il proposait de venir faire des BBL dans les entreprises. Connaissant David, j'ai sauté sur l'occasion en l'invitant début 2013 à la SG. C'est à peu près à ce moment-là que le mouvement BBL est né sur Paris, avec la création du site http://www.brownbaglunch.fr par Nathaniel Richand. Cela m'a poussé à contacter certaines de mes connaissances pour venir parler devant mes collègues. Le mouvement était ainsi lancé à la SG. Le hasard du calendrier a fait que c'était bien un David qui fut notre premier bagger à la SG, mais il s'agissait de David Pilato, venu nous présenter le moteur de recherche ElasticSearch. Pour l'anecdote, il faut savoir que David Pilato est un accroc des BBL, il doit facilement en présenter un par semaine.

Finalement, après une quinzaine de mois de BBL chez nous, on peut en faire un bilan extrêmement positif. Nous avons accueilli une trentaine d'intervenants sur des sujets très variés : bases NoSQL, langages alternatifs comme Scala, problématiques de tests (en particulier en JavaScript), etc. Désormais, à chaque session, nous atteignons les cinquante personnes présentes dans la salle, et on se voit malheureusement obligé de refuser du monde !

Nous avons eu plusieurs fois l'occasion de nous poser la question de l'affluence très importante chez nous. Que faire pour permettre à tous de bénéficier de ces BBL ? Louer, de plus grandes salles - voire des amphithéâtres - a été évoqué. Personnellement, je m'y suis opposé. D'une part à cause de la complexité induite par la location de telles structures, et d'autre part parce que cela ne correspondrait plus à l'idée que je me fais de ces événements : quelque chose de convivial, organisé et exécuté dans une ambiance bon enfant. Malgré la présence de nombreuses personnes, je crois que nous avons su garder cet état d'esprit, et c'est cela que j'apprécie dans les BBL.

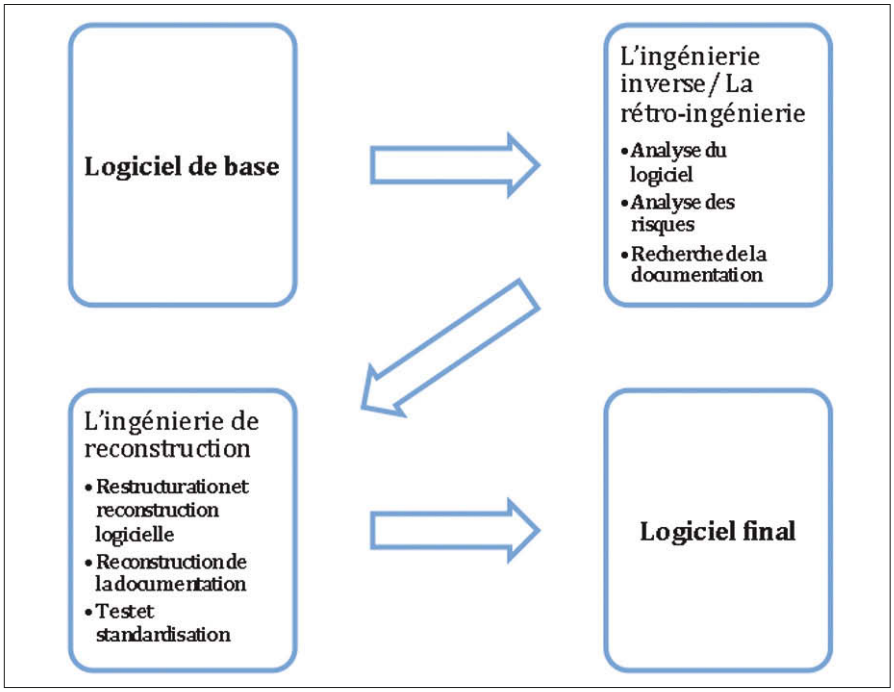
Cette année, ma société a été sponsor de la conférence Devoxx France. Nous avons eu la chance d'avoir une demi-douzaine de sessions retenues par le comité. Afin de se préparer le mieux possible, nous avons décidé d'utiliser ce format du BBL pour permettre à nos présentateurs de se rôder, et ainsi de peaufiner leurs présentations. Quand on voit les retours très positifs qui ont été faits sur nos présentations durant la conférence, on se dit que c'est aussi là le but de devenir un bagger : s'améliorer et avoir des retours constructifs sur les présentations.

Que voudrais-je améliorer ou changer ? Pas grand-chose à vrai dire. J'aimerais toutefois profiter de cette structure pour pousser des personnes en interne à devenir de vrais présentateurs. J'ai remarqué que certains hésitent à se lancer par timidité ou parce qu'ils pensent ne rien avoir à dire d'intéressant. Leur permettre d'exécuter une courte présentation - 10 à 15 minutes maximum - pourrait être un moyen de changer cela, et qui sait, de peut-être faire naître des vocations !

Un point qui me tiendrait à cœur concernant les BBL : sortir du domaine technique. Aujourd'hui, tous les BBL que j'ai organisés sont des BBL tournant autour du développement logiciel ou de la technique en général. J'aimerais tellement pouvoir proposer des BBL sur des sujets complètement différents, comme l'astronomie, la physique, etc. Mais cela est plus compliqué à mettre en place, déjà par manque de contact. Je vois l'initiative très courageuse d'Aude Amarrurtu et Betty Moreau qui ont lancé l'année dernière les BBL pour le monde des RH. Nous autres développeurs, avons la chance de vivre dans un milieu où le partage est omniprésent et même encouragé. L'initiative des BBL s'inscrit parfaitement dans cet état d'esprit, et faire de ce mouvement un succès n'est pas en soi si compliqué. Nous avons ainsi accueilli notre centième bagger inscrit sur le site http://www.brownbaglunch.fr début mai. Mais dans les autres domaines, en particulier celui des RH, cette notion de partage n'est pas inscrite dans les gènes. Cela fait que je suis encore plus admiratif du mouvement BBL RH, et je serais le premier ravi de voir des BBL apparaître sur de nouvelles disciplines ! ».

Refactoring, les fondamentaux

Il existe un terme français exact pour exprimer l'idée de refactoring : réusinage. Bien que peu employé, ce terme est important car il permet de comprendre le concept derrière un mot souvent utilisé de manière inexacte pour désigner toute modification de code existant.



Au même titre qu'un objet issu d'une chaîne de fabrication qui ne correspond pas aux standards de qualité retourne à l'usine pour y être retravaillé ou refondu, un morceau de code peut être réusiné. Cette idée est la pierre angulaire du refactoring : retravailler un morceau de code sans changer ses fonctionnalités originales.

Dans cet article, nous allons étudier les diverses motivations menant à cette démarche, la stratégie à adopter pour minimiser les risques des opérations de refactoring et enfin les différentes techniques utilisées pour mener à bien ces opérations.

Motivations

Les opérations de refactoring peuvent avoir diverses motivations, que l'on peut répartir en trois grandes catégories :

Faciliter la maintenance. Votre code doit fonctionner à un instant T, mais doit aussi permettre à un autre développeur d'apporter des évolutions ou de corriger des anomalies. Pour optimiser le temps d'intervention, le code doit être compréhensible immédiatement. Pour ce faire, il existe trois angles de travail : la lisibilité, la modularité et la factorisation du code.

- La lisibilité comprend toutes les modifications de forme telles que la précision du nommage, l'espacement du code (éviter de compresser un enchainement d'opérations en une seule ligne peu lisible). Ces attentions, répétées sur des milliers de lignes de code, améliorent grandement le temps de compréhension du code et donc la productivité.
- La modularité correspond au découpage de la logique en entités facilement testables et réutilisables. Ceci facilite aussi la localisation d'anomalies.
- La factorisation de code permet de centraliser une logique, c'est-à-dire un même code qui était initialement présent à plusieurs endroits dans les sources. Par conséquent, un code bien factorisé améliore la testabilité. Le cas échéant, il ne sera donc plus nécessaire de reporter une correction d'anomalie à chaque endroit où le code était présent, limitant ainsi la probabilité d'oubli sur le long terme.

Permettre d'étendre votre application. Dans une application réutilisable (framework, bibliothèque...), votre code devra fournir la possibilité d'être étendu.

Pour cela, dans les langages objets, on peut permettre par exemple l'héritage ou la délégation.

Optimiser les performances. Une fois que votre code fonctionne correctement, qu'il est maintenable et réutilisable, le refactoring de vos algorithmes permet d'optimiser le temps de calcul ou la consommation de ressources afin d'améliorer l'expérience utilisateur (cas d'une application front), ou encore de minimiser les délais de traitement (cas d'un traitement de masse ou d'accès à des bases de données).

Ces motivations sont sous-tendues par une motivation générale : la motivation économique. En effet, la maintenabilité permet d'effectuer facilement des corrections ; l'extensibilité permet d'ajouter des fonctionnalités à moindre coût et l'optimisation permet d'économiser des ressources et d'améliorer la productivité de l'utilisateur. Ces trois axes de travail diminuent les coûts liés aux travaux de développement.

Voyons maintenant comment réaliser les opérations de refactoring en réduisant au maximum les risques de régression.

Stratégie

Bien que n'étant pas destinée à ajouter de nouvelles fonctionnalités, l'opération de refactoring n'est pas anodine. Elle vise à modifier du code et peut donc causer des régressions comme tout développement. Changer tout un pan de code d'un coup est donc proscrit : il est nécessaire de procéder prudemment.

Afin de se prémunir contre les régressions, ou tout au moins d'être en mesure de les identifier le plus tôt possible, le code qui va être modifié doit être couvert par des tests automatisés (unitaires et d'intégration). Si exceptionnellement de tels tests n'existent pas déjà, ou ne sont pas complets, les écrire est la première chose à faire. On parle dans ce cas de tests de caractérisation. On comprend alors l'importance d'écrire les tests au moment du développement : cela facilite le travail des développeurs qui travaillent après en leur offrant une protection contre les régressions et en ne leur déléguant pas toute la charge de travail de l'écriture des tests.

Quand le harnais de test est présent, tout refactoring doit être effectué de manière itérative, en procédant pas à pas. De cette manière, on effectue une petite modification, on teste, on passe à la modification suivante, et si celle-ci cause une anomalie, on revient un cran en arrière et on reprend en changeant d'approche.

Comme pour le code « de production », le code des tests doit rester lisible et bien découpé. En effet, celui-ci vivra aussi longtemps que le code testé et sera donc aussi sujet aux interactions avec les autres développeurs. De plus, lors-

qu'ils sont bien découpés et nommés, il est aisé d'identifier la portion de code qui est en cause si l'un d'entre eux échoue, sans lire le code source du test.
Enfin, si les tâches de refactoring peuvent paraître fastidieuses, c'est parce qu'elles ne sont pratiquées que de manière sporadique, en dernier recours. Pratiqué régulièrement, par petites touches au cours de chaque nouveau développement, le refactoring permet d'avoir constamment du code propre. Cela est comparable au fait de ranger son bureau : si cette tâche est effectuée régulièrement, on trouve rapidement le document, stylo ou ustensile recherché. Sinon, les documents s'entassent anarchiquement, masquant d'autres éléments, et font que la moindre recherche ou utilisation du bureau est pénible et nuit à la productivité.

Techniques phares

Les tâches de refactoring peuvent également être rendues plus faciles par la maîtrise de différentes techniques à utiliser suivant la nature du refactoring à effectuer.
L'extraction est le groupe de techniques le plus fréquent. L'extraction de code consiste à déplacer un morceau de code cohésif, c'est-à-dire ayant un rôle précis, dans une nouvelle unité de code réutilisable. Cette unité peut être une variable locale, une fonction ou une unité de compilation (appelée classe en paradigme objet). Cela permet de découper une unité de code trop longue, donc peu lisible, ou bien de factoriser du code redondant.
Quel que soit le type d'extraction choisi, la procédure reste la même :

- isoler un morceau de code cohésif,
- créer la nouvelle entité (variable, fonction ou unité de compilation),
- nommer cette dernière,
- déplacer le code identifié dans la nouvelle entité.

Le nommage est capital : c'est l'équivalent d'un commentaire associé au bloc de code extrait qui est présent à chaque emploi. Le nom doit donc expliciter l'intention de ce morceau de code de façon très précise et lisible, il doit ainsi bien s'intégrer dans ses contextes d'utilisation.
Cependant, chaque type d'extraction a ses particularités.

- L'extraction de constante permet de centraliser les valeurs constantes comme les libellés ou les codes numériques afin de faciliter leur réutilisation et leur modification.
- L'extraction de variable locale intervient quand une expression devient trop complexe et nécessite une explication. C'est encore plus important quand cette expression est utilisée à plusieurs endroits du code. Si elle ne dépend pas du contexte d'exécution, c'est une constante qu'il faut extraire.
- L'extraction de fonction sert à faciliter la compréhension au premier coup d'œil ; attention, pour ce faire, il faut minimiser le nombre de paramètres.
- L'extraction de classe remplit le même objectif quand une classe devient trop longue ou trop complexe. Dans ce cas, il faut faire particulièrement attention à bien conserver les fonctionnalités existantes au moment de l'extraction grâce au harnais de tests explicité ci-dessus. Il faut également penser à répercuter ce changement au niveau des tests unitaires. Un autre avantage est la généralisation ainsi obtenue.

Le déplacement consiste à déplacer du code dans un endroit plus pertinent. Concrètement, cela peut être le déplacement d'une méthode statique vers une classe utilitaire, ou d'une méthode d'instance dans la classe d'un objet du contexte courant.
Il existe deux cas particuliers dans le contexte de la programmation objet :

- *pull up*, déplacement depuis la classe courante vers sa super-classe,
- *push down*, depuis la classe courante vers une sous-classe.

Le renommage peut concerner une classe, une méthode ou un attribut, après une revue de code, un changement de comportement de la méthode ou pour corriger une faute d'orthographe. Comme nous l'avons vu plus haut, le nommage est capital pour faciliter la compréhension du code pour les autres développeurs. Si votre API est publique, c'est à dire utilisée comme du code n'étant pas le vôtre, il est important de bien vérifier les conséquences de votre renommage.
Cependant, toutes ces différentes techniques sont dorénavant automatisables dans les environnements de développement intégrés (IDE) grâce à quelques outils clés.

IDE et outils

Dans les IDE les plus courants (Eclipse, IntelliJ IDEA, Netbeans), les techniques explicitées plus haut sont directement intégrées.
Ces environnements de développement s'occupent de la propagation des modifications dans le reste du code. Par exemple, quand on renomme un attribut (en paradigme objet), les accesseurs sont automatiquement et immédiatement renommés. La modification peut aussi être propagée dans les occurrences textuelles (commentaires).
De plus, les outils de refactoring intégrés aux IDE sont capables de prévenir certaines erreurs comme l'écrasement d'une variable dans la portée, ou scope, courante.
Ces outils offrent également la possibilité de prévisualiser les changements dans les fichiers sources induits par un refactoring. Ceci s'avère particulièrement utile dans le cas où l'opération de refactoring est sensible, comme par exemple lorsqu'elle touche des occurrences textuelles, c'est-à-dire des morceaux de code ne bénéficiant pas de système de typage fort vérifié à la compilation (fichiers de configuration, fichiers de templating, langages dynamique) ainsi que les commentaires.
Les spécificités entre langages ont une influence sur les outils de refactoring, qui restent difficiles à implémenter pour les langages à typage faible (tel que JavaScript). Cependant, certains IDE modernes, comme IntelliJ IDEA, mettent tout de même à disposition des outils pour de tels langages.
Enfin, même si le support du refactoring reste le point fort des IDE, les éditeurs de texte avancés proposent des fonctionnalités comparables. On peut mentionner par exemple la sélection multiple dans Sublime Text (multiple selections) et la commande de substitution (substitute) dans Vim.

Conclusion

Pour conclure, 3 aspects essentiels du refactoring sont à retenir.
Le refactoring facilite la maintenance, améliore l'extensibilité et les performances de vos applications, en un mot le refactoring diminue les coûts liés au développement.
Il est nécessaire de procéder avec prudence, avec un harnais de tests unitaires soignés et en progressant par petites touches.
Les techniques de refactoring courantes (extraction, déplacement et renommage) sont automatiquement exécutées par les IDE modernes et facilitées par les éditeurs de texte avancés.
Cet article s'attachait à présenter les principales techniques, applicables à tous les langages de programmation, mais chaque paradigme apporte ses propres techniques, comme par exemple les langages objet (extraction d'une super-classe ou d'une sous-classe, remplacement de l'héritage par la délégation, remplacement d'une condition par du polymorphisme, etc.) ou encore les langages fonctionnels (utilisation de fonctions d'ordre supérieur, monadification...)
Tout cela fait du refactoring une pratique incontournable du software craftsman, ce développeur moderne qui améliore constamment la qualité, l'efficacité et la pérennité de ses programmes, qui choisit soigneusement ses outils et les maîtrise parfaitement pour appliquer son savoir-faire.

Bastien **Bonnet** et Christophe **Pelé**
Xebia IT Architects

Du son dans vos apps avec le SDK Javascript de Deezer

L'API Deezer est un véritable miroir du site et de ses applications mobiles. Elle permet non seulement d'accéder aux metadata du catalogue de 30 millions de titres, mais aussi de streamer de la musique, d'enrichir la bibliothèque musicale des utilisateurs, d'accéder à l'historique d'écoutes, aux recommandations, etc. Si vous n'êtes pas allés sur Deezer depuis longtemps, c'est le moment !

Il n'est plus nécessaire de présenter Deezer, le service de musique en streaming français compte plus de 12 millions d'utilisateurs actifs, répartis dans plus de 180 pays.

L'API, quant à elle, a vu le jour en mai 2012 et son histoire ne déroge pas à la règle. D'abord mise en place pour les partenariats avec les constructeurs, celle-ci a ensuite été ouverte à tous les développeurs souhaitant intégrer de la musique dans leurs sites et applications mobiles. Les applications de Sonos, TuneWiki, Soundrop, ou encore Chordify, en sont quelques exemples.

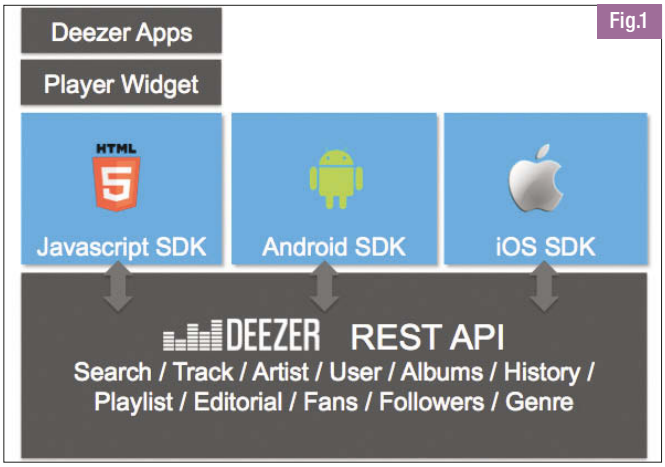
Introduction

L'API Deezer est exposée en REST. Pour la consommer, il suffit de faire une requête HTTP. Il est toutefois plus simple de passer par les SDK pour faire les requêtes API dans la mesure où ils gèrent automatiquement l'authentification et les permissions des utilisateurs Fig.1.

En résumé, les SDK servent 3 fonctions principales :

- le **player,** avec les méthodes permettant de lire des chansons, playlists, radios, etc. Pas besoin d'être un expert en streaming, c'est Deezer qui prend en charge la complexité du chargement.
- l'**authentification** des utilisateurs, avec support d'OAuth 2.0,
- les **requêtes API**, à utiliser pour tout le reste : les metadata sur le catalogue, les infos utilisateurs, etc.

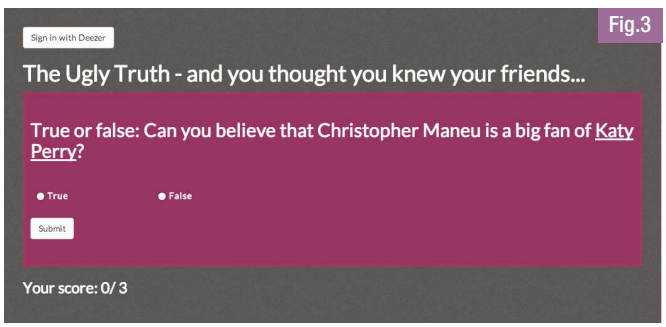
Nous allons détailler ici le SDK Javascript, qui servira pour vos sites web,



mais aussi pour développer des Deezer apps (applications à l'intérieur du site et des applications mobiles Deezer, dans l'App Studio).

Notez que si vous avez « juste » besoin d'un player web, il existe aussi un widget que vous pouvez créer à partir de n'importe quel album, chanson ou playlist, dans le format que vous voulez. Il est responsive, son format s'adaptera au conteneur dans lequel il se trouve Fig.2.

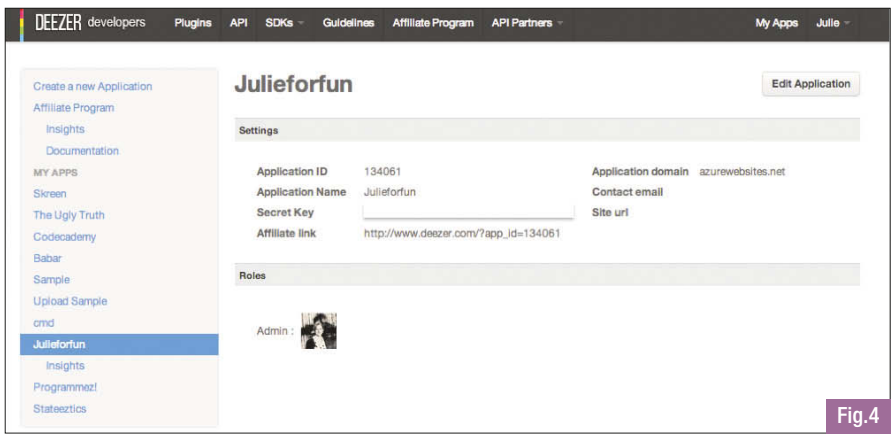
Reprenons par exemple l'application montrée lors de la session Coding4Fun aux Microsoft TechDays. *The Ugly Truth* fait un quiz sur les écoutes de vos amis. Sans rentrer dans tous les détails de la réalisation de cette application, nous allons voir comment connecter un utilisateur, récupérer son historique d'écoute et enfin utiliser le player Fig.3.



Créer une application Deezer

Pour commencer, il faut d'abord s'identifier en tant que développeur et créer une application sur http://developers.deezer.com. Il vous faudra pour cela un compte Deezer, que vous pourrez créer gratuitement.

Cela vous permettra de récupérer un *Application ID* que vous utiliserez pour initialiser le SDK Fig.4.



Charger et initialiser le SDK

Tout d'abord, il va falloir indiquer à votre site où trouver le SDK en ajoutant un lien vers le script :

```
<script src=»http://cdn-files.deezer.com/js/min/dz.js»></script>
```

Une fois intégré, il faut maintenant initialiser votre application avec deux paramètres :

- l'**app id**, récupéré précédemment sur Deezer Developers,
- une **channelUrl**, qui doit pointer vers une page du site, qui ne contiendra que le lien vers le script du SDK :

```
<script src=»http://cdn-files.deezer.com/js/min/dz.js»></script>
```

Avoir cette url est obligatoire car, en plus de permettre l'amélioration de la communication entre le player et votre site (notamment sur les navigateurs anciens), elle est indispensable au bon fonctionnement de l'authentification.

En effet, le SDK utilise un système d'iframe afin de communiquer avec Deezer lors du processus d'authentification. Pour ne pas polluer la structure HTML de l'application, il est indispensable de créer une balise dans laquelle le SDK pourra charger ces iframes. Pour qu'il puisse l'utiliser, il faudra lui attribuer l'id « dz-root ».

Attention à ne pas créer cette balise après l'exécution de DZ.init.

```
<div id=»dz-root»></div>
<script>
DZ.init({
appId : 'VOTRE_APP_ID',
channelUrl : 'http://VOTRE_DOMAINE/channel.html',
player : {
onload : function(){}
}
});
</script>
```

Authentification et permissions

Authentifier les utilisateurs permet de faire des requêtes avancées sur l'API, comme accéder à leur historique d'écoute. En effet, la fonction de login prend en paramètre la liste des permissions souhaitées. Ici il sera nécessaire de demander la permission '`listening_history`' pour avoir accès à l'historique : pour des raisons de confidentialité, il est impératif que l'utilisateur donne son accord. L'accès au catalogue ou la recherche sont publics et peuvent se faire sans être connecté.

Un autre intérêt, et pas des moindres, les abonnés Deezer Premium pourront écouter les chansons entières sans publicité.

Que vous soyez sur desktop ou sur mobile, appeler la fonction **`DZ.login`** va automatiquement ouvrir une nouvelle fenêtre de connexion. L'ouverture des popup en Javascript étant bloquée par les navigateurs, il ne faudra exécuter cette méthode que lors d'un évènement « click ».

Suite à la connexion, le retour à l'application est automatique à condition d'avoir bien paramétré la channelUrl. Attention par exemple à ne pas mettre le préfixe *http://* lors de la configuration de l'application sur le portail Deezer Developers Fig.5.

Fig.5

```
function login(){
  DZ.login(function(response) {
    if (response.authResponse) {
      DZ.api('/user/me', function(response) {
        console.log('Good to see you, ' + response.name + '.');
      });
    }
  }, { perms: 'email, basic_access, listening_history' }) ;
}
document.getElementById('dz-login-btn').onclick = login ;
```

La fonction renvoie notamment le user token : la clé d'accès pour faire les requêtes sur l'API nécessitant des autorisations spécifiques. Une fois l'utilisateur connecté, ce user token sera directement intégré dans les requêtes faites sur l'API.

Utiliser l'API

Pour explorer tous les contenus de l'API, vous pouvez aller voir la documentation complète : http://developers.deezer.com/api/explorer

Il existe une méthode commune pour tous les points d'entrée : `DZ.api`, qui gère le cross-domain et qui inclut le user token.

Dans le cas qui nous intéresse, à savoir récupérer l'historique :

```
DZ.api('/user/me/history', function(response){
….
}
```

qui retourne le résultat au format json :

```
{
  «data»: [
    {
      «id»: «14111875»,
      «readable»: true,
      «title»: «Omen»,
      «link»: «http://www.deezer.com/track/14111875»,
      «duration»: «216»,
      «preview»: «http://cdn-preview-5.deezer.com/stream/545f4804596d5859347b74834821f5d3-1.mp3»,
      «timestamp»: 1395307974,
      «artist»: {
        «id»: «85»,
        «name»: «The Prodigy»,
        «link»: «http://www.deezer.com/artist/85»,
        «type»: «artist»
      },
      «album»: {
        «id»: «1293327»,
        «title»: «Invaders Must Die»,
        «link»: «http://www.deezer.com/album/1293327»,
        «cover»: «https://api.deezer.com/album/1293327/image»,
        «type»: «album»
```

```
      },
      «type»: «track»
    },
    {
      «id»: «4198565»,
      «readable»: true,
      «title»: «Poison Lips»,
      «link»: «http://www.deezer.com/track/4198565»,
      «duration»: «234»,
      «preview»: «http://cdn-preview-6.deezer.com/stream/67aec0699db8f33f2b58586331f6e7de-0.mp3»,
      «timestamp»: 1395307787,
      «artist»: {
        «id»: «641»,

….}
```

Comme la liste peut contenir des dizaines de milliers de chansons, l'API renvoie un résultat paginé avec un attribut **next** pour accéder aux résultats suivants. Vous pouvez utiliser les fonctions de pagination avec les paramètres **index**, pour préciser à partir de quel index vous souhaitez récupérer, et **limit**, le nombre d'objets à retourner.
Voici quelques exemples d'autres requêtes que vous pouvez effectuer :

- faire une recherche : `DZ.api('search?q=metronomy', callback);`
- créer une playlist : `DZ.api('user/me/playlists', 'POST', {title : «my title»}, callback);`
- accéder aux recommandations : `DZ.api('user/me/recommendations/playlists', callback) ;`
- voir les tops - du pays depuis lequel la requête est émise : `DZ.api('editorial/0/charts', callback) ;`

Utiliser le player

Pour jouer la chanson de votre choix, une seule ligne de code suffit :

```
DZ.player.playTracks([75296931]) ;
```

Les id des chansons sont directement accessibles depuis le site www.deezer.com, dans l'url indiquée lors du partage.
Le player dispose de fonctions spécifiques pour chaque type de contenu : `playPlaylist` pour les playlists, `playAlbum` pour les albums, etc.
Remarque : Le respect des droits sur la musique impose l'utilisation de Flash Player, qui est donc nécessaire à la lecture des chansons en entier. Quelques détails sur les droits : seuls les utilisateurs authentifiés pourront accéder aux chansons en entier. Les autres devront se contenter des extraits de 30 secondes. Voici un exemple complet avec le player :

```
<!DOCTYPE html>
<html>
<head>
   <script  src=»http://cdn-files.deezer.com/js/min/dz.js»>
</script>
</head>

<body>
   <div id='dz-root'></div>

 <input type=»button» onclick=»DZ.player.playAlbum(6125830)
;» value=»Play»/>
 <script>
   DZ.init({
     appId : 'YOUR_APP_ID',
channelUrl : 'http://YOUR_DOMAIN/channel.html',
     player: {
           onload: function () {}
           }
   });
 </script>
</body>
</html>
```

Le SDK permet de créer et personnaliser entièrement le player. En plus de fonctions basiques comme play/pause/prev/next, il dispose de fonctions pour :

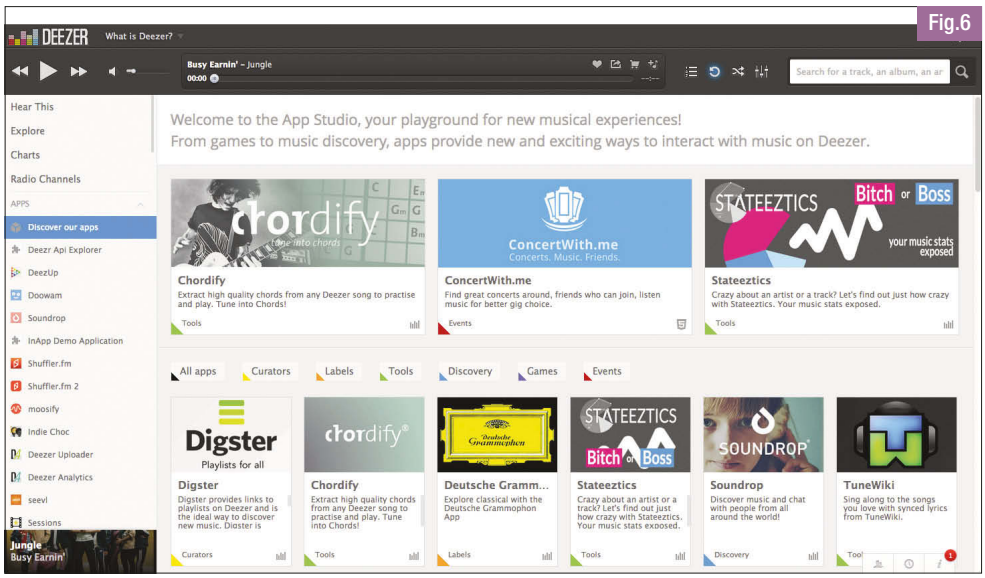


Fig.6

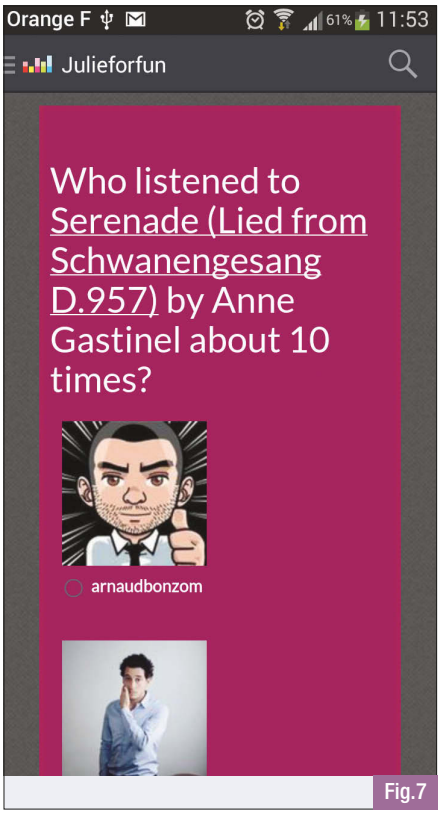


Fig.7

- contrôler le volume, DZ.player.setVolume,
- mettre en shuffle/repeat, DZ.player.setShuffle/setRepeat

Vous avez aussi à votre disposition des événements vous informant sur son état : savoir si la chanson est terminée ou pas, si le player est en train de jouer, où en est sa position, etc.

```
DZ.Event.subscribe('player_position', function(arg){
    ..
 });
```

1 code, 3 apps : l'intégration au Deezer App Studio

L'App Studio, lancé en novembre 2012, est l'espace permettant aux applications de s'intégrer directement dans Deezer (aussi bien le site web que les applications iOS et Android). Il compte maintenant plus d'une centaine d'applications. Ces dernières sont accessibles depuis le menu principal de Deezer, pour donner de la visibilité aux projets auprès de tous les utilisateurs du service.

Remarque : L'accès au test des applications pour Deezer est encore privé, mais un simple email à deezerdevs@deezer.com *vous permettra de récupérer un accès. Deezer se réserve ensuite la liberté de valider ou non les applications pour les mettre en ligne sur l'App Studio* Fig.6.

C'est là tout l'intérêt du SDK Javascript de Deezer : il fonctionne aussi bien dans un site web tiers qu'à l'intérieur de Deezer, qu'il s'agisse du site web ou des applications mobiles. En effet, il va détecter où vous vous situez, et en fonction, il va soit contrôler le player de deezer.com ou des applications natives, soit utiliser le player du SDK.

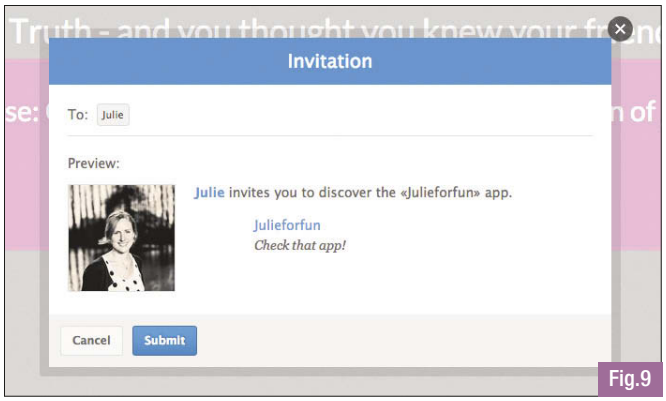


Fig.9

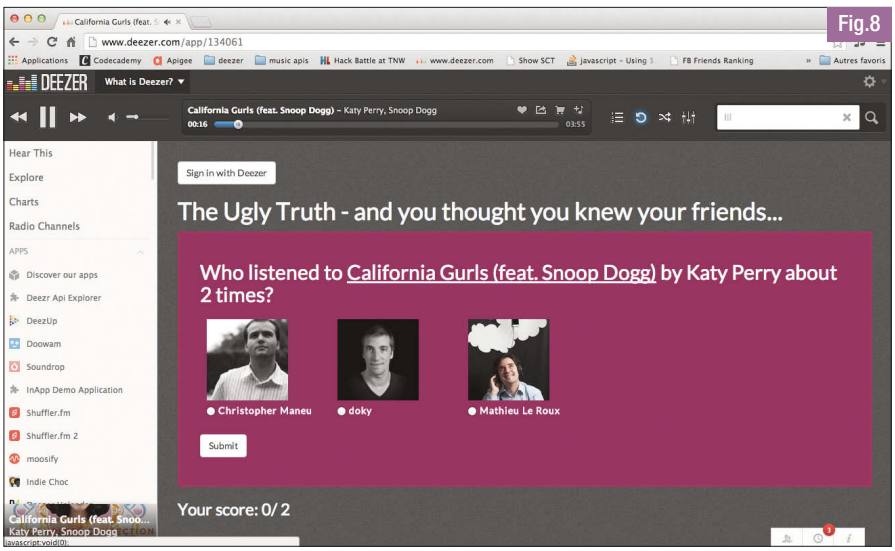


Fig.8

Cela veut dire que vous pouvez réutiliser le même code pour vos applications car vous appellerez toujours les mêmes fonctions : DZ.player.play(), DZ.api(), etc. Seul le chargement va différer légèrement, tous les détails sont précisés dans la documentation Fig.7 et 8.
Quelques fonctionnalités supplémentaires sont disponibles lorsque vous vous trouvez dans Deezer pour engager les utilisateurs :

- envoyer des **notifications**. Elles se matérialisent par un message dans la *live bar* de droite, et par un compteur à droite du nom de l'application dans le menu gauche.


```
DZ.api('user/USER_ID/notifications', 'POST', {message : «Guess who's a fan of Katy Perry ?»}, callback);
```

inviter des amis à utiliser votre application avec les **app requests** Fig.9.

```
DZ.ui.appRequest(
{
    to : 'USER_ID',
    message : «Check that app!»
},
function(response){
    console.log(response.status); // true | false
});
```

L'objectif est de fournir aux développeurs les meilleurs outils possibles pour créer une relation avec les utilisateurs.

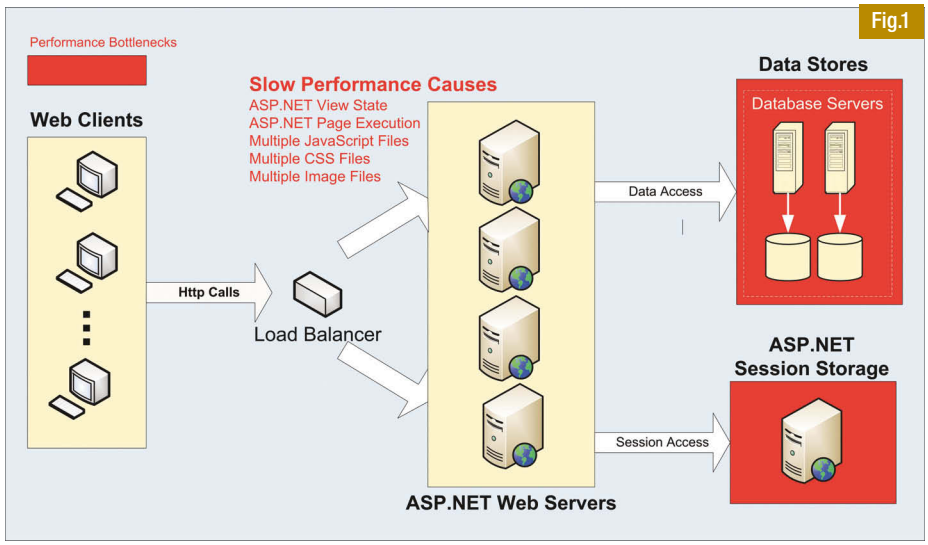
Dans ce cadre, les Deezer apps pourront bientôt recommander des contenus sur la page d'accueil « A écouter » des utilisateurs, qui regroupe toutes leurs recommandations personnalisées. Stay tuned !

Julie **Knibbe**
Platform Evangelist, Deezer
http://developers.deezer.com
http://twitter.com/julieknibbe

4 possibilités d'optimiser les performances d'ASP.Net

ASP.Net est une technologie très populaire pour développer des applications Web. De nombreux développeurs l'utilisent pour cela. Quand ces applications génèrent des trafics importants, elles sont déployées sur de multiples serveurs Web pour gérer la montée en charge, assurant un temps de réponse optimal et de hautes performances pour les utilisateurs. Mais plusieurs facteurs peuvent faire baisser les performances.

Dans cet article, nous allons nous concentrer sur l'ASP.Net View State qui ne concerne que les échanges entre le serveur et le navigateur. Plus il y a de requêtes et de View State, plus il y aura de trafic réseau entre les serveurs et le navigateur.



Les problèmes de performances sous ASP.Net

Les 4 préoccupations

Premièrement, le View State (littéralement "état de l'affichage")" peut charger le trafic et les requêtes HTTP. Le View State d'ASP.Net est un texte encodé envoyé comme un champ caché (_VIEWSTATE) dans une page ASP.Net que le navigateur reçoit.

Le View State peut peser quelques dizaines de ko, voire, plusieurs centaines. Or, il n'est pas utilisé par le navigateur. Le navigateur conserve le View State et le retourne au serveur quand un postback est généré sur l'application Web par l'utilisateur. Un HTTP postcall se génère. Le View State est un élément important pour l'application ASP.net pour afficher et rafraîchir les pages et les informations.

Deuxièmement, si l'output de la page n'est pas fréquemment modifié, la page est tout de même exécutée sur le serveur Web. Cette opération pénalise le temps de réponse de votre application.

Troisièmement, une page ASP.Net contient souvent du JavaScript et du CSS. Si c'est le cas, le navigateur émettra de multiples appels HTTP pour charger chaque fichier. Ces appels accroissent le trafic réseau. Multipliez ces appels par x milliers ou millions d'utilisateurs et les performances risquent de s'effondrer. Avoir de trop nombreux fichiers JavaScript impactent directement les performances et la bande passante.

Quatrièmement, avoir de nombreuses images à afficher dans les pages ASP.net. Le navigateur va charger séparément chaque image à travers des appels HTTP. Nous retrouvons les mêmes problèmes qu'avec le JavaScript et le CSS Fig.1.

ASP.Net View State et la charge

Quel est le problème avec le View State ? Le principal problème est son enbompoint et les multiples allers-retours serveurs – navigateurs, surtout, quand le navigateur n'a pas besoin de mettre à jour. Ce serait donc parfait si nous pouvions nous passer de View State sur le navigateur.

Pour ce faire, il faut utiliser des mécanismes de cache côté serveur et envoyer uniquement ce qui est nécessaire grâce à un système de clé. Cette clé peut être utilisée, plus tard, pour envoyer le View State quand le navigateur le demande. Nous recommandons d'utiliser la mémoire de cache distribué.

Si votre application fonctionne sur plusieurs serveurs Web, vous ne pouvez pas utiliser un cache "local". À chaque postback, la requête peut être envoyée à un serveur différent ayant son cache local. Et dans ce cas le View State caché sera introuvable (sauf à "tomber" sur le bon serveur). Un cache distribué évite ce problème et la clé est comme un identifiant permettant de localiser le View State et de l'envoyer au navigateur Fig.2.

Comme le montre la figure 2, cette technique réduit la charge. Par exemple, imaginons qu'une requête utilisateur représente un View State de 30 à 60 ko, multipliez ces ko par x millions de requêtes par jour... La mise en cache va à la fois améliorer le temps de réponse et optimiser la bande passante utilisée. Vous pouvez utiliser une solution tierce de cache distribué en mémoire. Elle sera implémentable en utilisant System.Web.UI.Adapters.PageAdapter qu'il faudra rajouter dans vos applications en modifiant le web.config. Il enregistrera le View State de la page dans le cache distribué. Au prochain postback, ce PageAdapter sera appelé et chargera le View State du cache distribué.

Les performances et le cache de sortie

ASP.Net propose un mécanisme spécifique : l'output cache ou cache de sortie. Il permet de mettre en cache une page ASP ou seulement une partie d'une page. Ainsi, si une page est appelée, elle ne sera pas réexécutée sur le serveur, car elle sera dans le cache et sera envoyée au navigateur. Gain de temps et performances assurées.

Mais si l'output ne change pas souvent et que la page semble toujours identique, ou du moins certaines parties, alors ces portions qui ne changent pas peuvent être incluses dans le cache de sortie. Ou encore, si l'intégralité de la page ne change pas, vous pouvez l'inclure dans le cache de sortie et cela vous fera économiser à la fois de la consommation mémoire vive et du temps CPU.

Avant .Net 4.0, l'architecture de l'output cache interdisait l'ajout d'un système de cache tierce. Il fallait donc utiliser un cache local sur chaque ser-

veur. Avec la disponibilité de .Net 4.0, il est désormais possible d'utiliser un cache externe. La figure 3 montre comment utiliser un cache externe :

```
<caching>
  <outputCache defaultProvider=»<OutputCacheProviderName> «>
    <providers>
      <add name=»DistributedOutputCache»
           type=»CustomOutputCacheProvider,
     DistributedOutputCacheProvider»/>
    </providers>
  </outputCache>
</caching>
```

Pourquoi le cache distribué ?

Pourquoi utiliser un cache distribué et non le cache par défaut d'ASP.Net ? Comme nous l'avons dit précédemment, le cache d'ASP.Net est local et doit être présent dans chaque worker process. Si vous avez plusieurs worker process, vous aurez donc plusieurs caches. Problème : ces caches ne sont pas synchronisés en eux. Donc, rapidement, vous allez avoir un problème d'intégration de données, car vous aurez des données plus anciennes ou plus récentes selon le cache. Le cache distribué supprime cet inconvénient. Le cache distribué saura tirer parti des ressources mémoires de vos serveurs et pourra ainsi améliorer les performances et le temps de réponse Fig.4.

Fusionner et réduire les fichiers JavaScript et CSS

Nous avons vu au début de l'article les problèmes liés à la multiplication des fichiers JavaScript et CSS. Il faut agir sur deux éléments : réduire le nombre de fichiers et réduire la taille de ces fichiers. Le JavaScript est déployé sur le serveur, mais il s'exécute sur le navigateur. Il transite donc du serveur vers le navigateur, nécessitant un trafic réseau.

Ces fichiers contiennent souvent beaucoup d'espaces et de caractères inutiles. En les supprimant, vous réduisez la taille des fichiers sans impacter le fonctionnement des codes JavaScript et Css. La 2e possibilité est de fusionner (on utilisera le terme merge / merging) les fichiers JavaScript en un seul fichier. Là encore, sans modifier le fonctionnement du code. Cette opération ne nécessite aucun développement, c'est la solution d'optimisation déployée sur les serveurs qui s'en occupera.
Ces deux optimisations évitent de multiplier les fichiers, de limiter les appels HTTP et de réduire la bande passante réseau. Gain de temps, meilleures performances et temps de réponse optimal! Cependant, la réduction de la taille des fichiers consomme du temps et des ressources.

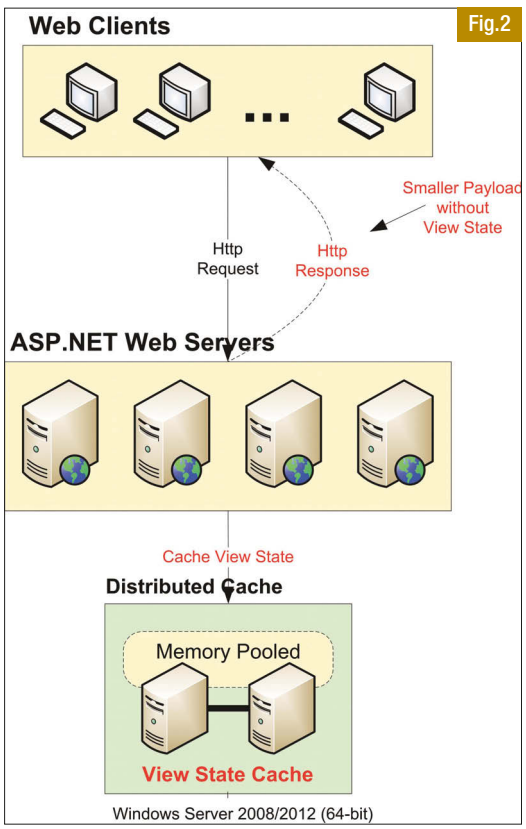
Et les images ?

Le nombre des images explose et on retombe dans les problématiques JavaScript et CSS : multiplication des fichiers images, appels HTTP. Mais quand les images sont fusionnées, il est possible d'utiliser la fonction de splicing. Il s'agit d'afficher une section de l'image selon des coordonnées précises. Par exemple : afficher une image de 0,0 à 3,4. Dans ce cas, seule la partie de l'image comprise entre ces coordonnées sera affichée. Mais en utilisant cette fonction, vous ne pouvez afficher qu'une partie de l'image.

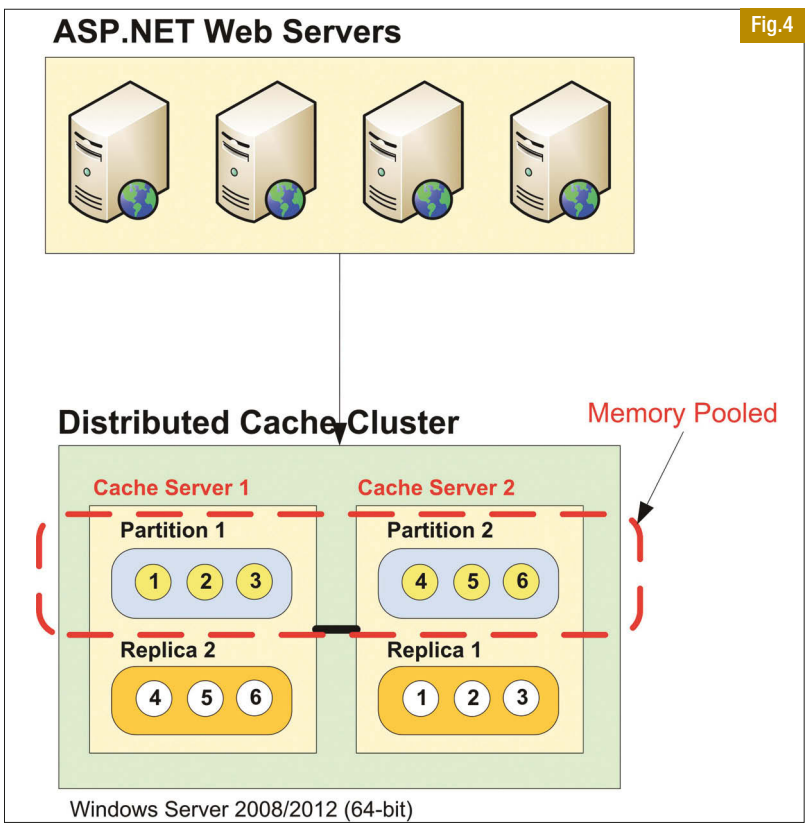
Conclusion

Dans cet article, nous avons évoqué plusieurs pistes pour améliorer les performances des applications ASP.Net. Nous recommandons l'usage d'un cache en mémoire distribué surtout quand vous déployez sur plusieurs serveurs de production. Ce cache pourra s'adapter aux ressources disponibles et vous pourrez mettre à profit l'ensemble de vos serveurs.

Iqbal **Khan**
Évangéliste technique à Alachisoft
www.alachisoft.com / iqbal@alachisoft.com
Alachisoft est spécialisé dans les solutions de cache distribué pour .Net et Java.



Le cache distribué réduit la charge réseau.



Cache distribué sur plusieurs serveurs

Pourquoi la méta-programmation va-t-elle révolutionner notre façon de coder dans le futur ?

Commençons par une définition de la méta-programmation : «Metaprogramming is the writing of computer programs that write or manipulate other programs (or themselves) as their data, or that do part of the work at compile time that would otherwise be done at runtime.» Wikipedia

Il y a donc deux parties bien distinctes dans la méta-programmation : l'analyse de code et l'utilisation de celui-ci en tant que donnée du programme que l'on écrit, et la génération de code. Bien entendu, il est possible de faire les deux en même temps. C'est exactement ce que fait par exemple un compilateur.

Pourquoi utiliser la méta-programmation ?

La génération de code est un moyen potentiellement considérable de gagner en productivité mais aussi en fiabilité, qualité et maintenabilité. Enfin j'ajouterai que cela permet également de rendre le métier de développeur plus intéressant.

La productivité, c'est évident; un ordinateur peut générer du code beaucoup plus rapidement que si nous devions l'écrire à la main. S'il est souvent plus long d'écrire du méta-code comparé à l'écriture du code, on sera très vite gagnant, dès lors que notre code sera redondant à l'intérieur même de notre projet, ou entre nos différents projets.

La fiabilité est en revanche rarement mise en avant, alors qu'elle représente pourtant un intérêt bien réel. Parfois, on est amené à appliquer des patterns qui nous poussent à écrire du code redondant, sans grand intérêt (ex : l'implémentation de INotifyPropertyChanged). De fait, pendant l'écriture de ce code, le développeur a souvent tendance à être moins concentré, et c'est là qu'il risque de faire des erreurs d'étourderie. L'avantage du méta-code c'est qu'une fois qu'il est validé, le code généré sera toujours bon, sans risque d'erreur.

Par ailleurs, la génération de code se base généralement sur un modèle dont la définition est plus contraignante que le code. Cela peut donc potentiellement éviter les excentricités de certains développeurs. Ainsi, il sera plus simple pour n'importe quel développeur de l'équipe de faire évoluer le code généré à partir du modèle d'un autre développeur.

Si certains développeurs apprécient d'effectuer des tâches redondantes, car, ce faisant, ils sont dans leur zone de confort, pour beaucoup d'autres, dont je fais partie, faire régulièrement la même tâche est quelque chose de particulièrement ennuyeux, qui plus est quand cette tâche n'est pas particulièrement intéressante. Aussi, l'utilisation de la méta-programmation pour générer ces tâches améliore l'intérêt de notre travail. Ceci d'autant plus qu'écrire le méta-code est une tâche souvent très intéressante.

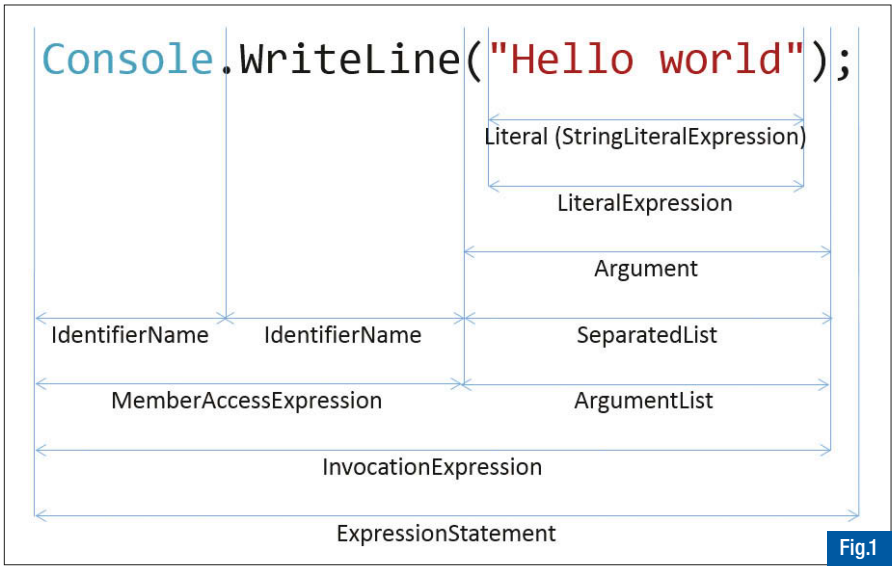


Fig.1

Pourquoi la méta-programmation ne s'est-elle jamais imposée ?

Au vu des avantages évoqués précédemment, on peut s'étonner que son utilisation soit si marginale et je vais tâcher de vous faire partager mon analyse.

Dans Visual Studio, pendant très longtemps, la génération de code se résumait essentiellement (mais pas uniquement) aux DSL et à T4. Dans un cas comme dans l'autre, ce n'était pas simple.

Visual Studio n'inclut pas d'éditeur de code pour T4. Si j'ai appris à coder sur ma calculatrice puis Emacs, le confort apporté par Visual Studio est tel qu'il est souvent difficile de revenir au mode bloc note pour éditer nos T4; cette difficulté se transformera en « no way » pour nombre de développeurs.

A noter, cependant, qu'il existe des providers tiers qui peuvent venir enrichir Visual Studio pour offrir un IDE de T4. Mais l'outillage reste, dans tous les cas, très inférieur à celui dont on dispose quand on édite un fichier CSharp dans Visual Studio.

Plusieurs personnes se sont essayées à l'approche DSL (Domain Specific Language) généralement accompagné d'un designer.

Il existe cependant, d'après-moi, deux problèmes à cette approche :

- Les possibilités d'expression du modèle sont souvent limitées,
- Si l'approche designer est souvent très utile pour convaincre les DSI, il est souvent beaucoup plus compliqué de convaincre les développeurs qui ont du mal à voir leur valeur avec cette approche, et qui n'ont aucune envie de « dessiner des rectangles avec des flèches » au lieu de coder. Enfin, la productivité avec un designer est souvent relativement décevante dûe à l'utilisation de la souris.

Il existe une autre raison beaucoup plus psychologique : la crainte de l'inconnu et le mot méta qui fait tout de suite penser à quelque chose de très compliqué.

Je ne suis pas en train de vous dire que la méta-programmation est quelque chose de basique à la portée de n'importe quel développeur débutant. Cependant, il ne faut pas exagérer non plus sa complexité, comme le font de nombreux développeurs.

Pourquoi Roslyn va tout changer ?

Dans le titre de cet article, j'évoque le fait que la méta-programmation va révolutionner notre

façon de coder. Le mot « révolutionner » n'est pas choisi au hasard. Je pense en effet que Roslyn va tout changer.

Les rares démonstrations faites par Microsoft à ce jour nous montrent comment améliorer Visual Studio relativement facilement en rajoutant des fonctionnalités de Refactoring et de Code review.

La plupart des développeurs qui ont eu la curiosité de faire de la veille techno sur Roslyn se sont donc arrêté à cela, y compris parmi les experts VS / .NET. Et la plupart se sont dit qu'ils n'avaient pas vocation à concurrencer Resharper, et que Roslyn ne leur servirait pas à grand-chose.

Cependant, ce n'est que la face émergée de l'iceberg et, vous vous en doutez, si je parle de révolution c'est que l'usage que l'on peut en faire va bien au-delà.

Que peut-on faire avec Roslyn ?

Roslyn nous permet d'accéder à la syntaxe de notre code, et à sa sémantique. Prenons un exemple :

```
Console.WriteLine(«Hello world»);
```

L'arbre syntaxique de ce code est le suivant : Fig.1

Comme vous pouvez le constater, l'arbre d'expression est très détaillé. Roslyn nous met également à disposition des visitors pour analyser cet arbre syntaxique et également pour pouvoir modifier le code (en mémoire).

Roslyn nous donne également accès à la sémantique. C'est-à-dire que là où l'analyse syntaxique nous informe que le code appelle une méthode WriteLine sur un membre Console, la sémantique nous permet de savoir qu'il s'agit d'une méthode statique définie dans la classe System.Console dont nous pourrons extraire de nombreuses informations telles que celles accessibles par Reflection.

Roslyn va aussi nous permettre de modifier notre solution en mémoire et de la compiler.

Roslyn nous donne également accès à des API de Visual Studio tels que le renommage global de la solution.

En quoi ces features augurent-elles d'une révolution ?

Aujourd'hui, pour faire de l'AOP (Aspect Oriented Programming), on pouvait utiliser des technologies comme Mono.Cecil. Cependant, cela s'avérait relativement compliqué de manipuler l'IL. Le fait de pouvoir, en mémoire, modifier la solution pour ensuite la compiler simplifie considérablement cela.

De plus, dans la même logique, on peut relativement facilement enrichir le langage C# à sa convenance.

Cependant, si faire cela est quelque chose de très intéressant, je pense que ce n'est pas vraiment souhaitable dans un cadre professionnel. En effet, imaginez que chaque entreprise utilise son propre compilateur C#. Cela risque de vite poser de gros problèmes pour des raisons évidentes de compétences des développeurs, sans même parler du fait que C# continuera de progresser indépendamment des compilateurs customs (même si, pour le coup, l'usage de Roslyn devrait grandement faciliter l'intégration des nouveautés).

Enfin, il ne faut pas oublier que si le code compilé n'est pas le code original, on perd les symboles des fichiers pdb, ce qui ne nous permet plus de débuguer, sauf à recréer les symboles associés à notre code source (comme c'est le cas pour le yield return ou le await par exemple).

A titre personnel (tous les experts de la méta-programmation ne sont pas d'accord sur le sujet), je pense que la vraie révolution réside dans l'utilisation de code comme un modèle pour notre génération.

C'est typiquement ce que fait WAQS (http://waqs.codeplex.com).

Prenons un exemple très simple avec des Order et des OrderDetails de la base Northwind.

Imaginons une application 3-Tiers dans laquelle on a besoin de calculer le total d'une commande.

Avec du code classique tel que nous le connaissons, nous allons définir une propriété calculée Total sur la classe Order, côté client et côté serveur (on pourrait partager le même code mais ce n'est pas forcément pertinent). Côté client, nous allons reprendre la logique de la méthode pour définir la logique des PropertyChanged / CollectionChanged afin que la modification d'une quantité provoque le recalcul automatique du Total. Si nous voulons effectuer une requête LINQ To Entities, nous allons devoir répéter la logique de la propriété Total, car Entity Framework ne connaissant pas cette propriété, il n'est pas possible de l'utiliser.

Résultat, nous avons dû dupliquer la logique de la propriété Total à de nombreux endroits dans le code, ce qui rendra plus compliquées la maintenabilité et l'évolutivité de notre code.

Avec WAQS, on va définir une méthode d'extension GetTotal prenant en paramètre un Order. Cette méthode ne sera jamais utilisée. Elle va juste servir de modèle à la génération des propriétés calculées sur le serveur et le client, ainsi que pour la génération des PropertyChanged. En effet, en analysant le code de la méthode GetTotal, on est capable de déterminer ses dépendances. Enfin, LINQ To WAQS et L2EW (LINQ To Entities + WAQS) supportent l'utilisation de la propriété Total. En effet, le code généré est capable d'extraire l'expression LINQ de notre méthode GetTotal, et de la combiner avec l'expression de la requête courante.

Afin d'éviter le problème des fichiers pdb, WAQS génère le code (via des T4) dans la solution. Ce qui fait que le code généré est bien celui compilé.

Utiliser des T4 a un autre intérêt. Imaginons que l'on veuille changer la formule de calcul du total d'une commande. Dans le cas normal, il va falloir repasser partout. Dans le cas de WAQS, il suffira de changer la méthode GetTotal puis de régénérer les T4, ce qui se fait très facilement depuis Visual Studio.

Avec cette approche, la méta-programmation apporte un niveau d'abstraction supplémentaire. Cela nous permet en effet de s'abstraire de la technique, pour se concentrer sur ce qui a de la valeur pour l'utilisateur final (les écrans et les règles métiers), tout en améliorant la qualité de notre développement à travers différents axes tels que :

- la productivité,
- la maintenabilité / évolutivité,
- l'intérêt pour le développeur (écrire des property changed, ce n'est pas passionnant...).

Dans le cadre de cet article, nous avons démontré l'intérêt d'une approche basée sur la méta-programmation et l'apport énorme de Roslyn dans celle-ci.

Dans le prochain numéro, nous reviendrons sur les API de Roslyn.

Matthieu **MEZIL**
Consultant
Infinite Square

INFINITE SQUARE

Raspberry : utiliser la carte PiFace


Fig.1

L'arrivée du Raspberry Pi, il y a deux ans, a ouvert un champ d'exploration pour l'initiation à l'informatique de la nouvelle génération. Alors que depuis des années la formation initiale se cantonne à l'enseignement de la bureautique, pompeusement baptisée «informatique» le Raspberry Pi offre aux enseignants un outil accessible et ludique. En particulier, la disponibilité en standard de Scratch permet aux élèves de développer des programmes sous forme graphique en s'appropriant les bases de l'algorithmique, tout en les aidant à créer et à travailler ensemble. Dans cet article vous apprendrez comment utiliser Scratch pour commander la carte d'interface numérique PiFace et agir sur le monde réel.

Prérequis : Savoir écrire un programme en Scratch

PRÉSENTATION DE LA CARTE PIFACE

L'idée de la carte PiFace a été lancée par Andrew Robinson, un chercheur de l'université de Manchester, spécialisé dans les processeurs embarqués et passionné par l'éducation. La carte a été conçue pour être facilement utilisable, initier les enfants à l'électronique et leur permettre d'interagir rapidement avec le monde réel. La PiFace se connecte directement sur le connecteur GPIO du Raspberry Pi. Les entrées et sorties se raccordent sur des borniers situés le long des côtés de la carte. La carte est équipée d'un MCP23S17 connecté sur le bus SPI du GPIO, et offrant 16 entrées/sorties numériques. Les 16 I/O se répartissent en 8 entrées (dont 4 équipées de boutons poussoirs) et 8 sorties.

Sorties de la carte PiFace

Les sorties (bornes jaunes) sont bufferisées par un ULN2803 et munies chacune d'une LED de contrôle. Chaque sortie est un Darlington en collecteur-ouvert qui supporte 50 V et peut commander une charge jusque 500 mA. Les sorties 0 et 1 sont équipées chacune d'un relai dont les contacts (NO/NF) sont accessibles sur un bornier à vis (noir).

Des cavaliers présents sur la carte permettent un certain nombre de réglages : adresse de la carte, alimentation externe, diodes anti-retour, désactivation des relais, désactivation des sorties.

Entrées de la carte PiFace

Les entrées de la carte PiFace sont maintenues à l'état haut (1) par une résistance de tirage activée par défaut. Les entrées sont mises à 0 (activées) par un bouton poussoir présent sur la carte (entrées 0 à 3) ou par un dispositif extérieur mettant l'entrée à la masse (entrées 0 à 7).

Remarque : *Les entrées et sorties sont numérotées de 1 à 8 sur la carte et non pas de 0 à 7 comme il est d'usage en informatique. Sur l'émulateur graphique, par contre les boutons de commande des sorties sont numérotés... de 0 à 7.*

INSTALLATION ET TEST DE LA CARTE

Installation et mise à jour du système Raspbian

Installez une version «propre» de Raspbian sur une carte SD de 4 Go minimum (*2014-01-07-wheezy-raspbian.zip* au moment de l'écriture de l'article). Mettez en place la carte PiFace sur le Raspberry Pi, en veillant à ce qu'il ne soit pas alimenté (la prise d'alimentation micro-USB ne soit pas connectée). Insérez la carte SD dans son support et branchez la prise d'alimentation. Le système démarre et affiche l'outil de configuration *raspi-config*. Dans le choix 4 Internationalisation Options, réglez les locales sur fr_FR.UTF8 UTF8 puis la zone sur Europe-Paris (si vous résidez en France), enfin configurez le clavier en AZERTY et redémarrez le Raspberry Pi.

Mettez le système à jour :

```
sudo apt-get update
sudo apt-get upgrade
```

Test de la PiFace en Python

Les versions récentes de Raspbian intègrent les outils de gestion de la carte PiFace par défaut. Les programmes de test en Python sont donc dis-

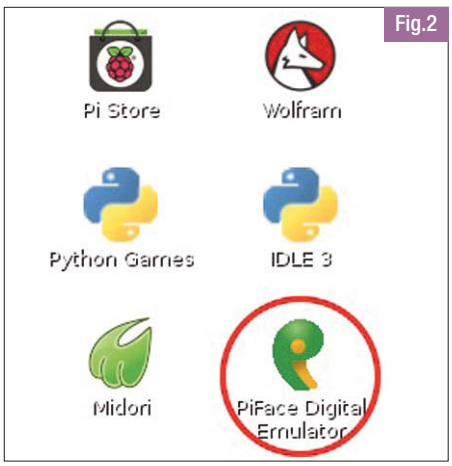


Fig.2

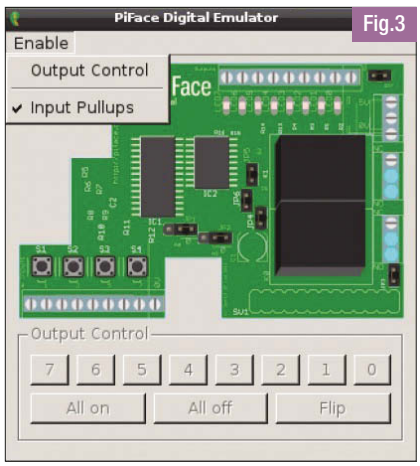


Fig.3

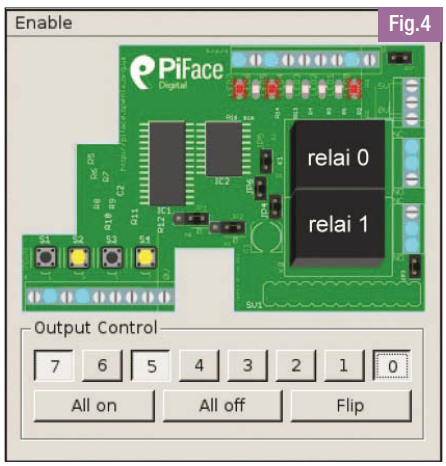


Fig.4

ponibles et vous pouvez lancer le programme *blink.py* qui fera clignoter la LED 7 une fois par seconde et permettra de vérifier que tout est en ordre :

```
pi@raspberrypi ~ $ python3 /usr/share/doc/python3-pifacedigitalio
/examples/blink.py
```

Le clignotement d'une LED est souvent utilisé en tant que «Hello World» de l'embarqué... La lecture des programmes présents dans le dossier *examples* constitue une bonne introduction au pilotage de la PiFace en Python.

Installation de l'émulateur graphique

Maintenant que nous avons vérifié que tout est en ordre, passons à l'installation de l'émulateur graphique.

```
pi@raspberrypi ~ $ sudo apt-get install python3-pifacedigital
-emulator
```

Cette installation va occuper environ 108 Mo supplémentaires sur la carte SD. Apres la fin de l'installation, redémarrez le Raspberry Pi :

```
pi@raspberrypi ~ $ sudo reboot
```

L'icône de l'émulateur PiFace est maintenant présente sur le bureau de LXDE. Double cliquez sur l'icône nouvellement installée pour l'interface graphique (Fig.2).

L'interface graphique représente la carte PiFace vue de dessus. Par défaut, les sorties sont dévalidées. Pour les mettre en service, cliquez sur **Enable** puis sur **Output Control** pour les mettre en service. En bas de la fenêtre, la zone **Output Control** qui était grisée et non accessible avant la validation des sorties, devient opérationnelle. Les *boutons* numérotés permettent d'activer/désactiver chaque sortie séparément. Vous pouvez vérifier le bon fonctionnement en observant l'allumage/extinction des LED, et pour les sorties 0 et 1, vous entendrez simultanément le bruit des relais.

Le bouton **All on** active toutes les sorties, **All off** désactive toutes les sorties et **Flip** inverse l'état de chaque sortie.

Testez les différentes possibilités en contrôlant que les informations fournies par l'émulateur coïncident avec ce que vous observez sur la carte.

Sur la Fig.4, les poussoirs S2 et S4 sont appuyés, les entrées correspondantes sont indiquées par un cercle coloré qui remplace le dessin de la vis. Les boutons appuyés apparaissent en jaune, et les LED allumées brillent en rouge.

Le relai 0 (situé sous la rangée de LED) est collé. Le contact est établi entre les deux bornes supérieures. Le relai 1 est au repos, le contact est établi entre les deux bornes basses de son bornier.

CONFIGURER SCRATCH POUR PILOTER LA CARTE PIFACE

Scratch est un environnement de développement graphique et un langage de programmation utilisant des blocs prédéfinis. Les blocs ont des formes qui ne permettent d'encastrer que les blocs autorisés, éliminant de fait les erreurs de syntaxe.

D'origine, Scratch anime des lutins (sprites) sur une scène. Cependant, dans le but d'autoriser le jeu multi-joueurs, une méthode nommée Mesh permet de partager entre plusieurs projets Scratch des variables et des messages diffusés en broadcast, même si ces projets sont hébergés sur des ordinateurs différents. Mesh envoie un message à tous les programmes connectés, chaque fois qu'un broadcast est envoyé dans Scratch, ou qu'une variable globale est modifiée. Dans notre cas, le serveur Mesh sera hébergé sur l'ordinateur local, et c'est sur le port 42001 que viendra se connecter le programme de gestion de la carte PiFace (Fig.5).

Mise en route de Mesh dans Scratch

Par défaut Mesh n'est pas activé dans Scratch. Normalement, il faut cliquer sur **Partage** dans la barre de menu, en appuyant sur Shift, pour accéder au menu permettant d'activer Scratch... Enfin ça c'est la théorie, parce que Mesh n'est pas accessible par défaut. Je vous propose donc de voir la procédure pour activer Mesh. Respectez bien toutes les étapes pour obtenir le résultat attendu. Si à l'issue de toutes les manipulations Mesh n'est pas activé, soyez persévérant, et recommencez en suivant scrupuleusement chaque étape (Fig.6).

1 – Ne démarrez pas Scratch en cliquant sur l'icône du bureau

Démarrez Scratch en super-utilisateur.

Ouvrez un terminal LXterminal et tapez en ligne de commande :

```
pi@raspberrypi ~ $ sudo /usr/bin/scratch
```

L'ouverture de Scratch avec *sudo* permettra après l'activation de Mesh, de sauvegarder les modifications. Si vous démarrez avec l'icône du bureau, les modifications ne seront pas enregistrées et il faudra recommencer la procédure.

2 – Accédez au menu caché de Scratch

L'accès au (premier) menu caché de Scratch se fait en maintenant la touche Shift appuyée et en cliquant dans le rond formé par la lettre R de Scratch en haut à gauche de l'écran (Fig.7). Dans le menu qui s'ouvre, cliquez sur **turn fill screen off**. Ceci a pour effet de réduire la taille de la zone occupée par l'interface graphique Scratch dans la fenêtre qu'il occupe. L'accès au second menu caché est maintenant possible (Fig.8).

3 – Accédez au second menu caché de Scratch

Déplacez la souris au bas de l'interface graphique de Scratch. L'action précédente a libéré un peu de place dans le bas de la fenêtre (Fig.9).

Cliquez dans la zone blanche libérée sous l'interface graphique. Un menu s'ouvre : cliquez sur l'item **open** puis dans le menu suivant cliquez sur **browser**. Ceci vous donne accès au navigateur d'objets de Scratch qui va (enfin) vous permettre d'activer Mesh.

4 – Modifier addServerCommandTo

Après ouverture du navigateur d'objets, Cliquez successivement sur **Scratch-UI-Panes** => **ScratchFrameMorph** => **menu/button actions** => **addServerCommanTo:** Une zone texte apparait dans la partie basse du navigateur. Remplacez le texte «**t2 true**» par le texte «**t2 false**».

Fig.5

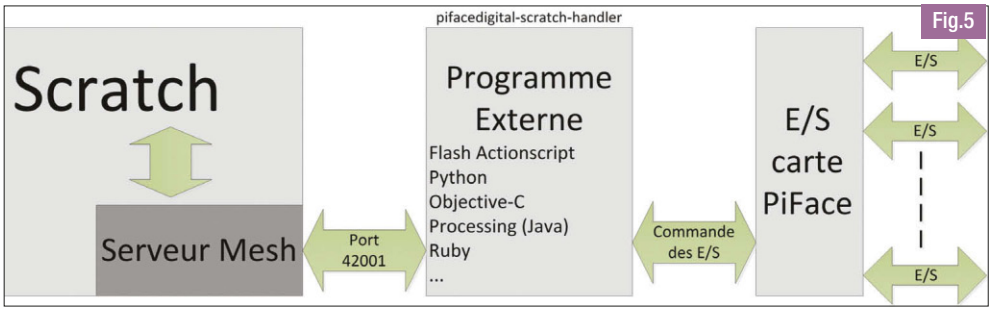


Fig.7

La modification est faite, il faut maintenant la sauvegarder. Juste à gauche de **addServerCommandTo:** et au-dessus de la flèche de défilement, figure un rectangle contenant un signe **-**. Cliquez sur ce rectangle pour ouvrir le menu permettant de valider la modification que vous avez effectuée.
Cliquez sur **accept(s)** pour valider les modifications que vous avez apportées à **addServerCommandTo:**. Le menu se ferme et vous pouvez quitter le browser d'objets en cliquant sur la croix située à gauche de la barre de titre **System Browser**.

5 – Enregistrer les modifications

Pour enregistrer les modifications que vous venez d'apporter à Scratch, il faut ouvrir à nouveau le premier menu caché, en maintenant la touche SHIFT appuyée et en cliquant dans le rond de la lettre R. Dans un premier temps cliquez sur **turn fill screen on** pour que l'éditeur graphique remplisse à nouveau la fenêtre. Ensuite cliquez sur **save image for end user** pour enregistrer les modifications. Répondez **Yes** pour confirmer que vous voulez sauver vos modifications. A l'issue de la sauvegarde, la fenêtre de Scratch se ferme, et vous revenez à LXTerminal. On ne peut pas dire que la mise en service de Mesh soit facilement accessible, mais si vous suivez l'ordre des opérations minutieusement, le serveur devrait être opérationnel sur le port 42001.

Test du fonctionnement de Mesh

Pour vérifier le bon fonctionnement de Mesh, ouvrez cette fois Scratch en cliquant sur l'icône du bureau. Maintenez la touche SHIFT appuyée et cliquez sur le menu **Partage**. Si Mesh fonctionne, deux lignes supplémentaires apparaissent en bas du menu (Fig.6).

Fig.6

Fig.13

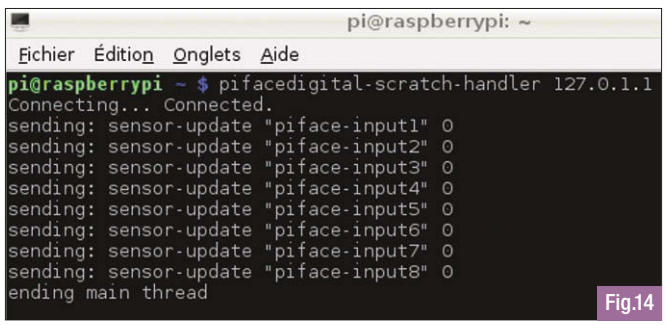
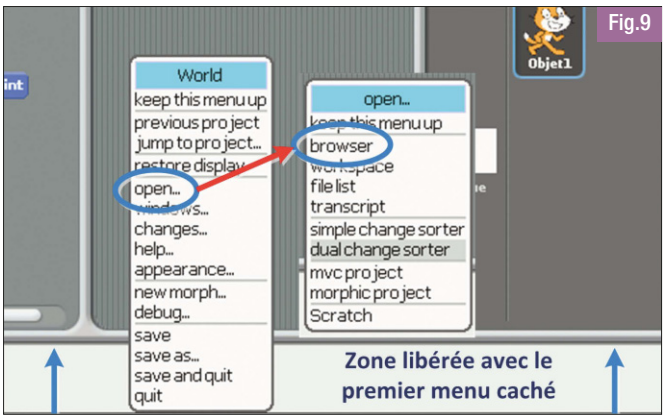


Fig.14

Fig.9



Relevé de l'adresse du serveur Mesh

Cliquez sur Show IP Address, une fenêtre s'ouvre et vous indique l'adresse du serveur Mesh (Fig.13). Notez cette adresse, c'est celle qui permettra au programme *pifacedigital-scratch-handler* de dialoguer avec Mesh. Vous pouvez choisir d'arrêter Mesh en cliquant sur **Stop Hosting Mesh**. La remise en route se fait dans le même menu, en cliquant sur **Host Mesh**, ce qui affiche automatiquement la fenêtre indiquant l'adresse IP du serveur Mesh.

Lancement du programme interface entre Scratch et la carte PiFace

Laissez Scratch ouvert, ou réduisez la fenêtre, mais Scratch doit fonctionner pour que le programme puisse se connecter au serveur Mesh. Ouvrez un terminal LXTerminal et saisissez :

```
pi@raspberrypi ~ $ pifacedigital-scratch-handler 127.0.1.1
```

L'adresse IP sera bien entendu celle que vous aurez relevée à l'étape précédente, sur votre propre Raspberry Pi. Le programme doit démarrer et se connecter au serveur Mesh (Fig.14).
Le programme initialise également la valeur des variables correspondant aux 8 entrées numériques de la carte PiFace.
Attention : la configuration de Scratch est enregistrée et Mesh sera lancé à chaque démarrage de Scratch. Par contre, il faudra répéter le lancement de *pifacedigital-scratch-handler* chaque fois que vous démarrerez Scratch pour l'utiliser avec la carte PiFace. Vous pouvez cependant automatiser le lancement en écrivant un script shell.

Commande d'une LED à partir de Scratch

Dans Scratch, Cliquez sur le bouton **Variables** puis sur **Nouvelle variable**.

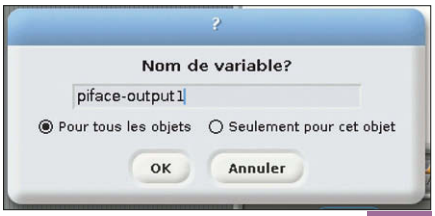
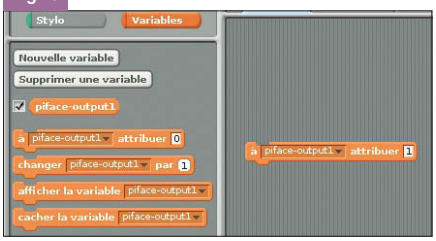


Fig.15

Fig.16



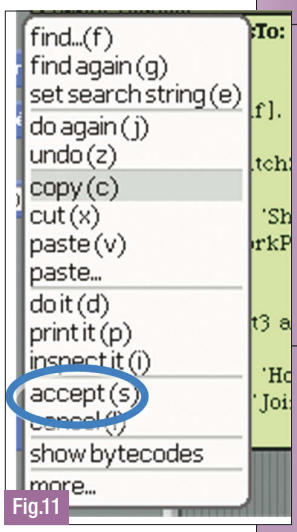
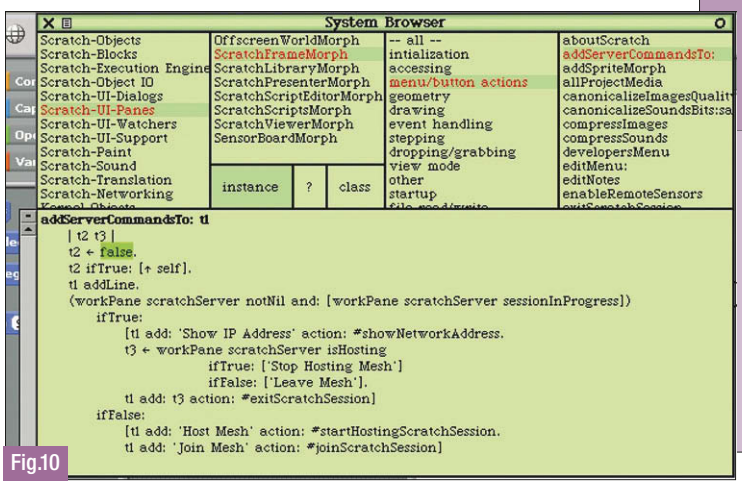


Fig.11

Fig.10



Dans la fenêtre qui s'ouvre saisissez le nom de la variable : **piface-output1**. Respectez absolument la façon de nommer la variable, c'est ce qui permet au programme *pifacedigital-scratch-handler* de la reconnaître ! Laissez coché le bouton radio **Pour tous les objets**, et cliquez sur **OK** pour enregistrer votre variable (Fig.15).

Vous disposez maintenant d'une variable qui est liée avec la sortie 1 de la carte PiFace. Déplacez le bloc **à piface-output1, attribuer 0** dans la zone des scripts (zone centrale de l'interface graphique). Modifiez la valeur de la variable dans le bloc que vous venez de déplacer, en remplaçant le 0 par un 1. Cliquez sur le bloc pour appliquer la modification. La sortie 0 de la carte PiFace est activée : la LED s'allume et le relai correspondant colle. Si vous remplacez le 1 par un zéro et cliquez à nouveau sur le bloc, la sortie repasse à 0 et le relai retombe (Fig.16).

Lecture de l'état d'une entrée de la carte PiFace

Dans Scratch, cliquez sur le bouton **Capteurs**. Cliquez sur le bloc **valeur du capteur potentiomètre** et déplacez-le dans la zone des scripts. Ouvrez la liste déroulante en cliquant sur la flèche située à droite de **potentiomètre** (Fig.17)

Choisissez l'entrée **piface-input1**. Pour utiliser la valeur de cette entrée, renvoyée par le bloc **valeur du capteur**, il va falloir écrire un court programme (Fig.18).

Répéter
 Si le bouton est appuyé
  Le chat dit : Appuyé !
 Sinon
  Le chat dit : Relaché !

En appuyant sur le bouton 1 de la carte PiFace, puis en le relâchant, le chat annonce le résultat en permanence.

PROGRAMME EN SCRATCH

Tous les outils nécessaires à la réalisation d'un programme Scratch utilisant la carte PiFace sont maintenant disponibles :

- Installation et test de la carte PiFace
- Installation de l'interface avec Scratch
- Communication entre Scratch et les sorties
- Communication entre Scratch et les entrées

Organisation du programme

Annoncer le mode d'emploi du programme
 Si le bouton 4 n'est pas actionné
  Si le bouton 1 est actionné
   Annoncer que le chenillard fonctionne
   Activer le chenillard (10 balayages des diodes 1 à 4)
  Si le bouton 2 est actionné
   Annoncer que le clignotant fonctionne
   Activer le clignotant (10 clignotements des diodes 1 à 4)
 Sinon
  Dire au revoir

La conversion de cet algorithme en Scratch ne pose pas de problème particulier, les blocs du programme correspondent aux blocs de l'algorithme. A partir de cet exemple, vous pouvez maintenant développer vos propres idées : jeu de dé, commande d'un moteur avec Scratch....

Sources :

http://wiki.scratch.mit.edu/wiki/Mesh
http://mchobby.be/wiki/index.php?title=PiFace-Scratch-Demarrer
http://www.piface.org.uk/guides/Install_PiFace_Software/Installing_PiFace_Digital_modules/
http://pi.cs.man.ac.uk/download/EnablingMeshInScratch.pdf

François **Mocq**
Passionné par l'électronique, il est l'auteur de "Raspberry Pi" aux éditions ENI


Fig.19

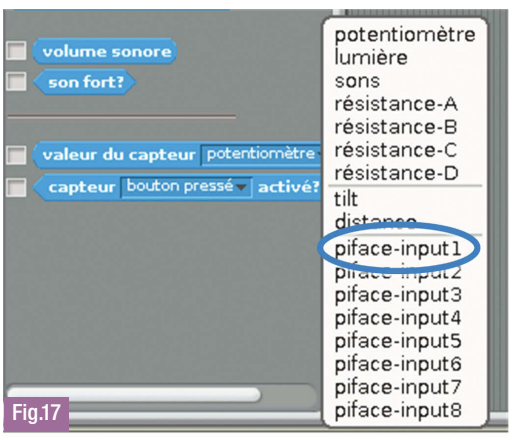


Fig.17

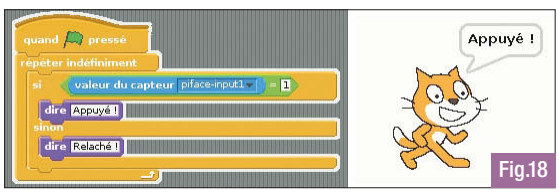


Fig.18

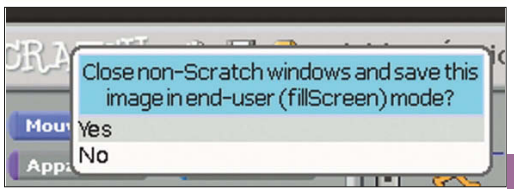
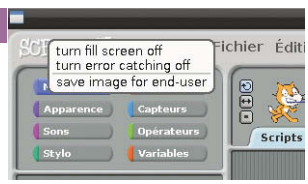


Fig.12

Fig.8



Java 8 : La révolution des Lambdas expressions

2e partie

La première partie de cet article aura permis de découvrir en profondeur les Lambdas expressions ainsi que les principales nouveautés associées apparues dans Java 8. Clairement, les Lambdas constituent un tournant dans l'histoire de Java qui propose désormais la programmation fonctionnelle à ses développeurs. Dans cette seconde partie, nous allons mettre en application de manière plus poussée les Lambdas au travers d'exemples mettant en lumière une autre grande nouveauté de Java 8 : les Streams.

L'introduction des Lambdas expressions dans Java 8 aura mis au grand jour le côté vieillissant de l'API Collections du JDK. En effet, cette API, au coeur des applications Java, date de plus de 10 ans maintenant, et il aurait été fort judicieux de lui créer un remplaçant de type «Collections II». Outre le fait que cet effort de remplacement aurait constitué une tâche considérable, décalant encore plus la sortie de Java 8, il aurait induit un gros travail pour les développeurs puisque les interfaces Collections ont un impact sur l'ensemble du JDK. De fait, une telle solution aurait eu l'effet indésirable de retarder son adoption par les développeurs Java.
Afin d'éviter une telle situation, les architectes du langage Java ont choisi une voie d'évolution toute en douceur en introduisant par exemple les méthodes virtuelles d'extension aux interfaces existantes. Ceci permettant d'étendre, sans souci de compatibilité, les interfaces Collection, List ou bien Iterable. Dans un second temps, cela facilite la création des Streams et leur ajout au JDK.

Streams

Définis au sein du package java.util.stream, les Streams constituent la grande nouveauté de Java 8 en terme d'abstraction. Un Stream représente une séquence de valeurs, et expose un ensemble d'opérations d'agrégations autorisant le développeur à exprimer des manipulations classiques sur ces valeurs de manière simple et claire. Au sommet de la hiérarchie des Streams, on retrouve l'interface Stream<T>, représentant un stream de références d'objets. Plusieurs objets spécialisent cette interface comme IntStream qui décrit un stream de primitives entières. Autre point important, les bibliothèques existantes permettent d'obtenir facilement des streams à partir de collections, de tableaux ou d'autres sources de données.
Les opérations proposées par les streams sont chaînées au sein de pipelines. Ainsi, il est possible de changer la couleur des objets Shape seulement s'ils sont bleus :

```
shapes.stream().filter(s -> s.getColor() == BLUE).forEach
(s -> s.setColor(RED);
```

La méthode stream de l'interface Collection produit une vue Stream des éléments de la collection. La méthode filter produit un stream filtré ne gardant que les objets Shape bleus, et, enfin, on utilise une itération interne via forEach pour appliquer la couleur rouge sur ces Shapes.
Les opérations proposées par les Streams sont soit de type intermédiaire, soit de type terminal avec les nuances suivantes :

- Une opération intermédiaire garde le stream ouvert et permet l'application des transformations suivantes. Les méthodes filter et map de cet exemple sont des opérations intermédiaires. Ainsi, elles retournent des objets Stream autorisant le chaînage d'opérations.
- Une opération terminale doit être l'opération finale invoquée sur un stream. Une fois cette opération appliquée, le stream est considéré comme consommé et n'est donc plus utilisable par la suite. Une opération terminale renvoie un type différent de Stream donc.

En reprenant l'exemple précédent, on utilise l'opération terminale sum de l'interface Stream afin de compter le nombre de Shapes de couleur bleue :

```
int count = shapes.stream().filter(s -> s.getColor() == BLUE)
.count();
```

Le schéma classique d'utilisation des Streams est ainsi limpide. Il est constitué de 3 étapes :
1 - Obtention d'un stream à partir d'une source de données
2 - Application d'une ou plusieurs opérations intermédiaires
3 - Application d'une opération terminale
La puissance de l'implémentation des Streams dans Java 8 réside avant tout dans la différence de comportement entre opérations intermédiaires et terminales. En effet, les opérations intermédiaires sont implémentées de manière lazy et peuvent donc être chaînées sans souci de performance puisque c'est uniquement lors de l'application d'une opération terminale que les opérations sur le stream seront effectivement réalisées. Ces dernières étant par nature plus coûteuses en temps d'exécution. Sans détailler de manière exhaustive les opérations de l'interface Stream, on peut citer les opérations intermédiaires map, filter, distinct, ou encore limit, qui, combinées aux opérations terminales forEach, toArray, collect ou reduce, mettent à la disposition du développeur Java tout ce qu'il faut pour réaliser des transformations de type filtrage / accumulation, filtrage / tri / parcours, etc ...
Enfin, on peut signaler que l'API propose des méthodes statiques dans l'interface Stream (ce qui constitue une nouveauté avec Java 8) permettant d'obtenir des streams infinis. De fait, on peut obtenir un stream infini d'entiers contenant la suite des puissances de 2 comme suit :

```
Stream.iterate(1, x -> x*2);
```

Le premier argument contient ici la valeur initiale du stream, et le second, la fonction permettant de passer de l'élément n à l'élément n + 1 dans le stream.

Streams en pratique

Afin de mettre en pratique les Streams de manière plus poussée, considérons un POJO Person :

```
public class Person {
  public enum Sex {
    MALE, FEMALE
  }
```

```
String name;
LocalDate birthday;
Sex gender;
String emailAddress;

Person(String n, LocalDate birth, Sex g, String mail) {
  name = n;
  birthday = birth;
  gender = g;
  emailAddress = mail;
}

public int getAge() {
  return birthday
      .until(IsoChronology.INSTANCE.dateNow())
      .getYears();
}

public void printPerson() {
  System.out.println(name + «, « + this.getAge());
}

public static int compareByAge(Person a, Person b) {
  return a.birthday.compareTo(b.birthday);
}

public static List<Person> createRoster() {

  List<Person> roster = new ArrayList<>();
  roster.add(
      new Person(
          «Sylvain»,
          IsoChronology.INSTANCE.date(1984, 5, 13),
          Person.Sex.MALE,
          «sylvain@mail.com»));
  roster.add(
      new Person(
          «Rachida»,
          IsoChronology.INSTANCE.date(1987, 4, 9),
          Person.Sex.FEMALE, «rachida@mail.com»));
  roster.add(
      new Person(
          «Adam»,
          IsoChronology.INSTANCE.date(2013, 2, 9),
          Person.Sex.MALE, «adam@mail.com»));
  roster.add(
      new Person(
          «Nathan»,
          IsoChronology.INSTANCE.date(2014, 2, 12),
          Person.Sex.MALE, «nathan@mail.com»));

  return roster;
 }
}
```

A partir de cet objet, nous allons créer plusieurs instances de Person grâce à la méthode statique createRoster. Dans un premier temps, nous allons simplement afficher les personnes de la liste à l'aide de la méthode forEach :

```
List<Person> roster = Person.createRoster();
roster.stream().forEach(p -> p.printPerson());
```

On remarque ici l'utilisation d'une Lambda expression en entrée de la méthode forEach. Cette Lambda étant une instance de l'interface fonctionnelle Consumer sur laquelle nous reviendrons plus tard. Avec cette liste de personnes, nous pouvons calculer l'âge moyen des personnes de type masculin :

```
double average = roster.stream()
      .filter(p -> p.getGender() == Person.Sex.MALE)
      .mapToInt(Person::getAge)
      .average()
      .getAsDouble();
```

Une fois le stream de la liste récupéré, on applique un filtrage sur les personnes de type MALE, avant d'obtenir le stream correspondant aux âges des personnes filtrées grâce à l'utilisation de l'opération intermédiaire mapToInt avec, en entrée, la référence de méthode Person::getAge. Reste ensuite à calculer la moyenne du stream d'entiers en appelant l'opération terminale average. L'opération mapToInt peut également être mise à profit pour calculer la somme des âges des personnes de la liste :

```
Integer totalAge = roster.stream()
      .mapToInt(Person::getAge)
      .sum();
```

Il est également possible de réaliser la même somme en employant les opérations de map et de reduce des Streams :

```
Integer totalAgeReduce = roster.stream()
      .map(Person::getAge)
      .reduce(0, (a, b) -> a + b);
```

L'opération reduce est utilisée ici avec un élément identité valorisé à 0. Il constitue la valeur initiale de l'accumulateur, et la valeur par défaut qui sera renvoyée si le stream sur lequel le reduce est appliqué est vide. En second paramètre, reduce prend une Lambda expression instance de l'interface fonctionnelle BinaryOperator<T> qui prend 2 éléments de type T en entrée et renvoie un élément de type T en sortie.

Streams parallèles

Un des moteurs de l'apparition des Lambdas expressions et des Streams dans Java 8 est la volonté d'Oracle de rendre le parallélisme plus accessible aux développeurs Java. Plus personne n'a de doute quant au fait que les applications de demain devront tirer profit au maximum de la parallélisation des traitements. Néanmoins, et bien que la plateforme fournisse un support important pour favoriser les traitements concurrents, peu de développeurs franchissent le pas devant les obstacles présents pour migrer leur code séquentiel vers du code parallèle. De fait, les architectes du langage ont décidé de promouvoir des idiomes pouvant être à la fois séquentiels ou parallèles. Le but étant de décrire quelles opérations doivent être appliquées, et non comment elles doivent être exécutées. Le tout pour faciliter les traitements parallèles au maximum sans toutefois les rendre invisibles pour les développeurs.

En suivant cette volonté, les opérations intermédiaires des streams qui sont traitées au sein de pipelines peuvent être exécutées soit en série soit en parallèle. Le basculement entre les modes étant un simple choix à faire au niveau d'un stream. Par défaut, les implémentations au sein du JDK retournent toujours des streams séquentiels. Le passage d'un stream

séquentiel à un stream parallèle se faisant en invoquant la méthode parallel(). L'opération inverse nécessitant quant à elle le recours à la méthode sequential(). Ainsi, le parallélisme demeure explicite, mais il n'est pas intrusif puisqu'il ne nécessite pas d'adaptations particulières pour le développeur. En reprenant l'exemple du calcul de la moyenne des âges, on peut paralléliser le traitement de la sorte :

```
double average = roster.parallelStream()
      .filter(p -> p.getGender() == Person.Sex.MALE)
      .mapToInt(Person::getAge)
      .average()
      .getAsDouble();
```

Devant la facilité de parallélisation des opérations des Streams, le développeur ne devra cependant pas perdre de vue que les sources de données associées aux streams sont généralement mutables, et qu'il existe donc des possibilités d'interférence si la source est modifiée pendant l'exécution des traitements d'un stream. Le contenu d'une source de données associé à un stream, doit donc rester constant lors de l'exécution des opérations du stream. On notera que cette condition nécessaire au bon déroulement des traitements des Streams se rapproche de ce que l'on connaît déjà à l'heure actuelle avec les restrictions concernant la modification d'une Collection lorsque l'on est en train d'itérer dessus par ailleurs.

Streams vs Collections

Bien que présentant un certain nombre de similarités en surface, les Streams et les Collections poursuivent en réalité des buts différents. Les Collections sont principalement destinées à gérer efficacement des éléments tant d'un point de vue mémoire que d'un point de vue accès. A contrario, les Streams n'ont pas vocation à manipuler ou accéder directement à ces éléments, mais plutôt à décrire de manière déclarative les opérations à effectuer sur une source de données. De ce fait, les Streams possèdent un certain nombre de propriétés spécifiques :

- Un Stream ne stocke pas de données contrairement aux Collections. Il se contente de les transférer d'une source de données quelconque (Collection, tableau, channel I/O, ...) vers un pipeline d'opérations.
- Un Stream est fonctionnel par essence. Ainsi, les opérations appliquées à un stream produisent un résultat sans modifier la source de données associée. Si des données doivent être modifiées, un nouveau stream est alors produit et ainsi de suite. Ce point se révèle essentiel pour garder une cohérence lors de la parallélisation des traitements.
- Les opérations intermédiaires appliquées à un Stream sont de type lazy afin d'optimiser les performances. De fait, le pipeline d'opérations d'un stream est exécuté une seule fois lors de l'appel à une opération finale.
- Un Stream n'est pas obligatoirement borné, contrairement à une Collection. En effet, certains problèmes nécessitent d'être exprimés comme des streams infinis en laissant le soin au client de consommer les valeurs jusqu'à ce qu'il soit satisfait. Attention toutefois à bien veiller à ce que les opérations appliquées sur un stream infini se terminent en un temps fini via l'emploi de méthodes comme limit(n) ou findFirst.
- Un stream ne peut être utilisé qu'une seule fois. Une fois consommé, il faudra recréer un nouveau stream.

Définis au sein d'une API à part entière, les Streams sont totalement indépendants des Collections. Alors qu'il est aisé d'obtenir un stream à partir d'une Collection via les méthodes stream() et parallelStream(), et d'effectuer l'opération inverse en créant une Collection depuis un stream via l'appel à collect() comme nous le verrons plus tard, les Streams ne se limitent bien évidemment pas aux Collections.

Il est ainsi possible d'effectuer des opérations avec des Streams sur de nombreuses classes du JDK qui ont été étendues pour l'occasion telles que BufferedReader, Random ou BitSet. En outre, la méthode Arrays.stream() permet d'obtenir une vue Stream d'un tableau donné. En fait, tout ce qui peut être décrit à l'aide d'un Iterator peut être utilisé comme source de données pour les Streams.

Itération interne

Comme on a pu le constater, les Streams proposent une méthode forEach permettant d'itérer sur les éléments du stream. Alors que les Collections, et plus généralement tout ce qui implémente l'interface Iterable, ont été bties sur le concept d'itération externe, Java 8 met à disposition l'itération interne directement sur les Collections.

Pour ce faire, l'interface Iterable a été étendue grâce à une méthode virtuelle d'extension forEach. Au lieu de contrôler l'itération, le client délègue ainsi ce travail à la bibliothèque standard, et passe en entrée le code à exécuter durant l'itération. Ainsi, pour modifier la couleur d'une liste d'objets Shape, la méthode forEach est employée comme suit :

```
shapes.forEach(s -> s.setColor(RED));
```

Bien plus qu'un simple sucre syntaxique, ce changement est très significatif. En effet, le contrôle de l'itération a été déporté du code client vers l'API sous-jacente, autorisant cette dernière à optimiser le traitement que ce soit en termes de parallélisme ou bien de lazy. Au final, l'itération interne permet d'obtenir un code plus clair séparant le quoi (appliquer la couleur rouge) du comment (itérer sur les éléments).

Collectors

Une fois obtenus à partir d'une source de données, les Streams permettent d'appliquer des opérations de transformations. Il se peut alors qu'à la suite de ces opérations, on souhaite obtenir un objet List ou un Set représentant le résultat de ces transformations.

Pour adresser ce cas d'usage, les Streams proposent la méthode collect() dont l'argument est un Collector qui représente la manière utilisée pour la création de la structure qui accueillera les données.

La classe Collectors propose des factories, pour la plupart des collectors communs. A ce titre, on citera toList() et toSet() qui sont les plus classiques et sont utiles pour obtenir la liste des noms des hommes :

```
List<String> namesOfMaleMembersCollect = roster
      .stream()
      .filter(p -> p.getGender() == Person.Sex.MALE)
      .map(p -> p.getName())
      .collect(Collectors.toList());
```

La classe Collectors met également à disposition des méthodes pour créer des maps que ce soit via toMap() ou via groupingBy() qui, dans notre cas, s'avère intéressant pour grouper les personnes de notre liste par genre :

```
Map<Person.Sex, List<Person>> byGender = roster.stream()
        .collect(Collectors.groupingBy(Person::getGender));

List<Map.Entry<Person.Sex, List<Person>>> byGenderList = new
 ArrayList<>(byGender.entrySet());
byGenderList.stream()
      .forEach(e -> {
        System.out.println(«Genre :  « + e.getKey());
        e.getValue()
          .stream()
          .map(Person::getName)
          .forEach(f -> System.out.println(f));
       });
```

Maintenant, nous allons mettre à profit la puissance des Collectors pour obtenir le total des âges des personnes de notre liste en groupant les personnes par genre :

```
Map<Person.Sex, Integer> totalAgeByGender = roster.stream()
        .collect(
           Collectors.groupingBy(Person::getGender,
              Collectors.reducing(0, Person::getAge, Integer:
:sum)
        ));

List<Map.Entry<Person.Sex, Integer>> totalAgeByGenderList = new
 ArrayList<>(totalAgeByGender.entrySet());

totalAgeByGenderList.stream()
      .forEach(e ->
        System.out.println(«Genre: « + e.getKey() +
          «, Total Age : « + e.getValue()));
```

Ce bout ce code met en avant toute la puissance des Lambdas et des nouveautés associées qui débarquent dans Java 8.
On retrouve ainsi l'utilisation d'un stream pour y appliquer un collect basé sur la méthode statique groupingBy de Collector, prenant en entrée la référence de méthode Person::getGender afin de grouper les personnes par genre. Le second argument est un Collector créé à l'aide de la méthode utilitaire reducing qui permet d'additionner les âges des personnes groupées par genre. La suite du code affichant les éléments via une itération interne.
Plus encore que tout ce que nous avons pu passer en revue jusqu'à présent, les Collectors révèlent toute la puissance de la programmation fonctionnelle que nous allons pouvoir mettre en oeuvre avec Java 8. Attention toutefois, cette puissance demandera un temps d'apprentissage certain aux développeurs avant de pouvoir en tirer pleinement partie.

Interfaces fonctionnelles standards

Les possibilités de programmation fonctionnelle introduites avec Java 8 ont mis en exergue la nécessité d'avoir un certain nombre d'interfaces fonctionnelles prêtes à l'emploi dans le JDK en standard.
Ces interfaces ayant vocation à couvrir des besoins classiques du monde fonctionnel. Dans la grande majorité des cas, elles ont été ajoutées au sein du package java.util.function et sont organisées dans les catégories suivantes :

- Function<T, R> : prend en entrée un paramètre T et retourne un objet R
- Predicate<T> : prend en entrée un paramètre T et retourne un booléen en sortie
- Consumer<T> : prend en entrée un paramètre T, effectue un certain nombre de traitements et ne retourne rien
- Supplier<T> : aucun paramètre en entrée et retourne un objet T
- BiFunction<T, U, R> : prend 2 paramètres en entrée respectivement de type T et U, renvoie un objet R

Il est bon de souligner que nous avons abondamment eu recours à ces interfaces fonctionnelles standards lors des exemples précédents. Les exemples les plus simples viennent de l'opération de filtrage des Streams qui prend en entrée un Predicate, de l'opération map qui prend une Function, et de l'itération forEach qui prend un Consumer en entrée. Dans nos exemples de filtrage, la Lambda expression passée en entrée est une instance de Predicate. En sus, ces prédicats peuvent être combinés pour en créer des plus complexes et également facilement réutilisables. On peut ainsi effectuer le filtrage sur les personnes dont le nom se termine par la lettre 'n' et comporte au moins 4 lettres :

```
Predicate<Person> endsWithN = p -> p.name.endsWith(«n»);
Predicate<Person> atLeast4 = p -> p.name.length() >= 4;

roster.stream().filter(endsWithN.and(atLeast4))
        .forEach(p -> p.printPerson());
```

On utilise l'opérateur logique AND via la méthode éponyme, mais les opérateurs OR et XOR sont également supportés. Toujours à partir de notre exemple fil rouge, on peut calculer la moyenne d'âge des membres de genre masculin via un Consumer spécialisé pour les entiers de type IntConsumer :

```
public class Averager implements IntConsumer {

  private int total = 0;
  private int count = 0;

  public double average() {
    return count > 0 ? ((double) total) / count : 0;
  }

  @Override
  public void accept(int i) {
    total += i;
    count++;
  }

  public void combine(Averager other) {
    total += other.total;
    count += other.count;
  }
}
```

Il ne reste plus ensuite qu'à employer notre classe Averager sur l'opération collect comme suit :

```
Averager averageCollect = roster.stream()
      .filter(p -> p.getGender() == Person.Sex.MALE)
      .map(Person::getAge)
      .collect(Averager::new,  Averager::accept,  Averager:
:combine);

System.out.println(«Moyenne âge : « + averageCollect.average());
```

Conclusion

L'introduction des Lambdas expressions dans le langage marquera un tournant dans l'histoire de Java. Le langage décrit comme moribond va regagner en attrait en proposant aux développeurs la possibilité de mêler le paradigme fonctionnel au modèle objet. Les architectes d'Oracle ont eu la bonne idée de ne pas s'arrêter à cet ajout en modifiant également en profondeur les bibliothèques au coeur du JDK et en introduisant les Streams afin de permettre aux développeurs de tirer pleinement partie de la puissance de la programmation fonctionnelle dans les applications Java d'entreprise. Enfin, le fait d'abstraire le développeur du travail de parallélisation des opérations liées aux Streams va booster la puissance des applications Java tout en laissant le développeur se concentrer pleinement sur les traitements métiers à forte valeur ajoutée dans son code.

Sylvain **Saurel** - *Ingénieur d'Etudes Java / Java EE*
sylvain.saurel@gmail.com

Lync et Skype : Créez votre scénario de communication

On dit souvent « la communication est la base de toute relation ». C'est donc pour cela que les solutions de communication unifiée prennent une place de plus en plus importante dans les outils d'entreprise, voire même dans certains cas se substituent au mail. Les entreprises souhaitent toucher directement leur clientèle en utilisant des outils comme les réseaux sociaux, ou les plateformes de communication instantanée.

Microsoft est présent sur le marché des communications unifiées en entreprise depuis quelques années maintenant : Live Communications 2003 et 2005, Office Communications 2007 et 2007 R2, Lync 2010 puis 2013 mais également depuis 2011, sur le marché du grand public, grâce au rachat de Skype. Ces deux produits ne s'adressent pas à la même cible mais sont très complémentaires. Ils communiquent directement via un processus de « fédération ». Toutes les modalités de communications sont possibles : messagerie instantanée, audio, vidéo, partage de contenu comme un Powerpoint, un sondage, un tableau blanc, téléphonie, etc ... Et pourquoi ne pas utiliser tous ces éléments pour créer son propre scénario de communication? Microsoft met à disposition des développeurs des API pour créer des applications permettant :

- D'améliorer la communication entre les personnes,
- De réduire la latence des communications,
- De trouver facilement les informations grâce à des scénarios de communications automatiques,

Voici les API et quelques exemples pour commencer.

Skype

La communication entre Lync et Skype est disponible jusqu'à la modalité audio. La vidéo sera disponible pour le milieu de l'année 2014.

Skype s'intègre assez facilement dans une application métier via les « Skype URI ». Elles sont supportées sur les environnements suivants :

- Page Web
- Email
- Application Windows 8
- Application Android
- Application iPhone

Le client Skype sera lancé puis automatisé sur la modalité configurée. Si le client n'est pas installé, vous serez invité à le télécharger. L'intégration la plus simple consiste à utiliser, dans une balise href la syntaxe « skype : »

```
<a href="skype:echo123?call">Call the Skype Echo / Sound Test Service</a>
```

suivi du contact et de la modalité ou de l'action (ajouter un utilisateur par exemple) à lancer : Fig.1 et 2.

Vous pouvez également ajouter d'autres participants en séparant les utilisateurs avec un point-virgule :

```
<a href="skype:participant1:participant2?call">Call the Skype Echo / Sound Test Service</a>
```

Pour pouvoir communiquer avec un utilisateur Lync, la syntaxe doit être adaptée en ajoutant un 2 juste avant l'utilisateur :

```
<a href="skype:2:john@contoso.com?call">Call the Skype Echo / Sound Test Service</a>
```

Vous pouvez également aller un peu plus loin en utilisant :

- Les « Skype Button » : En utilisant la librairie
- La fonction javascript Skype.UI est l'équivalent du bouton précédent mais ajoute des fonctions supplémentaires

```
<div id="call_32" style="width:20%;background-color:#0094ff">
    <script type="text/javascript">
        Skype.ui({
            name: "call",
            element: "call_32",
            participants: ["echo123"],
            imageSize: 32,
            imageColor: "white"
        });
    </script>
</div>
```

Toutes ces informations sont résumées sur le site developer.skype.com

Lync

Lync est une solution de communication instantanée. Elle se différencie de ses concurrents sur la partie développement grâce à ses API clientes et serveurs. La majorité sont basées sur le framework .net et utilisent le pattern « Begin / End » pour développer en mode asynchrone.

Lync 2013 SDK

Commençons par le plus simple et le plus abordable de toute cette liste. Lync 2013 SDK est utilisable côté utilisateur et s'appuie sur la connexion du client Lync 2013 pour communiquer avec les services exécutés sur le serveur pour la présence, les conférences, etc ...

Les cas d'usages les plus fréquents sont :

- Intégration de la présence dans un portail web ou dans une application de bureau,
- Automatisation d'une conversation à partir d'une application métier,
- Ajout de l'application dans Lync avec une contextualisation.

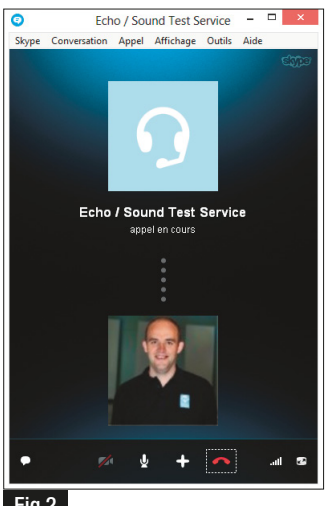
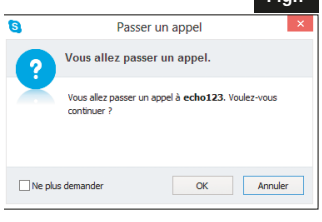


Fig.2

Comme vous l'aurez compris le client Lync est obligatoire pour développer, vous pouvez utiliser aussi bien le client Lync intégré à Office Pro ou le client Lync Basic disponible gratuitement.

Trois grandes « couches » sont exploitables et permettent de com-

Fig.1



prendre ce que nous sommes capables de faire avec cette boîte à outils : Comme son nom l'indique la couche supérieure présente une collection de contrôles WPF et Silverlight qui ne sont ni plus ni moins que les contrôles présents dans le client Lync Fig.3, 4 et 5. Ils peuvent être ajoutés dans des applications métiers et également personnalisés aux couleurs de la société. Ils possèdent également des propriétés supplémentaires pour étendre et enrichir la communication. La partie « Automation » fournit un objet pour paramétrer automatiquement une conversation Lync en fournissant les participants, la modalité de conversation à utiliser, et des paramètres de la conversation comme le premier message à envoyer.
La partie « Managed API » contient les fonctions bas niveaux de l'API pour exécuter des actions essentielles comme l'authentification ou bien encore la gestion de la liste des groupes et des contacts. Il existe un mode sans interface appelé « UI Suppression Mode » où Lync tourne en tâche de fond et donne la possibilité au développeur de réécrire complètement le client Lync. Avec toutes ces méthodes le client Lync est donc entièrement pilotable sans interagir avec l'interface par défaut.

```
var _lyncClient = LyncClient.GetClient();
var _automation = LyncClient.GetAutomation();
```

Pour terminer sur ce SDK, certains composants ou méthodes prennent en compte la contextualisation permettant de donner un sujet précis à la conversation. Une clé de registre appelée « contextual Package » contient les informations de l'application comme un GUID qui est lié avec le composant. L'utilisateur connait donc directement le sujet et le contenu de la conversation. Dans le cas d'un site web, la fenêtre de la conversation s'étend pour mettre en avant l'information Fig.6.

UCMA

Lync server 2013 présente la quatrième version d'UCMA (Unified Communication Managed API). Beaucoup de progrès ont été effectués surtout en matière de simplification. Il s'agit de l'API la plus complète et la plus puissante. Elle est utilisée par les services Lync comme les response group, pour Exchange pour la messagerie unifiée, ou bien elle sert de socle pour d'autres API (ex : UCWA ou Persistent Chat). Les applications UCMA sont exécutées sur un serveur ajouté dans l'infrastructure Lync. Il est déclaré comme «Trusted Applications Server» et maintient un lien sécurisé via un certificat. Dans certain cas il peut également être présent comme client pour simuler par exemple un utilisateur. Les cas d'usages les plus fréquents sont :

- Création de serveurs vocaux interactifs avec différentes modalités (Messagerie instantanée, Audio, ...),
- Distribution d'appel automatique,
- Diffusion de message ou d'alerte,
- Service d'enregistrement,
- Application de test pour la montée en charge.

Ces applications sont très puissantes et ont besoin d'accéder de la couche basse (SIP) à la couche Haute (Service de conférence par exemple) pour automatiser les communications. Toutes les classes implémentent des méthodes de gestion d'erreur et de répartition de la charge Fig.7. Certains développements nécessitent également des fonctions qui ne sont pas disponibles. Prenons l'exemple d'un serveur vocal interactif : vous pouvez utiliser le SDK Lync Server pour rediriger l'appel vers l'application UCMA. Celle-ci va exécuter un workflow de distribution d'appel et faire de la synthèse vocale. Cette partie est basée sur le SDK Microsoft Speech en version 11. Complété par d'autres API, vous pouvez créer des scénarios de communications sans limites !

Lync Server SDK

Cette API a été créée lors de la sortie de Lync 2010. Elle est moins importante que les deux précédentes mais possède des fonctions très puissantes. Elle est généralement utilisée en complément d'UCMA.
Les scénarios d'utilisation sont :

- Filtrer ou bloquer certains trafics de communication.
- Modifier le contenu de certains messages.
- Rerouter des messages vers une autre destination ou vers une application UCMA.
- Intégrer des logiciels dans une communication.
- Personnaliser le fonctionnement de certaines communications comme les appels téléphoniques ou bien la fédération.

Lync utilise le protocole SIP pour la signalisation des communications. C'est sur ce protocole que Lync Server SDK intervient. Il est capable de filtrer des types de messages INVITE, MESSAGE et de réaliser un traitement sur celui-ci. L'application est enregistrée sur le serveur Lync via une cmdlet Powershell (*-CsServerApplication) et utilise un manifest basé sur un langage de script appelé MSPL (Microsoft SIP Processing Language).

```
<?xml version="1.0"?>
<r:applicationManifest
r:appUri="http://www.microsoft.com/LC/SDK/Samples/ContentModification"
xmlns:r="http://schemas.microsoft.com/lcs/2006/05">
```

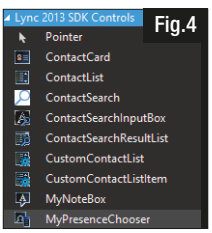


Fig.4

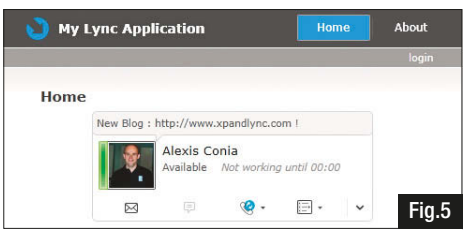


Fig.5

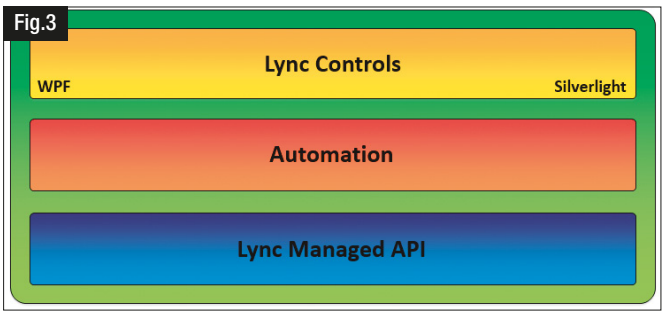


Fig.3

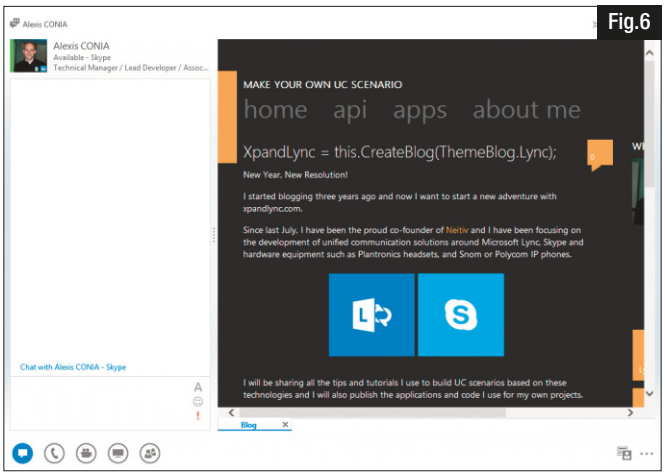


Fig.6

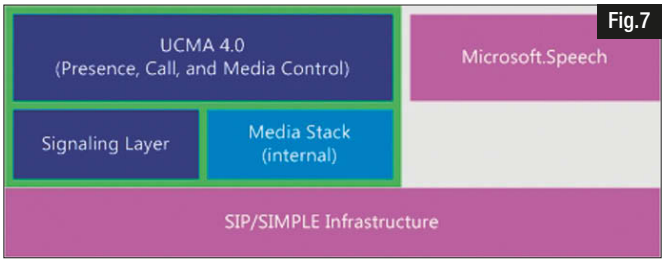


Fig.7

```
<r:requestFilter methodNames="INVITE,MESSAGE"
                 strictRoute="true"
                 registrarGenerated="false"
                 domainSupported="false"/>
<r:responseFilter reasonCodes="NONE"/>
<r:scriptOnly/>
<r:splScript>
…
```

Certaines méthodes peuvent être réalisées directement dans le script comme le forking, pour rerouter un appel vers une autre destination. Le traitement peut également être passé vers une application en code managé pour ajouter une interaction avec une base de données par exemple.
La puissance du SDK est définie par la possibilité de réaliser une action sur toutes les communications passant par le serveur, mais cela peut également être très dangereux. Une attention importante doit être donnée sur les performances afin de ne pas ajouter une latence supplémentaire aux flux temps réels.

UCWA

UCWA pour « Unified Communication Web API » est la seule API web disponible aujourd'hui sur Lync. C'est aussi une exception dans cette liste car elle est basée sur les technologies http, json, Oauth et javascript. Elle est donc utilisable à partir de n'importe quelle application même non Microsoft. Si vous utilisez Lync 2013, vous l'utilisez sans le savoir via le client Lync Mobile ou bien encore le client de conférence Lync Web App.
Il s'agit de la première version, et elle n'expose qu'une partie des modalités de communications :

- Présence.
- Gestion des informations de redirection d'appel.
- Planification et gestion des conférences en ligne.
- Messagerie Instantanée.
- Appel Audio en mode « Call via work ».

Contrairement à l'API Lync Client, il n'est pas nécessaire d'avoir un client Lync sur le poste. Les paramètres de connexion comme l'adresse du site se fait via une requête sur l'enregistrement DNS du domaine Lync comme par exemple lyncdiscover.contoso.com. Une fois authentifié, vous pouvez requêter en REST et récupérer le résultat en json : Fig.8. L'objectif de ces informations exposées par le serveur Lync est d'intégrer les données dans n'importe quelle application web. Si vous souhaitez tester les fonctionnalités et connaitre plus de détails, vous trouverez toutes les informations sur le site ucwa.lync.com

Fig.8

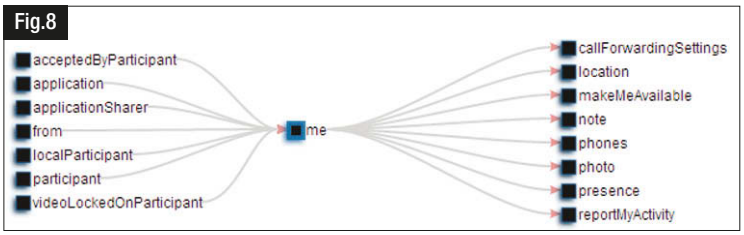


Fig.9

Lync 2013 Persistent Chat SDK

Le Persistent Chat est une fonctionnalité de Lync Server 2013, anciennement appelée « Group Chat » Elle met à disposition des utilisateurs des salles de discussion persistantes, pour par exemple créer un fil de discussion sur un projet. Chaque utilisateur vient enrichir la conversation. Les droits sont délégués à un « super – utilisateur » pour créer les salles et ajouter les différents utilisateurs Fig.9.
Certaines méthodes sont également disponibles dans le SDK Lync Client. La partie Persistent Chat est destinée à une utilisation serveur pour gérer les différentes salles, ajouter / supprimer un message mais aussi développer des plugins s'intégrant comme la contextualisation dans la fenêtre de la conversation.

Lync Software Defined Networking

Ce SDK est spécifique et lié à la partie réseau. Il contient un service supplémentaire, installé sur le serveur Lync, qui remonte les informations réseaux des trames SIP à un listener. Ce listener parse le message et renvoie les informations de qualité à l'outil de supervision. Il est par exemple utilisé par certains constructeurs d'équipement réseau pour améliorer les communications Lync sur le réseau.

A venir

La lync Conference a eu lieu lors de la troisième semaine de février à Las Vegas. Cet évènement a apporté son lot de nouveautés avec, entre-autres, l'annonce d'une nouvelle API nommé UCJA ou « Jlync ». Le nom sera confirmé lors de la sortie.
Comme UCWA, il s'agit d'une API web basée sur javascript reprenant quelques fonctionnalités mais surtout permet d'ajouter les modalités audio et vidéo directement dans une interface web. Elle offre un nombre incroyable de nouvelle interactions. Elle est annoncée pour la fin de l'année. Pour conclure, voici un récapitulatif des différentes API disponibles pour Lync :

	Client	Serveur	Online
Lync SDK	X		X
UCMA 4.0	X	X	
Lync Server SDK		X	
UCWA	X		
Lync Persistent Chat		X	
Lync SDN		X	

Grâce à ces outils et à la plateforme de communication, vous pouvez construire des scénarios très riches et personnalisés en manipulant les services aussi bien sur la partie client que serveur. Lors de la Lync Conference, Gurdeep Singh Pall a introduit le changement de l'unified communication vers l'universal communication, ou chaque personne peut communiquer de n'importe quel endroit avec n'importe quel périphérique. Ces outils vont nous aider à accélérer ce changement.

Alexis **Conia**
Lead Developer & MVP Lync – Neitiv
www.xpandlync.com - @alexis_conia

Développez avec Yammer c'est facile !

Nous sommes de plus en plus confrontés à l'utilisation des RSE (réseaux sociaux d'entreprise). Impossible de passer outre. Aujourd'hui le message de Microsoft est très clair; nous sommes dans l'air du Cloud Computing. De ce fait Yammer, plateforme de RSE, fait partie toute intégrante du nuage magique. Elle trouve sa place tout naturellement dans cet écosystème où se mêlent gestion documentaire, SharePoint et Office 365.

Comment aborder le développement ?

Yammer propose à ce jour et selon sa propre documentation, plusieurs API pouvant être exploitées assez facilement (http://developer.yammer.com/documentation/). Nous retrouvons ainsi dans le détail :

- API REST
- SDK : JavaScript, Ruby, Python
- SDK Mobile : IOS, Windows Phone 8
- Widget Embarqué : HTML/JavaScript
- OpenGraph : API centrée sur les activités liées à l'utilisateur

Vous l'aurez compris nous pouvons développer avec Yammer sur tout type de support que cela soit du Web, du client lourd, des Apps etc... les possibilités sont plurielles et les idées ne manquent pas, que cela soit sur des outils tiers, ex : outils d'analyse et de reporting (**GoodData Yammer Analytics**) ou des outils CRM. Un APP Store Yammer est présent et vous permet de consommer des apps et/ou d'en publier.

Si l'on se concentre un peu plus sur la partie REST API, nous distinguons plusieurs « end points » disponibles pour les développeurs (http://developer.yammer.com/restapi) :

- Messages
- Users
- Groups
- Search
- Activities
- Autocomplete
- Invitations
- Suggestions
- Networks

Plus concrètement, cette api se manipule uniquement en HTTPS et vous donne des réponses JSON.

Quant aux opérations fournies, uniquement du GET, POST et DELETE.

Qu'en est-il du modèle d'authentification ?

Le modèle d'authentification est basé sur OAuth 2.0, apportant une simplicité d'opération et d'implémentation sur tout type de support client. Lorsque l'on développe des Yammer Apps toute application est composée d'un identifiant applicatif public et d'un identifiant privé (secret).

Deux types d'authentifications existent. L'un pour application serveur (**Server Side Authentication Flow**) :

	Etapes	Requêtes
1	Demande d'authentification depuis une application server (client id, url de l'app)	https://www.yammer.com/dialog/oauth?client_id=[:client_id]&redirect_uri=[:redirect_uri]
2	Une redirection vers une autorisation de l'application est demandée	http://[:redirect_uri]?code=[:code]
3	Demande d'authentification depuis une application server (client id, code secret de l'app et code de l'app)	https://www.yammer.com/oauth2/access_token.json?client_id=[:client_id]&client_secret=[:client_secret]&code=[:code]
3	Un token (Bearer) est fourni afin d'accéder à une requête émise	GET /api/v1/messages/following.json HTTP/1.1 Host: www.yammer.com Authorization: Bearer abcDefGhi
4	Résultat de la requête en JSON	{«type»:»user»,»id»:1534234,»network_id»:464564,»state»:»active» .....

L'autre pour l'authentification cliente (**Client-Side Flow**) :

	Etapes	Requêtes
1	Demande d'authentification depuis une application cliente (un ID Client est fourni et l'url de l'app)	https://www.yammer.com/dialog/oauth?client_id=[:client_id]&redirect_uri=[:redirect_uri]&response_type=token
2	Redirection automatique vers l'application par le navigateur, avec le token fourni.	http://[:redirect_uri]#access_token=[:access_token]
3	Résultat de la requête en JSON	{«type»:»user»,»id»:1534234,»network_id»:464564,»state»:»active» .....

Quels outils dois-je utiliser pour gagner du temps ?

Il est vrai que compte tenu de la multiplicité des « endpoints », le résultat des objets envoyés par les requêtes sont assez complexes et les « parser » toutes une à une prendrait du temps. Si vous faites du C#, des outils comme **JSON C# Class generator** sur CodePlex (https://jsonclassgenerator.codeplex.com), ou encore un outil online tel http://json2csharp.com peuvent vous faire gagner un temps précieux. Des outils comme **RestSharp** (http://restsharp.org) peuvent vous aider dans le fait d'avoir une API purement cliente. Et sans oublier **Fiddler** (http://www.telerik.com/fiddler) pour débugger vos applications le plus finement possible.

Où puis-je trouver des exemples ?

Sans conteste la première source d'exemple est la plateforme GITHUB. Elle offre pléthore d'implémentations des API Yammer, et ce, dans plusieurs langages; vous pouvez trouver des « Wrapper » pour Ruby, JS, ou C#... (https://github.com/search?q=yammer)

Plus simplement un projet sur CodePlex, s'intitulant **Yammer.SimpleAPI**, est un simple « wrapper » pour C# et son utilisation est assez simple (http://yammersimpleapi.codeplex.com).

Un petit exemple pour la route

Ici nous utilisons une bibliothèque graphique JavaScript du nom de Infovis Toolkit (http://philogb.github.io/jit) permettant d'avoir une vue « Graphe » de nos objets. La librairie utilisée est la suivante (http://philogb.github.io/jit/ static/v20/Jit/Examples/ForceDirected/example1.html). Cela vous donnera une bonne base de départ. Dans notre exemple, après avoir créé et déclaré une application, les fichiers JavaScript Yammer, implémenté les méthodes

« **yam.config** » avec les identifiants applicatifs, appelé la méthode « **yam.getLoginStatus** », il me suffit d'implémenter simplement comme dans les exemples fournis dans la documentation Yammer, comme par exemple :

<<Code JS>>

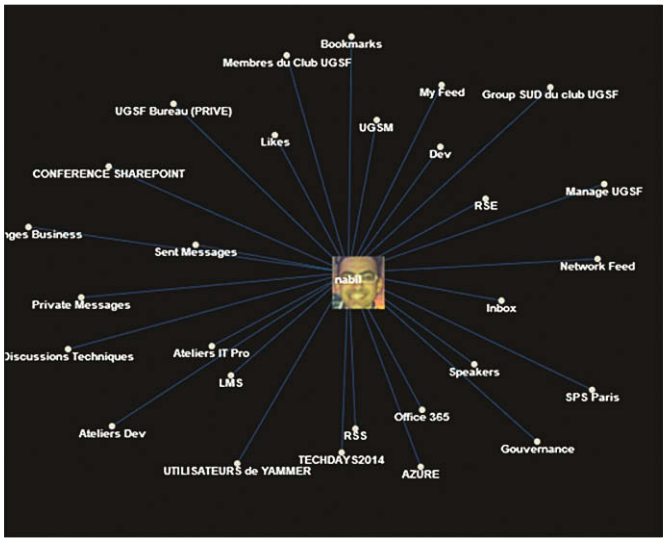
```
yam.config({appId: «APP ID»});
 yam.getLoginStatus(
     function(response) {
       if (response.authResponse) {
         var finalObject;
         yam.request({
           url: «https://www.yammer.com/api/v1/users/current.json»,
           method: «GET»,
           dataType: «json»,
           success: function (data) {
             init(data); //Appel de notre fonction Graph
           },
           error: function(xhr2, status2) {
             alert(xhr2.status);
           }
         });
       }
       else {
         alert(«not logged in»)
       }
     }
 );
```

<</CodeJS>>

Une fois que l'on a choisi quel « endpoint » nous voulons requêter, ici les données relatives à l'utilisateur courant, nous pouvons travailler avec les données JSON et les transmettre assez simplement dans notre librairie. Téléchargez l'exemple 1 et rendez-vous directement dans la fonction « init » du fichier « example1.js ».
Voici ce que nous faisons. Nous récupérons les données courantes de l'utilisateur ainsi que des préférences et nous les mettons sous formes d'adjacences dont le point de départ est l'utilisateur courant, ce qui nous donne :

<<CodeJS>>

```
function init(paramdata){
 // init data
 var adj=[];
 for (var i=0; i< paramdata.web_preferences.home_tabs.length; i++)
 {
   adj.push(  paramdata.web_preferences.home_tabs[i].name );
 }
 var json =[
   {
     //root
     id:paramdata[«id»],
     name:paramdata[«first_name»],
     data:{
       «$color»: «#557EAA»,
       «$type»: «image»,
       «$url»:paramdata[«mugshot_url»]
     },
     //dépendances
     «adjacencies»: adj
```




```
    }];
// suite du code
}
```

<</CodeJS>>

Par défaut le code fourni par cette librairie de prend pas en charge les images ! Si vous observez bien nous avons défini un nouveau type « Image ». Voici l'extension de paramètre pour cette librairie, celui-ci est à mettre en dehors de la fonction « init »

<<CodeJS>>

```
$jit.ForceDirected.Plot.NodeTypes.implement({
  'image': {
    'render': function(node, canvas){
      var ctx = canvas.getCtx();
      var pos = node.pos.getc(true);
      if( node.getData('image') != 0 ){
        var img = node.getData('image');
        ctx.drawImage( img, pos.x-15, pos.y-15);
      }
    },
    'contains': function(node,pos){
      var npos = node.pos.getc(true);
      dim = node.getData('dim');
      return this.nodeHelper.circle.contains(npos, pos, dim);
    }
  }
});
```

<</CodeJS>>

Tout cela nous donne le résultat suivant : voir figure ci-dessus.

Conclusion

Nous pouvons faire pléthore de chose avec l'API Yammer, cet article n'est que purement introductif sur les possiblités offertes. Ajoutons que l'exploitation de cette donnée est très rapide ce qui permet d'avoir des résultats assez prometteurs, dans un laps de temps très court. Avis au amateurs de développements mobiles qui souhaitent se lancer dans des applications révolutionnaires !

Nabil **Babaci**
MVP SharePoint - Consultant Senior SharePoint
http://dotnet4ever.fr

Timeline : 1998-présent-futur

Objet : Linux bouleverse l'industrie des serveurs

Conçu initialement comme un hobby, Linux va motoriser la majorité des serveurs Web en s'imposant juste avant l'arrivée de l'e-commerce, avec l'argument d'être un Unix qui fonctionne sur un simple PC.

Créateur de Linux, le finlandais Linus Torvalds continue d'en superviser le développement au sein de la Fondation Linux.

Durant les années 80, les étudiants en génie informatique ont une marotte : réécrire leur propre système Unix. Unix, c'est un système d'exploitation mis au point un peu par hasard vers le début des années 70 au sein des laboratoires Bell d'AT&T. Le but, qui paraît très simple aujourd'hui, était de permettre à un ordinateur d'exécuter plusieurs applications simultanément, et de manipuler des fichiers sur disque. Sous l'impulsion de ses deux concepteurs, les informaticiens bidouilleurs Ken Thompson et Dennis Ritchie, mais aussi pour des questions légales, Unix fut cédé en 1975 avec son code source aux milieux universitaires.

Ceci en plus d'être revendu sous forme de licences aux principaux constructeurs informatiques (IBM, Dec, HP...) afin qu'ils puissent mettre au point des solutions clients/serveur. Support de cours, Unix inspira aux universitaires de nombreux développements, eux-mêmes redistribués avec leur code source dans un but pédagogique.

Parmi ces développements, citons les systèmes BSD (du nom de l'université californienne de Berkeley) qui serviront d'OS aux premiers ordinateurs de Silicon Graphics et Sun, ainsi qu'aux premiers dispositifs de communication réseau en TCP/IP (le protocole de ce qui deviendra ensuite Internet). Mais de toutes les réécritures d'Unix, c'est celle de l'étudiant finlandais Linus Torvalds qui marquera durablement l'histoire, car elle permettra l'essor de la net-économie et, plus tard, des smartphones.

Du hobby au phénomène industriel

En 1991, Linus Torvalds se lance dans l'écriture d'un Unix qui fonctionnerait sur PC, au prétexte que le serveur Unix de son université n'était pas souvent disponible. Pour le réaliser, il télécharge la matière première depuis les serveurs du MIT, l'institut technologique de New York. Il y trouve Minix, une version simplifiée d'Unix mise au point par un professeur, et adapte son code source pour concevoir un noyau, à savoir la partie centrale du système, qui régit le multitâche et les entrées-sorties. Il ajoute ensuite autour toutes les commandes, bibliothèques et autres composants Unix que des étudiants New-yorkais ont déjà réécrits dans le cadre d'un certain projet GNU. L'histoire aurait pu s'arrêter là et le système Linux, ainsi nommé pour faire la contraction entre Linus et Unix, n'être jamais rien d'autre qu'un simple hobby.

Seulement voilà. Il se trouve que le projet GNU, initié dès 1983 par le doctorant en intelligence artificielle Richard Stallman, poursuivait un vrai but industriel, celui de bâtir un Unix libéré de toute propriété intellectuelle, dont n'importe qui pourrait faire le commerce sans avoir à reverser de droits. L'intérêt pour un tel système est fort chez la quantité de start-ups qui se créent alors dans la Silicon Valley pour vendre des technologies et des services informatiques en toute liberté. Problème, le noyau du système GNU ne sera jamais fonctionnel. Alors que celui de Linux l'est dès 1991. Actif sur les newsgroups - sortes de salon de discussion en mode texte - populaires dans les milieux des hackers de l'époque - Linus Torvalds se rend rapidement compte des opportunités de son système. En 1992, il annonce ainsi passer son noyau sous licence GPL, la licence « libre de tous droits » inventée par Richard Stallman, qui autorise quiconque à redistribuer le noyau Linux accompagné des outils GNU, tant qu'il n'y a pas de logiciel commercial dans le lot.

Livré sous forme de distributions

C'est alors que le phénomène des distributions Linux se multiplie, à savoir des kits téléchargeables sous forme d'images disque, à graver sur CD ou disquettes pour installer un Linux entièrement fonctionnel sur PC. Ces distributions Linux présentent deux intérêts. D'une part, elles mettent dans un simple PC à 6000 Frs (NDLR environ 900 €) les fonctions réseaux des serveurs Unix qui coûtent plusieurs dizaines de milliers de francs. De fait, il devient possible de proposer des serveurs Internet (routeurs, puis e-mail, FTP et Web) très peu chers. D'autre part, ce sont des environnements de développement complets, avec tous les langages disponibles et que les programmeurs peuvent utiliser gratuitement.

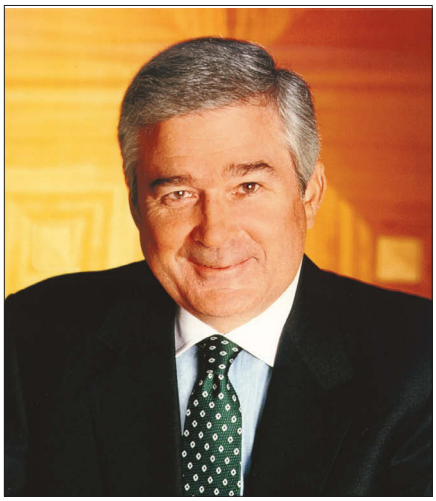
Citons la distribution Softlanding Linux System (1992), qui sera rebaptisé un an plus tard en Slackware, ainsi que sa version allemande, la Suse (1994). La Debian (1993), puis la Red Hat (1994) ajouteront chacune un système dit de gestion des paquets, lequel permet d'installer un logiciel avec tous les éléments (bibliothèques, etc.) nécessaires à son fonctionnement. Dès 1998, la Suse et la française Mandriva (initialement Mandrake) adopteront le système de paquets de Red Hat.

No 1 sur les serveurs des hébergeurs

Le succès de Linux a d'abord lieu chez les hébergeurs. En 1996, avec son logiciel serveur Apache et son matériel x86 (des PC sans écran) pas cher, Linux devient ainsi le système le plus utilisé pour motoriser les sites web, à la barbe des solutions de Sun. Microsoft décide alors de se lancer lui-aussi sur ce marché. Mais le couple Windows-serveur IIS, quoique tournant aussi sur x86, n'aura jamais la même réussite (il culminera au mieux à 10 points de parts de marché en-dessous de Linux-Apache entre 2007 et 2009).

Le décollage en fanfare de Linux surviendra vers la fin de l'année 2000, lorsque le géant IBM annonce avoir investi 1 milliard de dollars dans Linux, pour contribuer à son développement. Pour son PDG de l'époque, Louis Gerstner, le Web doit sous peu devenir une place de marché pour des boutiques et des services administratifs en ligne (il invente à cette occasion le terme e-Business). Et comme les rouages de cette nouvelle industrie seront majoritairement exécutés par des serveurs Linux, il est urgent que les

Initiateur du projet GNU et inventeur de la licence GPL, l'américain Richard Stallman gagne aujourd'hui sa vie en animant des conférences sur les logiciels libres.



L'homme d'affaires américain Lou Gerstner déclenchera le succès de Linux dans le domaine de l'e-business, en investissant un milliard de dollars dans son développement lorsqu'il était le patron d'IBM.

développeurs cessent de programmer pour Windows et qu'ils basculent sur Linux.

IBM a deux modèles : eBay et Amazon. Les deux tous premiers sites d'e-commerce (lancés en 1995) tournent sur Linux, avec le logiciel serveur Web Apache, la base de données MySQL qui mime le fonctionnement des bases de données Oracle, et les nouveaux moteurs applicatifs PHP, Perl ou Python. Tous ces logiciels sont Open Source et l'ensemble, dont l'acronyme devient LAMP, fait dès lors figure d'infrastructure standard pour toute entreprise qui veut avoir une présence sur Internet. Selon le W3Techs, Linux exécute encore aujourd'hui les deux tiers du Web mondial.

L'affaire SCO

Les premiers démêlés arrivent en 2003. L'éditeur Caldera réclame cette année-là à IBM des dommages et intérêts de 1, puis 3, puis 5 milliards de dollars pour avoir soi-disant mis du code Unix dans Linux. Dans la foulée, il exige que tous les utilisateurs de Linux lui reversent des droits de licence. Caldera, à ce moment-là rebaptisé SCO, argumente qu'il avait racheté trois ans auparavant Santa Cruz Operation, un éditeur Unix de la première heure, et qu'il a trouvé dans ses cartons les licences originales d'AT&T.

Celles-ci pourraient faire de lui l'héritier de la propriété intellectuelle d'Unix et de tous ses descendants. Mais jusqu'en 2007, année de son dépôt de bilan, SCO n'aura jamais été capable de prouver que Linux violait une propriété intellectuelle lui appartenant. Il faut dire que plus aucun système Unix ne comportait une seule ligne du code original depuis longtemps, ce que SCO avait oublié de vérifier avant d'intenter son procès.

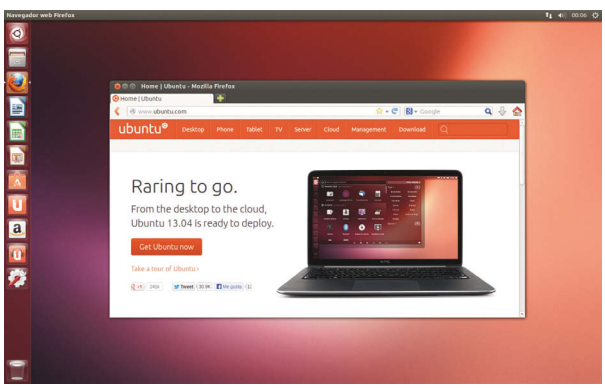
L'affaire SCO, quoique considérée par tous comme une énorme farce, aura pour effet d'exacerber les prises de position autour de la licence GPL qui régit Linux. En 2006, l'accord que passe Microsoft avec Suse - pour que chacun ait le droit de revendre le système de l'autre dans le but de l'administrer avec son propre système - ce qui permet à Microsoft de vendre du Windows aux hébergeurs pour chapeauter le fonctionnement de leurs serveurs - est à deux doigts d'être jugé illégal. Les premières box ADSL, dont la FreeBox, qui contiennent du Linux sans pour autant être livrées avec son code source, créent aussi la polémique. Les différends finiront par se tasser suite à la publication, en 2007, d'une nouvelle licence GPL.

Un succès d'estime sur PC

Du côté des machines clientes, Linux connaîtra un micro-succès éphémère sur PC au lendemain de la publication par Microsoft de son pire Windows : Vista, en 2007.

Ubuntu, une sorte de Debian conçue dès 2004 pour offrir l'expérience du Mac sur PC, canalise jusqu'en 2010 toutes les passions, mais n'équipe guère plus de 1% des PC connectés à Internet. Il offre un environnement bureautique gratuit, sans virus, mais n'exécute pas les applications grand public du PC. Malgré un bon capital sympathie, les entreprises préféreront maintenir sur leurs PC le vieux Windows XP, ou alors passer au Mac.

La notoriété d'Ubuntu chutera fortement dès 2011, avec l'arrivée d'une interface conçue pour les écrans tactiles alors que les PC en sont dépourvus. Les fervents utilisateurs de Linux sur PC (environ 30 millions dans le monde) lui préfèrent aujourd'hui la distribution Mint.



La distribution Ubuntu aura un temps tenté d'imposer Linux sur PC. En vain.

En revanche, Linux est en quelque sorte depuis 2010 le No 1 des systèmes pour Smartphones, puisque c'est sur lui que repose Android. Mis sur le marché en 2008, Android a profité de la puissance industrielle et marketing de Google, lequel n'a pas eu de comptes à rendre aux défenseurs de la licence GPL puisque le système ne contient pas les outils et librairies GNU habituels.

Yann Serra

Abonnement : Programmez, 17, Route 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

Une publication **Nefer-IT**
7 avenue Roger Chambonnet
91220 Brétigny sur Orge
redaction@programmez.com
Tél. : 01 60 85 39 96

Directeur de la publication & rédacteur en chef : François Tonic

Ont collaboré à ce numéro : Pablo Arrighi, Sylvain Saurel, Yann Serra, François Mocq

Secrétaire de rédaction : Olivier Pavie

Experts : M. Nimier-David, J-M Commets, A. G. Monod, A. Dicu, M. Hubert, P-H Gache, Z. Chalal, A. Talavera, C. Villeneuve, B. Doolaeghe, Noel Bardelot, M. Schneider-Dufeutrelle, F. Lagrede, B. Cotinat, C. Gilet, M. Belmokhtar, F. Duminy, M. Avomo, O. Vine, F. Chaachoua, I. Fofana, L. Bugnion, V. Kovalsky David, J. Knibbe, I. Khan, M. Mezil, CommitStrip, N. Babaci, A. Conia, C. Lakech, R. Linsolas, Bastien Bonnet, Christophe Pelé

Crédits couverture : 10-13-10 © mstay (tornade) 08-06-13 © whanwhanai (usine)
Maquette : Pierre Sandré

Publicité : PC Presse,
Tél.: 01 74 70 16 30, Fax : 01 41 38 29 75
pub@programmez.com

Imprimeur : S.A. Corelio Nevada Printing, 30 allée de la recherche, 1070 Bruxelles, Belgique.

Marketing et promotion des ventes : Agence BOCONSEIL - Analyse Media Etude

Directeur : Otto BORSCHA oborscha@boconseilame.fr
Responsable titre : Terry MATTARD
Téléphone : 0967320934

Ce numéro comporte deux encarts jetés sur une partie du tirage : Component Source et PC Soft.

Contacts
Rédacteur en chef : ftonic@programmez.com
Rédaction : redaction@programmez.com
Webmaster : webmaster@programmez.com
Publicité : pub@programmez.com
Evenements / agenda : redaction@programmez.com

Dépôt légal : à parution - Commission paritaire : 1215 K 78366 - ISSN : 1627-0908

Comment on pense que les applications Open Source sont maintenues

Comment sont maintenues les applications Open Source

Recruter un consultant

Recruter un chef de projet

Recruter un commercial

Recruter un codeur

CommitStrip.com

Sur abonnement ou en kiosque

Le magazine des pros de l'IT

Mais aussi sur le web

Ou encore sur votre tablette

L'INFORMATICIEN
www.linformaticien.com

ACHETEZ WINDEV MOBILE 19

OU WEBDEV 19 OU WINDEV 19

ET RECEVEZ 2 GALAXY S5

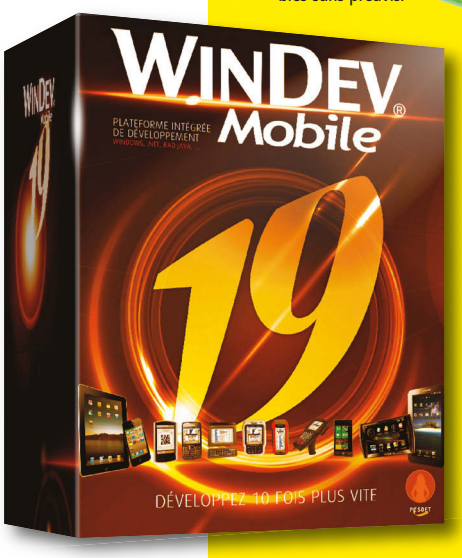
Aucun abonnement à souscrire pour bénéficier de cette offre.

2 Samsung GALAXY S5

Le tout dernier smartphone de SAMSUNG: GALAXY S5 • Ecran 5,1" 1920 x 1080 • Android 4.2.2 • APN 16M • Caméra frontale 2M • Capture vidéo full HD (1920 x 1080) • 145 grammes • Quadri Bande • 4G • GPS... • Etanche 30mn sous 1 mètre d'eau • Empreinte digitale • Emplacement carte Micro SD • Wifi • NFC • USB 3.0 • Etc...

Ou choisissez **2 Tablettes Galaxy Tab 4** (nouveau modèle) ou **2 PC portables Samsung** ou encore **1 Télé Samsung 140cm**.

Pour bénéficier de cette offre exceptionnelle, il suffit de commander WINDEV Mobile 19 (ou WINDEV 19, ou WEBDEV 19) chez PC SOFT au tarif catalogue avant le 4 juillet 2014.
Offre réservée aux sociétés, administrations, mairies, GIE et professions libérales..., en France métropolitaine. Aucun abonnement n'est à souscrire pour bénéficier de cette offre.
Le développement pour Android et iOS s'effectue avec WINDEV Mobile ou WEBDEV. Le développement pour Windows s'effectue avec WINDEV ou WEBDEV. Voir tous les détails et des vidéos sur : **www.pcsoft.fr**
Le Logiciel et le matériel peuvent être acquis séparément; merci de vous connecter au site www.pcsoft.fr pour consulter la liste des prix et les dates de disponibilité. Tarifs modifiables sans préavis.



BONDISSEZ SUR L'OPÉRATION 2 POUR 1 EURO DE +

JUSQU'AU 4 JUILLET

WINDEV Mobile 19 permet de créer facilement et rapidement des applications pour iOS, Android, Windows Phone et Windows Mobile. Liaison facile à votre SI et à toute base de données.

Fournisseur Officiel de la Préparation Olympique

www.pcsoft.fr

Des centaines de références sur le site

Java php AJAX iOS

WINDEV AGL N°1 en FRANCE

PC WORLD BEST BUY · COMPUTER MAGAZINE OPSCORE · Windows 2000 · PC PROFESSIONNEL · PC Expert · Windows · PC Mag · PC Soft

document non contractuel.