PROGRAMMEZ.COM

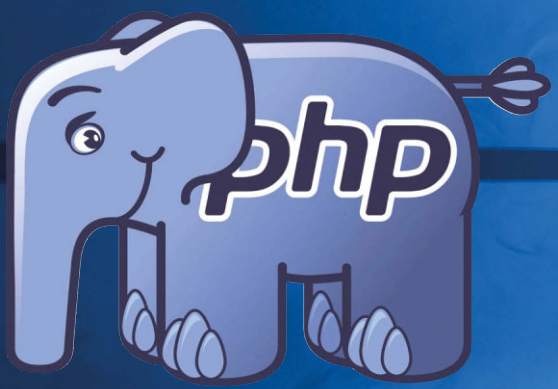
PROGRAMMEZ!

Mensuel n°188 - Septembre 2015

le magazine du développeur

PHP 7

PHP se réinvente



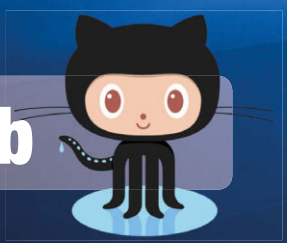
Android M

Les nouveautés d'Android

Windows 10

nouvelle architecture
nouveaux outils
nouveau store

Utiliser GitHub



Comprendre les thèmes sous Drupal 8

Carrière

Le développeur fullstack existe-t-il réellement ?

Babylon.js

Un moteur 3D pour le Web

Eliott, 14 ans, en finale de la Google Science Fair

Maker / DIY

Windows 10 IoT

sur Raspberry Pi 2

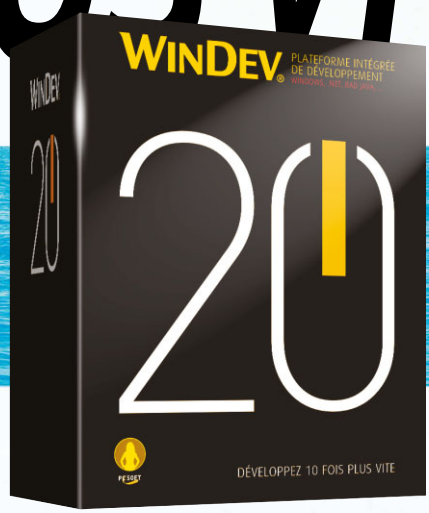
M 04319 - 188 - F: 5,95 € - RD

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

WINDEV
DÉVELOPPEZ
10 FOIS
PLUS VITE

140 pages de témoignages de sociétés prestigieuses sur simple demande (également en PDF sur pcsoft.fr)

Elu
«Langage le plus productif du marché»



Développez une seule fois, et recompilez pour chaque cible. Vos applications sont natives.

VERSION EXPRESS GRATUITE
Téléchargez-la !

Windows
Linux
Mac
Internet
Cloud
WinPhone
Android
iOS
...

Tél province: **04.67.032.032**
Tél Paris: **01.48.01.48.88**

Fournisseur Officiel de la Préparation Olympique

www.pcsoft.fr
120 témoignages sur le site

Saison 18 (2015-2016), épisode 188

ftonic@programmez.com

Nous sommes un peu comme une série TV, nous revenons en septembre pour une nouvelle saison, composée de 11 épisodes, avec moult péripéties, suspenses, dialogues haletants, rôles récurrents et les autres. Mais les scénarios sont écrits chaque semaine ou presque...

Résumé de l'épisode précédent : *alors que tous les développeurs étaient occupés par Google et Apple, une nouvelle intrigue apparaissait mois après mois : Windows 10. Jusque-là discret, Microsoft rêvait de rejouer le premier rôle et de décrocher les invitations très VIP pour les fêtes Android et iOS. La dernière image était insoutenable : le jeune Windows 10 allait-il se faire accepter par les autres ? Le petit frère, Windows 10 Mobile, faisait une apparition furtive...*

Episode 188 / spoiler alert. *Nous retrouvons notre jeune Windows 10 s'improvisant DJ pour séduire le plus de monde possible. Ce sera une des grandes intrigues de la saison. Mais rien n'est gagné, car, les autres regardent méchamment ce nouveau venu et là, l'affrontement final est inévitable. Mais la conquête du trône de Westeros est loin d'être terminée. Imaginez que Franck Underwood gouverne Port-Réal !*

Windows 10 est là ! *Sauf à être déconnecté du monde informatique depuis 6 mois, le nouveau système Windows est arrivé. Impossible de ne pas en parler dans ce numéro de rentrée de Programmez ! Pour ce premier mini-dossier, nous allons aborder la plateforme Windows, les nouvelles applications universelles, l'architecture et quelques-unes des nouveautés dédiées aux développeurs. Vous allez avoir du travail ! En extra bonus, nous vous proposons de découvrir et d'installer Windows 10 IoT sur votre Raspberry Pi 2 !*

PHP 7, Android, Docker, Drupal 8, BabylonJS, C++... *Une rentrée très chargée pour Programmez ! Nous vous proposons de découvrir d'urgence le prochain PHP : PHP 7. Si le projet PHP 6 a été un échec, la v7 s'annonce comme réussie avec beaucoup d'améliorations et de nouveautés. Nous faisons le point complet. Nous parlerons aussi des nouveaux Android : Android M. Nous parlerons aussi de Docker avec un cas d'usage très précis, des thèmes sous Drupal 8, du moteur 3D JavaScript, BabylonJS. Sans oublier, un peu de NoSQL et de Big Query ! On vous aidera aussi à bien démarrer avec GitHub.*

Côté carrière, nous évoquerons un sujet mal connu en France : le développeur fullstack, mythe ou réalité.

Merci de votre fidélité au magazine.

à lire dans le prochain numéro n°189 en kiosque le 30 Septembre 2015

PYTHON, LE LANGAGE MÉCONNU
Souvent méconnu des développeurs, Python est pourtant un langage surdoué et surpuissant.

NODE.JS DE A à Z :
Et si vous utilisiez JavaScript côté serveur ?

CONTENEUR
On parle beaucoup des conteneurs, de Docker, mais qu'est-ce cela change concrètement pour le développeur ?

Combien gagne un développeur indépendant ?

Hopwork a publié début juillet son baromètre de l'emploi freelance. Hopwork met en relation les freelances et les entreprises. Les chiffres donnés sont liés uniquement à cette plateforme, mais ils donnent une orientation.

Source : https://www.hopwork.com/stats/barometer

Les développeurs et graphistes / designers représentent la majorité des freelances enregistrés sur la plateforme.

Catégories	proportion	tarif moyen
Développement Web et Mobile	17,09%	337 €/jour
Graphistes, motion designers et photographes	32,94%	322 €/jour
Développement Backend	22,53%	391 €/jour
Marketing et chefs de projet	13,99%	459 €/jour
Professions administratives	2,54%	420 €/jour
Community managers et rédacteurs	10,91%	240 €/jour

Par langage

Langage	Recherches	Tarif moyen	Paris	Marseille	Lyon	Nice
java	40,6%	431 €/jour	522 €/jour	414 €/jour	449 €/jour	333 €/jour
PHP	29,1%	326 €/jour	394 €/jour	335 €/jour	375 €/jour	277 €/jour
JavaScript	10,3%	376 €/jour	433 €/jour	333 €/jour	411 €/jour	294 €/jour
ruby	8,9%	440 €/jour	488 €/jour	320 €/jour	455 €/jour	402 €/jour
python	3,5%	428 €/jour	512 €/jour	367 €/jour	456 €/jour	475 €/jour
scala	3,4%	575 €/jour	606 €/jour	-	638 €/jour	-
c#	2,5%	347 €/jour	434 €/jour	400 €/jour	394 €/jour	308 €/jour
c++	1,7%	365 €/jour	427 €/jour	315 €/jour	383 €/jour	445 €/jour

Java demeure largement demandé. PHP pèse 30 % des demandes. Les autres langages sont très loin derrière. Sans surprise, nous retrouvons les différences Paris – province. L'écart sur certains langages est important (Java, PHP, JavaScript). Par contre, des profils spécifiques ou visiblement relativement peu fréquents en province sont plus chers dans certaines villes de province : C++ (Nice), Scala (Lyon). On note parfois des écarts très importants entre les différentes villes. Marseille est souvent assez largement sous le tarif moyen. Lyon semble très dynamique.

Les bases de données

Base de données	Recherches	Tarif moyen	Paris	Marseille	Lyon	Nice
mysql	26,6%	351 €/jour	399 €/jour	365 e/jour	407 €/jour	386 €/jour
postgresql	17,8%	465 €/jour	534 €/jour	469 €/jour	468 €/jour	400 €/jour
mongodb	15%	468 €/jour	534 €/jour	350 €/jour	440 €/jour	400 €/jour
elasticsearch	14,7%	538 €/jour	600 €/jour	550 €/jour	466 €/jour	-
oracle	14,5%	453 €/jour	505 €/jour	517 €/jour	510 €/jour	330 €/jour
cassandra	7,1%	635 €/jour	664 €/jour	-	300 €/jour	400 €/jour
solr	2,4%	399 €/jour	518 €/jour	750 €/jour	500 €/jour	-
sqlserver	1,2%	384 €/jour	478 €/jour	-	600 €/jour	-
neo4j	0,7%	553 €/jour	568 €/jour	-	-	-

Sans grande surprise, la base de données est un marché éclaté. MySQL représente un quart des demandes. Là, encore nous trouvons une différence Paris – province. Marseille est régulièrement sous le tarif moyen. Cassandra est peu demandé, mais son tarif est élevé. Toulouse (non représenté ici) arrive à 790 € / jour !

Le projet Titan d'Apple

(qui serait une voiture autonome) viderait les équipes des autres projets

Microsoft

se sépare de 1200 personnes qui s'occupaient de la plateforme de publicité

Java 9

est attendu, ou pas, pour le mois de septembre 2016.

Swift 2.0

sera-t-il disponible sur Windows ? Apple a annoncé le portage officiel sur Linux mais rien sur Windows. Microsoft proposera quelque chose. Mais quoi et quand ?

Google

mise tout sur Android Studio et arrête Android Developer Tools in Eclipse

Procès Google – Oracle :

la cour suprême américaine a rejeté une demande de Google et devra donc justifier et défendre l'usage loyal (Fair Use) des lignes de codes Java dans Android...

Le projet

Ara de Google

n'est pas mort ! la preuve : https://www.youtube.com/watch?v=HyUQ4uD9Y3U

Samsung

agrandit encore les écrans avec le Galaxy S6 Edge+...

Les **montres connectées**

seront-elles toutes rondes ?

Des grands patrons de la **Silicon Valley** **investissent dans une douche économe en eau**

Bonjour **Alphabet**, **Google simple filiale...**

WebAssembly : le futur du Web ?

Entre 2 librairies **DHT11**, **mon code balance avec des bugs (private joke)**

Edge **est performant mais pas forcément bon sur HTML5**

La **NASA** **lance un concours pour le design de sa future app pour montre connectée !**

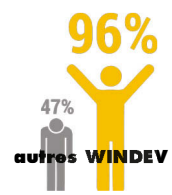
DÉVELOPPEMENT PROFESSIONNEL

WINDOWS, LINUX, IOS, ANDROID, CE, WEB...

100 TEMOIGNAGES DE SOCIÉTÉS PRESTIGIEUSES, C'EST UTILE !

Vos clients, vos utilisateurs ont besoin de solutions rapides et fiables, dans tous les environnements, sur **tous les matériels**.
Quoi de mieux qu'un AGL-ALM **compatible avec tous les systèmes et tous les matériels**?
Nous sommes heureux de vous offrir ce numéro spécial publi-dossier de la revue **01 Net** présentant **140 pages de témoignages dans tous les domaines (applications stratégiques, applications départementales, sites Internet et Intranet, applications mobiles)...**, sur tous les matériels.
Découvrez ce que **WINDEV** vous apporte, et apporte à vos utilisateurs et clients.
Vous ne vous satisfaites pas des simples messages commerciaux, bien entendu.
C'est pour cela que ces 140 pages de témoignages récents vous permettent de vous forger **votre propre avis**.
Disponible en **lecture libre sur le site www.pcsoft.fr**, ou sur votre bureau demain (offre d'envoi gratuit réservée aux professionnels).

Philips, VINCI Autoroutes, Quick, Système U, Fédération Française de Basket-Ball, Bolloré Africa, Casio, Taittinger, CCI de Bordeaux, VOLVO Car France, AIGLE, Siemens VAI, Truffaut, Air Calédonie, HONDA, Comtesse du Barry, Ministère de l'Education Nationale, Ecole d'ingénieurs de Paris, EcoleDirecte.com, Isotoner, Hôpitaux de Paris, Autosur, Société Générale, Photomaton®, Groupama, ...



WINDEV :
UN TAUX DE SUCCÈS DES PROJETS SANS ÉQUIVALENT

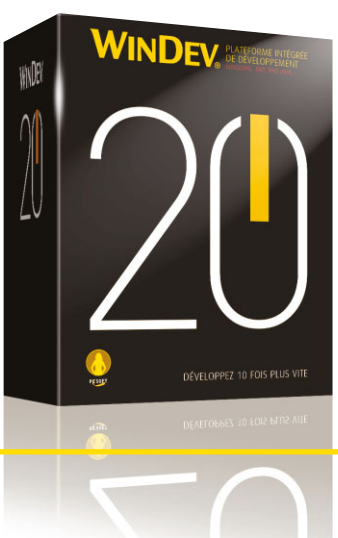
PC SOFT 25 ANNÉE Au service des développeurs

WINDEV®

WINDEV AGL N°1 en FRANCE

PC WORLD BEST BUY · COMPUTER MAGAZINE TOP SCORE · Windows 2000 · PC PROFESSIONNEL · APPS · Windows · PC PRATIQUE · W PLUS · PC EXPERT · 1

Fournisseur Officiel de la Préparation Olympique



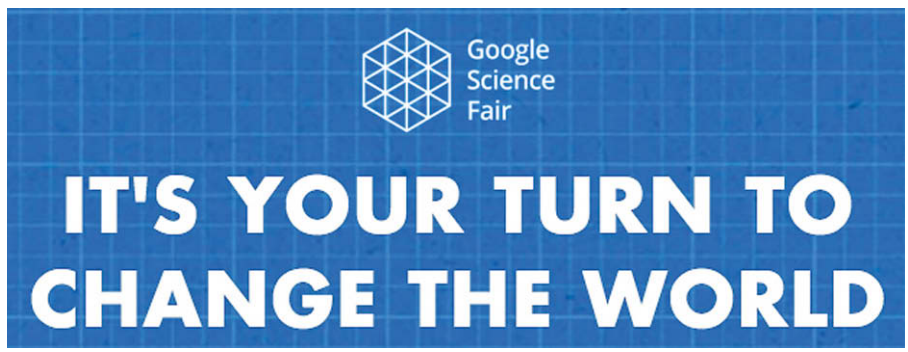
www.pcsoft.fr

Logiciel professionnel.

Eliott, 14 ans : la passion du code et de l'Arduino lui ouvre les portes de la finale mondiale de Google Science Fair

Il n'y a pas d'âge pour faire du code et de l'électronique ! Début août, ce fut une belle surprise pour Eliott Sarrey, 14 ans seulement, Google venait de dévoiler la liste des projets sélectionnés pour la finale du Google Science Fair. Son projet, Bot2Karot était bel et bien sélectionné. Il lui reste moins de 3 semaines pour terminer le prototype, améliorer le code, réaliser les vidéos et peaufiner les présentations qui seront faites durant la finale.

François Tonic



Google Science Fair est un concours international pour les 13-18 ans. La finale se déroulera dans les locaux de Google. La sélection a été rude, seuls 20 projets sont sélectionnés pour le grand jour (le 21 septembre). Eliott est le seul Français sélectionné.

Du très concret !

La première idée de Bot2Karot est venue très simplement. « J'ai simplement regardé l'état de notre jardin et du manque d'arrosage. Des jeux sur smartphones existent, mais n'ont pas d'utilités réelles. C'est là que j'ai eu l'idée d'une solution pour arroser le jardin. Mais je voulais aller plus loin et j'ai alors imaginé de nouvelles fonctions comme biner. » Raconte le jeune développeur. Malgré son jeune âge, Eliott n'est pas un débutant en informatique et en technologie. Très tôt, il utilise le Lego Mindstorm et programme quelques montages. Le bricolage, il l'a dans le sang ! « J'ai une fraiseuse à commande numérique. Elle est très limitée, mais je peux usiner les pièces que je veux ! Et J'ai été à l'initiative du club informatique de mon collège. » poursuit enthousiaste Eliott.

Il passe le cap du Maker et du Do It Yourselft (le fameux DIY) avec Arduino. « L'Arduino est vraiment très simple. Les branchements ne sont pas très difficiles, on trouve très facilement du code pour faire fonctionner les composants. J'ai rapidement pu tester et monter de petits servomoteurs. » Explique-t-il.

Peu à peu l'idée germe et se construit. Quand il entendu parler du concours Google. Il s'inscrit. Et surtout, comme il le dit lui-même : avoir une date précise lui a permis de fixer un objectif pour imaginer, concevoir et donner vie à son Bot2Karot.

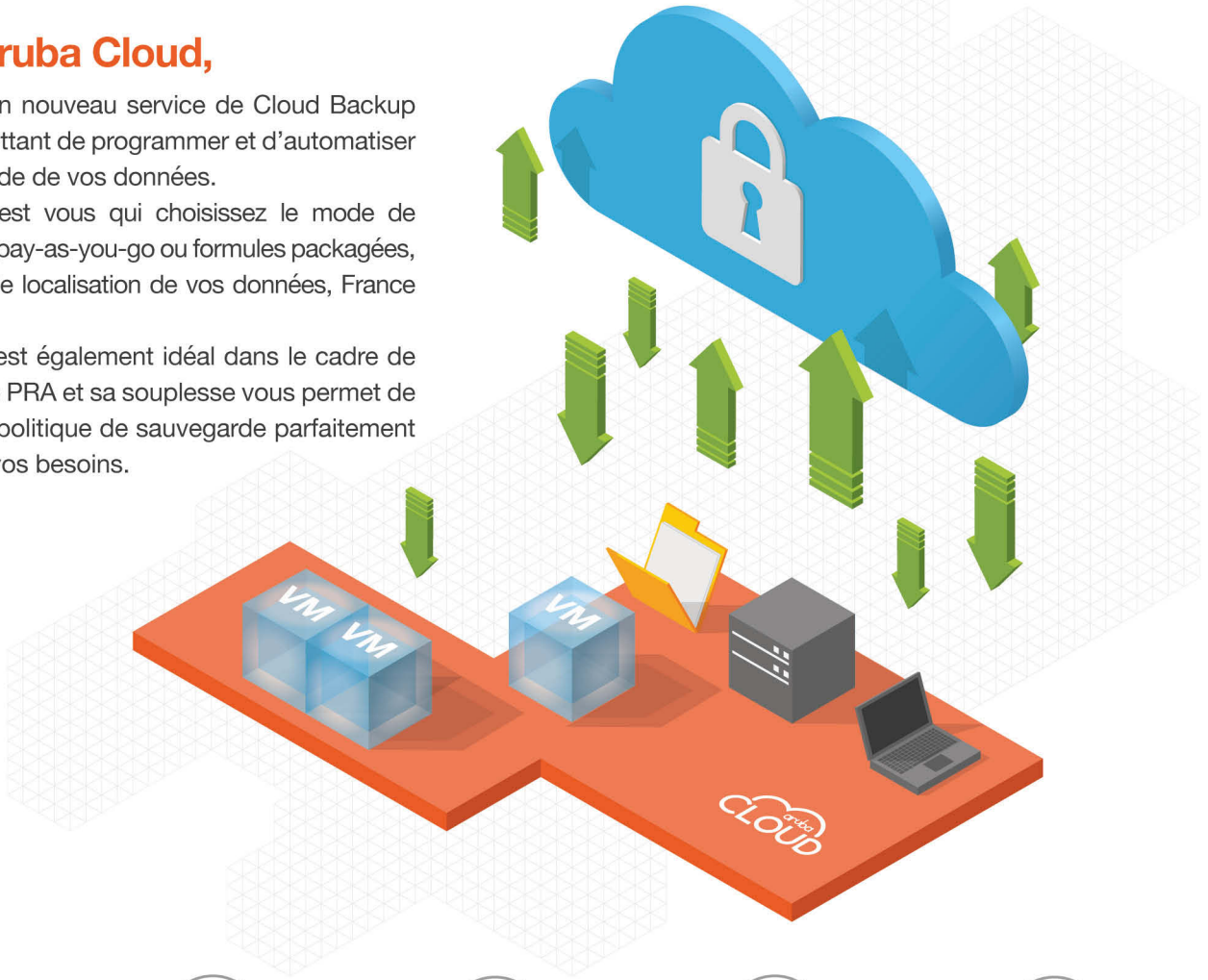
Bot2Karot : le potage à l'heure du numérique

La première fonction de Bot2Karot a été de gérer et d'assurer l'arrosage, mais rapidement, d'autres idées arrivent. Comment avoir des légumes frais dans son potage ? Comment repiquer une plante depuis une serre ? Eliott imagine alors un véritable potager miniature s'organisant en carrés. Un robot s'assure de l'arrosage et du repiquage en pleine terre des plantes.

Comment automatiser *mes sauvegardes* sur le Cloud ?

Avec Aruba Cloud,

profitez d'un nouveau service de Cloud Backup vous permettant de programmer et d'automatiser la sauvegarde de vos données.
De plus, c'est vous qui choisissez le mode de facturation, pay-as-you-go ou formules packagées, et le pays de localisation de vos données, France ou étranger!
Le service est également idéal dans le cadre de politique de PRA et sa souplesse vous permet de définir une politique de sauvegarde parfaitement adaptée à vos besoins.



- Sécurité optimale
- Simple d'utilisation
- Sauvegarde automatique
- Economique
- Multi-plateforme

1 Quitte à choisir une infrastructure IaaS, autant prendre la plus performante!
Aruba Cloud est de nouveau **N°1 du classement des Cloud**
JDN / CloudScreener / Cedexis (avril 2015)

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

aruba CLOUD

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

MY COUNTRY. MY CLOUD.*

*MON PAYS, MON CLOUD.

Bot2Karot s'organise autour de plusieurs éléments :

- Un robot qui roule avec un bras articulé. Il possède différents outils et fonctions.
- Une application smartphone pour piloter le potager
- Le potager en lui-même créé et organisé pour faciliter le travail du robot.

Le potager est organisé en carrés et utilise des plaques en résines que l'on emboîte. Le robot circule sur des chemins définis pour éviter la boue et la terre. Le jardin s'adapte donc aux besoins réels de la personne.

Eliott a aussi cherché la composition idéale de la terre. Son grand-père l'a beaucoup aidé dans cette recherche avec les jardins lasagnes.

Une longue préparation

Eliott a géré Bot2Karot comme un vrai projet. Avant de se lancer dans la réalisation, il a tout modélisé avec SketchUp, un des outils 3D les plus populaires. « Cela m'a pris plus de 2 mois. J'ai modélisé l'ensemble de Bot2Karot : le potager, le robot, les déplacements. Petit à petit, j'ai corrigé des erreurs que je n'avais pas vues ou qui sont arrivées au fur et à mesure. » Explique simplement le jeune modeleur.

Mais entre le modèle 3D et la réalité, parfois il y a des couacs. « La modélisation 3D c'est super, mais parfois on ne voit pas tout et le modèle donne l'impression que les pièces sont grandes, mais en réalité, il y a un problème de dimensions. Quand j'ai commencé à usiner et imprimer les nombreuses pièces nécessaires, je me suis aperçu que des plaques prévues dans le modèle 3D m'empêchaient d'installer les roulements à billes pour les roues ! J'ai dû reprendre la modélisation, modifier le modèle et réusiner les pièces. » Comme un vrai pro !

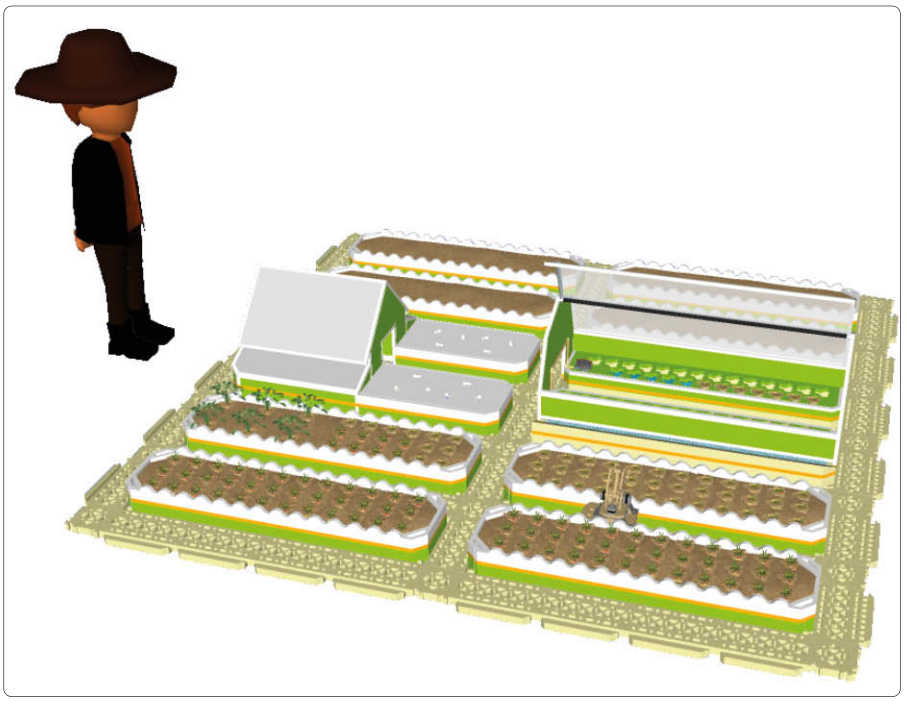
Avec ce projet, Eliott a touché tout : robotique, électronique, programmation, biologie ! Il connaissait un peu la programmation, mais son père l'a aidé à coder certaines fonctions et un ami, Virgile, lui a montré comment optimiser la vitesse du robot. La partie programmation a pris une grosse semaine.

Un robot simple et complexe à la fois !

Le robot est la pièce maîtresse du projet. Il remplit plusieurs fonctions : arrosage, transport des plantes, repiquages, contrôle les niveaux d'eau, prendre des photos, gestion de ses batteries, etc. Il possède plusieurs outils comme une herse, un tuyau d'eau. Le bras articulé est l'élément le plus complexe du projet. Plusieurs dizaines de pièces ont été nécessaires.

Le cœur du robot est animé par une Arduino Mega. À cela s'ajoutent 5 moteurs pas à pas avec des shields Pololu pour les moteurs, et des shields RAMPS 1.4. Ces shields simplifient les branchements et l'utilisation des moteurs. « La partie électronique a été finalement assez simple, car c'est quasiment la même qu'une imprimante 3D ! » précise Eliott. L'utilisation des RAMPS a aidé Eliott dans la programmation en évitant de tout coder et gérer les multiples connexions.

Basiquement, le robot connaît sa position (approximative). Pour mieux se repérer, les bordures des bacs sont endurées et une roulette installée sur le robot lui permet de se positionner correctement (par exemple : savoir où s'arrêter, où aller pour faire un trou et poser la plante). La future app mobile permettra une navigation plus précise.



« J'ai été surpris et super content »
(suite à sa sélection)

Une app mobile pour les contrôler tous !

Le pilotage de l'ensemble se fait (se fera) par une application mobile. Le potager apparaîtra en 3D sur son mobile. On sélectionne le bac à arroser et l'app proposera les actions possibles. Une fois la fonction choisie, le robot s'exécute. Pour assurer la connectivité entre l'application, la serre et le robot, une carte Raspberry Pi a été installée par assurer la partie serveur / réseau. L'app mobile reste à créer, mais pour Eliott, ce n'est pas forcément un gros problème, car l'app ressemble beaucoup à un jeu mobile.

D'autre part, actuellement, le robot n'est pas énergétiquement indépendant, idéalement, il faut installer des panneaux solaires ou des éoliennes et il faudra mieux connaître la consommation électrique de l'installation et la consommation d'eau. Mais, Eliott a déjà réalisé un très gros travail et le robot fonctionne.

Ce n'est pas fini !

« J'attendais avec impatience l'annonce des finalistes devant mon ordinateur, avec de nombreux amis. » se rappelle Eliott. « J'ai été super content. Mais je me suis tout de suite dit qu'il y avait encore beaucoup de travail : prototype à peaufiner et à terminer, codes à améliorer, présentations à préparer, faire des vidéos » poursuit-il. Il lui reste moins de 3 semaines pour tout faire... Et après le concours ? « Ce sont des portes qui s'ouvrent à moi, je ne sais pas encore lesquelles, mais ce concours est très positif. Les commentaires sont plutôt encourageants. Lors de la finale, je vais rencontrer d'autres personnes. Ce projet m'a fait toucher à tout. Le code me plaît bien, mais la modélisation aussi. » conclut-il. À la rédaction, nous sommes impressionnés par le travail réalisé et l'ampleur du projet. Bravo à Eliott pour Bot2Karot.

Ressources :

Site officiel : http://www.bot2karot.website

Google Science Fair : https://www.googlesciencefair.com/fr/

1ER ÉVÉNEMENT EUROPÉEN
LIBRE & OPEN SOURCE

OPEN FOR INNOVATION

opensourcesummit.paris

#OSSPARIS15

PARIS OPEN SOURCE SUMMIT

18&19 NOVEMBRE

DOCK PULLMAN
Plaine Saint-Denis

PARTENAIRES INSTITUTIONNELS

SPONSORS PLATINUM

SPONSORS GOLD

SPONSORS SILVER

POUR TOUTE INFORMATION COMPLÉMENTAIRE :
Email : contact@opensourcesummit.paris – Tel : 01 41 18 60 52

un événement

Windows 10 IoT arrive sur Raspberry Pi 2

Windows 10 est sorti le 29 juillet dernier. A peine deux semaines plus tard, l'édition IoT de Windows 10 a connu une très importante mise à jour. Voici un tutoriel pour l'installer pas à pas. La rédaction.

Michaël FERY
Consultant Infinite Square
INFINITE SQUARE

Nous allons installer Windows 10 IoT Core Insider Preview sur un Raspberry Pi 2 et y déployer notre première application.
Les tutoriels proposés par Microsoft sont très complets et détaillés et je vous invite à les consulter en suivant ce lien : http://ms-iot.github.io/content/win10/SetupRPI.htm
Nous aborderons également les problématiques que l'on peut rencontrer durant cette installation.

Prérequis :

Il faut avoir une machine sous Windows 10. Celle-ci sera nécessaire, non seulement pour l'installation de Win10 IoT Core sur une carte SD (minimum 8 Go) mais également pour le développement et le déploiement de vos apps sur le Raspberry Pi 2.
Enfin, au niveau matériel, vous devez disposer d'un Raspberry Pi 2 et d'une carte micro SD.

Installation

Première étape : Installation sur la carte SD

Téléchargez l'image de Windows 10 destinée au Raspberry Pi 2 sur le Microsoft Download Center.
Lancez l'ISO téléchargé pour en voir le contenu et exécutez le fichier Windows_10_IoT_Core_RPi2.msi.
Décompressez le fichier téléchargé. Ouvrez alors le répertoire C:\Program Files (x86)\Microsoft IoT\FFU\RaspberryPi2 et vérifiez qu'il contient le fichier flash.ffu qui servira à l'installation.
Insérez ensuite la carte micro SD dans le PC tournant sous Windows 10.
Jusqu'à la release d'août il était nécessaire d'utiliser la console en mode administrateur avec des commandes barbares pour flasher sa carte SD du genre dism.exe /Apply-Image /ImageFile:flash.ffu /ApplyDrive:\\.\PhysicalDriveN/SkipPlatformCheck .
Désormais, la tâche est simplifiée puisque le programme que nous avons installé auparavant nous a fourni un outil nommé IoTCoreImageHelper.exe. L'utilisation de la recherche du « nouveau » menu démarrer vous aidera à le trouver Fig.1.
Une fois le programme lancé, sélectionnez votre carte SD cible et cliquez sur « Flash ».
Vous pouvez ensuite retirer la carte SD de l'ordinateur en pensant bien sûr à la retirer "en toute sécurité" pour éviter de se retrouver avec une image corrompue.
Il ne vous reste alors pour l'installation qu'à insérer la carte dans votre Raspberry Pi 2 et à le brancher. Pensez également à le connecter au réseau, avec un câble RJ45 tout simplement, car c'est ainsi que nous pourrons l'administrer à distance.
Si votre carte est connectée à un écran via son port HDMI vous devriez voir cet écran : Fig.2.
Vous pouvez y voir le nom de votre carte ainsi que l'adresse IP qui lui a été attribuée sur le réseau.

Web Server

L'image Windows 10 que nous avons installée a également installé un serveur web sur la carte et celui-ci nous permet d'y accéder depuis un simple navigateur avec l'adresse IP que nous avons récupérée :
De nombreuses fonctionnalités sont disponibles comme le listing des applications installées ou les performances CPU ou I/O Fig.3.
Nous pouvons d'ailleurs y voir que l'écran par défaut que nous avons affiché sur notre Raspberry est en réalité celui de l'application

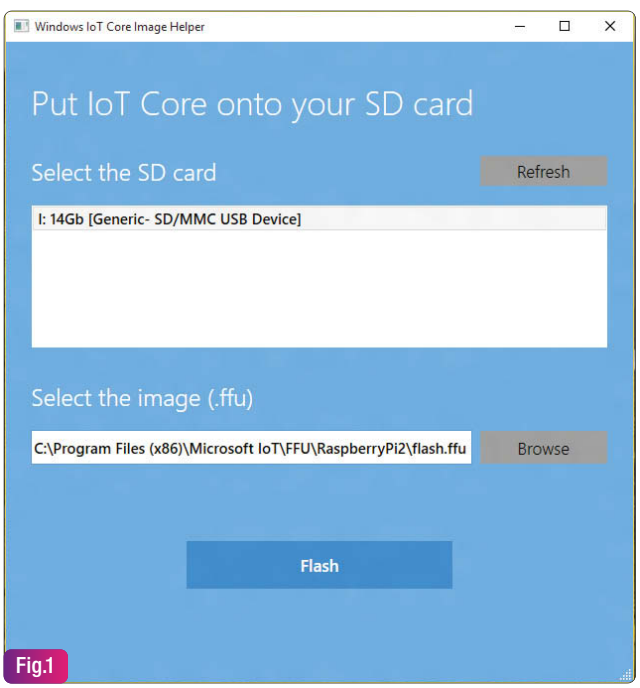


Fig.1

Fig.3

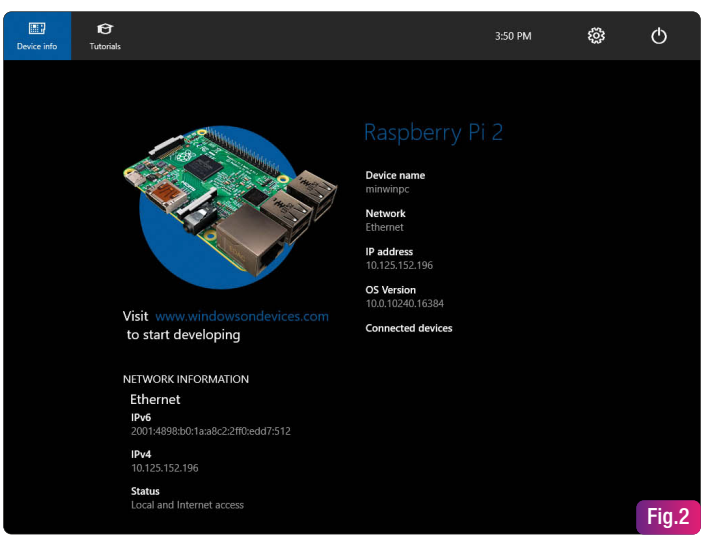


Fig.2

Faites le plein de codes pour la rentrée

Abonnez-vous à PROGRAMMEZ! le magazine du développeur

Nos classiques

Tarifs France métropolitaine

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

Nos goodies

Clé USB ..... 29,90 €

Clé USB contenant tous les numéros de Programmez depuis le n°100

Nos offres de la rentrée

Abonnement

1 an → 64,90 € *
2 ans → 94,90 € *

(*) Valeur du CPL 79,90 €
Valeur clé USB 29,90 €
Frais logistiques : 15,90 €

Inclus un kit complet CPL dLAN 550 de DEVOLO et une clé USB contenant tous les numéros de Programmez depuis le n°100

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

Toutes nos offres sur www.programmez.com

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 2 ans au magazine :*** 79 €*
- ☐ ***Abonnement étudiant 1 an au magazine :*** 39 €*
  Photocopie de la carte d'étudiant à joindre
- ☐ ***Clé USB contenant tous les numéros de Programmez! depuis le n° 100 :*** 29,90 €*

Offre de rentrée :
Programmez + kit CPL dLAN 550 + Clé USB

- ☐ ***Abonnement 1 an :*** 64,90 € (au lieu de 158,80 €) *
- ☐ ***Abonnement 2 ans :*** 94,90 € (au lieu de 188,80 €) *

☐ 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 188

***Offre limitée**, valable jusqu'au 30 septembre 2015*

par défaut installée sur celui-ci : Fig.4.
A nous désormais d'installer notre première application.

Développement et déploiement pour Raspberry Pi 2

Visual Studio 2015 et Windows 10 SDK

Pour commencer à développer sur Windows 10 IOT Core vous devez avoir installé Microsoft Visual Studio 2015 RC.
L'installation de VS2015 est disponible sur Windows 8.1 et le développement d'apps Win 10 également mais vous n'aurez à ce jour pas accès au designer ni au déploiement. Je vous conseille donc bien de développer depuis Windows 10.
La subtilité à ne pas rater est qu'il faut choisir une installation custom et de bien installer le SDK de développement Windows 10 avec VS2015. Si, comme moi, vous rencontrez des problèmes à installer ces outils en même temps que Visual Studio, je vais vous faire gagner du temps, des reboots inutiles et des nerfs qui lâchent. Essayez tout simplement de récupérer l'installateur autonome de ces outils « Standalone Windows 10 SDK for Windows 10 » : à l'adresse suivante : https://dev.windows.com/en-US/downloads/windows-10-developer-tools

SDK IoT Core

Au niveau code, nous n'allons pas rentrer dans le détail mais sachez qu'il existe déjà de nombreux samples sur le site de Microsoft pour bien débuter à cette adresse : https://ms-iot.github.io/content/win10/StartCoding.htm

Il est bon toutefois de noter que ceux-ci possèdent déjà une référence vers le SDK Windows 10 IoT Core mais que si vous souhaitez en créer une from scratch vous devrez l'inclure vous-même. Ce SDK vous permet notamment l'accès au namespace Gpio qui est utile lors des accès aux I/O (entrées/sorties) des broches du Raspberry Pi Fig.5 et 6.

Watcher

Il est possible que vous ne souhaitiez pas brancher d'écran sur votre Raspberry mais que vous souhaitiez voir à tout moment quels sont les micro-ordinateurs tournant sous Windows 10 IoT Core sur votre réseau. Pour cela, vous devez installer le msi qui se situe dans le dossier que vous avez décompressé au début pour installer l'image sur la micro SD. Ce msi installera un Watcher qui liste donc tous les Windows 10 IoT Core sur le réseau avec leurs IPs et leur petit nom. C'est utile si l'affectation des adresses IPs est amenée à changer régulièrement sans avoir à monitorer celles-ci directement depuis le Raspberry Fig.7.

Déploiement

Microsoft a bien fait les choses puisque l'on peut s'appuyer sur l'intégration complète du développement sur les plateformes IoT de type Raspberry Pi. Ainsi, créer une application pour IoT Core revient à créer un nouveau projet sous Visual Studio 2015 et à presser F5. En effet il est possible d'effectuer le déploiement et debug de nos apps à distance toujours depuis Visual Studio. Pour cela, Windows10, VS2015 et le Windows 10 SDK sont nécessaires mais tous disponibles gratuitement avec VS2015 Community et la migration Windows 10 des anciennes versions depuis Seven.
Le déploiement/debug s'apparente énormément à celui d'une app Windows Store puisqu'elle utilise la fonctionnalité déjà existante de Remote Device.
Comme nous l'avions déjà indiqué en retour de la Build, le déploiement d'une app Win10 IOT Core est très simple. En effet, si vous avez l'habitude de débugger des apps Windows 8 ou 8.1 sur des tablettes distantes en wifi, vous ne serez pas dépaysés. Dans les propriétés de votre projet, dans l'onglet Debug, vous pourrez choisir "Remote Machine" et sélectionnez la cible. Il arrive que le listing ne fonctionne pas bien et dans ce cas-là, saisir l'IP ou le nom de la machine cible fonctionne tout aussi bien Fig.8.
Vous avez votre cible et votre application, alors déployons : Fig.9.
Vous devriez désormais vous retrouver sur votre Raspberry Pi 2 avec votre application lancée et celle-ci devrait apparaître dans les Running Apps de la machine.
Vous pourrez alors décider de la passer en application par défaut afin que ce soit celle qui soit lancée au démarrage du Raspberry.

Conclusion

Il est important de noter que la version actuelle de Windows 10 IoT Core est enfin sortie dans sa version finale à peine quelques jours après la version Desktop et que comme ses grandes sœurs pour mobile et bureau, elle sera probablement amenée à évoluer dans les mois à venir. Le développement de drivers pour certains devices et pour écrans tactiles notamment permettra de débloquer des scénarios plus aboutis.

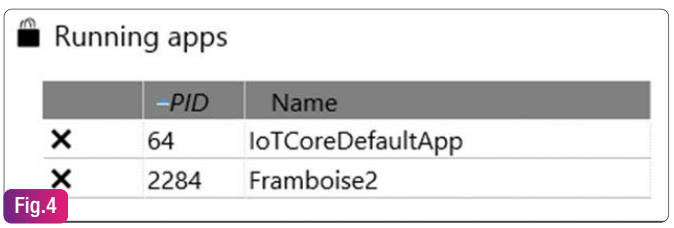


Fig.4

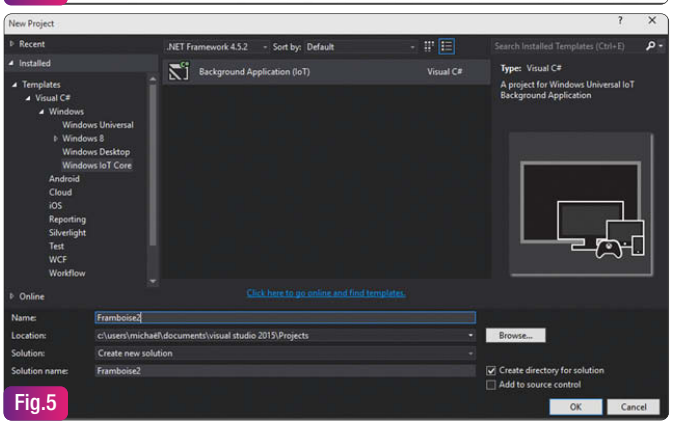


Fig.5

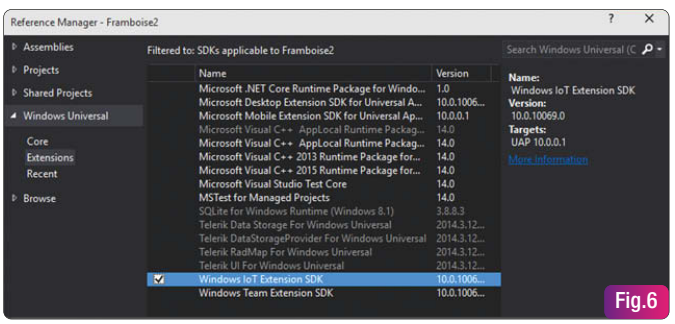


Fig.6

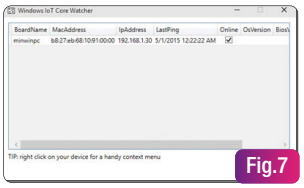


Fig.7

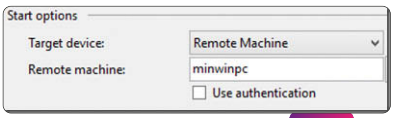


Fig.8

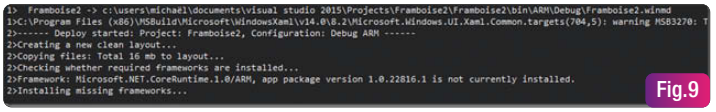


Fig.9

Le développeur fullstack existe-t-il ?

Depuis quelques mois, un nouveau « profil » de développeur apparaît régulièrement : le développeur fullstack. Pour lancer le troll, on peut dire qu'il fait tout, sait tout et s'occupe de tout. Mais la définition même du dév fullstack pose problème.

François Tonic

Notre sondage express « qu'est-ce qu'un développeur fullstack », mené sur juin et juillet, donne des résultats très marqués :

- 41 % répondent qu'il s'agit d'un dév qui fait tout et beaucoup trop,
- 19 % que c'est un superman du code,
- 31 % répondent : je ne sais pas (ce que c'est),
- et seulement 9 % disent être un dév fullstack.

Nous voyons bien l'ambiguïté du profil et sa perception réelle par les développeurs eux-mêmes. Au printemps dernier, Stackoverflow.com avait publié sa grande enquête développeur 2015. Surprise (?), 34 % se disaient être des dévs fullstack, un peu plus qu'en 2014.

Un (réel) problème de définition

Mais que signifie réellement être un dév fullstack ? Une des définitions possibles est :

- Mener des missions, des projets très variés (back-office, mobile, Cloud, multi-technologie, architecture Web),
- Il connaît les technologies actuelles, les principaux langages, avec compétences de base,
- Capacité de passer d'un langage à un autre, d'une plateforme à une autre dans la même journée,
- Il est touche-à-tout, il fait beaucoup de veille technologique, et s'autoforme.

Ce profil n'est pas forcément très courant en France que ce soit dans les offres d'emplois et les développeurs eux-mêmes. Il peut être vu comme un super développeur – intégrateur sur une dominante. Il sera très polyvalent, capable de s'adapter rapidement aux diverses situations. Bref, il pourrait être assimilé à un développeur polyvalent.

Tout fullstack qu'il est, le développeur aura tout de même des affinités pour certaines piles : mobile, Web, Cloud, desktop. Sur le mobile par exemple, il devra connaître les 3 plateformes et donc les 3 langages ou alors un environnement multi-plateformes, mais avec une solide expérience sur les différents systèmes. Il pourra être fullstack JavaScript, Web, mobile, PHP, etc.

Mais être fullstack Web, PHP, JS est-ce normal ? Un vif débat a eu lieu sur notre Facebook sur le sujet. Pour des intervenants : être fullstack PHP ou JS est un contresens. Le principe du fullstack est de ne pas maîtriser un langage, mais tous les principes, techniques et couches logicielles tout en étant agnostique sur le langage et la plateforme. Attention donc aux abus de langages (sic).

Un développeur a réagi avec raison : le développeur 2015 doit élargir sa palette de compétences, car le périmètre technique est vaste et s'agrandit sans cesse ! Le développeur devient un peu fullstack même inconsciemment. Mais il faut aussi savoir dit stop aux intitulés de postes de développeurs irréalistes où le recruteur veut tout avoir.

Il y a quelques semaines, OVH proposait sur son site une offre d'emploi : « développeur fullstack – datacenter / supply chain ».

Les compétences demandées étaient les suivantes :

- À l'aise en développement : langages de script back-end, framework Web (Angular), et bases de données (SQL/Mongo)
- Maîtrise de l'environnement Linux et outil de versionning (Git)
- Capacité autodidactique
- Capacité à travailler dans un environnement international, anglais opérationnel indispensable

Mais finalement, tout le monde peut y mettre dans ce profil tout ce qu'il veut. Au-delà du développement, on peut y voir des expertises sur les tests, l'assurance qualité, les besoins métiers, les interfaces, les bases de données, les réseaux, etc.

Question philosophique : le dév fullstack n'est-il pas la reconnaissance implicite que le développeur doit être indépendant de tout langage ou technologie (merci à J-M pour la réflexion).

À trop vouloir charger la barque...

... elle coule. Le développeur fullstack peut être cette métaphore. Oui un profil, mais il ne va pas se généraliser. Un développeur doit déjà maîtriser beaucoup de choses et les faire bien. À force de rajouter telle ou telle compétence, le risque de bâcler les tâches et des développements est réel. Le développeur ne peut pas se dupliquer et la journée est incompressible.

Il faut, selon nous, être très prudent sur les offres d'emploi avec une liste de compétence longue comme un log de serveur.

Le dév fullstack est déjà mort

Aux États-Unis, ce développeur surhumain est pour beaucoup déjà du passé. TechCrunch et Javaworld avaient argué en novembre 2014 qu'il devenait virtuellement impossible pour un seul développeur de programmer avec l'ensemble des piles (techniques actuelles). Et de nombreux développeurs-blogueurs ont enterré ce profil.

En France, la Force est troublée.

DES SALAIRES QUI NE SUIVENT PAS FORCÉMENT

Malgré les compétences du dév fullstack, ce n'est pas pour autant que le salaire va être forcément supérieur aux fourchettes moyennes.

Par exemple, nous avons vu un poste fullstack PHP, sur Paris, entre 35 /50 k€ selon le profil et l'expérience. C'est à dire, une fourchette standard d'un développeur PHP et avec une liste de technologies très longue (JS, JQuery, PHP, Zend, Symfony, Angular, Cordova, etc.). Pour un dév fullstack Ruby dans une startup, la rémunération de départ était de 35 000 €/an, selon expérience. Pour un développeur Ruby/Rails, c'est une proposition très basse.

Il y a aussi les recruteurs qui vont mettre un beau sticker « fullstack » pour charger la liste des compétences, mais au salaire constant, avec peu ou pas d'avantages. Soyez vigilants.

Monitoring d'applications en ligne : quelles solutions adopter ?

Le monitoring d'applications est peu abordé. Face à la montée en puissance du Cloud, c'est pourtant un sujet primordial. Lors de notre interview de Brice Cornet, CEO de Simple CRM (http://crm-pour-pme.fr), interview consacrée à son point de vue sur le langage JAVA, nous avions également abordé la question du monitoring. Simple CRM étant dans le Cloud depuis 10 ans, les réponses apportées sont riches en idées et méthodologies.

La Rédaction

PROGRAMMEZ : Comment monitorez-vous les serveurs Simple CRM?

Nous travaillons sur 3 niveaux. D'abord un monitoring classique de présence en ligne et de mesure de vitesse. Nous utilisons à la fois des solutions internes sous Centreon (https://www.centreon.com) et également des solutions externes comme StatusCake (https://www.statuscake.com), ServiceUptime (http://www.serviceuptime.com) et MonitorUS (http://www.monitor.us).

Ces systèmes externes permettent, pour un budget symbolique, d'effectuer du monitoring depuis l'Asie, l'Amérique du Nord et du Sud. Certains de nos clients français sont souvent en déplacement dans ces régions. Cela nous permet donc de mesurer leur niveau de confort lorsqu'ils travaillent depuis New-York ou Tokyo. De même, cela nous permet de monitorer nos serveurs présents aux USA et à Hong-Kong au travers d'un monitoring de proximité territoriale.

Ensuite, nous monitorons les accès disques, le CPU et la RAM. Pour cela nous utilisons à nouveau Centreon mais aussi Salt (http://saltstack.com/community/) qui nous permet d'avoir une vue en temp réel sur l'état des ressources. Pour ceux et celles que cela intéresse, il y a un tuto de prise en main de Salt sur mon blog personnel (http://www.ihaveto.be/2013/12/monitoring-des-cpu-de-plus-serveurs.html).

Enfin, on monitore les temps de réponse des serveurs, des DNS, la localisation géographique, les pages les plus rapides, les plus lentes, les pages les plus demandées, les durées des cessions, mais aussi des informations comme le type de device, l'âge ou le sexe supposés de nos utilisateurs. Pour ce faire, on utilise tout simplement Google Analytics.

PROGRAMMEZ: Google Analytics ? C'est plutôt surprenant !

Oui et non. C'est surprenant car peu d'entreprises le font, cependant c'est un outil parfait car il répond à un nombre important de questions.

Connaître les temps des réponses du serveur, la vitesse de chargement de tes pages et les performances du DNS, sont de précieux indicateurs de l'état de tes infrastructures.

La localisation géographique te montre où tu dois placer tes CDN.

Les pages les plus utilisées et les plus lentes t'indiquent les pages que tu dois absolument optimiser, tant du côté code que du côté requêtes en DB.

Les pages les plus rapides t'indiquent les bons élèves; les best practices de code à reproduire.

Enfin, des informations comme l'âge, le sexe ou le type de devices utilisés vont t'aider dans la construction de l'IHM, afin que de créer des interfaces réellement adaptées à tes clients.

PROGRAMMEZ: Il est étonnant de constater qu'un outil de marketing est finalement aussi une aide technique.

Le but est que tes clients restent clients, il est donc essentiel pour les codeurs de s'intéresser à des indices de mesure marketing.

Imaginons que tu as 1000 contrats d'abonnement à ton app mais que tu n'as que 100 cessions utilisateurs par jour. Cela t'indique clairement qu'il y a un soucis! Cela veut dire que ton marketing et ton département vente savent présenter et vendre le produit mais qu'au final, les utilisateurs ne sont pas du tout séduits par la réalité du software.

De même; imagine que tu viennes d'ajouter une super fonctionnalité et qu'après déploiement de ta super fonctionnalité, le temps passé à utiliser ton software baisse de 20%; cela veut dire que tes utilisateurs n'apprécient pas cette nouvelle fonctionnalité.

Google Analytics est un outil précieux d'auto-critique des codeurs. Ils peuvent voir, en temps réel si nécessaire, l'activité des utilisateurs. Les indices de valeur présents dans les tableaux de bord permettent de se poser les vraies questions et de trouver des pistes d'amélioration.

PROGRAMMEZ: Peut-on modifier Google Analytics pour l'adapter à des besoins spécifiques ?

Oui tu peux, mais si je peux me permettre un conseil, avant de réinventer la roue, apprends d'abord à la maîtriser.

Si certains de tes lecteurs ne connaissent pas l'outil, je les encourage à visionner une formation complète consacrée à Google Analytics, sur le blog marketing de Simple CRM: http://astuces-marketing-vente-management.blogspot.fr/2015/03/comment-bien-utiliser-google-analytics.html

Ensuite, il existe la possibilité de créer ses propres dashboard de monitoring, sur mesure, directement depuis l'interface de Google Analytics.

Nous avons trois tableaux de monitoring principaux dans la Google Analytics de Simple CRM. Ces trois tableaux sont tous des tableaux réalisés sur mesure via les outils présents dans l'interface Analytics. D'abord il y a la vue en temps réelle qui nous permet de visualiser l'activité des serveurs ainsi que le profil matériel et logiciel de nos utilisateurs, de même que les pages appelées et leurs localications géographiques (vous pouvez appliquer ce tableau à votre Google Analytics via ce lien : https://www.google.com/analytics/web/template?uid=Su_FEftcQbWH5v1Q-A3TuQ), ensuite on a un tableau qui mesure l'engagement des utilisateurs (lien : https://www.google.com/analytics/web/template?uid=NR6o-zE_RkqB1u5BT410Pw) et un tableau centré sur les performances serveurs (lien: https://www.google.com/analytics/web/template?uid=7O9UJbsPScyd7S-T1eapQQ).

On a aussi une vue spécifique au mobile, car il est important de suivre les marques et les OS les plus utilisés (https://www.google.com/analytics/web/template?uid=5zI67UlKR4qopxJJNck2_A) .
Si tu veux aller plus loin, tu peux bien entendu le faire. Il existe une bibliothèque en Java, une JS mais qui est toujours en version Beta, une .Net, une PHP et une en Python.
Tu peux donc lier tes données Google Analytics à ton système de monitoring central si tu le désires.

PROGRAMMEZ: En pratique, ça se passe comment ?
Tu as deux niveaux d'API: une API pour la gestion des comptes et une API pour la présentation des rapports. L'API de présentation des rapports n'est pas une seule et unique API mais un ensemble d'API, chaque API ayant un rôle précis. Tu as une API pour la création des rapports de base, une API plutôt orientée sur l'acquisition multi-canaux, une API réservée à la collecte des métas données et enfin l'API dédiée aux analyses en temps réel.
Enfin, par dessus tout cela, tu as un proxy spécifique appelé le Google Analytics superProxy.

PROGRAMMEZ: A quoi sert-il ?
Il sert principalement à rendre les données de Google Analytics publiques; de pouvoir les sortir de l'environnement Google et d'effectuer des conversions en temps réel des données aux formats CSV ou autres.
Si tu veux par exemple mettre dans ton graphique d'analyse Centreon, en parallèle des performances du CPU, le nombre de pages appelées, tu peux transiter via le proxy et l'API temps réel.

PROGRAMMEZ: Présenté comme cela, ça a l'air un peu lourd comme infrastructure.
Tu as totalement raison et en fait Google en est conscient, raison pour laquelle tout ceci est en train d'être modernisé au profit d'un nouveau système nommé Google Universal Analytics et qui va, à terme, remplacer le Google Analytics que l'on connait.

PROGRAMMEZ: Quelles sont les améliorations prévues ?
Le but de Google Universal Analytics est de ramener l'analyse au niveau du user via un système d'ID.
En clair, tu vas pouvoir voir non pas X cessions sur tablette Android et X sous Mac OS. Ce que tu vas voir que c'est que user ID 0456 s'est connecté à 10h05 sur une tablette Android et à 16h03 sous Mac OS. Tu verras ensuite que le lendemain, il s'est par exemple connecté avec un téléphone IOS. Les rapports seront beaucoup plus précis, c'est la première des améliorations. Il est à noter que c'est déjà accessible: tu peux migrer ton compte Analytics en Universal Analytics dès maintenant.
Enfin, le proxy va tirer sa révérence et les API vont être complètement changées.

PROGRAMMEZ: Quels conseils donnerais-tu à ceux qui veulent se lancer dans une customisation sur mesure d'Analytics ?
De lever le pied! La migration vers Universal a été prévue en 4 phases et nous sommes déjà au trois quart de la phase 3. Après cela, les anciens codes développés pour Google Analytics seront encore compatibles pendant 2 ans, puis ce sera la fermeture définitive des API actuelles.
Le meilleur conseil que je puisse donner actuellement est donc de voir si les dashboards personnalisés ne peuvent pas remplir la mission demandée, quitte à recadrer un peu la mission.
Ensuite, s'il est réellement nécessaire de coder, n'investissez pas trop de temps et donc d'argent sur le sujet car la durée de vie de votre code est plus que limitée.
Enfin, si vous voulez vraiment vous y mettre, testez d'abord l'explorateur de requêtes sur https://ga-dev-tools.appspot.com/explorer/ qui vous permettra de voir rapidement si vos besoins spécifiques entraineront ou non un dev long.
Pour terminer, je dirais que maintenant qu'ils savent que les données de Universal Analytics seront ramenées au niveau d'un ID, je conseille de réfléchir à comment valoriser ces données. Ce que propose Google avec cette mise à jour, c'est une masse d'informations façon Big Data. Il y a donc à mon avis un nouveau marché potentiel qui s'ouvre afin de valoriser réellements ces données. Si jusqu'à présent, coder des ponts Google Analytics était assez anecdotiques, les perspectives offertes par Universal Analytics sont immenses et les besoins des départements marketing et e-commerce face aux traitements et à l'exploitation de ces données le seront tout autant.

MariaDB MaxScale : le proxy dédié aux bases de données

Les bases de données relationnelles telles que MariaDB ou MySQL forment le noyau d'Internet et d'innombrables applications. Quand il s'agit de satisfaire aux exigences actuelles, les outils des systèmes de gestion de bases de données sont dépassés depuis bien longtemps. Après le clustering et la réplication maître-esclave, MariaDB tente désormais avec MaxScale de trouver une solution aux principaux problèmes des bases de données à l'ère du Big Data. Pour ce faire, les finnois suivent un concept concluant.

Véronique Loquet
Fondatrice de l'agence RP AL'X Communication.
Spécialiste de l'Open Source et de la sécurité des SI.
sur Twitter @vloquet

Les personnes qui développent des applications ou exploitent des sites Web utilisent généralement une base de données comme élément central. De nombreuses entreprises et développeurs misent ici sur des solutions open source, et ce pas uniquement pour des raisons de coûts. Ce n'est pas un hasard si MySQL et son fork MariaDB font partie des solutions de gestion de bases de données les plus fréquemment utilisées : elles sont gratuites tant que vous n'avez pas besoin de faire appel à l'assistance professionnelle. Dans la zone critique, l'ensemble des services indispensables à l'exploitation sûre et fiable d'une application basée sur ces derniers peuvent être activés en supplément. Néanmoins, les technologies de la base de données relationnelle et de la langue de la base de données SQL présentent des limites. Un problème fondamental qu'il devient de plus en plus urgent de résoudre au vu des importantes quantités de données à traiter est l'évolutivité. Deux méthodes se sont établies dans ce domaine, permettant un comportement de réponse satisfaisant et une bonne sécurité de fonctionnement des serveurs de bases de données : la grappe avec réplication maître-esclave et le sharding. Toutefois ces deux approches, qui sont fréquemment combinées, rendent très complexe l'administration de l'infrastructure des bases de données. Sans compter que le développement des applications est souvent plus compliqué, sachant que les développeurs doivent en partie tenir compte de la structure maître-esclave dans leurs applications. Dans le cas du sharding, la charge administrative augmente en outre considérablement lorsqu'il faut modifier un tableau. Le clustering et le sharding permettent certes d'obtenir une excellente disponibilité, tout en permettant à l'administrateur de garantir des temps de réponse exceptionnels, néanmoins la nécessité de maintenir les applications disponibles 24 heures sur 24 contrecarre de plus en plus cette organisation. Les maintenances planifiées au sein d'une infrastructure complexe entraînent systématiquement un travail conséquent pour pouvoir être menées en bonne et due forme. Les mises à jour et les mises à niveau doivent être entièrement testées au préalable, le cas échéant des ajustements de l'application ou du système de bases de données sont inévitables. On peut y remédier en procédant au découplage de la couche données et de la couche application, réduisant ainsi la complexité.

Un proxy spécifique aux bases de données

MariaDB a recours à une telle approche. Le dernier proxy présenté, du nom de MariaDB MaxScale, forme une couche d'abstraction entre l'application et les bases de données. Le concept d'un proxy dédié aux bases de données n'est pas tout à fait nouveau, on compte déjà quelques solutions de proxys sur le marché. Toutefois ces derniers se concentrent souvent sur certains sous-domaines tels que la réplication ou la répartition de charge. MariaDB va ici largement plus loin et dote MaxScale de nombreuses fonctions devant faciliter la tâche à l'administrateur et s'étendant au-delà des possibilités du proxy MySQL, déjà établi et très similaire. L'élément design essentiel de MaxScale, le serveur proxy, est totalement transparent pour les applications. Pour la couche application, MaxScale reproduit simplement une instance de base de données des plus normales. Cela signifie que les modifications apportées aux bases de données n'ont généralement aucune répercussion sur les applications ; par ailleurs la base de données peut évoluer indépendamment de l'application. Le travail des développeurs d'applications est alors considérablement simplifié. Mais les administrateurs des bases de données auront également la part belle. Contrairement à d'autres proxys DB, MaxScale traite les requêtes exactement de la même manière que le fait MariaDB ou MySQL, ce qui promet des avantages considérables en termes de vitesse.

Les fonctions du proxy MaxScale reposent sur cette transparence. Il sert d'abord de répartiteur de charge et veille à une répartition homogène des requêtes et des connexions. La particularité de MaxScale est qu'il est optimisé pour les bases de données. Contrairement à la répartition de charge générique, le proxy est en mesure d'analyser le trafic de données et par conséquent d'agir de manière contextuelle sur la base des requêtes SQL. Le proxy maîtrise aussi bien la répartition de charge sous forme de connexions que sous forme de requêtes. Dans le cas de la répartition de charge sous forme de connexions, le proxy ne vérifie pas le contenu des requêtes envoyées à la base de données. Le routage des accès en lecture pour la réplication maître-esclave repose exclusivement sur la connexion établie entre les applications et la base de données. On utilise ici Round-Robin comme algorithme

SHARDING

Le sharding, également appelé partitionnement horizontal, est une méthode permettant de réduire la charge de certains serveurs de bases de données. Pour ce faire, les blocs de données d'une relation sont répartis sur plusieurs tableaux. Chaque sous-tableau (tesson) est alors déposé dans une propre partition ou sur un propre serveur, pour former une instance de base de données physique autonome. D'un point de vue logique, tous les tessons pris ensemble forment une base de données. Si une base de données avec 1 TB est répartie sur quatre tessons, chacun des tessons comprend 250 GB. Ainsi la charge est répartie sur plusieurs instances, ce qui peut contribuer à améliorer de manière significative la performance du serveur et le comportement de réponse. L'inconvénient de cette méthode est sa complexité : sachant que les blocs de données sont divisés, il faudra effectuer une éventuelle modification des tableaux sur tous les tessons.

d'ordonnancement. Le proxy travaille de façon similaire en combinaison avec des grappes Galera. MaxScale travaille sans transition avec la grappe et peut piloter la répartition de charge sur plusieurs maîtres dans la grappe. D'autre part, le proxy surveille tous les nœuds de la grappe et transmet la requête uniquement aux nœuds entièrement synchronisés. Ainsi la disponibilité d'une grappe peut encore être augmentée : avec cet algorithme sous forme de connexions, il est possible de retirer de la grappe ou d'ajouter certains nœuds très facilement et de manière dynamique, ce qui constitue un véritable avantage pour les travaux de maintenance planifiés tels que les mises à jour. MaxScale garde également la connexion avec l'application en cas de défaillance de l'esclave, l'application n'a pas besoin d'établir de nouveau la liaison.

Un proxy qui allège le travail

Puisque les applications doivent être conçues explicitement pour l'interaction avec une réplication maître-esclave notamment pour les accès en écriture, il est intéressant de constater que MaxScale propose une solution de contournement pour les applications ne faisant pas la séparation entre connexion lecture et écriture.

Dans le cas de la répartition de charge sous forme de requêtes, l'application communique avec le proxy via une simple connexion. Le proxy analyse les requêtes et décide si elles sont transmises au maître ou à un esclave. Ainsi il n'est plus indispensable de développer des applications « sensibles à la réplication » pour l'évolutivité des accès en écriture.

La capacité de MaxScale d'analyser les requêtes présente d'autres avantages encore. Il est par exemple possible de contrôler les requêtes SQL avant leur transmission à la base de données, réduisant le risque d'une injection SQL. MaxScale permet en outre l'exploitation d'applications héritées pour des versions plus actuelles de MariaDB ou MySQL, sans devoir modifier le code de l'application. Le proxy tient compte des modifications de la syntaxe via un filtre et les remplace dans les requêtes et les réponses. La commande

```
CREATE TABLE… TYPE = InnoDB
```

des anciennes versions MySQL donne la variante actuelle :

```
CREATE TABLE… ENGINE = InnoDB
```

Il est ainsi possible d'exploiter des bases de données avec des dialectes SQL différents pour des applications existantes. Cette possibilité est particulièrement judicieuse lorsque les bases de données doivent être migrées. Grâce aux filtres, les serveurs DB peuvent passer à une nouvelle version de manière asynchrone sans temps d'arrêt. De plus, MaxScale permet de dupliquer des requêtes pour d'autres systèmes de bases de données, moteurs de stockage ou applications. Il est possible de cette manière d'envoyer des données à des tableaux également avec différents moteurs.

Un proxy extensible par modules

Les nombreuses fonctionnalités reposent sur l'architecture de plugin du proxy dédié aux bases de données. MaxScale a recours ici au concept modulaire déjà éprouvé avec MySQL et la base de données MariaDB. Le nombre de modules disponibles est jusqu'à présent encore gérable. Comme c'est le cas avec les protocoles clients : pour le lancement du produit, MaxScale travaille avec MariaDB Enterprise, MariaDB Enterprise Cluster, MariaDB 5.5, MariaDB 10 et Oracle MySQL. La répartition de charge implique l'utilisation de la grappe MariaDB Galera, de la réplication maître-esclave MariaDB ou de la réplication Oracle-serveur MySQL. Il est prévu de supporter à l'avenir d'autres systèmes de bases de données.

D'autres fonctions de base telles que l'authentification ou le monitoring sont également réalisées comme modules. Les modules sont généralement des objets partagés externes pouvant être chargés en cours de fonctionnement. Comme les API sont généralement connues dans le cas d'une solution open source, il est possible pour une moindre charge de travail d'adapter MaxScale aux propres besoins et à de nouveaux scénarios d'utilisation. On ne devrait donc pas tarder à voir une communauté active regroupée autour du proxy. Le fabricant a lui-même mentionné sur sa feuille de route certaines fonctions essentielles pour l'emploi notamment dans des environnements plus complexes. Il est par exemple prévu d'implémenter un module spécifique à Galera pour le routage sous forme de requêtes, ce qui doit permettre de réduire le risque d'interblocage en cas d'échecs de transactions. Un modèle de routage hiérarchique est également prévu qui permettra de combiner comme pipeline différentes méthodes de routage dans une suite constante.

MaxScale peut être exploité aussi bien dans des environnements virtuels que sur du matériel physique. Le proxy dédié aux bases de données supporte actuellement les dérivés de Linux suivants dans la variante 64 bits comme plateforme :

- CentOS 5, 6 et 7
- Red Hat Enterprise Linux 5 et 6
- Fedora 19 et 20
- Debian 6 et 7
- Ubuntu 12.04 LTS, 13.10 et 14.04

Une version doit être prochainement disponible pour OpenSuse 13.1. Comme MaxScale en tant que proxy dédié aux bases de données doit faire face à un débit de données très important, il fait d'après le fabricant un usage très intensif de la capacité I/O asynchrone du système Epoll de Linux. Il est donc recommandé d'exploiter MaxScale comme serveur dédié et de n'exécuter aucune autre charge de travail sur cette machine.

MaxScale est actuellement disponible sous la licence open source GPL v2. Les données binaires peuvent être téléchargées à l'adresse https://mariadb.com/user/login?destination=my_portal/download#maxscale, et le code source via Github https://github.com/mariadb-corporation/MaxScale. ⊞

ÉVOLUTIVITÉ HORIZONTALE ET VERTICALE

Les bases de données doivent être évolutives pour rester performantes en cas de hausse des volumes de données et des chiffres de fréquentation. Nous disposons pour ce faire de deux approches : une approche horizontale et une approche verticale. Dans le cas de l'évolutivité verticale, si nous simplifions les choses, la puissance de calcul du serveur de bases de données est augmentée. Un plus grand nombre de CPU et un espace de stockage plus important permettent de multiplier les volumes de données et les opérations de calcul. Cette méthode présente avant tout des limites économiques, sachant que les coûts de serveur augmentent considérablement avec la performance.

Dans le cas de l'évolutivité horizontale, la charge des opérations de bases de données est cette fois-ci répartie sur plusieurs serveurs, ce qui est nettement plus avantageux en termes de coûts. Les différents serveurs sont regroupés en grappes. Chaque grappe dispose d'un maître et d'un nombre donné d'esclaves. Les requêtes sont réparties de manière homogène sur toutes les instances via la répartition de charge. Le sharding est une forme d'évolutivité horizontale.

PHP 7 : la version des changements

Le 8 juin 2015, le langage PHP a fêté ses 20 ans d'existence et, pour cette année anniversaire, une nouvelle version majeure sera publiée. Nous vous proposons de découvrir les changements qui vous attendent, les impacts dans vos développements et à quel moment les utiliser.

Christophe Villeneuve
Consultant IT pour Neuros, 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, PHPTV..

Pascal Martin
Passionné de développement en général ainsi que de Web et de PHP en particulier, Pascal travaille aujourd'hui à Lyon chez TEA. Il publie régulièrement des articles techniques sur son blog et est l'auteur du livre Développer une Extension PHP.

L'idée de publier une nouvelle version majeure de notre langage de prédilection n'est pas nouvelle : on l'a vu avec le projet PHP 6, qui a finalement été mis en sommeil en 2010 puis abandonné officiellement en 2014 pour laisser la place à une autre version. Mais, cette année 2015 va plus loin, puisque PHP 7 va voir le jour : plusieurs préversions ont été mises en ligne cet été et la première version stable sortira d'ici quelques mois !

PHP 7 s'inscrit dans la logique des dernières versions de PHP 5, en constituant une évolution,et non pas une révolution. En effet, de nouvelles fonctionnalités nous sont proposées, une nette amélioration est à noter au niveau des performances et quelques efforts ont été faits pour améliorer la cohérence du langage, mais nos applications ne devraient demander que peu d'adaptations lors de la migration de PHP 5 à PHP 7.

À cette occasion et pour marquer cette année, nous vous dévoilons le logo validé de PHP7, réalisé par Vincent Pontier, aka El Roubio, le créateur de l'ElePHPant PHP.

De PHP 6 à PHP 7

Les développements ont été lancés sur une nouvelle version majeure, nommée "PHP 6", aux environs de 2005. PHP 6 aurait notamment dû intégrer un support Unicode complet: tout le moteur et toutes les fonctions fournies auraient été compatibles avec Unicode. Cela nous aurait bien sûr permis d'écrire du code, des noms de fonctions, de variables, avec n'importe quels caractères, mais, surtout, les chaînes de caractères de PHP n'auraient plus été considérées comme de simples tableaux d'octets, mais comme de véritables chaînes de **caractères**.

Pour plusieurs raisons (tâche herculéenne, puisqu'il fallait adapter chacune des fonctions du langage et de ses extensions, faible nombre de développeurs travaillant sur ces adaptations ...), le développement de PHP 6 a stagné de nombreux mois, avant d'être officiellement arrêté. En parallèle et par la suite, une bonne partie des évolutions prévues pour PHP 6 ont été intégrées aux différentes versions mineures de PHP 5, à commencer par les espaces de noms avec PHP 5.3.

Alors que PHP 5 approchait ses dix ans d'existence, l'idée d'une nouvelle version majeure a été relancée : elle permettrait un gros travail sur le moteur, la suppression de plusieurs fonctionnalités obsolètes, ainsi qu'une amélioration de cohérence sur plusieurs points du langage.

Lorsque s'est posée, l'an dernier, la question du nom à donner à cette version, plusieurs voix ont fait remarquer que, même si PHP 6 n'était jamais officiellement sortie, de nombreuses conférences avaient été données sur le sujet, il existait beaucoup d'articles sur Internet traitant de PHP 6 — des livres ont même été rédigés à propos de PHP 6 !

UN BREF HISTORIQUE : 20 ANS DE PHP !

La toute première version de PHP, à l'époque nommée "Personal Home Page Tools" et composée d'un ensemble de binaires CGI — écrits en C — développés par Rasmus Lerdorf, a été annoncée le 8 juin 1995. Et en 2015, PHP a donc eu vingt ans. PHP 3, publiée en juin 1998, s'est accompagnée d'un changement de nom, pour adopter celui, récursif, que nous utilisons encore aujourd'hui : "PHP: Hypertext Preprocessor". Peu après, Andi Gutmans et Zeev Suraski commencèrent la réécriture du moteur interne de PHP, appelé *Zend Engine*, qui servit de base à la version 4 de PHP.

PHP 5, un langage orienté professionnels

La sortie de PHP 5.0 en juillet 2004 s'est accompagnée d'une réécriture d'une partie du moteur de PHP (Zend Engine 2), ainsi que, point certainement le plus visible par les utilisateurs, de la mise en place d'un nouveau modèle objet nettement plus évolué que celui que proposait PHP 4. Chaque version mineure, depuis, a apporté son lot d'améliorations et d'évolutions, en tâchant de rester au maximum compatible avec les précédentes.

En se limitant aux points les plus marquants, on peut citer, pour PHP 5.3, l'arrivée des espaces de noms, la possibilité de créer des fonctions anonymes et des fermetures, le Late Static Binding, ou encore l'intégration d'extensions comme Intl (gestion d'internationalisation et de localisation), Fileinfo (obtention d'informations sur des fichiers) ou Phar (manipulation d'archives permettant de distribuer bibliothèques ou logiciels sous forme d'un seul fichier).

PHP 5.4 s'est accompagnée de la mise en place officielle d'un cycle de vie plus clair pour chaque version de PHP: une mise à jour mineure par an, avec un support de deux ans (plus un an pour correctifs de sécurité uniquement) pour chaque version.

C'est aussi avec PHP 5.4 que le langage s'est enrichi de fonctionnalités comme les traits (facilitant la réutilisation horizontale de code), de nombreuses améliorations au niveau de la syntaxe (écriture courte pour les tableaux, balise <?= indépendante du paramétrage de la directive short_open_tag, possibilité d'appeler des méthodes dès instanciation d'une classe comme (new MaClasse())->maMethode(), type-hint callable, ...), ou encore d'un serveur Web de test !

Enfin, point marquant illustrant sans doute un changement de mentalité autour de PHP, cette version a été l'occasion d'effectuer une passe de ménage se débarrassant de plusieurs boulets historique, comme la fonctionnalité register_globals ou le safe_mode, qui étaient marqués comme obsolètes depuis PHP 5.3; leur utilisation étant considérée comme une mauvaise pratique, souvent nuisible à la sécurité des applications, depuis des années. Alors qu'il avait fallu attendre plusieurs mois pour bénéficier d'une solution gratuite et open source de cache d'opcode (évitant de recompiler les scripts à chaque exécution, apportant ainsi un gain important en performances) pour la version précédente, PHP 5.5 a vu l'intégration de l'extension opcache à la distribution standard; PHP dispose donc, depuis cette version, d'un cache d'opcode intégré !

PHP 5.5 a également apporté un nombre conséquent de nouvelles fonctionnalités, comme les générateurs, le mot-clef finally pour la gestion d'exceptions, une API simplifiée de hachage de mots de passe, ainsi que plusieurs améliorations mineures visant à rendre la syntaxe du langage plus cohérente et celui-ci plus agréable à utiliser. Voici un exemple de générateur, qui permet d'afficher les multiples de 3 entre 0 et 15 :

```
<?php
$listeMultiples = multiplesDe(3, 15);
```

```
foreach ($listeMultiples as $i => $multiple) {
   echo "$i => $multiple\n";
}
?>
```

Le résultat obtenu en exécutant cette portion de code sera le suivant :

```
0 => 0
1 => 3
2 => 6
3 => 9
4 => 12
5 => 15
```

L'API simplifiée de hachage de mots de passe s'utilise quant à elle de la manière suivante :

```
<?php
$password = 'ProGr@mM3z';
$hash = password_hash($password, PASS
WORD_DEFAULT);
var_dump($hash);
?>
```

Et cette fois, la sortie obtenue sera la suivante :

```
string '$2y$10$EqctHvy2Zx2OqQUJs.0fDexsJR11
sdV8RVPtqN9iXLvqK/M9wBn7e' (length=60)
```

Continuant sur la logique de ménage entamée depuis plusieurs versions, c'est avec PHP 5.5 que l'extension mysql, peu maintenue depuis longtemps, et ne permettant pas de bénéficier des fonctionnalités avancées de MySQL, a été marquée comme obsolète (au profit de PDO ou de mysqli).
Le support actif de PHP 5.5 s'est terminé cet été : cette version ne recevra plus que des correctifs de sécurité, et ce jusqu'à juillet 2016.
La dernière version mineure de PHP 5, la seule actuellement encore activement supportée, est PHP 5.6 — qui a été publiée en août l'an dernier et sera donc supportée jusqu'en août l'an prochain (et jusqu'à 2017 pour les correctifs de sécurité). Encore une fois, cette nouvelle version mineure a apporté un ensemble d'améliorations. En termes de sécurité, de nombreuses évolutions ont été effectuées autour des composants SSL/TLS, avec également des options par défaut plus sécurisées. On peut aussi noter l'ajout de la fonction hash_equals() qui permet de comparer des chaînes de caractères sans fuite d'information. Pour ce qui est des fonctionnalités, on notera la mise en place d'une nouvelle syntaxe pour les fonctions variadiques ainsi que l'unpacking d'arguments, la possibilité d'utiliser des expressions scalaires lors de la définition de constantes, un nouvel opérateur d'exponentiation **, les instructions use function et use const, ou encore la possibilité pour les extensions PHP de surcharger des opérateurs (l'extension GMP en tire d'ailleurs profit). Par exemple, il est possible de définir une fonction variadique de la manière suivante :

```
function fonc01($a, $b, ... $params) {
   var_dump($a, $b, $params);
}
```

Les premiers paramètres passés à cette fonction lors de son appel seront stockés vers $a et $b, alors que les éventuels paramètres supplémentaires seront regroupés vers le tableau $params.
PHP 5.6 permet d'utiliser des scalaires lors de la définition de constantes ou de valeurs par défaut de paramètres :

```
class MaClasse {
   const FLAG = 1 << 8;
   const PI = 3.14;
   const VAL = 12 + self::FLAG / self::PI;

   public function maMethode($param =
60*10+self::FLAG) {
      var_dump($param);
   }
}
```

Bien entendu, les développeurs travaillant sur PHP sont conscients de l'importance des performances et y veillent à chaque version ! PHP 5.4 a ainsi fait de gros progrès par rapport à PHP 5.3 (lui-même plus rapide que les versions précédentes), PHP 5.5 a intégré un cache d'opcodes, et chaque version a des performances égales ou supérieures à la précédente, malgré les ajouts de fonctionnalités et améliorations. Pour garder trace des dates de sortie de chaque version de PHP et, surtout, des dates jusqu'auxquelles elles sont supportées, vous pouvez vous reporter à la page "Supported Versions", à l'adresse suivante : http://php.net/supported-versions.php

Pour éviter toute confusion entre "PHP 6" tel qu'il n'a jamais existé, et la prochaine version majeure du langage, il a finalement été décidé de l'appeler "PHP 7".

Le processus d'évolution de PHP

PHP est développé en suivant un processus communautaire et ouvert, où chacun est libre de proposer une évolution. Elle sera ensuite discutée et pourra aller jusqu'à un vote déterminant son inclusion ou non pour une prochaine version de PHP.
Plusieurs entreprises, parfois représentées par des membres vocaux de la communauté, participent activement au développement de PHP. Mais, dans tous les cas, elles n'ont pas un pouvoir de décision absolu sur l'évolution du langage : leurs contributions suivent les processus communautaires. Il est également à noter que, contrairement à d'autres langages, le développement PHP n'est pas dirigé par une personne unique qui jouerait le rôle de *dictateur* pouvant trancher en cas de conflit.
Les discussions autour de PHP et de son évolution se font par le biais d'une liste de discussion nommée "internals", active depuis 1997, qui voit passer chaque mois entre plusieurs centaines et quelques milliers de mails. Elle est publique et chacun peut consulter les échanges qui y ont lieu ainsi qu'y prendre part.
Chaque proposition d'évolution est formalisée par un document nommé "RFC", hébergé sur le wiki de php.net. Cette RFC est pendant quelques temps discutée sur la mailling-list, où chacun peut la commenter en vue de l'améliorer. Une fois mûre et au minimum deux semaines après sa rédaction, la RFC peut passer en phase de vote.
Le plus fréquemment, une RFC décrivant une évolution ou une nouveauté est accompagnée du code l'implémentant; PHP est développé en C. En effet, puisque chacun choisit de travailler sur les fonctionnalités qui l'intéressent, une proposition qui n'est pas accompagnée de l'implémentation correspondante n'aura que peu de chance d'être retenue. Mais, bien sûr, et c'est souvent le cas, une RFC peut être portée par plusieurs intervenants: un pour proposer une idée et l'autre pour la mettre en place.
Pour être acceptée, une RFC doit atteindre une majorité de 2 / 3 de "oui" dans le cas d'une modification du langage lui-même, ou de 50% + 1 "oui" dans le cas d'une fonctionnalité au périmètre plus réduit. Tous les développeurs ayant le droit de commiter du code sur le projet PHP peuvent voter, ainsi que quelques membres représentatifs de la communauté.
Gardons à l'esprit que PHP vise plusieurs types de publics, allant du développeur débutant au professionnel aguerri; ces publics n'ont pas tous les mêmes attentes. De même, chaque contributeur n'a pas nécessairement la même vision quant à l'évolution du langage.
En complément, chaque nouveauté peut impliquer de choisir entre stabilité et incompatibilité ascendante, ou entre complétude et accessibilité du langage ! Quoi qu'il en soit, PHP est un langage vivant, comme l'illustre bien le nombre de commits sur le projet : Fig.1.
Enfin, même si la majorité des RFCs portent sur des ajouts de fonctionnalités ou modifications du langage, le principe de "RFCs" est également utilisé pour décider des processus encadrant son évolution. Ainsi, il y a eu une RFC "PHP 6 ou PHP ?", une RFC "Timeline PHP 7", ou encore une RFC déterminant si une version PHP 5.7 allait ou non être développée avant de basculer à PHP 7.

Les nouveautés de PHP 7

PHP 7 apporte plus de 44 nouveautés et évolutions. Certaines étaient attendues depuis longtemps, ou ont été débattues pendant des années, alors que d'autres sont arrivées beaucoup plus naturellement.
Pour commencer, passons en revue quelques nouveautés ou évolutions qui méritent, d'après nous, d'être approfondies.

Exceptions dans le moteur

RFC : https://wiki.php.net/rfc/engine_exceptions_for_php7

Un des points faibles souvent reproché à PHP qui est tout particulièrement gênant dans le contexte de programmes à longue durée de vie comme des démons, est sa tendance à lever des erreurs, en particulier des erreurs fatales, face à certains types de problèmes. Ces erreurs fatales ne pouvaient pas, avec PHP 5, être interceptées, et mettaient fin à l'exécution du script. En réponse à cette critique, de nombreuses erreurs fatales ont été, avec PHP 7.0, transformées en exceptions, ce qui permet de les rattraper. Ces exceptions sont des instances de la classe Error ou d'une de ses classes filles. Par exemple, là où appeler une méthode sur une donnée null au lieu d'un objet générait précédemment une erreur fatale et mettait fin à l'exécution d'un programme, il est à présent possible d'intercepter cette erreur :

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

Voici un second exemple, où l'erreur survient lors de la compilation d'une portion de code que nous tentions d'exécuter avec la fonction eval(). L'erreur interceptée est, cette fois, une ParseError :

```
<?php
$php = <<<'PHP'
$a = 10 // parse error
printf("\$a = %d\n", $a);
PHP;

try {
   eval($php);
}
catch (ParseError $e) {
   // string(44) "syntax error, unexpected 'printf' (T_STRING)"
   var_dump($e->getMessage());
}
?>
```

Les erreurs de ces types ne seront généralement pas gérées par les développeurs de la même façon que la majorité des exceptions, et les classes Error ne s'intégrent donc pas à la hiérarchie standard des exceptions. En particulier, Error n'hérite pas de Exception, et les erreurs ne sont donc pas interceptées par un catch (Exception). À la place, l'équipe de développement de PHP a choisi de mettre en oeuvre une nouvelle hiérarchie, plus complète, basée sur une interface commune Throwable. Viennent ensuite les deux types Exception et Error, chacun sur sa propre branche de l'arbre. Les différentes erreurs héritent de la classe parente Error et les exceptions, comme avec PHP 5, continuent à hériter de la classe Exception :

```
interface Throwable
   Exception implements Throwable
      // Toutes les exceptions usuelles
   Error implements Throwable
      AssertionError extends Error
      ParseError extends Error
      TypeError extends Error
```

Scalar Type Hints et Return Types

RFC : https://wiki.php.net/rfc/scalar_type_hints_v5

RFC : https://wiki.php.net/rfc/return_types

PHP a un long historique de langage au typage souple: une chaîne de caractères contenant un nombre est automatiquement considérée comme étant un nombre, un entier ne valant pas 0, est considéré comme s'il était vrai, un entier trop grand pour tenir sur 32 ou 64 bits est transformé en nombre flottant. Mais, en parallèle, alors que PHP est depuis plus de dix ans employé sur des applications majeures, plusieurs voix s'élevaient régulièrement pour réclamer la mise en place d'un mécanisme de typage un peu plus strict. Répondant en partie à ces deux points de vue souvent diamétralement opposés, PHP 7.0 introduit la possibilité de définir des type-hints scalaires: le mécanisme de typage de paramètres existant pour les objets et tableaux et callables en PHP 5 est étendu à plusieurs types simples. Il devient, ainsi, possible de définir une fonction de la manière suivante, en spécifiant dès son écriture qu'elle attend en paramètres deux nombres entiers :

```
<?php
function add(int $a, int $b) {
   return $a + $b;
}
```

L'approche souple de PHP étant par défaut conservée, cette fonction peut être appelée en lui passant en paramètres des données qui sont entières ou qui peuvent être converties, sans perte d'information, en nombres entiers :

```
var_dump( add(10, 20) );  // int(30)
var_dump( add('10', '20') );  // int(30)
```

Par contre, et c'est la nouveauté, il n'est pas possible d'appeler cette fonction en lui passant en paramètre une donnée qui ne peut pas être convertie en entier (comme une chaîne de caractères contenant autre chose que la représentation textuelle d'un entier). Une exception de type TypeError (voir, plus haut, la nouvelle gestion d'erreurs) serait alors levée :

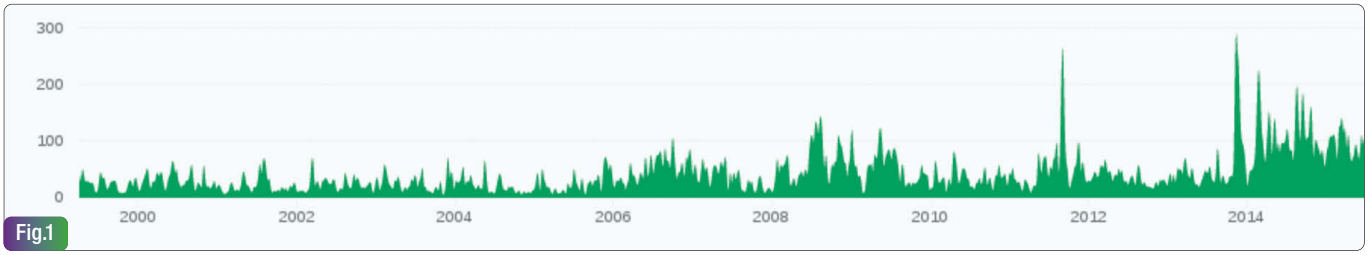


Fig.1

Nombre de commits sur le projet php-src, https://github.com/php/php-src/graphs/contributors

```
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 .../test-01.php on line 12"
   var_dump( $e->getMessage() );
}
```

En complément, il devient également possible de définir le type de retour d'une fonction. Par exemple, ici, pour une fonction censée retourner un nombre flottant :

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

Cette fonction, certes écrite un peu naïvement, retournant une chaîne de caractères au lieu d'un flottant, lèvera une erreur du même type que précédemment :

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

Si vous ne souhaitez pas que PHP applique de conversion et 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');
```

Ici, notre fonction est appelée avec en second paramètre une chaîne de caractères contenant un nombre entier (en mode de typage *souple*, elle serait donc automatiquement convertie en entier), mais, puisque nous avons activé le mode de typage *strict*, nous recevrons une erreur :

```
Argument 2 passed to add() must be of the type integer, string given
```

Pour rester compatible avec les versions précédentes, PHP 7.0 travaille par défaut en mode de typage *souple*. L'instruction declare() permettant d'activer le mode de typage strict doit être placée tout en haut d'un fichier et s'applique sur l'ensemble des appels de fonctions et de méthodes effectués depuis le fichier où elle est placée.
Ainsi, c'est celui qui appelle une fonction qui sait si son propre code est capable ou non de fonctionner en mode strict, et pas celui qui définit cette fonction: lui sait qu'il recevra une donnée du type qu'il a spécifié, mais n'a pas à savoir si elle a été obtenue via une conversion ou non.
Pour les types de retour, la directive declare() s'applique sur le code de la fonction : là aussi, sur le code écrit par celui qui positionne la directive.

Opérateur de comparaison <=>

RFC : https://wiki.php.net/rfc/combined-comparison-operator

De nombreuses fonctions PHP, en particulier pour effectuer des tris, se basent sur une fonction de comparaison de deux éléments, qui doit retourner -1 si le premier est plus petit que le second, 1 si le premier est plus grand que le second et 0 si les deux éléments sont égaux.
Pour faciliter l'écriture de ce type de fonction de comparaison, PHP 7.0 introduit un nouvel opérateur, dit opérateur « spaceship » en raison de sa forme, qui retourne une de ces trois valeurs selon les cas :

```
var_dump( 'a' <=> 'b' );  // int(-1)
var_dump( 'a' <=> 'a' );  // int(0)
var_dump( 'b' <=> 'a' ); // int(1)
```

Cet opérateur fonctionne également avec des tableaux :

```
$tab01 = [ 'a', 'b', 'c' ];
$tab02 = [ 'a', 'b', 'd' ];
var_dump( $tab01 <=> $tab02 );  // int(-1)
```

Null coalesce operator

RFC : https://wiki.php.net/rfc/isset_ternary

En restant sur les opérateurs, il nous arrive à tous d'utiliser l'opérateur ternaire ?: de la manière suivante, en le combinant à isset() pour déterminer si une donnée existe, et se rabattre sur une valeur par défaut dans le cas contraire :

```
$val = isset($_GET['val']) ? $_GET['val'] : 'default';
```

Pour éviter l'écriture redondante d'une partie de cette expression, PHP 7.0 introduit un nouvel opérateur ?? à deux opérandes : l'expression utilisant ?? sera évaluée à la valeur de la première opérande si elle existait, et à la valeur de la seconde dans le cas contraire.
Par exemple, avec une variable $nom qui n'existe pas (rappel : isset() retourne false pour une variable contenant null), la valeur par défaut passée en seconde opérande sera employée :

```
// Hello, World!
$nom = null;
printf("Hello, %s!\n", $nom ?? "World");
```

Par contre, si la variable existe, c'est la valeur qu'elle contient qui sera exploitée :

```
// Hello, Lecteur!
$nom = "Lecteur";
printf("Hello, %s!\n", $nom ?? "World");
```

Notons que ce nouvel opérateur utilise en interne le même mécanisme que isset() et qu'aucune notice n'est levée pour une variable ou donnée inexistante :

```
$tab = ['plop' => 'glop'];
// Ceci est un essai
echo $tab['test'] ?? "Ceci est un essai" . "\n";
```

Enfin, il est possible de chaîner plusieurs opérateurs ?? :

```
$a = null;
$b = null;
$c = 42;
var_dump( $a ?? $b ?? $c );   // int(42)
```

Anonymous classes

RFC : https://wiki.php.net/rfc/anonymous_classes

PHP 5.3 avait ajouté à PHP la notion de fonctions anonymes, qui, avec recul, se sont avérées être un réel succès ! Dans la suite logique, PHP 7.0

introduit la possibilité de définir et manipuler des classes anonymes.
Les classes définies par ce biais sont particulièrement pratiques lorsqu'elles n'ont pas besoin d'être documentées ou si elles ne sont utiles qu'une seule fois, lors de leur instanciation. Comme pour une fonction lambda, un objet obtenu depuis une classe anonyme ne peut pas être sérialisé, puisque PHP ne saurait pas comment le dé-sérialiser.
La syntaxe permettant de déclarer une classe anonyme est très fortement inspirée de la syntaxe habituellement utilisée, à cela prêt qu'aucun nom de classe n'est positionné :

```
$obj = new class ("Monde") {
   protected $qui;

   public function __construct($qui) {
      $this->qui = $qui;
   }

   public function hello() {
      printf("Hello, %s !\n", $this->qui);
   }
};

var_dump($obj);
$obj->hello();
```

La sortie obtenue en exécutant ces quelques lignes de code sera la suivante : notre objet est manipulable comme s'il était une instance d'une classe usuelle :

```
object(class@anonymous)#1 (1) {
  ["qui":protected]=>
  string(5) "Monde"
}
Hello, Monde !
```

Les classes anonymes peuvent bien sûr utiliser un Trait, hériter d'une autre classe existante, implémenter une interface. Cela pointe assez bien vers plusieurs cas d'usage que l'on peut d'ores et déjà imaginer, comme la définition d'un logger ou d'un observer instance de SplObserver utilisé conjointement avec SplSubject.

Group Use

RFC : https://wiki.php.net/rfc/group_use_declarations

Avec la généralisation de l'emploi des espaces de noms, un reproche que l'on entend parfois est que nos fichiers PHP commencent par un long bloc d'instructions use, pas toujours *drôle* à écrire ou maintenir :

```
use Mon\Espace\De\Nom\ClasseAa;
use Mon\Espace\De\Nom\ClasseBb;
use Mon\Espace\De\Nom\ClasseCc;
use Mon\Espace\De\Nom\Enfant\AutreClasse as ClassDd;
```

Pour répondre à cette remarque et diminuer la quantité de code à écrire pour travailler avec des classes placées dans différents espaces de noms, PHP 7.0 introduit une syntaxe raccourcie pour use, qui permet de regrouper les utilisations. Ainsi, les quatre lignes reproduites plus haut peuvent désormais être écrites comme suit :

```
use Mon\Espace\De\Nom\ {
   ClasseAa, ClasseBb, ClasseCc,
   Enfant\AutreClasse as ClassDd
};
```

Cette nouvelle syntaxe est aussi acceptée pour les fonctions et les constantes (qui peuvent être utilisées depuis PHP 5.6). Les quelques lignes suivantes pourraient donc être ré-écrites :

```
use function Mon\Nom\fonc01;
use const Mon\Nom\CONST01;
use Mon\Nom\Class01;
```

PHP 7.0 permettra d'écrire ceci :

```
use Mon\Nom\ {
   function fonc01,
   const CONST01,
   Class01
};
```

Exceptations

RFC : https://wiki.php.net/rfc/expectations

PHP fournit depuis très longtemps un mécanisme d'assertions, mais qui n'a jamais été réellement adopté par les développeurs, peut-être parce que ces assertions restaient évaluées y compris en production.
PHP 7.0 fait évoluer ce mécanisme: les assertions pourront désormais être activées en environnement de développement et être totalement ignorées en production.
Pour rappel, une assertion est une condition, encapsulée dans un appel à assert(), qui entrainera une levée d'avertissement si elle est évaluée à false :

```
$a = 20;
assert($a == 10, "Echec assertion !");
printf("a = %d\n", $a);
```

La sortie obtenue ici serait la suivante, puisque $a n'est bien évidemment pas égal à 10 :

```
Warning: assert(): Echec assertion !
```

En configurant à 1 la directive de configuration assert.exception, une exception peut être levée à la place de cet avertissement :

```
try {
   assert($a == 10);
}
catch (Error $e) {
   var_dump($e);
}
```

Cette exception s'intègre à la nouvelle gestion d'erreurs de PHP 7.0 vue plus haut, puisqu'il s'agit d'une AssertionError, qui hérite de Error.
Le code placé au sein de l'appel à assert() ne sera pas exécuté si la directive de configuration zend.assertions est à 0 ; et même pas du tout compilé si elle est à -1, ce qui est recommandé en production.

Amélioration de la cohérence du langage

Une nouvelle version majeure marque la possibilité de casser (de manière limitée et occasionnelle, pour ne pas bloquer la migration vers cette nouvelle version !) la compatibilité avec les versions précédentes. PHP 7.0 en

tire notamment profit pour revoir plusieurs points, en vue d'améliorer la cohérence du langage, au risque de supprimer quelques cas peu fréquemment utilisés.

Ainsi, PHP 7 ne supporte plus les balises alternatives comme <% ou <%= (https://wiki.php.net/rfc/remove_alternative_php_tags) ou ne permet plus d'avoir plusieurs cas default dans un switch (https://wiki.php.net/rfc/switch.default.multiple). Utiliser des constructeurs de classe façon PHP 4, où le constructeur est une méthode de même nom que la classe, lèvera désormais un avertissement indiquant qu'ils sont obsolètes (https://wiki.php.net/rfc/remove_php4_constructors).

Le niveau d'avertissements E_STRICT, qui n'était pas toujours bien compris, a été supprimé et les levées d'erreurs correspondantes ont été reclassifiées vers d'autres niveaux, comme E_DEPRECATED ou E_WARNING (https://wiki.php.net/rfc/reclassify_e_strict).

Suite à la mise en place des type-hints scalaires, et en pensant à l'avenir, où d'autres types pourraient éventuellement être reconnus, de nouveaux mots-clefs, comme true, false, numeric ou resource, ont été réservés et ne peuvent plus être utilisés comme noms de classe, interface ou trait (https://wiki.php.net/rfc/reserve_more_types_in_php_7 et https://wiki.php.net/rfc/reserve_even_more_types_in_php_7).

Enfin, pour terminer en citant encore que quelques modifications, list() et foreach() ont été retouchées pour des cas particuliers (https://wiki.php.net/rfc/fix_list_behavior_inconsistency et https://wiki.php.net/rfc/php7_foreach), le support des nombres hexadécimaux dans les chaînes de caractères a été supprimé (https://wiki.php.net/rfc/remove_hex_support_in_numeric_strings), la conversion tableau vers chaîne lève désormais une erreur E_RECOVERABLE_ERROR (https://wiki.php.net/rfc/array-to-string), l'output buffering reste désormais utilisable même si la connexion est annulée (https://wiki.php.net/rfc/continue_ob) et une passe globale a été effectuée sur la syntaxe de manipulation des variables, permettant l'emploi de nouvelles écritures qui n'étaient jusqu'à présent pas supportées (https://wiki.php.net/rfc/uniform_variable_syntax).

Avec cette dernière RFC, il devient par exemple possible d'utiliser des écritures comme celles-ci :

```
$foo()['bar']()
[$obj1, $obj2][0]->prop
[$obj, 'method']()
```

Et la sécurité ?

PHP 7.0 apporte plusieurs nouveautés visant à faciliter la sécurisation de nos applications, dont deux s'avèreront tout particulièrement utiles.

RFC : https://wiki.php.net/rfc/secure_unserialize

En premier lieu, il est important de se rappeler que lors de la dé-sérialisation d'un objet à l'aide de la fonction unserialize(), la méthode __wakeup() de la classe correspondante peut être appelée; ce qui peut mener à des injections ou à des exécutions involontaires de code.

Pour éviter toute exécution non souhaitée de code, PHP 7.0 permet de spécifier un second paramètre à unserialize(). Celui-ci est constitué d'un tableau associatif pouvant contenir une clef nommée allowed_classes.

Si la valeur associée est false, les objets éventuellement représentés par la chaîne sérialisée seront dé-sérialisés vers des instances de la classe __PHP_Incomplete_Class, qui ne porte aucune méthode qui risquerait d'être appelée à cette occasion :

```
$data = unserialize($serialized, ["allowed_classes" => false]);
```

Une alternative, dans le cas où vos chaînes sérialisées peuvent contenir des instances d'objets d'une liste de classes que vous maitrisez, est de spécifier cette liste de noms de classes :

```
$data = unserialize($serialized, [
  "allowed_classes" => [
    MaClasse::class,
    'MonAutreClasse',
  ]
]);
```

Utiliser la valeur true pour allowed_classes correspond au cas par défaut, où tout objet sera désérialisé vers une instance de sa classe si elle existe, comme en PHP 5.

À un autre niveau, une erreur souvent commise est d'utiliser les fonctions de génération de nombres aléatoires rand() et mt_rand() dans un contexte où un nombre réellement aléatoire, cryptographiquement fort, est nécessaire, ceci alors que ces fonctions ne sont pas réellement aléatoires, et ne sont pas en mesure de générer des nombres sûrs.

L'extension mcrypt n'étant pas toujours disponible et l'extension openssl n'étant pas toujours la plus intuitive à manipuler, PHP 7.0 s'enrichit de deux fonctions de génération de données aléatoires.

RFC : https://wiki.php.net/rfc/easy_userland_csprng

La première permet de générer un nombre entier :

```
// int(56)
var_dump( random_int(10, 100) );
```

Alors que la seconde retournera un nombre d'octets spécifié en paramètre (la représentation hexadécimale d'un nombre est deux fois plus longue que le nombre d'octets de celui-ci, ce qui explique les 24 caractères affichés ci-dessous) :

```
// string(24) "6da3bfc0238881a961b2d8e6"
var_dump( bin2hex(random_bytes(12)) );
```

Des améliorations internes majeures

PHP 7.0 comporte également des améliorations et des évolutions internes: des changements qui ne seront généralement pas visibles par les utilisateurs du langage, mais plus par ses développeurs ou ceux travaillant sur l'écriture d'extensions.

Pour commencer, le processus de compilation de scripts PHP a été revu: désormais, l'analyseur ne génère plus directement des opcodes et une structure intermédiaire: un arbre syntaxique abstrait ou AST, a été introduite (https://wiki.php.net/rfc/abstract_syntax_tree). Cette évolution apporte deux avantages majeurs: la maintenance des analyseurs et compilateurs est facilitée, et il devient possible de gérer certaines spécificités syntaxiques qui ne pouvaient auparavant pas l'être.

Le passage à un lexer sensible au contexte (https://wiki.php.net/rfc/context_sensitive_lexer) permet, en complément, une analyse plus fine des scripts. En conséquence, il devient possible d'utiliser certains mots-clefs, qui étaient auparavant réservés, comme noms de classes ou de méthodes. Par exemple, PHP 7 accepte l'écriture suivante :

```
class Query {
   public function where() { return $this; }
   public function and() { return $this; }
   public function or() { return $this; }
   public function not() { return $this; }
}

$obj = new Query();
$obj->where('condition')
```

```
->or('autre condition')
->and()->not('encore');
```

D'autres améliorations portent par exemple sur la gestion des nombres entiers et l'uniformisation de certains de leurs comportements, notamment entre différentes plateformes (https://wiki.php.net/rfc/integer_semantics et https://wiki.php.net/rfc/zpp_fail_on_overflow), ou sur l'uniformisation des comportements de classes internes (https://wiki.php.net/rfc/ internal_constructor_behaviour).

Et ce n'est pas terminé !

La liste de nouveautés et d'évolutions présentée plus haut n'est pas exhaustive, et il n'est pas possible de détailler toutes les améliorations de PHP 7.0 en un seul article.

Toutefois, en voici quelques-unes, qui nous semblent moins importantes que les précédentes, et dont l'impact sera sans aucun doute plus ciblé — à vous d'aller consulter la documentation de PHP pour en savoir plus :

- Une nouvelle fonction intdiv() permet d'effectuer des divisions entières (https://wiki.php.net/rfc/intdiv),
- La fonction json_encode() supporte une nouvelle option JSON_PRESERVE_ZERO_FRACTION s'il est nécessaire de conserver la partie flottante nulle d'un nombre (https://wiki.php.net/rfc/json_preserve_fractional_part),
- L'extension json, dont la licence n'était pas compatible avec celle de PHP, a été remplacée par l'extension jsond, sans impact réellement visible pour les utilisateurs (https://wiki.php.net/rfc/jsond),
- L'extension intl s'enrichit d'une nouvelle classe IntlChar (https://wiki.php.net/rfc/intl.char),
- Il est désormais possible de retourner une valeur depuis un générateur (https://wiki.php.net/rfc/generator-return-expressions) ainsi que de déléguer des opérations à un Traversable (https://wiki.php.net/rfc/generator-delegation).

Et une dernière pour terminer: nous pouvons désormais positionner des caractères Unicodes dans nos chaînes de caractères à l'aide de leur code. Par exemple :

```
// Un chat :
echo "Un chat : \u{1F638} \n";
```

N'oublions pas les performances !

La performance sera aussi un des axes de la prochaine version de PHP, notamment avec le travail effectué par le biais de la branche 'PHPNG', qui signifie PHP Next Generation. Cet ensemble d'améliorations est à la base de ce qui sera le successeur du moteur Zend Engine 2 que vous connaissez actuellement et c'est pour cela qu'une version majeure du langage a été lancée. A cette occasion, PHP 7 sera complètement 64 bits.

Ce changement est transparent pour les développeurs car il s'agit d'un nouveau cœur pour PHP. Son but a été une refonte des composants internes pour améliorer la performance à travers, notamment, la modification de plusieurs structures de données, et une refonte de la gestion des passages de paramètres aux fonctions de l'API.

A l'heure où cet article a été rédigé, les gains de performances sont au moins de 30 % à 50 % au niveau du cœur par rapport à la version de PHP 5.6 et peuvent aller jusqu'à 100% sur certaines applications. Des gains majeurs (ici aussi, de l'ordre de 30% sur certaines applications) en termes d'occupation mémoire sont également visibles pour PHP 7.0 par rapport à PHP 5.6 Fig.2.

Ces gains extrêmement importants (plus de 50% !) de performances se retrouvent également au niveau des frameworks sur lesquels nous sommes susceptibles de baser nos propres applications.

Les fonctions qui ne seront pas présentées

En suivant la logique du processus d'évolution de PHP, où chaque proposition est décrite par une RFC qui peut être acceptée ou rejetée, une partie des idées qui ont été évoquées ne figureront finalement pas dans PHP 7.0. Voici quelques points qui ont été évoqués plusieurs fois en public et ont, pour certains, soulevé des conversations intéressantes, mais qui n'ont finalement pas été retenus :

Les classes statiques (https://wiki.php.net/rfc/abstract_final_class), ne pouvant contenir que des méthodes statiques et ne pouvant pas être instanciées,

Les constructions loop… or (https://wiki.php.net/rfc/loop_or) où un bloc de code aurait été exécuté dans le cas où une boucle n'aurait jamais été jouée (dans l'idée de l'instruction else des boucles for de Twig, par exemple),

La possibilité d'utiliser des objets comme clefs de tableaux (https://wiki.php.net/rfc/objkey) : l'idée proposée n'allait pas jusqu'au bout du besoin et il a été préféré ne pas mettre en place de solution incomplète,

Des fonctions de transtypage levant des exceptions en cas de types incompatibles (https://wiki.php.net/rfc/safe_cas),

Ou encore la possibilité de ne pas spécifier les valeurs de paramètres optionnels lors d'appels de fonctions (https://wiki.php.net/rfc/skipparams).

Comment passer à PHP 7 ?

La compatibilité native avec les distributions Linux ne sera pas assurée dès la sortie de PHP 7.0: la plupart des distributions, notamment en versions stables, mettront plusieurs mois avant de fournir PHP 7.0 par défaut. Cela signifie que, pour ceux qui souhaiteront adopter PHP7.0 rapidement, il faudra certainement compiler PHP à la main ou se reposer sur des dépôts alternatifs, qui, eux, fourniront probablement PHP 7.0 très rapidement; certains le font déjà avec les versions de test !

Ensuite, pour commencer un nouveau projet, il semble raisonnable d'utiliser PHP 7 de suite: après tout, c'est la version majeure qui sera supportée pour les prochaines années, alors que PHP 5.x va rapidement ne plus l'être ! Par contre, pour un projet existant, il sera souvent plus facile de commencer par le migrer vers PHP 5.6 en premier lieu (si cette migration n'a pas déjà été faite) en respectant bien les compatibilités et les fonctionnalités dépréciées. Ainsi, vous n'aurez pas de difficulté majeure à passer ensuite vers PHP 7. Enfin, quelle que soit votre décision au niveau de la montée de version et du moment où vous l'effectuerez, il est important de lire les guides de migration fournis par php.net, aussi bien pour ce qui

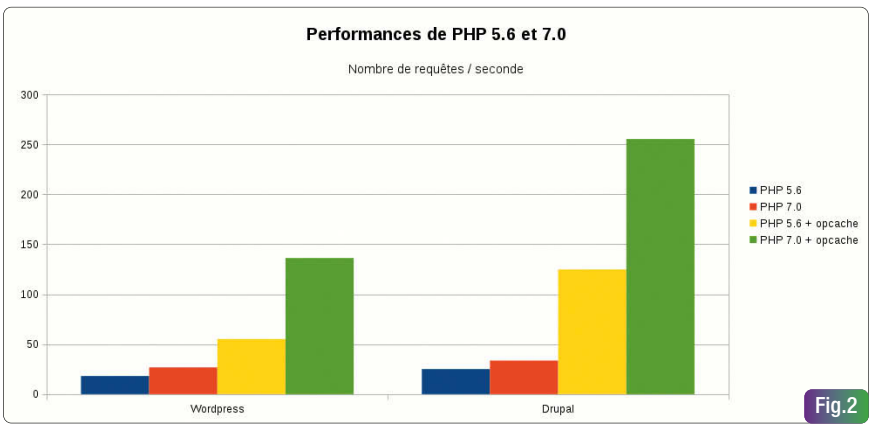


Fig.2

est des ajouts (fonctions, classes, mots-clés) que des suppressions, et des nouvelles erreurs possibles. En complément des nombreuses nouveautés et améliorations que nous avons découvertes plus haut; qui dit nouvelle version majeure dit également possibilité de "BC-break", c'est-à-dire de rupture de compatibilité. Ainsi, PHP 7.0 fait le *ménage* en supprimant de nombreuses fonctionnalités qui étaient marquées comme *obsolètes* (ou "deprecated", comme le soulignait l'avertissement qui accompagnait leur utilisation) depuis parfois plusieurs années. On peut notamment penser aux différents points suivants :

- Affectation par référence d'un object créé avec new,
- La fonction dl() pour fpm-fcgi,

set_magic_quotes_runtime() et magic_quotes_runtime() ; ainsi que quelques autres

- fonctions rarement utilisées,
- Les directives de configuration xsl.security_prefs et celles en lien avec l'encodage pour les extensions mbstring et iconv,
- Le paramètre $is_dst de mktime() et gmmktime(),
- Les commentaires commençant par # dans les fichiers .ini,
- Les noms de catégories sous forme de chaînes de caractères pour la fonction setlocale(),
- Les uploads "non-sûrs" de fichiers via curl,
- Ou encore, le modificateur /e pour preg_replace(),

En allant un peu plus loin, certaines extensions (ext/mssql, ext/sybase_ct), qui n'étaient plus maintenues, ont également été supprimées du cœur de PHP. Elles pourront, au besoin, trouver une nouvelle vie sur PECL.

Pour faciliter la migration vers PHP 7.0 du plus grand nombre, et afin qu'elle soit faite aussi rapidement que possible, les développeurs de PHP ont vivement encouragé tous ceux qui le peuvent à tester leurs applications sous cette nouvelle version, et ce, dès les premières versions de test publiées cet été.

Conscients que devoir compiler ces versions de test de PHP est un frein majeur à leur utilisation (sur machines de développement ou d'intégration uniquement, bien sûr, tant que la première version stable n'a pas été publiée !), plusieurs ont commencé à fournir des paquets pour différentes distributions. Par exemple, Zend en construit régulièrement pour Ubuntu et CentOS : http://php7.zend.com/ (en échange, à vous de signaler les bugs que vous rencontrerez éventuellement !).

Pour ce qui est de la migration des applications en elles-mêmes, les premiers développeurs à s'y être mis et à avoir publié leurs retours semblent ne pas avoir rencontré de réelles difficultés, ce qui est extrêmement encourageant et témoigne de la volonté, régulièrement répétée ces deux dernières années, de construire une nouvelle version majeure sans pour autant causer de catastrophe au niveau de la compatibilité du langage !

En conséquence, de nombreuses applications Open Source fonctionnent déjà, au moins pour la majeure partie de leur cœur, sous PHP 7: par exemple Drupal ou Wordpress.

Pour nos applications, la manière dont j'ai procédé pour chaque nouvelle version de PHP depuis PHP 5.3 est de l'installer sur le poste d'un développeur motivé, qui corrigera les premiers problèmes qui pourraient être bloquants (encore une fois, les guides de migration, listant l'ensemble des changements apportés par chaque version, sont d'une grande aide !) et s'assurera du passage de l'ensemble des tests automatisés du projet — cette étape prend en général de quelques heures à quelques semaines, selon la taille du projet, pendant lesquelles peu de temps sera finalement réellement consacré à la montée de version.

Cette première étape peut être facilitée par l'utilisation d'outils effectuant de l'analyse statique de code: certaines RFC étaient déjà accompagnées de scripts allant en ce sens et il y a fort à parier que d'autres seront publiés sur les prochains mois !

Ensuite, j'installe cette nouvelle version de PHP sur un serveur d'intégration continue, en parallèle à la version actuellement en place. Cela permet d'effectuer une montée de version sur tous les postes de développement, tout en profitant de la sécurité apportée par l'intégration continue pour s'assurer qu'aucune régression qui empêcherait le projet de fonctionner sur la version de PHP encore en production n'est introduite.

Cette seconde étape est l'occasion parfaite pour rejouer sur un environnement de test une partie du trafic réel de production ; des outils comme gor (https://github.com/buger/gor) peuvent grandement faciliter cette tâche. C'est aussi le moment rêvé pour faire monter en compétence l'ensemble des membres de l'équipe sur les nouveautés apportées par la nouvelle version qui arrivera bientôt en production, par exemple par le biais de commentaires en revues de code.

Enfin, une fois l'application validée, il ne reste plus qu'à effectuer la montée de version en production !

Conclusion

Le passage à une nouvelle version majeure permet aussi de se débarrasser de quelques poids et d'améliorer plusieurs points de syntaxe, en vue d'optimiser la cohérence globale du langage, parfois en supprimant des cas étranges et peu utilisés. Toutefois, et c'était une volonté importante dans l'esprit des développeurs de PHP, cette version n'est pas une révolution et la montée de migration depuis PHP 5 devrait se faire sans douleur pour de nombreuses applications — quitte à effectuer quelques retouches mineures ici et là.

En complément, n'oubliez pas que chaque version de PHP a une date de fin de support. PHP 5.6, notamment, ne sera activement supportée que jusqu'en août 2016 — avec en complément une année supplémentaire pour les correctifs liés à des failles de sécurité.

Et, pour terminer : travailler avec une version récente de PHP vous permet de bénéficier de meilleures performances, d'une sécurité accrue... Et est motivant pour les membres de votre équipe, présents ou futurs ! ⊞

L'avenir de PHP

Le processus menant à la publication de PHP 7.0 est tracé: les dernières RFC ciblant cette version ont été rédigées en mars cette année, laissant trois mois jusqu'à juin pour une phase de stabilisation et quelques derniers développements. Les premières versions de test ont été rendues publiques mi-juin et se sont succédées sur tout l'été, apportant à chaque fois de corrections de bogues.

Après trois mois pour cette phase de tests, PHP 7.0.0 devrait être annoncée courant octobre ou novembre, en fonction des retours : la date finale de sortie peut être décalée de quelques semaines, en vue d'arriver à une version la plus stable possible.

Ensuite, il semble fort probable que PHP 7.0 suive la logique en place depuis plusieurs années, avec une version corrective par mois : 7.0.1, puis 7.0.2 et ainsi de suite.

Même si, aujourd'hui, l'effort de développement se concentre sur la stabilisation de PHP 7 et la sortie de PHP 7.0, quelques RFCs commencent d'ores et déjà à cibler PHP 7.1. D'autres idées, comme les paramètres nommés ou la mise en place de compilation JIT, n'ont pas pu être intégrées à temps pour PHP 7.0 et pourraient donc arriver sur de futures versions mineures, comme PHP 7.1 ou 7.2.

En conséquence, on peut raisonnablement penser que le cycle de versions de PHP 7.x suivra celui mis en place depuis PHP 5.4, avec une nouvelle version mineure par an apportant à chaque fois un ensemble d'améliorations et d'évolutions.

Je pratique le C++

Partie 2/5

Nous vous proposons une série d'articles sur la pratique de C++ pour que vous puissiez tous vous y mettre. En C++, nous utilisons les classes pour définir nos propres types de données.

Christophe PICHAUD | .NET Rangers by Sogeti
Consultant sur les technologies Microsoft
christophepichaud@hotmail.com | www.windowscpp.net

Cet article va vous donner une idée du support de l'abstraction et de la gestion des ressources en C++. Comment définir des nouveaux types définis par l'utilisateur mais aussi les propriétés basiques, les techniques d'implémentation et les possibilités du langage pour les classes concrètes, les classes abstraites et les hiérarchies de classes? Le langage supporte le style de programmation orientée objet et de programmation générique (avec les templates).

La fonctionnalité principale du langage C++ est la classe. Une classe est un type défini par l'utilisateur qui permet de représenter un concept dans le code d'un programme.

N'importe quel design d'un programme possède des concepts, des idées, des entités, etc. que nous essayons de traduire en classes de telle manière que la lisibilité, la maintenance et l'évolution du programme en soit améliorée. Un programme est un ensemble de classes définies par l'utilisateur pour faire une tâche bien précise. Les librairies sont des ensembles de classes mis à disposition.

On distingue deux catégories de développeurs : celui qui fait les classes et celui qui les utilise. L'approche est complètement différente. La plupart des techniques de programmation évoquent comment designer et implémenter des types de classes. Il existe 3 sortes de classes : les classes concrètes, les classes abstraites et les classes dans les hiérarchies de classes.

Nous allons nous attacher à l'importance de l'abstraction des données, ce qui permet de séparer l'implémentation d'un objet des opérations que cet objet peut effectuer. Les objets peuvent être copiés, déplacés ou détruits. Il est même possible de définir ses propres opérateurs. Les idées fondamentales derrières les classes sont l'abstraction des données et l'encapsulation. L'abstraction des données est une technique de programmation qui se base sur la séparation de l'interface et de l'implémentation. L'interface d'une classe présente les opérations qu'un utilisateur de la classe peut exécuter. L'implémentation contient les données membres de la classe, le corps des fonctions présentes dans l'interface et toutes les fonctions internes à la classe pour faire son job. L'encapsulation permet la séparation de l'interface de classe de son implémentation. Une classe cache son implémentation, et les utilisateurs de cette classe n'ont pas accès (des fois) à son implémentation. C'est le mécanisme des librairies dans lequel on ne possède que le point .h (header) de la classe, et l'implémentation est fournie sous forme de binaire (.dll).

Une classe possède un header ; c'est un fichier d'entête .h et un corps, c'est un fichier cpp. Par convention, il est possible que les headers soient déposés dans un répertoire INCLUDE et les fichiers CPP dans un dossier SRC. Vous allez me dire « ouaip mais en java ou en C#, on met tout dans la classe et puis c'est tout ! ». Oui c'est vrai mais en C++ on ne fait pas comme ça.

Examinons une classe de log toute simple au travers de son header. La classe est préfixée du nom de l'entreprise pour laquelle elle a été faite (Cerius) en 1995. Cette classe utilise un type CString. C'est un type qui provient d'une librarie très connue faite par Microsoft qui se nomme MFC (Microsoft Foundation Classes).

```
class CerLog
{
public:
   CerLog(const CString& path);
   ~CerLog();

   BOOL Log(ORB_REQUETE * pReq);

private:
   CString m_path;      // Répertoire du log
   static long m_stCompt;
};
```

Cette classe est dans un fichier CerLog.h qui est dans le répertoire INCLUDE. Que remarque t-on ? Elle possède un constructeur qui porte le nom de la classe. Ce constructeur sera appelé dès la création d'un objet. Le ctor prend une chaîne de caractères en paramètre. C'est obligatoire ; il n'y a pas de ctor vide. La première partie de la classe est dans un bloc public mais sur la fin on voit du private, ce qui implique que ce sont des données membres que l'on ne pourra pas utiliser. C'est réservé à la classe. Donc, dans cette classe, on a un ctor, un dtor (destructeur) et une méthode Log, c'est tout. Ouvrons le code de cette classe CerLog :

```
#include "cerlog.h"

long CerLog::m_stCompt = 0;

CerLog::CerLog(const CString& path) : m_path(path)
{
   SECURITY_ATTRIBUTES  sa;
…
   ::CreateDirectory(m_path,&sa);
}

CerLog::~CerLog()
{}

BOOL CerLog::Log(ORB_REQUETE * pReq)
{
   HANDLE hFic;
…

   bDone = ::WriteFile(hFic,(LPSTR) pReq,
         (DWORD)pReq->usLgRequete,
         &dwBytesWritten,
         NULL);
   ::CloseHandle(hFic);
return TRUE;
}
```

On remarque que le membre static est initialisé et on trouve le code du ctor, du dtor et la méthode Log. Rentrons dans les détails du fonctionnement d'une classe.

Définir une fonction membre

Il est possible de définir la signature dans le header et de mettre l'implémentation dans le fichier cpp. Mais pour certains membres, on peut les définir dans le header. Ainsi les propriétés simples ont leur place dans le header. Voici comment on aurait designer la classes CerLog2 pour une utilisation simple :

```
void Discover_Class()
{
   CerLog2 log("c:\\temp");
   log.Log("hello the logger");
}


class CerLog2
{
public:
   CerLog2(const string &path) : m_path(path) {}
   ~CerLog2() {}

   string GetPath() const
   {
      return m_path;
   }

   void Log(string message)
   {
      //...
   }

private:
   string m_path;
};
```

Et là vous me dites : « mais ça ressemble à du Java ou du C# ! ». En effet, si on met tout le code dans le header... Mais généralement on propose une solution avec deux personnages différents. Il y a celui qui construit et designe la classe et il y a celui (ou celle) qui l'utilise.

Introduction au this et const

This représente un pointeur sur l'objet à l'intérieur d'une classe.

```
string GetInternalPath() const { return this->m_path;  }
```

Le fait de préciser que la fonction membre est const indique que l'on ne peut pas modifier les valeurs des données membres. Le this devient un this const. Les objets qui sont const et les références ou pointeurs vers des objets const ne peuvent appeler que des fonctions membres const. Il faut noter que lorsqu'on écrit une fonction membre à l'extérieur du header, il faut préciser le nom de la classe avec :: et respecter les paramètres de la fonction.

Définir une fonction qui retourne l'objet « this »

Ajoutons une méthode Merge dans la classe MyLogger pour merger un logger.

```
class MyLogger
{
…
public:
   MyLogger& Merge(const MyLogger &logger);
…
};
```

Voici l'implémentation :

```
MyLogger& MyLogger::Merge(const MyLogger &logger)
{
   m_path = logger.m_path;
   return *this;
}
```

Le logger positionne le path et retourne un objet this dans sa totalité avec le * sur this. C'est une référence qui est retournée.

Définir des fonctions non membres d'une classe

Des fois, il est nécessaire de créer des fonctions auxiliaires (read,write, print) qui travaillent avec notre classe. Dans ce cas, il faut définir la fonction dans le même header que la classe.

Dans le header :

```
void LogAMessage(string message);
```

Dans le cpp :

```
void LogAMessage(string message)
{
   MyLogger logger("c:\\temp");
   logger.Log(message);
}
```

Le constructeur

Par défaut, le compilateur définit un ctor qui ne fait rien. Chaque classe définit comment les objets sont initialisés. Les classes contrôlent l'initialisation des objets en définissant une ou plusieurs fonctions membres qui portent le nom de la classe et ce sont des constructeurs (ctor). Le ctor initialise les données membres d'un objet de classe. Un constructeur est exécuté lorsque l'objet d'une classe est créé. Les constructeurs n'ont pas de type de retour et peuvent être un bloc vide. Une classe peut avoir plusieurs constructeurs qui diffèrent par leurs paramètres. Un constructeur ne peut pas être marqué const. La classe MyLogger peut avoir les ctor suivants :

```
   MyLogger()
   {
      m_path = "c:\\temp";
   }

   MyLogger(string path)
   {
      m_path = path;
   }
```

Les ctor et l'initialisation par liste

Il est possible de fournir une liste au ctor. C'est le C++ 11 qui permet cela. Dans le header :

```
   MyLogger(string path, initializer_list<string> log_headers);
```

Dans le cpp:

```
MyLogger::MyLogger(string path, initializer_list<string> log_headers)
{
    m_path = path;
    for (auto it = log_headers.begin(); it != log_headers.end(); ++it)
    {
        Log(*it);
    }
}
```

Et voici comment utiliser ce ctor :

```
    MyLogger log4("c:\\temp", { "begin log", "1 juin 2015", "Application Totor" });
    log4.Log("the logger log4");
```

Contrôle d'accès et encapsulation

Lorsque nous définissons une interface pour notre classe, rien ne force l'utilisateur à respecter les appels dans le bon ordre ou le choix des méthodes. C'est la raison pour laquelle nous cachons l'implémentation dans des blocs private. Le contrôle d'accès garantit l'encapsulation. Les membres définis après public sont accessibles dans toutes les parties du programme. Les membres publics définissent l'interface de la classe. Les membres définis après private sont accessibles aux fonctions membres de la classe, mais ne sont pas accessibles au code qui utilise la classe. Les sections private encapsulent (cache) l'implémentation.

Class ou struct, il faut choisir

Class et struct ont le même sens si ce n'est que dans struct par défaut, tout est public, et que dans class, par défaut, tout est privé. Mais c'est la même chose.

Les fonctions ou classes friend (amies)

Reprenons la classe MyLogger et ajoutons-lui un membre private pour la sécurité (un exemple) :

```
class MyLogger
{
…
private:
    string m_path;
    SECURITY_ATTRIBUTES sa;
};
```

Et définissons la fonction LogAsAdministrator :

```
void LogAsAdministrator(string message)
{
    MyLogger logger("c:\\temp");
    // fake function :)
    logger.m_sa = ::CreateRestrictedToken(Windows::Administrator);
    logger.Log(message);
}
```

Cette fonction doit accéder au membre private m_sa qui est le jeton de sécurité. Problème, cette fonction n'est pas dans la classe. Donc, il faut la déclarer en friend (amie) et ainsi elle aura le droit d'accéder à tous les membres de la classe. Magique !

```
class MyLogger
{
    friend void LogAsAdministrator(string message);

…
```

La mécanique est la même pour les classes friend.
Je vais ajouter une classe qui fournit le privilège fictif administrator.
Dans le header de MyLogger :

```
    void LogAsAdministrator(const string & message);
private:
    CSecurityHelper m_sec;
};
```

Dans le cpp de MyLogger:

```
void MyLogger::LogAsAdministrator(const string & message)
{
    m_sec.m_sa = m_sa;
    m_sec.EnableAdministratorMode();
    Log(message);
}
```

La classe MyLogger au travers sa méthode LogAsAdministrator va renseigner un membre private de CSecurityHelper qui est m_sa en lui fournissant le sien pour demander une élévation de privilège.
Pour pouvoir accéder au membre privé, il faut que la classe MyLogger soit friend de CSecurityHelper dont voici le header :

```
#pragma once

class CSecurityHelper
{
    friend class MyLogger;

public:
    CSecurityHelper() {}
    ~CSecurityHelper() {}

    void EnableAdministratorMode();

private:
    SECURITY_ATTRIBUTES m_sa;
};
```

Voici le cpp :

```
#include "stdafx.h"
#include "MyLogger.h"
#include "SecurityHelper.h"


void CSecurityHelper::EnableAdministratorMode()
{
    // use m_sa
    // fake function :)
    m_sa.lpSecurityDescriptor = NULL; // ::CreateRestrictedToken(Windows::Administrator);

}
```

Pour que cela compile, il faut faire un #include de MyLogger.h pour que le compilateur connaisse la définition de la classe friend.

Les typedef dans les classes

Pour définir des types utilisateurs, on utilise parfois le typedef à l'intérieur d'une classe. Cela rend la classe plus lisible.
Reprenons la classe MyLogger qui possède des entêtes avant de logger, et que nous allons stocker dans un vector<string>.
Nous pouvons stocker le vector en private et déclarer les itérateurs comme typedef avec un nom plus simple...
Voici à quoi cela ressemble :

```
class MyLogger
{
...
public:
   typedef vector<string> HEADERS;
   typedef vector<string>::const_iterator CIT;

private:
   HEADERS m_headers;
```

Le typedef est là pour vous aider à rendre les types plus lisibles.

La résolution des noms

Le compilateur passe son temps à chercher les noms de fonctions qui matchent. De temps en temps dans le code on trouvera une ligne de code comme ::WriteFile() ; cela veut dire que le scope recherché est global et que cela ne se trouve pas dans la classe dans laquelle on utilise cette fonction.

Les membres static

Une classe peut avoir des membres static mais ils doivent être initialisés explicitement dans le fichier cpp. Exemple : dans le fichier header:

```
class NewLogger
{
public:
   static void Log(const string& message);
   static string m_path;
};
```

Dans le cpp :

```
string NewLogger::m_path = "c:\\temp";

void NewLogger::Log(const string& message)
{
   cout << message << endl;
}
```

Dans le programme qui l'utilise :

```
   NewLogger::Log("here is a static logger");
```

Il n'y a rien de compliqué.

La surcharge des opérateurs

Il est possible de surcharger tous les opérateurs de ++ en passant par -> en passant par [] ou ==.

```
bool operator==(const MyLogger &left, const MyLogger &right)
{
   return left.GetPath() == right.GetPath();
}
```

Introduction à la programmation orientée objet

Les idées clés dans la programmation orientée objet sont l'abstraction de données, l'héritage et le binding dynamic. Avec l'abstraction de données, on peut définir des classes qui ont une séparation entre l'interface et leur implémentation. Au travers de l'héritage, on peut définir des classes qui forment un modèle de relation avec des types similaires. Avec le binding dynamic, on peut utiliser des objets avec des types dont on ignore les différences par rapport à la classe de base. Prenons un exemple simple qui explique l'héritage, les classes abstraites et les fonctions virtuelles. Il y a la classe abstraite Animal et deux classes dérivées que sont Cat et Dog :

```
class Animal
{
public:
   Animal() {}
   virtual ~Animal() {}
public:
   virtual void Eat() = 0;
   virtual string Type() { return "Animal"; }
};

class Cat : public Animal
{
   virtual void Eat()
   {
      cout << "whyskas croquettes for Cat" << endl;
   }
   virtual string Type() { return "Cat"; }
};

class Dog : public Animal
{
public:
   virtual void Eat()
   {
      cout << "whyskas croquettes for Dog" << endl;
   }
   virtual string Type() { return "Dog"; }
};
```

J'ai volontairement tout mis dans le header pour faire plus concis. Voici ce qu'il faut retenir : la classe Animal est une classe abstraite car elle contient la méthode Eat() qui est virtuelle pure (=0) ; ça veut dire que toute classe qui hérite de Animal a l'obligation de redéfinir la méthode Eat. Le destructeur est annoté virtual : c'est obligatoire pour les dtor dans les classes mères et dérivées.

```
   Animal * ptrAnimal;
   Cat c1;
   Dog d1;
   // Pointe sur le Cat
   ptrAnimal = &c1;
   cout << ptrAnimal->Type() << endl;
   ptrAnimal->Eat();
   // Pointe sur le Dog
   ptrAnimal = &d1;
```

```
cout << ptrAnimal->Type() << endl;
ptrAnimal->Eat();
```

Ce petit bout de code montre comment faire un binding avec un pointeur sur la classe abstraite. On n'a pas le droit de déclarer un type Animal, car c'est une classe abstraite ; en revanche, on a le droit de s'en servir comme pointeur et de pointer sur des types enfants. On pointe sur un Cat puis sur un Dog, et les méthodes virtuelles sont appelées comme par magie.

Le concept clé : le polymorphisme en C++

L'idée clé derrière l'OOP est le polymorphisme. Ce mot est dérivé d'un mot grec qui veut dire plusieurs formes. On parle des types liés à l'héritage comme des types polymorphiques parce que nous pouvons utiliser plusieurs formes de ces types tout en ignorant les différences entre eux. Quand on appelle une fonction dans une classe de base au travers d'une référence ou d'un pointeur de la classe de base, on ne connait pas le type de l'objet sur lequel ce membre est appelé. L'objet peut être un objet de la classe de base ou un objet de la classe dérivée. Si la fonction est virtuelle, alors la décision de savoir quelle fonction va s'exécuter est repoussée à l'exécution. La version de la fonction virtuelle qui tourne est celle définie par le type d'objet avec lequel la référence est liée ou pour un pointeur sur le type d'objet pointé. D'un autre côté, les appels à des fonctions non virtuelles sont déterminés à la compilation.

Contrôle d'accès et héritage

Comme dans une classe pour cacher ses propres membres, chaque classe contrôle si ses membres sont accessibles à une classe dérivée. Une classes utilise protected pour les membres qu'elle veut accessibles à ses classes dérivées mais veut protéger depuis un accès général. L'accès protected se positionne entre le private et le public. Comme private, les membres protected sont inaccessibles aux utilisateurs de cette classe. Comme public, les membres protected sont accessibles aux membres et friends des classes dérivées de cette classe. Un membre d'une classe dérivée ou friend peut accéder aux membres protected de la classe de base seulement au travers d'un objet dérivé. La classe dérivée n'a aucun accès spécial sur les membres protected des objets de la classe de base.

Le ctor par copie

Le ctor par copie prend en paramètre une référence const sur l'objet.

```
class Foo {
public:
  Foo();          // constructeur par défaut
  Foo(const Foo&);  // constructeur de copie
  // ...
};
```

Si nous ne définissons pas de constructeur par copie, le compilateur en fournit un pour nous. Le type de chaque membre détermine comment le membre est copié : les membres de classe sont copiés par le ctor de copie de cette classe.

Initialisation par copie

Nous pouvons maintenant déterminer la différence entre l'initialisation directe et l'initialisation par copie.

```
string dots(10, '.');          // directe initialisation
string s(dots);                // directe initialisation
string s2 = dots;              // copie initialisation
string null_book = "9-999-99999-9"; // copie initialisation
string nines = string(100, '9');   // copie initialisation
```

Lorsque nous utilisons l'initialisation directe, nous demandons au compilateur de choisir un constructeur qui matche les paramètres que nous fournissons. Quand nous utilisons l'initialisation par copie, nous demandons au compilateur de copier l'opérande de droite dans l'objet à créer. L'initialisation par copie utilise le constructeur de copie.

L'opérateur de copie

Comme une classe contrôle comment les objets de cette classe sont initialisés, elle contrôle aussi comme les objets de cette classes sont assignés (=).

```
Book ref, primer ;
ref = primer ; // utilise l'opérateur de copie de Book
```

Comme avec le ctor par copie, le compilateur fournit un opérateur de copie si la classe ne définit pas le sien.
L'opérateur de copie prend en argument le même type que la classe :

```
class Foo {
public:
  Foo& operator=(const Foo&); // opérateur d'assignement (copie)
  // ...
};
```

Les opérateurs d'assignement doivent retourner une référence sur un type. Les classes qui ont besoin d'un destructeur ont besoin d'un ctor de copy et d'un opérateur d'assignement. C'est une règle en C++ qui permet de déterminer si on a besoin d'un opérateur de copie et d'assignation ; avant tout, on détermine si la classe a besoin d'un destructeur. Si on est sur qu'il faut un destructeur, alors c'est sur et certain qu'elle nécessite un ctor de copie et un opérateur d'assignement.

Conclusion

En C++, la classe est l'inséparable alliée de la programmation orientée objet. Cet article se concentre sur la manière d'expérimenter simplement les fonctionnalités offertes par une classe. Il y a encore beaucoup de choses à appréhender comme la sémantique de déplacement, la redéfinition des opérateurs, les fonctions virtuelles. Avec cet article, vous avez les bases pour faire votre propre construction.
Créez des classes, assemblez-les et n'oubliez pas : il y a deux casquettes, celui qui conçoit la classe et celui qu'il l'utilise. Il est bien plus simple d'être utilisateur que concepteur de classes. La maîtrise des principes de la programmation objet passe par de l'entraînement et de l'expérience. En C++, il n'y a jamais (ou presque) de classes deprecated. Donc une fois créée, la classe est là et pour longtemps. Prenez le framework de classes des MFC ; élaboré vers 1990, ce framework ne cesse de grossir et d'évoluer. Quand vous faites du C++, vous avez en tête que le code marche pour longtemps. C'est un exercice que l'on ne rencontre pas avec les langages dits modernes comme Java ou C#. En C++, on garde tout. La pyramide grossit mais on ne jette rien. Regardez vers le monde Linux : GTK, Kde, Xfce, toutes ces librairies évoluent depuis des années et sont toujours utilisées. Regardez QT, il ne s'est jamais aussi bien porté. Les ISV adorent QT car il permet de faire des GUI multiplateformes de folie. Allez, lancez-vous !

La 3D sur le Web avec BabylonJS

Les versions actuelles des navigateurs Web supportent désormais la spécification en cours d'HTML5. Finie la guerre des standards : quand une fonctionnalité est implémentée, elle fonctionne exactement de la même façon dans IE, Firefox, Chrome, Safari, Opéra, que ce soit sur un PC ou sur un smartphone. Ceci est une bonne chose pour les développeurs qui souhaitent voir leur application toucher le maximum d'utilisateurs.

Jérôme Bousquié
est ingénieur de recherche à l'IUT de Rodez et contributeur au projet BabylonJS.

Cependant, jusqu'à l'avènement de HTML5, la représentation graphique était loin d'être la panacée dans un navigateur. HTML5 propose maintenant un élément du DOM nommé *canvas* et dédié à l'insertion d'images pouvant être dynamiquement élaborées par le code exécuté dans une page Web. Cet élément *canvas* fournit au développeur deux contextes de programmation Javascript différents: *2D* et *WebGL*. Il est à noter que les appellations *2D* ou *WebGL* ne caractérisent pas en elles-mêmes le fait de faire ou non de la 3D. Tant que nous ne disposons pas d'écrans holographiques, nous affichons bien toujours une image en deux dimensions uniquement, même lorsqu'on parle de 3D. Le contexte *2D* signifie ici que l'environnement de programmation donne accès au développeur directement au système de coordonnées 2D (abscisse, ordonnée) à l'intérieur du *canvas*. C'est donc au travers de cet élément *canvas* que nous allons pouvoir afficher des images dites « en trois dimensions » et ceci, nativement, sans l'installation d'un plug-in de tierce partie et en n'utilisant que Javascript.

Pourquoi faire de la 3D dans le navigateur ?

Pour mille raisons qu'il serait impossible de toutes détailler ici. S'il fallait n'en donner que quelques-unes, on pourrait commencer par citer les mêmes raisons qui font qu'on utilise la couleur plutôt que le monochrome, qu'on utilise des interfaces graphiques plutôt qu'une simple console en ligne de commande, qu'on utilise des contenus multimédias plutôt que du simple texte, etc. En d'autres mots parce qu'il s'agît avant tout d'une amélioration de l'expérience utilisateur.

Bien entendu, cela ne se limite pas, loin de là, à de simples considérations de confort d'utilisation. En effet, disposer d'une dimension supplémentaire, la profondeur, fournit un apport essentiel dans les domaines de l'architecture, de l'imagerie médicale, de la conception mécanique, ou même, de la simple représentation de données numériques (charts, etc), sans parler évidemment du caractère immersif pour le monde des jeux vidéo.

Les techniques de la 3D

Les données

Pour représenter une vue 3D, on manipule un ensemble de concepts, toujours les mêmes :

- Une scène qui est le monde conteneur de tout ce qui sera potentiellement visible dans la vue,
- Une ou plusieurs sources de lumières (*lights*) qui vont éclairer les objets de la scène,
- Un ou plusieurs objets 3D, appelés encore maillages (*mesh*), qui représentent des volumes dans la scène,
- Un point de vue sur cette scène, nommé caméra.

Un mesh est constitué d'un ensemble de facettes (*faces*) contiguës. Chacune de ces facettes est simplement un triangle ayant pour sommets (*vertex, vertices au pluriel*) trois points dans l'espace (*vectors*), trois points suffisant à déterminer un plan. De cette manière on peut représenter n'importe quel volume, y compris une sphère : imaginez-vous une boule disco couverte de centaines de minuscules triangles plans réfléchissants.

Récapitulatif : un *vertex* est un sommet, il nécessite donc trois nombres flottants (*floats)* pour ses coordonnées x, y, z dans l'espace. Il faut trois sommets pour constituer une facette. Il faut des centaines ou des milliers de facettes pour représenter un unique *mesh*. Une scène peut contenir des dizaines, des centaines ou des milliers de meshes.

Ceci commence à donner une idée du volume de données numériques à manipuler dans le code. Mais nous sommes pourtant loin du compte. En effet, la scène étant éclairée, il faut calculer et associer à chaque vertex de chaque mesh un vecteur, nommé normale (*normal*), soit trois *floats x, y, z* de plus, définissant dans quel sens la lumière va se réfléchir en ce point sur le mesh. Par ailleurs, des textures (*texture*), c'est à dire des images 2D pouvant provenir de fichiers jpg ou png, sont généralement appliquées sur certaines faces du mesh pour l'habiller et lui donner par exemple l'aspect d'un matériau particulier. Ceci nécessite donc d'associer à chaque vertex une paire de coordonnées supplémentaires *u, v* dans le plan de l'image utilisée pour la texture, afin de définir en quels endroits cette dernière doit se « plier » ou « s'enrouler » sur la surface du mesh.

On comprend maintenant que le volume de données numériques à traiter risque d'être assez conséquent.

Les traitements

Nous savons maintenant de quelles données nous avons besoin pour décrire un ou plusieurs volumes dans une scène, à savoir essentiellement des coordonnées de sommets, et des relations entre ces sommets pour définir des faces. Nous n'avons donc jusqu'à présent défini que des relations entre des constituants d'un même volume dans un référentiel local à ce volume nommé *Model*. Cela ne suffit évidemment pas. Il nous faut en effet placer tous ces différents volumes les uns par rapport aux autres dans un référentiel commun, celui de la scène, appelé *World.* Il faut aussi donner à chacun une orientation dans cette scène, orientation définie par trois valeurs d'angle de rotation autour des axes *X, Y, Z* du *World*. Il faut enfin leur donner une échelle, c'est à dire un ratio d'étirement sur les axes *x, y , z* du *Model* (ce qui permet d'étirer un cube vers la forme d'un pavé par exemple). Ces trois opérations successives, mise à l'échelle, rotation, et translation, font donc passer le mesh de son référentiel local à sa place finale dans la scène. Cette transformation est nommée *Model to World*; elle est définie mathématiquement par une matrice de transformation de dimensions 4 x 4. On multiplie donc tous les triplets de coordonnées connus dans *Model* (ils sont très nombreux comme nous venons de le voir précédemment) par cette matrice de transformation *Model to World* pour obtenir les triplets de coordonnées transformés dans *World.* Beaucoup de données et beaucoup de calculs jusqu'ici... et pourtant ce n'est pas fini. Nous ne venons que de placer les meshs dans le *World.* Encore faut-il les visionner. Selon les coordonnées de la caméra dans la scène et la direction de vue choisie, on définit une matrice 4 x 4 supplémentaire nommée *World to View.* Et comme auparavant, on obtient les coordonnées de tous les

meshes dans le référentiel de vue, en multipliant les résultats précédents par cette matrice *World to View.* Nous y sommes presque. Il nous reste à passer de la vue 3D à l'écran, c'est à dire à opérer une projection des dernières coordonnées spatiales obtenues en coordonnées de l'écran. Ces dernières vont dépendre entre autres de la forme de l'écran (4/3, 16/9, autre) et de l'angle de vision choisi, un peu comme lorsqu'on règle un zoom sur un appareil photo. Encore une dernière fois, cette projection se réalise par la multiplication du dernier résultat obtenu par une matrice de projection nommée *View to Projection*.

En résumé :
coord. à l'écran = (*View to Projection*) * (*World to View*) * (*Model to World*) * coord. Model

Par ailleurs, j'ai volontairement omis d'évoquer les mêmes transformations à opérer sur les vecteurs des normales, sur le calcul des couleurs de chaque pixel en fonction des lumières et des éventuels ombrages, ou de l'application des textures ...
On se rend donc vite compte de deux choses :
Le nombre de traitements à implémenter en Javascript est lourd et fastidieux, d'autant que ses fonctions mathématiques natives sont plutôt limitées (pas de calcul matriciel!). On aura donc tout intérêt à utiliser une librairie ou un framework qui propose une fois pour toutes ces opérations.
Le nombre de calculs flottants à réaliser dans chaque traitement est particulièrement conséquent d'autant qu'on peut avoir des dizaines de milliers de coordonnées à traiter. Le temps de traitement est en effet critique : le navigateur peut rafraîchir l'affichage, c'est à dire redessiner toute la scène, à la fréquence optimale de 60 trames par seconde (*frames per second ou fps*), soit un délai de 16 ms entre deux trames. Il faut donc parvenir à effectuer tous ces traitements dans ce laps de temps avec Javascript, un langage monothreadé à typage dynamique ... et encore, je n'ai pas parlé des autres traitements qui consisteraient à donner un peu de logique applicative à la scène par des mouvements des solides, des déformations, ou la mise en œuvre d'un moteur physique pour émuler la gravité ou calculer les collisions entre les meshes.

La solution à ce deuxième point va être l'utilisation de WebGL.

WebGL est une partie de l'API HTML5 accessible depuis l'environnement JS du browser. Elle permet d'envoyer des commandes OpenGL ES (*OpenGL for Embedded Systems*) directement à la carte graphique. La bonne nouvelle c'est que la quasi-totalité des appareils actuels, du PC, au smartphone, en passant par la tablette, intègrent maintenant une carte graphique dédiée, le GPU, compatible WebGL.
On peut donc déléguer une énorme quantité de traitements, et bien entendu, tout le rendu au GPU, qui justement est capable de paralléliser intensément tous les calculs matriciels de flottants. C'est ce que l'on appelle l'*accélération matérielle* puisqu'un autre processeur que le CPU se charge des calculs dédiés à l'affichage. Ce déport de traitements sur le processeur graphique décharge donc d'autant le CPU dont les ressources pourront alors être utilisées pour coder principalement la logique applicative en Javascript. L'accès à WebGL demeure très verbeux et de bas niveau : il faut déclarer des buffers de transfert de données du CPU vers le GPU, récupérer des pointeurs de variables utilisées sur le GPU, réaliser des liaisons (*binds*) entre les deux contextes, etc. Il est donc très complexe d'implémenter la logique applicative côté GPU, contrairement aux calculs matriciels déjà cités qui sont toujours les mêmes quelle que soit l'application. Ce qui nous amène à la solution du premier point : le choix d'un moteur 3D HTML5. Tous les moteurs 3D Web existants dignes de ce nom utilisent exactement de la même façon les points énoncés précédemment, à savoir des transformations matricielles entre des systèmes de coordonnées successifs calculés par le GPU.
Les choix se feront donc sur d'autres critères comme les fonctionnalités proposées, les objectifs visés par le moteur, ou les paradigmes utilisés.
Actuellement on dénombre trois grandes catégories de moteurs : les plateformes commerciales non-web de développement de jeux vidéo, les plateformes commerciales et/ou libres de développements de jeux HTML5, et les moteurs libres 3D HTML5.
Les plateformes non-web de développement de jeux telles que les grands standards *Unity3D* ou *Unreal Engine* fournissent un ensemble d'outils qui permettent d'élaborer des jeux vidéo quasiment sans programmation de la part de l'utilisateur. Les exécutables de jeux sont générés ensuite pour les OS cibles. Il est possible d'exporter le produit fini au format HTML5. Malheureusement, le code produit dans le cas de la 3D est beaucoup trop lourd pour être raisonnablement déployé sur le Web, si bien qu'il peut s'avérer plus judicieux d'utiliser ces plateformes pour concevoir les objets 3D et les exporter ensuite dans un outil Web plus performant.
Les plateformes de développement de jeux HTML5, comme *Playcanvas*, *Turbulenz* ou *Goo Engine* fournissent en général à l'utilisateur une API de programmation libre, et un accès payant à des outils en ligne de conception (éditeur 3D, réalisation de scènes), de génération du code (build) ou de distribution des jeux conçus. Ils se focalisent sur la réalisation de jeux uniquement, et sur une plateforme cible : le navigateur.
Enfin les moteurs libres 3D Web ne fournissent en général qu'une API de programmation et n'imposent aucune contrainte au développeur. Pour trier parmi la masse de projets existants, on ne retiendra que ceux qui ont une activité depuis au moins un an, et qui ont déjà sorti au moins une version stable, à savoir les deux projets opensource *Three.js* et *Babylon.js*. Three.js, le plus ancien des deux, n'est pas dédié a priori uniquement à la conception de jeux et permet d'effectuer des rendus visuels aussi bien en WebGL, qu'en Canvas 2D ou qu'en CSS3. Aussi fournit-il une quantité impressionnante de fonctionnalités afin d'essayer de couvrir tous les usages. Babylon.js se concentre au contraire uniquement sur le rendu WebGL, et la performance dans les animations. Bien qu'il ne soit pas uniquement dédié à cet usage, il se veut capable de produire de vrais jeux 3D pour le navigateur, objectif pour lequel il a déjà été éprouvé avec succès comme le montrent par exemple les réalisations de Microsoft Edge (Flight Arcade) ou d'Ubisoft (Assassin's Creed Pirates). C'est à Babylon.js que nous allons nous intéresser dans la suite de cet article.

Babylon.js

Babylon.js a été créé par deux Français travaillant chez Microsoft, David Catuhe et David Rousset, en 2013. Il s'agit d'un port vers Javascript et WebGL d'un précédent moteur 3D éponyme conçu par David Catuhe pour la plateforme XNA. BabylonJS est un projet open source sous licence Apache 2.0 accessible à tous sur Github. Il propose de plus un site Web portail http://babylonjs.com donnant accès à tout un écosystème d'outils: le dépôt Github. Il s'agit d'un forum animé par une communauté particulièrement dynamique et réactive où de très nombreux français sont présents même si la langue d'échange y est l'anglais. C'est un site de documentation dédié aussi bien pour l'API que pour les tutoriels, des exemples de code par types de fonctionnalités, un éditeur en ligne associé à un rendu en direct, un éditeur de matériels, un éditeur de shaders (des programmes exécutés directement côté GPU), un importateur de données à partir de formats externes, etc. Les fonctionnalités proposées par BabylonJS sont trop nombreuses pour être détaillées ici. Je vous invite à les découvrir dans la documentation, les nombreux tutoriels, cours vidéo ou les exemples de démonstrations en ligne. Le credo de BabylonJS est « simple, powerful WebGL ». Ceci signifie qu'il suit principalement deux

buts dans sa conception: fournir à l'utilisateur l'API la plus simple possible en termes d'abstraction de la couche WebGL, et toujours optimiser la performance. A tel point qu'il est actuellement le seul moteur à implémenter des *webworkers* afin de threader le calcul des collisions entre la caméra et les meshes de la scène, ou encore de prendre déjà en compte dans les navigateurs qui le supportent la technologie *SIMD (single instruction on multiple data)* pour paralléliser sous Javascript les calculs matriciels de flottants demeurant côté CPU.

Une application Babylon.js

Concrètement, une application BabylonJS n'est rien d'autre qu'une page HTML5 embarquant du code Javascript, la librairie babylon.js, et contenant un élément *canvas*. Tout est exécuté dans le navigateur. Il sera nécessaire cependant de charger la page HTML depuis un serveur Web, qui peut être local, plutôt que directement depuis le système de fichiers de la machine, du fait des restrictions de sécurité dans les navigateurs.
Commençons donc par une page HTML minimale : *index.html*.
Dans cette page nous allons charger *babylon.js*, la librairie BabylonJS elle-même, *hand.js,* un petit programme permettant de prendre en charge et d'abstraire tous les types de pointeurs (souris, doigts, etc). Ce dernier nous permettra de déplacer la caméra de la même façon sur un PC ou une tablette. Nous chargerons également le fichier dans lequel nous coderons notre application : *mon_appli.js*.
Comme vous le voyez, une fois la page chargée, la fonction *init* est appelée. Nous la coderons donc dans *mon_appli.js*

index.html :

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset = 'utf-8'>
  <title>Ma première application BJS</title>
  <script src = 'babylon.js'></script>
  <script src = 'hand.js'></script>
  <script src = 'mon_appli.js'</script>
 </head>
 <body>
  <canvas id='renderCanvas'></canvas>
  <script>
   window.onload = init;
  </script>
 </body>
</html>
```

mon_appl.js : On code la fontion *init* qui récupère l'élément canvas du DOM et qui crée une instance de moteur BABYLON avec ce canvas. Le moteur 3D est lancé, il affiche une scène que nous coderons dans la fonction *createScene()* dans le même fichier.

```
var init = function() {
 var canvas = document.querySelector('#renderCanvas');
 var engine = new BABYLON.Engine(canvas, true);
 var scene = createScene(canvas, engine);
 engine.runRenderLoop(function() { scene.render(); });
};
```

Ces seuls deux petits bouts de code réutilisables suffisent à mettre en place l'environnement d'exécution du moteur 3D. Il reste donc maintenant à créer la scène, son contenu et ajouter la logique applicative.

Commençons par un exemple simple, c'est à dire une scène contenant un cube (*box*) posé sur un sol (*ground*) et éclairé depuis le haut. BabylonJS fournit de nombreuses fonctions permettant de créer en une seule ligne ces types d'objets. Nous allons donc successivement créer un objet *scene*, un objet *camera,* et lui attacher les contrôles afin que cette caméra puisse être déplacée avec la souris, le clavier, ou au toucher sur un écran tactile et une source lumineuse dont on donnera la direction.
Ensuite nous créerons un objet *ground* et un objet *box* auxquels nous associerons à chacun un *material* pour leur donner une couleur et une opacité, par exemple.

```
var createScene = function(canvas, scene) {
 var scene = new BABYLON.Scene(engine);

 // camera
 var camera = new BABYLON.ArcRotateCamera('cam', 0, 0, 0, BABYLON.Vector3.Zero(), scene);
 camera.setPosition(new BABYLON.Vector3(0, 10, -60));
            //placement de la caméra dans la scène
 camera.attachControl(canvas);  // attache des contrôles
 // lumière
 var light = new BABYLON.HemisphericLight('light', new BABYLON.Vector3(0, 1, 0), scene);

 // sol
 var ground = BABYLON.Mesh.CreateGround('ground', 100, 100, 4, scene);
 ground.material = new BABYLON.StandardMaterial('groundMat', scene);
 ground.material.diffuseColor = new BABYLON.Color3(0.7, 0.7, 0.7);

 // cube
 var box = BABYLON.Mesh.CreateBox('box', 10, scene);
 box.material = new BABYLON.StandardMaterial('boxMat', scene);
 box.material.diffuseColor = BABYLON.Color3.Blue();
 box.material.alpha = 0.7;      // transparence
 box.position.x = 5;          // placement du cube dans la scène
 box.position.y = 5;
 box.rotation.y = Math.PI / 3;  // rotation du cube autour de son axe Y

 return scene;
};
```

Ce qui nous permet d'obtenir la scène de la figure suivante qui, à part la simplicité de son code, ne présente pas grand intérêt. En général, les utilisateurs qui souhaitent construire des scènes sophistiquées conçoivent leurs objets dans des logiciels de modélisation 3D comme Blender, 3ds Max ou Unity avant de les importer dans BabylonJS Fig.1.
Cependant l'utilisation des formes de mesh basiques (cube, anneau, sphè-

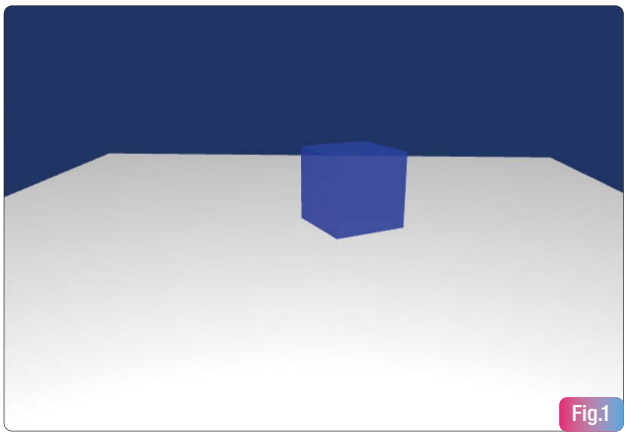


Fig.1

re, tube, etc) proposées par BJS peut être suffisante pour la représentation de données. Pour ceci nous allons nous intéresser à un type de formes particulier : les formes paramétriques.

Les formes paramétriques : le ruban

Les formes paramétriques n'ont pas de forme prédéfinie, contrairement par exemple à une sphère pour laquelle on s'attend à voir une forme sphérique. Leur forme va dépendre d'un ensemble de données qui leur seront passées en paramètre à la construction du mesh. On a donc ici un moyen simple de transformer des jeux de données en visualisation graphique 3D. Les formes paramétriques de BJS présentent de plus la particularité d'être toutes dynamiquement modifiables, c'est à dire que la forme initialement construite pourra être ensuite modifiée si le jeu de données passé en paramètre est modifié. Examinons la forme paramétrique de base de BJS, à savoir le ruban (*ribbon*). Un ruban représente la surface entre deux (ou plus) chemins, un chemin n'étant qu'une suite de points consécutifs dans l'espace Fig.2. Si on dispose de plusieurs jeux de données (des mesures selon des intervalles de temps par exemple), il est assez facile de transformer chaque jeu en coordonnées de points, constituant alors chacun un chemin. Prenons un exemple concret : toutes les 5 secondes, je reçois du serveur (soit par une requête xhr, soit par une websocket) dans un tableau au format JSON un lot de 50 valeurs numériques qui peuvent être des mesures de métrologie, des valeurs boursières, etc. Je reçois simplement 50 nouvelles valeurs de ce qu'on mesure toutes les 5s et je décide d'afficher constamment 5 minutes de données, soit 5 x 60 = 300 jeux de données. Dans le code précédent, supprimons tout ce qui concerne le sol et la box, et créons un simple ruban plat pour commencer avec 300 chemins, chacun contenant 50 points.

```
var paths = [];
for (var i = 0; i < 300; i++) {
  var path = [];
  for (var j = 0; j < 50; j++) {
    path.push(new BABYLON.Vector3(25 - j, 0, i - 30));
    }
  paths.push(path);
}

var ribbon = BABYLON.Mesh.CreateRibbon('ribbon', paths, null, null, 0, scene, true);
ribbon.material = new BABYLON.StandardMaterial('mat', scene);
ribbon.material.wireframe = true;
```

Nous allons maintenant modifier ce ruban toutes les 5s avec les données reçues Fig.3. Pour ceci, nous allons simplement modifier les valeurs contenues dans chaque *path* du tableau *paths* à l'aide des nouvelles données. Nous rappellerons alors la méthode *createRibbon()* en lui donnant uniquement les nouvelles données et l'instance de *ribbon* en paramètre (le reste sera *null*) ce qui provoquera la mise à jour de la forme.

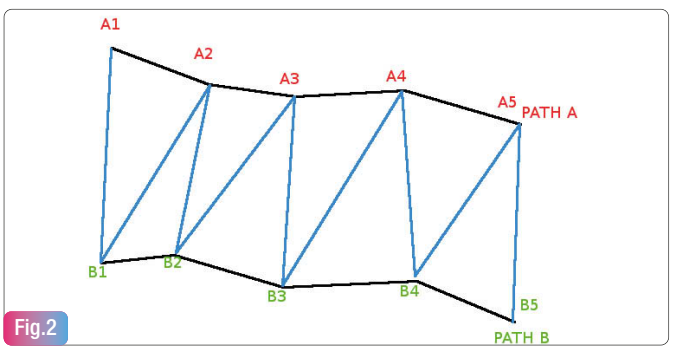


Fig.2

```
var delai = 5000;
window.setInterval(function() {
 var data = recupereDonnees(); // récupère le tableau de 50 données
 var path =[];
 // on construit un nouveau premier chemin
 // la donnée reçue est l'ordonnée y du point courant
 for (var j = 0; j < 50; j++) {
  path.push(new BABYLON.Vector3(j, data[j], 0));
 }
 // on décale tous les chemins existants en actualisant le z de chaque point
 for (var i = 299; i > 0; i--) {
  paths[i] = paths[i -1];
  var courant = paths[i];
  for (var k = 0; k < 50; k++) {
   courant[k].z = i;
  }
 }
 paths[0] = path;        // ajoute le nouveau chemin au début
 // on actualise enfin la forme
 BABYLON.Mesh.CreateRibbon(null, paths, null, null, null, null, null, null, ribbon);
}, delai);
```

L'essentiel du code traite ici des changements de coordonnées des points de chaque chemin. La mise à jour de la forme se code ensuite en une seule ligne Fig.4.

Conclusion

La 3D sur le Web procure de nombreux apports quels que soient les domaines applicatifs, y compris la simple visualisation de données numériques. Les avantages bien connus du déploiement Web (unicité de la plateforme cible : le navigateur, du langage, pas d'installation) pourraient être contrebalancés par la complexité et la charge des calculs à mettre en œuvre sur le client. Heureusement l'usage d'un framework WebGL comme BabylonJS remédie à cette difficulté en permettant à l'utilisateur final de faire abstraction de la couche du moteur 3D, et de se concentrer uniquement sur la logique applicative de son besoin.

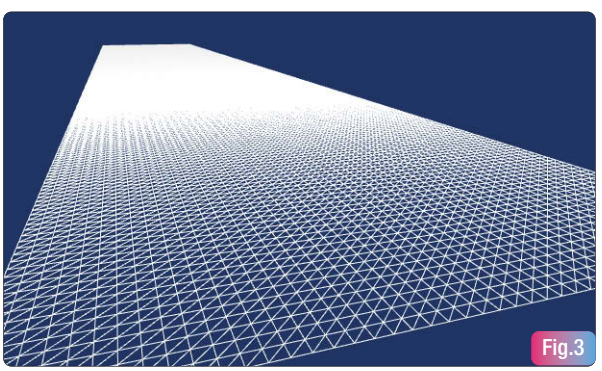


Fig.3

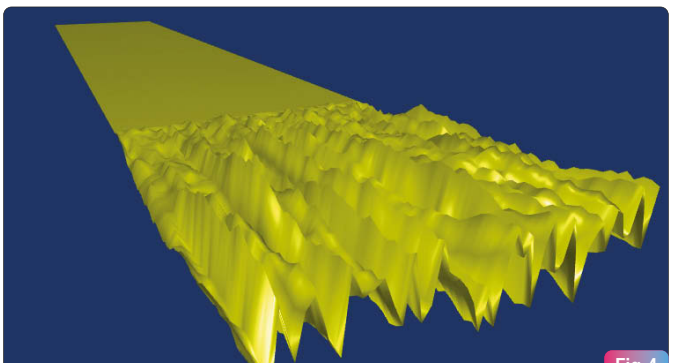
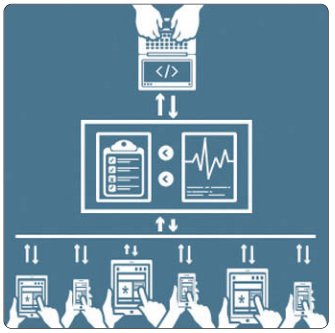


Fig.4

Quels outils pour « Beta-Tester » une application mobile ?



Cet article a pour objectif de présenter le beta test, son importance dans le cycle de développement des applications mobile, ainsi qu'une rapide étude comparative des outils disponibles sur le marché pour tester les applications iOS et Android.

Amine Bouzid
Mobilité & UX Design
NEOXIA

Il est évident que la fiabilité d'un produit logiciel, sa sécurité, ainsi que son aptitude à l'utilisation permettent d'évaluer sa qualité. Pourtant, des erreurs logicielles et des dysfonctionnements sont fréquemment rencontrés suite à la phase de développement. En effet, plus le service rendu est complexe, plus le nombre d'erreurs risques de croitre de façon exponentielle. Ainsi, la mise en place d'une démarche de test rigoureuse est primordiale, surtout sachant que l'être humain ne peut, par sa capacité réduite, gérer toutes complexités / tous les scénarios possibles. Dans ce contexte, il existe deux familles de tests distinctes:

- **ALPHA TEST :** cette catégorie comporte les tests unitaires, les tests d'intégration ainsi que les tests systèmes.
- **BETA TEST** : Il s'agit d'un nombre de tests effectués sur la version Beta obtenue par pre-release d'une version de l'application en question. Dans cette phase, l'application est confrontée à un échantillon représentatif d'utilisateurs cibles qui vont la tester, produisant ainsi un feed-back à l'éditeur pour qu'il puisse corriger et améliorer son produit.

Quelles spécificités pour les applications mobiles ?

L'intérêt d'effectuer des campagnes de tests pour les applications mobiles, avant leur publication sur les stores, est primordial. Du fait que le développement des applications mobiles se distingue du développement logiciel classique par des spécificités à savoir :

- Le développement très rapide du marché mobile.
- La segmentation importante du marché, que ce soit sur le volet matériel (les différentes constructeurs) ou logiciel (les différentes versions d'OS pour un même constructeur !).
- La connectivité limitée des mobiles via le réseau cellulaire.

Et, quelles obligations pour l'éditeur d'une application mobile :

- Trouver le juste milieu entre le Time To Market (TTM) et le temps de test approfondi.
- Tester sur un maximum de devices (terminaux mobile) représentant l'audience cible.
- Optimiser le transfert de données entre client et serveur, prévoir une version hors ligne de l'application, le cas échéant des messages d'erreurs claires et précis pour améliorer l'expérience utilisateur.

Face à ces contraintes, le test d'une application mobile peut s'avérer complexe et fastidieux. D'où l'intérêt de recourir à des outils pour cadrer et organiser cette phase de Beta Test. Dans ce qui suit, nous allons présenter une étude comparative de ces outils indispensables.

Les principales fonctionnalités d'un outil BETA TEST pour mobile

Avant d'entamer cette étude, on va tout d'abord définir ce qu'est un bon outil de beta test pour mobile et qu'elles sont les fonctionnalités qu'il doit impérativement comporter.

Comme première réflexion à ces questions, nous avons établi une liste de fonctionnalités par ordre de priorité. D'après notre retour d'expérience, un bon outil de beta test pour mobile doit assurer :

- La facilité d'acheminement de l'application aux beta testeurs, ce qui n'est toujours pas une tâche triviale surtout pour les applications sous iOS. Bien évidemment, la version beta ne peut pas être sur les stores publiques et doit être distribuée à une communauté restreinte. Le rôle des outils Beta Test et de rendre cette étape de récupération et d'installation des versions beta plus facile et plus fluide pour les beta-testeurs.
- La récupération des rapports de bugs ou des anomalies depuis les sessions de test des utilisateurs. Généralement, plus l'outil remonte des informations des sessions utilisateur, plus il est considéré comme efficace. Comme exemple de ces information l'OS, la marque du device, le type de réseau utilisé, la disponibilité de la RAM sur l'appareil, la charge du Processeur au moment où l'application tourne, des captures écrans, etc.
- Un environnement privé pour l'éditeur de l'application, avec la gestion des droits des utilisateurs qui peuvent y accéder.
- La facilité d'intégration du SDK de l'outil de test avec le produit.
- Un support pour l'Intégration Continue.

Par ailleurs, ces fonctionnalités doivent être assurées sans impacter l'expérience utilisateur (lenteur, redirection, intrusion, etc.). Un utilisateur en phase de Beta Test doit pouvoir simuler une utilisation 100% normale de l'application.

Quelques outils de BETA TEST pour mobile

Établir une liste de ces outils par ordre de préférence n'est pas chose facile surtout pour un marché aussi segmenté que le marché mobile. Ici, nous nous concentrerons sur les deux OS les plus répandus actuellement, à savoir Android et iOS. Nous établirons dans ce cadre une liste de recommandations basées sur le retour d'expérience de NEOXIA sur ce sujet.

Mais avant de présenter ces outils, jetons tout d'abord un coup d'œil sur les grandes firmes mobiles qui sont considérées comme les grands utilisateurs (consommateurs) de ces outils. Le tableau suivant présente une liste des dernières acquisitions effectuées par ces firmes.

Nom	Outil de test acheté	Année acquisition
Twitter	Crashlytics	2013
Apple	TestFlight	2014
Microsoft	HockeyApp	2014
Google	Appurify	2014

Liste des acquisitions d'outils de beta test pour applications mobiles

Il faut noter aussi que Facebook ainsi qu'Amazon développent leur propre outils de beta test pour mobile respectivement Airlock (not open) et Amazon's A/B Testing Service

En dehors de ces grandes firmes mobiles, il est bien clair que chaque éditeur/développeur d'application mobile devrait se munir d'un bon outil de beta test. Les questions qui se posent alors sont : comment choisir entre les différents outils disponibles sur le marché ? Que faut-il chercher au juste ?

TestFairy (gratuit/$)

TestFairy est un outil de beta test qui a l'avantage d'être facilement intégrable sur l'application Android. Il suffit d'uploader le build de l'application Android .apk sur le compte TestFairy et inviter les beta testeurs soit par mail ou à travers la communauté de testeur TestFairy. Aucun ajout sur le code de l'application n'est nécessaire, les serveurs de TestFairy s'occupent des injections du code nécessaire pour remonter les logs et les captures d'écran.

TestFairy propose aussi une fonctionnalité très pratique qui permet de forcer la mise à jour des anciennes versions de l'application, cela évite la confusion créée par différentes versions beta sur les appareils des testeurs.

TestFairy a ajouté récemment le support pour les applications iOS. Il est maintenant possible de distribuer ces applications de test iOS via cet outil. Cependant, nous n'avons pas encore eu l'occasion de tester cette fonctionnalité qui est elle-même actuellement en mode beta test !

Nous citons d'autres points forts de cet outil :

- Gestion des groupes de testeurs.
- Possibilité d'ajouter 500 testeurs pour le compte gratuit.
- Possibilité d'ajouter 10 applications différentes pour le compte gratuit.
- Rapport sur la durée de session de chaque testeur.
- Rapport sur la localisation des testeurs.
- Possibilité d'ajouter des checkpoints
- Enregistrement de capture d'écran et de vidéo qui aide à reproduire les bugs facilement.
- Rapport de couverture des différents OS, devices et résolutions.

Les schémas ci-après illustrent des exemples d'usage et de rapports de cet outil Fig.1, 2 et 3.

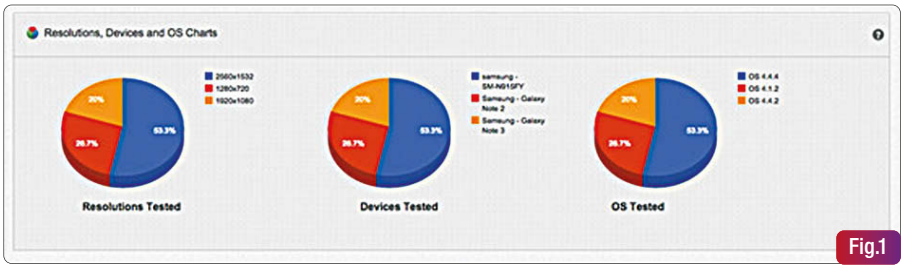


Fig.1

Statistiques des résolutions devices et OS.

Fig.2

Capture vidéo, informations générales et performances.

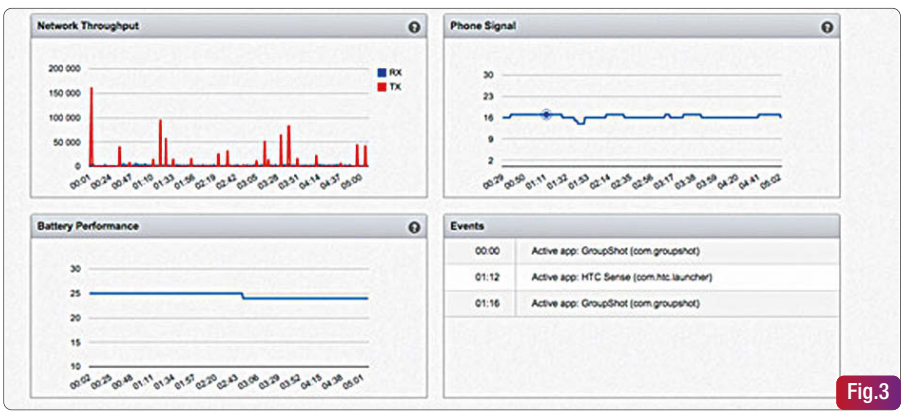


Fig.3

Consommation : batterie, connexion réseau et micro

TestFlight (gratuit)

TestFlight était sans doute le leader sur le marché du beta test et de la distribution des applications sur iOS et Android. Cependant, après l'acquisition de Burstly par Apple, TestFlight a abandonné le support pour Android, contraignant ainsi plusieurs développeurs qui ciblent les deux OS mobiles les plus répandus à devoir chercher un autre outil pour la distribution et le beta test sur Android, ou à chercher un outil multiplateforme pour ne pas avoir à gérer deux outils, deux communautés de test, etc.

Mais pour rendre à César ce qui est à César, TestFlight, à travers son SDK, représentait un outil incontournable pour les développeurs d'application iOS avec des fonctionnalités comme :

- Le rapport de crash en temps réel.
- Les checkpoints pour vérifier la couverture des tests.

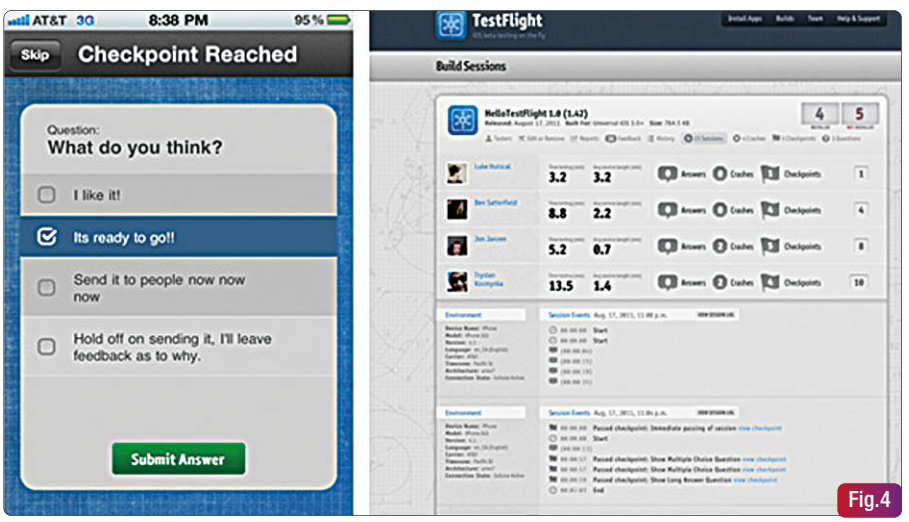


Fig.4

L'ancienne interface de TestFlight: – Fenêtre de feedback pour les testeurs. – Compte rendu des sessions + logs

- Les questions In-App pour poser des questions aux utilisateurs au moment opportun.
- La récupération du feedback des testeurs.
- Les mises à jour automatiques des versions beta.

L'intégration du SDK est plus ou moins facile : Il faut ajouter un **fichier .h** et un **Framework .a** puis une ligne de code au lancement de l'application pour démarrer la capture de la session de test.

L'ajout de testeurs s'effectue via une invitation par mail, ensuite l'utilisateur ajoute un device de test, le développeur ajoute les UDIDs au provisioning et upload la version beta de l'application. Pour information, un **UDID (Unique Device Identifier)**, comme son nom l'indique, est un identifiant unique d'un terminal mobile Fig.4.

Depuis le 26/02/2015 le service de TestFlight est arrêté, et complètement intégré sur iTunesConnect. Les changements qu'Apple a apporté à TestFlight sont les suivants :

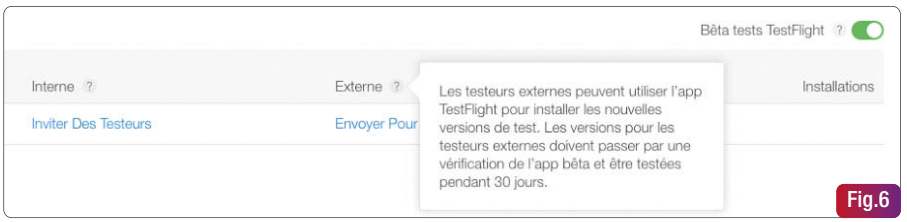
- Les devices de tests sont gérés par email et plus par UDIDs
- La limite du nombre de terminaux associés à un compte est passée de 100 à 1000
- Chaque nouvelle version bêta de l'application est soumise à une vérification par l'équipe Apple (qui peut aller jusqu'à 30 jours !) avant de pouvoir la partager avec les testeurs externes Fig.5 et 6.

AppSee (gratuit/$)

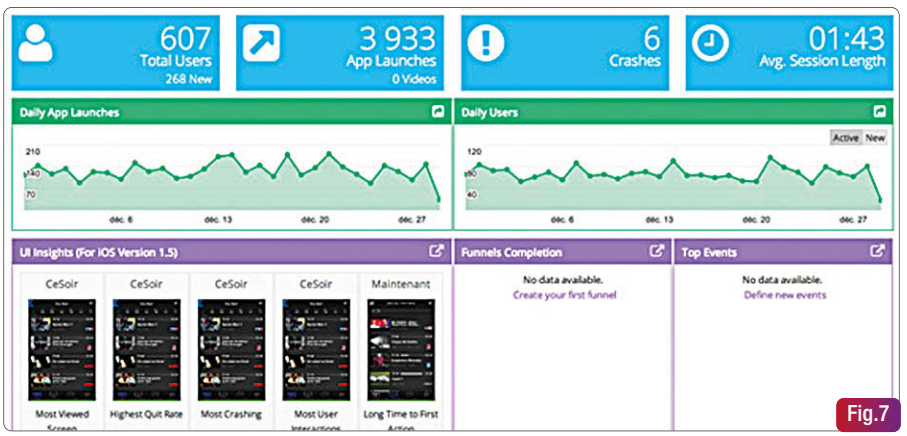
AppSee est un outil très puissant pour le monitoring des applications beta, que ce soit pour la reproduction et la correction des crashs ou pour avoir des idées d'optimisation. AppSee présente également un bon support. AppSee offre dans ce cadre des fonctionnalités de monitoring très intéressantes :

- Enregistrement vidéo des sessions : cela permet d'étudier en détail le comportement des utilisateurs de l'application, et facilite la reproduction des bugs. Cette fonctionnalité donne aussi la possibilité d'exclure des parties de l'application qui contiennent des informations sensibles (login/password). Les champs de mot de passe sont exclus par défaut.
- HeatMap de click : AppSee enregistre tous les types de touche, les combine et crée une carte de la densité des click sur tout l'écran. Cela permet au développeur de savoir les parties de l'écran les plus attractives pour l'utilisateur, et les boutons/fonctionnalités non utilisées ou non intuitives.
- Analyse de l'utilisation de l'application en temps réel : cette fonctionnalité constitue un outil très puissant pour l'optimisation de l'application car elle se base sur les statistiques d'utilisabilité de l'application. Comme

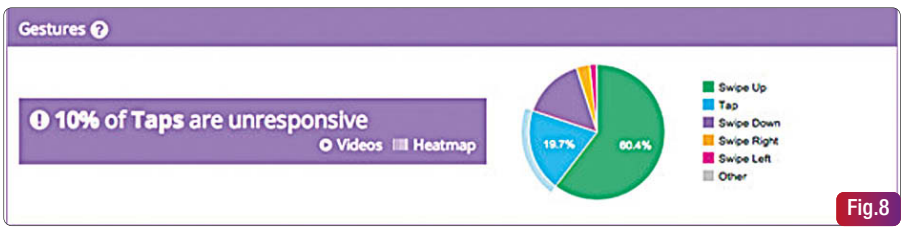
Possibilité d'ajout de 1000 testeurs externes à l'organisation.



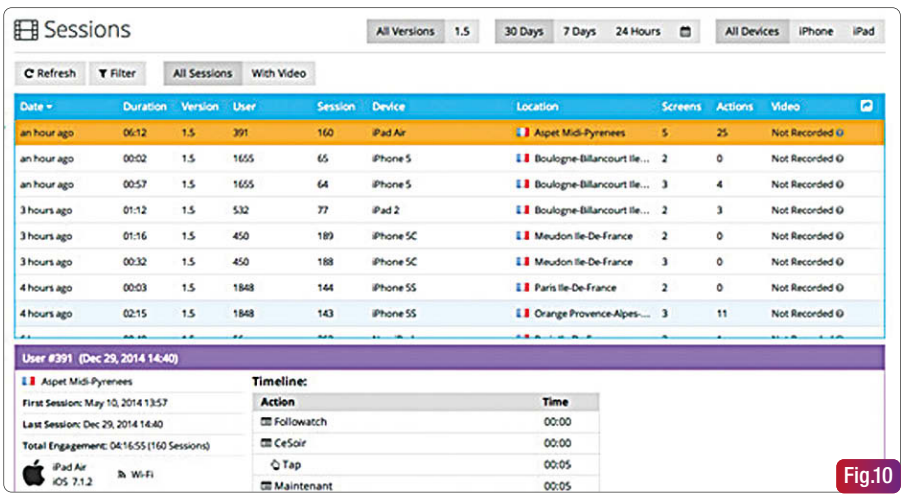
Soumission du build bêta à la validation, avant de parvenir aux testeurs.



Liste des écrans les plus visités + généralités sur l'application



Rapport de la résponsitivité de l'application aux gestes



Liste des sessions et timeline des actions

exemple : les écrans où les utilisateurs passent le plus de temps, les statistiques sur les écrans où les utilisateurs quittent l'application, l'ordre de navigation au sein de l'application, etc.

- Ajout d'évènements automatiques : en plus de la vidéo de la session, le SDK AppSee ajoute une timeline descriptive des actions effectuées par le testeur.
- Statistique d'usage de l'application : la tendance des utilisateurs, le nombre d'utilisateurs récurrents, le temps moyen des sessions, le temps entre deux sessions, etc.

Il est vrai que ces fonctionnalités sont liées à l'analytique plus qu'au test qui nécessite bien sûr une gestion du processus de distribution, mais ces fonctionnalités d'AppSee aident considérablement à comprendre l'attitude des testeurs et même des vrais utilisateurs de l'application, dans le but d'améliorer l'expérience utilisateur. Cependant, deux points sont à reprocher à AppSee :

- La capture vidéo consomme parfois énormément de charge du processeur ce qui ralentit l'application, même s'ils affirment sur leur site le contraire « **The recording process is completely transparent to the end user, and has no NOTICEABLE effect on your app's performance.**». Heureusement le ratio d'enregistrement de la vidéo est paramétrable, il faut cependant faire plusieurs essais pour trouver le bon ratio afin de récupérer le maximum d'information sans pénaliser les performances de l'application.
- AppSee ne gère pas la distribution des applications et la gestion des comptes testeurs.

Les figures suivantes présentent des exemples de rapport d'AppSee Fig.7, 8, 9 et 10.

	AppSee	TestFairy	TestFlight	HockeyApp
Prix	Essai/custom price	gratuit	gratuit	payant
Plateformes	Android, iOS	Android, iOS	iOS	iOS, OSX, Android, WP
Distribution	-	+++	++	+
Gestion utilisateurs / groupes	-	++	+++	++
Facile à implémenter	+	+++	+/-	-
Support pour l'IC	-	++	-	-
Rapports avancés	+++	++	-	++
Rapports Vidéo	+++	++	-	-
Rapports de Crash	++	+++	+++	++
Rapport de couvertures	+++	+++	-	+/-

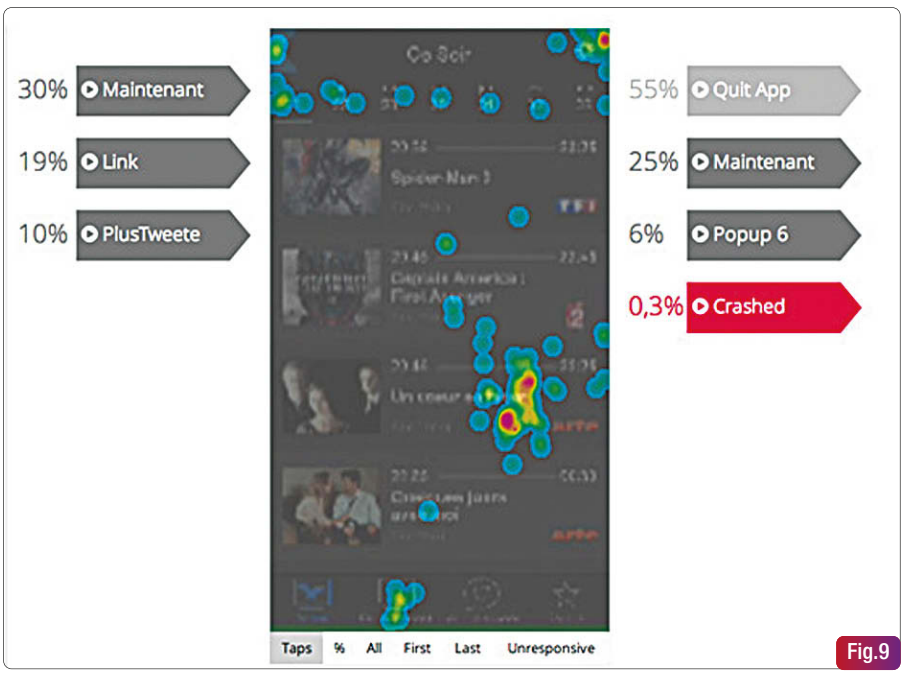


Fig.9

HeatMap d'un écran et pourcentage de navigation vers les autres écrans

Récapitulatif

Ce qu'il faut retenir :

Le Beta test :

- Est une phase critique du cycle de développement des applications mobiles.
- Ne se limite pas à la découverte des bugs.
- Se base sur la collaboration étroite entre l'équipe de développement et les testeurs.

Les outils de Beta test :

- Facilitent la vie aux développeurs en permettant la distribution des différentes versions beta de l'application sur les testeurs.
- Simplifient la récupération des rapports des sessions des tests et les feedbacks des testeurs.

Le tableau ci-dessus présente une synthèse des principaux outils Beta Test pour les applications mobile avec notre propre évaluation de la qualité et de la pertinence des fonctionnalités offertes par ces outils.

Windows 10 pour les développeurs

Avec la sortie de Windows 10, les développeurs peuvent désormais s'amuser avec le nouveau Windows Software Development Kit (SDK). Un nouveau SDK implique de nouvelles fonctionnalités, et le SDK Windows ne déroge pas à cette règle. Ce SDK nous permet de réaliser des Universal Windows Platform (UWP) apps, comprendre ici applications universelles pour les familles de devices Windows 10 : Windows Desktop, Windows Mobile, Windows IoT, Xbox...

Teddy DESMAS / Daniel DJORDJEVIC
Maxime FRAPPAT
Consultants Infinite Square

Le concept d'application universelle est une notion très présente de nos jours. Nous avons l'exemple avec les applications universelles Windows 8.1 et l'envie de permettre au développeur de partager son code métier entres les différentes plateformes, tout en développant des vues spécifiques pour celles-ci. Cependant, nous parlons dorénavant d'une variété de device beaucoup plus grande, avec des interactions variées, des formes d'écrans aux proportions différentes, des densités de pixels inégales et d'autres caractéristiques uniques, propres à chaque plateforme Fig.1.

Chaque famille apporte son API spécifique en plus de l'API commune, pour pouvoir offrir à l'utilisateur une expérience plus adaptée en fonction du device utilisé. Le développeur, de son côté, peut choisir les fonctionnalités qu'il désire apporter à une plateforme, pour distinguer clairement l'une de l'autre s'il en ressent la nécessité.

Pour cela, le développeur peut voir le rendu de son application sur les différentes familles de device, avec Visual Studio. Par défaut, le développeur aura sûrement l'intention de viser un panel de famille plus large. Cependant, il pourra choisir ceux qui l'intéressent dans un premier temps, puis en ajouter par la suite Fig.2.

C'est dans ce cadre que le développeur d'applications Windows 10, avec cette notion de mise en commun de la logique métier, peut utiliser les nouveaux concepts, outils et contrôles fournis par ce nouveau SDK. Voyons immédiatement ces nouveautés.

Fig.1

Fig.3

Windows 10 et le XAML

Le développement d'application Windows 10 cible donc de nombreuses familles de devices, c'est pourquoi il faut, plus que jamais, se préparer à développer des interfaces utilisateurs pour de nombreux formats. C'est pourquoi, nous disposons maintenant de nouveaux contrôles et éléments qui vont permettre aux développeurs de fournir une expérience user-friendly, quel que soit le périphérique sur lequel l'utilisateur est, que ce soit sur téléphone, tablette, ou des écrans plus larges.

D'une manière générale, on peut constater de manière générale que le XAML s'oriente vers un format plus clair, en essayant de gommer les fioritures du langage, afin de pouvoir écrire un XAML moins verbeux, et plus simple à comprendre, sans s'y perdre.

RelativePanel, le nouveau conteneur

Parmi les nouveaux contrôles, notamment de type conteneur, nous avons maintenant le RelativePanel. Depuis de nombreuses années de nombreuses années que Grid, Canvas, StackPanel, ScrollViewer et compagnie ont pour tâche d'agencer les différents éléments des vues de nos applications. Dorénavant, une nouvelle façon d'organiser son contenu est à notre disposition. Le nom du contrôle en dit beaucoup sur son principe : la relativité Fig.3.

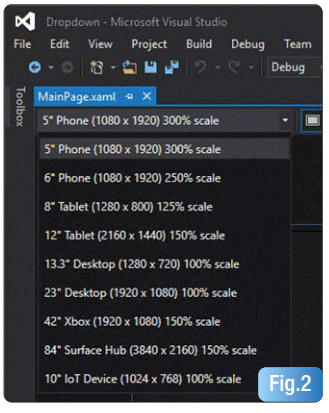


Fig.2

En effet, ce nouveau contrôle dispose les éléments en déclarant des relations entre ses enfants. C'est-à-dire que l'on va pouvoir positionner un élément A à droite d'un élément B, ce qui signifie que le déplacement de l'élément B entraînera le déplacement de l'élément A, celui-ci étant toujours positionné à la droite de B.

Cette nouvelle façon de penser semble intrigante à première vue, mais cela offre réellement de nombreuses possibilités une fois qu'on s'est habitué à son fonctionnement. C'est avec plus de 16 Attached Properties que l'on peut positionner les enfants du RelativePanel entre eux, mais aussi entre le RelativePanel lui-même et un enfant particulier, voyons un peu comment cela se passe au niveau du code.

On s'aligne en fonction du RelativePanel

Tout d'abord, on déclare notre RelativePanel et on y ajoute un élément sans rien lui spécifier.

```
<RelativePanel>
    <Rectangle x:Name="Rect1" Width="50" Height="50" Fill="Crimson"/>
</RelativePanel>
```

Résultat :

On observe un comportement plutôt normal, l'élément vient se placer tout en haut à gauche du RelativePanel Fig.4.

Ensuite, on utilise les Attached Properties pour positionner notre rectangle au centre du RelativePanel. Là où, dans une Grid, nous aurions fait HorizontalAlignment et VerticalAlignment à Center, nous avons :

```
<RelativePanel
    <Rectangle x:Name="Rect1" Width="50" Height="50" Fill="Crimson"
```

```
        RelativePanel.AlignHorizontalCenterWithPanel="True"
        RelativePanel.AlignVerticalCenterWithPanel="True" />
</RelativePanel>
```

Résultat :
Ici, pas de surprise, notre carré est bien au centre du RelativePanel. On voit que les deux Attached Properties sont plutôt explicites Fig.5.
Vous pouvez donc jouer avec ces différentes propriétés pour agencer vos enfants en fonction du Panel :

```
<Rectangle RelativePanel.AlignVerticalCenterWithPanel="True"
      RelativePanel.AlignHorizontalCenterWithPanel="True"
      RelativePanel.AlignRightWithPanel="True"
      RelativePanel.AlignLeftWithPanel="True"
      RelativePanel.AlignTopWithPanel="True"
      RelativePanel.AlignBottomWithPanel="True"/>
```

A savoir : si l'on spécifie AlignLeftWithPanel et AlignRightWithPanel à True, le rectangle sera centré.

Agencement entre éléments
C'est l'essence même du RelativePanel, on est toujours sur des Attached Properties comme pour le RelativePanel, cependant, à la place du booléen, il est cette fois-ci nécessaire de spécifier l'élément relatif à l'action. On a donc par exemple :

```
<RelativePanel>
    <Rectangle x:Name="Rect1" Width="50" Height="100" Fill="Crimson"
        RelativePanel.AlignHorizontalCenterWithPanel="True"
        RelativePanel.AlignVerticalCenterWithPanel="True"/>

    <Rectangle x:Name="Rect2" Width="50" Height="50" Fill="CadetBlue"
        RelativePanel.AlignVerticalCenterWith="Rect1"/>
</RelativePanel>
```

Résultat :
Ici, on a notre rectangle bleu (Rect2) qui va s'aligner verticalement avec le rectangle rouge (Rect1).
Supposons que dans notre développement, nous changeons d'avis, et nous voulons que le rectangle bleu soit à la même hauteur que le rectangle rouge, nous avons juste à remplacer le AlignVerticalCenterWith par AlignTopWith Fig.6.


Fig.4

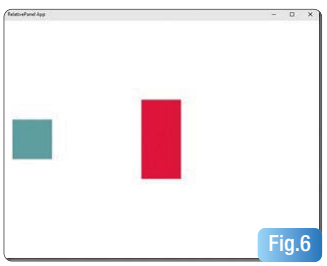


Fig.6

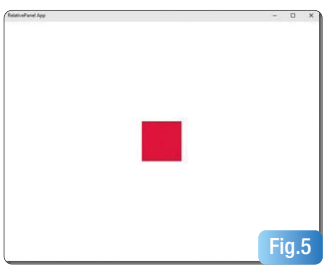


Fig.5

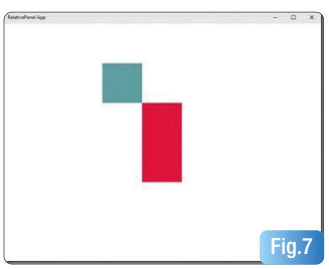


Fig.7

On peut donc jouer avec les propriétés suivantes :

```
<Rectangle x:Name="Rect2" Width="50" Height="50" Fill="CadetBlue"
      RelativePanel.AlignHorizontalCenterWith="Rect1"
      RelativePanel.AlignVerticalCenterWith="Rect1"
      RelativePanel.AlignLeftWith="Rect1"
      RelativePanel.AlignRightWith="Rect1"
      RelativePanel.AlignTopWith="Rect1"
      RelativePanel.AlignBottomWith="Rect1"/>
```

Cependant, la puissance du RelativePanel ne s'arrête pas là. Outre l'alignement des enfants, on peut aussi définir une position.

LeftOf, RightOf, Above, Below
Les propriétés étant assez explicites, voyons immédiatement ce qui se passe en spécifiant LeftOf et Above sur notre rectangle bleu (Rect2) :

```
<Rectangle x:Name="Rect2" Width="50" Height="50" Fill="CadetBlue"
      RelativePanel.LeftOf="Rect1"
      RelativePanel.Above="Rect1"/>
```

Résultat :
En reprenant les exemples précédents, le rectangle bleu (Rect2) va venir se coller au-dessus du rectangle rouge (Rect1). Cela va nous permettre ainsi de gérer des positions en même temps que des alignements dans notre vue Fig.7.
Le RelativePanel est un nouvel outil qui vient densifier la panoplie de contrôles XAML dont nous disposons. Et par son comportement spécifique, permettre aux développeurs de revoir la façon dont la vue sera développée. Le RelativePanel apporte un peu de fraîcheur à notre code XAML, en nous évitant des imbrications multiples de Grid dans certains cas. Nous avons tous connu à un moment les fameuses hésitations de la sorte : « Suis-je dans cette Grid là, ou l'autre ? C'est donc dans la Row 1, ou la 2 ? Je dois mettre le RowSpan ici, non ? »
Enfin, là où le RelativePanel va exceller, c'est dans cette optique d'unifier la vue, où l'on va pouvoir définir nos différents positionnements et alignements pour telle ou telle vue. Ainsi, nous pouvons passer à la nouveauté suivante, qui est justement très utile quand on l'associe à un RelativePanel : l'AdaptiveTrigger.

AdaptiveTrigger, adaptons notre vue

En tant que développeur d'applications Windows Phone et Windows Store, vous avez probablement eu affaire au VisualStateManager pour manipuler votre vue en fonction de la taille de la fenêtre, pour gérer le snap de Windows 8.1. Cependant il fallait s'abonner à l'évènement SizeChanged de la Frame de l'application, puis changer l'état soi-même grâce à la méthode GoToState du VisualStateManager.
Tout d'abord, il n'est plus obligatoire de définir une animation pour chaque état de son VisualStateManager, car celui-ci supporte maintenant les Setters, cela nous permet d'avoir, encore une fois, un XAML moins dense.
Ensuite, avec Windows 10 et les AdaptiveTriggers, le développeur va pouvoir définir des évènements pour la vue, avec un comportement similaire à celui que l'on vient d'aborder, et tout cela en pur XAML, sans aucun code-behind.
Prenons un exemple classique : on désire avoir un label à gauche d'une TextBox lorsqu'on dispose d'assez d'espace en longueur, et avoir le label au-dessus de la TextBox lorsque l'espace est restreint. En somme, on aimerait avoir un formulaire responsive, voici un exemple avec le ResponsivePanel :

```
<Grid x:Name="LayoutRoot" Background="White">
  <VisualStateManager.VisualStateGroups>
    <VisualStateGroup>
      <VisualState x:Name="NormalState">
        <VisualState.StateTriggers>
          <AdaptiveTrigger MinWindowWidth="800" />
        </VisualState.StateTriggers>

        <VisualState.Setters>
          <Setter Target="TxtBox.(RelativePanel.RightOf)"
                  Value="Label"/>
        </VisualState.Setters>
      </VisualState>

      <VisualState x:Name="SpecialState">
        <VisualState.StateTriggers>
          <AdaptiveTrigger MinWindowWidth="0" />
        </VisualState.StateTriggers>

        <VisualState.Setters>
          <Setter Target="TxtBox.(RelativePanel.Below)"
                  Value="Label"/>
        </VisualState.Setters>
      </VisualState>
    </VisualStateGroup>
  </VisualStateManager.VisualStateGroup>

  <RelativePanel Padding="24">
    <TextBlock x:Name="Label" Text="Label" />
    <TextBox x:Name="TxtBox" Background="Blue" />
  </RelativePanel>
</Grid>
```

On définit ici que le VisualState « NormalState » est utilisé lorsque la Width est supérieure à 800 avec :

```
<VisualState.StateTriggers>
<AdaptiveTrigger MinWindowWidth="800" />
</VisualState.StateTriggers>
```

Dans cet état-là, nous avons assez d'espace, donc nous utilisons la propriété RelativePanel.RightOf pour indiquer que la TextBox doit se positionner à la droite du Label.

```
<Setter Target="TxtBox.(RelativePanel.RightOf)" Value="Label"/>
```

Dans le cas contraire, si nous disposons d'une Width inférieure à 800, l'autre VisualState « SpecialState » est déclenché, et c'est donc la propriété Below qui est utilisée Fig.8.

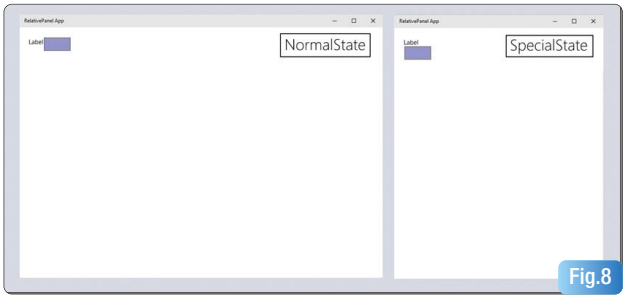


Fig.8

Cet exemple est relativement simple, mais montre bien que l'AdaptiveTrigger va devenir essentiel au développement d'applications responsives, afin d'avoir un affichage adapté à chaque format d'écran.

SplitView, un panneau latéral

Parmi ces nouveaux contrôles XAML, nous avons dorénavant la possibilité d'utiliser une SplitView dans nos applications. L'objectif de ce contrôle est de permettre au développeur d'intégrer la gestion d'un panneau latéral dans son application. Attention tout de même, ce n'est pas un Hamburger menu intégré et déjà prêt à l'utilisation. Si vous désirez un Hamburger menu authentique, il va falloir combiner la SplitView avec un bouton qui va manipuler la SplitView, rien de bien compliqué en soi !
Nous avons donc le SplitView.Pane qui regroupe les différents menus de l'application, la gestion du profil, les paramètres, par exemple. Puis le SplitView.Content qui, comme son nom l'indique, va accueillir la page de contenu.

```
<SplitView>
<SplitView.Pane>
   // Le menu latéral
   </SplitView.Pane>
   <SplitView.Content>
      // Le contenu
   </SplitView.Content>
</SplitView>
```

Pour manipuler notre SplitView, pas de secrets, certaines propriétés sont indispensables.
La propriété **IsPaneOpen** permet d'ouvrir/fermer votre menu.
PanePlacement, qui peut être égal à Left ou Right, détermine le côté par lequel le menu apparaît, et **OpenPaneLength**, la taille qu'il prend.
La propriété **DisplayMode** vous permet de choisir entre quatre différents types d'affichage :

- **Inline**, le menu latéral va pousser le contenu quand il va être ouvert.
- **Overlay**, le menu latéral apparaît au-dessus du contenu et n'impacte en rien celui-ci.
- **Compact Inline ou Overlay**, lorsque le menu est rétracté, une partie reste visible pour laisser les icônes visibles, comme on peut le voir sur l'illustration suivante Fig.9.

Ce nouveau contrôle permet au développeur de mettre en place des menus plus aisément, et ainsi offrir la meilleure expérience utilisateur possible, d'un point de vue ergonomique et dynamique.

Une vue différente pour chaque type d'écrans

Nous avons vu que l'AdaptiveTrigger est très utile lorsqu'on souhaite faire du responsive, cependant, en réalité, il est souvent nécessaire de développer une UI complètement différente pour chaque famille (Desktop, Mobile, Xbox, IOT, etc.), et donc par conséquence obtenir un fichier XAML très imposant. C'est pourquoi il est dorénavant possible de créer des vues spécifiques pour une famille. Prenons l'exemple du célèbre MainPage.xaml : nous avons notre MainPage.xaml à la racine de notre projet, celui-ci va être le XAML standard. Supposons ensuite que nous voulons un design particulier

Fig.9

Suite page 44

Mission on Mars Robot Challenge : rendre la robotique accessible à tous !

A l'occasion du salon Innorobo, l'éditeur de logiciels MathWorks organisait la grande finale de son concours de programmation robotique « Mission on Mars Robot Challenge ». La mission : explorer la planète Mars en programmant des robots Rover. Douze équipes aux profils variés (étudiants, professionnels, passionnés de robotique) se sont affrontés le 3 juillet dernier au cours d'une compétition qui a été intense et féroce.

Pendant 3 mois, les 220 équipes inscrites au concours – un réel succès – ont pu optimiser et peaufiner le modèle basique de simulation du robot Rover fourni par MathWorks en bénéficiant de versions temporaires des logiciels MATLAB, Simulink et Stateflow. Après une phase de pré-qualification pour départager les équipes inscrites, seules 12 équipes ayant réalisé les simulations les plus rapides et les plus innovantes se sont rendues à Lyon pour participer à la grande finale.

La première édition du concours s'était déroulée lors de la Maker Faire Paris 2014. Innorobo s'est imposé cette année, pour MathWorks, comme le lieu idéal pour organiser cette seconde édition et ainsi approcher le monde industriel, de la recherche, de l'enseignement ainsi que le grand public. Une opportunité de présenter ses solutions pour la robotique et mettre un coup de projecteur sur sa toute nouvelle Robotics System Toolbox. *« A travers cette compétition et son partenariat avec Innorobo, MathWorks réaffirme sa volonté d'offrir aux professionnels de l'industrie, aux chercheurs et à la communauté académique des outils robustes capables de répondre aux challenges de la robotique. Ces outils ont pour objectif d'accompagner au mieux les concepteurs de robots afin de les rendre plus fiables mais aussi les utilisateurs de robots existants pour développer de nouvelles applications robotiques. Enfin, ils constituent également de bons outils pédagogiques pour les enseignants afin de favoriser l'apprentissage des notions clés de la robotique telles que la cinématique, la dynamique et le contrôle. »* déclare Paul Cox, ingénieur d'application robotique.

3 mois pour optimiser !

L'objectif de la 2e édition est simple : explorer la planète Mars en détectant de façon autonome l'emplacement des sites à visiter en évitant des obstacles. Le but de la compétition n'est donc pas de construire un robot Rover (fourni aux participants lors de la finale) mais d'optimiser les algorithmes et le modèle de simulation régissant son comportement pour réaliser la mission d'exploration la plus rapide et intelligente. A partir d'un modèle de simulation permettant d'effectuer la mission sur Mars avec un temps de parcours basique, chaque équipe a dû optimiser à l'aide des outils MathWorks la partie de l'algorithme qui implémente la stratégie de déplacement du robot. Etant données la position estimée du robot, les informations de la caméra et les distances mesurées, ce module permet de décider de la trajectoire et de la distance que le robot va emprunter. Une interface graphique est également fournie pour visualiser la simulation des déplacements du robot.

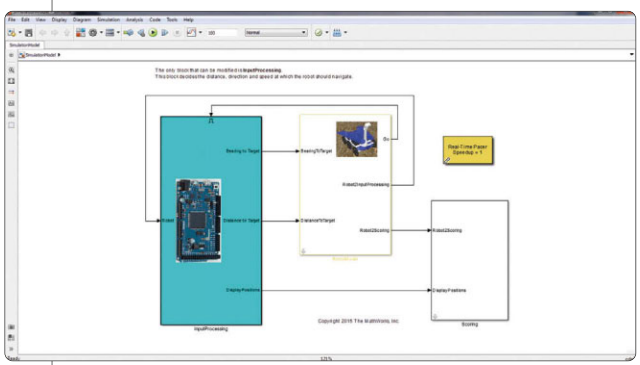
Plusieurs pistes pour optimiser le modèle se sont donc offertes aux équipes :

- Améliorer la stratégie existante basée sur de l'exploration pure

ROBOTICS SYSTEM TOOLBOX : UN ENVIRONNEMENT COMPLET POUR LES SYSTÈMES ROBOTIQUES

Dévoilé au printemps dernier, Robotics System Toolbox apporte un ensemble d'algorithmes et une interface matérielle pour le développement d'applications robotiques mobiles autonomes. Vous y trouverez des algorithmes de cartographie, planification et suivi de trajectoires pour des robots à entraînement différentiel ; grâce auxquels vous pouvez concevoir et prototyper des applications de contrôle moteur, vision par ordinateur, machines à états dans MATLAB ou Simulink et les intégrer avec les algorithmes de Robotics System Toolbox.

Cette toolbox est munie d'une interface entre MATLAB, Simulink et le Robot Operating System (ROS) afin de tester et vérifier des applications sur robots compatibles ROS et des simulateurs tels que Gazebo.



La génération de code C++ est supportée, vous permettant ainsi de générer un nœud ROS depuis un modèle Simulink et le déployer sur un réseau ROS. Plusieurs exemples sont fournis, illustrant l'utilisation de cette toolbox avec des robots virtuels dans Gazebo ainsi qu'avec de vrais robots compatibles ROS.

La Robotics System Toolbox permet aux chercheurs, ingénieurs, et étudiants de développer une vaste gamme d'applications de robotique mobile pour l'automobile, l'aérospatiale, la défense, le médical, et les systèmes d'automates industriels dans un environnement complètement intégré comprenant MATLAB, Simulink et les simulateurs et robots compatibles ROS.

- Construire une nouvelle stratégie de navigation basée sur un historique du passage pour éviter de revisiter certaines parties de la surface de Mars

La sélection pour départager les 12 finalistes a été difficile et les équipes de haut niveau ; les critères retenus étant : le nombre de sites visités - le robot devant s'arrêter 3 secondes pour valider le passage - l'évitement des nombreux obstacles et le temps de parcours du robot pour effectuer la mission d'exploration.

« Rendre la robotique ludique »

Les 4 temps forts de la finale

JOPIMARA, UTMars, AM&UROPE4MARS, Esig'inno, Duckgo, iVengers, Vachefolle, Master IPS, Mater Tabula, Adeline's Little Green Men, The Planners, Accrea, les finalistes se sont retrouvés à Innorobo pour s'affronter. Retour sur les temps forts de la grande finale, comme si vous y étiez.

Etape 1 : découverte du robot Rover et session d'entraînement

Les participants découvrent pour la 1ère fois le Rover, robot imprimé en 3D et composé de cartes à bas coût (Arduino DUE et Raspberry Pi), d'une webcam, de 2 moteurs à courant continu ainsi que d'une batterie. Le robot navigue sur Mars en utilisant trois capteurs principaux : la Webcam qui, après traitement, fournit la position des sites détectés, 2 encodeurs qui fournissent une information sur la distance parcourue par les roues et un capteur de distance monté sur un servomoteur qui fournit une information de distance sur les obstacles et bords de l'arène. Les équipes testent donc leur modèle de simulation sur un vrai robot. L'objectif étant d'optimiser leur algorithme d'exploration et de tirer parti de l'environnement de création d'algorithme discret Stateflow afin de découvrir un maximum de sites, représentés par des points verts, tout en évitant les obstacles. La confrontation entre les codes simulés et la réalité n'est pas toujours de tout repos. L'entraînement permet aussi de se familiariser avec les arènes et les différentes cartes de sites de Mars.

Etape 2 : le stress de la qualification

Les participants se confrontent à 3 nouvelles cartes de sites qu'ils tentent d'explorer. Une fois la simulation lancée, les équipes observent le comportement du robot qui devient autonome. L'accent est donc mis sur l'intelligence du robot face à son environnement. Pour chaque parcours, un score est attribué par un juge électronique et affiché sur l'écran central. La tension monte : seules les 4 meilleures équipes concourront les demi-finales ! La pression est à son comble !

Le vainqueur : Master IPS

L'équipe gagnante vient de l'Université Pierre et Marie Curie (Paris). Elle est composée de : Antoine Herbert, Manon Riviere et Corentin Foret, tous étudiants en Master Ingénierie Pour la Santé. Tous les trois ont suivi des cursus très différents. Après une année en médecine, Manon a obtenu une Licence en ingénierie et marketing des produits de santé à l'Université de Lille. Corentin a une Licence de physique fondamentale ainsi qu'une Licence d'ingénierie mécanique de l'UPMC. Antoine, quant à lui, détient une Licence d'ingénierie électronique de l'UPMC.

Ce qui les a incités à participer au concours ? Les aspects pédagogiques et concrets du projet : en apprendre davantage sur la robotique et passer un bon moment.

Etape 3 : plus que 4 équipes !

La complexité s'accentue et les équipes font face à des parcours ardus avec des obstacles plus rapprochés, des sites plus difficilement accessibles. C'est dans la bonne humeur tout en gardant à l'esprit la victoire que les équipes Master IPS et iVengers arrachent leur place en finale !

Etape 4 : le grand gagnant est...

La grande finale est enfin là. Pour cette ultime bataille, Master IPS et iVengers se surpassent et ajustent les modèles pour visiter un maximum de sites et gagner quelques précieuses secondes de temps de parcours. La stratégie de Master IPS fait la différence et permet à l'équipe d'arracher la victoire ! « C'est super... on est très content. Les algorithmes MATLAB et Simulink étaient assez simples d'utilisation même si on n'en avait pas spécialement fait avant, ça ne nous a pas posé de problème de pouvoir les modifier et les optimiser. » déclare Manon Riviere, accompagné d'Antoine Herbert.

Suite de la page 41

sur Windows Mobile. Il nous suffit de créer le dossier suivant : *DeviceFamily-Mobile* et d'y ajouter une vue XAML avec le même nom : MainPage.xaml Fig.10.

Ainsi, lorsqu'on affiche cette vue sur Windows Mobile, c'est le fichier présent dans *DeviceFamily-Mobile* qui est utilisé, et non celui présent à la racine du projet. Les développeurs ont désormais toutes les cartes en main pour pouvoir implémenter des vues adaptées à toutes les familles, tout en gardant une logique commune, unique.

Les nouveautés côté code

Les applications Universal Windows Platform n'offrent pas que des nouveautés concernant le design et l'ergonomie des applications. Elles amènent également plusieurs innovations concernant le code ainsi que l'intégration de nos applications dans leur environnement qu'est Windows. Nous en détaillerons les principales dans la suite de cet article.

Nouveauté d'accès à Windows via Windows.System.Launcher

Dans les applications Win RT, il était possible d'ouvrir des fichiers ainsi que des url à partir de l'application en utilisant les méthodes proposées telles que LaunchFileAsync, LaunchUriAsync. Avec le SDK Windows 10, nous pouvons aller plus loin et ainsi proposer de nouvelles fonctionnalités dans nos applications.

Ouvrir des dossiers (LaunchFolderAsync)

La méthode LaunchFolderAsync (https://msdn.microsoft.com/library/windows/apps/windows.system.launcher.launchfolderasync.aspx), qui fait son apparition, permet comme son nom l'indique de demander l'ouverture d'un explorateur de fichier vers le dossier que nous avons spécifié. Comme pour l'ouverture de fichier, des restrictions sont mises en place. Nous devrons dans certains cas, spécifier dans le Manifest de l'application (via les Capabilities), le droit d'interagir avec le dossier.

Voici un exemple avec le dossier Musique (qui a été au préalable autorisé dans le Manifest) :

```
var musicFolder = KnownFolders.MusicLibrary;
var success = await Launcher.LaunchFolderAsync(musicFolder);
```

Ouvrir des paramètres natifs (LaunchUriAsync)

À partir de la méthode LaunchUriAsync (https://msdn.microsoft.com/library/windows/apps/windows.system.launcher.launchuriasync.aspx), que nous utilisions pour ouvrir une adresse internet, nous pouvons maintenant ouvrir les paramètres natifs de Windows 10. Pour ce faire, Microsoft met à disposition un protocole (ms-settings) qui expose de nombreuses pages de paramètres, une liste est disponible sur le MSDN (https://msdn.microsoft.com/fr-fr/library/windows/apps/xaml/Dn741261.aspx).

Voici un exemple pour ouvrir l'écran de gestion du WiFi sous Windows 10 :

```
var success = await Launcher.LaunchUriAsync(new Uri("ms-settings:network-wifi"));
```

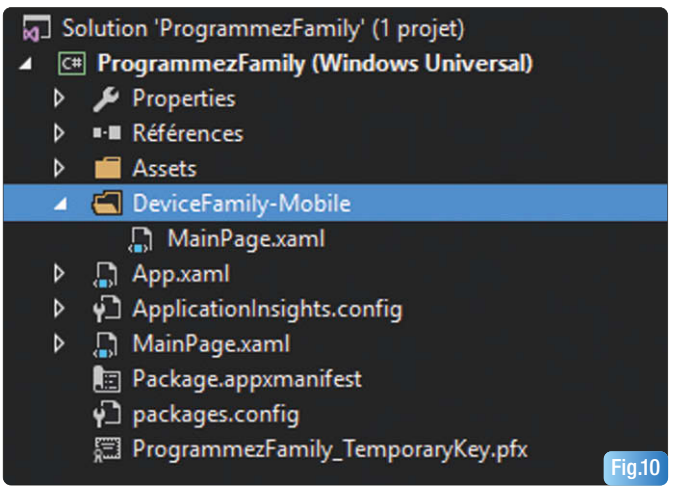


Fig.10

Communication entre applications

Dans les applications WinRT, nous avions peu de possibilités pour échanger avec d'autres applications faisant partie de l'environnement de l'utilisateur. Pour réaliser ce dialogue, nous étions obligés de tricher (en créant un fichier contenant nos données, à implémenter un protocole) et a demandé à Windows d'ouvrir celui-ci. Cependant, ce procédé a ses limites :

- parfois Windows n'autorise pas l'ouverture de l'application,
- l'application recevant le fichier ou le protocole doit être configurée pour pouvoir les lire,
- et surtout, nous ne pouvons pas avoir de retour de l'application appelée, le dialogue ne se fait donc que dans un sens.

Echanger des données entre applications (LaunchUriForResultsAsync)

Avec la communication « App-to-App », Microsoft nous met à disposition une vraie manière de dialoguer avec une autre application de l'environnement utilisateur.

Nous allons prendre l'exemple de notre application « A », qui est le « client », qui va vouloir communiquer avec notre autre application « B », qui joue le rôle de « serveur ». Il faut bien entendu que l'application « B » soit configurée pour pouvoir répondre à l'appel de « A ».

Configuration de B

Il faut dans un premier temps, spécifier dans le manifest de « B » (dans l'onglet déclarations), le support d'un protocole en spécifiant son nom, ainsi que son type de retour (sans, avec ou parfois) (le type de retour ne peut être renseigné qu'en XML).

Ensuite il faut surcharger dans le fichier App.xaml.cs la méthode OnActivated pour obtenir les arguments passés lors de l'appel pour ensuite pouvoir y répondre. Dans mon exemple, j'y réponds dans ma MainPage, je dois donc lui transmettre les arguments.

```
protected override void OnActivated(IActivatedEventArgs args)
{
    Frame rootFrame = Window.Current.Content as Frame;

    // Créer la frame si elle n'est pas encore définie
    if (rootFrame == null)
    {
        rootFrame = new Frame();
        Window.Current.Content = rootFrame;
    }

    // Obtention des arguments passés lors de l'appel
    var protocolForResultsArgs = (ProtocolForResultsActivatedEventArgs)args;

    // Transfert des arguments à la MainPage
    rootFrame.Navigate(typeof(MainPage), protocolForResultsArgs);

    Window.Current.Activate();
}
```

Et dans le OnNavigatedTo de la page MainPage, j'effectue mon traitement :

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    // Obtention des arguments passés depuis le App.xaml.cs
    var protocolForResultsArgs = e.Parameter as ProtocolForResultsActivatedEventArgs;

    if (protocolForResultsArgs == null)
    {
        return;
    }

    // On définit l'objet stockant les valeurs à retourner
    var result = new ValueSet();

    // Si les arguments transmis contiennent celui que l'on attend
    if (protocolForResultsArgs.Data.ContainsKey("Request"))
    {
        string dataRequest = protocolForResultsArgs.Data["Request"] as string;

        if (dataRequest == "Est-ce que tu m'entends ?")
        {
            // On défini une réponse
            result["Response"] = "Hey ho !";
        }
    }

    // On termine l'opération et on retourne le résultat
    protocolForResultsArgs.ProtocolForResultsOperation.ReportCompleted(result);
}
```

Appel à l'application B dans l'application A et exploitation du résultat

Une fois l'application B configurée, le plus dur est fait, il nous suffit maintenant de connaître le PackageFamilyName (présent dans le manifest) de l'application B et d'effectuer l'appel via la méthode LaunchUriForResultsAsync (https://msdn.microsoft.com/en-us/library/windows/apps/dn956688.aspx). Voici comment utiliser cette méthode et échanger des données entre A et B :

```
var options = new LauncherOptions()
{
  // On définit le PackageFamilyName
  TargetApplicationPackageFamilyName = "5f926382-faf0-4039-9847-e085b01b248d_jsdmg9dqt3zzj"
};

// On renseigne nos données à échanger
var inputData = new ValueSet();
inputData["Request"] = "Est-ce que tu m'entends ?";

// On lance l'appel avec les données
var result = await Launcher.LaunchUriForResultsAsync(new Uri("infinitesquare://"), options,
 inputData);

// Si le résultat est positif et qu'il contient ce que l'on attend
if (result.Status == LaunchUriStatus.Success &&
  result.Result != null &&
  result.Result.ContainsKey("Response"))
{
  // On affiche à l'utilisateur le retour
  var messageDialog = new MessageDialog(result.Result["Response"].ToString());
  await messageDialog.ShowAsync();
}
```

Il est important de savoir que les données échangées dans l'objet ValueSet ont une taille maximale de 100ko.

Savoir si une application supporte l'appel (QueryUriSupportAsync)

La méthode QueryUriSupportAsync (https://msdn.microsoft.com/en-us/library/windows/apps/dn893515.aspx) peut se combiner à la méthode précédente pour ouvrir des applications et obtenir un retour. Grâce à cette méthode, nous pouvons savoir si une application supporte l'appel qu'on soumet. Un appel est défini par son uri (fichier ou protocole) ainsi que par son type (uri ou uri avec résultat). Contrairement à WinRT, nous pouvons mieux filtrer les cas où l'utilisateur ne disposerait pas de l'application nécessaire, pour ainsi l'aiguiller sur la manipulation à effectuer pour rendre cette fonctionnalité disponible (installer l'application, la fermer, ...). Voici comment s'utilise cette nouvelle méthode :

```
// Obtenir le statut concernant l'ouverture du protocole infinitesquare en mode Result
var appStatus = await Launcher.QueryUriSupportAsync(new Uri("infinitesquare://"),
LaunchQuerySupportType.UriForResults);

var messageDialogContent = string.Empty;

switch (appStatus)
{
    case LaunchQuerySupportStatus.AppNotInstalled:
        messageDialogContent = "Erreur : Auncune application n'est installée";
        break;
    case LaunchQuerySupportStatus.AppUnavailable:
        messageDialogContent = "Erreur : L'application n'est pas disponible";
        break;
    case LaunchQuerySupportStatus.Available:
        // Ne rien faire, OK
        break;
    case LaunchQuerySupportStatus.NotSupported:
        messageDialogContent = "Erreur : L'application ne supporte pas la demande";
        break;
    case LaunchQuerySupportStatus.Unknown:
        messageDialogContent = "Erreur : inconnu";
        break;
}

if (!string.IsNullOrEmpty(messageDialogContent))
{
    var messageDialog = new MessageDialog(messageDialogContent);
    await messageDialog.ShowAsync();
}
```

Gestion des modes d'affichage (plein écran, fenêtré, tactile ou non)

Dans certains cas, il peut être bon de savoir comment l'utilisateur se sert de notre application, voire d'agir sur l'affichage de l'application pour proposer une meilleure expérience utilisateur.

Dans les UWP nous pouvons savoir si l'application est utilisée en mode tablette (touch) ou en mode classique (clavier, souris). Pour se faire la propriété UserInteractionMode de UIViewSettings nous est grandement utile.

Voici un exemple de comment l'utiliser :

```
var userInteractionMode = UIViewSettings.GetForCurrentView().UserInteractionMode;

switch (userInteractionMode)
{
    case UserInteractionMode.Mouse:
        Debug.WriteLine("Utilisation de la souris et du clavier.");
        break;
    case UserInteractionMode.Touch:
        Debug.WriteLine("Utilisation du mode touch.");
        break;
}
```

Outre le fait de savoir si une application est utilisée en mode touch/souris, nous pouvons savoir si elle est affichée en plein écran et également agir sur son affichage. Nous pouvons effectuer plusieurs actions sur l'affichage de l'application :

- entrer en mode plein écran,
- sortir du mode plein écran,
- définir la taille de la fenêtre de l'application.

Pour interagir avec la fenêtre de l'application, nous devons obtenir la fenêtre courante en utilisant la méthode GetForCurrentView (https://msdn.microsoft.com/en-us/library/windows.ui.viewmanagement.applicationview.getforcurrentview.aspx) qui permet d'obtenir la vue active.

```
// Obtention de la vue active
ApplicationView applicationView = ApplicationView.GetForCurrentView();

// IsFullScreen permet de savoir si l'application est en plein écran
if (!applicationView.IsFullScreen)
{
    // Essayer de passer en plein écran
    var successFullScreen = applicationView.TryEnterFullScreenMode();
}
else
{
    // Sortir du mode plein écran
    applicationView.ExitFullScreenMode();
}

// Essayer de redimensionner la fenêtre à la taille voulue
var successResize = applicationView.TryResizeView(new Size(400, 400));
```

Toast Notifications

Les notifications elles aussi évoluent. Sous WinRT, nous ne pouvions qu'utiliser les notifications par défaut. Hors à part le texte et l'image celles-ci ne sont pas personnalisables. Avec Windows 10, la personnalisation des notifications est devenue très large.

Une notification est constituée de 3 parties :

- La partie « Visual » : ici il n'y a qu'un seul template de base que nous pouvons personnaliser en ajoutant/modifiant/supprimant l'icône, des images, du texte ... Ce visuel s'adapte automatiquement aux différentes plateformes (smartphone, PC, tablette, Xbox).
- La partie « Actions » : elle située en dessous de la partie « Visual » et peut accueillir au maximum 3 boutons de type « Input » et/ou une case de texte pour permettre à l'utilisateur d'interagir avec la notification et donc avec votre application.
- La partie « Audio » : comme sur Windows Phone 8.1, la partie audio reste inchangée, elle peut jouer un son personnalisé depuis l'application (ms-appx, ms-appdata).

Grâce à cette personnalisation et notamment les actions, la notification peut dialoguer avec l'application de plusieurs manières :

- Ouvrir l'application au premier plan avec une action spécifique,
- Ouvrir l'application en fond,
- Ouvrir une autre application,
- Effectuer une action système qui se limite aux alarmes et aux rappels (pour le moment).

Il est également possible de spécifier le type de notification que l'on envoie, appelé scénario. En fonction de celui-ci, la notification ne sera pas gérée pareil par Windows qui pourra la mettre en plein écran, gérer l'audio différemment ... Les scénarios possibles sont à spécifier dans le XAML sur l'élément racine toast et peuvent prendre les valeurs (default, alarm, reminder, incomingCall).

Cortana

Cortana la célèbre assistante présente dans Windows 10 peut également être utilisée dans vos applications ! En l'intégrant dans notre programme, nous donnons la possibilité aux utilisateurs d'interagir avec celui-ci sans avoir besoin de le lancer. Par le biais de Cortana, l'utilisateur peut effectuer des actions simples avec notre application. Prenons le cas d'un logiciel de réservation d'hôtel, l'utilisateur demande à Cortana un hôtel proche. Celle-ci pourra lui faire des propositions à partir des réponses générées par notre programme. Il pourra ensuite aller jusqu'à la réservation sans même avoir ouvert le logiciel ni avoir touché à son clavier !

De plus, si notre application est paramétrée pour interagir avec Cortana, celle-ci peut naviguer, effectuer des actions, écrire dans l'application. Les scénarios peuvent être assez vastes.

Dans la plupart des cas, Cortana va permettre de gagner du temps, et également réduire les efforts de l'utilisateur (rechercher ce qu'il a besoin, prendre en main l'application ...)

Cette nouveauté demande donc un développement particulier, l'intégration étant assez longue, nous n'allons pas nous y attarder dans cet article. Cependant, les personnes intéressées et qui souhaitent mettre à profit Cortana dans leurs applications, trouveront la marche à suivre sur MSDN (https://msdn.microsoft.com/library/windows/apps/xaml/dn974230.aspx).

Conclusion

Windows 10 apporte de nombreuses nouveautés à nous, développeurs, pour améliorer toujours plus l'expérience utilisateur de nos applications. Du relative panel à la split view en passant par les nouveaux protocoles d'accès à Windows et aux applications tiers, nos applications vont avoir de moins en moins de limites (comparé aux limites que nous avions avec WinRT).

Pour nous développeurs, les nouveaux ajouts vont nous permettre de simplifier notre XAML, et de simplifier également les déploiements puisqu'un seul package est généré pour toutes les plateformes (tablette, smartphone, Xbox, PC, IOT ...). Les particuliers quant à eux, vont être enchantés de par ces nouveaux contrôles, par l'intégration de Cortana et la possibilité de contrôler l'application par la voix. Tandis que les professionnels, eux, vont pouvoir réfléchir à des processus métiers qui mettent les applications Windows 10 au cœur de leur métier.

Il est donc temps de vous lancer dans le développement d'applications Windows 10 pour tout support et de mettre en place les différentes nouveautés énoncées ci-dessus.

Nouveautés autour de la cartographie

Avec Windows 10, Microsoft enrichit son SDK avec plus de 2500 fonctionnalités ! La partie cartographie n'y échappe pas et les améliorations qui arrivent ouvrent davantage de scénarios pour nous développeurs !

Julien LO PRESTI
SOAT

Pour rappel sous Windows/Windows Phone 8.1, les services cartes, c'était plutôt ça : Fig.1. Il faut l'avouer entre les services Bing, Here et les applications en preview, il fallait toujours jongler et combiner tout cela pour en arriver à notre besoin. De plus les services n'étant pas tous disponibles sur les 2 plateformes, il était compliqué de partager son code. Aujourd'hui avec les nouvelles API et avec l'uniformisation des OS, nous avons accès à un seul et unique service/contrôle XAML disponible sur toutes les plateformes !

La plateforme Maps sous Windows 10 se résume en 1 image : Fig.2.

La plateforme Maps se repose donc sur Bing et Here, et inclut toutes les fonctionnalités de ces 2 services. Dans les nouveautés on y retrouve le mode offline pour le Desktop/Tablette qui était jusque-là réservé aux téléphones, le mode 3D et Streetside qui est ni plus ni moins que le Streetview à la sauce Microsoft !

Back to basic : les permissions

Commençons par la géolocalisation, qui subit un changement au niveau de la gestion des permissions.

Sous Windows Phone 8.1 : l'autorisation d'utiliser la localisation se faisait lors de l'installation; c'était prompt qui apparaissait lors de l'installation d'une appli mais que personne ne lisait. Il fallait également gérer le premier accès en affichant un prompt dans notre application, gérer un paramètre dans celle-ci pour connaître son état, et, enfin prendre en compte le paramètre présent dans les paramètres du téléphone!

Sous Windows 8.1 : le système s'améliore, l'autorisation se fait automatiquement lors du premier accès à la localisation en affichant un prompt dans notre application. La gestion des permissions se fait via les paramètres de l'OS avec la possibilité de gérer l'accès de façon global. Pour gérer le cas où la personne aurait révoqué la permission nous pouvions utiliser un beau try/catch !

Sous Windows 10, le système s'améliore encore un peu plus (surtout pour Windows Phone en fait), on se rapproche du fonctionnement de Windows 8.1 avec en plus la possibilité de connaître le statut de la permission ! La best practice pour accéder à la position est :

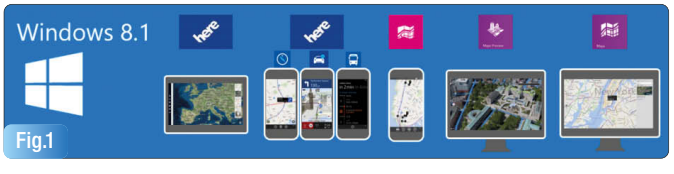


Fig.1

```
private async Task GetPositionAsync()
{
  // Obtention de la permission et affichage d'un prompt si c'est le premier accès
  var accessStatus = await Geolocator.RequestAccessAsync();

  switch (accessStatus)
  {
    case GeolocationAccessStatus.Allowed:
      //L'utilisateur a accepté l'accès à sa position
      Geolocator geolocator = new Geolocator();
      Geoposition pos = await geolocator.GetGeopositionAsync();
      break;
    case GeolocationAccessStatus.Denied:
      DisplayErrorMessage();
      //L'utilisateur n'a pas accepté ou a revoqué l'accès à sa position
      break;
    case GeolocationAccessStatus.Unspecified:
      //La permisssion n'a pas encore été spécifiée
      break;
  }
}
```

N'oubliez pas d'ajouter la capability, au moment de l'écriture de l'article, l'interface pour le manifeste n'existe pas. Il faut éditer directement le XML

```
<Capabilities>
  <Capability Name="internetClient" />
  <DeviceCapability Name="Location"/>
</Capabilities>
```

On peut également rediriger l'utilisateur sur l'écran des permissions afin qu'il active le droit dans le cas où il l'aurait refusé précédemment.

```
await Launcher.LaunchUriAsync(new Uri("ms-settings://privacy/location", UriKind.Absolute));
```

Il faudra donc penser à repasser sur vos applications si vous souhaitez les mettre à jour en prenant en compte ce changement.

- Supprimer vos prompts ainsi que le paramètre pour gérer la localisation dans votre application Windows Phone,

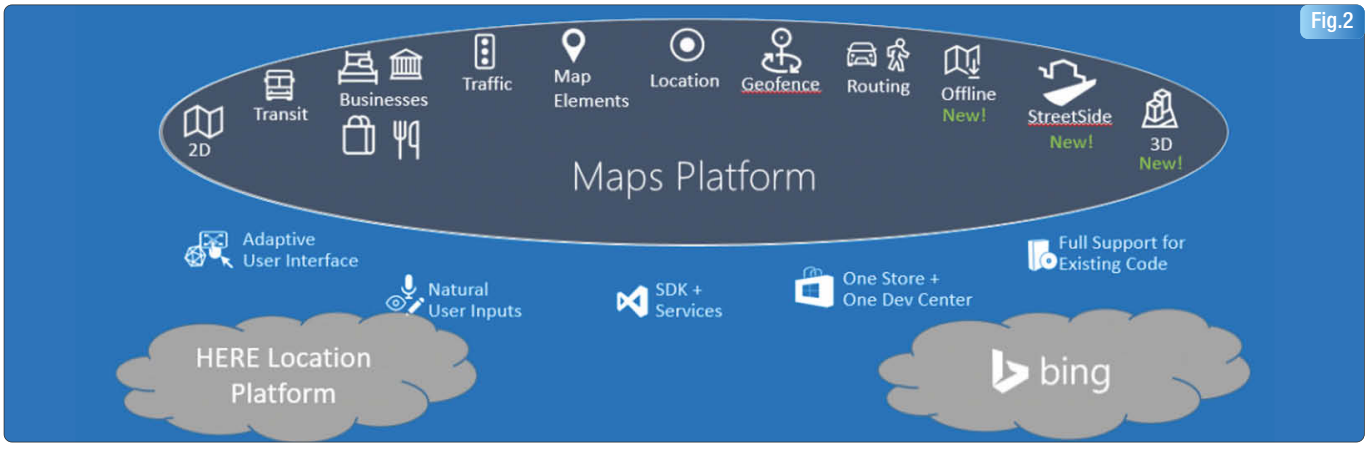


Fig.2

- Utiliser Geolocator.RequestAccessAsync() pour demander/connaître l'état de la permission,
- Proposer à l'utilisateur d'aller sur la page des permissions dans le cas où celle-ci a été refusée.

Quoi de neuf pour le MapsControl : les nouvelles propriétés

Le MapsControl a été mis à jour pour bénéficier des services HERE ! De nouvelles propriétés font leur apparition :

- **TransitFeaturesVisible**: permet d'afficher ou pas, les arrêts de tram/metro/rer/aeroport etc... sur la carte,
- **BusinessLandmarksVisible** : permet d'afficher, ou pas, les POIs genre musée/parc,
- **ActualCameraChanged/ActualCameraChanging** : 2 évènements pour agir lorsque on effectue un pan/zoom/tilt,
- **ZoomInteractionMode/PanInteractionMode/RotateInteractionMode/TiltInteractionMode** : pour autoriser ou pas l'usage de ces mouvements sur la carte,
- **MapElementClick/MapElementPointerEntered/MapElementPointerExited** : des évènements pour interagir avec nos pushpins,
- **CollisionBehaviorDesired** : pour déterminer le comportement d'affichage lorsque 2 pushpins se chevauchent,

De nouvelles API concernant la 3D et StreetSide sont accessibles, j'en parlerai dans la dernière partie de l'article. Pour obtenir votre MapServiceToken ce n'est plus via le Dev Center mais directement sur le site Bing. Il faudra se rendre à l'adresse https://www.bingmapsportal.com/Application et déclarer votre application. On a toujours la possibilité d'ouvrir l'application maps pour afficher un pushpin, tracer un itinéraire via les 2 uris suivantes:

- bingmaps:?cp=40.726966~-74.006076 // pour ouvrir maps centré sur les coordonnées
- ms-drive-to:?destination.latitude=47.64514137971944&destination.longitude=-122.1419647336014&destination.name=Redmond, WA // pour tracer un itinéraire

Je vous conseille d'aller jeter un coup d'œil pour connaître tous les paramètres possibles :

- bingmaps : https://msdn.microsoft.com/en-us/library/windows/apps/xaml/jj635237.aspx#examples
- ms-drive-to : https://msdn.microsoft.com/library/windows/apps/dn614997.aspx

Scene API et Streetside ...

Passons maintenant à 2 nouvelles fonctionnalités qui font leur apparition, il s'agit de Scene API et StreetSide. Scene API permet de manipuler la camera pour la diriger sur un point précis. Pour l'utiliser, c'est super simple :

```
private async Task ShowSoatBuilding()
{
  //on vérifie que le device support la 3D pour afficher la carte en 3D :)
  if (myMap.Is3DSupported)
  {
    this.myMap.Style = MapStyle.Aerial3DWithRoads;
    var status = await Geolocator.RequestAccessAsync();
    if (status == GeolocationAccessStatus.Allowed)
    {
      var gl = new Geolocator();
      var currentPosition = await gl.GetGeopositionAsync();
      //on fait du geocoding pour récupérer les coordonnées de l'adresse de Soat
      var points = await MapLocationFinder.FindLocationsAsync("87, Quai Panhard et Levassor, 75013 Paris", currentPosition.Coordinate.Point);
      if (points.Status == MapLocationFinderStatus.Success && points.Locations.Any())
      {
        var soatCoordinate = points.Locations[0].Point;
        //on initialise la scene en spécifiant :
        // - les coordonnées du bâtiment
        // - la distance entre le bâtiment et notre point de vision
        // - la direction du point de vue : 0 ou 360 = Nord, 90 = Est, 180 = Sud, and 270 = Ouest.
        // - la hauteur du point de vue : 90 on regarde vers l'horizon et 0 on regarde vers le bas
        MapScene soatBuildingScene = MapScene.CreateFromLocationAndRadius(soatCoordinate, 150, 00, 45);

        //on affiche la scene sur la map avec une belle animation!
        await myMap.TrySetSceneAsync(soatBuildingScene, MapAnimationKind.Bow);
      }
    }
  }
}
```

Concernant Streetside je trouve que c'est LA fonctionnalité qui permet d'avoir une expérience cartographique riche. Et en plus, c'est hyper simple à utiliser, alors ne vous en privez pas si ça a du sens dans votre application !

```
private async Task ShowSoatBuildingStreetsideExperience()
{
  var status = await Geolocator.RequestAccessAsync();
  if (status == GeolocationAccessStatus.Allowed)
  {
    var gl = new Geolocator();
    var currentPosition = await gl.GetGeopositionAsync();
    var points = await MapLocationFinder.FindLocationsAsync("87, Quai Panhard et Levassor, 75013 Paris",
        currentPosition.Coordinate.Point);
    if (points.Status == MapLocationFinderStatus.Success && points.Locations.Any())
    {
      var soatCoordinate = points.Locations[0].Point;
      //on récupère le streetside si la fonctionnalité est disponible à cette adresse
      StreetsidePanorama panoramaSoatBuilding = await StreetsidePanorama.FindNearbyAsync(soatCoordinate);

      if (panoramaSoatBuilding != null)
      {
        streetside.Visibility = Visibility.Visible;
        //on affiche streetside sur la carte
        streetside.CustomExperience = new StreetsideExperience(panoramaSoatBuilding)
        {
          OverviewMapVisible = false,
          ExitButtonVisible = false,
          ZoomButtonsVisible = false
        };
      }
    }
  }
}
```

Le code précédent permet d'obtenir ce type de navigation cartographique : Fig.3.


Fig.3

Conclusion

Avec l'uniformisation des OS, Microsoft nous facilite l'usage des API de la cartographie. L'unique contrôle Maps permet plus de partage de code et fournit une expérience riche à l'utilisateur notamment avec les nouvelles fonctionnalités 3D, Scene et Streetside. On appréciera également de meilleures performances sur la partie Desktop, qui était jusque-là assez pauvre.

DATABINDING COMPILÉ

Windows 10 apporte de nombreuses nouveautés aux développeurs d'applications XAML, notamment du côté du databinding. Il s'agit probablement d'une des fonctionnalités les plus importantes (en tout cas ma préférée !), qui vient gommer le désavantage du binding actuel : le manque de performance.

Cyril Cathala

Le binding actuel

```
<TextBlock Text="{Binding FullName, Mode=OneWay}" />
```

Tous les développeurs XAML connaissent cette syntaxe pour **connecter les données entre l'interface graphique et le modèle de données**. Mais concrètement, comment ça marche ?
Au runtime, le framework XAML utilise la ***réflexion*** pour retrouver les propriétés bindées et s'abonne aux événements adéquats pour être **notifié en cas de modification**.

Plusieurs soucis :

- Si on change le nom d'une propriété (par exemple FullName devient Name), le binding ne fonctionne plus. Et on ne le constate qu'au runtime, aucun avertissement, aucune erreur de compilation, rien !
- On le sait, la *reflexion* n'est pas gratuite et impacte les performances. Il est d'ailleurs souvent conseillé de limiter le nombre de bindings, pour éviter de dégrader l'expérience sur les appareils peu puissants.

Mais tout ça, c'était avant !

Binding compilé

Le XAML des Universal Windows Apps apporte la compilation du databinding. Un parseur génère le code nécessaire pour lier l'UI aux données, le tout à la compilation.

```
<TextBlock Text="{x:Bind FullName, Mode=OneWay}" />
```

On voit la nouvelle syntaxe à base de **{x:Bind}**, qui le différencie de son grand frère le **{Binding}**. **Ce binding est fortement typé** et résolu à la compilation, ce qui nous épargne quelques fautes de frappe malheureuses, plus le droit à l'erreur !
Attention, **par défaut le binding compilé est en mode OneTime**, il faut explicitement demander du OneWay ou TwoWay, ce qui diminue les performances.
Pour répercuter les changements en **OneWay**, il faut toujours binder sur une classe qui implémente au choix:

- *INotifyPropertyChanged*
- *Dependency Property*
- *INotifyCollectionChanged / IObservableVector*

Pour du TwoWay, il faudra que la propriété de l'UI soit de type Dependency Property. La TextBox reste un cas à part, puisque le binding sera mis à jour au moment du LostFocus.
Toutes les syntaxes habituelles sont supportées :

- **{x:Bind Model.FullName}** : chemin vers des sous-propriétés,
- **{x:Bind Model.Address[0].Street}** : support de l'index dans une collection,
- **{x:Bind Model.IsVisible, Converter={StaticResource BoolToVisilbity}}** : support des converters,

Note importante : dans la version actuelle du framework (*au 14/05/15*), les converters utilisés avec x:Bind doivent absolument être déclarés dans les ressources de l'*Application*. Autrement vous aurez droit à une exception au runtime.

Adieu datacontext

Avec le {Binding}, la **source par défaut** du databinding est le **DataContext**. Cette propriété provient de la classe FrameworkElement et peut être changée à tout moment. Le compilateur a besoin de connaître le type des objets bindés pour les résoudre et générer le code correspondant. C'est pourquoi il n'était pas possible d'utiliser le DataContext qui est de type object. Dès lors, le **contexte du {x:Bind} est désormais la Page ou UserControl** lui-même, il est fixe et ne peut être modifié. Le binding est compatible avec les champs et les propriétés. La différence majeure se situe lorsqu'on souhaite binder sur un élément graphique de la page. Comme il s'agit d'une propriété de la Page, voici ce que le binding devient :

```
<!-- Binding actuel -->
<TextBlock Text="{Binding IsOn, ElementName=ToggleSwitch1}"/>

<!-- Binding compilé -->
<TextBlock Text="{x:Bind ToggleSwitch1.IsOn, Mode=OneWay}" />
```

On note bien que l'on peut directement accéder au ToggleSwitch, sans passer par ElementName ou RelativeSource=Self.

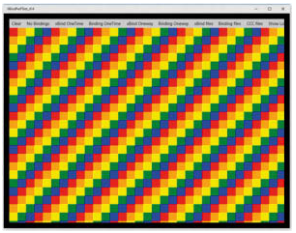
N.B.: le TextBlock est notifié des changements de valeur de la propriété "IsOn", car il s'agit d'une Dependency Property. Les DP possèdent un mécanisme de notification en cas de modification de sa valeur et sont compatibles avec le binding OneWay, comme indiqué précédemment.

Gain de performances

Quelques tests de performance ont été réalisés durant la session de la Build. Les résultats sont largement en faveur du binding compilé, sans surprise.

CPU

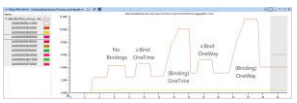
Binding sur le background de 1600 rectangles :



No Bindings	xBind OneTime	Binding OneTime	xBind OneWay	Binding OneWay
27ms	72ms	490ms	71ms	553ms
			Changement : 16ms	Changement : 60ms

On voit bien que x:Bind l'emporte sur le Binding classique, que ce soit pour l'initialisation ou le changement de valeur.

MÉMOIRE



Le **x:Bind** est également **moins gourmand en mémoire** puisqu'en comparaison, le Binding opère tous ses traitements au runtime.
Pour l'anecdote, l'application People, après adoption du binding compilé, a vu une amélioration de 11% au démarrage, et 21% de réduction dans l'utilisation mémoire.

Datatemplates

TYPE À PRÉCISER

Comme x:Bind est fortement typé, le framework a besoin de connaître le type appliqué à un DataTemplate à la compilation. On le spécifie avec un nouveau markup : ***x:DataType***.

RESOURCEDICTIONARY

Pour profiter du **binding dans les ResourceDictionary**, par exemple pour y déclarer des DataTemplates, il faut désormais:

- Lier le ResourceDictionary à une classe dans laquelle le **code behind** pourra être généré,
- S'assurer que le code behind est une partial class qui appelle **InitializeComponent() dans son constructeur,**
- Déclarer le ResourceDictionary dans les MergedDictionaries en l'**instanciant directement**, et non via une ResourceDictionary + Source,

```
  <!-- ResourceDictionary déclaré dans un fichier et contenant un DataTemplate -->
<ResourceDictionary
  x:Class="xBindSampleCS.MyTemplates"
  xmlns:model="using:xBindSampleModel">

  <DataTemplate x:Key="FullNameTemplateInRDFile" x:DataType="model:IEmployee">
    <TextBlock Text="{x:Bind Name}" />
  </DataTemplate>

</ResourceDictionary>
  // Code behind associé au fichier ci-dessus
namespace xBindSampleCS
{
  public partial class MyTemplates
  {
    public MyTemplates()
    {
      InitializeComponent();
    }
  }
}
  <!-- Déclaration du ResourceDictionary via instanciation -->
<UserControl.Resources>
  <ResourceDictionary>
    <ResourceDictionary.MergedDictionaries>
      <!-- avec binding -->
      <local:MyTemplates/>
      <!-- sans binding -->
      <ResourceDictionary Source="filename" />
    </ResourceDictionary.MergedDictionaries>
  </ResourceDictionary>
</UserControl.Resources>
```

Rendu progressif des éléments d'une liste

DANS LES ÉPISODES PRÉCÉDENTS

Lorsqu'on scrolle dans une liste virtualisée, plusieurs choses se déroulent. Tout d'abord, lorsqu'un item disparaît de l'écran, celui-ci va être recyclé: on réutilise ses composants graphiques pour les réafficher avec un contexte de données différent. Ainsi, on évite de réinstancier les contrôles graphiques, une opération coûteuse pour le CPU.
Pour éviter que cette opération de recyclage ne soit trop visible pour l'utilisateur, elle est au maximum effectuée en dehors de l'écran. Problème: si le PC ou téléphone est trop lent et qu'on scrolle rapidement, l'opération devient visible à l'écran, une expérience utilisateur qu'on préférerait éviter.
Pour remédier à ce problème, le SDK Windows 8.1 proposait l'évènement *ContainerContentChanging*. Il était possible de prioriser l'affichage de certains éléments, histoire d'éviter l'effet "page blanche" lorsqu'on scrolle rapidement. Malheureusement, l'implémentation nécessitait un code complexe, souvent au détriment du databinding.

X:PHASE À LA RESCOUSSE

Avec Windows 10, ***x:Phase*** vient à notre rescousse ! Ce merveilleux attribut allège le CPU en **priorisant l'ordre d'affichage des bindings au sein d'un DataTemplate**.

```
  <DataTemplate x:DataType="model:Profile">
 <StackPanel Width="200" Height="100">
  <Image Source="{x:Bind ImageUrl}" x:Phase="1" />
  <TextBlock Text="{x:Bind Name}" />
 </StackPanel>
</DataTemplate>
```

Exemple classique : une liste contient des images et du texte. Il semble logique de prioriser le chargement du texte en premier, puis l'image en second. Non seulement cela améliore les performances lors du scroll, mais on évite en prime de charger inutilement certaines images qu'on passerait trop rapidement.
Le rendu se fera donc dans l'ordre spécifié par les différentes *x:Phase*.
Plusieurs points à noter:

- Les nombres n'ont pas besoin d'être contigus,
- Il est recommandé d'avoir un **nombre de phases raisonnable**, au risque de perdre ses bénéfices.

Binding sur évènements

Oui, ils l'ont fait : il est possible de **binder un événement de l'UI** vers une méthode !

```
  <Button Click="{x:Bind Model.Profiles[0].Coucou}">Fais coucou</Button>
```

Dans ce cas, la méthode "bindée" peut avoir plusieurs signatures:

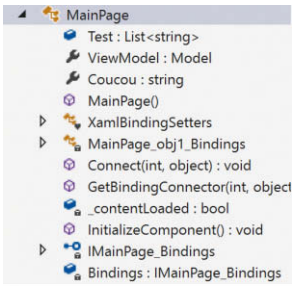
```
  // Aucun paramètre
void Coucou() {…}

// Correspond aux paramètres de l'évènement
void Coucou(object sender, RoutedEventArgs e) {…}
```

```
// Correspond aux paramètres de base / interface de l'évènement
void Coucou(object sender, object e) {…}
```

L'overloading n'est pas supporté, le binding cherchera la première version de Coucou si plusieurs sont déclarées.
Tous les évènements sont supportés, et on voit bien son utilité en **remplacement des commandes**(*ICommand*), habituellement utilisées en MVVM. A noter cependant qu'il n'y a pas de support pour les paramètres de commande, ou la validation via *ICommand.CanExecute*.

Sous le capot

Ok, la syntaxe change et on gagne en performance, en utilisation mémoire, etc. Mais concrètement, comment ça marche ?
Tous les bindings provenant du ***x:Bind*** sont en réalité **générés à la compilation**. Il suffit de se rendre dans le dossier obj/ du projet pour y trouver une classe partielle "XXX.g.cs" étendant la Page/UserControl. C'est cette classe qui est enrichie à la compilation et implémente principalement le mécanisme de binding.
Quand le XAML est chargé, l'évènement *Loaded* signale lorsque le rendu graphique est terminé. Le framework XAML voit l'ajout de l'évènement *Loading*, déclenché avant *Loaded* et au "dernier meilleur moment" pour initialiser le binding.



Pour accéder aux bindings générés, on peut manipuler le champ "Bindings" de la Page. Plusieurs méthodes s'offrent à nous :

- **Initialize()** : appelée au Loading pour initialiser le binding
- **Update()** : utile pour mettre à jour tous les bindings (incluant les OneTime bindings), par exemple suite à un appel asynchrone.
  N.B.: On ne peut pas mettre à jour un binding en particulier, c'est du tout ou rien.
- **StopTracking()** : stoppe la mise à jour automatique des bindings, pratique lorsqu'on fait un traitement de masse. Pour le réactiver, appeler Bindings.Update().

Considérations

La conversion des données se fait automatiquement dans certains cas :

- Vers une string : utilise le .ToString()
- Depuis une string : utilise le parser XAML (ex.: lorsqu'on initialise une valeur de Background avec "Red", automatiquement retranscrit en Colors.Red).
- Autrement : si la conversion nécessite du code, il faudra utiliser des converters

Le mode **OneWay coûte plus cher que OneTime**. La méthode *Bindings.Update()* peut être appelée pour mettre à jour tous les OneTime bindings, ce qui peut s'avérer plus efficace que du OneWay dans certains scénarios.

Et les limitations ?

Pourquoi avoir ajouté une nouvelle annotation au lieu de remplacer le comportement du {*Binding*} actuel ?
La raison est simple, *x:Bind* est légèrement différent et possède plusieurs limitations :

- **Setter de Style** : il n'est pas possible d'utiliser x:Bind dans le Setter d'un Style,
- Comme aucun objet de type Binding n'est généré, les bindings compilés **ne peuvent être générés ou modifiés au runtime** (forcément, ils sont "compilés"...),
- Dynamic incompatible: le binding compilé ayant besoin d'un typage fort, il n'est **pas compatible avec des objets dynamiques,**
- Réutilisation des DataTemplates : les DataTemplates demandent un x:DataType, il n'est donc plus possible de **réutiliser un template d'un modèle à l'autre** (dans le cas où les noms des propriétés sont identiques),
- **RelativeSource=TemplatedParent n'est pas supporté**, x:Bind ne fonctionne pas au sein d'un control template. TemplateBinding reste l'alternative optimisée pour ce genre de scénario.

Une autre limitation peu commode, issue d'un scénario relativement usuel :

```
<DataTemplate>
  <!-- On souhaite réagir au clic du bouton -->
  <Button x:Name="CoucouButton" Click="OnClick" Content="{Binding Name}" />
</DataTemplate>
private void OnClick(object sender, EventArgs args)
{
  // Récupération de l'élément sur lequel on a cliqué (le bouton en réalité)
  var element = sender as FrameworkElement;
  // Récupération du contexte de données
  var profile = element.DataContext as Profile;

  if (profile != null)
  {
    // Traitement
  }
}
```

Comme le DataContext n'existe plus avec *x:Bind*, il est plus difficile de récupérer le contexte de données d'un DataTemplate. Une possibilité serait de plutôt se baser sur le binding sur évènement et déclencher une méthode dans l'objet Profile.
De même, voici un autre scénario classique :

```
  <DataTemplate>
  <! -- Astuce pour changer le contexte de données du Binding -->
  <Button Command="{Binding DataContext.ClickCommand, ElementName=Root}" />
</DataTemplate>
```

Le *x:Bind* ne permet pas de sortir du contexte de binding du DataTemplate qui est imposé par l'attribut *x:DataType*. Il faudra donc ruser autrement parmi toutes les autres possibilités offertes !

Conclusion

On peut réellement parler de **révolution du binding au service des performances**. Toutes ces nouveautés feront le bonheur des développeurs XAML sous Windows 10, attention néanmoins à bien en saisir les limites. Evidemment, le binding classique reste disponible et nous rendra probablement encore quelques services.

Nouveautés du XAML de Windows 10

Pour les applications destinées à son Store, Windows 8 a introduit un nouveau modèle de programmation ainsi qu'un ensemble d'APIs sous forme d'une hiérarchie de classes. Les interfaces graphiques des applications dites modernes peuvent être développées avec trois technologies différentes: XAML, WinJS et DirectX. Windows 8.1 a été l'occasion d'une première mise à jour majeure du XAML Moderne avec des optimisations de performances ainsi que l'introduction de nouveaux contrôles et le support de Windows Phone 8.1.

Alain ZANCHETTA
Développeur dans l'équipe XAML de Windows

La sortie de Windows 10 s'accompagne d'une mise à jour importante de XAML, liée à l'évolution de la plateforme Windows elle-même:

- Accomplissement de la vision des applications Windows universelles capables de s'exécuter aussi bien sur un téléphone portable que sur un PC avec plusieurs écrans haute définition,
- Exécution des applications modernes en mode fenêtré,
- Et un effort renouvelé sur les performances.

Pour les développeurs qui ont choisi les technologies basées sur XAML, présentes depuis longtemps dans Windows avec WPF puis Silverlight, Windows 10 est une excellente nouvelle car l'utilisation de XAML y est omniprésente dans le système avec par exemple le menu démarrer, l'écran de connexion ou encore Spartan, le nouveau navigateur de Microsoft.

Il serait fastidieux d'énumérer toutes les modifications de XAML (quasiment tous les contrôles se voient enrichis), aussi cet article va illustrer les trois axes d'évolution de XAML cités ci-dessus à l'aide d'exemples représentatifs.

Applications universelles

Convergence des contrôles

Windows Phone 8.1 et Windows 8.1 partagent la plus grande partie des contrôles XAML, mais d'une part certains contrôles ne sont disponibles que sur l'une de ces plateformes (typiquement le Pivot qui est un héritage de Silverlight), et, d'autre part, un certain nombre de templates ou de comportement diffèrent. Avec Windows 10, ces différences sont gommées, voici quelques exemples représentatifs :

- Le Pivot n'est plus réservé au téléphone. Par ailleurs, ce contrôle illustre assez bien l'adaptabilité implémentée au sein de XAML : s'il peut afficher tous les en-têtes des PivotItems, il adopte une apparence proche des onglets en mettant en gras l'en-tête du PivotItem sélectionné : Fig.1. S'il ne peut pas afficher tous les en-têtes, la sélection est toujours alignée à gauche du pivot. Par ailleurs, des boutons apparaissent lors du survol par une souris alors qu'ils n'apparaissent jamais lors d'interactions tactiles Fig.2.
- La ListView et la GridView utilisant un ItemsStackPanel ou ItemsWrapGrid supportent les *Sticky Headers* sous Windows et Windows Phone. Il est possible de les désactiver à l'aide de la propriété AreStickyGroupHeadersEnabled:

```
<ListView.ItemsPanel>
  <ItemsPanelTemplate>
    <ItemsStackPanel AreStickyGroupHeadersEnabled="False"/>
  </ItemsPanelTemplate>
</ListView.ItemsPanel>
```

- La réorganisation des éléments d'une liste utilisait deux approches différentes : sous Windows, une ListView ayant CanReorderItems=True et AllowDrop=True pouvait toujours être réorganisée alors que sous Windows Phone, il fallait passer la liste dans un mode spécial (annulé par Back ou dès que la liste perdait le focus) en positionnant sa propriété (non existante sous Windows) ReorderMode à ListViewReorderMode.Enabled.

Sous Windows 10, seul le mode hérité de Windows est supporté, à la fois sur desktop et sur mobile.

VisualState Setters et Triggers

Les applications universelles doivent pouvoir s'exécuter sur différents types de matériels avec des tailles et résolutions d'écran variées. Par ailleurs, l'exécution en mode fenêtré sur le bureau demande aussi aux applications universelles de savoir s'adapter à différentes tailles de fenêtres. XAML possédait déjà la base de l'adaptabilité des interfaces à travers le VisualStateManager et les modèles Visual Studio pour les applications Windows 8.1 s'appuyaient sur celui-ci pour proposer un début de réponse à la problématique des vues à géométrie variable. Néanmoins, la syntaxe du VisualStateManager et des classes associées – VisualStateGroup et VisualState – restait très lourde.

Le XAML de Windows 10 continue à s'appuyer sur le VisualStateManager mais en simplifie la syntaxe et il réduit aussi le code nécessaire aux choix et déclenchements des changements d'état en introduisant deux notions : les *Setters* et les *Triggers*.

Imaginons une page comportant deux parties principales, que l'on veut arranger différemment selon que la fenêtre est plutôt de type paysage ou plutôt de type portrait : Fig.3.

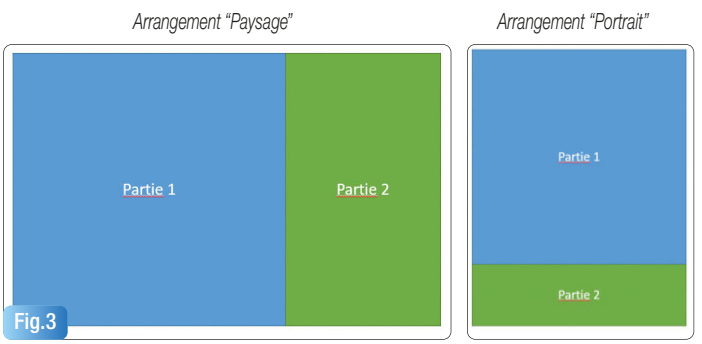


Fig.3

Fig.2

Fig.1

Une manière de réaliser cet arrangement est de définir une grille de 2x2 et de modifier les propriétés Row et Column de l'élément représentant la partie 2. Le XAML de Windows 8.1 pour modifier la position de partie 2 serait similaire à ceci :

```
<VisualState x:Name="Narrow">
  <Storyboard>
    <ObjectAnimationUsingKeyFrames Storyboard.TargetProperty="(Grid.Row)" Storyboard.TargetName="TheBorder">
      <DiscreteObjectKeyFrame KeyTime="0">
        <DiscreteObjectKeyFrame.Value>
          <x:Int32>1</x:Int32>
        </DiscreteObjectKeyFrame.Value>
      </DiscreteObjectKeyFrame>
    </ObjectAnimationUsingKeyFrames>
    <!-- même chose pour la colonne -->
  </Storyboard>
</VisualState>.
```

Les *Setters* de Windows 10 à la syntaxe proche de celle des styles simplifient cette écriture :

```
<VisualState x:Name="Narrow">
  <VisualState.Setters>
    <Setter Target="TheListView.(Grid.Row)" Value="2"/>
    <Setter Target="TheListView.(Grid.Column)" Value="0"/>
  </VisualState.Setters>
</VisualState>
```

La deuxième composante de cette adaptabilité est constituée par les déclencheurs (*Triggers)* : on peut associer des déclencheurs à chaque état. Le runtime XAML active l'état dont un des déclencheurs est actif. Dans notre exemple, l'état Paysage est déterminé par une largeur de fenêtre supérieure à une certaine valeur :

```
<VisualState x:Name="Wide">
  <VisualState.StateTriggers>
    <AdaptiveTrigger MinWindowWidth="800" />
  </VisualState.StateTriggers>
…
</VisualState>
```

La classe StateTrigger peut être utilisée directement dans la définition des états et sa propriété IsActive liée au conteneur XAML ou au modèle associé, par exemple :

```
<VisualStateGroup>
  <VisualState x:Name="ConnectionDisabled"/>
  <VisualState x:Name="ConnectionEnabled">
    <VisualState.StateTriggers>
      <StateTrigger IsActive="{x:Bind IsConnectionEnabled.Value, Mode=OneWay}"/>
    </VisualState.StateTriggers>
    <VisualState.Setters>
      <Setter Target="ConnectedCB.IsEnabled" Value="True"/>
    </VisualState.Setters>
  </VisualState>
</VisualStateGroup>
```

Enfin il est possible de définir ses propres déclencheurs : il suffit de dériver de StateTriggerBase, d'exposer les propriétés qui seront utilisées dans la déclaration des états, puis de s'abonner aux changements permettant d'activer le déclencheur en appelant la méthode SetActive, par exemple au sein de l'événement SizeChanged de l'objet Window :

```
public class SmallWindowTrigger : StateTriggerBase {
  double _maxWindowHeight = double.MaxValue;

  public SmallWindowTrigger() {
    Window.Current.SizeChanged += (sender, e) => EvaluateTrigger();
  }
  private void EvaluateTrigger() {
    Rect bounds = Window.Current.Bounds;
    SetActive(bounds.Height < _maxWindowHeight);
  }
  public double MaxWindowHeight {
    set {
      _maxWindowHeight = value;
      EvaluateTrigger();
    }
  }
}
```

Puis :

```
<VisualState x:Name="Small">
  <VisualState.StateTriggers>
    <local:SmallWindowTrigger MaxWindowHeight="400" />
  </VisualState.StateTriggers>
  <VisualState.Setters>
    <Setter Target="TheListView.Visibility" Value="Collapsed"/>
  </VisualState.Setters>
</VisualState>
```

Lorsque plusieurs états peuvent être actifs, le runtime XAML utilise certaines règles de priorité pour définir quel sera l'état activé. Les déclencheurs personnalisés ont en particulier la préférence par rapport aux déclencheurs standards.

RelativePanel

Le RelativePanel permet de positionner assez facilement des contrôles XAML les uns par rapport aux autres. Il peut s'utiliser seul mais c'est avec les *Triggers* et *Setters* des *VisualStates* qu'il atteint sa pleine mesure. Pour l'utiliser, il faut nommer les différents éléments que le RelativePanel contient et puis les positionner les uns par rapport aux autres à l'aide des propriétés attachées de la classe RelativePanel. L'organisation horizontale illustrée ci-dessous s'implémente de la manière suivante :

```
<RelativePanel HorizontalAlignment="Center" VerticalAlignment="Center">
  <TextBlock x:Name="AddressTextBlock" Text="Address:" FontWeight="Bold"/>
  <TextBox x:Name="StreetTextBox" Header="Street:"  Text="425 15th Ave E"
        RelativePanel.Below="AddressTextBlock" Margin="0,8,0,0" Width="200"/>
  <TextBox x:Name="ZipCodeTextBox" Header="ZipCode" Text="98112"
        RelativePanel.Below="StreetTextBox" Margin="0,8,0,0" Width="100"/>
  <TextBox x:Name="CityTextBox" Header="City" Text="Seattle"
        RelativePanel.RightOf="ZipCodeTextBox" RelativePanel.AlignBottomWith="ZipCodeTextBox"
        Margin="8,0,0,0" Width="200"/>
</RelativePanel>
```

Fig.4.

Les propriétés attachées du RelativePanel peuvent être rangées en trois groupes :

- Above, Below, RightOf, LeftOf: dans l'exemple ci-dessus ZipCodeTextBox est *below* StreetTextBox, ce qui signifie que le haut de ZipCodeTextBox est égal au bas de StreetTextBox
- Align{Left,Top,Bottom,Right,HorizontalCenter,VerticalCenter}With est utilisé pour positionner les éléments dans la direction perpendiculaire ; la gauche de ZipCodeTextBox est alignée sur la gauche de StreetTextBox
- Align{Left,Top,Bottom,Right,HorizontalCenter,VerticalCenter}WithPanel : permet de positionner les éléments par rapport au RelativePanel lui-même.

Deux points importants sont à noter :

- L'arrangement des éléments se fait en séquence,
- Un élément incorporé dans un RelativePanel sans propriété attachée correspondante est aligné en 0 x 0 c'est-à-dire comme ayant les propriétés AlignTopWithPanel et AlignLeftWithPanel actives.

Exécution des applications en mode fenêtré

Positionnement de la fenêtre

Les applications universelles peuvent maintenant essayer de contrôler la manière dont elles sont affichées à l'aide de la classe Windows.UI.ViewManagement.ApplicationView (remarque : celle-ci n'appartient pas au framework XAML et est donc accessible à l'ensemble des applications universelles).

Tracé dans la barre de titre

Il est possible de faire en sorte que le contenu XAML d'une fenêtre s'étende dans la partie réservée à la barre de titre Fig.5.

Cela se fait à l'aide de la classe CoreApplicationViewTitleBar :

```
CoreApplicationViewTitleBar coreTitleBar = CoreApplication.GetCurrentView().TitleBar;
coreTitleBar.ExtendViewIntoTitleBar = true;
```

Il est généralement souhaitable de déclarer un élément XAML comme faisant office de barre de titre : cela permet à l'utilisateur de déplacer la fenêtre en cliquant dans cette zone :

```
<Page…
  <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
    <Grid.RowDefinitions>
      <RowDefinition Height="Auto" />
      <RowDefinition Height="*" />
    </Grid.RowDefinitions>

    <Border x:Name="CustomTitleBar">
      <TextBlock Style="{StaticResource HeaderTextBlockStyle}" Text="XAML in Windows 10"/>
    </Border>
    <!-- Contenu "réel" de la page -->
    …
```

De plus, la CoreTitleBar expose deux événements qu'il peut être nécessaire de traiter selon les scénarios: IsVisibleChanged permet de gérer le masquage de la barre de titre par exemple lors du passage en mode tablette; LayoutMetricsChanged permet de suivre l'évolution de la taille de la barre de titre, au cas où la barre de titre XAML soit destinée à avoir une hauteur standard.

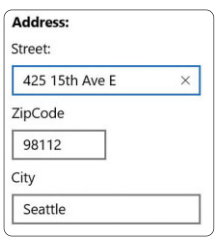
Drag and Drop

Le Drag and Drop (glisser-déplacer) est présent depuis très longtemps dans Windows : d'abord réservé à l'explorateur de fichiers qui reste son utilisation principale, il a connu ses jours de gloire avec OLE2 et les documents composites. Il était totalement supporté avec les technologies Winform et WPF. A l'arrivée de Windows 8, les nouvelles applications s'exécutaient essentiellement en plein écran : le Drag and Drop n'était plus adapté et les « charmes » furent mis en avant pour le partage de données entre applications (même l'irremplaçable copier / coller s'est trouvé un peu en retrait de par l'absence des barres de menus). Windows 10 change à nouveau la donne en rétablissant le fonctionnement en mode fenêtré de toutes applications, modernes et classiques : le Drag and Drop retrouve donc toute sa justification et réapparaît dans les différentes APIs pour les applications modernes.

Un Drag and Drop se traduit par les opérations suivantes :

- **Démarrage du Drag and Drop**
  L'utilisateur clique sur un objet graphique et commence à le déplacer : l'application possédant l'objet, appelée la source, prépare un DataPackage contenant les données partageables.
  Elle peut aussi modifier l'aspect visuel de l'élément déplacé, indiquer quelles opérations sont acceptées (copie, déplacement, lien) voire tout simplement annuler le Drag and Drop.
  Remarque : le Drag and Drop tactile est entièrement supporté. Il faut néanmoins maintenant le doigt appuyé pendant quelques temps avant de commencer à le déplacer afin que le système puisse différentier le Drag and Drop des autres interactions comme le défilement.
- **Survol**
  Lors du survol d'une zone de dépose possible, l'application correspondante (cible) inspecte les données et indique quelle action résulterait de cette dépose : aucune (format inconnu), copie, déplacement ou lien. Elle peut aussi modifier l'aspect visuel de l'élément déplacé ou lui associer un texte. Un exemple de cette modification graphique est donné par le Menu Démarrer : lorsqu'un élément de la liste « Tous les programmes » arrive sur la grille, il prend l'apparence d'une tuile.
- **Dépose**
  Lorsque l'utilisateur lâche l'objet, l'application cible accède au DataPackage et traite les données associées. Comme l'utilisateur peut lâcher l'objet à n'importe quel moment, il est toujours possible que la cible ne comprenne pas les données. Il est aussi possible de lâcher l'objet en dehors de toute cible ou d'interrompre le Drag and Drop en appuyant sur Escape.
- **Finalisation**
  Une fois que l'éventuelle cible a traité les données déposées, l'application source est prévenue de la fin de l'opération et peut à son tour effectuer les opérations nécessaires, en particulier dans le cas d'un déplacement.

Dans tous les cas, il faut noter que l'infrastructure de Drag and Drop est au



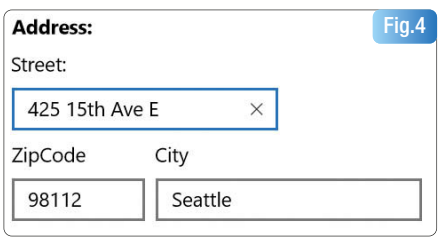


Fig.4

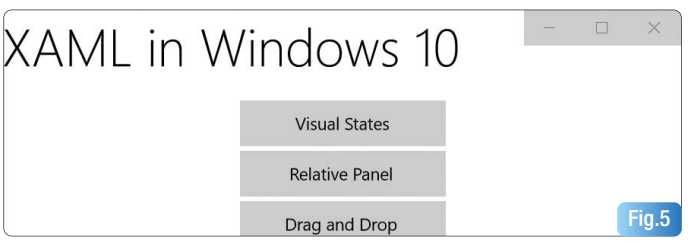


Fig.5

service de l'utilisateur et des applications et ne sert que de vecteur de communication : elle n'interdit pas à l'utilisateur de lâcher l'objet manipulé sur une cible ne sachant pas traiter les données (l'événement Drop ne sera néanmoins pas déclenché dans ce cas), tout comme elle n'est pas en mesure d'imposer la suppression des données dans l'application source si une application cible demande uniquement un déplacement.

Deux classes XAML sont au cœur du Drag and Drop :

- La ListView qui le supportait déjà sous Windows 8.1 et de manière plus limitée (réorganisation uniquement) sous Windows Phone 8.1. Cette classe supporte maintenant le drag and drop inter-applications et se voit donc ajouter l'évènement DragItemsCompleted (qui n'était pas nécessaire quand la cible appartenait nécessairement à la même fenêtre que la source). Même s'il s'appuie sur la même infrastructure, le Drag and Drop de la ListView se veut une solution clés en mains, sans grandes possibilités de personnalisation.
- Tout UIElement peut être source ou cible de Drag and Drop. C'est en se basant sur les évènements de cette classe que le développeur peut vraiment personnaliser l'expérience utilisateur.

Démarrage du Drag and Drop

Le démarrage d'un Drag and Drop peut être automatique si la propriété CanDrag vaut True, ou est explicite via l'appel à StartDragAsync. Dans les deux cas, le démarrage de l'opération se traduit par l'évènement DragStarting pendant lequel la source va remplir le DataPackage qui représente les données échangées mais peut aussi personnaliser le visuel déplacé par l'utilisateur :

```
private async void TheBorder_DragStarting(UIElement sender, DragStartingEventArgs args)
{
  args.Data.SetText(…);
  args.Data.RequestedOperation = DataPackageOperation.Copy;
  args.DragUI.SetContentFromDataPackage();
}
```

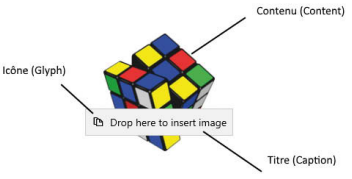
Le visuel peut prendre plusieurs formes :

- par défaut, un instantané de l'UIElement source du D&D est pris,
- DragUI.SetContentFromDataPackage() demande au système de choisir une représentation en fonction des données transférées,
- DragUI.SetContentFromBitmapImage() affiche la BitmapImage passée en paramètre,
- DragUI.SetContentFromSoftwareBitmap() affiche le SoftwareBitmap passé.

Le SoftwareBitmap est une nouvelle classe permettant d'échanger des objets images entre les différentes parties de l'API des applications universelles. Dans le cas de XAML, il est facilement créé à partir d'une RenderTargetBitmap qui permet de générer un bitmap à partir d'un élément XAML appartenant à l'arbre visuel. Dans le cadre de l'évènement DragStarting, il faut utiliser un Deferral pour signifier que le traitement de l'évènement sera asynchrone (puis en signaler la fin) :

```
var deferral = args.GetDeferral();
HiddenTextBlock.Text = text;
RenderTargetBitmap rtb = new RenderTargetBitmap();
await rtb.RenderAsync(HiddenBorder);
IBuffer buffer = await rtb.GetPixelsAsync();
SoftwareBitmap sb = SoftwareBitmap.CreateCopyFromBuffer(buffer,
            BitmapPixelFormat.Bgra8, rtb.PixelWidth, rtb.PixelHeight,
            BitmapAlphaMode.Premultiplied);
args.DragUI.SetContentFromSoftwareBitmap(sb);
deferral.Complete();
```

Survol



Lorsque l'objet déplacé passe par-dessus une cible de Drag and Drop, les évènements existant déjà sous Windows 8.1 sont déclenchés. La cible doit préciser quelle opération elle accepte (sinon, l'événement Drop ne sera pas déclenché) et elle peut à son tour personnaliser le visuel de l'opération via la propriété DragUIOverride de l'évènement :

- Le contenu peut être alimenté à l'aide d'une BitmapImage ou d'un SoftwareBitmap (args.DragUIOverride.SetContentFromSoftwareBitmap()). Il peut être aussi masqué :

```
e.DragUIOverride.IsContentVisible = false;
```

- Le titre peut être modifié ou masqué :

```
e.DragUIOverride.Caption = "Drop here to insert image";
e.DragUIOverride.IsCaptionVisible = true;
```

- Enfin, l'icône peut être masquée mais sa personnalisation se fait uniquement au travers de l'opération acceptée (i.e. il n'est pas possible d'afficher des icônes personnalisées) :

```
e.AcceptedOperation = DataPackageOperation.Copy;
e.DragUIOverride.IsGlyphVisible = true;
```

Dépose

Le Drop se traduit par le déclenchement de l'évènement Drop. La cible peut alors récupérer les données du DataPackage et retourner l'opération effectuée :

```
private async void Border_Drop(object sender, DragEventArgs e)
{
  if (e.DataView.Contains(StandardDataFormats.StorageItems))
  {
     e.AcceptedOperation = DataPackageOperation.Copy;
     foreach (var storageItem in await e.DataView.GetStorageItemsAsync())
     {
       …
     }
…
```

Finalisation

La source peut connaître le résultat de l'opération à l'aide de l'évènement DropCompleted ou via le retour de StartDragAsync dans le cas d'un Drag and Drop déclenché par StartDragAsync.

Amélioration des performances

Liaisons de données compilées

Omniprésent dans toutes les applications XAML, la liaison de données (*Data Binding*) s'est toujours appuyée sur une découverte et une invocation dynamiques des propriétés des objets liés. Dans le cas de .NET, ceci est naturellement implémenté via la réflexion ; dans le cas de C++/CX, l'attribut BindableAttribute ajouté aux classes à lier permet la génération de code offrant ces invocations dynamiques. Cela dit, la dynamique des appels se paie en termes de performances, aussi Windows 10 apporte la notion de liaisons compilées : l'idée est de préparer les appels liés à la liaison de données lors de la compilation, ce qui améliore grandement les performances (et permet au passage de découvrir la plupart des erreurs à ce moment-là au lieu de voir une page blanche lors de l'exécution).

Visual Studio génère donc le code nécessaire aux liaisons lors de la compilation : cela nécessite de connaître le type des objets liés aux composants graphiques. Cette information n'étant pas disponible via le DataContext

sur lequel s'appuie le Data Binding classique, une nouvelle extension XAML – {x:Bind} – apparaît.
Dans une large mesure, {x:Bind} s'utilise de manière similaire à {Binding} avec toutefois quelques nuances. Par exemple :

<Page … <TextBox Text="{Binding CustomFilter}"/> La propriété CustomFilter est lue dans l'objet constituant le DataContext de la TextBox, éventuellement hérité d'un ancêtre de celle-ci	<Page … <TextBox Text="{x:Bind CustomFilter}"/> C'est la propriété CustomFilter de la page qui est lue.
Par défaut, le mode est OneWay <Page … <CheckBox x:Name="UseFilterCheckBox"…/> <TextBox IsEnabled="{Binding IsChecked, ElementName=UseFilterCheckBox}"/>	Par défaut, le mode est OneTime <Page … <CheckBox x:Name="UseFilterCheckBox"…/> <TextBox IsEnabled="{x:Bind UseFilterCheck Box .CustomFilter.IsChecked.Value}"/>
ElementName permet de designer un autre élément de la page comme source de la liaison	Comme x:Name=UseFilterCheckBox déclenche la création d'un membre de ce nom et de type CheckBox au niveau de la page, on peut l'utiliser directement dans le chemin d'accès à la propriété.
Le framework effectue lors de l'exécution la conversion entre le type de IsChecked (IReference<Boolean>) et celui de IsEnabled (Boolean)	Comme la liaison est fortement typée et résolue à la compilation, on doit lire la propriété Value pour obtenir le booléen attendu par IsEnabled

Comme il est dit plus haut, le mode par défaut est OneTime (l'idée est d'avoir par défaut le scénario le plus rapide) mais il est bien évidemment possible de le modifier en OneWay ou TwoWays.
Une alternative est de conserver ce mode et de déclencher la mise à jour des liaisons une fois le modèle mis à jour via Bindinds.Update().
Cette approche peut-être très efficace lorsqu'un objet contenant en particulier des sous-objets est mis à jour de manière globale.
Même si on utilise un autre mode, on peut aussi suspendre temporairement les liaisons via Bindings.StopTracking() et les reprendre une fois le modèle mis à jour :

```
private void Refresh()
{
  Bindings.StopTracking();
  … // Mise à jour du modèle
  Bindings.Update();
}
```

La liaison de données est aussi très utilisée au sein des DataTemplates, typiquement dans le cas des ListViews.
Dans ce cas, le type portant les propriétés liées pour chaque élément n'est ni la page, ni la ListView, ni les ListViewItems, et il faut donc l'indiquer explicitement au compilateur dans le XAML à l'aide de l'attribut x:DataType :

```
<DataTemplate x:Key="GameTemplate" x:DataType="local:Game">
  <Grid>
…
    <TextBlock Text="{x:Bind Result}"…
</DataTemplate>
```

Une nouveauté intéressante liée à x:Bind est la possibilité de lier des évènements à des méthodes de la classe liée, par exemple :

```
<DataTemplate x:Key="GameTemplate" x:DataType="local:Game">
…
   <Button Content="Save" Click="{x:Bind Save}"/>
</DataTemplate>
```

Il y a quelques contraintes sur la signature de la méthode liée : elle peut ne pas avoir de paramètres ou accepter les paramètres de l'évènement correspondant (penser à la génération de l'appel : les types des bases sont donc acceptés). C'est une généralisation de l'approche Command mais qui permet de lier plus d'évènements sans passer par des classes intermédiaires de type *EventToCommand*.

Simplification de l'arbre visuel

En ajoutant la propriété UseSystemFocusVisuals à la classe Control et les propriétés Border, BorderThickness et Padding aux différents panels, XAML permet de simplifier l'arbre visuel en supprimant des éléments Grid, Border ou encore Rectangle et les états VSM associés, ce qui a un impact positif sur performances des applications.

ListView

La ListView étant par construction un contrôle affichant un grand nombre d'éléments, la plupart du temps liés à des données, elle fait l'objet d'améliorations à chaque nouvelle version de XAML.
En termes de performances, on peut noter au moins trois évolutions :

- ChoosingGroupHeaderContainer et ChoosingItemContainer
  Ces deux évènements permettent à l'application de contrôler le recyclage des éléments et des en-têtes de groupes lors de la virtualisation.
- x:phase
  Windows 8.1 avait apporté l'évènement ContainerContentChanging permettant de court-circuiter le Data Binding et aussi de réaliser un affichage progressif. L'exemple classique est l'affichage d'une liste de films : si l'utilisateur fait défiler la liste si rapidement que l'affichage des templates complets n'arrive pas à suivre, on peut afficher dans un premier temps uniquement le nom des films, ce qui permet à l'utilisateur de voir où il en est dans le défilement, et de compléter l'affichage au fur et à mesure que des cycles CPU sont disponibles.
  x:phase permet d'implémenter cette logique d'affichage progressif sur des éléments liés.
- Sélection par bloc
  Lorsqu'une liste possède des milliers d'éléments (par exemple une collection de mp3 ou de photos), la gestion de la sélection par élément individuel pose parfois des problèmes de performance résolus par les méthodes SelectRange et DeselectRange.

Conclusion

Comme on a pu le voir dans cet article, le framework XAML continue d'évoluer pour servir au mieux les besoins des applications Windows en supportant complètement l'approche Applications Universelles tout en poursuivant l'effort permanent d'optimisation des performances. Son utilisation au sein du système Windows lui-même ne fait que croître : le menu démarrer, le centre de notifications, l'écran de connexion ou le nouveau navigateur Microsoft Edge en sont les exemples les plus emblématiques et devraient conforter les développeurs dans la pérennité de cette plateforme.

Android M : Une nouvelle mouture à fort impact !

Un an après la révolution visuelle Material Design apportée par Android Lollipop, les équipes de Google proposent une version 2015 d'Android que l'on peut qualifier d'évolution modeste mais dont les différentes améliorations auront néanmoins un fort impact sur les développeurs et les utilisateurs finaux. Tour d'horizon de cette nouvelle mouture attendue officiellement pour début Octobre 2015 et répondant au doux nom d'Android M.

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

Comme à l'accoutumée désormais, Google aura profité de sa conférence Google I/O courant Mai pour annoncer la future mouture de son système d'exploitation maison. Ainsi, le géant de Mountain View a sorti la version 2015 d'Android en developer preview fin Mai tout en présentant le lot de nouveautés l'accompagnant. Si le cru 2014 aura marqué une véritable révolution visuelle dans le monde Android avec l'apparition du Material Design, Android M se pose plutôt en évolution de l'OS avec un certain nombre d'améliorations qui auront néanmoins un impact fort tant au niveau des développeurs qu'au niveau des utilisateurs finaux.

Une gestion des permissions affinée

Problématique centrale des OS mobiles, la gestion des permissions s'offre un nouveau modèle de gestion avec Android M. Le but derrière ce nouveau modèle étant la simplification et la clarification pour les utilisateurs du processus d'installation et de mise à jour des applications. Ainsi, pour les applications ciblant Android M comme version d'exécution, l'utilisateur n'aura plus à autoriser des permissions particulières à l'installation. Au lieu de ça, les applications devront demander à l'utilisateur d'utiliser une permission au moment précis où ils en ont besoin au sein de leurs applications. Le système proposera alors une boîte de dialogue à l'utilisateur qui pourra accepter ou refuser d'autoriser l'utilisation de cette permission. Il est bon de préciser que les applications supportant ce nouveau modèle de gestion des permissions pourront quand même être exécutées sur des anciennes versions d'Android basées sur l'ancien modèle de gestion.

Fig.1

Groupes de Permissions	Permissions
android.permission-group.CALENDAR	✓ android.permission.READ_CALENDAR ✓ android.permission.WRITE_CALENDAR
android.permission-group.CAMERA	✓ android.permission.CAMERA
android.permission-group.CONTACTS	✓ android.permission.READ_CONTACTS ✓ android.permission.WRITE_CONTACTS ✓ android.permission.READ_PROFILE ✓ android.permission.WRITE_PROFILE
android.permission-group.LOCATION	✓ android.permission.ACCESS_FINE_LOCATION ✓ android.permission.ACCESS_COARSE_LOCATION
android.permission-group.MICROPHONE	✓ android.permission.RECORD_AUDIO
android.permission-group.PHONE	✓ android.permission.READ_PHONE_STATE ✓ android.permission.CALL_PHONE ✓ android.permission.READ_CALL_LOG ✓ android.permission.WRITE_CALL_LOG ✓ com.android.voicemail.permission.ADD_VOICEMAIL ✓ android.permission.USE_SIP ✓ android.permission.PROCESS_OUTGOING_CALLS
android.permission-group.SENSORS	✓ android.permission.BODY_SENSORS ✓ android.permission.USE_FINGERPRINT
android.permission-group.SMS	✓ android.permission.SEND_SMS ✓ android.permission.RECEIVE_SMS ✓ android.permission.READ_SMS ✓ android.permission.RECEIVE_WAP_PUSH ✓ android.permission.RECEIVE_MMS ✓ android.permission.READ_CELL_BROADCASTS

Figure 1 : Groupes de permissions

Concrètement, le modèle de gestion des permissions d'Android M est axé autour des points clés suivants :

- Déclaration des permissions : chaque application déclare au sein du manifest Android les permissions dont elle aura besoin comme cela se faisait précédemment sous Android.
- Groupes de permissions : les permissions sont désormais regroupées en groupes de permissions (figure 1) basés sur les fonctionnalités qu'elles autorisent. Ainsi, le groupe de permissions CONTACTS contiendra les permissions pour lire et écrire des informations sur les contacts de l'utilisateur ainsi que des informations sur les profils.
- Autorisations limitées de permissions à l'installation : les permissions sous Android sont classées suivant différents niveaux de protection. Désormais, le système autorisera par défaut à l'installation toutes les permissions de type PROTECTION_NORMAL listées au sein du manifest. Cela concerne par exemple les permissions liées à l'utilisation d'Internet dans une application. Les permissions d'autres classes devront être autorisées au runtime par l'utilisateur le moment venu. Dans tous les cas, il n'y aura plus d'autorisation des permissions proposées à l'utilisateur durant l'installation d'une application.
- Autorisation des permissions par l'utilisateur au runtime : lorsqu'une application va requêter une permission, le système proposera une boîte de dialogue à l'utilisateur qui pourra l'autoriser ou la refuser. A charge ensuite au développeur de gérer le retour de cette boîte de dialogue pour utiliser ou non la fonctionnalité qui nécessitait cette permission au sein de son application Fig.1.

Ce nouveau modèle de gestion des permissions va simplifier la vie aux utilisateurs finaux mais nécessitera de la part du développeur un certain nombre d'adaptations. Il sera ainsi nécessaire de toujours vérifier si une permission a été octroyée par l'utilisateur avant de l'utiliser. Si ce n'est pas le cas, il faudra lui en demander l'autorisation via la boîte de dialogue système prévue à cet effet. Charge au développeur de gérer proprement dans son code l'absence d'autorisation pour une permission donnée. Enfin, l'utilisateur peut à tout moment révoquer des permissions octroyées à une

application donnée. En pratique, pour vérifier si une application a une permission donnée, il faudra utiliser la méthode checkSelfPermission de la classe Activity en passant en entrée la permission appropriée. Suivant le résultat retourné par cet appel, il pourra alors être nécessaire de requêter l'utilisateur pour obtenir cette permission via la méthode requestPermissions de la classe Activity qui fonctionne de manière asynchrone. Prenons l'exemple d'une application souhaitant lire les contacts d'un utilisateur, le code suivant devra être employé sous Android M au sein d'une activité :

```
// Constante utilisée pour le callback
public static final int MY_PERMISSIONS_REQUEST_READ_CONTACTS = 1;

...

if (checkSelfPermission(Manifest.permission.READ_CONTACTS) !=  PackageManager.
PERMISSION_GRANTED) {
  // si permission non autorisée, alors on la requête
  requestPermissions(new String[]{Manifest.permission.READ_CONTACTS},
      MY_PERMISSIONS_REQUEST_READ_CONTACTS);
  return;
}
```

Suite à l'appel à requestPermissions, le système propose une boîte de dialogue à l'utilisateur. Une fois que ce dernier aura autorisé ou refusé l'octroi de la permission requêtée, le système va invoquer la méthode callback onRequestPermissionResult qu'il est donc nécessaire de surcharger au sein de l'activité où l'appel initial a été réalisé. La constante entière MY_PERMISSIONS_REQUEST_READ_CONTACTS utilisée lors de l'appel devant être utilisée pour vérifier que la réponse que l'on va traiter correspond bien à notre appel :

```
@Override
public void onRequestPermissionsResult(int requestCode, String permissions[], int[] grantResults) {
  switch (requestCode) {
    case MY_PERMISSIONS_REQUEST_READ_CONTACTS:
      if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
        // permission accordée, on peut lire les contacts ...

      } else {
        // permission refusée, il faut désactiver la fonctionnalité qui en dépendait
      }

      return;

    // on peut gérer d'autres appels de permissions ici
  }
}
```

Lors de la mise en place de ces vérifications de permissions, il sera opportun de vérifier que l'utilisateur est bien au minimum en version Android M avant de recourir à ces appels de méthodes qui n'ont pas de sens sur l'ancien modèle de gestion des permissions.

Authentification par empreinte digitale

Avec l'arrivée d'Android Pay et devant le support custom par certains constructeurs de l'authentification par empreinte digitale, les équipes en charge d'Android se devaient de proposer une API dédiée permettant aux développeurs de gérer l'authentification par empreinte digitale. Placée sous le package android.hardware.fingerprint, cette API d'authentification nécessite que le périphérique sous-jacent soit muni d'un dispositif de reconnaissance par empreinte digitale. Pour ce faire la méthode utilitaire suivante pourra être employée :

```
public boolean isFingerprintAuthAvailable() {
  return mFingerprintManager.isHardwareDetected() && mFingerprintManager.hasEnrolled
Fingerprints();
}
```

Si un capteur d'empreinte digitale est bien présent, il reste ensuite à déclarer la permission idoine au sein du manifest Android :

```
<uses-permission android:name="android.permission.USE_FINGERPRINT" />
```

Pour utiliser le capteur d'empreinte digitale (figure 2), il faut ensuite appeler la méthode authenticate de la classe FingerprintManager en précisant en entrée le callback utilisé. Ce dernier devant être une classe dérivant de FingerprintManager.AuthenticationCallback au sein duquel se fait la gestion du retour de l'authentification par scan de l'empreinte digitale Fig.2.

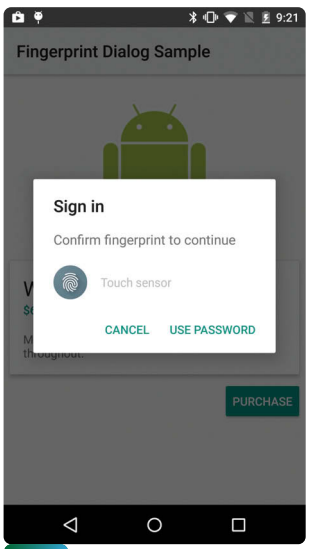


Fig.2 *Authentification par empreinte digitale*

API de Partage direct

Avec la prédominance des réseaux sociaux, le partage de données d'une application est devenu essentiel pour les utilisateurs. Faciliter le partage de données depuis son application est devenu essentiel à sa réussite sur le Google Play Store. Dans ce contexte, Android M propose une nouvelle API rendant le partage plus intuitif et plus rapide pour les utilisateurs finaux.
Pour cela, il est nécessaire de définir une cible de partage direct en charge de lancer une activité spécifique au sein d'une application.
Ces cibles de partage étant exposées aux utilisateurs via le menu Share. Ainsi, la fonctionnalité de partage direct peut permettre de lancer une activité à l'intérieur d'un réseau social en laissant le soin à l'utilisateur de partager le contenu directement à un ami ou à une communauté depuis l'application appelante.
Pour activer cette nouvelle API, il faut dans un premier temps définir un service étendant android.service.chooser.ChooserTargetService et le déclarer au sein du manifest de l'application :

```
<service android:name=".ChooserTargetService"
    android:label="@string/service_name"
    android:permission="android.permission.BIND_CHOOSER_TARGET_SERVICE">
  <intent-filter>
    <action android:name="android.service.chooser.ChooserTargetService" />
  </intent-filter>
</service>
```

Ensuite, pour chaque activité que l'on souhaite exposer au service ChooserTargetService, il faut ajouter un élément <meta-data> de nom android.service.chooser. chooser_target_service lors de la déclaration de l'activité dans le manifest :

```
<activity android:name=".MyShareActivity"
    android:label="@string/share_activity_label">
  <intent-filter>
    <action android:name="android.intent.action.SEND" />
  </intent-filter>
<meta-data
    android:name="android.service.chooser.chooser_target_service"
    android:value=".ChooserTargetService" />
</activity>
```

Là encore, rien de révolutionnaire mais une évolution toujours bienvenue pour améliorer le partage de contenus.

Gestion de la caméra plus poussée

Toujours dans le cadre des évolutions apportées à l'API Android au fur et à mesure des versions, on notera l'apparition de nouvelles APIs liées à la Caméra. Pour utiliser la lumière du flash de la Caméra comme simple torche, il fallait jusqu'alors recourir à différents stratagèmes plus ou moins bien supportés suivant les versions d'Android et suivant les constructeurs. Avec Android M, une API simple est proposée, centrée autour de la classe CameraManager. La méthode setTorchMode de celle-ci permet d'allumer ou d'éteindre la lumière du flash sans avoir à ouvrir la caméra du périphérique. Une nouveauté bienvenue qui aurait du être proposée depuis des lustres déjà à vrai dire.

Précisons également qu'il n'y a pas de verrou pour une application lors de l'activation de la lumière, ce qui implique qu'une autre application peut alors éteindre cette lumière à son tour. De fait, pour être averti des modifications d'état de la lumière de la caméra il faut s'enregistrer auprès du CameraManager via sa méthode registerTorchCallback prenant en entrée un objet de type CameraManager.TorchCallback. C'est la méthode onTorchModeChanged qui sera alors invoquée lors d'un changement d'état de la lumière. Mieux encore, l'API Camera2 est étendue pour supporter le retraitement d'images au format YUV et opaque. Charge au développeur de déterminer si la caméra du périphérique supporte ce type de retraitement en consultant la propriété CameraCharacteristics.REQUEST_AVAILABLE_CAPABILITIES, puis en ouvrant une session de capture de ce type via un appel à la méthode createReprocessableCaptureSession de la classe CameraDevice.

Android M ouvre ainsi la voie à la capture d'images au format RAW, c'est-à-dire dont les données de pixels sont disponibles sans traitement, directement depuis le capteur de la caméra. La récupération d'une caméra de ce type pouvant se faire via le code suivant :

```
private boolean setUpRawCamera() {
  Activity activity = getActivity();
  CameraManager manager = (CameraManager) activity.getSystemService(Context.CAMERA
_SERVICE);

  if (manager == null) {
    // pas de support de l'API Camera 2
    return false;
  }

  try {
    // on liste les caméras
    for (String cameraId : manager.getCameraIdList()) {
      CameraCharacteristics characteristics = manager.getCameraCharacteristics(cameraId);

      // on ne veut que celles supportant le format raw
      if (!contains(characteristics.get( CameraCharacteristics.REQUEST_AVAILABLE_CAPABILITIES),
CameraCharacteristics.REQUEST_AVAILABLE_CAPABILITIES_RAW)) {
        continue;
      }

      StreamConfigurationMap map = characteristics.get( CameraCharacteristics.SCALER
_STREAM_CONFIGURATION_MAP);

      // Taille d'image la plus large possible
      Size largestJpeg = Collections.max( Arrays.asList(map.getOutputSizes(ImageFormat.JPEG)),
new CompareSizesByArea());

      Size largestRaw = Collections.max( Arrays.asList(map.getOutputSizes(ImageFormat.
RAW_SENSOR)), new CompareSizesByArea());

      synchronized(mCameraStateLock) {
        if (mJpegImageReader == null || mJpegImageReader.getAndRetain() == null) {
        mJpegImageReader = new RefCountedAutoCloseable<>( ImageReader.newInstance
(largestJpeg.getWidth(), largestJpeg.getHeight(), ImageFormat.JPEG, /*maxImages*/5));
        }

        mJpegImageReader.get().setOnImageAvailableListener( mOnJpegImageAvailable
Listener, mBackgroundHandler);

        if (mRawImageReader == null || mRawImageReader.getAndRetain() == null) {
        mRawImageReader = new RefCountedAutoCloseable<>( ImageReader.newInstance
(largestRaw.getWidth(), largestRaw.getHeight(), ImageFormat.RAW_SENSOR, /*maxImages*/ 5));
        }

        mRawImageReader.get().setOnImageAvailableListener( mOnRawImageAvailableListener,
mBackgroundHandler);

        mCharacteristics = characteristics;
        mCameraId = cameraId;
      }

      return true;
    }
  } catch (CameraAccessException e) {
    // ...
  }

  return false;
}
```

L'ouverture de l'enregistrement de la caméra sélectionnée au format RAW se faisant ensuite via le code suivant :

```
if (!setUpRawCamera()) {
  return;
}

// Camera Id setté
// Ouverture de la caméra supportant le format RAW
manager.openCamera(cameraId, mStateCallback, backgroundHandler);
```

Introduction des App Links

Le système des Intents au cœur d'Android est un puissant mécanisme permettant aux applications de traiter au mieux le contenu et les requêtes

entre applications. Plusieurs applications peuvent ainsi se déclarer comme matchant un certain pattern URI dans leur filtrage d'intents. Ainsi, quand un utilisateur clique sur un lien Web qui n'a pas de lanceur par défaut, la plateforme propose une boîte de dialogue à l'utilisateur pour sélectionner parmi une liste d'applications ayant déclaré matcher ce type de pattern. Android M étend ce système en proposant aux développeurs d'associer une application avec un domaine Web leur appartenant. Une fois cette association créée, la plateforme pourra automatiquement déterminer l'application par défaut à utiliser pour traiter un lien Web donné sans avoir à demander à l'utilisateur final d'effectuer un choix.
Concrètement, le propriétaire d'un site Web doit définir un fichier JSON statements.json sur son domaine dans un répertoire .well-known de la sorte :

```
http://<domain>/.well-known/statements.json
```

Le contenu du fichier statements.json servant à indiquer à Android l'application devant être utilisée par défaut pour les URLs sous ce domaine est le suivant :

```
[{
  "relation": ["delegate_permission/common.handle_all_urls"],
  "target": {
    "namespace": "android_app",
    "package_name": "com.ssaurel.myapp",
    "sha256_cert_fingerprints": ["6C:EC:C5:0E:34:AE....EB:0C:9B"]
  }
}]
```

Ici, sha256_cert_fingerprints représente l'empreinte SHA256 du certificat de l'application signée com.ssaurel.myapp. Au niveau de l'application Android, il faudra requêter une vérification de tous les liens définis sur le domaine Web dans les éléments du fichier statements.json en plaçant la propriété android:autoVerify à true dans l'intent-filter de l'activité concernée par le lien comme suit :

```
<activity ...>
  <intent-filter android:autoVerify="true">
    <action android:name="android.intent.action.VIEW" />
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.BROWSABLE" />
    <data android:scheme="http" android:host="www.android.com" />
    <data android:scheme="https" android:host="www.android.com" />
  </intent-filter>
</activity>
```

Divers

Outre ces nouveautés marquantes, Android M propose tout un tas de petites évolutions qu'il peut être bon de connaître pour pouvoir en tirer parti le cas échéant. Citons ainsi la mise en place d'un backup automatisé pour les applications et données d'un périphérique comme cela pouvait se faire sur les smartphones Samsung par exemple. L'apparition d'une API de gestion des interactions vocales permettant à une application de savoir si elle a été lancée via une commande vocale et agir en conséquence. En outre, les notifications subissent leur lifting annuel avec la possibilité désormais de savoir quelles notifications envoyées par une application sont toujours actives et donc non traitées par l'utilisateur. Enfin, on notera un certain nombre d'améliorations tournant autour d'Android for Work, signe que le géant de Mountain View vise toujours à imposer son OS comme solution de référence au sein des entreprises.

Conclusion

Loin de la révolution que représentait Android Lollipop, Android M s'inscrit dans la lignée des versions permettant de faire évoluer positivement la plateforme avec pour thématique centrale une simplification du modèle de gestion des permissions. De fait, les impacts d'Android M seront importants tant au niveau des utilisateurs finaux qu'au niveau des développeurs. En parallèle à ces nouveautés autour d'Android, Google aura également annoncé la sortie prochaine de Brillo, un OS dérivé d'Android destiné aux périphériques connectés IoT. Plus intéressant encore, la firme de Mountain View a présenté Weave un protocole de communication unifié et ouvert à destination des périphériques IoT. Pas de developer preview pour Brillo pour le moment, mais beaucoup de promesses et d'attentes qui confirment que Google souhaite également imposer sa domination sur le monde des objets connectés dans les années à venir.

Dans les coulisses de Google BigQuery

Aurélie Vaché
Développeuse Web chez atchikservices à Toulouse, Duchess Toulouse et contributrice pour le blog Duchess France. @aurelievache

"Big Data" != "Hadoop"

Lorsque l'on pense à des technologies liées à la Big Data, on pense de suite à l'écosystème **Hadoop**, ou bien à **Elasticsearch,** ou bien ces temps-ci, beaucoup à **Spark**. Mais il y a un « petit service » de Google qui ne fait pas beaucoup parler de lui qui peut cependant tirer son épingle du jeu dans différents cas de figure. Pour ma part, comme beaucoup d'entreprises, nous avons besoin de faire parler la donnée pour nos équipes en interne et nos clients. La première architecture technique que nous avions mise en place reposait sur l'écosystème Hadoop avec un cluster sous Cloudera d'une dizaine de serveurs hébergés. Les performances étaient au rendez-vous, mais le temps passé pour la maintenance et l'opérationnel est élevé. Lorsque BigQuery fut assez mature et répondait à nos besoins sur le papier, nous avions décidé de créer une branche de notre projet, et de tester cette nouvelle technologie de Google.
Résultats : une centaine d'euros d'économie par mois pour l'électricité, les coûts concernant l'opérationnel sur ce projet ont disparu, et la facture de Google s'élève à environ **0,20$ par mois** pour des performances au final qui sont comparables avec celles dans le cluster Hadoop, et qui répondent toujours au besoin des utilisateurs finaux.

Histoire Fig.1

Comme beaucoup d'outils et de services, BigQuery a été conçu pour résoudre un problème. Les ingénieurs de Google avaient du mal à suivre le rythme de croissance de leurs données. Le nombre d'utilisateurs de Gmail augmentait constamment et était de l'ordre de centaines de millions; en 2012, il y avait plus de 100 milliards de recherches Google effectuées chaque mois. Essayer de donner un sens à toutes ces données prenait un temps fou et était une expérience très frustrante pour les équipes de Google. Ce problème de données a conduit à l'élaboration d'un outil interne appelé **Dremel**, qui a permis aux employés de Google d'exécuter des requêtes SQL extrêmement rapides sur un grand ensemble de données. Selon Armando Fox, professeur d'informatique à l'Université de Californie à Berkeley, *« Dremel est devenu extrêmement populaire chez Google. Les ingénieurs de Google l'utilisent des millions de fois par jour. »* Le moteur de requêtes Dremel a créé une façon de paralléliser l'exécution des requêtes SQL sur des milliers de machines. Dremel peut scanner 35 milliards de lignes sans un index en une dizaine de secondes. Il existe deux technologies que Dremel utilise pour atteindre la lecture d'1 To de données en quelques secondes. Le premier est appelé Colossus : un système de fichiers distribués et parallélisables développé chez Google comme un successeur de Google File System (GFS). Le second est le format de stockage, appelé ColumnIO, qui organise les données d'une manière à ce que ce soit plus facile de les interroger. En 2012, lors du Google I/O, Google a lancé publiquement **BigQuery**, qui a permis aux utilisateurs en dehors de Google de profiter de la puissance et la performance de Dremel.

Type de Fichier	Compressé	Non compressé
CSV	4 GB	◦ Avec de nouvelles lignes dans les chaînes de caractères : 4 GB ◦ Sans nouvelles lignes dans les chaînes de caractèress : 5 TB
JSON	4 GB	5 TB

Fig.3

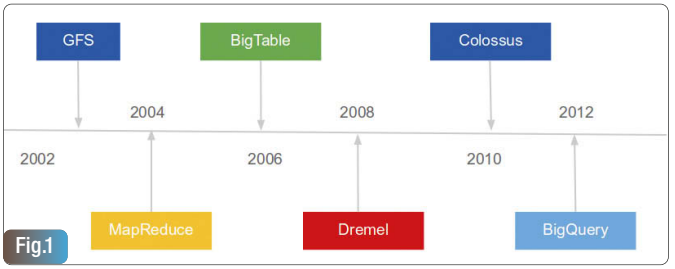


Fig.1

BigQuery, c'est quoi ?

Google BigQuery est une solution de type **AaaS - Analytics as a service** qui repose sur la plateforme Cloud de Google et de ce fait sur sa puissance de calcul. Grâce à BigQuery on peut stocker, effectuer des requêtes et analyser des grands volumes de données : requêter des tables de plusieurs Tera/Peta Octets de données ne prend que quelques secondes. Cette solution s'intègre bien avec Google App Engine mais également avec d'autres plateformes. Techniquement, le service BigQuery est juste un serveur qui accepte les requêtes HTTP et renvoie les réponses au format JSON. Il communique avec Dremel, le moteur de requêtes qui communique avec Colossus Fig.2.

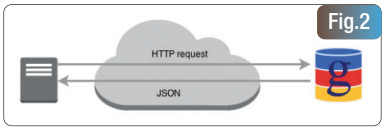


Fig.2

Pourquoi utiliser BigQuery ?

- Sécurisé,
- SLA 99.9% (https://cloud.google.com/bigquery/sla),
- Infrastructure de Google,
- Pas de coût de serveurs, d'opération et de maintenance,
- Moins complexe que l'écosystème Hadoop,
- BigQuery SQL,
- Scalabilité,
- Rapide,
- « Pay only for what you use »,
- Requêtes synchrones et asynchrones,
- Facilité d'interconnexion avec outils tierces.

Inconvénients/Limitations

- « Append-only tables » : on ne peut pas modifier (UPDATE) ou supprimer (DELETE) des entrées dans une table, seulement y ajouter des entrées,
- Des latences réseau peuvent se produire,
- Performant sur des énormes tables (Go, To, Po), moins sur de petites tables.

Quotas

Requêtes : 20 000 requêtes par jour (et jusqu'à 100 To de données)
Chargement/Upload de données :

- Limite quotidienne : 1000 jobs de chargement par table par jour, 10 000 jobs de chargement par projet et par jour (y compris les échecs pour les deux).
- 1000 fichiers max par job de chargement.
- Taille maximum par type de fichier : Fig.3.

Streaming : 100 000 lignes insérées par seconde par table maximum.
A noter que toutes les données qui sont en cache, ne sont pas comptées dans les quotas.
Vous pouvez consulter la liste complète des quotas sur le site de Google BigQuery qui peut être sujet à des changements : https://cloud.google.com/bigquery/quota-policy.

Fig.4

Coûts

Les utilisateurs de BigQuery sont actuellement facturés pour deux choses : le stockage et les requêtes. Les deux coûts sont proportionnels à la taille des données.
Importer des données via un fichier CSV ou JSON est gratuit Fig.4.
Les prix sont sujets à changement donc veuillez vous tenir informés sur le site officiel : https://developers.google.com/bigquery/pricing.
Depuis le 12 Août 2015, les frais d'insertion ont subi une modification : Google ne facture plus par lignes insérées mais par octets insérés, il faut donc compter 0.01$ pour 200 Mo insérés.

Architecture technique

Comme on l'a déjà vu ci-dessus, BigQuery est une implémentation externe de Dremel. Pour essayer de comprendre en quelques mots comment fonctionne BigQuery, il faut connaître les deux technologies de base de Dremel :

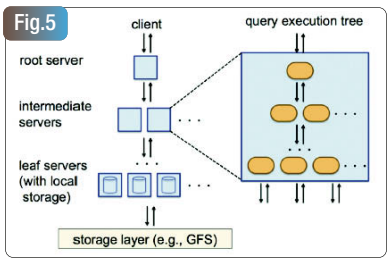


Fig.5

Source : Google BigQuery Technical White-paper

Architecture en arbre Fig.5.

Ce type d'architecture est utilisé pour dispatcher les requêtes et agréger les résultats à travers des milliers de machines en quelques secondes.

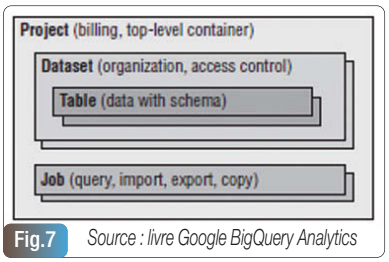


Fig.7 *Source : livre Google BigQuery Analytics*

Base de données orientée colonne Fig.6.

Dremel utilise BigTable et ColumnIO (des bases de données orientées colonne) et stocke les données sous forme de colonne. Grâce à cela, on ne charge que les colonnes dont on a vraiment besoin. Dremel stocke les enregistrements par colonne sur des volumes de stockage différents, alors que les bases de données traditionnelles stockent normalement l'ensemble des données sur un seul volume. Cela permet un taux de compression très élevé.

Fig.8

Schema

date	TIMESTAMP	REQUIRED	Describe this field...
author	STRING	REQUIRED	Describe this field...
title	STRING	REQUIRED	Describe this field...
word_count	INTEGER	NULLABLE	Describe this field...

Composants Fig.7.

- Projects : Toutes les données dans BigQuery sont stockées dans des projets. Un projet est identifié par un ID unique ainsi qu'un nom. Il contient la liste des utilisateurs autorisés, les informations concernant la facturation et l'authentification à l'API.
- Dataset : Les tables sont groupées par dataset qui peut être partagé à d'autres utilisateurs. Un dataset peut contenir une ou plusieurs tables.
- Table : Les tables sont représentées comme ceci : <project>:<dataset>.<table_name>. Pour comprendre cette règle de nommage, regardons de plus près une requête SQL :

```
SELECT date, title FROM [programmez:184.article]
```

Les champs date et title sont sélectionnés dans la table *article* qui est dans le dataset *184* qui est dans le projet programmez.
Lorsque l'on crée une table il faut définir son schéma. Exemple : Fig.8.

```
date:TIMESTAMP, author:STRING, title: STRING, word_count:INTEGER
```

Premiers pas

Si vous n'avez jamais utilisé Google BigQuery, ce paragraphe est fait pour vous, sinon, vous pouvez passer au à la section suivante.

- Il faut en tout premier accéder à la Google API Console : https://console.developers.google.com/ Fig.9.
- Vous pouvez créer votre tout premier projet en cliquant sur le bouton "Créer un projet" : Fig.10.
- Il faut maintenant que vous cliquiez sur le lien "Profiter d'un essai gratuit" afin de pouvoir accéder aux services de Cloud de Google, dont BigQuery. Une fois le formulaire rempli (attention même l'essai gratuit nécessite de fournir les informations d'une carte bancaire), vous devez cliquer sur le projet que vous avez créé pour pouvoir accéder à Google BigQuery puis créer votre premier dataset si vous le désirez, et effectuer vos premières requêtes Fig.11 et 12.

Moyens d'accès

Pour communiquer avec BigQuery il existe plusieurs moyens :

- Via la Google API Console (Web UI : https://bigquery.cloud.google.com/)

Cette interface permet d'effectuer la plupart des opérations dans l'API : lister les tables disponibles, afficher leur schéma et leurs données, partager des dataset avec d'autres utilisateurs, charger des données et les exporter vers Google Cloud Storage. La console de Google est très pratique lorsque l'on veut exécuter/tester/fine-tuner (optimiser) des requêtes. En quelques clics on crée sa requête, on l'exécute et on obtient les résultats en quelques secondes Fig.13.

- En ligne de commande :

Si l'on souhaite par exemple ajouter une colonne dans une table, en ligne de commande c'est possible.

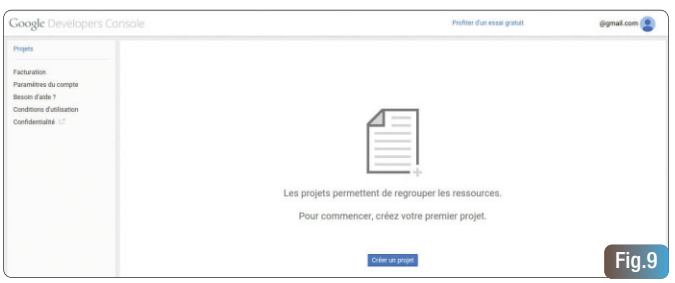


Fig.9

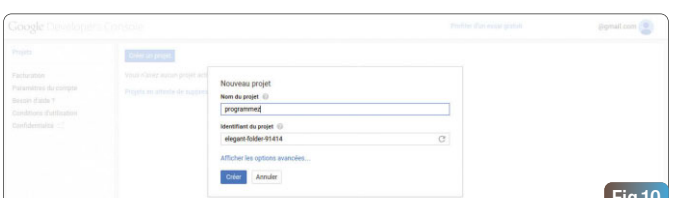


Fig.10

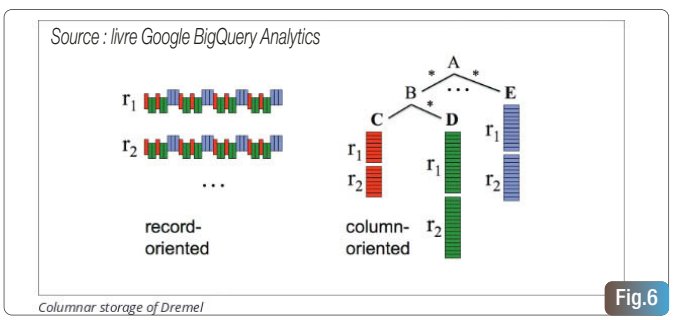


Fig.6

Columnar storage of Dremel

0. Pré-requis : vous devez installer ou avoir Python d'installé sur votre machine. Puis vous devez vous authentifier :

```
gcloud auth login
```

1. Il suffit de récupérer le schéma de la table :

```
bq --format=prettyjson show yourdataset.yourtable > table.json
```

2. Il faut maintenant modifier le fichier, tout supprimer excepté le schéma (garder [{ « name »: « x » ... }, ...]) et ajouter la nouvelle colonne à la fin par exemple.

3. Mettre à jour le schéma de sa table :

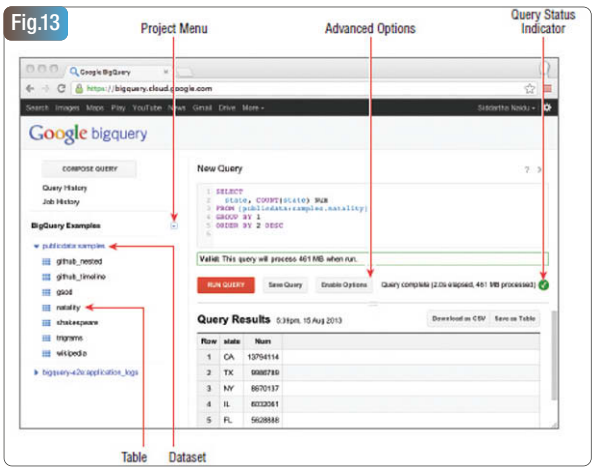
```
bq update yourproject.yourtable table.json
```

- Via les API :

– Il existe une dizaine de librairies clientes qui sont mises à disposition par Google. Elles sont disponibles pour Java, Python, PHP, Ruby, .NET ...
– Et bien entendu il existe une API de type REST.

A noter que les APIs utilisent l'API d'authentification OAuth2.

- Via des connecteurs JDBC (Starschema),
- Via le connecteur Excel,
- Via le connecteur pour Hadoop,
- Via le connecteur pour le nouveau service de Google : Google Cloud DataFlow.



Source : livre Google BigQuery Analytics

Chargement des données

Il existe deux moyens pour charger des données dans une table :

- Soit, via la Google console, lors de la création de la table, on définit une source de données. Ce peut être un fichier au format CSV, JSON ou AppEngine Datastore, que vous uploadez ou que vous importez du Google Cloud Storage Fig.14.

Puis il faut bien penser à définir le schéma de cette table : Fig.15.
Le job qui a procédé au chargement des données et à la création de la table a bien fonctionné : Fig.16.

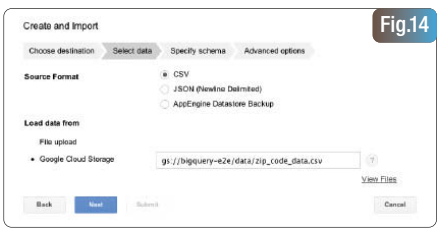
* Soit, via l'API, on peut charger un fichier CSV, ou JSON également, ou bien insérer des entrées dans une table :

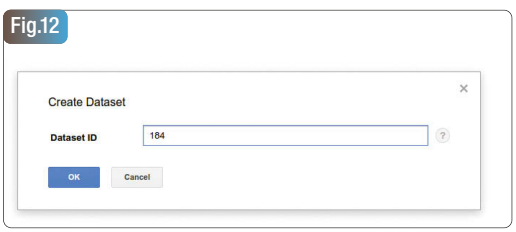
```
List<TableDataInsertAllRequest.Rows> rowList = new
ArrayList<TableDataInsertAllRequest.Rows>();
rowList.add(new TableDataInsertAllRequest.Rows()
   .setInsertId(""+System.currentTimeMillis())
   .setJson(new TableRow().set("adt", null)));

TableDataInsertAllRequest content = new TableDataInsertAllRequest().setRows(rowList);
TableDataInsertAllResponse response = bigquery.tabledata().insertAll(
   PROJECT_NUMBER, DATASET_ID, TABLE_ID, content).execute();
System.out.println("kind="+response.getKind());
System.out.println("errors="+response.getInsertErrors());
System.out.println(response.toPrettyString());
```

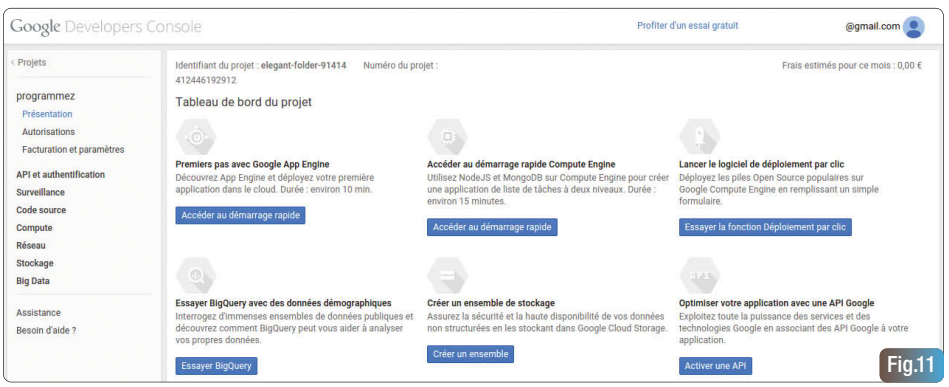
BigQuery SQL

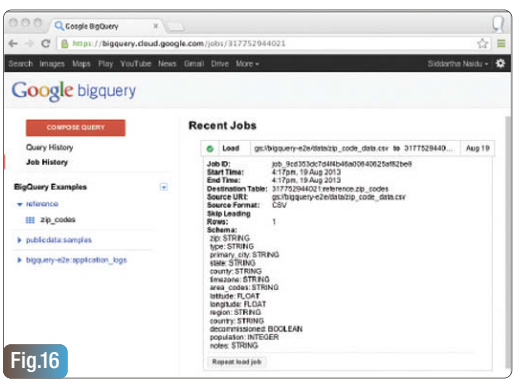
Les requêtes sont écrites en utilisant une variante de l'instruction SQL SELECT standard. BigQuery prend en charge une grande variété de fonctions telles que COUNT, les expressions arithmétiques, et les fonctions de chaîne. Vous pouvez consulter la page Query Reference (https://cloud.google.com/bigquery/query-reference#top-function) pour tout savoir sur le langage de requête de BigQuery. Tout comme le SQL standard, l'instruction SELECT s'écrit de cette manière :

```
SELECT expr1 [[AS] alias1] [, expr2 [[AS] alias2], ...]
   [agg_function(expr3) WITHIN expr4]
   [FROM [(FLATTEN(table_name1|(subselect1)] [, table_name2|(subselect2), ...)]
   [[INNER|LEFT OUTER|CROSS] JOIN [EACH] table_2|(subselect2) [[AS] tablealias2]
    ON <em>join_condition_1</em> [... AND <em>join_condition_N</em> ...]]+
```









```
[WHERE condition]
[GROUP [EACH] BY field1|alias1 [, field2|alias2, ...]]
[HAVING condition]
[ORDER BY field1|alias1 [DESC|ASC] [, field2|alias2 [DESC|ASC], ...]]
[LIMIT n]
;
```

A noter quelques fonctionnalités intéressantes de BigQuery SQL :

- TABLE_DATE_RANGE :

Au lieu de lister toutes les tables quotidiennes dans votre SELECT, comme ceci :

```
SELECT TIMESTAMP_TO_MSEC(rdt), uid, uname FROM myproject.MEETUP_20150317, myproject.MEETUP_20150318, myproject.MEETUP_20150319 WHERE rdt >= '2015-03-17 16:45:00' AND rdt < '2015-03-19 10:50:00' ORDER BY uname LIMIT 500
```

Voici ce qu'il est possible de faire avec cette fonction de wildcard :

```
SELECT TIMESTAMP_TO_MSEC(rdt), uid, uname FROM (TABLE_DATE_RANGE(myproject.MEETUP_, TIMESTAMP('2015-03-17'), TIMESTAMP('2015-03-19'))) WHERE rdt >= '2015-03-18 16:45:00' AND rdt < '2015-03-19 10:50:00' ORDER BY uname LIMIT 500
```

Lorsqu'il s'agit de faire un SELECT sur une liste de tables quotidiennes sur plusieurs jours/semaines/mois, imaginez le gain pour l'écriture de la requête et les performances avec cette fonction.

- REGEXP_MATCH :

Vous pouvez également utiliser des expressions régulières directement dans vos requêtes :

```
SELECT TIMESTAMP_TO_MSEC(rdt), uid, uname FROM [myproject.mytable_20141108] WHERE cl=18 AND rdt >= '2014-11-07 23:00:00' AND rdt < '2014-11-08 22:59:00' AND REGEXP_MATCH(uid, '(?i)^(123456789|55884772)$') AND REGEXP_MATCH(mid, '(?i)^(7423456|3465465415)$') ORDER BY uname LIMIT 123
```

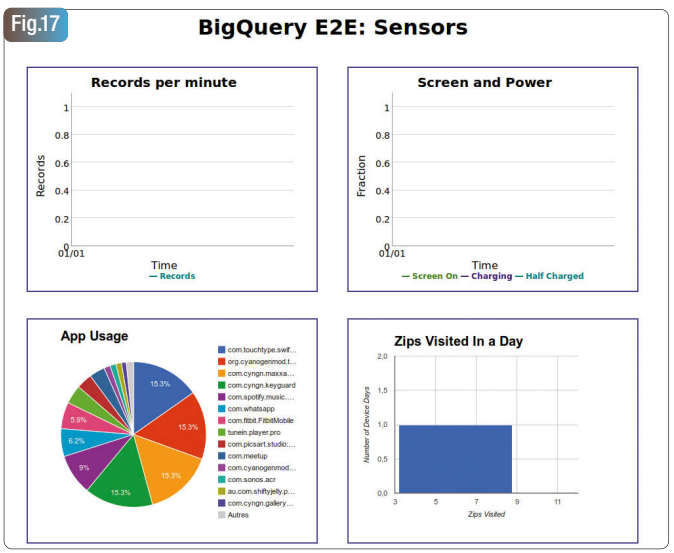
Cas d'utilisation

Il existe plusieurs solutions pour visualiser les données qui sont stockées dans BigQuery.

- Utiliser Google App Scripts pour écrire les requêtes et visualiser les résultats dans des graphiques dans Google Sheets. Un module complémentaire existe sur le Chrome Store également mais je ne l'ai pas encore testé : OWOX BI BigQuery Reports.
- Développer une application (Web/mobile ...) via les APIs et librairies clientes mises à disposition afin de visualiser les données avec l'API Google Charts par exemple.

Voici quelques exemples du rendu que l'on peut avoir : Fig.17, 18 et 19.
Utiliser des outils tiers qui s'interconnectent très bien avec Big Query : Fig.20.

Fig.17

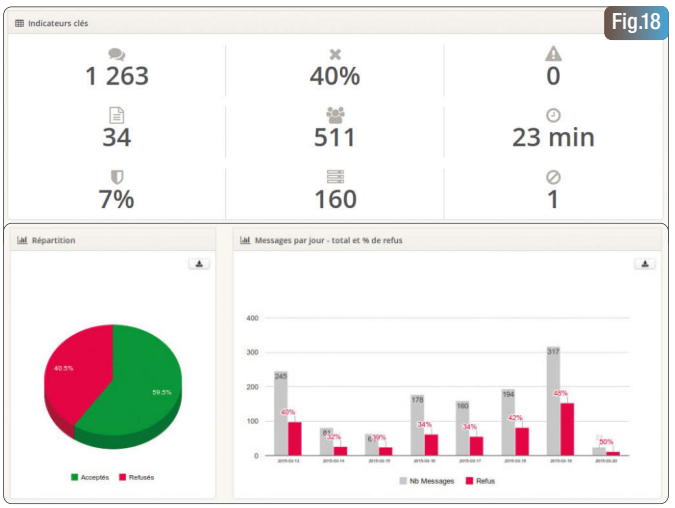


Source : http://bigquery-sensors.appspot.com/console

Conclusion

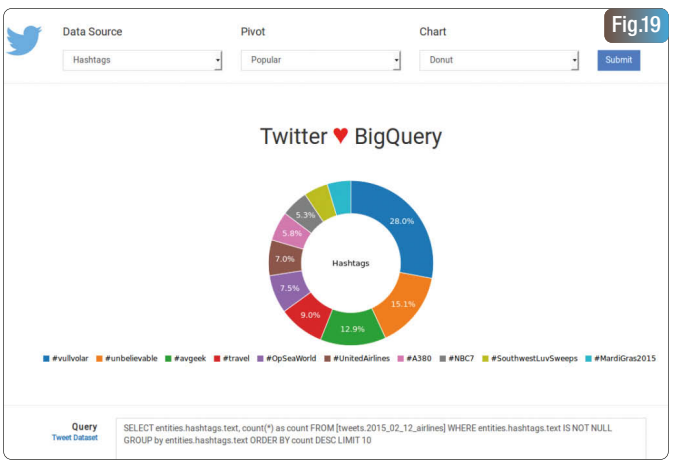
Je tenais à vous parler de Google BigQuery parce qu'il s'agit d'une techno que j'utilise au travail et qui ne fait pas assez parler d'elle à mon goût. Ce n'est pas une solution miracle, cela n'existe pas, mais elle peut répondre à un besoin donné. Le fait de notamment pouvoir interroger la base de données en SQL est vraiment très pratique.
En quelques secondes on peut exécuter sa requête sur la Web UI et obtenir un export en CSV.

Fig.18



Source : Un des outils de reporting et de BI d'atchikservices.

Fig.19



Source : Twitter for BigQuery.

Fig.20

NoSQL: le modèle orienté graphe, un modèle original dans l'univers des bases de données.

Cet article a pour but de décrire les forces et les faiblesses du modèle NoSQL orienté graphe, et de le comparer avec le modèle relationnel classique en s'appuyant sur différents critères.

Fabrice Chapuis
Consultant en informatique.
Spécialiste en bases de données
fchapuis@ip-worldcom.ch

Le terme NoSQL a été utilisé pour la première fois en 1998, il signifie Not Only SQL. Comme ce nom l'indique il ne s'agit pas de la négation du SQL mais plutôt d'une alternative aux bases de données relationnelles (O. Mallassi, 2011). Comme pour d'autres technologies de bases de données possédant leur propre paradigme (base de données XML, base de données objets), la technologie NoSQL possède certaines caractéristiques qui lui sont propres. Dans les années 90 de nouveaux besoins en termes de stockage des données sont apparus; une volumétrie des données qui explose liée à l'Internet, des temps de réponse toujours plus critiques et la disponibilité des services qui ne doit souffrir d'aucune interruption. Par conséquent les grands acteurs du Web, comme Google ou Amazon ont fini par développer leur propre solution de base de données pour stocker des informations récoltées sur toute la toile. Ces nouveaux développements ont donné naissance à la technologie NoSQL. Les technologies NoSQL implémentent plusieurs modèles de stockage. La taxonomie des modèles NoSQL les plus répandus est illustrée dans la Fig.1 ci-dessous :

Le modèle relationnel est un « vieux » modèle qui a été proposé en 1970 par le mathématicien E.F. Codd du centre de recherche IBM à San José. C'est un modèle formel capable de décrire en termes informatiques les données opérationnelles liées à un système d'information. L'information y est structurée dans un modèle prédéfini, le **modèle entité-association** (les mots en gras dans le texte sont expliqués dans le glossaire technique à la fin de cet article) dans lequel les données sont représentées sous forme tabulaire. Le modèle relationnel fournit des mécanismes qui permettent de garantir la consistance des données au sein du système, de gérer les accès concurrents et de garantir les propriétés **ACID** des transactions. Les données sont manipulées par un langage spécifique pour les bases de données, il s'agit du SQL (Structured Query Language). C'est un **langage déclaratif** qui permet de réaliser des opérations sur un ensemble de tables. Le fondement de ce langage repose sur des concepts mathématiques que l'on retrouve dans **l'algèbre relationnel**. Au début des années 80 c'est Oracle qui va proposer la première implémentation commerciale du modèle relationnel. Par la suite, ce modèle va devenir très populaire et beaucoup d'autres constructeurs vont l'implémenter.

Les bases NoSQL sont conçues pour fonctionner dans une architecture distribuées et le paradigme lié à cette technologie vise à optimiser les performances du système à plusieurs niveaux. Les données sont distribuées de manière automatique sur plusieurs serveurs. Afin de garantir le meilleurs temps de réponse possibles, les mécanismes transactionnels sont abandonnés et les données sont dénormalisées. Pour augmenter la puissance du système, il est possible de rajouter des machines dans le « cluster », ces modifications n'ont pas d'incidence sur la disponibilité des données, on parle de redimensionnement horizontal à chaud. Le langage SQL n'est plus le langage utilisé pour interroger ces bases mais d'autres syntaxes sont utilisées.

Le modèle orienté graphe

Ce modèle est inspiré de la théorie des graphes. Les données sont modélisées sous la forme de nœud. Les nœuds sont reliés entre eux par des arcs orientés. On retrouve l'implémentation de ce modèle dans les réseaux sociaux pour lesquels on peut intuitivement percevoir l'intérêt d'utiliser un modèle de graphe, un nœud correspond à un utilisateur et les arcs représentent les liens entre ceux-ci. Des algorithmes de parcours de graphe vont permettre de naviguer à travers cette structure. Le modèle de graphe possède les propriétés suivantes :

- Il contient des nœuds et des relations
- Les nœuds contiennent des propriétés (paire clé-valeur)
- les relations sont nommées et possèdent une direction
- Les relations peuvent aussi contenir des propriétés

Ce modèle présente l'avantage de pouvoir facilement repérer une valeur dans un réseau d'informations interconnectées, par exemple pour trouver le produit qui pourrait potentiellement intéresser un client.

> *Dans la suite de cet article, les exemples proposés s'appuient sur un modèle d'application pour une plateforme de e-commerce. Celle application est modélisée dans le graphe de la Fig.2.*

Ce graphe présente l'historique des ordres pour l'utilisateur Alice. Chaque ordre contient un ou plusieurs produits. Le graphe permet également de connaître la chronologie des achats pour Alice grâce aux relations *most_recent* et *previous*.

Neo4J

Il existe plusieurs logiciels libres qui implémentent le modèle orienté graphe (FlockDB développé par Twitter, Neo4J). Neo4J est répandu et bien documenté, de surcroît il a l'avantage d'être une base de données

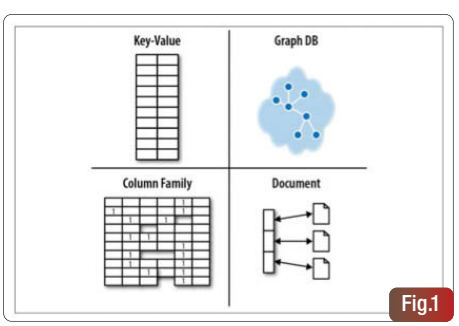


Fig.1

Taxonomie des modèles NoSQL (I. Robinson, J. Webber, E. Eifrem, 2013)

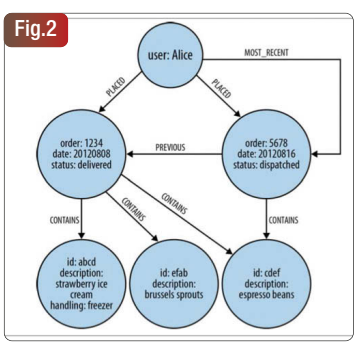


Fig.2

Historique des ordres de l'utilisateur Alice

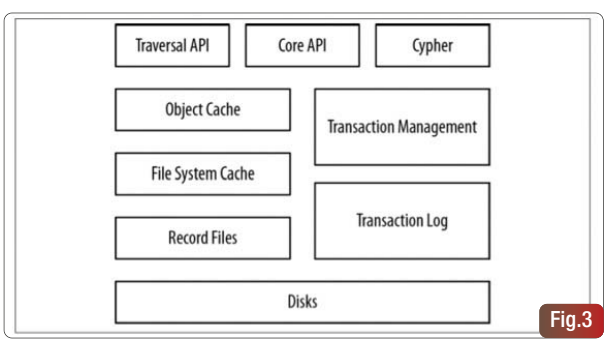


Fig.3

Architecture de Neo4J (I. Robinson, J. Webber, E. Eifrem, 2013)

« ACID compliant » qui stocke des données structurées sous la forme de graphe. Neo4J est implémentée en Java, la dernière version est la version 2.2.2, elle supporte OpenJDK Java 8. La société Neotechnology fournit la version « enterprise » de Neo4J.
Il existe différentes API pour travailler avec Neo4J, comme par exemple Java ou Ruby. Un serveur REST est directement disponible après l'installation de Neo4J. L'architecture simplifiée de la base de données Neo4J est montrée dans la Fig.3 :
Le « transaction log » (journal de transaction) conserve les écritures validées et garantit la durabilité d'une modification dans la base.
La Fig.4 présente l'organisation interne des données dans Neo4J sous forme de graphe.

Critères d'évaluation des modèles

Les critères suivants ont été choisis pour évaluer en parallèle le modèle orienté graphe et le modèle relationnel:

- Nature des données (**structurées**, **faiblement structurées**),
- Relation entre les données (**intégrité référentielle, relations hiérarchiques**),
- Cycle de vie des données (**versionning**),
- **Schéma** et opérations **CRUD,**
- Consistance des données (propriétés **ACID**),
- Performance (**indexation,** volumétrie),
- Analyse des données (rapports, prédiction).

Nature des données :

Données structurées [1] (L'index entre [] permet de repérer facilement les critères dans la matrice chromatique qui se trouve dans la conclusion)
Comme pour le modèle relationnel, le modèle orienté graphe est conçu pour stocker des données structurées. Ces données sont stockées dans les nœuds du graphe.

Données faiblement structurées [2]
La part des données non structurées est toujours plus importanes au sein des entreprises. Ces données existent sous forme de texte stocké dans des documents, manuels, rapports, messages, pages Web et présentations. Certains SGBDR (Oracle ou Postgres) prennent en charge ce type de données. Ces données faiblement structurées vont cohabiter à côté des données structurées. Une seule et même requête sera capable de travailler avec les deux types de données simultanément (recherche d'un terme dans un document et avec un filtre sur le nom de l'auteur).
Neo4J ne possède pas de fonctionnalités pour stocker et travailler avec du texte libre de grande taille.

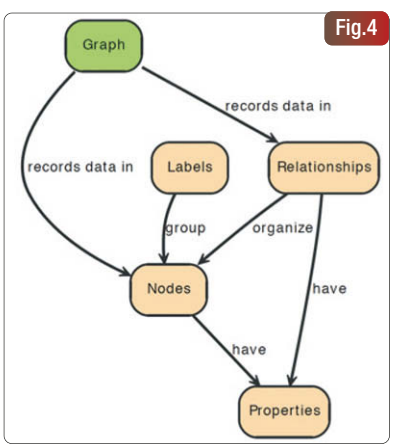


Fig.4

Organisation des données dans Neo4J (I. Robinson, J. Webber, E. Eifrem, 2013)

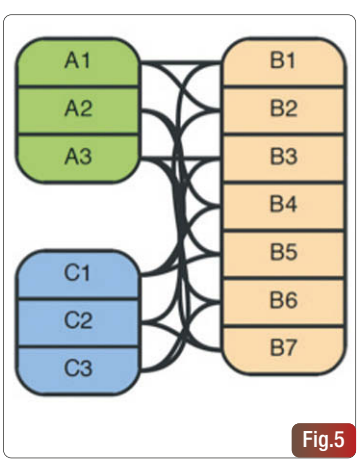


Fig.5

Organisation des tuples dans un RDBMS

Relation entre les données

Dans le modèle orienté graphe, les données sont reliées entre elles localement par des relations. Si on dissocie tous les tuples pour un ensemble de tables et que l'on conserve toutes ces relations, alors on obtient un graphe. Ce concept est représenté dans la Fig.5 et la Fig.6 ci-dessous :
Le modèle orienté graphe stocke l'ensemble de ces relations. La flexibilité de ce modèle permet de rajouter des nœuds et des relations sans compromettre les données existantes. Ce type de modèle est parfaitement adapté pour les données connectées.
Les requêtes vont utiliser des chemins qui sont stockés de façon naturelle dans le graphe, par conséquent ces requêtes seront beaucoup plus performantes que dans d'autres modèles.
Pour le cas du modèle relationnel, les requêtes impliquent souvent l'utilisation de jointures entre les tables composant le schéma, ceci a des conséquences sur les performances du système.
Si on considère l'ensemble des modèles utilisés par les technologies NoSql, c'est le modèle orienté graphe qui se distingue le plus du modèle relationnel. Dans le modèle relationnel, les tables contiennent des ensembles de lignes elles-mêmes composées d'un ensemble de valeurs. Les relations correspondent aux jointures réalisées entre les lignes des différentes tables, elles ne correspondent pas à des relations de sémantique entre les termes (les valeurs). Dans le modèle orienté graphe les valeurs ne font pas partie d'un ensemble, elles sont directement reliées entre elles.

Intégrité référentielle [3]
La transformation d'un modèle conceptuel en un modèle physique est relativement intuitive si on utilise des graphes. Cette transformation n'est pas toujours évidente lorsque l'on travaille dans le monde relationnel. En effet lorsque l'on passe d'un schéma entité-relation à un modèle logique de données, les contraintes d'intégrité référentielles ont été ajoutées. Toutes ces métadonnées servent à la base de données mais pas directement à l'utilisateur (Robinson, Webber, Eifrem, 2013). Les graphes maintiennent naturellement l'intégrité référentielle qui existe entre les données.

Relation hiérarchique [4]
Les graphes gèrent naturellement les relations hiérarchiques qui existent entre des données. Ces structures sont facilement exploitées à l'aide des algorithmes de parcours de graphe.
En revanche, le traitement des données ayant des liens de parenté sont difficiles à gérer dans une base de données relationnelle. Le problème vient du fait que le SQL standard ne permet pas le parcours récursif des données. Pour pallier ce problème, des fonctions spécifiques ont été ajoutées pour étendre le SQL standard et combler cette lacune, il s'agit par exemple de la fonction START WITH ... CONNECT BY implémentée dans Oracle.

Cycle de vie

Versionning [5]
Le « Multiversion Concurrency Control » abrégé MVCC (Philip Bernstein and Nathan Goodman, 1981) est une méthode informatique de contrôle des accès concurrents fréquemment utilisés dans les systèmes de gestion de bases de données. Lorsque qu'une donnée est modifiée dans la base, une nouvelle version est créée, ainsi les données anciennes sont conser-

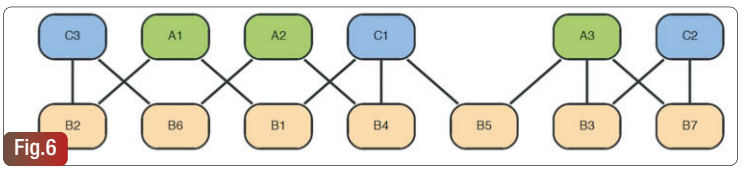


Fig.6

Organisation dans un modèle orienté graphe.

vées. Ce mécanisme permet d'éviter l'utilisation de verrous pour les opérations en lecture/écriture. Les données obsolètes devront être purgées par le système. De nombreuses bases de données utilisent le MVCC (MySQL, Postgres, Oracle);ce mécanisme est également implémenté dans certains SGBD NoSQL (CouchDB).
Dans le cas du modèle orienté graphe, le « versionning » peut être conceptualisé à l'aide d'un graphe:

```
(V3)-[:PREV]->(V2)-[:PREV]->(V1)
```

Les nœuds du graphe comportent les numéros de version. Dans l'exemple ci-dessus la version la plus récente est contenue dans un nœud qui comporte la propriété V3. Ces numéros de versions sont reliés entre eux par les relations « previous ».

Schéma et opérations CRUD

Schéma [7]

Bien qu'il n'y ait pas à proprement parler de schéma dans le modèle orienté graphe, le typage (string, integer) des valeurs qui rentrent dans la composition d'une propriété (association nom-valeur) est possible avec Neo4J. Il est également possible de créer des contraintes, celles-ci vont s'assurer que les valeurs des propriétés sont uniques pour un ensemble de noeuds. La commande suivante va créer une contrainte d'unicité pour le numéro d'identification d'un article.

```
CREATE CONSTRAINT ON (article:Article) ASSERT article.id IS UNIQUE
```

CRUD [8]

Pour rechercher de l'information dans un modèle orienté graphe, la méthode utilisée est le « graph walking ». Il existe plusieurs langages pour parcourir des graphes dans Neo4J: Gremlin, le langage **SPARQL** et Cypher. Cypher est spécifique à Neo4J, j'ai choisi celui-ci pour les exemples car il est facile à comprendre et il suit une logique de représentation de graphes. Cypher permet d'effectuer des recherches sur des données qui sont en correspondance avec un motif spécifique. Il y a différentes clauses pour construire des requêtes, les trois clauses principales sont START, MATCH et RETURN. La clause START permet de spécifier un point de départ, la clause MATCH permet de fournir un motif spécifique et la clause RETURN renvoie à l'utilisateur les éléments pour lesquels il existe une correspondance avec le motif. Les commandes qui suivent vont créer le graphe représenté dans la Figure 2 vue plus haut.
Dans l'exemple suivant nous créons l'utilisateur Alice, deux ordres (chacun avec un identifiant et deux propriétés: date et status) et trois produits (chacun avec un identifiant et une propriété: description).

```
CREATE ({ user: 'Alice'});

CREATE ({ order: 1234, date: 20120808, status:'delivered' });
CREATE ({ order: 5678, date: 20120816, status:'dispatched' });

CREATE ({ id: 'abcd', description: 'strawberry ice cream' });
CREATE ({ id: 'efab', description: 'brussels sprouts' });
CREATE ({ id: 'cdef', description: 'espresso beans' });
```

L'exemple ci-dessous montre comment les ordres sont reliés avec les produits.

```
MATCH (order {order:1234}), (article {id :'abcd'})
CREATE (order)-[:CONTAINS]->(article);

MATCH (order {order:1234}), (article {id :'efab'})
CREATE (order)-[:CONTAINS]->(article);

MATCH (order {order:1234}), (article {id :'cdef'})
CREATE (order)-[:CONTAINS]->(article);

MATCH (order {order:5678}), (article {id :'cdef'})
CREATE (order)-[:CONTAINS]->(article);
```

Il reste à présent à relier chaque ordre avec l'utilisateur Alice et à créer les relations pour pouvoir retrouver les ordres chronologiquement.

```
MATCH (user {user: 'Alice'}),(order {order:5678})
CREATE (user)-[:PLACED]->(order);
MATCH (user {user: 'Alice'}),(order {order:1234})
CREATE (user)-[:PLACED]->(order);

MATCH (user {user: 'Alice'}),(order {order:5678})
CREATE (user)-[:MOST_RECENT]->(order);
MATCH (o2 {order:5678}), (o1 {order:1234})
CREATE (o2)-[:PREVIOUS]->(o1);
```

La commande suivante retourne l'ensemble du graphe stocké dans la base de données :

```
MATCH (n)-[r]->(m)
RETURN n as from, r as `->`, m as to;
```

Pour supprimer l'entrée complète du graphe, on peut utiliser cette commande :

```
MATCH (n)
OPTIONAL
MATCH (n)-[r]-()
DELETE n, r ;
```

Remarque : les opérations qu'il est possible de réaliser sur les graphes sont très souples, cependant la création de nouveaux nœuds avec leur relation est peu performante.

Consistance

Propriétés ACID [9]

Neo4J est « ACID compliant », cela signifie que les opérations sont exécutées dans le système en respectant les contraintes ACID. Le niveau d'isolation par défaut est READ_COMMITED, ce niveau d'isolation garantit qu'une transaction va lire uniquement des données qui sont validées (committed values) et éviter ainsi les « **dirty read** ». Neo4J possède également un mécanisme de détection des verrous mortels « **DEAD LOCK** ». La suppression d'un nœud ou d'une relation est effectuée de façon consistante, toutes les propriétés liées à ce nœud ou à cette relation seront également supprimées.
Dans la version actuelle de Neo4J, il est possible de configurer des contraintes d'unicité pour certaines entités composant le graphe.

Performance

Les mécanismes d'exécution d'une requête sur le modèle orienté graphe et ceux d'une requête SQL sont assez différents. Voici un exemple : on souhaite connaître les noms des agents qui suivent l'utilisateur numéro 23, pour obtenir ce résultat, une possibilité est d'écrire la requête SQL suivante :

```
SELECT agents.*
FROM users
INNER JOIN users_agents ON users.userid = users_agents.userid
INNER JOIN agents ON users_agents.agent_id = agents.agent_id
WHERE users.userid = 23;
```

Les trois piles de la Fig.7 représentent trois tables: *users, agents* et *users_agents*. La table users_agents est une table d'association, elle permet de relier les données des tables *users* et *agents*. La requête SQL va devoir aller chercher des données dans ces trois tables pour répondre à notre problème.
La même requête est formalisée cette fois-ci avec le langage Cypher :

```
START user = node(23)
MATCH user-[user-agent]®agent
RETURN agent
```

La Fig.8 met en évidence le nombre d'éléments impliqués dans la requête, la relation *[user-agent]* permet de faire directement le lien entre l'utilisateur et ses compétences. Il n'est pas nécessaire d'utiliser une table d'association comme dans le modèle relationnel. De plus, pour obtenir ce résultat, seule une petite portion du graphe est parcourue.

Index [10]

Les SGBDR sont dotés depuis longtemps de mécanismes d'indexation performant. L'utilisation de statistiques calculées à partir des données contenues dans les tables permettent d'optimiser l'utilisation de ces index.
Neo4J offre la possibilité d'indexer les propriétés associées aux nœuds du graphe. Les index dans Neo4J sont de type « éventuellement disponible », cela signifie que la commande retourne immédiatement lors de la création d'un index, la création de l'index se fait en tâche de fond. Pendant la construction de l'index, les requêtes continuent de s'exécuter sur un graphe sans index. Les index sont maintenus automatiquement par le système.

Volumétrie [11]

Dans le Tableau 1, Partner et Vukotic (Robinson, Webber, Eifrem, 2013) ont comparé les performances entre un SGBDR et Neo4J pour un réseau social constitué avec un million d'utilisateurs, chacun en relation avec 40 amis, les relations entre les utilisateurs vont jusqu'à un niveau de profondeur de 5.

Tableau 1: Comparaison des performances entre un RDBMS et Neo4J

Depth	RDBMS execution time (s)	Neo4j execution time (s)	Records returned
2	0.016	0.01	~2500
3	30.267	0.168	~110,000
4	1543.505	1.359	~600,000
5	Unfinished	2.132	~800,000

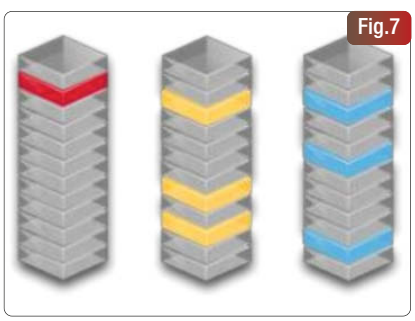
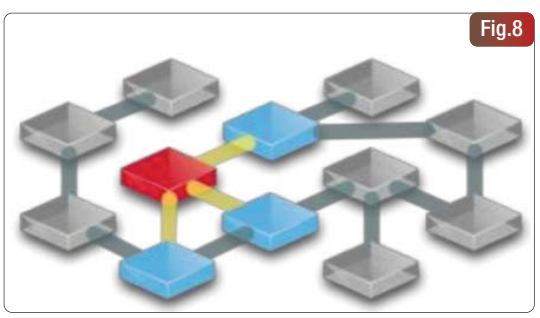


Figure 7: Répartition des données dans le modèle relationnel



Répartition des données dans le modèle orienté graphe.

A un niveau de profondeur de 2, Neo4J et le RDBMS montrent un temps de réponse similaire. Cependant déjà à partir d'un niveau de profondeur de 3, le RDMBS donne un temps de réponse inacceptable pour une application Web en ligne. Le modèle orienté graphe est bien adapté pour des données connectées (c'est à dire des données qui sont liées sémantiquement entre elles) Chaque nœud possède des références directes par rapport aux nœuds qui lui sont adjacents, cela signifie que le temps de réponse ne dépend pas de la taille globale du graphe, seule une partie du graphe est concernée. La complexité de l'algorithme de recherche va être directement proportionnelle au nombre de nœuds traversés; pour traverser un réseau de n nœuds, la complexité est O(n).
Une base de données relationnelle est conçue pour fonctionner sur un serveur centralisé et garantir un certain nombre de propriétés liées aux transactions. Dans le cas où un système de base de données nécessite plus de puissance (volume de données qui augmente, mise en production de nouvelles applications) il faudra migrer les données de la base sur une machine de calibre supérieur (« upgrade in the box »), ce qui en soi est une opération compliquée et qui nécessite un arrêt du service.
Comme toutes les bases Nosql, Neo4J fait partie de la catégorie des bases de données « Big Data », les données peuvent être distribuées sur plusieurs machines afin de répartir la charge.
La réplication entre un nœud maître et des nœuds esclaves permet également d'augmenter les performances en diminuant le temps de réponse pour les requêtes en lecture.

Analyse

Rapports, prédiction [12]

Les rapports d'analyse ne sont pas générés directement sur les systèmes opérationnels. Les «entrepôts de données» stockent les données courantes et historiques extraites de divers systèmes opérationnels et les regroupent pour permettre la génération de rapports (K. & J. Laudon, 2010). Les données une fois entrées dans l'entrepôt sont stables, en lecture seule, non modifiables par les utilisateurs. Ceci dans le but de conserver la traçabilité des informations et des décisions prises. Les DWH possède une structure particulière pour stocker des données qui sont dénormalisées; **modèle en étoile** qui s'écarte du modèle relationnel. Généralement il y a de très gros volumes de données au sein d'un entrepôt, c'est un modèle performant pour faire de l'analyse. Ce modèle qui fonctionne aussi à l'intérieur d'un SGBDR vient compléter le modèle relationnel. Hormis les réseaux sociaux, il y a beaucoup de domaines pour lesquels il est intéressant de représenter les données à l'aide d'un graphe, par exemple la bio-informatique, les statistiques en sport ou le commerce électronique (I. Robinson, J. Webber, E. Eifrem, 2013). Ces secteurs d'activité vont également bénéficier de ce modèle pour faire de l'analyse. Reprenons comme référence l'exemple de la figure 6 sur laquelle sont représentés des ordres. Je vous propose de construire un petit moteur d'inférence extrêmement simplifié pour réaliser de la vente croisée. Nous avons remarqué que l'utilisateur Alice commande fréquemment les produits « abcd » et « cdef », l'objectif est de recommander un de ces produits à un autre utilisateur qui possède certains points en commun avec Alice. Nous pouvons rechercher un utilisateur qui possède un certain nombre de points communs avec Alice, par exemple un utilisateur qui a son lieu de résidence proche de chez Alice et qui a commandé le produit « cdef » mais pas le produit « abcd ». Cet utilisateur peut potentiellement s'intéresser au produit « abcd ».
La requête suivante va permettre d'obtenir ce résultat :

```
MATCH (article)<-[:CONTAINS]-(order)<-[:PLACED]-(user)-[:NEAR]->(uAlice {user:Alice})
WHERE article.id = 'cdef' AND article.id <> 'abcd'
RETURN user;
```

Conclusion :

La matrice chromatique proposée ci-dessous est une synthèse de l'ensemble des critères évoqués pour les deux modèles étudiés plus haut.

Rubriques	Catégories		Modèle orienté graphe	Modèle relationnel
Données	Nature des données	1	données structurées	données structurées
		2	données faiblement sutrucrurées	données faiblement sutructurées
	Relation entre les données	3	Intégrité référentielle	Intégrité référentielle
		4	relations hiérarchiques	relations hiérarchiques
	Cyle de vie	5	versionning	versionning
Requêtes	Schéma et opérations CRUD	7	schéma	schéma
		8	opérations CRUD	opérations CRUD
	Consistance des données	9	Propriétés ACID	Propriétés ACID
Performance	Peformance	10	Index	Index
		11	volumétrie	volumétrie
Analyse	Analyse	12	rapports , prédiction	rapports , prédiction

mauvais (dégradation importante) ou pas implémenté
possible
Bon, point clé du modèle

Bien que le modèle orienté graphe soit une très bonne solution dans un contexte « Big Data », il peut être également une solution dans d'autres situations pour lesquelles la gestion d'un grand volume des données n'est pas la préoccupation centrale. C'est un modèle « scalable », moins rigide que le modèle relationnel; il est mieux adapté pour supporter les modifications qu'impose l'évolution du « business ». C'est un modèle de choix pour traiter des données connectées entre elles ou faire de l'analyse. Pour l'aspect transactionnel, le modèle orienté graphe est « ACID compliant », cependant si on se place dans un contexte dans lequel les données sont répliquées sur plusieurs nœuds, le système va perdre en consistance, on dit alors qu'il est « eventually consistant ».

D'un point de vue de la sécurité des données, Neo4J n'offre pas des mécanismes aussi performants que ceux que l'on trouve au niveau de certains SGBDR (Oracle, SQL Server) qui implémentent une très grande granularité dans la configuration des droits d'accès aux données.

Il est possible de conclure en disant qu'un seul modèle est rarement optimal pour répondre à l'ensemble des besoins d'une application, et le modèle orienté graphe avec ses caractéristiques est un bon complément au modèle relationnel. ⊞

Références :

- *Chen P.,* **The Entity-Relationship Model - Toward a Unified View of Data**, 1976
- *T. Harder and A. Reuter.* **Principles of transaction-oriented database recovery,** 1983
- *K. & J. Laudon,* **Management des Systèmes d'Information,** 2010
- *O. Mallassi,* **La mort des bases de données relationnelles et du SQL, *Programmez!,*** Février 2011
- *I. Robinson, J. Webber, E. Eifrem,* **Graph Databases**, O'REILLY, 2013
- *Philip Bernstein and Nathan Goodman,* **Concurrency Control in Distributed Database Systems**, 1981.
- **Neo4J online manual v. 2.2.2**

Glossaire technique

ACID (Atomicité, Consistance, Isolation, Durabilité) (Harder and Reuter, 1983) : c'est un ensemble de propriétés qui garantissent que les transactions dans la base de données sont fiables.

Atomicité : l'opération exécutée sur des données réussit entièrement ou sinon échoue entièrement.

Consistance : lorsqu'une transaction se termine normalement, et après validation des résultats, la base de données est préservée dans un état consistant.

Isolation : des évènements appartenant à une transaction doivent être invisibles pour d'autres transactions concurrentes.

Durabilité : la propriété de durabilité assure que lorsqu'une transaction est validée, elle est persistante dans le système, même si une panne survient après la validation.

Algèbre relationnel : c'est une théorie mathématique proche de la théorie des ensembles qui définit des opérations qui peuvent être effectuées sur des relations.

CRUD : Create, Read, Update, Delete.

Dead Lock : un « Dead Lock » (en français verrou mortel) se produit lorsque deux transactions se bloquent mutuellement.

Dirty Read : un « Dirty Read » se produit lorsqu'une transaction lit une donnée modifiée par une autre transaction sans que la modification ait été validée.

Données structurées (Abiteboul et al.,1999) : des données structurées suivent un schéma prédéfini.
Les données vont se conformer aux spécifications fournies dans le schéma. Exemple : base de données relationnelle.

Données faiblement structurées : le terme de « faiblement structurée » signifie qu'il n'y a aucune structure identifiable pour ces données. Les données non structurées sont aussi décrites comme des données ne pouvant pas être stockées dans des tables dans une base de données relationnelle. Exemple : un document dans un répertoire, une vidéo, une image.

Index : structure organisée contenant de données (arbres) permettant d'accélérer la recherche dans une base de données.

Intégrité référentielle : les contraintes d'intégrité référentielles permettent au SGBD de gérer automatiquement la présence de données référencées dans différentes relations de la base. La notion de lien permet de spécifier de telles propriétés.

Langage déclaratif : un langage déclaratif est un paradigme de programmation, son objectif est de décrire ce que le programme doit accomplir plutôt que d'exprimer comment il doit le faire.

Modèle entité-association (Chen P., 1976) : c'est un modèle de données qui permet de décrire un système d'information de manière abstraite dans le but de l'implémenter dans une base de données relationnelle. Ce modèle est composé d'objets (les entités) et de relations qui peuvent exister entre ces objets.

OLAP : Online Analytical Processing

Relation de hiérarchie : les relations de hiérarchie sont représentées par des arbres. Les relations entre les nœuds de cet arbre sont de type parent-enfant.

Schéma de données: il permet de décrire formellement la structure des données, leurs types. On parle de tables dans le cas des bases de données relationnelles.

SPARQL : c'est un langage de requêtes au même titre que le SQL pour les bases de données relationnelles qui permet de rechercher, d'ajouter, de modifier ou de supprimer des données RDF disponibles à travers Internet. Son nom est un acronyme qui signifie SPARQL Protocol and RDF Query Language.

Versioning : Le « versionning » est généralement associé à une stratégie de migration de données qui a pour but de déplacer les données d'une ancienne version vers une nouvelle.

Performances PHP : garder un œil sous le capot avec Zend Server

Si le code nous apparaît si familier, il est curieux de se rendre compte à quel point nous sommes aveugles à la réalité de son exécution. Toutes ces boucles, requêtes, objets qu'on manipule à travers le code sont des objets abstraits dont le seul objectif est fonctionnel; le coût de tous ces éléments en termes de temps et de ressources reste hors de notre portée.

Sophie BEAUPUIS
Consultante PHP - Zend Technologies.

Pourtant des outils existent pour évaluer ce coût à l'exécution. XDebug et XHprof permettent de profiler des requêtes et d'analyser la répartition du temps et des ressources dans l'arbre d'exécution. Mais disons-le tout de suite, ces outils ne sont pas suffisamment précis, ni aisément utilisables dans des environnements de production, c'est à dire là où ils seraient pourtant les plus pertinents. Zend Server Z-Ray est une nouvelle génération d'outil d'APM (Application Performance Management) dédié à PHP. Il offre des facultés d'analyse du code en cours d'exécution encore inégalées, et ce, en développement comme en production. Pour vous en donner un aperçu, je vous propose de suivre cet article pas à pas.

Un peu de mise en place

Selon que vous utilisez Windows, Linux, OS X, le procédé d'installation est différent mais jamais compliqué. Pour Windows et OS X, Zend Server s'installe comme une application classique. Pour Linux, un script d'installation automatique est fourni. On peut aussi faire une installation manuelle en utilisant les gestionnaires de paquets habituels.

Voici les liens pour télécharger les différentes versions de Zend Server : http://www.zend.com/en/products/server/downloads. Bien entendu, si Zend Server est une solution payante, on vous laisse la faveur de l'utiliser gratuitement et sans limitation durant 30 jours.

En vous connectant sur http://<votre host>:10081 vous accédez à l'interface de gestion de Zend Server. Avant de pouvoir être totalement utilisable Zend Server doit s'amorcer et vous demande de créer un mot de passe et de définir un mode de fonctionnement par défaut. Choisissez le mode « development » pour disposer d'un serveur en mode « paranoïaque », prêt à faire face à tout évènement suspect. Zend Server utilise une base de données:, par défaut il utilisera SQLite. C'est amplement suffisant pour notre exercice.

Il ne nous reste plus qu'à installer l'application de test. Pour cela, Zend Server dispose d'un outil de déploiement basé sur des packages qu'il suffit d'importer via l'interface graphique. En vous rendant sur la rubrique « Guide page » un lien « Deploy now » permet de télécharger et d'installer le package d'un squelette ZF2. Il suffit de se laisser guider et l'application est opérationnelle en quelques secondes. Par commodité, installez l'application sur le server principal dans le chemin d'URL « /demo ». On accède alors à l'application de démonstration à l'adresse http://<votre host>/demo. Evidemment, ce type de package de déploiement est parfaitement utilisable pour votre application, je vous conseille de lire la documentation à ce sujet : http://files.zend.com/help/Zend-Server/zend-server.htm#deploying_code_with_zend_server.htm.

Introspection de code avec Z-ray

Le premier des outils d'analyse est Z-ray. Il se présente sous la forme d'une barre de débogue en apparence classique située en pied de chaque page de l'application.

Rendez-vous sur la page d'accueil de l'application de test, vous devriez voir cette barre en bas de votre page. Si ce n'est pas le cas, rendez-vous sur l'UI de Zend Server, sélectionnez « Debugging / Z-Ray » et cochez « enabled », dites que vous êtes vraiment sûr, et validez Fig.1.

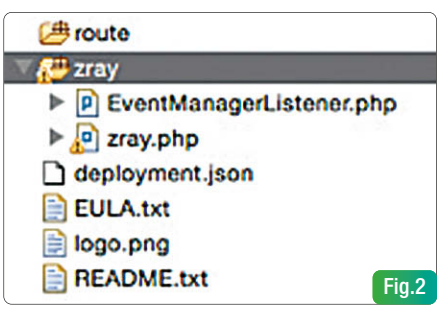


Fig.2

Z-Ray est assez voisin d'un débogueur, il recueille des informations sur le code à des moments précis de l'exécution mais Z-Ray ne stoppe pas le flot d'exécution. Les informations récoltées sont conservées en base de données puis mises en forme et affichées dans la barre de débogue à la fin de la requête.

Z-Ray se différencie des autres barres de débogue parce qu'il voit PHP depuis le moteur et non pas à travers PHP lui-même. Ce que les barres de débogue font pour quelques point fixes et spécifiques, Z-Ray est capable de le faire pour tout appel de méthode ou de fonction. Il le fait évidemment plus vite et sans risque d'interaction inattendu avec le code.

Sans entrer dans les détails, Z-Ray fournit bon nombre de données sur l'exécution du code. On peut analyser très simplement les requêtes passées en base de données, les appels aux méthodes ou fonctions. Notez que si la page courante génère une requête Ajax celle-ci sera aussi analysée par Z-Ray, et ses données disponibles dans la barre de débogue.

Mais la vraie originalité se trouve dans les onglets situés à droite et en haut de la barre de débogue. Z-Ray a reconnu de façon automatique le framework utilisé par l'application - Zend Framework 2 - et en a analysé des données spécifiques. La capacité de Z-Ray à placer des points de collecte où il veut, lui permet d'analyser en profondeur des fonctions spécifiques à un framework ou un CMS. Des plugins sont fournis pour Magento, Symfony, Zend Framework, Drupal, WordPress, etc. On peut parfaitement créer soi-même ses propres plugins car le kit de développement est open source et extrêmement simple d'utilisation. Pour s'en convaincre on va créer et déployer nous-même notre plugin Z-Ray en quelques minutes.

Fig.1

```php
<?php
namespace ZRayListeners;

require_once 'EventManagerListener.php';
$emListener = new EventManagerListener();
$zrayExtension = new \ZRayExtension('zf2-listeners');
$zrayExtension->setEnabledAfter('Zend\Mvc\Application::init');
$zrayExtension->traceFunction('Zend\Stdlib\CallbackHandler::getCallback',function(){},array($emListener,'onGetCallBack'));
$zrayExtension->traceFunction('Zend\EventManager\EventManager::triggerListeners',function(){},array($emListener,'onPostTriggerListener'));
$zrayExtension->traceFunction('ZRayListeners\ZrayShutdown',array($emListener,'onShutdown'),function(){});
function ZrayShutdown(){}
register_shutdown_function('ZRayListeners\ZrayShutdown');
```
Fig.3

Créer un plugin pour Z-ray

On ne va perdre de temps à programmer car le code du plugin est disponible sur GitHub: https://github.com/sophpie/zs81plugin. Il s'agit d'un plugin destiné à enregistrer tous les écouteurs appelés à la suite du lancement des évènements dans une application Zend Framework 2.

Clônez le dépôt dans un projet de votre IDE préféré et analysons son contenu : Fig.2.

Le fichier zray.php contient ce qui est nécessaire à l'utilisation du plugin par Z-Ray : Fig.3.

- **new \ZrayExtension()** : crée l'instance de l'extension.
- **ZrayExtension::setEnabledAfter('fonction')** : indique que l'extension doit s'activer après l'appel à la fonction « fonction ». Ici, on active l'extension dès qu'une application ZF2 est initialisée.
- **ZrayExtension::traceFunction('fonction', Callable $onEnter, Callable $onLeave)** : indique de collecter les données au moment de l'appel à la fonction « fonction ». Les deux Callables « $onEnter » et « $onLeave » sont deux fonctions qui recevront en paramètre les données collectées et seront exécutées respectivement avant et après l'appel de la fonction tracée. Dans notre exemple on collecte les données des fonctions :

  - « Zend\Stdlib\CallbackHandler::getCallback » pour connaître la nature des écouteurs utilisés.

  « Zend\EventManager\EventManager::triggerListeners » pour connaître l'évènement qui a été lancé.

On définit aussi une fonction de « shutdown » et on place un point de collecte sur cette fonction. Le but est d'accumuler les données durant la requête, et de les consolider une fois la requête terminée. Les Callables en charge de traiter les informations collectées ont été rassemblées dans la classe « ZRayListeners\EventManager Listener » : Fig.4.

Chaque méthode utilise deux paramètres :

- « $collect » est l'ensemble des données collectées sur l'appel de la fonction. C'est un tableau avec notamment les clés suivantes :

Clé	Valeur
functionName	Nom de la fonction ou de la méthode
functionArgs	Liste des arguments passés à la fonction
this	L'object $this
returnValue	Valeur de retour de la fonction
calledFromFile	Nom du fichier qui à générer l'appel
calledFromLine	Ligne du fichier où se situe l'appel

- « $storage » est le tableau de données qui sera renvoyé à Z-Ray pour afficher les données dans la barre de débogue.

Reste à déployer ce plugin sur Zend Server. Cette opération est triviale, il suffit de « zipper » le projet, de se rendre sur l'interface de Zend Server rubrique « Plugins / installed », de cliquer sur « Deploy plugin » et de suivre les instructions. En quelques secondes votre plugin est disponible dans la barre de débogue Z-Ray : Fig.5.

On voit sur ce simple exemple qu'on peut tracer tout ce qu'on veut et personnaliser de façon très forte l'analyse des requêtes. Il est aussi possible de redéfinir l'affichage de Z-Ray en utilisant des panneaux personnalisés en HTML. On peut ainsi introduire des éléments externes, des graphes, des codes Javascript, etc.

Pour en savoir plus je vous invite à consulter la documentation et les exemples de code à l'adresse suivante : https://github.com/zend-server-extensions/Z-Ray-Documentation

Passons en production...

Là où le moteur est le plus gros et le plus soumis aux contraintes, c'est en production. Pour l'application c'est un environnement hostile loin du cocon chaleureux du laptop du développeur. Le monitoring de l'application est primordial pour assurer son bon fonctionnement.

Z-Ray peut jouer un rôle, quand on approche du cas désespéré il est vrai, mais il est tout à fait possible de l'activer en production pour analyser « live » le comportement de l'application. Evidemment Zend Server vous proposera d'effectuer cette analyse pour un temps limité, pour un lot d'url donné et depuis un lot d'IP clients bien précis. La criticité des données affichées par Z-Ray invite à la plus grande prudence.

Zend Monitor

Mais avant d'arriver à cette extrémité, on peut commencer par poser des garde-fous pour être informé des dysfonctionnements. N'oublions pas que la détection est la première étape de la résolution d'un bug.

En vous rendant sur l'UI de Zend Server dans la rubrique « monitoring » et « monitoring rules » vous verrez qu'une liste de sondes est disponible par défaut. Là_aussi je vous laisse découvrir le détail de ces sondes, mais on peut en dire rapidement qu'il s'agit de règles qu'on se fixe, de limites ou de points d'attention qu'on met en évidence. On trouve des sondes pour la durée d'exécution d'un lot de fonctions ou pour la totalité de la requête, des moniteurs des valeurs de retour de certaines fonctions, etc. Chaque règle est personnalisable et on peut en créer autant que l'on veut en se basant sur des templates de base. L'exemple suivant vous montre comment on détecte les requêtes trop lentes : Fig.6.

Ce qu'il faut retenir est que si une de ces règles est activée par un mauvais comportement de l'application (requête trop lente, valeur de retour

Fig.4

```php
<?php
namespace ZRayListeners;

class EventManagerListener
{
    protected $currentTriggerredEvent;

    protected $eventStorage=array();

    public function onPostTriggerListener($context,&$storage)
    {
        $eventName = $context['functionArgs'][0];
        $this->currentTriggerredEvent = $eventName;
        $this->eventStorage[$eventName] = array();
    }

    public function onGetCallback($context,&$storage)
    {
       $file = basename($context['calledFromFile']);
       $line = $context['calledFromLine'];
       if ($file != 'EventManager.php' || $line != '464') return;
       $listener = $context["returnValue"];
       $listenerName = 'undefined';
       $listenerName = self::getListenerName($listener);
       $this->eventStorage[$this->currentTriggerredEvent][] = array('name' => $listenerName);
    }

    protected static function getListenerName($listener = null)
    {
        if (!$listener) return 'null';
        if (is_array($listener)) {
            return get_class($listener[0]) . '::' . $listener[1];
        }
        if ($listener instanceof \Closure) return 'Closure';
        if (is_object($listener)) return get_class($listener) . '::__invoke()';
    }

    public function onShutdown($context,&$storage)
    {
        $storage['Listeners'] = $this->eventStorage;
    }
}
```

Fig.6

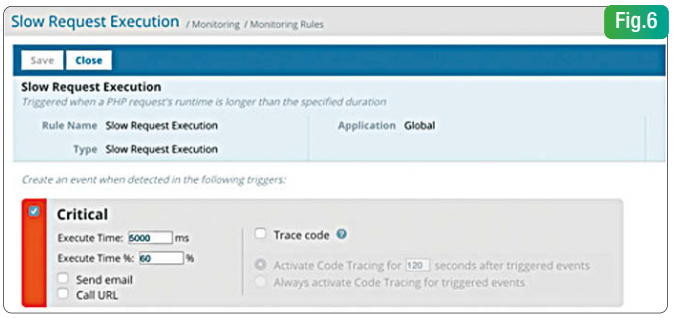
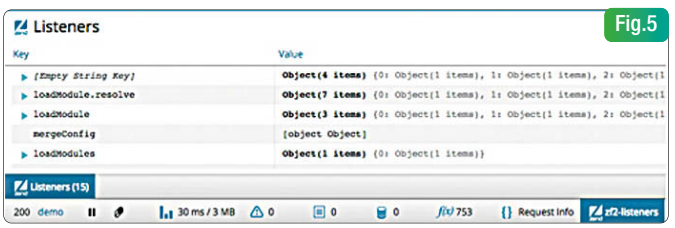


Fig.5



inadéquate) un évènement va être créé dans la base de données de Zend Server. Au besoin on peut aussi indiquer à la règle de monitoring d'envoyer un e-mail contenant les données de l'évènement pour avertir les équipes compétentes. Les évènements créés par le serveur sont disponibles sur la page « Monitoring » dans l'onglet « Events »: Fig.7.

On peut évidemment avoir des informations détaillées sur l'évènement : quand il a eu lieu, quelle règle a été violée, quelques données sur la requête, le nombre d'occurrences et sur quels serveurs dans le cas d'un cluster.

On a donc une détection des dysfonctionnements en amont et une mémoire de ces évènements, ce qui est déjà beaucoup mieux que ce dont la plupart des entreprises disposent en production aujourd'hui. Mais on peut aller encore plus loin avec Zend Server, on peut « profiler », ou « tracer » la requêtes fautive et l'analyser après coup. Ce « bug killer » absolu se nomme Zend Code Tracing.

Zend Code Tracing

Pour commencer on va tracer une requête manuellement, mais on peut déclencher la génération des profils de façon automatique.

Rendez-vous sur l'interface de Zend Server dans la rubrique « Debugg / Code tracing ». Saisissez l'url de la page d'accueil de l'application de démonstration et cliquez sur « Create trace ». Il peut être souhaitable de désactiver ZRay si vous ne souhaitez pas voir les fonctions des plugins apparaître dans la trace. Une fois la trace créée, cliquez sur le lien pour afficher les données: Fig.8.

Le profil de la requête contient la liste exhaustive de toutes les fonctions appelées avec, pour chacune, le temps d'exécution, les paramètres, la valeur de retour et le point du code où l'appel a eu lieu : nom du fichier et numéro de la ligne. On peut aussi faire apparaître la quantité de mémoire consommée.

En dépliant progressivement l'arbre d'exécution, on voit facilement les appels les plus lents. En utilisant une vue statistique dans l'onglet « Statistics per fonction » on peut aussi retrouver une vision plus proche des outils de profilage habituels.

Le profilage des requêtes par Code Trace, peut être déclenché automatiquement quand un évènement de monitoring est généré. Zend Server devient alors capable de détecter les dysfonctionnements et d'en prendre un instantané exploitable directement par les développeurs. Aucun autre outil ne permet une telle précision d'analyse et surtout pas en production. Quand on sait que 90% du temps de débogage réside dans sa reproduction et son identification, on comprend le gain de productivité que va apporter le couple Zend Monitor / Code tracing.

Evidemment Monitor et Code Tracing peuvent aussi être utilisés en développement en conjugaison avec Z-Ray. L'ensemble forme une chaîne qui permet d'analyser le comportement en profondeur de son code tout au long du cycle de vie de l'application.

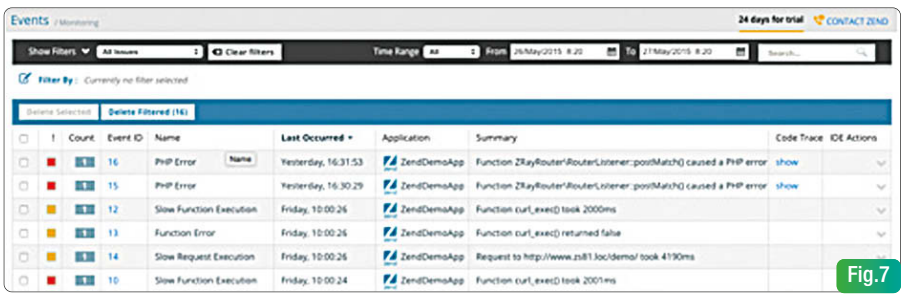


Fig.7

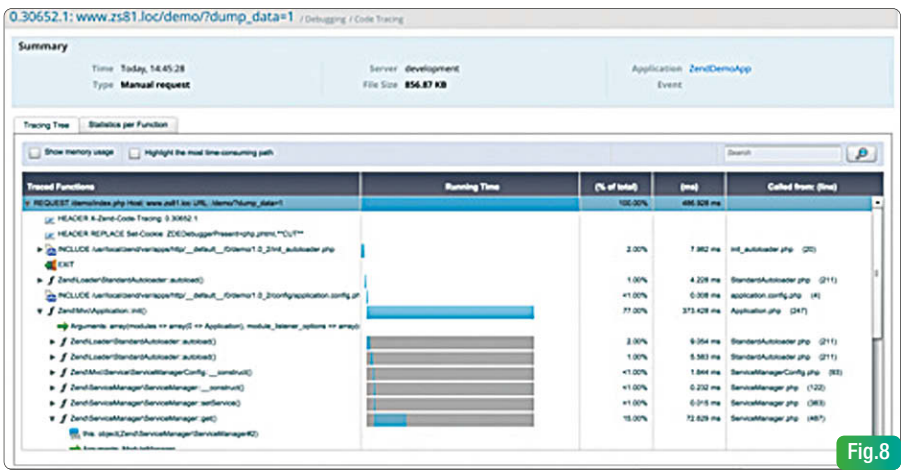


Fig.8

Pour finir

Garder un œil sur la consommation des ressources de son code est nécessaire pour assurer la robustesse de ses applications en production. Disposer d'outil d'investigations tout au long du cycle de vie est incontournable pour réduire les coûts de débogage et améliorer les taux de disponibilités des applications. Zend Server est une solution tout-en-un capable d'assurer ce service.

Zend Server c'est aussi des outils de gestion de cluster pour faciliter la scalabilité des systèmes, différents niveaux de cache, un gestionnaire de requêtes asynchrones et beaucoup d'autres fonctionnalités.

Je débute avec Github

Si vous avez besoin de partager votre code avec d'autres développeurs, Github est fait pour vous.

Céline Louvet
et Geoffrey Garnotel,
SFEIR

En effet, il s'agit d'une plateforme collaborative pour les développeurs. Sa fonctionnalité principale est de permettre d'héberger du code sous Git.
Vous aurez aussi la possibilité d'associer un wiki et une page Web à votre projet, et de faire un suivi des bugs et des versions.
De plus, Github offre de nombreuses fonctionnalités liées aux réseaux sociaux, telles que la possibilité de suivre des projets, des utilisateurs, d'échanger sur les développements.
Pour la suite, nous allons prendre l'exemple d'un nouveau projet, sous Git, dont vous avez la responsabilité d'héberger les sources sur Github pour pouvoir les partager avec vos collègues.

Premiers contacts

Tout d'abord, vous aurez à vous créer un compte sur Github. Une fois identifié, vous avez la possibilité de créer un nouveau dépôt Fig.1. Pour pouvoir créer ce dépôt, vous devez remplir le nom du dépôt, une description et choisir si le dépôt sera public ou privé. Vous pouvez aussi choisir d'initialiser le dépôt avec un README, un .gitignore, une licence Fig.2.
Si nous visualisons le dépôt, nous y retrouvons le gitignore initialisé selon le langage choisi, la licence et notre ReadMe qui est initialisé avec le titre du dépôt et sa description Fig.3.
Et voilà ! Notre projet est initialisé dans github, nous allons pouvoir publier notre code et gérer la configuration du dépôt.

Portage collaboratif

A présent, nous allons pouvoir ajouter des nouveaux participants à notre projet qui auront le droit de modifier les contenus de notre dépôt, d'y ajouter de nouvelles sources, modifier ou supprimer celles existantes. Pour ce faire, nous allons dans la section settings du dépôt, là où il est possible d'ajouter des collaborateurs; ceux-ci doivent avoir un compte sur Github Fig.4.
Notre dépôt étant créé et partagé à nos collègues, nous allons publier du code.
Il existe de nombreuses méthodes de travail avec Git. La plupart prônent la création de branches dédiées pour développer les nouvelles fonctionnalités. Vous pouvez créer ces branches de différentes manières, soit via l'interface de Github, soit via l'application Github téléchargée, soit en ligne de commande Fig.5.
Nous avons créé une branch dev pour les développements de nos sprints, nous allons dans les Settings du projet, afin de la définir comme branche par défaut pour nos futures Pull Requests (PR pour les intimes) Fig.6.
Notre dépôt étant créé, configuré et partagé à nos collègues, nous allons pouvoir publier du code. Tout d'abord, il va nous falloir le récupérer en local. Pour cela, nous pouvons faire un clone via ssh / https ou utiliser le bouton "clone in desktop" qui va nous proposer de télécharger l'application Github. Une fois, le dépôt récupéré, nous pouvons l'ouvrir dans notre IDE favori et coder. Nous avons choisi d'effectuer le développement de notre évolution dans une branche locale "story-one". Nous allons maintenant la publier pour pouvoir créer une PR. Pour cela, il nous suffira d'effectuer un "push" via notre l'outil Git de notre choix (application, ligne de commande, etc.). Automatiquement, Github va nous proposer la possiblité de la comparer avec la branche par défaut et de créer une PR, si la différence nous convient Fig.7.

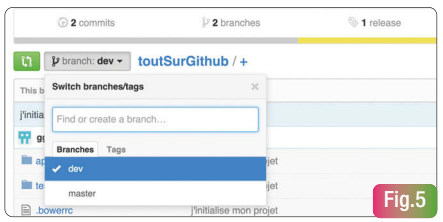


Fig.5

Pour ajouter et visualiser les branches

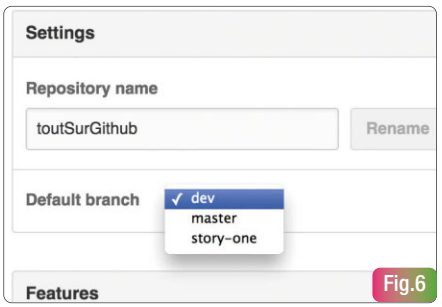


Fig.6

Pour configurer la branche par défaut

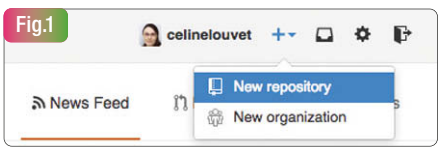


Fig.1

Pour créer un nouveau dépôt

Fig.4

Pour ajouter et visualiser les collaborateurs

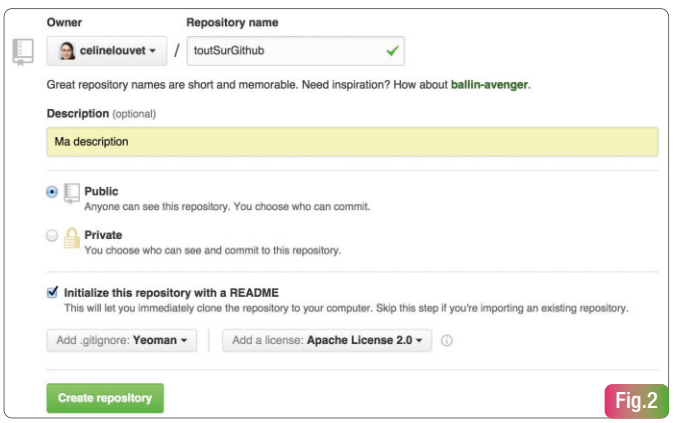


Fig.2

Création d'un nouveau dépôt

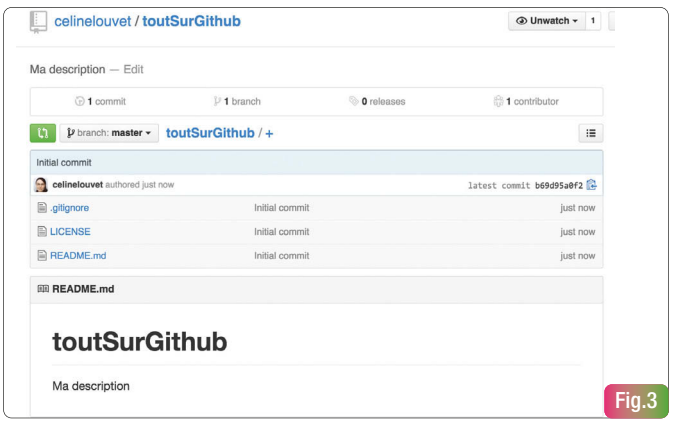


Fig.3

Le dépôt créé et initialisé par Github

Fig.7

Github nous propose de créer la Pull Request

Nous choisissons de créer la PR.
Github nous propose par défaut de fusionner le code de notre branche story-one avec celui présent sur dev, mais nous pouvons sélectionner une autre branche selon les besoins.
Nous mettons un titre à notre PR ainsi qu'une description dans laquelle nous pouvons utiliser du Markdown. Github nous indique si la PR sera "mergeable", c'est-à-dire qu'il peut être fusionné sans conflits avec le code existant sur dev Fig.8.
Une fois, la PR créée, on va pouvoir visualiser plusieurs choses : les différents commits en différence, les modifications dans les fichiers.
Une séance partagée de revue de code va pouvoir commencer. Les autres collaborateurs vont pouvoir commenter les commits ou les différences dans les fichiers. Une discussion va pouvoir être menée sur chacun des développements proposés. Suivant les commentaires, nous allons pouvoir reprendre le code proposé et le modifier et ainsi de suite Fig.9.
Une fois que le code est validé, un des membres de l'équipe va pouvoir valider la PR. Github va se charger de "merger" notre branche distante avec la branche de base, ici dev, et nous pouvons à présent visualiser l'ensemble du code de "story-one" sur "dev".
D'autres informations sont alors modifiées et accessibles au niveau de chaque répertoire et fichier, notamment le dernier commit, qui nous permet de connaître la dernière personne à avoir livré un commit, la date, le titre et le sha1 de ce commit Fig.10.
Pour pouvoir récupérer votre code, vos collègues n'auront qu'à récupérer en local le contenu du dépôt via un "fetch" ou un "pull". Ils pourront ainsi proposer leur propre code via de nouvelles PRs. A ne pas oublier le README de votre projet, que vous pourrez modifier à votre guise afin d'apporter une description de votre projet, des cas d'exemple, des illustrations, un tutoriel sur comment utiliser et exécuter votre projet ... Et voilà, vous savez tout pour pouvoir vous lancer rapidement dans Github. Mais vous pouvez aller encore plus loin.

Notion de fork

Tout d'abord Github offre la possibilité d'effectuer un "fork" d'un projet. Ce fork va réaliser une copie du projet initial dans un nouveau dépôt. Il conservera un lien vers le projet. L'intérêt ? Pouvoir proposer du code sur des projets où vous n'avez pas les droits en écriture. Ainsi, cette méthode est particulièrement utilisée pour les projets open source et peut être étendue à une gestion plus serrée des droits dans votre dépôt. Par exemple, sur notre projet, Céline et Djo ont les droit pour écrire et valider des PRs, mais si une personne veut contribuer au projet, elle va pouvoir le forker, y apporter les modifications qu'elle souhaite, puis les proposer via une PR; cette dernière ne pourra être validée que par Céline et Djo. Une discussion pourra ainsi commencer entre les acteurs du projet via le système de commentaires que nous avons décrit précédemment. A présent, si vous voulez gérer les dépôts de votre société, association ou autre structure, vous pourrez créer des "Organisations", cela va vous permettre de rassembler tous les dépôts que vous devez gérer au sein d'une même entité et de gérer les droits des participants en les répartissant en "Team"; chaque Team aura des droits qui lui seront propres... Par exemple, vous pouvez avoir une team d'administrateurs qui auront tous les droits sur tous les dépôts, puis une team par dépôt pour les développeurs respectifs des dépôts. En organisant ces développeurs en plusieurs teams, vous pourrez différencier ceux qui auront les droits en écriture et lecture de ceux qui n'auront que les droits en lecture. A tout cela, nous pouvons ajouter la possibilité de suivre les bugs et les évolutions en cours sur le projet et nous pourrons aussi ajouter un wiki à notre projet qui pourra être complémentaire à notre README pour apporter une documentation plus détaillée de notre projet. Ainsi nous disposons directement d'un bugtracker et d'un wiki qui seront accessibles à l'ensemble des participants de notre projet, ou plus si notre repo est public. Voilà nous avons vu comment créer et administrer notre projet mais Github ne s'arrête pas là. Nous pouvons avoir accès à un certain nombre de statistiques et d'autres outils proposés par la plateforme.

Conclusion

Comme nous avons pu le voir ensemble, Github est une véritable plateforme de communication et d'échange entre les différents participants d'un projet, mais va aussi nous permettre de "follower" d'autres projets.

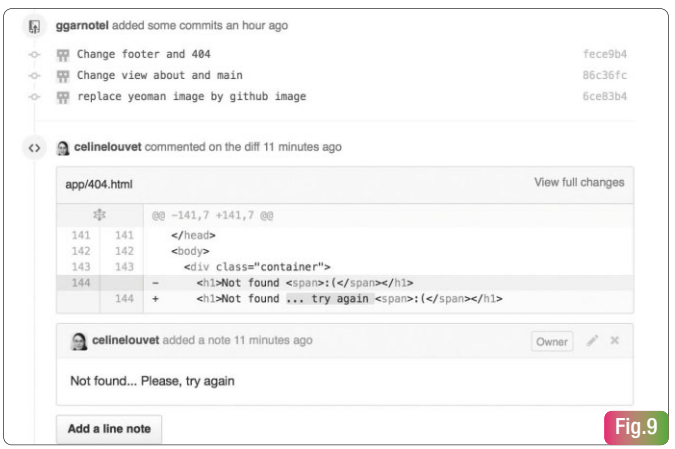


Fig.9

Échange de commentaires

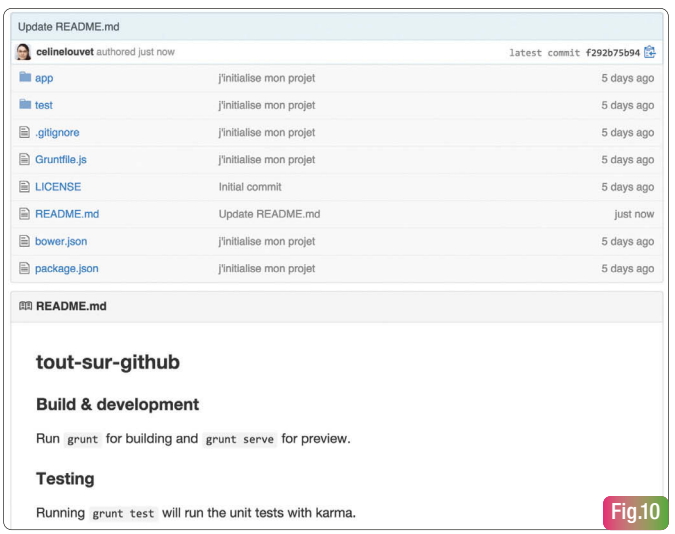


Fig.10

Du code et un README

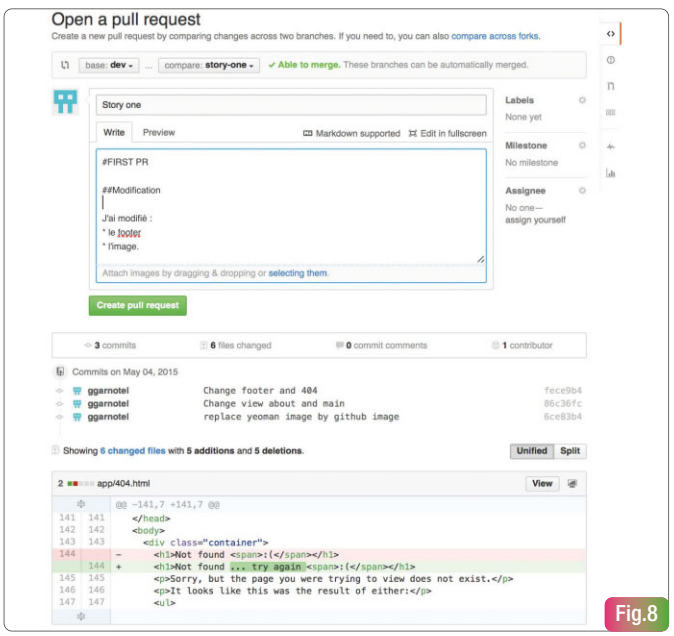


Fig.8

Création d'une Pull Request

Intégration continue pour une application mobile avec : Docker, Jenkins, Azure

L'objectif de cet article est de présenter la mise en place d'un serveur d'intégration continue dans le Cloud Azure avec le gestionnaire de conteneurs Linux Docker.

Régis Locatelli
Ingénieur JavaEE - VISEO.

Une première partie consiste en l'installation des logiciels nécessaires à l'utilisation de Docker (boot2docker, docker machine) sur un poste fonctionnant sous Windows 7 Professionnel. Une deuxième partie consiste en la création de la machine virtuelle sous Azure et nous verrons comment déployer Jenkins avec des plugin Gradle, Git et un sdk dans le Cloud Azure. Dans une troisième partie, nous verrons comment paramétrer le serveur Jenkins pour générer nos apks et les déployer sur notre compte de stockage Azure.

Installation des logiciels nécessaires pour l'utilisation de Docker sur un poste Windows

Docker

Docker est un gestionnaire de conteneurs Linux. Celui-ci va nous permettre de définir des environnements logiciels de développement, de test, d'intégration, de production de manière identique. Nous construisons une image Docker depuis laquelle nous pouvons déployer notre conteneur Docker sur n'importe quelle plateforme, grâce aux commandes Docker. De plus, un serveur peut exécuter plusieurs conteneurs. Pour construire ou stocker ces images Docker, il existe un référentiel : https://registry.hub.docker.com . Architecture Docker : **Fig.1.**

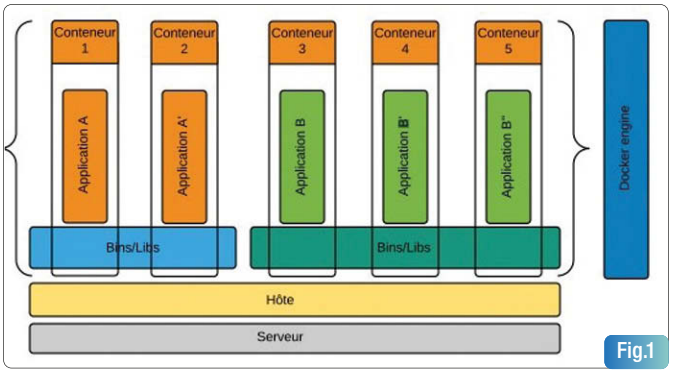


Fig.1

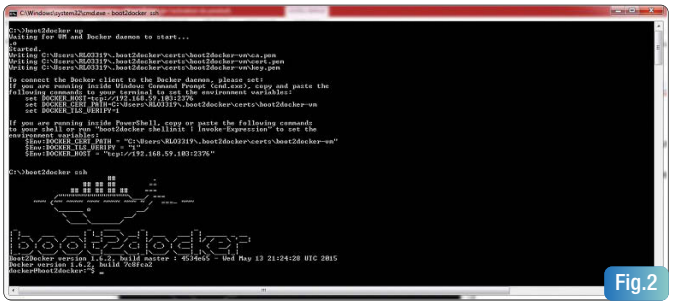


Fig.2

Boot2docker

Boot2docker est l'outil que nous allons démarrer pour pouvoir utiliser docker sur notre poste Windows. C'est à l'intérieur de cette machine virtuelle que nous allons utiliser le client Docker pour Windows pour construire une image Docker, la lancer... Voici deux liens qui vont nous assister dans cette étape, n'oubliez pas d'installer MINGW si celui-ci fait défaut sur votre poste. https://youtu.be/TjMU3bDX4vo et https://github.com/boot2docker/windows-installer/releases/latest

A la fin de l'installation, en lançant la commande « boot2docker ssh », vous obtiendrez l'écran suivant : **Fig.2.**

Docker machine

Nous allons utiliser docker machine pour déployer une machine virtuelle sur Azure. Pour installer cet outil : lancez MINGW et exécutez les commandes.

```
$ curl -L https://get.docker.com/builds/Windows/x86_64/docker-latest.exe > /bin/docker
$ curl -L https://github.com/docker/machine/releases/download/v0.2.0/docker-machine_windows-amd64.exe > /bin/docker-machine
```

Le dockerfile

```
FROM jenkins
# deploy a jenkins to build android sample https://github.com/googlesamples/android-TextLinkify
MAINTAINER REGLY

# update to be able to run aapt under jenkins
USER root
RUN dpkg --add-architecture i386 && apt-get update && apt-get install -y lib32z1 libstdc++6:i386

# install necessary plugins under jenkins
RUN mkdir -p /usr/share/jenkins/ref/plugins
ADD https://updates.jenkins-ci.org/download/plugins/scm-api/latest/scm-api.hpi /usr/share/jenkins/ref/plugins/scm-api.hpi
ADD https://updates.jenkins-ci.org/download/plugins/git-client/latest/git-client.hpi /usr/share/jenkins/ref/plugins/git-client.hpi
ADD https://updates.jenkins-ci.org/download/plugins/git/latest/git.hpi /usr/share/jenkins/ref/plugins/git.hpi
ADD https://updates.jenkins-ci.org/download/plugins/credentials/latest/credentials.hpi /usr/share/jenkins/ref/plugins/credentials.hpi
ADD https://updates.jenkins-ci.org/download/plugins/ssh-credentials/latest/ssh-credentials.hpi /usr/share/jenkins/ref/plugins/ssh-credentials.hpi
ADD https://updates.jenkins-ci.org/download/plugins/gradle/latest/gradle.hpi /usr/share/jenkins/ref/plugins/gradle.hpi
ADD  https://updates.jenkins-ci.org/download/plugins/windows-azure-storage/latest/windows-azure-storage.hpi /usr/share/jenkins/ref/plugins/windows-azure-storage.hpi

RUN mkdir -p /usr/share/jenkins/android
RUN chown -R jenkins "$JENKINS_HOME" /usr/share/jenkins/android

USER jenkins
# Install Android SDK
RUN cd /usr/share/jenkins/android && wget -nv -O - http://dl.google.com/android/android-sdk_r24.1.2-linux.tgz | tar --no-same-permissions --no-same-owner -xvzf - && chmod -R a+rX android-sdk-linux

ENV ANDROID_HOME /usr/share/jenkins/android/android-sdk-linux
```

```
ENV PATH $PATH:$ANDROID_HOME/tools
ENV PATH $PATH:$ANDROID_HOME/platform-tools

# Install Android SDK components
ENV ANDROID_SDK_COMPONENTS platform-tools,build-tools-22.0.1,android-22,extra-android
-m2repository,extra-google-m2repository
RUN echo y | /usr/share/jenkins/android/android-sdk-linux/tools/android update sdk --no
-ui --all --filter "${ANDROID_SDK_COMPONENTS}"

USER root
RUN chown -R jenkins /usr/share/jenkins/ref/plugins
```

Ce dockerfile se base sur l'image officielle Jenkins. Il contient les plugins nécessaires pour faire de l'intégration continue avec Jenkins. Il contient également les mises à jour utiles à la réussite du build gradle de l'apk sous Jenkins. Pour lancer cette image, il suffira d'exécuter la commande sous boot2docker :

```
docker run -d -p 8080:8080 regly/jenkinsandroid:v0
```

Création de la machine virtuelle sous Azure, déploiement du serveur Jenkins

Gestion des certificats

1) Pour pouvoir créer notre machine virtuelle sous Azure, nous devons créer un certificat sur notre poste, que nous allons importer sous le portail Azure. Pour créer ce certificat nous utilisons openssl. Celui-ci étant installé, nous allons exécuter les commandes suivantes dans son répertoire d'installation : C:\openssl

```
$ openssl req -x509 -nodes -days 365 -newkey rsa:1024 -keyout mycert.pem -out mycert.pem
```

Voici pour exemple ce que vous pourrez répondre à l'exécution de cette commande : Fig.3.

```
$ openssl pkcs12 -export -out mycert.pfx -in mycert.pem -name "My Certificate"
$ openssl x509 -inform pem -in mycert.pem -outform der -out mycert.cer
```

2) Vous téléchargerez ce certificat "mycert.cert" sous votre compte Azure Fig.4 et 5.

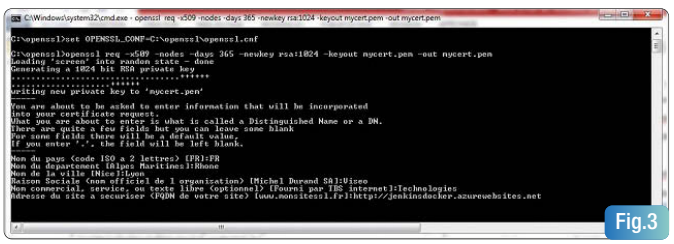


Fig.3

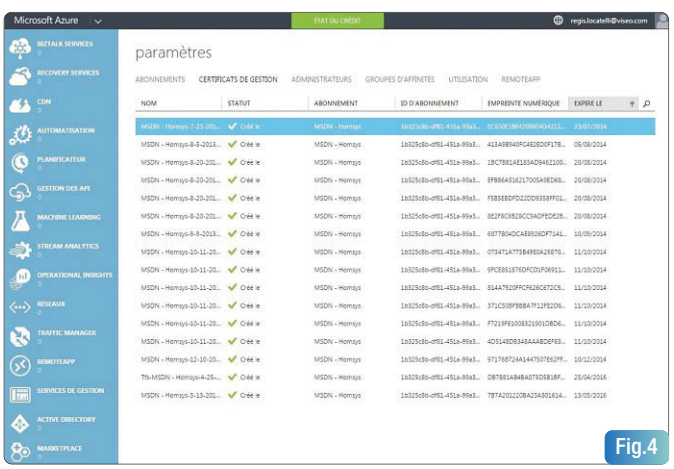


Fig.4

Création de la machine virtuelle

1) Nous récupérons sous Azure notre identifiant de souscription et utilisons MINGW pour lancer les commandes suivantes :

```
$ cd c:/openssl
$ export OPENSSL_CONF=/C/openssl/openssl.cnf
$ docker-machine create -d azure --azure-subscription-id="ID_SUBSCRIPTION_AZURE" -
-azure-subscription-cert="mycert.pem" jenkinsdocker
```

Résultat sur le portail Azure, la machine virtuelle est instanciée :
Sur le portail officiel : Fig.6.
Sur le portail en mode « bêta » : Fig.7.
2) Nous créons le point de terminaison http sous Azure pour pouvoir accéder à notre serveur Jenkins sur la machine virtuelle pour le port 8080, Fig.8.
3) Nous créons un compte de stockage « jenkinsstorage » pour déposer nos apks, Fig.9 et 10. Celui-ci propose des conteneurs dans lesquels nous pouvons stocker des blobs, nos apks dans notre cas.
Nous utiliserons la clef d'accès principale dans le paramétrage de Jenkins.

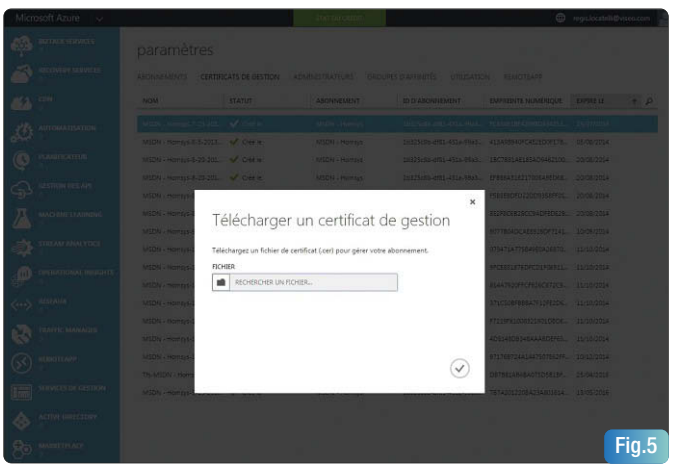


Fig.5

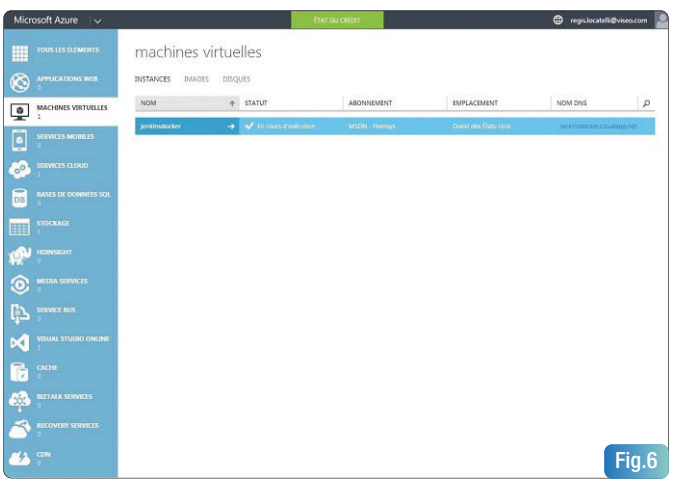


Fig.6

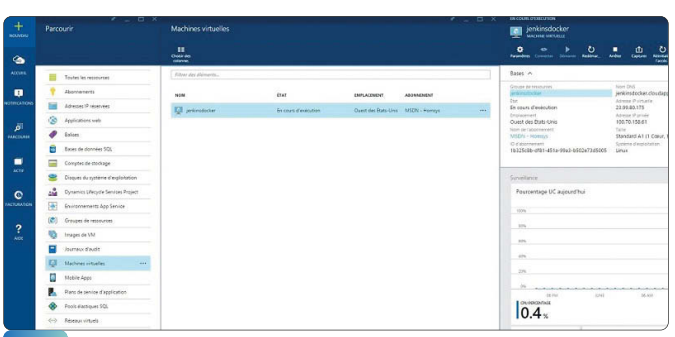


Fig.7

Construction et lancement de l'image Docker

Dans notre éditeur MINGW, lancez la commande précisée suite à la création de la machine virtuelle :

```
eval "$(/c/Program\ Files\ \(x86\)/Git/bin/docker-machine env jenkinsdocker)"
```

Cette commande nous permet de faire le lien entre notre client Docker et le host sur la machine virtuelle instanciée sous Azure. Elle sera utilisée à chaque fois que nous démarrons une nouvelle session pour prendre la main sur ce host. Ensuite, lancez la commande suivante :

```
docker run –name=myjenkins -d -p 8080:8080 regly/jenkinsandroid:v0
```

Cette commande va démarrer le conteneur myjenkins, de manière détachée, sur le port 8080 de la machine virtuelle Azure.

Accès au serveur Jenkins

Nous accédons au serveur Jenkins via l'url jenkinsdocker.cloudapp. net :8080, Fig.11.

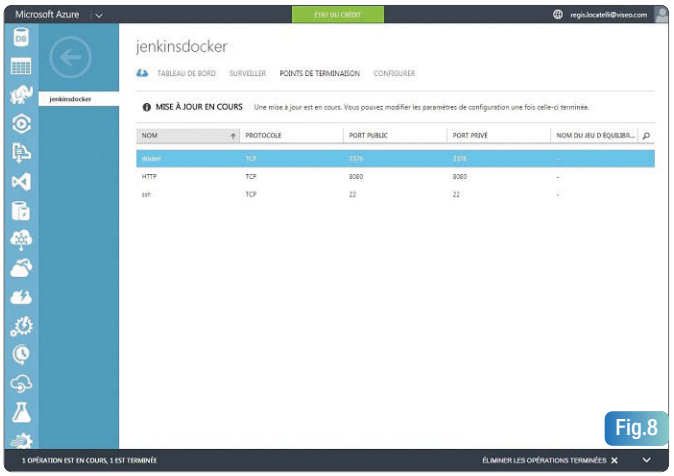


Fig.8

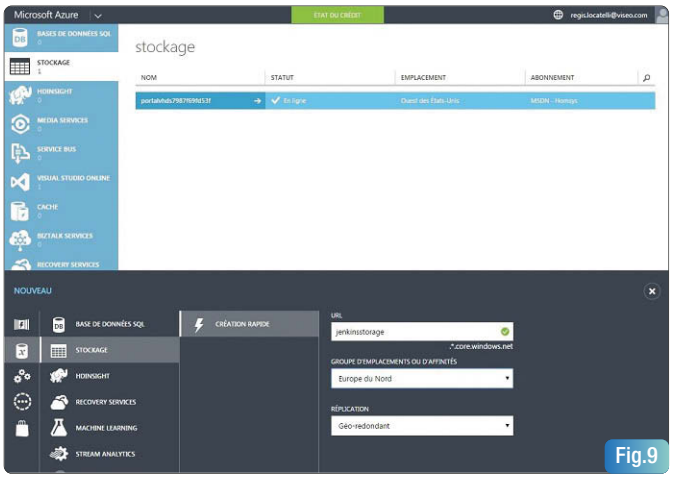


Fig.9

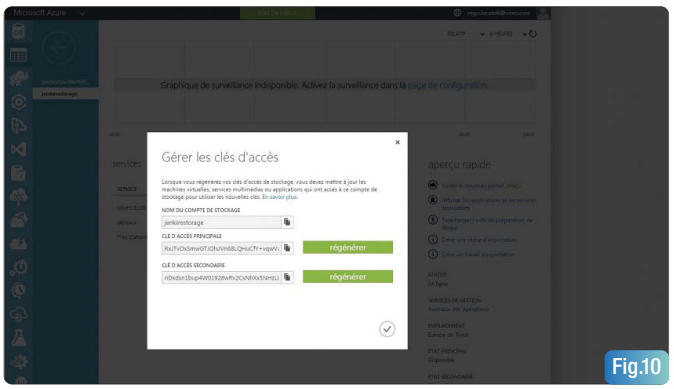


Fig.10

Paramétrage du serveur Jenkins et transfert de l'apk dans le compte de stockage Azure

Paramétrage du serveur Jenkins

1) Nous vérifions que les plugins définis dans l'image Docker sont correctement installés, Fig.12.

2) Nous devons définir la variable ANDROID_HOME et utiliser le plugin Gradle pour définir la version de Gradle à utiliser, Fig.13.

3) Nous utilisons le plugin Windows Azure Storage et pointons sur le compte de stockage « jenkinsstorage » défini précédemment : Fig.14.

Fig.11

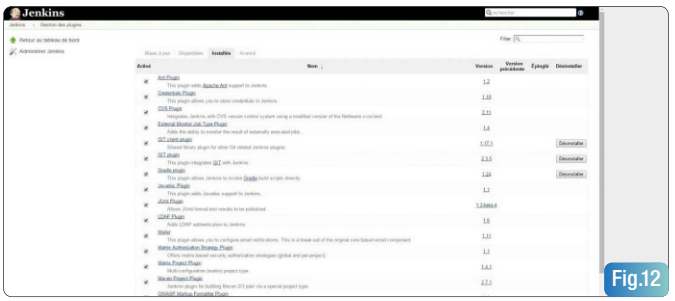


Fig.12

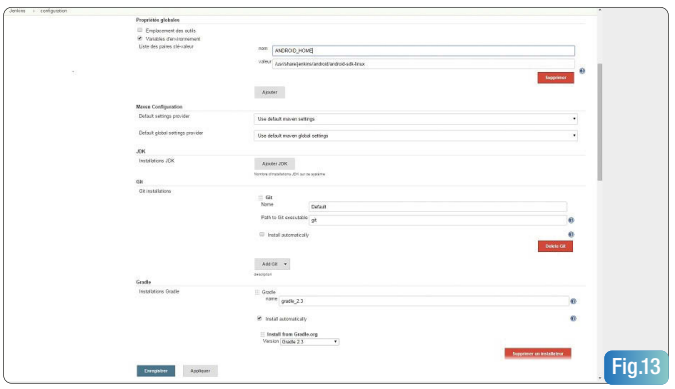


Fig.13

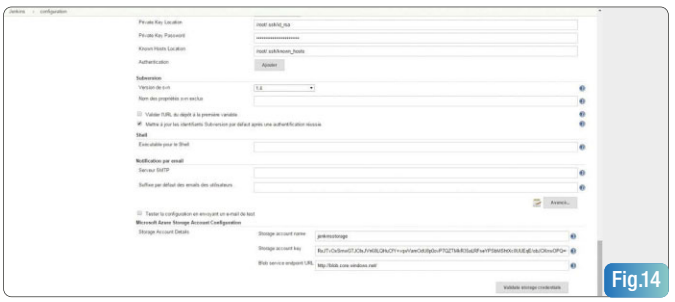


Fig.14

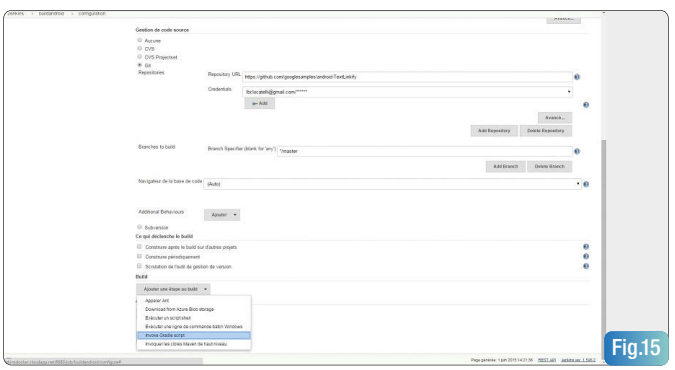


Fig.15

4) Nous créons le job Jenkins pour construire notre apk situé sous Github : https://github.com/googlesamples/android-TextLinkify . Fig.15, 16 et 17.

Etape 1 : ce job Jenkins lancé manuellement va récupérer les sources du projet Android via Git sous Github.

Etape 2 : nous allons invoquer la tâche build de Gradle sur ce projet.

Etape 3 : nous allons, suite à ce build, faire un upload de tous les apks générés sur notre compte de stockage Azure « jenkinsstorage ». Le conteneur dans jenkinsstorage porte le nom « buildandroid ». Pour pouvoir identifier les différents uploads, nous préciserons dans le path l'identifiant du build et son numéro.

5) Exécution du job : Fig.18 et 19.

Le build a réussi, nous remarquons sous Jenkins à plusieurs endroits, des liens précisant l'utilisant du Cloud Azure pour stocker les apks.

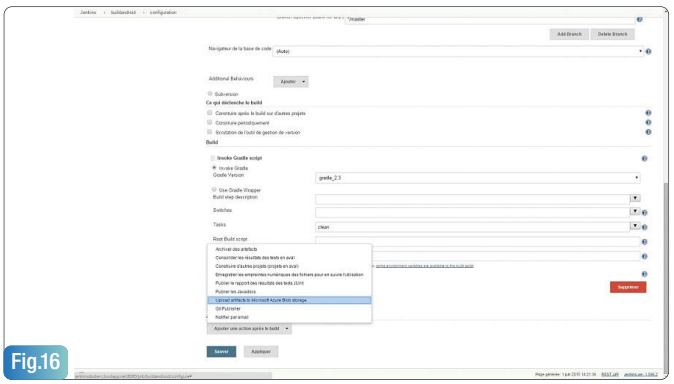


Fig.16

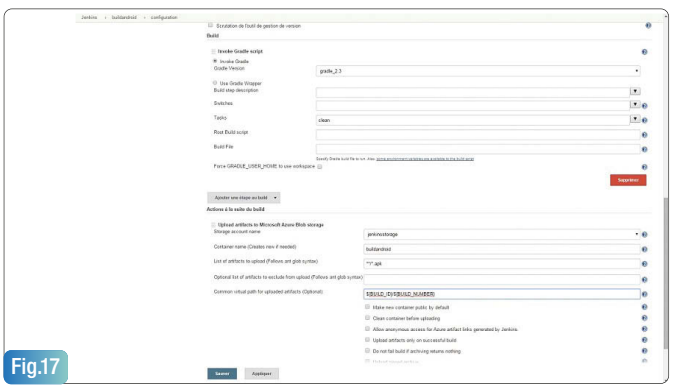


Fig.17

Fig.18


Fig.19

Résultat sous le compte de stockage azure

Sur le portail officiel, nous retrouvons nos apks : Fig.20 et 21.

Sur le nouveau portail, ça fonctionne moins bien... Fig.22 et 23.

Conclusion

Cet article nous a permis d'entrevoir les possibilités de plusieurs technologies innovantes. Nous pourrons retenir que grâce à notre image Docker, le déploiement de notre intégration continue peut être effectué en quelques minutes sur n'importe quel type de poste (Windows, Linux, Mac), en local ou sur un serveur, voire dans le Cloud. Nous avons pris en main Jenkins en limitant les efforts d'installation. Nous avons pu voir un exemple concret de déploiement d'une machine virtuelle dans Azure.

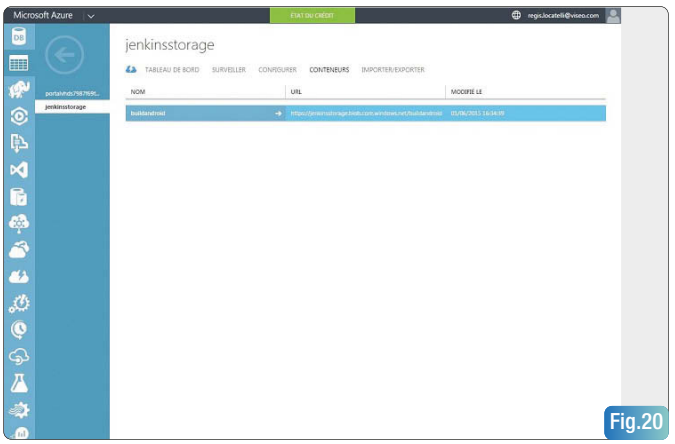


Fig.20

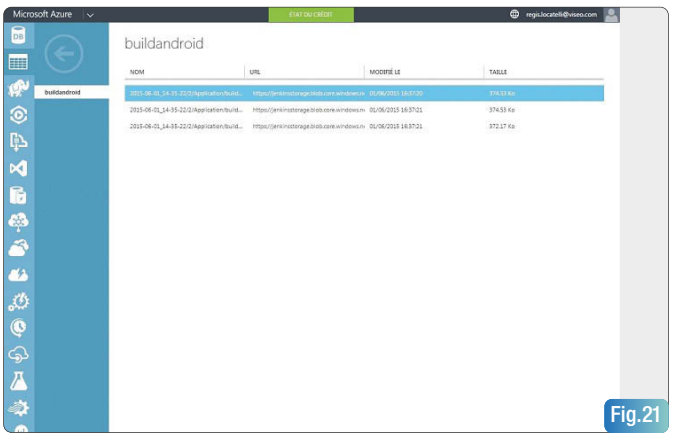


Fig.21

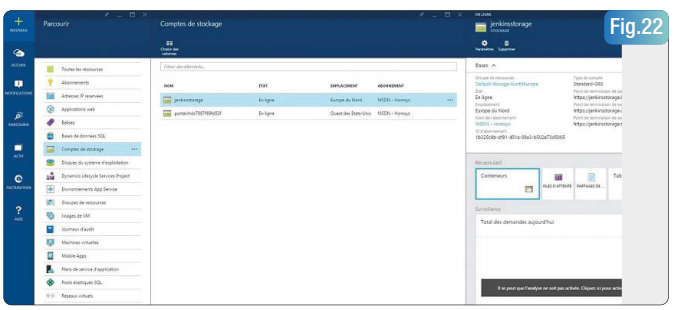


Fig.22

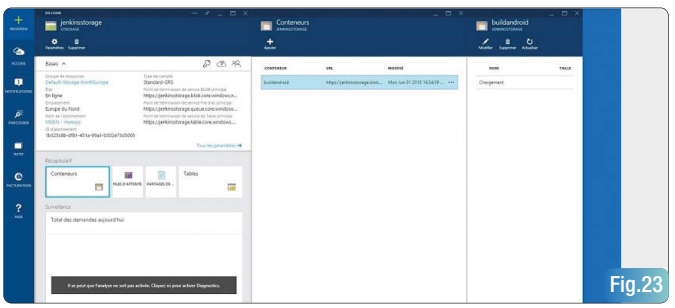


Fig.23

THEMING AVEC DRUPAL 8

Drupal 8 est dans les starting blocks et apporte son lot de nouveautés touchant tous les niveaux : site building, développement et theming. On peut citer quelques points fondamentaux comme l'utilisation de composants Symfony2, l'intégration de l'éditeur WYSIWYG CKeditor ou bien l'arrivée d'un outil de déploiement (Configuration Management). Plus particulièrement en ce qui concerne la couche de thème : quelles sont les nouveautés par rapport à la version 7 ? Comment s'adapter lorsque l'on connaît déjà Drupal ? Entrons dans le vif du sujet.

Romain JARRAUD
Directeur Technique / Responsable Formation chez Trained People
Passionné, évangéliste, missionnaire Drupal depuis 2008.
romainj sur drupal.org

Ce que l'on voit d'un site Web dans un navigateur dépend de ce dernier, mais surtout du thème utilisé. Lorsque l'on souhaite créer un site Web, c'est classiquement que l'on a du contenu que l'on veut mettre en avant et partager, pour des raisons aussi bien artistiques que commerciales. On a donc deux aspects : le fonctionnel et la présentation. Le premier est de la responsabilité de Drupal et des modules, tandis que cette dernière relève du thème. Le theming est l'art de montrer.

Pratiquement, tous les modules produisent des données de façon générique ; ils ont donc besoin de les afficher d'une façon ou d'une autre. Ils proposent pour ce faire des templates par défaut (par exemple le module node définit le hook de thème "node" et son fichier de template *node.html.twig*). En pratique ils passent au système des données avec un certain formatage (via un render array).

Drupal doit alors déterminer quel est le "bon" template à utiliser. Il y a le template par défaut mais il peut être surchargé et Drupal peut potentiellement en utiliser un plus précis dépendant du contexte. On parle de suggestion de hook de thème. On dispose de suggestions de base, mais aussi de suggestions ajoutées par le thème afin de disposer d'un template adapté à une situation bien particulière (par exemple en fonction du rôle de l'utilisateur, ou bien de l'heure courante...). Le système va d'abord chercher le template correspondant le mieux au contexte, puis s'il ne le trouve pas, utiliser le fichier par défaut (défini par le module correspondant).

Par la suite, Drupal va récupérer les données à formater, en donnant la possibilité à tous les modules et au thème de les altérer avant de les passer effectivement au template. Cette modification des données est réalisée par les différentes fonctions de preprocess. Le thème peut modifier ou ajouter en dernier des données à afficher, ce qui lui donne la main sur tous les modules.

TWIG

L'adoption de **TWIG est un changement majeur par rapport aux versions passées de Drupal. TWIG** est un langage de templating destiné aux intégrateurs. C'est une librairie externe qui a été intégrée à Drupal 8, remplaçant ainsi le moteur de thème PHPtemplate... TWIG utilise une syntaxe simple et lisible. L'abandon du PHP dans les fichiers de template est une très bonne chose : plus simple à lire et mieux sécurisé.

Versions avant D8 : l'affichage d'une variable sans contrôle de son contenu est un problème récurrent dans les templates utilisés avec **Drupal 6 et 7**. En résultent des failles de sécurité de type Cross Site Scripting (XSS) dues à une connaissance trop superficielle du PHP qui n'est pas un langage destiné aux intégrateurs.

TWIG pallie ce problème en échappant automatiquement le contenu des variables.

Mais **TWIG** offre bien davantage, avec entre autre la possibilité de créer des filtres, de faire des boucles et des conditions, ou bien encore de traduire des chaînes de caractères. Notons également que la documentation officielle est très bien faite. Alors plus d'excuse pour ne pas s'y mettre !

```
 */
#}
<div{{ attributes }}>
  {{ title_prefix }}
  {% if label %}
    <h2{{ title_attributes }}>{{ label }}</h2>
  {% endif %}
  {{ title_suffix }}
  {% block content %}
    {{ content }}
  {% endblock %}
</div>
```

Extrait du fichier /core/modules/block/templates/block.html.twig

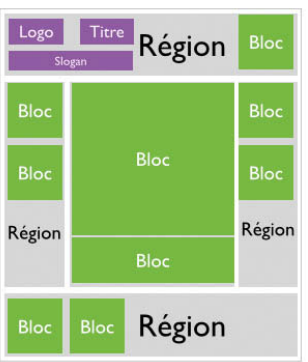
BLOCS

D8 apporte ici des améliorations notables :

Tout d'abord en généralisant le principe des entités et en l'appliquant au système de blocs : il est maintenant possible d'avoir plusieurs instances du même bloc et de créer des types de bloc fieldables (on peut ajouter des champs). Il est par exemple possible de retrouver un même bloc dans des régions différentes selon les pages.

De plus, le fil d'Ariane, les menus, les messages... désormais tout est bloc. On obtient ainsi un template de page ne contenant que les régions et les titres, le logo et le slogan du site. Cela semble logique avec le découpage de la page en régions dans lesquelles un administrateur peut positionner les blocs à volonté. Il est également possible de s'affranchir du fil d'Ariane, en n'assignant le bloc correspondant à aucune région du thème.

CSS



L'organisation des fichiers de CSS est du ressort du themeur, en tout cas du point de vue des noms et des styles de fichiers, Drupal 8 adopte l'architecture **SMACSS** (Scalable and Modular Architecture for CSS, *smacss.com*).

La gestion et le nommage des fichiers de CSS quant à eux, reviennent à l'intégrateur, facilitant les habitudes acquises sur d'autres systèmes. Quelle que soit la version de Drupal, on constate que de nombreux fichiers sont plus ou moins bien nommés. C'est toujours le cas avec Drupal 8, mais le développeur est fortement encouragé à adopter l'architecture **SMACSS.** Cette dernière consiste à créer des fichiers de CSS en fonction du type de sélecteurs. Par exemple toutes les propriétés relatives aux balises HTML de base (souvent appelé *reset* ou *normalize*) doivent faire partie du groupe de fichier *base*. Lorsqu'il s'agit de styles propres à des éléments réutilisés sur l'ensemble du site (comme des éléments de formulaire

ou des boutons), on préfère les réunir dans le groupe *component* (et non module comme défini par **SMACSS**, mais qui pose un problème de terminologie évident avec ce que l'on appelle module dans l'univers drupalien).

JS

Javascript est un incontournable du développement Web actuellement et Drupal 8 ne déroge pas à la règle. De très nombreuses bibliothèques et nouveaux frameworks sortent chaque semaine. Il n'est pas facile de faire coexister différentes librairies sur une même page Web car des conflits peuvent survenir comme par exemple des incompatibilités de versions ou bien un ordre de chargement inadapté.

Drupal 8 propose un gestionnaire de bibliothèques permettant d'unifier la déclaration et de gérer les dépendances. Le développeur indique simplement en quoi consiste sa librairie (version, fichiers, dépendances...) et à quel moment il en a besoin. Drupal s'occupe alors du chargement et des dépendances pour toutes les librairies nécessaires.

Drupal 8 continue l'intégration de bibliothèques externes entamée avec jQuery et Drupal 5. Nous disposons ainsi nativement de jQuery, jQuery UI, Backbone.js, ou encore CKEditor pour ne citer que certaines d'entre elles. Le chargement se fait page par page; ainsi aucune librairie n'est chargée par défaut, en particulier pas jQuery.

Liste des librairies disponibles nativement :

- backbone
- classList
- ckeditor
- domready
- drupal
- drupalSettings
- drupal.active-link
- drupal.ajax
- drupal.annonce
- drupal.autocomplete
- drupal.batch
- drupal.collapse
- drupal.debounce
- drupal.dialog
- ...
- html5shiv
- jquery
- jquery.cookie
- ...
- jquery.ui
- matchmedia
- matchmedia.addListener
- modernizr
- normalize
- picturefill
- underscore

Il est conseillé d'utiliser le mode strict dans ses propres scripts comme cela est le cas pour les librairies déclarées par le système. Ceci permet entre autre de lever des erreurs silencieuses qui peuvent engendrer des bugs. Globalement l'intégration du Javascript est plus « propre ».

LIBRAIRIES

Déclaration : tous les assets doivent être déclarés sous forme de librairie. Il n'est plus possible de charger à la volée des CSS ou JS. Chaque librairie est un "objet" avec un titre, une version, un ensemble de fichiers... et d'autres propriétés (licence, dépendance...). La déclaration se fait via un fichier YAML.

```yaml
picturefill:
  remote: https://github.com/scottjehl/picturefill
  version: "2.3.0"
  license:
    name: MIT
    url: https://github.com/scottjehl/picturefill/blob/2.3/LICENSE
    gpl-compatible: true
  js:
    assets/vendor/picturefill/picturefill.min.js: { weight: -10, minified: true }
  dependencies:
    - core/matchmedia
```

Extrait du fichier /core/core.libraries.yml

Le chargement d'une librairie n'est pas automatique. Il est nécessaire d'indiquer au système à quel moment vous en avez besoin. Il est possible de demander un chargement sur l'ensemble du site (via le fichier .info.yml du thème), ou bien ponctuellement, par exemple pour un bloc en particulier.

Drupal 8 propose un système de dépendances entre librairies et prend ainsi en charge leurs chargements. Si votre librairie nécessite jQuery, alors vous déclarez votre dépendance à cette librairie et Drupal la chargera chaque fois que votre librairie est utilisée.

BREAKPOINTS

Les breakpoints peuvent être déclarés au système et ainsi sortir du contexte simple du CSS. Ils sont définis dans un fichier dédié, au format YAML. Ainsi Drupal peut les exposer côté back-end aux modules. C'est le cas du module Responsive Image (désinstallé par défaut) qui permet ainsi d'utiliser des styles d'image différents en fonction des breakpoints. On peut ainsi aborder l'Adaptive Design sereinement, en n'affichant pas forcément le même contenu sur la page d'accueil en fonction de l'appareil utilisé.

```yaml
bartik.mobile:
  label: mobile
  mediaQuery: ''
  weight: 2
  multipliers:
    - 1x
bartik.narrow:
  label: narrow
  mediaQuery: 'all and (min-width: 560px) and (max-width: 850px)'
  weight: 1
  multipliers:
    - 1x
bartik.wide:
  label: wide
  mediaQuery: 'all and (min-width: 851px)'
  weight: 0
  multipliers:
    - 1x
```

Fichier /core/themes/bartik/bartik.breakpoints.yml

CONFIGURATION

Le thème et la configuration du site ne sont pas totalement dissociés. Certains thèmes permettent à l'utilisateur de paramétrer le site d'un point de vue "rendu". Avec Drupal 8 la configuration est définie sous forme de fichiers YAML et stockée en base par le système. Par exemple chaque type de contenu, chaque style d'image ou chaque vue fait l'objet d'un fichier de configuration. Il est maintenant facile d'embarquer n'importe quel élément de configuration dans un module ou bien un thème. Il suffit d'exporter la configuration via le back-office de Drupal et de placer le fichier correspondant dans le dossier /themes/mon_theme/config/install.

```yaml
langcode: en
status: true
dependencies: {  }
name: mon_style
label: 'Mon style'
effects:
  f3e49851-a8e4-43e0-8f00-cda11e9eeca4:
    uuid: f3e49851-a8e4-43e0-8f00-cda11e9eeca4
    id: image_scale
    weight: 1
    data:
      width: 321
      height: null
      upscale: true
```

Fichier /themes/mon_theme/config/install/image.style.mon_style.yml

C'est lors de l'installation d'un nouveau thème que Drupal importe la configuration. Notez bien que cette configuration ne doit pas être déjà présente sur le site. Vous ne pouvez pas, par exemple embarquer le style d'image mon_style, si ce dernier fait partie de la configuration active. Le système de configuration de Drupal 8 ne fonctionne pas comme Features ! On ne peut pas re-déclarer ou surcharger une configuration active.

CONCLUSION

Le système de theming de Drupal 8 s'est largement amélioré par rapport à Drupal 7, sans pour autant abandonner certains principes puissants (surcharge de template, suggestion de hook de thème...). Il s'est même simplifié en abandonnant par exemple les fonctions de thème et de process et en apportant une plus grande simplicité dans l'écriture des fichiers de template (merci TWIG). Les intégrateurs vont apprécier ces évolutions qui vont grandement leur faciliter le travail.

Septembre

Meetup Firefox OS / 9 septembre

Deux conférences seront proposées dans cette réunion. La première sur B2G Installer, la seconde sur le Service Workers.
http://www.meetup.com/fr/Firefox-OS-France-User-Group/

Intel IoT Roadshow à Paris / 12 – 13 septembre

La mini-carte Edison d'Intel revient à Paris pour un grand hackathon ! Préparez-vous ! Lieu : Usine.IOT
http://events.bemyapp.com/events/view/france/paris/usine-io/intel-iot-roadshow-paris-1

Meetup Mesosphère / Spark / 14 septembre

Découvrez la très puissante plateforme Mesosphere. Attention meetup payant. http://www.meetup.com/fr/Paris-Big-Data-Classes/events/221738115/

Meetup Meteor à Paris / 15 septembre

La communauté meteor organise sa soirée de rentrée à Paris. Le meetup s'organise autour d'une table ronde et de lightning talks.
http://www.meetup.com/fr/Meteor-Paris/

Meetup Cozy Cloud à Lyon / 17 septembre

Découvrez Cozy Cloud, la plateforme, les fonctionnalités. Lieu pas encore défini. http://www.meetup.com/fr/Meetup-des-utilisateurs-de-Cozy-Cloud-en-France/

Meetup Cordova / PhoneGap / 23 septembre

Vous développez sur Cordova / PhoneGap, ce meetup est fait pour vous ! Vous y trouverez la communauté, des talks techniques et les dernières actus. http://www.meetup.com/fr/PhoneGap-Paris/

Octobre

Conférence Rebuild 1er octobre

Le 1er octobre se tiendra à Nantes la nouvelle édition de la conférence Rebuild dédiée aux technologies et outils Microsoft. L'événement est organisé par les communautés Microsoft avec plus de 40 sessions et ateliers techniques.

Au menu :

- Une table ronde "Pourquoi les solutions collaboratives Microsoft s'autosuffisent-elles ? avec l'intervention de Bernard OURGHANLIAN (Directeur Technique et Sécurité de Microsoft) - Pierrick MARTIN (DSI du CHU Nantes) - Tristan PIRON (Responsable Pôle Numérique du Groupe LA POSTE) - animée par Patrick GUIMONET
- Près de 40 sessions techniques réparties sur 3 parcours (Décideur, IT et DEV) pour vous aider à organiser votre planning,
- De rencontrer les professionnels du numérique présents sur le salon
- Un espace d'échange avec les experts Microsoft et MVP

Agenda des sessions : http://bit.ly/1FQpgR0

Novembre

Droidcon Paris 2015 : 9 & 10 novembre

La conférence Droidcon revient à Paris, pour 2 journées pleines de sessions, d'ateliers et de rencontres. Plus de 600 développeurs et experts Android sont attendus. Plus de 30 sessions techniques sont prévues. Co-organisé par le PAUG et BeMyApp. Site : http://droidcon.fr

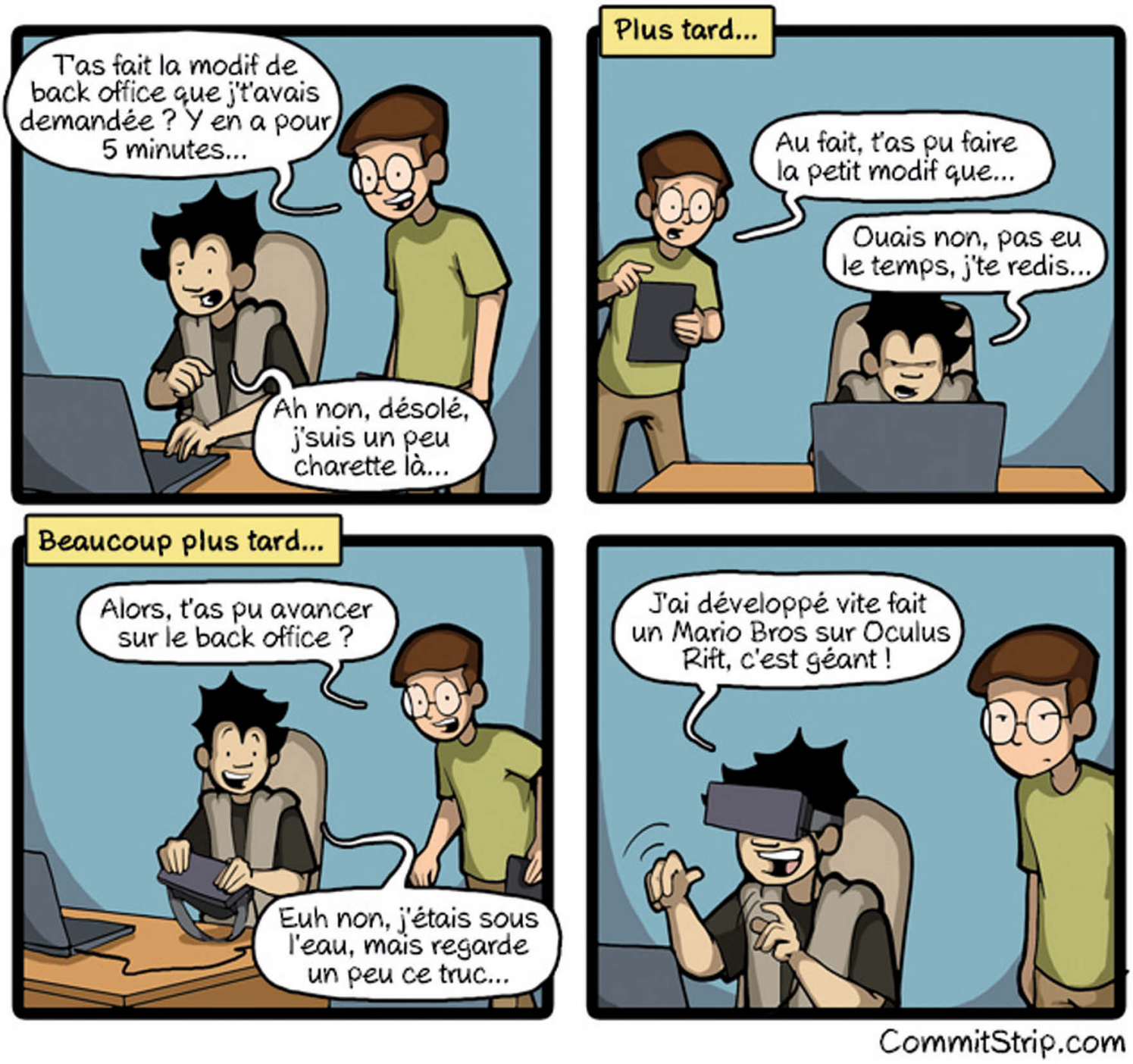
Paris Open Source Summit : 18 & 19 novembre (Paris)

Solutions Linux, le salon Linux et Open Source, et Open World forum fusionnent pour donner un unique grand événement Open Source français : Paris Open Source Summit. Il s'agit de concentrer les efforts et de faire de Paris une place incontournable du monde Open Source et Linux en Europe.

Forum PHP : 23 & 24 novembre (près de Paris)

Le monde PHP fête les 20 ans du langage et de la sortie de 7 PHP. Le Forum PHP est la référence française des développeurs et entreprises utilisant ce langage open source ! Cette année, 2 formats de conférences seront proposés : format 40 minutes et format 20 minutes.
Un des thèmes abordés sera PHP 7 qui sortira presque en même temps. Un track dédié sera proposé et le sujet promet d'être très largement abordé !
site : http://www.afup.org/pages/forumphp2015/

Sous l'eau...



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 : V. Loquet, S. Saurel

Experts : C. Louvet, G. Garnotel, M. Fery, C. Pichaud, R. Locatelli, S. Beaupuis, F. Dupuis, A. Vaché, A. Zanchetta, A. Bouzid, J. Bousquié, C. Villeuneuve, P. Martin, T. Desmas, D. Djordjevic, M. Frappat, B. Cornet, R. Jarraud, J. Lo Presti, C. Cathala

Photos/illustrations : Microsoft, Google, Git Hub, PHP, Drupal

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

Ce numéro comporte un encart jeté sur une partie du tirage : Microsoft Azure

Contacts
Rédacteur en chef : ftonic@programmez.com
Rédaction : redaction@programmez.com
Webmaster : webmaster@programmez.com
Publicité : pub@programmez.com
Evenements / agenda : redaction@programmez.com

Dépôt légal : à parution - Commission paritaire : 1215 K 78366 - ISSN : 1627-0908

Sur abonnement ou en kiosque

Le magazine des pros de l'IT

Mais aussi sur le web

Ou encore sur votre tablette

ikoula
HÉBERGEUR CLOUD

LE CLOUD GAULOIS, UNE RÉALITÉ ! VENEZ TESTER SA PUISSANCE

EXPRESS HOSTING

Cloud Public
Serveur Virtuel
Serveur Dédié
Nom de domaine
Hébergement Web

sales@ikoula.com
01 84 01 02 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