programmez!
#197 - Juin 2016
le magazine des développeurs

Swift de A à Z

Le nouveau langage d'Apple peut-il s'imposer ?

DevOps

De la démarche aux conteneurs : les mutations d'une philosophie qui peut vous aider

SQL Server 2016

Toutes les nouveautés

Android N

Google annonce du (très) lourd

© StockFinland

Développez des apps pour l'Apple Watch

REACT : la nouvelle terreur du web

Windows 10 : la communication inter-applications

M 04319 - 197 - 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

MICRO FOCUS | SERENA™

Enterprise DevOps

Développement accéléré
Déploiement sécurisé
Mise en service en toute sérénité

www.serena.com
01 70 92 94 94
frinfo@serena.com

RTFM

Ah le fameux « lis le putain de manuel » (Read The Fucking Manual) ! Que ferait-on sans lui ? Comme tout geek, on se lance sur un nouveau matériel, un nouveau code ou gadget sans vraiment faire attention à un objet bizarre que l'on appelle « manuel »... Bon ok, c'est une espèce en danger car il est de moins en moins présent. Tout juste a-t-on droit à un : télécharger le manuel sur x.fr ou x.com. A condition que le lien soit présent, ce qui n'est pas toujours le cas. Bref, c'est : « gentil acheteur, merci, mais démerde-toi pour le reste ». Dommage, car certains manuels étaient aussi clair en Français que je lis l'Allemand.

Ah le miracle de la traduction automatique... C'est comme si tu traduis Raspberry Pi par Framboise Pi. Un peu con, non ?

Après quand tu te plantes, effectivement, on ne se prive pas de te le faire remarquer, du genre : tu as regardé le manuel ? Où as-tu mis le manuel ? Bizarrement, à chaque fois que je trouve un guide d'installation ou une doc, c'est poubelle... Mais le RTFM n'est jamais très loin. C'est à contre-cœur que tu lis la doc... On regarde bien les tutos, alors pourquoi pas la doc ?...

Oui, je sais, c'est dur de l'admettre, mais finalement, le RTFM peut sauver ta vie de geek et de développeur. Et en plus, tu gagnes du temps pour jouer au nouveau Doom ou faire une petite course sur Forza. Y'a des choses importantes dans la vie.

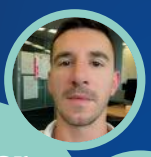
RTFM, ton meilleur ami ? Hum, faut peut être pas abuser... mais ce n'est pas comme si on t'annonce, que tu vas faire une base Access pour ton nouveau projet Web, ou pire, reprendre un projet IE6, voire, mais là c'est sadique, faire une mise en production le vendredi à 18h !

En attendant, lis ton magazine.

ftonic@programmez.com

















À lire dans le prochain numéro n°198 en kiosque le 30 juin 2016

Retour sur la conférence
Devoxx France 2016

Les devoirs d'été
- Réalité virtuelle avec CardBoard et Hololens
- Créer son jeu vidéo avec BabylonJS
- Hacker sa Tesla
- Comment créer un langage de programmation ?

.Net devient (presque)
open source

Intel vient de tuer le futur Atom Broxton et la puce SoFIA. Ce sont toutes les puces Atom pour mobiles qui sont arrêtées par le fondeur ! Une mauvaise nouvelle pour certains.

ADN va-t-il servir de stockage ? Microsoft a racheté une technologie de Twist Bioscience, un ADN synthétique pour le stockage de données.

Snif ! Apple a totalement abandonné WebObjects, un des outils phares des années 1990 – 2000 pour créer des sites dynamiques. R.I.P.

Le kit développeur **Hololens** est livré aux développeurs contre 3000 $.

Une impression 3D sans imprimante ? **Lix Pen** permet de créer des objets directement en dessinant comme avec un stylo normal sans papier, sans support ! Env. 140 €.

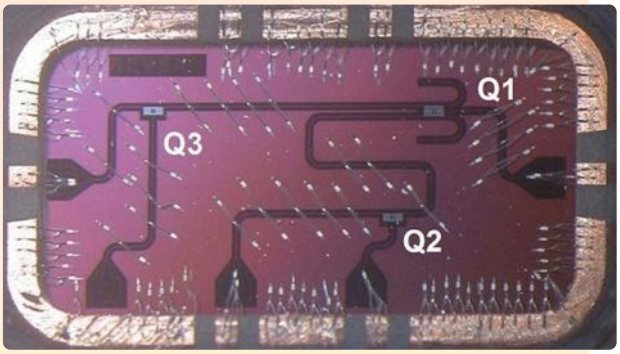
La SNCF investit dans le projet, un peu fou mais totalement novateur : Hyperloop !

Tesla propose des options en achat in-app pour ses voitures pour augmenter certaines fonctions comme la batterie...

La guerre des talents continue dans la Silicon Valley. La co-fondatrice de **Google X**, Yoky Matsuoka, aurait rejoint les équipes santé d'Apple.

IBM grille le quantique à Google et aux autres !

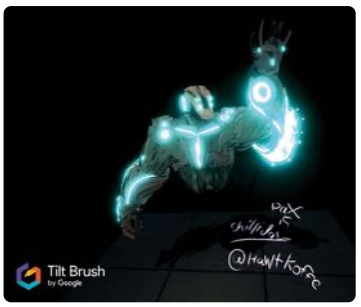
Lien : https://www.research.ibm.com/quantum/



Annonce un peu surprise début mai. IBM annonce le déploiement d'un service quantique sur la plateforme Cloud Bluemix : IBM Quantum Experience ! Ce service est issu d'IBM Research. Pour l'occasion, les équipes ont installé des processeurs dits quantiques. Il sera possible d'exécuter des algorithmes, des simulations. Chaque processeur quantique d'IBM a une puissance de 5 qubits. Pour IBM, dans les 10 ans à venir, il sera possible de construire des ordinateurs quantiques de 50 à 100 qubits. Dès 1981, Richard Feynman proposait de construire un ordinateur utilisant les principes de la physique quantique. Il y a quelques mois, Google et la NASA investissaient plusieurs millions dans cette recherche. Dans l'informatique actuelle, nous utilisons le binaire, donc 0 ou 1. Dans le quantique, nous dépassons cette frontière, ouvrant des possibilités immenses.
Le qubit est au-delà du 0 et 1 car il peut ainsi stocker 0 et 1 simultanément (= en même temps), c'est le principe de superposition. Ainsi, nous avons 4 valeurs simultanées possibles avec 2 quabits : 00, 01, 10 et 11. On multiplie la puissance et les possibilités mais cela nécessite aussi des processeurs totalement nouveaux et des outils et langages de programmation adaptés.

EDGE, OÙ ES-TU ?

Microsoft avait beaucoup d'espoir pour son nouveau navigateur Edge, remplaçant Internet Explorer dont certaines versions avaient traumatisé les développeurs Web (maudit IE 6).
Les dernières stats netmarketshare font mal : 0,54 % ! (Desktop). Dans le même temps, IE continue à décroître, Chrome est désormais à la hauteur et le dépasse légèrement. Firefox sombre sous les 10 %, Safari se rapproche des 5 %. Sur les terminaux mobiles, Chrome est presque à 50 %, Safari à 29 %, IE va peut être dépasser les 4 %... Un jour. On se dit, que w3schools pourrait être un peu plus sympa. Finalement, non, Edge arrive péniblement à 1,2 % en mars 2016 ! Chrome écrase tout à 69,9 %.



Un « stylo » pour la réalité virtuelle

La réalité virtuelle est aussi bien un marché matériel que logiciel, avec un potentiel énorme dans les prochaines années pour les startups, les développeurs et les constructeurs – éditeurs actuels. Google a dévoilé un outil surprenant, qui démontre parfaitement comment un éditeur peut investir dans ce nouveau marché : le Tilt Brush. Il s'agit d'une application pour le casque HTC Vive. Elle transforme les manettes en pinceau, en interface créative. Les démos sont assez impressionnantes ! Disponible sur Steam, au prix de 27,99 €.

L'INDEX TIOBE DU MOIS

Mai 2016	Mai 2015	Evolution	Langage	% de recherches	Evolution en %
1	1		Java	20.956%	+4.09%
2	2		C	13.223%	-3.62%
3	3		C++	6.698%	-1.18%
4	5	^	C#	4.481%	-0.78%
5	6	^	Python	3.789%	+0.06%
6	9	^	PHP	2.992%	+0.27%
7	7		JavaScript	2.340%	-0.79%
8	15	^^	Ruby	2.338%	+1.07%
9	11	^	Perl	2.326%	+0.51%
10	8	v	VB .NET	2.325%	-0.64%

Jusqu'au 4 Juillet 2016

COMMANDEZ WINDEV MOBILE 21

OU WEBDEV 21 OU WINDEV 21

ET RECEVEZ LE NOUVEAU

SAMSUNG Galaxy S7 edge

Ou choisissez parmi:

- 1x Tablette **Galaxy Tab Pro S** 12" Full HD+. WiFi. 4Go+SSD 128Go. Clavier. Windows 10 Home
- 1x **TV SAMSUNG 4K Ultra HD 138cm** WiFi
- 2x **Galaxy Tab S2** 9,7"
- 2x Smartphone **Galaxy A5**

D'autres matériels sont proposés sur le site www.pcsoft.fr
Aucun abonnement à souscrire.

SAMSUNG
Galaxy S7 edge
32 Go.
Choix de la couleur sur le site

SAMSUNG
Galaxy S7
32+128 Go.
Choix de la couleur sur le site

OPÉRATION POUR 1 EURO DE PLUS

Pour bénéficier de cette offre exceptionnelle, il suffit de commander WINDEV 21 (ou WINDEV Mobile 21, ou WEBDEV 21) chez PC SOFT au tarif catalogue avant le 4 Juillet 2016. Pour 1 Euro de plus, vous recevrez alors le ou les magnifiques matériels que vous aurez choisis. Offre réservée aux sociétés, administrations, mairies, GIE et professions libérales, en France métropolitaine. L'offre s'applique sur le tarif catalogue uniquement. Voir tous les détails et des vidéos sur : www.pcsoft.fr ou appelez-nous (04.67.032.032).
Le Logiciel et le matériel peuvent être acquis séparément. Tarif du Logiciel au prix catalogue de 1.650 Euros HT (1.973,40 TTC). Merci de vous connecter au site www.pcsoft.fr pour consulter la liste des prix des matériels et les dates de disponibilité. Tarifs modifiables sans préavis.

Elu
«Langage
le plus productif
du marché»

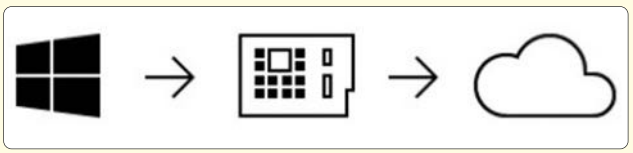
Document non contractuel. Logiciel professionnel.

DAS :Galaxy S7 DAS pour la tête : 0,406 W/kg | DAS pour le corps : 0,621 W/kg
Galaxy S7 edge DAS pour la tête : 0,264 W/kg | DAS pour le corps : 0,507 W/kg
Le DAS (débit d'absorption spécifique des téléphones mobiles) quantifie le niveau d'exposition maximal de l'utilisateur aux ondes électromagnétiques, pour une utilisation à l'oreille. La réglementation française impose que le DAS ne dépasse pas 2 W/kg.

www.pcsoft.fr

programmez!
le magazine des développeurs

DevCon #1 :
après-midi Windows 10 IoT



- 4 sessions architecture / développement / projets
- 2 mini-keynotes
- 1 session Q&A
- 1 Pizza Party

Les thèmes abordés :
architecture, miroir magique, Rapsberry Pi, programmation, créer des objets connectés...

15 juin 2016

Campus Microsoft France (Issy-les-Moulineaux)
A partir de 14h.

Inscription gratuite sur
www.inwink.com/events/devcon

Tous les détails sur www.programmez.com

Sponsors : Sigfox, Microsoft

Partenaires techniques : Cellenza, Infinite Square, Microsoft, Sigfox, SQLi,

La première conférence technique Programmez !

Un événement organisé avec la solution de gestion événementielle d'Infinite Square

juin

EclipseCon France - Toulouse : du 7 au 9 juin

Cette conférence organisée par la fondation Eclipse, se déroule à Toulouse au Centre de Congrès Pierre Baudis comme ces deux dernières années et sera totalement en anglais. Au menu, 3 journées bien remplies avec des ateliers et des meetups sur l'IoT et l'embarqué, le Cloud, l'optimisation des performances de l'IDE Eclipse, le DevOps, les Sciences, le Polarsys, le Java 9 ... Bref the place to be pour tout ce qui concerne Eclipse et sa communauté.
Inscription : https://www.eclipsecon.org/france2016/registration

Best Of Web 2016 - Paris : les meilleurs talks de l'année le 10 juin

En 2015, huit groupes *meetups* se sont fédérés pour proposer une conférence réunissant leurs meilleures présentations : Best Of Web. Une nouvelle édition nous est proposée en 2016, plus ambitieuse encore : cette fois ce ne sont pas huit mais douze *meetups* qui sont réunis, et dont le Best Of sera visible le 10 juin : AngularJS-Paris, Backbone JS Paris, Paris API, CSS Paris, etc.
Une première journée organisée le 9, en plus petit comité, où seront proposées des formations, ainsi que d'autres surprises. L'ambition demeure la même : proposer un événement dans l'esprit *meetup,* communautaire, abordable, participatif et convivial, différent des conférences plus classiques où s'expriment des stars. Best Of Web 2016 se déroulera les 9 et 10 juin 2015 à la Grande Crypte (Paris 16e) où 500 participants sont attendus.
Site : http://bestofweb.paris/
Twitter : https://twitter.com/bestofwebconf
Email : contact@bestofweb.paris

Riviera Dev - Nice (Sophia Antipolis) : les 16 et 17 juin

2 jours de conférences, 24 présentations au campus Sophia Tech a Sophia-Antipolis. Le CFP est ouvert jusqu'au 30 Avril donc le programme n'est pas encore acté mais il devrait y avoir du Java, du software craftmanshift, du Jenkins ...
Inscription : http://rivieradev.fr/

Agile France - Paris : 16 & 17 Juin

Pendant deux journées on y parle de méthodes agiles, de logiciels, de technologies, de bien être au travail, d'expérience utilisateur ... et de soi.
Plus d'infos : http://2016.conf.agile-france.org/

XAMARIN : QUELLES OPPORTUNITÉS POUR VOTRE ENTREPRISE ? 16 Juin

Tout au long de cette matinée, John Thiriet (Cellenza) accompagné de Matthew Larson (Xamarin) et Neil Smith (Xamarin) vous proposent de découvrir Xamarin, d'évaluer ses bénéfices, ses coûts et de partager avec vous quelques retours d'expériences afin que vous puissiez évaluer la pertinence de cette solution pour votre propre contexte.
A partir de 9h. Evènement orienté décideur.
Lieu : UX-REPUBLIC - 11 Rue de Rome, 75008 Paris-8e-Arrondissement
Une conférence Cellenza.
Informations : https://www.eventbrite.fr/e/billets-xamarin-quelles-opportunites-pour-votre-entreprise-25405304951

PAESSLER
THE NETWORK MONITORING COMPANY

DEVENEZ LE MAÎTRE DE VOTRE UNIVERS INFORMATIQUE !

Ne vous perdez pas dans l'espace infini de votre univers informatique !

Avec **PRTG Network Monitor** vous n'avez besoin que d'une seule solution pour surveiller tous les composants de votre infrastructure informatique: le matériel informatique, les applications et les environnements virtuels des leaders de l'industrie !

TELECHARGEZ ICI
www.paessler.fr/it-master

PRTG NETWORK MONITOR

Paessler AG
info@paessler.com
www.paessler.fr

691613/FR/20160503

Google I/O 2016 : entre réalité virtuelle et développeurs

Quelques heures avant d'imprimer ce numéro de Programmez !, Google avait inauguré sa grande conférence Google I/O. La session d'ouverture du 18 mai a dévoilé de belles choses. Les développeurs vont être contents ! Nous reviendrons très prochainement sur les annonces.

La rédaction

Google a vu les choses en grand : 7000 personnes pour la keynote et beaucoup d'annonces : Android N, Home, Wear 2.0, Android Studio 2.2, DayDream, machine learning, etc.

DayDream : la réalité virtuelle selon Google

Depuis l'arrêt des Google Glass, nous savions que Google travaillait activement sur le sujet. La bataille est désormais lancée et Google veut être présent et imposer sa vision.

DayDream est une plateforme complète de réalité virtuelle supportant les smartphones et les casques. Surtout, c'est tout un écosystème que Google veut créer autour de sa plateforme et l'éditeur mise beaucoup sur les applications et le contenu. Les apps seront cruciales pour le succès d'une plateforme. Un app store VR sera proposée. Google propose même un design de référence pour les casques et manettes intégrant des smartphones et plusieurs constructeurs sont déjà annoncés. Mais pour exploiter DayDream, les smartphones devront intégrer Android N (qui a un mode VR) et des composants spécifiques. Les premiers matériels compatibles seront disponibles cet autonme. CardBoard n'est pas abandonné. Google veut améliorer le confort d'utilisation et améliorer le support logiciel.

Android Studio 2.2 & Instant Apps

L'IDE dédié au développement Android va connaître une belle évolution avec la v2.2. Cette pré-version propose des nouveautés appréciables :

- nouveau socle IntelliJ
- amélioration des performances (ce qui ne sera pas un luxe)
- support de Java 8, amélioration sur la partie C++
- nouveau designer pour les interfaces
- amélioration des outils d'analyse de codes
- disponibilité d'un plugin Firebase (outil de statistiques pour les apps, racheté par Google)

L'annonce qui a suscité le plus d'intérêt est sans aucun doute Instant Apps, ou comment Google invente les apps Android jetables ! Pour tester une app, vous devez l'installer entièrement même si vous ne l'utilisez qu'une unique fois. Instant Apps permet de tester une app sans l'installer entièrement. Et si vous avez une app spécifique pour un paiement ou pour un usage unique, Instant App permet de le faire, là encore, sans tout installer. Bref, des apps « jetables ». Les développeurs doivent intégrer cette fonction pour que l'app soit Instant Apps. L'App sera disponible via une simple URL.

Est-ce que les autres plateformes mobiles suivront ?

Pour en savoir plus : https://developer.android.com/topic/instant-apps/index.html

Google Home contre Amazon Echo

Amazon avait ouvert une brèche : proposer un objet connecté pour les tâches quotidiennes à la maison, diffuser du contenu, etc. Google lancera son propre Amazon Echo : Google Home.

Il ressemble à un cylindre à installer chez soi. Il inclut différents composants et le tout nouveau Google Assistant. Home peut se connecter à d'autres matériels (s'ils sont compatibles), comme ChromeCast, Nest. Il inclut la diffusion audio, les commandes vocales, l'assistant, un minuteur, etc. Les développeurs pourront étendre l'usage de Home. L'objet devrait être disponible avant la fin de l'année. Aucun tarif annoncé.

Android N pour cette année

Google avait révélé Android N en mars dernier. L'éditeur veut laisser du temps aux développeurs pour tester et développer dessus. Cette version inclura de nombreuses nouveautés et améliorations. Une des priorités de N concerne les performances et la sécurité. Pour en savoir plus : voir notre dossier Android N dans ce numéro.

Android Wear 2.0

Après deux ans d'existence, Google veut renouveler le système Android dédié aux montres et mieux se différencier des plateformes concurrentes comme celle d'Apple...

Wear 2.0 va couper le lien entre montre et téléphone. Les apps seront totalement indépendantes. Cette version introduira aussi une nouvelle interface, de nouveaux mécanismes de notifications et de messages. La partie santé fera aussi un grand bond. Cette version devrait être disponible fin de l'année.

Abonnez-vous à programmez!

le magazine des développeurs

1 an ..... 49€ + 1€ = **50€ ***
11 numéros + un livre au choix

2 ans ..... 79€ + 1€ = **80€ ***
22 numéros + un livre au choix

Etudiant ..... 39€ + 1€ = **40€ ***
1 an - 11 numéros + un livre au choix

PDF ..... **30€ ***
1 an - 11 numéros + une vidéo au choix

Pour 1€ de + un livre numérique des Editions ENI au choix

valeur : 29,26 €

ou

valeur : 40,50 €

JavaScript
Développez un client Web en Full JavaScript

valeur : 29,99 €

ou

Applications mobiles multiplateformes
Technologie et contexte d'utilisation

valeur : 29,99 €

© StockFinland

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

(*) durée de l'offre du 31 mai au 30 septembre
Tarifs France métropolitaine

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
- ☐ ***Abonnement 1 an + Livre numérique*** ☐ ***AngularJS ou*** ☐ ***Git :*** 50 €
- ☐ ***Abonnement 2 ans + Livre numérique*** ☐ ***AngularJS ou*** ☐ ***Git :*** 80 €
- ☐ ***Abonnement étudiant 1 an + Livre numérique*** ☐ ***AngularJS ou*** ☐ ***Git :*** 40 €
  Photocopie de la carte d'étudiant à joindre

PROG 197
Offre limitée, *valable jusqu'au 30 juin 2016*

☐ M. ☐ Mme 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*

Méchante matrice

François Tonic
Programmez!

Un geekulture un peu spécial ce mois-ci, à mi-chemin entre la culture geek et la chronique technologique. Un sondage du cabinet Evans Data souligne la peur du développeur face à l'Intelligence Artificielle (IA) : 29,1 % des sondés ont peur que l'IA les remplacent (ou plutôt qu'on les remplace par une IA). Cela me rappelle un autre sondage : que les robots allaient détruire 7 millions d'emplois dans les prochaines années... Plus surprenant (ou pas) : 23 % ont peur que les plateformes qu'ils utilisent deviennent obsolètes. Là, dirons-nous c'est l'évolution normale de l'informatique et des technologies depuis le début de l'Homme.

Les fabricants de parchemins et les copistes n'ont pas apprécié l'arrivée de l'imprimerie en Europe et du véritable papier. Rupture technologique assurée.

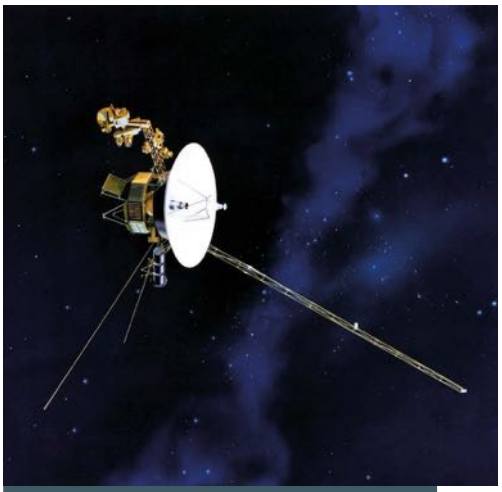
L'IA est un fantasme les plus marqués avec la robotique évoluée comme Chapie ou encore mieux, dans Battlestar Galactica, et surtout Caprica. Nous pouvons joindre HAL de 2001 ou le dépressif robot de H2G2. Nous voyons finalement les deux aspects de l'IA : le bon et le mauvais. Et ce n'est pas Blade Runner qui va nous dire le contraire.

Un projet peut faire flipper : Gigster. Ce projet est un moteur IA qui prendra une idée d'app (bien définie) et générera l'application et donc le code nécessaire... Le projet a été fondé en 2014 et a reçu fin 2015 un investissement de 10 millions $. Mais qui développe ce moteur ? Des développeurs et ingénieurs ! On se console comme on peut...

> *« La matrice est universelle. Elle est omniprésente. »*
> Morphéus / Matrix

Se pose la question suivante : les robots pourront-ils se révolter ? Oui. Mais pas demain, ni après-demain. Le robot peut être intelligent, d'une certaine manière mais il ne pense pas comme le cerveau humain. Mais on peut effectivement envisager des robots capables d'agir, de penser, de réfléchir, indépendamment de l'Homme. L'indépendance du robot vis-à-vis des humains. Etre capable de fabriquer d'autres robots, de développer son intelligence, d'apprendre, etc. Matrix, Terminator, Caprica / Battlestar Galactica... Le point de départ est la robotique au service des humains, puis de robots de plus en plus « intelligents ». Peu à peu, elle va échapper aux créateurs avec une capacité d'évoluer et de muter, avec un instinct de survie.

Star Trek propose une vision totalement intégrée où les machines et les humains vivent ensemble. Ce qui n'empêche pas d'avoir des robots-tueurs. L'épisode « the ultimate computer » (s02e24) montre comment un ordinateur, ici M-5, va peu à peu dévier de sa mission initiale. Son créateur va le défendre jusqu'au bout. Cette déviance et la corruption de sa mission initiale rappellent Nomad (« the changeling », s2e03) ou encore V'ger (Star Trek, le film). Nous pouvons aussi considérer les Borg, êtres hybrides, humains-robots, dans The Next generation et dans Enterprise. Hostiles aux autres espèces, les Borgs s'étendent et prospèrent en assimilant les mondes. Bien entendu, l'Homme n'est pas sans fautes. L'exemple le plus célèbre reste Khan Noonien Singh, dans sa version originale : s1e20 et La colère de Khan.



Retour vers le passé

Cela fait désormais partie de la geekulture, notamment depuis Star Trek, le film, qui reste tout de même de la bonne SF : la NASA cherchait un développeur parlant couramment les langages vieux de 60 ans ! Les sondes Voyager furent lancées en été 1977. Mais les ingénieurs et informaticiens qui ont créé et géré le projet sont aujourd'hui à la retraite. L'ultime ingénieur encore en fonction est parti il y a 1 an. Mais pour continuer à faire fonctionner le projet et le maintenir, la NASA n'avait plus d'ingénieurs capables de lire, de maintenir et d'écrire du code assembleur et FORTRAN ! Et les ressources des sondes sont très limitées : 64 ko ! Il faut actuellement 17 heures, par ondes radio, pour communiquer avec les Voyager 2 !

Mon agenda série

Silicon Valley saison 3 : depuis le 24 avril

GoT saison 6 : les 4 derniers épisodes de la s6 en juin !

Mr Robot saison 2 : démarrage le 13 juillet

Halt and catch fire saison 3 : juin ?

Twin Peaks saison 3 : seulement en 2017 sauf surprise

The man in the high castle saison 2 : aucune annonce officielle

Sur abonnement ou en kiosque

Le magazine des pros de l'IT

Mais aussi sur le web

Ou encore sur votre tablette

Retour sur Xamarin Evolve 2016

Xamarin Evolve est le plus grand évènement au monde pour les passionnés de mobilité et de développement « cross Platform ». Il a eu lieu à Orlando, en Floride, du 24 au 28 avril dernier. C'est le rendez-vous immanquable des développeurs et des acteurs incontournables de l'industrie informatique. Xamarin Evolve en chiffres, ce sont plus de 1700 experts mobilité invités en Floride pendant 4 jours, mais aussi des dizaines de milliers d'internautes qui suivent l'évènement en direct à travers plus de 150 pays. Une fois n'est pas coutume, la société Xamarin place la barre très haut : révélations, démonstrations, nouveautés, annonces, ...

François BOTTE
Microsoft Technical Lead,
SQLI Toulouse

Les SDKs de Xamarin en open source

Ce n'est pas vraiment une grosse surprise puisque depuis le rachat de Xamarin par Microsoft, le passage en open source de certains SDKs était déjà connu. Mais c'est durant la « keynote » que Miguel de Icaza, co-fondateur de Xamarin et responsable de l'engineering mobile development tools de Microsoft, annonce l'ouverture officielle du tout nouveau site (open.xamarin.com) dédié à l'open source pour ses trois SDK principaux qui sont :

- Xamarin.iOS et Xamarin.Mac : applications mobiles natives pour iOS, watchOS, tvOS et OS X avec .Net ;
- Xamarin.Android : applications mobiles natives pour Android, Android Wear et Android TV avec .Net ;
- Xamarin.Forms : interface graphique native pour iOS, Android et Windows, basée sur du code partagé Fig.1.

Xamarin désire ainsi renforcer ses produits en permettant à tout le monde de contribuer de différentes manières à la belle aventure. Que ce soit par le développement de nouveaux contrôles utilisateurs compatibles sur toutes les plateformes, de rapports de bug précis et efficaces, d'ajouts de nouvelles fonctionnalités manquantes, de tests de performance ou de discussions, ou encore d'approches stratégiques sur les solutions techniques à adopter : Xamarin permet à chacun de s'exprimer et d'apporter sa pierre à l'édifice, afin de rendre ses produits toujours plus performants.

Xamarin Studio 6

Xamarin Studio est l'environnement de développement pour iOS. Sous Windows, avec plus de 15 ans d'existence, Visual Studio n'a plus besoin de faire ses preuves. Il fallait donc que Xamarin marque le coup. Ainsi lors de cette présentation, c'est toute une salle qui a ovationné l'arrivée de la version 6 avec son lot de nouveautés :

- L'intégration de Roslyn permet à Xamarin Studio de ne plus avoir à rougir de son grand frère qu'est Visual Studio. En effet, cela place l'éditeur au même niveau que ce dernier concernant les fonctionnalités de complétion de code. De plus, l'IntelliSense continue de fonctionner même si le code contient des erreurs de compilation : Fig.2.
- Le nouveau template de projet permet une compatibilité entre Xamarin Studio et Visual Studio. Cela permet de travailler sous les deux environnements et de bénéficier ainsi des avantages de chaque IDE.
- Nouvelle version pour un nouveau look, Xamarin Studio a été complètement redessiné et repensé afin d'offrir une meilleure expérience aux développeurs Mac. De plus, il embarque le fameux thème « Dark », cher à certains utilisateurs.
- Les développeurs F# pourront désormais utiliser Xamarin Studio pour créer des projets partagés ou des « Portable Class Library ». L'IDE se positionne ainsi comme le seul et unique environnement de développement supportant le F# dans l'univers du « Cross-Platform » .

iOS USB Remoting

Les utilisateurs de Microsoft Visual Studio n'ont pas été délaissés malgré tous les efforts apportés sur Xamarin Studio. En effet, conscient des soucis de performances et de tests des applications iOS à partir de Windows, Xamarin annonce très prochainement l'arrivée de l'iOS USB Remoting. Il suffira de raccorder son appareil à sa machine Windows 10 via le port USB et le tour sera joué. A partir de Visual Studio, il ne restera plus qu'à sélectionner l'appareil pour lancer le débogage et tester l'application Fig.3.

iOS Simulator Remoting

Le Remote Simulator to Windows fait lui aussi partie des dernières nouveautés annoncées lors de cette Evolve2016 de Xamarin. Il permet tout simplement de tester et débugger vos applica-

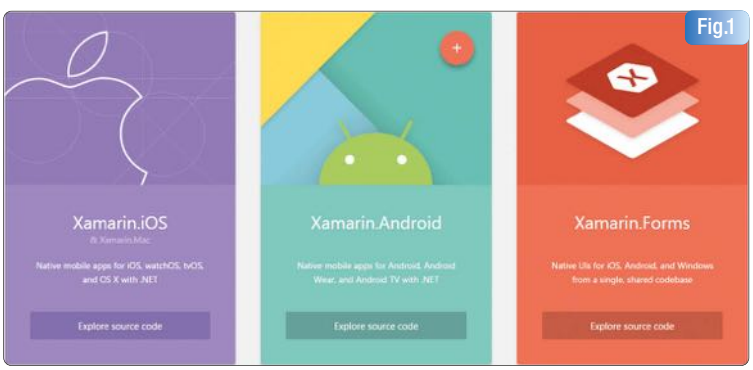


Fig.1

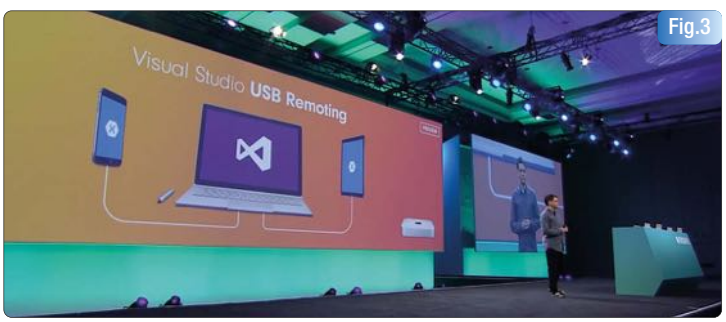


Fig.3

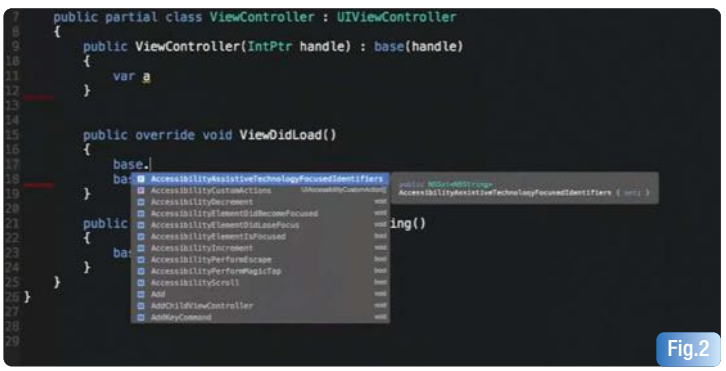


Fig.2

Décoder **le passé**

Kiosque | Abonnement | PDF

www.pharaon-magazine.fr

tions iOS en restant dans l'environnement Visual Studio. Néanmoins, il vous faudra toujours un Mac connecté en réseau pour compiler votre code. Il est déjà disponible en version bêta : Fig.4.

Xamarin.Forms Previewer

C'est incontestablement la grosse nouvelle de cette session 2016. Lors de la démonstration devant les utilisateurs, l'annonce a vraiment fait sensation. Xamarin.Forms Previewer permet d'avoir un rendu en temps réel des interfaces utilisateurs développées en XAML à partir de Xamarin Studio. C'est donc un gain de temps très précieux puisqu'il est dorénavant inutile de devoir compiler et exécuter l'application pour constater le rendu. De plus, Xamarin.Forms Previewer permet également de simuler un affichage « Phone » ou « Tablette » aussi bien sous iOS que sous Android : Fig.5.

Xamarin.Test.Recorder et Test.Cloud.Live

Xamarin.Test.Recorder permet d'effectuer des tests sur l'interface graphique. Il enregistre le comportement de l'utilisateur dans un script qui pourra être automatisé et permet l'exportation vers Xamarin.Test.Cloud. Aujourd'hui, il devient disponible sous Visual Studio et permet de générer en temps réel l'écriture de tests unitaires dans l'IDE de Microsoft : Fig.6.

Mais Xamarin frappe encore plus fort avec l'arrivée de Test.Cloud.Live. Il sera bientôt possible d'exécuter et de déboguer directement à partir de l'IDE de Microsoft vers un appareil mobile disponible sur le Cloud de Xamarin. C'est une belle initiative que propose la société et qui permet ainsi de s'affranchir des différentes machines virtuelles permettant de simuler des appareils Android ou iOS : Fig.7.

Xamarin.iOS et Xamarin.Android

Xamarin.iOS et Xamarin.Android bénéficient également de pas mal de nouveautés. On y retrouve en effet une meilleure gestion des architectures permettant la réduction du poids des applications déployées, la prise en charge de ModernHttpClient (les opérations HttpWebRequest du Framework .Net sont transformées en NSURLSession native pour iOS par exemple) accélérant les opérations réseaux et permettant l'encryption, etc.

Xamarin.Forms

Enfin, concernant les nouveautés sur Xamarin.Forms, un gros effort a été fait sur les accès aux données (et plus particulièrement sur le Cloud de Microsoft Azure). L'arrivée des « DataPages » permet à partir d'une data source (de type « Json » par exemple) de générer automatiquement des formulaires de présentation sans une seule ligne de code (ou presque). Cela peut s'avérer très utile dans le monde de l'entreprise pour créer rapidement des applications type démonstration, puis les adapter aux clients grâce aux thèmes de personnalisation disponibles : Fig.8.

Xamarin Evolve 2016 : Conclusion

S'il fallait conclure en un seul mot sur cet évènement, cela serait sans aucun doute : ouah ! Quand on pense qu'il y a moins d'un an, Microsoft n'arrivait toujours pas à se positionner sur le marché du « Cross Platform ». Aujourd'hui, on peut rêver d'un monde dans lequel plateforme Windows Universelle (UWP) et Xamarin ne feraient plus qu'un, donnant naissance à de véritables applications universelles à partir d'un seul et même outil et d'un seul langage...

En Xamarin le mot XAML se traduirait : Xamarin And Microsoft Loves.

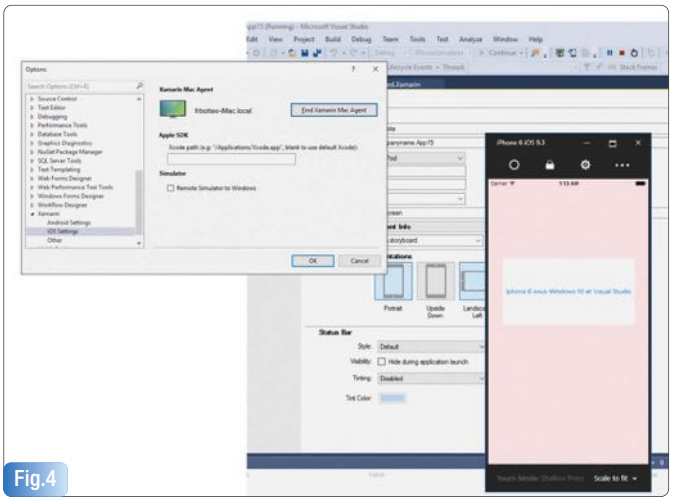


Fig.4

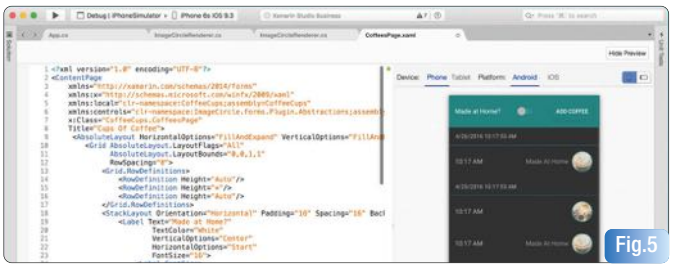


Fig.5

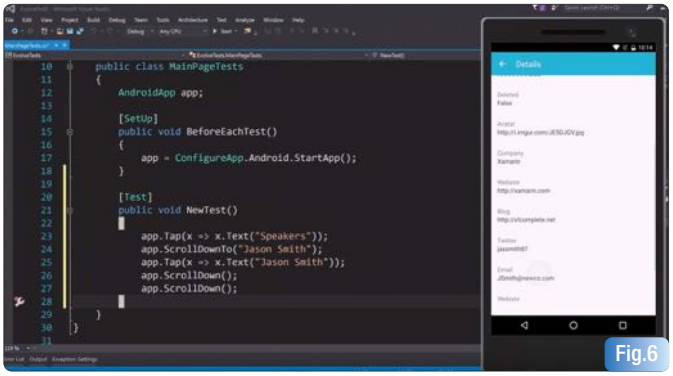


Fig.6

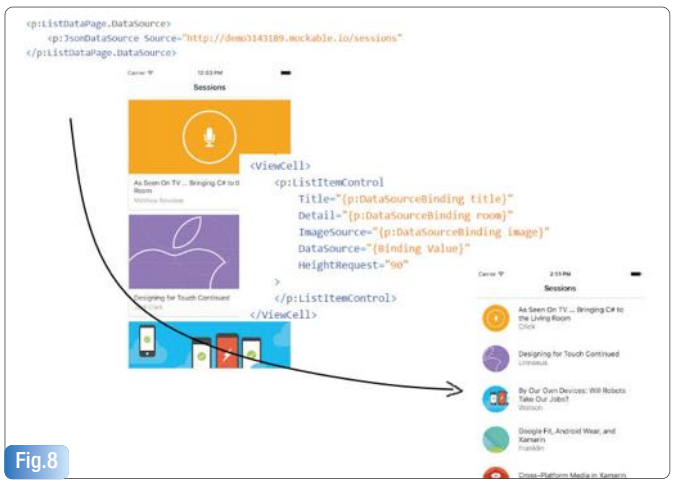


Fig.8

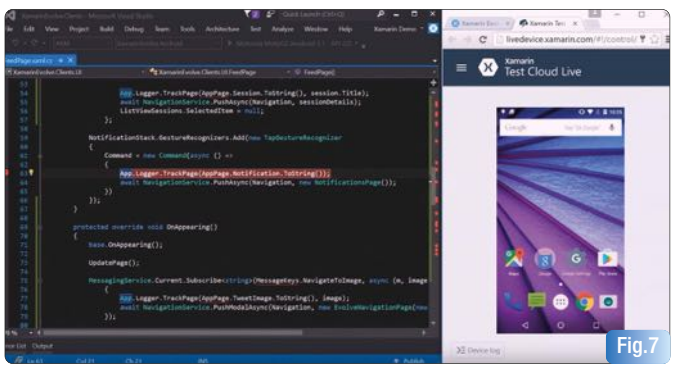


Fig.7

Swift : tout nouveau tout beau

S'il était au départ réservé au développement sur les plateformes d'Apple, Swift est aujourd'hui un projet Open Source hyperactif et son ouverture à Linux est la promesse d'une grande variété d'utilisations. S'il atteint ses ambitions élevées, il pourrait bien devenir un langage omniprésent dans les années qui viennent. Je vous propose de voir d'abord les grands principes du langage, son installation sur Linux et les spécificités du projet Open Source, puis dans une seconde partie, ses aspects techniques plus en détail.

Pascal Batty
Expert iOS chez **SOAT**

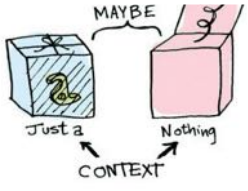
Depuis 2001, les plateformes de développement d'Apple ont été associées à un seul langage : Objective-C.
Il y a bien eu un support officiel de Java, WebObjects pour le Web, des API en C avec Carbon... Mais aujourd'hui quand on pense au développement sur une plateforme Apple, on visualise bien Objective-C : ce superset du C qui date des années 80, avec ses noms de méthodes à rallonge, sa gestion mémoire à la main, et surtout ses nombreux crochets !
Et pour cause : le langage, ainsi qu'une grande partie des fondations qui composent ce qu'on connaît maintenant sous le nom d'OS X ont été injectés dans l'écosystème d'Apple lors du rachat de NeXT en 1996. De nombreuses API que les développeurs OS X et iOS utilisent encore aujourd'hui ont été conçues dans les années 90, et elles marchent toujours très bien.
C'est en 2008 qu'Objective-C a repris du poil de la bête. Avec la sortie du SDK iPhone, il était le passage obligé pour développer sur cette nouvelle plateforme très attractive. De plus, Apple a passé des années à faire évoluer le langage pour le garder frais et actuel. Malgré cela, son côté austère en fait toujours un obstacle à l'apprentissage du développement iOS et, même s'il peut continuer d'évoluer, son vieil âge implique qu'il ne pourra probablement jamais rattraper les langages plus récents en termes de sécurité et de facilité de développement. L'obsolescence arrive. En juin 2014, lors de sa conférence annuelle aux développeurs, après les annonces des mises à jour d'OS X et iOS, Apple surprend avec l'inauguration d'un tout nouveau langage : Swift ! Il est encore très jeune mais fonctionnel, et l'ambition derrière Swift est qu'il devienne un "langage à tout faire" : depuis l'applicatif haut niveau jusqu'au système d'exploitation, sans compromis sur la lisibilité, la sécurité ou la performance. Ambitieux !
De plus, plutôt que de faire table rase de toutes les API tout-à-fait efficaces que les développeurs utilisent déjà pour développer sur iOS et OS X, le langage est interopérable avec le runtime Objective-C sans efforts.
Le message est clair : vous pouvez dès aujourd'hui écrire vos applications iOS et OS X en Swift, ou intégrer du code Swift à vos projets Objective-C existants.
Pour être honnête, en 2014 et sur une bonne partie de 2015, le langage était très instable et l'utiliser en production relevait plus de l'aventure risquée. Aujourd'hui, même s'il est difficile d'avoir des chiffres précis, il semble qu'une part notable de sociétés aient décidé de sauter le pas et d'intégrer Swift dans leurs applications, voire de commencer de nouveaux projets en 100% Swift, même si le langage n'est toujours pas à l'abri de "breaking changes" dans son état actuel. D'un autre côté, ce n'est pas la première transition technique qu'Apple rencontre, et les précédentes ont été très bien accompagnées (PPC vers Intel, 32bits vers 64 bits sur Intel puis ARM). Les dernières actualités du langage sont un passage en version 2.0 en juin 2015, puis le passage en projet Open Source en décembre. Alors qu'y a-t-il sous le capot de Swift ?

Tour d'horizon

"Objective-C without the C", c'est la phrase choc que Craig Federighi – SVP Software Engineering – avait choisie pour introduire le langage. En réalité, Swift s'assume comme un membre de la "C Family" et souhaite rester proche de cette syntaxe qui a fait ses preuves. Si vous venez du C++, du C# ou du Java, vous ne serez pas complètement dépaysé.
Pour être un "langage à tout faire" et même permettre l'écriture d'un système d'exploitation, Swift a besoin d'un benchmark de performances hors-norme.
À son annonce, Apple présentait des comparatifs vertigineux avec Objective-C (dont les performances sont à peine inférieures à celles du C et de C++). Pour garder la forme, Swift se démarque de son grand frère par plusieurs aspects, tout en gardant les avantages d'un langage moderne et "high-level". Mais comment fait-il ?
Premièrement, il est fortement typé. Pour voir ce que ça change, parlons de cette simple ligne de code :

```
var chocolate = milk + cocoa
```

Dans un langage comme Javascript, le type de la variable `chocolate` serait déterminé au moment de l'exécution du langage, en fonction du code à exécuter pour l'opérateur + qui dépend des types de `milk` et `cocoa`. En JavaScript d'ailleurs, le type de toute variable peut changer à chaque ligne. Swift utilise un système de Type Inference pour déterminer à la compilation le type d'une variable en fonction de l'instruction qui a défini son assignation, et il est impossible d'en changer après. Cela permet au CPU de faire moins d'efforts à l'exécution.
Son système de compilation est d'ailleurs assez complexe, passant en tout par 3 étapes entre le code source et le code machine. Le code Swift est d'abord transformé en Swift Intermediate Language (SIL) avant d'être optimisé puis digéré par la stack LLVM habituelle (celle utilisée par Apple pour Objective-C). Ces multiples étapes permettent d'intégrer de nombreux raccourcis comme par exemple le fait que l'addition de deux entiers en Swift soit réellement traduite en assembleur comme l'addition de deux entiers.
Ce système de type crée un environnement de développement plutôt sécurisé : en utilisant Xcode, l'IDE d'Apple, le compilateur s'assure pendant l'écriture du code que l'utilisation des types est cohérente.
Certaines notions s'ajoutent à celle-ci pour renforcer cette sécurité : l'immutabilité, déjà très importante en Objective-C, fait ici partie du langage lui-même. Swift intègre également la notion d'Optional, une implémentation de la Maybe Monad utilisée dans les langages fonctionnels,



qui permet d'expliciter si une valeur peut ou non être nil (équivalent à Null). Ces notions peuvent paraître un peu frustrantes lors de la découverte du langage, mais elles s'avèrent à terme être un atout considérable : le compilateur va éliminer d'office une bonne partie des erreurs que l'on pourrait autrement rencontrer en runtime.

Quelques exemples

Pour définir une variable, on utilise le mot-clé var. Il n'est pas nécessaire de spécifier son type si on lui assigne une valeur au moment de la déclaration, le système de Type Inference fera le travail tout seul. Pour spécifier le type, on utilise : entre le nom de la variable et le type. Notez l'absence de point-virgule en fin de ligne :

```
var name = "Riker"
var age:Int = 29
```

Bien sûr la définition du type est obligatoire si l'on décide de ne pas assigner de valeur au moment de sa déclaration. Pour faire la même chose en Objective-C, il fallait utiliser des pointeurs et la notation littérale des chaînes :

```
NSString *name = @"Riker";
int age = 29;
```

Swift apporte le mot-clé let qui permet de définir une constante. Contrairement aux constantes en C, la valeur qu'on lui assigne peut-être calculée à l'exécution. Pour les types références (classes), cela signifie que le pointeur ne peut pas changer, mais que l'état de l'objet le peut encore ; tandis que pour les types valeur (structures, enums), l'état lui-même d'une let devient immutable. Pour le reste la notation est la même.

```
let answer = random()
```

Le système de Type Inference fonctionne aussi sur l'itération dans les différents types de collections. Ici, team est un tableau de chaînes de caractères et l'utilisation d'un template string est claire :

```
let team = ["Smith", "Peck", "Baracus", "Murdock"]

for member in team {
   print("\(member) is part of the team")
}
```

En Objective-C, un NSArray est un tableau de pointeurs et peut contenir n'importe quel type, même d'un élément sur l'autre :

```
NSArray *team = @[@"Smith", @"Peck", @"Baracus", @"Murdock"];

for (NSString* member in team) {
   NSLog(@"%@ is part of the team", member);
}
```

L'appel du constructeur d'un type se fait sans mot-clé spécifique, comme si l'on appelait une fonction portant le nom du type :

```
let yesterday = NSDate(timeInterval: -24*60*20, sinceDate: NSDate())
```

La même chose en Objective-C donnerait cela :

```
NSDate *yesterday = [[NSDate alloc] initWithTimeInterval:-24*60*60 sinceDate:[NSDate date]];
```

Si l'on perd les crochets au passage, les paramètres nommés restent présents. On le voit dans cette construction de date avec les paramètres timeInterval: et sinceDate:. Cela permet une lisibilité accrue du code produit, ainsi qu'un bridge plus transparent entre Swift et Objective-C.
Enfin, les blocks/lambdas/closures sont un aspect incontournable d'une

programmation moderne. Les closures sont très importantes dans Swift : toute fonction est une closure nommée. Il est possible de les instancier puis de les manipuler, et bien sûr de les passer en paramètre d'une autre fonction comme map(), un grand classique.

```
let numbers = (1...100).map { x in x * 2 }
```

Effectuer cette opération qui permet d'obtenir tous les nombres de 1 à 100 (avec une notation de range très lisible) multipliés par 2 nécessite beaucoup plus d'étapes en Objective-C :

```
NSMutableArray *numbers = [@[] mutableCopy];

for (int i = 1; i <= 100; i++) {
   [numbers addObject:[NSNumber numberWithInt:i * 2]];
}
```

On voit bien ici la différence entre du code impératif et du code déclaratif, l'opération importante (multiplier par 2) est perdue au milieu de détails d'implémentation d'une boucle for côté Objective-C alors qu'elle est mise en évidence dans l'utilisation du map() côté Swift. Les fonctions filter, reduce et sort sont disponibles dans tous les types de collections, même dans ceux que vous fabriquerez grâce aux Protocol Extensions ! Objective-C permet également l'utilisation de blocks mais sa syntaxe est beaucoup plus alambiquée.

The Swift Way

Swift est un langage Objet et vous pouvez écrire votre code comme vous le feriez en Java ou en C#, mais il y a un "Swift Way" qui se dessine progressivement et repose sur des notions différentes.
Les types valeur sont très puissants en Swift avec l'utilisation des Protocoles et des Extensions hérités d'Objective-C.
La quasi-totalité des types de la Standard Lib ne sont d'ailleurs pas des classes mais des structures ! Pour cette raison, la culture déjà très forte autour du langage tend à pousser l'utilisation des "value semantics" plutôt que les types références que l'on a l'habitude d'exploiter dans les langages Objets.
Les fonctions et closures y sont également très puissantes et les notions de Programmation Fonctionnelle font partie des sujets qui font beaucoup parler d'eux dans cet écosystème déjà bouillonnant.
Le langage parvient donc à être à la fois familier dans sa structure tout en proposant des notions plus modernes.

Comme il grandit vite !

Depuis son introduction, le langage a déjà connu de nombreuses mutations. En juin 2015 était annoncée la version 2.0 qui ajoutait quelques fonctionnalités très notables.

Elle apporte une gestion des erreurs sous forme de try/catch, une absence très remarquée dans la version précédente. On trouve également dans la même catégorie les nouveaux mots-clés guard, qui permet d'écrire une clause if pour sortir du scope, et defer qui permet de garantir l'exécution d'un bloc de code en sortie de scope.

Ces nouveaux fonctionnements ne changent pas littéralement la face de Swift, mais ils peuvent créer des conventions, comme guard dont l'existence implique qu'il est préférable d'éliminer les cas limites en début de scope plutôt que de chercher à ne faire qu'un seul return.

Pour la gestion des erreurs, elle semble pour le moment très élémentaire, il est pour le moment incertain si l'on est plutôt du côté Java/C# où les exceptions sont souvent remontées puis gérées par le code, ou du côté d'Objective-C où les exceptions n'étaient utilisées que pour de vraies erreurs techniques souvent irrécupérables.

Une autre amélioration notable est celle du Pattern Matching, ce système qui permet de décorer des structures algorithmiques avec des conditions avancées, en utilisant par exemple le mot-clé where dans une boucle for in. Une bonne utilisation du Pattern Matching peut rendre un code complexe tout-à-fait lisible et son amélioration est la bienvenue.

Enfin, les Protocol Extensions sont un nouvel aspect qui peut changer considérablement la façon d'envisager une architecture. Les Protocols en Swift (comme en Objective-C) sont l'équivalent aux Interfaces dans les autres langages Objet. Les Extensions viennent aussi d'Objective-C, elles permettent l'ajout de fonctions ou de propriétés calculées à un type existant, mais aussi la conformance à un Protocol. Les Protocol Extensions permettent d'ajouter une implémentation par défaut aux méthodes d'un Protocol, et ainsi d'autoriser une certaine forme d'héritage multiple.

C'est cela qui permet par exemple de bénéficier automatiquement de toute la gamme de fonctions map(), filter(), reduce() dès lors que l'on se conforme au protocole SequenceType, alors qu'avant Swift 2 ces fonctions étaient dans le scope global, ce qui pouvait créer une confusion au niveau de la syntaxe. Et depuis son passage en Open Source, le langage n'a pas cessé d'évoluer.

Un projet Open Source

En effet, début décembre 2015, Apple a mis Swift à disposition sur Github. Si la société avait déjà proposé ou participé à des projets Open Source comme WebKit, OpenDarwin et LLVM, c'est bien la première fois qu'elle montre autant d'implication dans le processus. Ici, nul "initial commit" qui masquerait tout le déroulement du développement du projet, tout l'historique jusqu'à 2010 est visible et les participants sont représentés par leurs noms, et pas masqués derrière une entité abstraite Fig.1.

Meilleur encore, le projet est complètement participatif. Que ce soit sur la maintenance du langage, sa documentation et même son évolution : le projet accepte les Pull Requests ! Tout un repository est d'ailleurs dédié à l'avenir du langage (swift-evolution), mais si vous souhaitez y participer vous devrez sûrement être actif sur les mailing-lists qui permettent des échanges (très fournis !) sur les différents sujets du moment.

Tout le processus de contribution est scrupuleusement décrit sur le nouveau site du langage : Swift.org, le point de départ idéal pour qui souhaite démarrer sur le langage Fig.2.

Si on peut voir ici l'expression d'une nouvelle ère d'Apple, plus transparente et participative, c'est également le signe que la société se donne les moyens de ses ambitions : faire sortir Swift du simple langage propriétaire cantonné à son écosystème (iOS, OS X, watchOS, tvOS), afin de lui permettre de devenir ce fameux langage à tout faire.

Enfin, détail qui a son importance, avec sa sortie en Open Source, le langage a également été porté sur Linux !

Support de Linux

Si l'IDE Xcode reste une exclusivité Mac, il est aujourd'hui tout-à-fait possible de coder en Swift sur Ubuntu 14.04 ou 15.10.

L'installation est très bien documentée et ne prend que quelques minutes. Vous aurez d'abord besoin d'installer quelques dépendances :

```
$ sudo apt-get install clang libicu-dev
```

Puis, une fois téléchargé le snapshot de votre choix sur swift.org/download/#linux, vous pouvez le décompresser et le poser où vous le souhaitez. Ajouter le répertoire usr/bin situé dans le répertoire du snapshot à votre PATH :

```
$ export PATH=/path/to/usr/bin:"${PATH}"
```

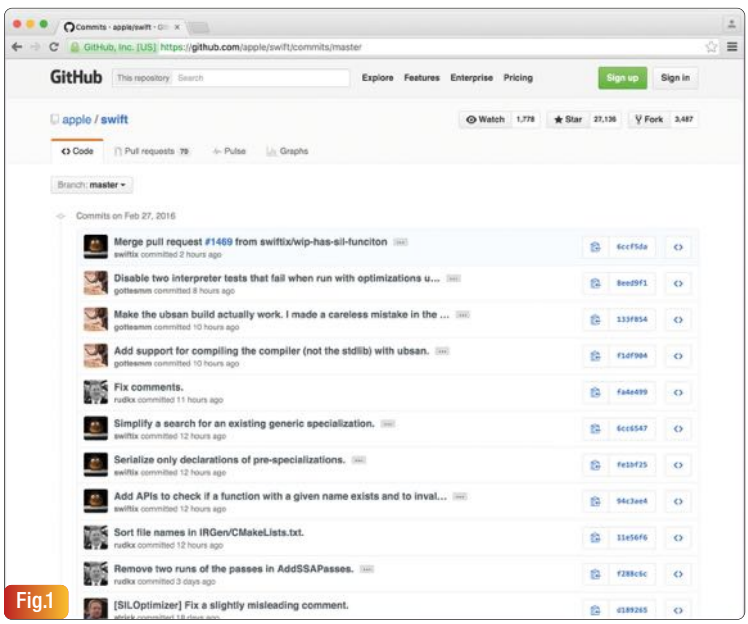


Fig.1

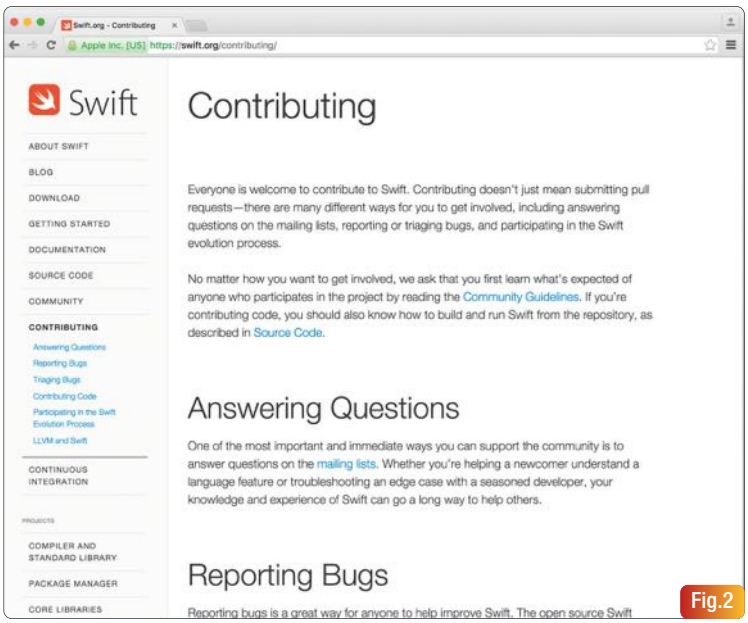


Fig.2

Vous pouvez vérifier que Swift est bien présent et que vous pointez vers la bonne version avec la commande swift -version.
Vous pourrez ensuite lancer la commande swift sans argument, ce qui permet d'accéder à un REPL (read-eval-print loop) pour taper, directement dans le terminal, quelques lignes de Swift comme par exemple :

```
> print("Hello !")
> 39 + 3
> let name = "Swift"
```

Pour sortir, tapez Ctrl+D, ou bien :q comme dans VIM Fig.3.
En parlant de VIM, Apple propose sur le repository Swift (répertoire utils) un plugin VIM pour Swift, avec notamment la colorisation syntaxique. Il existe également des modes pour Emacs, produits par des tierces parties. Vous pouvez ensuite créer votre premier fichier Swift, appelons-le par exemple hello.swift, et écrivons-y ces quelques lignes :

```
for i in 1...10 {
   print(i)
}
```

Ensuite, une commande vous permet d'exécuter ce fichier, vous devriez ensuite voir un compte de 1 à 10 :

```
$ swift hello.swift
```

Pour changer ce fichier en script exécutable, vous pouvez rajouter le shebang en en-tête de votre fichier :

```
#!/usr/bin/env swift
```

Puis avec un appel à chmod +x hello.swift, vous autorisez l'exécution de votre script :

```
$ ./hello.swift
```

Pour compiler un exécutable, vous pouvez respecter la convention de Swift Package Manager. Créez un répertoire du nom de votre choix (disons HelloSwift) puis à l'intérieur un nouveau répertoire appelé sources. Dans ce répertoire sources copions notre fichier hello.swift mais sous le nom main.swift.
Ensuite, dans notre répertoire HelloSwift, ajoutons un fichier Package.swift (attention à la casse), dans lequel on trouvera ce contenu :

```
import PackageDescription

let package = Package(
   name: "HelloSwift"
)
```

Si vous avez utilisé des systèmes de package management comme NPM, vous devez reconnaître ce genre de configuration. Swift intègre Swift Package Manager, qui permet de compiler, partager et gérer les dépendances. Le nom est le seul paramètre essentiel pour notre package. La commande de build va automatiquement compiler les fichiers du répertoire sources et utiliser main.swift comme exécutable par défaut. Pour compiler votre package, utilisez la commande suivante :

```
$ swift build
```

Et vous trouverez à l'endroit indiqué un exécutable en lignes de commandes : .build/debug/HelloSwift
Voilà pour le moment l'étendue de ce que vous pouvez faire avec Swift sur Linux. En effet, le projet n'inclut pas les frameworks Cocoa et CocoaTouch, développés en Objective-C, qui permettent de réaliser des applications sur les plateformes d'Apple. Si vous souhaitez faire une application iPhone, même en Swift, vous aurez toujours besoin d'un Mac avec Xcode. Toutefois, si le projet continue d'attirer autant d'attention et d'excitation, il est probable que de nouveaux frameworks émergent. Il existe déjà une poignée de frameworks de développement Web (Perfect, Taylor, Zewo) mais tout est encore très jeune.
Notons qu'IBM s'investit beaucoup dans ce langage et propose également son framework Web : Kitura.

Pour les autres plateformes

Si vous êtes sur Mac, la procédure d'installation pour Linux marche également, la solution la plus simple restant toutefois d'installer Xcode et de créer un Playground. Il s'agit d'un fichier de code interactif, où les instructions sont évaluées dans une colonne de droite au fil de votre écriture, et les changements répartis automatiquement Fig.4.
Pour Windows, le support n'est toujours pas disponible, ni même prévu officiellement, bien que cela ait des chances d'arriver prochainement. En attendant, la solution la plus simple sera sûrement l'utilisation d'une VM Linux, en attendant l'arrivée de Ubuntu et Bash dans Windows 10.
Enfin, sachez que vous pouvez écrire et exécuter du Swift directement dans votre navigateur grâce à une sandbox proposée par IBM sur Bluemix. Vous y trouverez notamment plusieurs exemples de code plutôt intéressants Fig.5. La société a également donné accès à une image

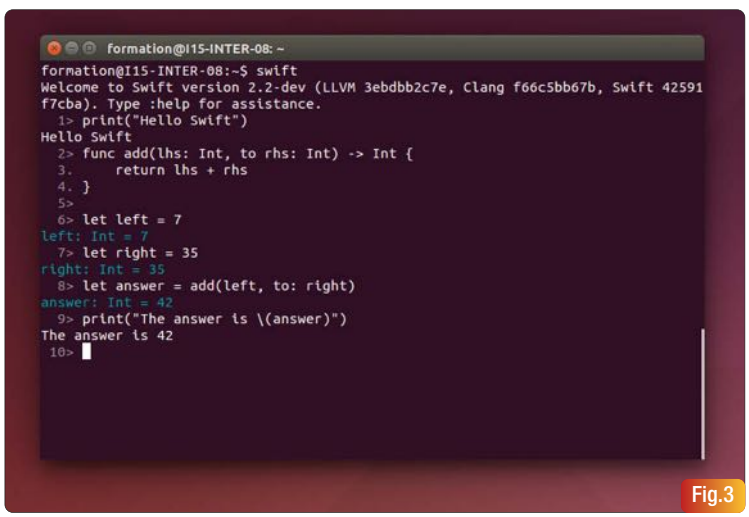


Fig.3

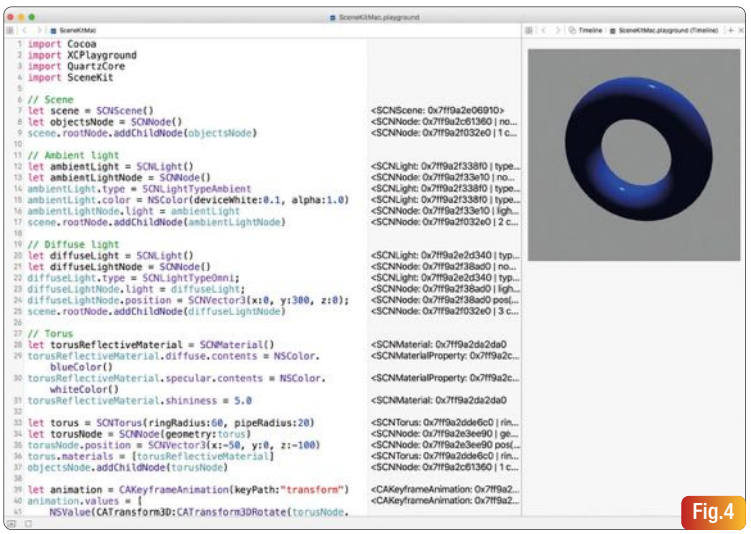


Fig.4

Docker pour exécuter du Swift : github.com/IBM-MIL/Samples
Ça peut être utile si vous n'avez pas d'environnement de développement sous la main, ou si vous êtes sur Windows et que vous ne souhaitez pas attendre que le langage soit porté sur votre OS.

L'avenir

Avec le concours de la communauté, le langage continue d'évoluer à grande vitesse. La version 2.2, sortie au début du printemps, intègre principalement des changements au niveau de la syntaxe ainsi que de nombreux correctifs. Fig.6. La version 3.0 a entamé mi-mai son cycle de developer previews et avance par cycles de 4 à 6 semaines jusqu'à sa release qu'on espère à horizon fin d'année.
Elle intègrera plusieurs changements majeurs. En dehors de la suppression de quelques features très proches du C (notamment les opérateurs ++, – et la boucle for avec compteur qui va avec), c'est surtout la réécriture d'une grande partie des API pour assurer le respect de nouvelles Guidelines de nommage qui va changer beaucoup de choses.
L'Apple d'antan, dans sa Tour d'Ivoire semble bien loin, lorsque l'on voit le bouillonnement participatif qui pousse ce projet. Swift a bénéficié très vite d'une popularité hors-norme et avec une communau- té aussi impliquée, il pourrait bien devenir un langage incontournable dans les années qui viennent.

Apprendre Swift

Maintenant que l'on a vu les principes et l'installation du langage, je vous propose de voir une par une les différentes fonctionnalités du langage. À l'issue de ce dossier, vous devriez pouvoir écrire des programmes simples en Swift, et je vous donnerai les pistes à suivre pour continuer votre apprentissage.
Les blocs de code qui suivent fonctionnent en Swift 2.2. Pour suivre et expérimenter, vous pouvez les recopier dans un Playground sous Xcode (7.3 minimum), dans le REPL dans le terminal ou dans la Sandbox IBM.
Notez que pour faire apparaître les sorties de la fonction print() dans un Playground, vous devez faire apparaître le panneau de debug : menu **View > Debug Area > Show Debug Area** (⇧+⌘+Y)

Hello World

```
print("Hello World")
```

Ce grand classique permet de faire plusieurs remarques :

- Pas de point-virgule en fin de ligne : il permet juste de séparer plusieurs instructions sur la même ligne ; du code peut être exécuté dans le **scope global** de Swift, nous n'avons créé aucune structure de données ;
- print est une fonction du scope global, son paramètre est une chaîne de caractères, on retrouve les parenthèses et les guillemets familières aux langages de la C-Family.

Manipulation de variables

On a déjà vu les mots-clés var et let et le principe de Type Inference dans les exemples de code. La Standard Lib propose la plupart des types attendus : Bool, Int, Float, Double et String.
Le système de Type Inference fonctionne naturellement avec l'assignation d'une valeur littérale, mais vous pouvez indiquer explicitement un type à votre variable lors de la déclaration pour le contourner.
Par exemple let height = 100 donnera une variable de type Int alors que let height:Float = 100 vous permet de signaler que vous voulez ce nombre en virgule flottante. C'est dans ce cas, où lorsque vous déclarez une variable sans lui assigner de valeur, que vous avez besoin d'expliciter le type. La conversion de type n'est pas automatique en Swift. Pour passer une valeur d'un type à l'autre, on utilise un constructeur :

```
let age = 29
let ageAsString = String(age)
```

On trouve les structures algorithmiques les plus courantes, l'utilisation de parenthèses pour séparer le mot-clé de l'instruction est facultative.

```
let width = 10
let height = 14

if width == height {
  print("Hey I'm a square !")
}
else {
  print("I'm just a rectangle…")
}

var remainingPeople = 4

while remainingPeople > 0 {
  print("Hello there!")
```

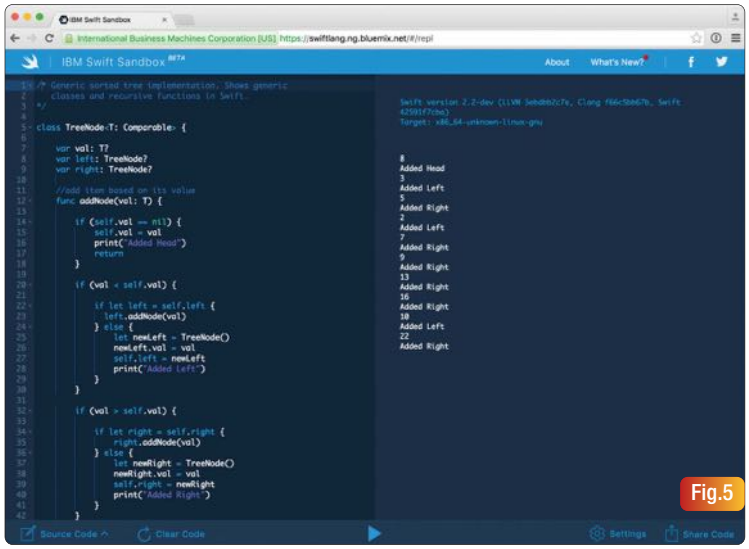


Fig.5

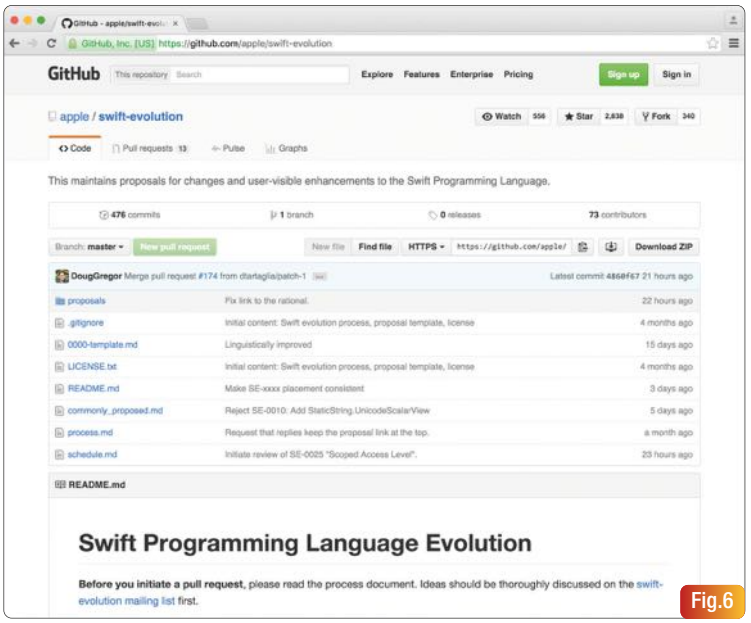


Fig.6

```
    remainingPeople -= 1
}
```

À noter que pour mettre la condition du while en fin de boucle on utilisera repeat :

```
repeat {
    print("Hello there!")
    remainingPeople -= 1
} while remainingPeople > 0
```

For permet l'itération, ici dans un tableau de chaînes :

```
let friends = ["Alice", "Bob", "Charlie"]

for friend in friends {
    print("Hello " + friend)
}
```

Le type String est automatiquement choisi pour friend.
On remarque que la concaténation de chaînes se passe sans problème. On peut également référencer une valeur dans une chaîne : print("Hello \(friend)"). Si for peut aujourd'hui être utilisé dans sa forme "C" for var i=0 ; i<10 ; i++ cette pratique est dépréciée et disparaîtra en version 3.0 (ainsi que l'opérateur ++). À la place, vous pouvez utiliser un Range avec les opérateurs ... (inclusif) et ..< (exclusif). Ce code compte de 0 à 9 :

```
for index in 0 ..< 10 {
    print("i")
}
```

Notre rectangle de l'exemple précédent peut également être représenté sous forme de tuple, ici avec deux valeurs :

```
let rect = (10,14)
print ("Rect width: \(rect.0)")
print ("Rect height: \(rect.1)")
```

Pour plus de clarté, on peut aussi nommer les différents composants d'un tuple :

```
let rect = (height: 10,width: 14)
print ("Rect width: \(rect.height)")
print ("Rect height: \(rect.width)")
```

Swift propose également la clause switch, dans laquelle on peut mettre des conditions assez complexes :

```
let rect = (height: 10,width: 14)

switch rect {
case (let height, let width) where height == width:
    print ("Hey I'm a square !")
case (let height, _) where height > 100:
    print ("I am very tall!")
case (let height, let width) where height <= 0 || width <= 0:
    print ("I'm an impossible rectangle !")
default:
    print ("Nothing special")
}
```

Le fallthrough n'est pas implicite. Le switch nous permet ici de déclarer des variables extraites du Tuple et d'exécuter des tests dessus avec where, cette fonctionnalité s'appelle du Pattern Matching. Notez l'utilisation du underscore _ pour explicitement ignorer une variable comme en Haskell par exemple.

Collections

Pour manipuler des collections d'objet, Swift dispose de 3 types : Array<Element>, Dictionary<Key : Hashable, Value>, Set<Element : Hashable>
Ces types sont génériques et vous permettent donc de spécifier des types associés, avec certaines contraintes comme la clé du dictionnaire et la valeur d'un set qui doivent être Hashable (pour assurer leur unicité). En respectant ces contraintes, vous pouvez réaliser un Array, un Dictionary ou un Set de n'importe quel type. Une syntaxe simplifiée vous permet facilement de déclarer des tableaux et dictionnaires :

```
// Arrays

let members = ["John", "Paul", "George", "Ringo"]
var emptyIntArray = [Int]()

// Dictionaries

var scores = ["Alice": 8, "Bob": 6, "Charlie": 9, "Donna": 12]
var emptyDict = [String : String]()
```

Le REPL attribue le type [String] (équivalent à Array<String>) à la constante members et [String : Int] (équivalent à Dictionary<String, Int>) à la variable scores. Ces types peuvent être utilisés dans une boucle for…in comme on l'a vu précédemment car ils se conforment au protocole (autrement dit : implémentent l'interface) SequenceType. Dans le cas du Dictionnaire, l'itération se fait sur un Tuple, par exemple avec notre variable scores :

```
for (name, score) in scores {
    print("\(name) has \(score) points")
}
```

De la même manière, si vous voulez itérer sur les éléments d'un tableau en ayant leur index vous pouvez utiliser la méthode enumerate() :

```
for (index, name) in members.enumerate() {
    print("Member #\(index) : \(name)")
}
```

On a dans le chapitre précédent utilisé la même boucle sur un Range<Int>, un type qui se conforme lui aussi à SequenceType. Je continue d'employer le terme « Type » pour parler des différentes collections alors que ce sont des classes dans la plupart des langages objet. En Swift, ces types sont des structures !

Structures de données

Voyons la déclaration et l'utilisation d'une structure simple en Swift :

```
struct Rectangle {

    // Propriétés avec valeurs par défaut
    var width: Int = 0
    var height: Int = 0

    // Propriété calculée
    var area: Int {
        return self.height * self.width
    }

    // Méthode d'instance
    func draw() {
        print("Drawing a \(self.width) by \(self.height) rectangle")
    }
```

```
    // Méthode statique
    static func buildRegularSquare() -> Rectangle {
        return Rectangle(width:100, height:100)
    }
}

// Appel du constructeur
let rect = Rectangle(width: 640, height: 480)
rect.area

// Le constructeur peut être appelé sans paramètre car toutes les propriétés ont une valeur par défaut
let zeroRect = Rectangle()
```

Un constructeur a été implicitement écrit pour cette structure, reprenant ses propriétés dans l'ordre dans lequel elles apparaissent. Ici, area est une propriété calculée, on aurait pu en faire une méthode mais sémantiquement il est plus simple d'en faire une propriété.
Le code marcherait aussi bien en remplaçant struct par class, à ceci près qu'il faut déclarer explicitement un constructeur pour la classe, et remplacer le mot-clé static par class pour la méthode statique.

```
class Rectangle {

    var width: Int = 0
    var height: Int = 0

    // Deux constructeurs déclarés, dont un sans paramètre
    init() {}

    init(width: Int, height: Int) {
        self.width = width
        self.height = height
    }

    var area: Int {
        return height * width
    }

    func draw() {
        print("Drawing a \(width) by \(height) rectangle")
    }

    class func buildRegularSquare() -> Rectangle {
        return Rectangle(width:100, height:100)
    }
}

let rect = Rectangle(width: 640, height: 480)
rect.area

let zeroRect = Rectangle()
```

On déclare un constructeur avec init(), on peut également décrire des constructeurs supplémentaires pour une structure.

Notez l'utilisation de self pour référencer l'instance courante, valable dans toutes les structures. Son utilisation est optionnelle quand il n'y a pas d'ambiguïté. On trouve aussi l'enum :

```
enum Land {
    case Forest
    case Island
    case Mountain
    case Plain
    case Swamp
}

var land = Land.Plain
```

Mais un enum en Swift a presque les mêmes capacités qu'une structure, ici par exemple on rajoute une propriété calculée à notre Land :

```
enum Land {
    case Forest
    case Island
    case Mountain
    case Plain
    case Swamp

    var mana: String {
        switch self {
        case .Plain : return "White"
        case .Island: return "Blue"
        case .Forest: return "Green"
        case .Mountain: return "Red"
        case .Swamp: return "Black"
        }
    }
}

var land = Land.Forest
var darkLand:Land = .Swamp
```

Notez que lorsque le système de typage sait quel type il manipule, il n'est pas nécessaire de mentionner explicitement le nom de l'Enum. Ici, comme la variable darkland est explicitement typée, on peut lui attribuer .Swamp, qui est automatiquement résolu comme Land.Swamp par le compilateur. Un enum ne peut pas avoir de propriétés stockées, cependant on peut associer un ou plusieurs types à chaque case, par exemple ici pour la description d'un avatar :

```
enum Avatar {

    enum DefaultAvatarModel {
        case Male
        case Female
        case Other
    }

    case Default(DefaultAvatarModel)
    case Image(url: String)

    func display() {
        switch self {
        case let .Image(url): print("Custom image at URL: \(url)")
        case let .Default(model): print("Default avatar with model: \(model)")
        }
    }
}
```

```
let avatar = Avatar.Default(.Female)
avatar.display()
```

On remarque ici qu'un type peut contenir d'autres types, comme DefaultAvatar ici. Pour le switch on utilise la syntaxe case let pour extraire la valeur associée à chaque cas, une chaîne de caractère pour .Image et une valeur de type DefaultAvatarModel pour .Default. D'une manière générale, l'utilisation des types valeurs (structure et énumération) est très fréquente en Swift. Ces types ont énormément de capacités et il est de surcroît possible de contrôler leur mutabilité. En effet, l'utilisation du mot-clé let pour un type valeur ne garantit pas que sa référence reste immobile mais également tout son état. D'ailleurs, au sein d'une structure, chaque méthode qui modifie son état devra être notée comme mutating. L'immutabilité est une notion qui devient vite confortable lors de l'écriture de code Swift, le compilateur vous encourage d'ailleurs à remplacer vos var en let lorsque vous ne modifiez pas les valeurs. Cette notion encourage l'exploitation de principes de programmation fonctionnelle.

Protocoles et Extensions

En Swift comme en Objective-C, les interfaces sont appelées Protocoles. Tout type peut se conformer à un protocole. On peut par exemple définir un protocole Polygon et changer notre structure Rectangle pour qu'elle s'y conforme :

```
protocol Polygon {
   var numberOfSides: Int { get }
   var area: Int { get }

   func draw()
}

struct Rectangle: Polygon {

   // Propriétés avec valeurs par défaut
   var width: Int = 0
   var height: Int = 0

   var numberOfSides: Int { return 4 }

   // Propriété calculée
   var area: Int {
      return self.height * self.width
   }

   // Méthode d'instance
   func draw() {
      print("Drawing a \(self.width) by \(self.height) rectangle")
   }

   // Méthode statique
   static func buildRegularSquare() -> Rectangle {
      return Rectangle(width:100, height:100)
   }
}
```

Notez qu'en plus de signatures de méthodes, un protocole peut définir des propriétés, soit en lecture seule soit avec un getter et un setter { get set }.
Les protocoles peuvent être utilisés dans la généricité, il serait par exemple possible de définir un tableau de polygones.

```
var polygons = [Polygon]()
```

Il est également possible de définir des Extensions sur tous les types existants, afin de rajouter des méthodes ou des propriétés calculées. Par exemple ici au type Int :

```
extension Int {

   var isOverNineThousand: Bool { return self > 9000 }

   func squared() -> Int { return self * self }

   static var zero: Int { return 0 }

   static func negate(input: Int) -> Int { return -input }
}

Int.negate(9).squared()
9002.isOverNineThousand
Int.zero
```

On peut aussi faire ce que l'on appelle des Protocol Extensions, qui permettent de décupler les capacités de cette fonctionnalité. Cela prend deux formes :

- On peut faire une extension d'un protocole pour définir des comportements par défaut ;
- On peut faire une extension d'un type pour indiquer qu'il se conforme à un protocole.

C'est sur ces capacités que s'appuie le Protocol-Oriented Programming, qui permet d'étendre le comportement des types en s'appuyant davantage sur la composition de protocoles plutôt que sur l'héritage. Dans la Standard Library, par exemple, les fonctions map(), reduce() et filter() sont automatiquement disponibles dès lors que l'on respecte le protocole SequenceType. Avant Swift 2.0, qui a apporté les Protocol Extensions, ces fonctions étaient globales. Cela permet une grande souplesse dans les architectures réalisables sans sacrifier la robustesse apportée par le typage fort.

Optionals

On a vu que l'on utilisait un constructeur pour passer d'un type à l'autre. Mais que se passe-t-il si l'on essaye de transformer une chaîne en entier ?

```
let age = Int("29")
```

Le type de notre constante age est d'un type un peu particulier : Int?. Ce point d'interrogation accolé au type indiqué qu'il s'agit d'un Optional, ce qui veut dire que cette valeur peut être nulle (nil en Swift).
Dans ce cas, le constructeur Int(String) est un « Failable Initializer », un constructeur qui peut échouer (si la chaîne de caractères ne peut pas être changée en entier), auquel cas il renvoie nil.
Si vous souhaitez déclarer une variable dans le REPL sans lui attribuer de valeur en même temps, vous devez la déclarer comme Optional :

```
var answer:String?
var rect:Rectangle?
```

Cela signifie que si un type n'a pas ce point d'interrogation, il ne peut pas avoir la valeur nil, le compilateur s'en assure ! Il ne vous laisse pas manipuler un Optional sans quelques précautions.
Imaginez que votre valeur soit mise dans une sorte de boîte, qui peut contenir soit une valeur du type spécifié, soit nil. Avant de l'utiliser, vous devez ouvrir cette boîte pour voir ce qu'elle contient, et en tirer la valeur le cas échéant. On appelle cette opération « unwrapping » :

```
var optionalInput:Int?
optionalInput = 7

if let input = optionalInput {
   print(input + 5)
}
```

Ma variable optionalInput étant de type Int?, je ne peux pas lui ajouter un Int, l'opérateur + ne le permet pas. Je dois commencer par « déballer » (Unwrap) la valeur qui s'y trouve, s'il y en a une. C'est ma condition if let qui permet ceci : je déclare une variable input qui sera de type Int (donc non optionnelle) et qui prendra la valeur de optionalInput **si elle en a une**. Si optionalInput est nil, le bloc conditionnel n'est pas exécuté. Il est possible de mettre plusieurs déballages dans une seule condition if, pour éviter les imbrications interminables de conditions.
C'est un verrou très appréciable quand on sait que la NullReference Exception est probablement la cause d'erreur la plus souvent rencontrée dans une application. Il y a des raccourcis pour effectuer cette opération de déballage plus rapidement. Le **Optional Chaining** permet d'appeler des fonctions ou des propriétés d'un Optional avec l'opérateur ?. :

```
var rect: Rectangle?

let width = rect?.width // width est de type Int?
                  // il prend la valeur nil si rect est nil

rect?.draw()         // Aucune opération n'est jouée si rect est nil
```

Le **Nil Coalescing** (??) permet de redescendre vers une valeur différente si une valeur est nil :

```
var name: String?
let displayName = name ?? "Sans Nom" // DisplayName est de type String
```

Bien sûr ces opérateurs peuvent être enchaînés et combinés.
Vous pouvez spécifier qu'une valeur est optionnelle lors de la déclaration d'une variable, dans un paramètre ou bien une valeur de retour. Ainsi, le compilateur est capable de surveiller toute la chaîne. Ce verrou peut parfois être de trop et il est possible de dire au compilateur de lâcher la bride, avec un caractère qui doit appeler à la vigilance quand on l'aperçoit : !. On peut l'utiliser en regard d'un type (par exemple Int!) pour signaler qu'il s'agit d'un **Implicitly Unwrapped Optional** : il peut avoir la valeur nil, mais le compilateur ne le vérifiera pas. On peut aussi l'utiliser pour forcer le déballage d'un Optional. Dans ces deux cas, un crash aura lieu à l'exécution si une valeur nil est rencontrée, donc attention.

```
var rect: Rectangle?
let width = rect!.width // Si rect est nil au runtime
                  // un crash aura lieu ici
```

En réalité, Optional est un type générique Optional<Wrapped>, il s'agit d'une implémentation de la monade Maybe que l'on trouve dans certains langages comme Haskell ; si vous êtes déjà familier/ère avec le principe, ça ne devrait pas être un obstacle.

Fonctions

Vous l'avez peut-être compris en utilisant print() dans les exemples précédents : des fonctions peuvent exister dans le scope global. Si vous avez suivi la marche à suivre pour faire un exécutable, vous aurez d'ailleurs compris qu'il n'y a pas de fonction main mais que ce qui s'exécute est la série d'instructions du scope global dans le fichier main.swift. Une fonction se déclare avec le mot-clé func, comme pour les méthodes de notre Rectangle :

```
func sayHello() {
   print("Hello !")
}

sayHello()
```

Les paramètres s'inscrivent naturellement entre les parenthèses de la déclaration :

```
func sayHello(name: String) {
   print("Hello \(name)")
}

sayHello("Programmers")
```

Ces deux fonctions peuvent tout-à-fait cohabiter, il est possible de surcharger une méthode avec des jeux de paramètres différents.
La surprise apparaît lorsque l'on ajoute un second paramètre, au moment de l'appel de la fonction :

```
func sayHello(name: String, iterations: Int) {
   for _ in 1...iterations {
      print("Hello \(name)")
   }
}

sayHello("Programmers", iterations: 3)
```

Remarquez que je dois spécifier le nom du second paramètre. Il faudra de cette manière nommer tous les paramètres à partir du second, sans que cela vous donne pour autant le droit de les donner dans le désordre. En Swift 3.0, prévu pour la fin de l'année, il faudra nommer tous les paramètres y compris le premier. Il est possible de spécifier un nom de paramètre différent pour l'intérieur de la fonction et son appel.

```
func sayHello(to name: String, times iterations: Int) {
   for _ in 1...iterations {
      print("Hello \(name)")
   }
}

sayHello(to: "Programmez", times: 3)
```

Cette notion de paramètres nommés est héritée de l'Objective-C et vise à produire un code lisible, où les appels de fonctions se lisent comme une phrase, pour lever les ambiguïtés. Par exemple, la fonction insert() du type Array :

```
var names = ["Athos", "Porthos", "Aramis"]

names.insert("D'Artagnan", atIndex: 2)
```

Pour certains cas où le nommage explicite ne serait pas utile, on peut annuler cette obligation de nommer des paramètres lors de l'appel en utilisant le marqueur _ :

```
func addition(lhs: Int, _ rhs: Int) {
   print(lhs + rhs)
}

addition(3, 6)
```

Si une fonction retourne une valeur, il faut le spécifier dans sa signature avec le marqueur -> :

```
func divide(dividend: Int, by divisor: Int) -> Int {
    return dividend / divisor
}

let quotient = divide(6, by: 3)
```

Swift permet également l'utilisation des Closures, des fonctions qui peuvent être passées en paramètre ou stockées dans une variable. En réalité, toutes les fonctions que l'on a écrites plus haut sont des Closures nommées, c'est d'ailleurs facile à vérifier :

```
let divideClosure = divide

divideClosure(21, by: 7)
```

Ceci fonctionne sans problème divideClosure est du type (Int, Int) -> Int soit une fonction qui prend 2 entiers en paramètre et renvoie un entier en retour. Une fonction peut également renvoyer une Closure.

```
func isDivisibleBy(divisor: Int) -> (Int) -> Bool {
    return { (dividend: Int) -> Bool in
        return dividend % divisor == 0
    }
}

let isEven = isDivisibleBy(2)

isEven(100)

isEven(37)
```

Ici la fonction isDivisibleBy prend un entier en paramètre (le diviseur) et retourne une Closure qui prend un autre entier en paramètre (le dividende) et renvoie vrai si le reste de la division est zéro. Son type est (Int) -> (Int) -> Bool. Ici, isEven est une variable de type Closure (Int) -> Bool utilisée comme une fonction par la suite. L'utilisation des Closures est très puissante en Swift, c'est ce qui fait que de nombreux concepts de programmation fonctionnelle s'intègrent progressivement dans la culture du langage.

Gestion des erreurs

La gestion des erreurs a été rajoutée à Swift dans sa version 2.0, présentée en juin 2015. L'utilisation du try/catch en Objective-C n'était pas la norme, elle était plutôt réservée à des erreurs impossibles à récupérer. En Swift, une convention est d'utiliser un retour de type Optional pour indiquer qu'une valeur peut ne pas être renvoyée (dans un constructeur, par exemple). Mais parfois, on veut être en mesure de remonter des erreurs plus claires qu'une simple absence de valeur. Une méthode capable de lancer une erreur doit le déclarer dans sa signature avec le mot-clé throws reprenons notre méthode divide() qui a un cas d'erreur évident :

```
func divide(dividend: Int, by divisor: Int) throws -> Int {
    if divisor == 0 {
        throw DivisionError.dividingByZero
    }
    return dividend / divisor
}
```

Ici notre fonction lancera une erreur si le diviseur est 0. Qu'est-ce que DivisionError.dividingByZero ? Il s'agit d'un type (ici un enum) qui se conforme simplement au protocole ErrorType, un protocole vide qui sert juste d'indicateur.

```
enum DivisionError : ErrorType {
    case dividingByZero
}
```

Les types enum se prêtent bien à la remontée d'erreurs comme ils peuvent avoir plusieurs cas spécifiques et porter des valeurs associées. Avant de voir comment on appelle une méthode capable de lancer une erreur, j'aimerais remplacer ma condition par une instruction plus sémantiquement claire :

```
func divide(dividend: Int, by divisor: Int) throws -> Int {
    guard divisor != 0 else { throw DivisionError.dividingByZero }

    return dividend / divisor
}
```

La clause guard fonctionne comme un if inversé (comme unless en Ruby), sa contrainte est que le bloc conditionnel qui le suit doit obligatoirement sortir du scope actuel (return, throw, break, continue). Il permet d'expliciter la prise en charge des cas limites en avance tout en évitant l'imbrication des blocs if que l'on peut trouver dans ce genre de cas.
Maintenant que ma fonction divide() remonte clairement les erreurs, voyons comment l'utiliser dans une autre fonction. La première solution est la plus simple : seulement remonter l'erreur :

```
func printSomeDivisions() throws {
    let quotient = try divide(42, by: 7)
    print(quotient)

    print (try divide(81, by: 9))
}
```

Comme notre méthode printSomeDivisions() est marquée comme pouvant lever des erreurs, elle peut appeler la méthode divide() sans prendre trop de précautions. Notez qu'en Swift, le mot-clé try ne permet pas de commencer un bloc mais s'écrit avant l'appel d'une fonction pouvant échouer – même dans le cas où l'on va remonter l'erreur sans y toucher. Maintenant, mettons que nous ne voulons pas voir notre fonction remonter l'erreur mais la gérer directement :

```
func printSomeDivisions() {
    do {
        let quotient = try divide(42, by: 7)
        print(quotient)
        print (try divide(81, by: 9))
    }
    catch DivisionError.dividingByZero {
        print("Attempted division by zero")
    }
    catch {
        print("Unspecified error while dividing: \(error)")
    }
}
```

C'est donc do qui permet d'initier un bloc avec des instructions risquées, suivi d'un ou plusieurs blocs catch. Le mot-clé catch peut être suivi d'un Pattern pour spécifier les conditions d'interception d'une erreur, comme pour une instruction switch. Comme on ne spécifie nulle-part le type d'erreur qu'une fonction peut renvoyer, il est attendu qu'un catch inconditionnel arrive en toute fin pour gérer les erreurs de tous types. Notez que ce dernier génère automatiquement une constante error contenant l'erreur remontée. Il reste d'autres moyens de gérer les erreurs : try? et try!
Avec try? il est possible de transformer un appel de fonction en Optional.

Par exemple, dans la fonction printSomeDivisions(), je peux changer mon assignation de cette manière :

```
func printSomeDivisions() {

   let quotient = try? divide(42, by: 7)
   print(quotient)
}
```

La constante quotient prend donc le type Int?, elle sera nil si l'appel à divide() provoque une erreur, mais cette erreur ne sera pas remontée plus haut. Bien sûr il est possible d'utiliser try? dans une clause if let ou guard let comme on traiterait un Optional classique.
Comme toujours, l'utilisation du point d'exclamation implique un risque : try! permet de désactiver la propagation d'une erreur. Si une erreur est remontée sur un tel appel, c'est une interruption au Runtime qui aura lieu. On associe souvent la clause finally au système try/catch, afin de déclarer des opérations à effectuer quelle que soit l'issue d'une sortie. Si finally n'existe pas en Swift, le langage propose le bloc defer.

```
func printSomeDivisions() {

   defer { print("… And that's it!") }

   do {
      let quotient = try divide(42, by: 7)
      print(quotient)
      print (try divide(81, by: 9))
   }
   catch DivisionError.dividingByZero {
      print("Attempted division by zero")
   }
   catch {
      print("Unspecified error while dividing: \(error)")
   }
}
```

L'instruction defer permet de programmer une instruction qui s'exécutera au moment de la sortie du scope dans lequel il est déclaré, que ce soit en retour ou sortie de fonction, en fin d'une itération boucle ou d'un if. Ici, que la fonction se termine normalement ou qu'elle rencontre une erreur, le message … And that's it ! sera affiché à la fin.

Pour aller plus loin

Et voici un avant-goût conséquent du langage Swift !
Maintenant que vous connaissez le langage et savez comment écrire des scripts ou compiler des exécutables, il ne vous reste plus qu'à expérimenter !. Pour vous entraîner, les plateformes en ligne **Codingame** et **Exercism** proposent de résoudre leurs nombreux exercices en Swift. Pour en apprendre davantage sur le langage, le livre d'Apple « **The Swift Programming Language** » est une bonne référence. Il est disponible gratuitement sur iBooks si vous avez un appareil Apple, ou en ePub sur le site **Swift.org**. Pour une recherche rapide dans la documentation, le site **SwiftDoc.org** est aussi une référence, avec la possibilité de choisir quelle version de Swift vous ciblez.

Suivre l'actualité

Swift est un langage très dense et les propositions d'évolution sont revues et acceptées chaque semaine. Suivre les listes de diffusion et les dépôts GitHub en direct peut vite s'avérer chronophage. Le site **Swift Weekly Brief** propose une newsletter hebdomadaire revenant sur les évolutions du dépôt principal, les propositions soumises, acceptées et rejetées, et même des tâches faciles à prendre en main pour commencer à participer au projet. Quelques développeurs sont très actifs sur Internet et partagent leur découverte. Je vous conseille de suivre **Erica Sadun** qui fait partie des principaux participants au projets, et dont le blog a un flux soutenu et très intéressant. Pour le reste, il est encore difficile de séparer l'actualité du langage de celle de l'écosystème du développement iOS. Si ça ne vous dérange pas, vous pouvez obtenir davantage de matériel en vous inscrivant aux newsletters **This Week in Swift** et **iOS Dev Weekly**. Du 13 au 17 juin aura lieu la WWDC, la conférence développeur Apple annuelle, où devraient avoir lieu des annonces et des talks concernant le langage et son évolution. Toutes les sessions seront disponibles gratuitement sur le site d'Apple à mesure qu'elles auront lieu.

Accompagner l'évolution

Il n'y a pas de grands secrets sur l'évolution de Swift. Tout a lieu dans la plus grande transparence, et vous pouvez déjà voir sur le dépôt **Swift-Evolution** quels sont les changements attendus dans la version 3.0 prévue pour la fin de l'année. Cette version marquera notamment une véritable consolidation de la syntaxe, autour des conventions qui se sont créées à mesure que le langage est utilisé. Elle comprendra de nombreux « breaking changes » qui seront le prix à payer pour une plus grande stabilité par la suite. Le mot d'ordre est le suivant : s'il y a des choses à casser, mieux vaut le faire pour la version 3.0 plutôt qu'après. Apple accompagne chaque nouvelle version du langage avec un outil de migration intégré à Xcode pour limiter les dégâts. Cette version devrait également comprendre le portage Linux de **Foundation**, l'extension de la Standard Library issue de Cocoa qui contient certaines mécaniques plus complexe comme la manipulation des dates (NSDate, NSCalendar), le parsing (NSJSON, NSXMLDocument) et les accès réseau (NSURLSession). Des évolutions plus techniques sont également au menu, comme une amélioration de la généricité, avec par exemple la possibilité de créer des typealias génériques : définir un type StringDictionary<T> qui soit un Dictionary<String, T> ; ou un contrôle d'accès plus fin (le mot-clé private permet aujourd'hui de limiter la visibilité au sein d'un fichier et non du scope d'un type.) On y verra aussi la disparition des opérateurs ++ et -- ainsi que de la boucle for avec compteur qu'on a l'habitude d'utiliser en C. Espérons que les versions suivantes verront l'arrivée de fonctionnalités très attendues comme la gestion de la concurrence, un équivalent aux Promise ou à async/await serait tout-à-fait bienvenu. En attendant, le portage de Grand Central Dispatch sur Linux est en marche. On espère également une évolution d'Xcode pour supporter les projets basés sur Swift Package Manager, à qui il ne manque que ça pour standardiser vraiment la gestion des dépendances en Swift car aujourd'hui on utilise surtout des outils tiers comme CocoaPods ou Carthage pour les applications iOS et OS X. Swift est encore un langage jeune et même si son style va vraisemblablement se cristalliser avec cette version 3.0, il est toujours possible de participer à son évolution et à sa correction. Les différentes listes de diffusion sont publiques et ouvertes à tous, il vous est possible de vous joindre aux conversations et de donner votre avis afin d'aider à façonner un langage qui vous plaise. De même, il est possible de corriger un bug sur le Jira du langage (bugs.swift.org) et de proposer votre Pull Request et ainsi participer activement à sa consolidation. La communauté est bouillonnante et on trouve déjà des initiatives très sérieuses comme le portage des Reactive Extensions avec **RxSwift**, un investissement très fort d'IBM avec son framework Web **Kitura** mais aussi son **Swift Package Catalog**. En somme, Swift est un langage solide et moderne qui a, grâce à sa communauté hyperactive, toutes ses chances d'atteindre son objectif : devenir un langage de référence, polyvalent et omniprésent. ⊞

DevOps et le développeur

saison 3

© StockFinland

Comme vous le savez sans doute, le DevOps est une de nos approches fétiches. En réalité, nos préoccupations remontent à la notion d'ALM (Application Lifecycle Management). Le DevOps est en quelque sorte un héritier de l'ALM mais en plus souple et en plus complet, et en s'inscrivant dans les démarches agiles. Le DevOps réconcilie les équipes techniques et les équipes de production, tout en rapprochant les personnes du métier et du business de l'univers IT et donc des développeurs et vice-et-versa.

Rappelons tout de même un élément essentiel : DevOps est une démarche, une philosophie pour les développeurs et la production (déploiement, exécution, infrastructure). Il s'agit de (ré)concilier des mondes qui ont du mal à se parler et encore plus à se comprendre. Ils utilisent des outils et des méthodes différents, ne facilitant pas les échanges.

DevOps va des besoins (qui définissent une application, un projet) à la mise en production. Entre les deux, il y a différents acteurs ayant des objectifs parfois très différents. DevOps propose une approche globale à 360° pour intégrer tous les acteurs, les processus liés à chaque étape du projet et les outils nécessaires.

Vous l'aurez compris, DevOps n'est ni une méthode, ni un outil. En revanche, vous allez utiliser des méthodes agiles dans la démarche DevOps et intégrer des outils s'inscrivant dans cette approche : intégration continue, gestion de code, planification, IDE, etc.

Les conteneurs deviennent un composant central des environnements de tests et de déploiement des applications et des équipes développeurs et opérationnelles. La sécurité est bien entendue une thématique importante mais parfois mal comprise en contexte DevOps. Mais peu à peu, le terme DevOps est détourné de sa signification originale. Etant à la mode, vous risquez de le voir un peu partout. Et d'entendre, « votre outil est-il DevOps ? »...

François Tonic

Dans la famille DevOps, je voudrais le développeur !

Le DevOps représente un changement culturel et organisationnel, qui doit logiquement s'accompagner de bonnes pratiques et d'outils adaptés. Nous regarderons pourquoi et comment le DevOps impacte le métier de développeur et ses domaines de responsabilité.

Mos Amokhtari
Directeur Technique CA Technologies
Mos Amokhtari occupe la fonction de Directeur Technique chez CA Technologies France depuis avril 2014.

Olivier Laplace Senior
Principal Consultant CA Technologies
Olivier participe auprès de ses clients à la nécessaire mutation technologique des applications. Spécialiste des architectures distribuées composites, il porte ici un regard particulier sur la manière dont le DevOps change le métier de développeur.

Fig.1

Modèle Agile de livraison incrémentale

DevOps : impact sur le développeur et son métier

Les démarches « Agile »(Fig.1), par l'usage d'itérations courtes (en comparaison avec les approches classiques en « V » qui reposent sur des cycles plus longs), visent à coder et livrer plus fréquemment de la valeur métier dans les versions successives.

L'approche Lean, dans l'IT, consiste schématiquement à l'amélioration continue du système d'information, de manière itérative, par l'apprentissage avec des aller-retours fréquents entre les différentes équipes comme par exemple Développements et Tests.

L'Agile et l'approche Lean sont des démarches qui accompagnent le DevOps, et au-delà, la volonté de rapprocher Métier et IT dans un contexte économique fort de risque de « désintermédiation ». Les exemples connus de tous, sont des acteurs comme Uber ou Airbnb qui se substituent aux acteurs historiques et assurent à leur place la « médiation » entre l'offre et le consommateur final.

Par delà les approches et démarches, c'est bien une vision nouvelle du métier de développeur qui émerge.

Considérons un instant l'implémentation DevOps telle que vécue par les startups : quand une poignée d'individus motivés doit adresser l'ensemble des besoins de l'entreprise naissante, le développement applicatif vu par le prisme des couches, avec des spécialistes par domaines (développements Mobile / Client / Serveur ; modélisation relationnelle / NoSQL / Scripts d'Intégration continue ; Tests ; scripts de déploiement ...) n'est pas envisageable. Les bases non structurées vs relationnelles (une seule base capable d'adresser des données de natures différentes), NodeJS vs Java (un seul langage pour les développements Serveur et Client), le Web responsive vs Site Web / Tablette / Mobile (un seul développement capable d'adresser des supports différents), font partie des évolutions qui participent au changement du métier de développeur : être capable de traiter l'ensemble de la stack logicielle ... et même plus.

On retrouve aussi cette perspective avec l'approche micro-services : des composants auto-suffisants, développés, testés et même parfois opérés et maintenus par les développeurs.

DevOps : de la théorie à la pratique

Les contraintes liées à la livraison continue résident dans deux types de Silos :

- *Un Silo humain* : différents départements, avec des domaines de responsabilité propres, afin de couvrir l'usine logicielle depuis le développement jusqu'aux opérations ;
- *Un Silo technologique* : différentes solutions (internes, open-source, éditeurs) pour répondre aux besoins spécifiques : code, revues de code, gestion de configuration des développements (versions), intégration continue, déploiement, tests unitaires, tests X à X, d'intégration, fonctionnels, de performances ...

Le concept de DevOps vise à adresser et répondre à ces contraintes.

Pour les développeurs, cela ne passe plus simplement par la compilation de son code, mais aussi par la capacité à déployer, à tester plus tôt et plus fréquemment, fonctionnellement et en performance. C'est ce à quoi doit se préparer le développeur Full-Stack. La livraison continue d'une nouvelle version d'une application représente des développements jusqu'à la mise en production; un DevOps que l'on peut qualifier d'horizontal. Les startups et les entreprises qui ont initié une transformation digitale ambitieuse implémentent un DevOps davantage vertical, dans lequel le développeur joue un rôle essentiel.

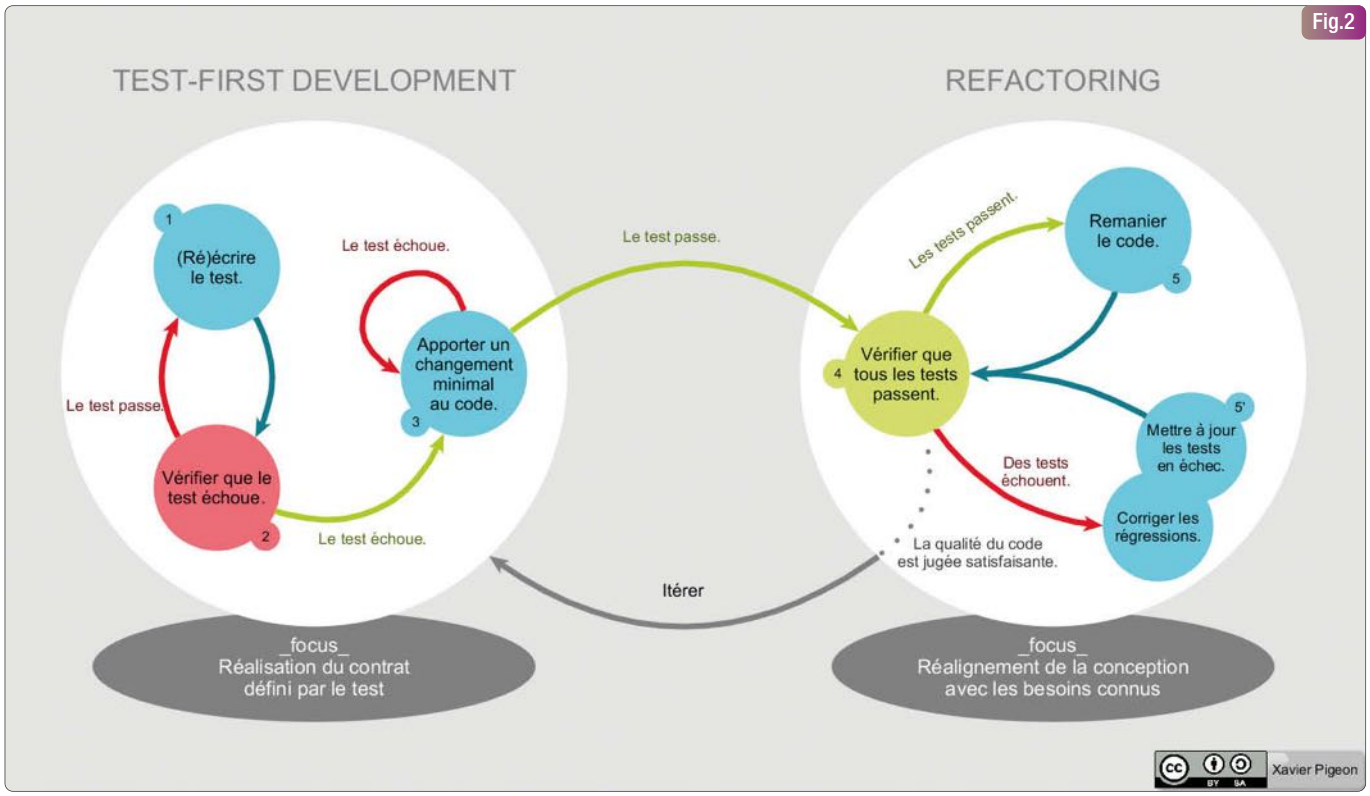
Pour une fonctionnalité donnée (user story), il doit coder, compiler mais aussi déployer et tester en simulant les systèmes connexes non accessibles (par exemple un service encore non développé ou un système qu'il est difficile de rendre accessible).

Le « Lean Startup » avec son cercle vertueux « développer / déployer / tester / apprendre ... et recommencer », participe à la mise en œuvre de cycles courts et à une démarche Agile.

Ce DevOps vertical préfigure le rôle et les responsabilités du développeur Full-Stack, pour les startups comme pour les entreprises plus traditionnelles soucieuses d'améliorer la qualité ou le « Time to Business » (valeur métier). S'y préparer en qualité de développeur n'est donc plus une option mais une nécessité. La bonne nouvelle réside dans le fait que de bonnes pratiques et des outils peuvent vous y aider :

- *Une approche orientée « test » des développements* : puisque le développeur va davantage devoir assumer sa part dans la qualité de l'application, en testant plus, mieux et plus tôt, pourquoi ne pas commencer par cet aspect ? C'est tout l'objet du « test-driven development » (TDD) ou Développement piloté par les tests (Fig.2) à commencer par définir et écrire les tests avant de commencer à coder. Il conviendra de définir avec pertinence et de manière non ambiguë les tests à exécuter ; l'objectif est de cibler un minimum de cas de tests à

Fig.2



Représentation graphique du TDD (tiré de wikipédia auteur Xarawn)

exécuter tout en garantissant une couverture maximale des cas d'usage.

- *Des développements et des tests sans contraintes* : les développements ne doivent pas être en situation d'attente parce qu'un système ou un service n'est pas disponible. Disposant déjà d'un périmètre conséquent à adresser, les développeurs Full-Stack ne peuvent et ne doivent pas passer de temps à développer des bouchons. Leur véritable valeur réside dans leur capacité à « coder » dans l'application des fonctionnalités, promesses de la valeur métier attendue.
- *Une expérience utilisateur à la hauteur des exigences* : s'assurer que l'expérience des utilisateurs finaux sera à la hauteur des attentes passe par une compréhension et une analyse du comportement technique de l'application.
- *Une gestion maîtrisée et sécurisée de ses API* : dans des architectures multi-supports où les API portent la valeur métier, il convient d'en assurer un accès maîtrisé et sécurisé.
- *La livraison continue, colonne vertébrale de la qualité et de la valeur métier* : de façon déclenchée ou automatisée, des processus de déploiement garantissent la fiabilité de livraison et de mise à disposition des applications. Ils peuvent, dans certains contextes, être pilotés directement par les développeurs afin de pousser des modifications jusqu'en production.

DevOps : les outils disponibles

Si les pratiques nous indiquent le chemin à suivre, les outils sont garants de la standardisation des procédés, car il ne peut exister d'automatisation (vecteur du DevOps) sans standardisation.

Citons quelques exemples de solutions à même de supporter les bonnes pratiques évoquées :

- *Une approche orientée « test » des développements* : dans une approche Model Based Testing, une solution de conception d'exigences Agile couplée à un gestionnaire de données de test pour la mise à disposition des données artificielles nécessaires aux tests supporte efficacement la démarche TDD.
- *Des développements et des tests sans contraintes* : la virtualisation de services permet de s'affranchir des contraintes d'indisponibilités et de simuler simplement et rapidement le fonctionnel, les données et les performances de tout flux au-dessus de TCP.
- *Une expérience utilisateur à la hauteur des exigences* : une solution de gestion des performances applicatives apporte une visibilité et une compréhension technologique des usages et des performances de l'application ; une porte ouverte vers une expérience utilisateur de qualité.
- *Une gestion maîtrisée et sécurisée de ses API* : une solution de gestion des API assure à la fois l'on-boarding des consommateurs, la médiation et la sécurisation d'accès des API.
- *La livraison continue, colonne vertébrale de la qualité et de la valeur métier* : une solution d'automatisation des versions assure un déploiement automatisé de bout en bout des applications, sans écriture de scripts, avec gestion des dépendances inter et intra-solutions.

Il faut garder à l'esprit que le DevOps n'est que le chemin et pas la finalité. En fonction des objectifs (valeur métier, qualité, expérience utilisateur ...), le développeur adaptera sa démarche DevOps en conséquence : réduire les contraintes et dépendances aux services ou systèmes non accessibles, déploiements automatisés sur l'ensemble des environnements jusqu'à la production, automatisation des tests de non régression, amélioration de la couverture de tests fonctionnels. L'objectif ultime restant bien sûr de traduire une idée en service à valeur ajoutée le plus rapidement possible et avec une qualité optimale pour donner à son entreprise un avantage concurrentiel, le fameux « concept to cash » de nos amis outre-Atlantique. .

Pour les développeurs habités de curiosité et prêts au changement, c'est avec le DevOps un nouvel univers des possibles qui s'ouvre, riche d'une diversité de technologies et d'un domaine de responsabilités élargi. Le respect des pratiques appropriées et l'usage de solutions de qualité seront les meilleurs compagnons de route des développeurs.

DevSecOps ou DevOpsSec : choisir ou pas…

Nous avions le DevOps. Terme déjà assez barbare, mais on parle aussi de DevSecOps. Le Sec signifie Sécurité. Bref, du DevOps mais à la sauce sécurité informatique. Vous me direz : oui normal de faire de la sécurité, c'est la base. Mais est-ce aussi évident que cela ? On parle beaucoup de sécurité, de sécurité du code, de corriger le plus tôt possible les failles et d'avoir une programmation la plus sûre possible. En y regardant de plus près, le DevSecOps est une autre manière de voir le DevOps et donc son informatique.

François Tonic
Programmez!

Le DevSecOps est relativement peu connu en France et particulièrement des développeurs et des entreprises. Ce « mouvement » fait parler de lui. Il y a des présentations et sessions techniques durant la conférence RSA 2015 et plus récemment le sujet a été abordé à Devoxx 2016 (DevOps vu par un hacker).

DevSecOps n'est pas DevOps + Sécurité ?

Certains diront oui, d'autres non. Nous pourrions aussi le voir comme un mélange de SecOps et de DevOps. Le SecOps est la sécurité côté production, exécution, infrastructure. Tout dépend du point de vue que l'on prendra. Le DevSecOps rassemble la sécurité côté développeur/développement/projet et la sécurité côté opérationnel (déploiement, exécution, production, infrastructure). Il s'agit d'intégrer la sécurité de bout en bout. Le DevOps peut être un bon vecteur pour favoriser et améliorer la sécurité des applications et des environnements de production. Et les règles et méthodes s'appliquent à tout le monde durant tout le cycle. Mais au lieu de parler DevSecOps, faut-il parler de DevOpsSec quand nous sommes en contexte purement DevOps ? DevOpsSec ressemble plus à de la sécurité que l'on rajoute dans sa démarche DevOps. La question ici n'est pas de savoir qui a raison ou qui a tort, mais comment, nous allons réellement utiliser et rajouter la sécurité dans une démarche DevOps pour aboutir à des projets, des codes mieux sécurisés, avec une surface d'attaque moindre et donc moins de failles.

Intrinsèquement, les objectifs de chaque équipe sont difficiles :

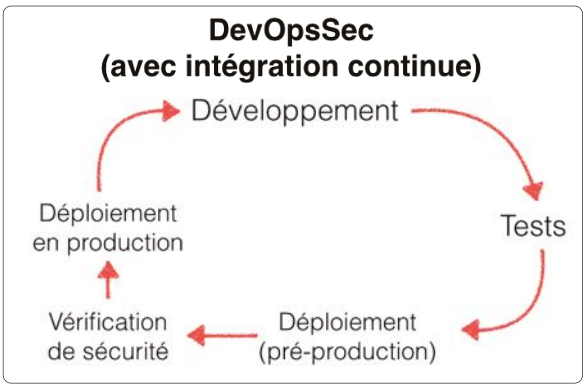
- Devs : on code tout ;
- Ops : ne cassez pas tout ;
- Sec : « tout casser » (ou l'inverse).

La sécurité a souvent été sacrifiée dans les développements pour gagner du temps et livrer plus rapidement. Et la pression sur les délais, livrer pour hier le projet, sont toujours d'actualité. Bref, il faut laisser aller et faire rapidement les changements. Mais cela pose de réels soucis de sécurité, de stabilité et de failles/bugs. Si le DevOps est bien compris et mis en place, vous allez gagner en stabilité du code, avec moins de bugs. L'intégration continue aide à cet objectif. Et l'introduction de batteries de tests systématiques permet de supprimer les bugs courants. L'usage de bonnes pratiques et de règles de programmation permet un code lisible et qui doit normalement être plus stable (nous ne parlons pas de performances ici).

La notion de sécurité doit être introduite très tôt comme les tests doivent l'être dans le cercle vertueux du DevOps : développement, tests, déploiement. En DevOpsSec, nous pourrions avoir : développement, tests, pré déploiement, vérification de sécurité, déploiement en production. Comme pour les bugs, plus tôt une faille de sécurité est détectée en conception/développement, plus il sera (théoriquement) facile de la corriger(1). Vous pouvez, et vous devez, utiliser les bonnes pratiques de sécurité et de la programmation sécurisée dès la conception. Vous avez de bonnes chances de réduire les failles les plus courantes (saisie de champs, appels illégaux, chemins en clair, etc.). Les tests joueront un rôle crucial.

> *« Ops who think like devs*
> *Devs who think like ops »*

Avec un usage du DevOps et de ses principes, la sécurité doit être une des préoccupations au quotidien. Car les failles se multiplient et touchent tous les nouveaux usages : objets connectés, voitures, domotique, infrastructure stratégique, etc. Il ne faut pas croire que le DevOps, en soi, est complet, couvre tout. Le risque est aussi d'isoler les équipes concernées, des autres équipes, donc de recréer des silos.



Pour le développeur, ce serait de trop faire confiance aux outils et aux méthodes.

DevOpsSec impose la sécurité

Dans cette approche, la sécurité devient le 4e pilier du DevOps :

- Business/métier : expression des besoins, des fonctions nécessaires, des applications pour l'entreprise, un client, bref pour la vie de l'entreprise ;
- Développement : architecture, design, développement et tests ;
- Opérationnels : infrastructures & plateformes, environnement & déploiement, gestion des incidents, gestion des versions et des changements ;
- Et donc la sécurité : les règles et politiques de sécurité de l'entreprise et pour les applications & données. Informations légales, tests de sécurité, sécurité de l'infrastructure.

Déjà, les développeurs peuvent réaliser des tests statiques et dynamiques, de l'intégration continue (qui va détecter d'autres failles et erreurs). C'est le premier niveau d'une approche Sec en DevOps.

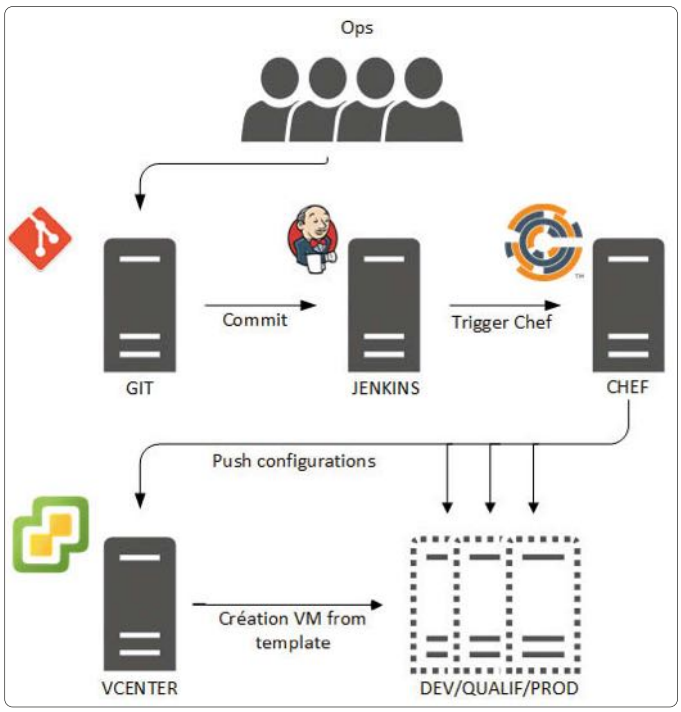
Soyons clairs : le DevOpsSec impose à considérer sérieusement les tests et la sécurité. Si dans un développement, le client, l'entreprise coupe le budget alloué à la sécurité ou aux tests, il faut être alors conscient du risque potentiel : l'application déployée aura une surface d'attaque non maîtrisée, non connue. Et le coût de corrections sera, de facto, élevé (en temps et en budget).

(1) Nuançons le propos. Car cela va dépendre de la nature de la faille découverte en conception / développement. S'il s'agit d'une faille liée à l'architecture ou aux spécifications du projet, il faudra modifier plus en profondeur le projet.

Les Ops aussi font de l'intégration continue

La philosophie DevOps a toujours eu un écho très fort du côté des développeurs bien qu'il soit tout à fait possible que les équipes d'infrastructure intègrent les outils ainsi que les bonnes méthodes de nos amis développeurs.

Fabien Dibot
Architecte Infrastructures et MVP Cloud and Datacenter Management, Fabien conseille ses clients sur des solutions Microsoft complexes On Premise ou dans le Cloud au sein du Groupe SII.



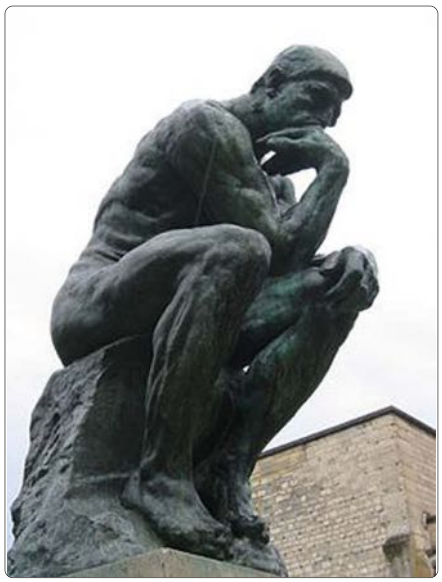
Lorsque nous avons souhaité avancer sur la mise en place d'une intégration continue pour l'infrastructure chez certains clients, le plus gros problème rencontré a été le silotage des différentes équipes système, parfois regroupées dans des entités juridiques différentes avec des objectifs et des compétences disparates. Un accompagnement fort au changement a été nécessaire avant de choisir les technologies à mettre en œuvre : il fallait revoir l'organisation de la DSI en intégrant les équipes techniques dans les décisions. Les modèles historiques consistant à séparer l'opérationnel du support pour chaque technologie sont obsolètes, les équipes doivent travailler entre elles avant de pouvoir penser à travailler avec les équipes de développeurs.
Deuxième problème et non des moindres : l'écosystème. L'informatique bouge vite, et les évolutions ne sont pas suivies en interne. Certaines personnes, travaillant dans l'écosystème Microsoft considèrent que l'interface graphique est toujours d'actualité ! La firme de Redmond, même si elle a mis le paquet sur l'automatisation ces dernières années, n'a pas convaincu tous ses utilisateurs. Au contraire des utilisateurs de systèmes GNU/Linux pour qui cela ne change (presque) rien à ce qu'ils faisaient précédemment. En plus d'une guerre pour son pré carré, il a fallu faire évoluer les mentalités sur l'adoption d'outils sortant complètement du périmètre de certains.
En fait, le plus simple a été de faire discuter les développeurs et les opérationnels ensemble ! Le bon sens l'emportant sur les vieilles rengaines qui pouvaient subsister, j'irais même plus loin en disant que les besoins, quand ils sont exprimés clairement, sont compris, et les solutions pour les résoudre mieux implémentées... Tiens ça ne serait pas un des buts du DevOps ?
Des nouvelles notions sont apparues dans les discussions, et j'ai parfois eu la forte impression de glisser vers un profil développeur. Les équipes système qui se mettent à parler de gestion de source, de tests unitaires... C'est un indicateur clair des changements qui se sont opérés dans le quotidien. Alors bien sûr, les produits continuent d'être étudiés et validés par des équipes transverses ; mais avant, une documentation d'installation manuelle était produite alors que ce sont dorénavant des scripts de configuration qui sont produits, et j'insiste dessus, avec les tests unitaires qui permettent de valider leur bon fonctionnement. Dorénavant, les scripts ne sont plus localisés sur de vagues partages réseaux, mais dans des outils tels que Git, ce qui pour les Ops, fut un énorme changement.
Les univers hétérogènes que sont les infrastructures informatiques ne permettent pas de faire un choix unique de technologie. De nombreux acteurs sur le marché permettent de faire ce qui est communément nommé Infrastructure as a Code. Le choix pour ce client s'est arrêté sur une solution Jenkins pour l'orchestration et Chef, afin d'assurer l'idempotence des configurations autant sur les serveurs Redhat que Windows. Jenkins et Chef sont les deux piliers centraux de la plateforme d'intégration continue. En effet, Jenkins et son slave vont orchestrer les actions et Chef les appliquer sur les différents éléments précédemment cités avec en plus la création des machines virtuelles.
Les premiers tests lors des développements sont effectués sur les machines des équipes dans des Kitchens Chef, et, suite au commit, un job Jenkins lance des tests unitaires avec le framework Pester pour les scripts PowerShell ainsi que BATS pour les scripts shell dans le pipeline. L'intérêt de ces deux frameworks étant qu'ils génèrent des logs au format xUnit pour pouvoir afficher correctement les résultats dans Jenkins. Et, en plus de tester le code, ils servent à valider que, du déploiement de la machine virtuelle à la livraison des softwares, tout est configuré correctement.
Il reste encore beaucoup de travail pour ce client. Le monitoring qui n'a pas réellement évolué devrait être optimisé en prenant en compte les demandes des développeurs ; cette action est en cours d'implémentation. De plus, l'intégration n'est pas complètement automatique, le passage entre chaque environnement nécessite une intervention humaine. Des outils pour auditer la qualité des scripts tels que PSScriptAnalyzer pourraient être rajoutés dans le processus pour aussi tendre vers l'amélioration continue. En fait pour avoir une vraie intégration continue il faudrait aussi que le client intègre aussi les équipes réseau dans cette philosophie, c'est ce qu'on appelle le DevOpsSec. L'avènement des containers amène aussi des nouvelles réflexions dans la boucle.
Tout ceci nous a permis, pour les applications cibles, de passer de 3 jours pour livrer un serveur en production à 2 heures, et de se rapprocher des rythmes de livraison des développeurs. Le tout avec une meilleure vision sur la configuration, et donc une gestion simplifiée. ⊞

MICRO FOCUS | SERENA™

Enterprise DevOps

Développement accéléré
Déploiement sécurisé
Mise en service en toute sérénité

www.serena.com
01 70 92 94 94
frinfo@serena.com



Agissez avec réflexion, non par réflexe

Le texte qui suit est un élan du cœur ayant pour modeste but d'amener le lecteur à prendre du recul sur sa façon de travailler et surtout à analyser ses réactions face aux problèmes rencontrés. Il n'a pas vocation à présenter des théories générales, ni défendre des best practices. Il ne s'agit pas non plus de remettre en cause votre organisation ou les processus qui guident votre travail. L'ambition est moindre. Dans le meilleur des cas, j'aimerais qu'à la suite de cette lecture, vous vous arrêtiez un instant pour mettre en doute vos solutions, vos méthodes et leur utilisation à l'instant T.

Raphaël Squelbut
Software Craftsman Arolla

A travers quelques exemples réels, j'espère montrer que pour un même problème, il existe de nombreuses bonnes solutions. En revanche pour plusieurs problèmes, il existe très rarement une unique bonne solution.

EXEMPLE 1

L'informatique de gestion tend à appliquer des règles métiers à des flux d'information.
Pour ce faire, il existe de multiples façons de procéder, par exemple : paramétrage de valeur en bases de données, moteur de règles, modélisation du métier dans l'applicatif (DDD).
Le conseil péremptoire qui va suivre a de fortes chances d'oublier des parties du problème :

Mettez vos règles métiers dans une base de données ! Comme ça, on peut en rajouter au fur et à mesure, sans relivrer.

Pourquoi pas, mais est-ce adapté dans notre cas ? Les règles sont-elles nombreuses ? Complexes ? Les deux ? Faut-il les modifier fréquemment ? Les remplacer ? Les superposer ? Les composer ? Sont-elles ordonnées ?
En fonction des réponses à ces questions, on pourra rechercher une approche qui répondra au mieux à notre problème.
Dans une de mes missions, nous avons codé un fichier Drools paramétrant le menu d'une application Web. Ainsi, le responsable opérationnel pouvait, en modifiant des dates dans ce fichier ou en spécialisant des catégories d'utilisateurs, masquer ou afficher certaines entrées du menu dépendant de la période de l'année. C'était parfait pour le besoin étant donné qu'il n'y avait pas de possibilité d'anticiper les dates correspondant aux entrées. De surcroît, ce n'était pas lourd, à la portée d'un non-technicien et peu risqué pour le site.
Initialement, j'avais prévu de paramétrer ce menu via une base de données. Mais pour mettre à jour mon menu, il aurait fallu :

- Soit créer une page d'administration sur le site, avec un rôle spécifique et les DAO et services de mise à jour, donc beaucoup de développement ;
- Soit bien créer des scripts de mise à jour, qu'il aurait fallu lier à des utilisateurs spécifiques pour la base de données qui avait des contraintes de sécurité différentes des miennes.

Ces solutions ne semblant pas adaptées au besoin en termes de coûts et/ou de complexité, nous avons réfléchi et trouvé cette solution Drools.
Pour autant, si on me demande aujourd'hui :
« J'ai un problème de règles qui sont difficiles à mettre à jour. Qu'est-ce qu'on peut faire ? »

Je ne répondrais pas forcément :
« Ok, je connais, on va utiliser Drools, c'est parfait pour ce genre de problématique ! »

Idem pour :
« J'ai des problèmes de performance sur mon application Web au niveau des accès bases. »

La réponse ne sera pas forcément :
« Ne bouge pas, j'ai ce qu'il te faut !! Avec Hibernate et un cache aux petits oignons, ça va rouler tout seul. »

Avant de choisir une solution, il faut se demander quel est le problème à adresser.
Pour cela, il est sage de se demander : quelle est la nature de ce problème ?
Qui est impacté ? A quelle fréquence ?
Comment est contourné le problème actuellement ? Est-ce humain ? Est-ce technique ? Fonctionnel ? Organisationnel ?
Est-ce vraiment important ? On le voit, l'arbre des questions est infini. Il n'y a donc pas de réponse toute faite et tout réflexe ne fait pas gagner du temps.

EXEMPLE 2

Autre biais possible, en craftsmen consciencieux, on se tient au courant de la « mode » dans le monde du développement logiciel. Et il est très tentant (inconsciemment peut-être) d'essayer de résoudre un nouveau problème avec l'idée défendue pertinemment par tel ou tel cador/coach/gourou.
Alors que l'an dernier, tout le monde essayait de dockeriser son application, cette année, on essaye de la splitter en micro-services, dans l'espoir que nos applications répondront ainsi au besoin, seront pérennes, scalables, évolutives, etc.
Mais dans les conférences ou les articles, la culture d'entreprise n'est pas abordée. Bien souvent, cette culture est ce qui conditionne le plus la qualité de l'application.

- Pour qui la scalabilité est-elle un besoin réel ?
- Qui a la capacité de mettre en place des équipes sachant appréhender ces nouvelles problématiques ?

Peu de monde.
Pour autant, même si l'on se sait incapable d'embrasser la totalité de ce paradigme, il ne faut pas rejeter en bloc ces idées. Il convient de se demander quelles en sont les parties qui

pourraient nous être utiles, par exemple :

- Arrêter de couper les applications en couches ;
- Utiliser le vocabulaire métier dans la modélisation ;
- Extraire de l'application globale une partie plus gourmande en ressource que les autres ;
- Etc.

En résumé, n'embrassez pas la nouveauté parce que c'est nouveau et bien vu mais parce qu'elle permet de résoudre vos problèmes de manière plus légère qu'actuellement.

EXEMPLE 3

Dernier exemple de cet article : les excès de wiki.

Un wiki est un espace de collaboration où chacun peut participer à hauteur de ses compétences.

C'est un outil formidable permettant d'obtenir des documentations souvent complètes s'adaptant aux différents points de vue des lecteurs. Chaque personne n'ayant pas compris un aspect de la documentation peut rajouter ou expliciter le paragraphe incompris. Et ceci, très simplement.

Cela ne signifie pas pour autant que ce wiki doive devenir un espace de stockage que chacun s'approprie en y ajoutant ce qu'il pense manquer dans la doc principale.

Par exemple :

- Un tutoriel pour installer Ruby sur son poste ;
- Une documentation sur les annotations en Spring.

Ces documents/tutoriels souvent clairs avec des réponses aux questions générales pullulent sur Internet. Pourquoi en héberger un ersatz sur notre wiki ?

Il y a quelque mois, quelqu'un a mis en place un nouvel outil dans la mission, basé sur deux ou trois technologies et frameworks connus : Ruby, Coffeescript entre autres.

Ce dernier crée un README à la racine du projet expliquant :

- A quoi sert le projet ;
- Quelles sont les technologies utilisées ;
- Comment installer le projet ;
- Un guide de démarrage ;

Le tout en moins de 50 lignes. Parfait.

On n'alourdit pas l'outil de documentation de toute la société en ajoutant une nouvelle documentation, qui sera placée presque au hasard. En outre, pas besoin de rechercher cette documentation, puisqu'elle est à la racine du projet.

Ici intervient un autre développeur souhaitant contribuer au projet. En lisant la documentation, il installe les différents modules mais butte sur l'installation de la bonne version de tel ou tel outil. Après quelques recherches sur Internet, il parvient à tout faire tourner.

Et là, il se dit :

« J'ai eu tellement de mal que je vais coller ce README dans le wiki puisque c'est la documentation générale de l'entreprise et y ajouter un tutoriel pour montrer comment installer la version x de telle dépendance qui m'a posé problème. »

La volonté de partager son savoir et d'aider son prochain est un très bon réflexe. Mais encore faut-il réfléchir à comment l'aider le mieux possible.

En effet désormais :

- On a 2 documentations dont on ne saura jamais laquelle est à jour sur telle ou telle partie ;
- Le wiki étant déjà touffu, la nouvelle doc ne sera pas simple à trouver et sera, de plus, inutile à la majorité de la société.
- On va ajouter dans le wiki de la documentation technique déjà existante sur Internet et dont on sait qu'elle sera dépréciée au prochain changement de version de telle ou telle dépendance.

N'aurait-il pas mieux valu, dans le README :

- Décrire le problème rencontré ?
- Expliquer qu'il s'agit d'un problème de version de telle dépendance ?
- Préciser qu'on peut le résoudre en allant voir ici et là sur Internet ?

Ainsi, on a la documentation toujours sous la main, qui guide, aide à comprendre et à surmonter les éventuels écueils rencontrés, ce qui est bien plus formateur que de donner une solution sans explication.

« Donne un poisson à un homme, tu le nourris pour un jour. Apprends-lui à pêcher, tu le nourris pour toujours. » dit le dicton. C'est un peu le même principe ici.

Voici un exemple de bonne volonté réflexe devenue contre-productive par manque de réflexion.

CONCLUSION

A la lecture de ces lignes, j'imagine que quelques exemples vécus vous viendront à l'esprit, comme ce collègue qui crée un fichier pom avec Spring, Hibernate et un driver JDBC à chaque début de projet.

Mais alors, que faire ? Y a-t-il des signes trahissant un comportement compulsif ? Comment reconnait-on l'adhérent au culte du Cargo ?

Mis à part me méfier instinctivement des collègues...

- Qui me disent « YAGNI » dès que je me pose des questions de modélisation ;
- Qui énumèrent leurs réunions de la veille lors du Daily Meetup ;
- Qui songent à rajouter une étape dans un process dès que l'équipe a rencontré un problème ;

Je me méfie surtout des coéquipiers qui font ce qu'on leur demande sans jamais remettre en cause ce qui leur est demandé.

Attention aussi à ne pas sombrer dans l'excès inverse consistant à tout remettre en cause et ne jamais rien décider. Michel Audiard disait :

« Un imbécile qui marche ira toujours plus loin que deux intellectuels assis. »

Je ne creuserai pas plus ces problématiques intéressantes ici. Avoir plus d'ambition à ce propos serait vain en ces quelques lignes. Nous l'avons vu, le sujet est vaste, les exemples nombreux et la bonne méthode infaillible, inexistante. ;-)

programmez! le magazine des développeurs

DevCon #1 :
après-midi Windows 10 IoT

15 juin 2016
Campus Microsoft France (Issy-les-Moulineaux)
A partir de 14h.

Inscription gratuite sur
www.inwink.com/events/devcon

Tous les détails sur www.programmez.com

AGILE, quelques conseils pour réussir son projet informatique

Je ne vous apprends rien en vous disant que les méthodes agiles ont grandement contribué à transformer notre mode opératoire depuis une quinzaine d'années. Toutefois, saviez-vous que selon le « Standish Group – 2015 CHAOS Report », le taux de succès moyen des projets agiles n'atteint que 39 % (Etude menée sur plus de 50000 projets à travers le monde durant l'année 2014)


Walid Ammar
Cellenza

Plusieurs organisations interprètent le concept d'agilité comme un nouveau modèle de processus remplaçant les modèles classiques (communément connus sous les noms « Waterfall » et « Cycle en V »), ceci est à mon avis une erreur car l'agilité n'est pas que « Processus ».

L'agilité est avant tout un état d'esprit, une attitude, une culture mise en œuvre par un ensemble de disciplines visant à augmenter la productivité des équipes tout en garantissant la santé de leurs écosystèmes et la qualité de leurs livrables.

Ces disciplines agissent directement sur les cycles de « Demande-Réponse » appelés aussi « boucles de Feedback », elles permettent ainsi aux organisations un haut niveau d'adaptation et une meilleure réactivité face à un marché polymorphe et en constante évolution.

On peut classer ces disciplines en deux grandes catégories :

- **Des disciplines orientées Processus** qui relatent des modèles de fonctionnement, de collaboration et d'organisation d'équipes ; type de planification à adopter, la cadence de travail à suivre, la répartition des rôles au sein d'une organisation, la manière de recueillir le besoin métier, la manière de restituer le résultat du travail de l'équipe...
- **Des disciplines orientées Technique** qui relatent des pratiques plus spécifiques à la manière de développer, évoluer, maintenir et délivrer du logiciel informatique ; comment estimer un travail à faire, comment produire et maintenir du code source, comment tester un logiciel, comment délivrer des incréments de valeurs et à quelle fréquence, comment aboutir à un niveau de qualité logicielle acceptable et surtout comment le maintenir dans le temps.

Les disciplines agiles ne résolvent pas les problèmes, elles permettent seulement de les mettre en évidence, c'est à nous de réagir en conséquence et c'est à partir de là que nous serons capables de mesurer notre « degré d'agilité ». Cet article regroupe certains conseils et illustre certaines pratiques améliorant les chances de succès d'un projet agile.

Bien construire les choses est aussi important que construire les bonnes choses

Beaucoup de personnes pensent que faire les bonnes choses (« Building the right things ») serait plus prioritaire et plus important que faire bien les choses (« Building the things right »).

Il est bien vrai que construire les bonnes choses est essentiel à la réussite d'un projet informatique ; cette ordonnance instrumentalisée par toutes les disciplines agiles orientées processus (DSDM, FDD, Crystal, Scrum, Kanban ...) engage les équipes projets à dédier une bien grande partie de leurs efforts à augmenter la pertinence des livrables qu'ils produisent.

Toutefois, construire les choses bien ne manque pas d'importance car cela garantit la pérennité des investissements informatiques d'une organisation et lui permet de capitaliser sur un patrimoine robuste et évolutif. Un logiciel pertinent mais instable est aussi mauvais qu'un logiciel impertinent !

« Certains Consultants et Coachs Agiles adorent évoquer la transformation Agile de Toyota et le succès qu'elle a eu en l'associant à de nouvelles démarches/processus de recherche et de développement. Ceci n'est pas totalement vrai car la marque japonaise ne doit pas son succès qu'à son seul contexte méthodologique mais aussi, et pour beaucoup, à **la qualité** de ses voitures. »

Comment veiller à ce que les choses soient bien faites ? Et bien c'est avant tout une histoire de « mindset ». Les organisations doivent d'abord investir en leurs capitaux humains pour les pousser à l'excellence, l'amélioration de la qualité du logiciel n'en sera que conséquence logique !

- Une équipe motivée et beaucoup plus productive qu'une équipe délaissée ;
- Une équipe formée régulièrement est beaucoup plus efficace qu'une équipe stagnante ;
- Une équipe libre d'expression est beaucoup plus pertinente qu'une équipe « muselée ».

Ces dernières années ont vu l'émergence du mouvement « Craftsmanship » qui prône l'excellence, le professionnalisme et la fierté dans le métier de développeur. Ceci est un très bon début pour toute organisation désirant faire bien les choses.

Rejoindre ce mouvement et s'inscrire dans cette démarche de partage et de contribution communautaire (Meetups, Dojos, Hackathons, Séminaires & Events...) changera profondément le regard d'un développeur sur son métier et lui permettra de se mettre en perspective pour atteindre l'excellence.

Il existe plusieurs communautés « Craftsmanship » dans le monde ; la communauté « Software Craftsmanship Paris » fut fondée en Octobre 2011 et regroupe aujourd'hui plus de 1500 professionnels de l'informatique, elle est accessible à tous et à toutes sans distinction technologique ou élitiste.

Toute personne désirant améliorer ses connaissances, perfectionner sa technicité, augmenter son sens de l'éthique et son professionnalisme y trouvera son compte.

Certains ouvrages sont aussi disponibles pour une initiation progressive aux concepts et aux principes du Craftsmanship – « The Clean Coder – Uncle Bob », « The Software Craftsman – Sandro Mancuso »,

Allez-y, vous avez tout à gagner et qui sait, vous aurez peut-être votre propre page dans le futur « Wandering Book ».

Le planning est sans importance, seule la planification compte !

Le titre de ce paragraphe correspond approximativement à la traduction française des citations suivantes :

- *"Plans are of little importance, but planning is essential" - Winston Churchill*
- *"Plans are nothing; planning is everything" - Dwight D. Eisenhower*

Churchill et Eisenhower expriment tous les deux la même idée : Il faut bien préparer son plan d'attaque avant de commencer une bataille, cependant une fois l'affrontement commencé, le plan ne servira plus et seules les manœuvres tactiques détermineront l'issue du conflit.

Malheureusement, beaucoup d'équipes accordent une grande importance au planning et en oublient sa vocation initiale. Le planning est un chemin/une carte menant à une fin et non une fin en soi.

Dans certains cas, se fixer un planning et s'y tenir quoi qu'il arrive peut coûter cher aussi bien à la santé d'un logiciel informatique (dégradation de la qualité, dysfonctionnement...) qu'à la santé des équipes qui le conduisent (démotivation, désengagement...).

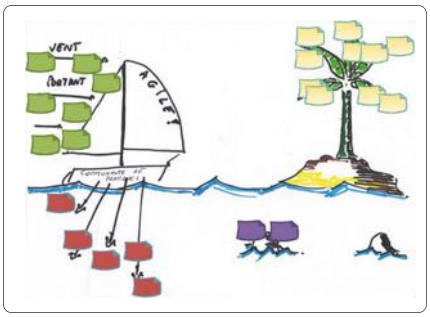
Certains managers accordent une importance capitale au planning simplement pour démontrer leurs capacités à contrôler les imprévus. Ces individus se trompent car personne ne peut contrôler un « imprévu », nous contrôlons seulement nos réactions face à ces imprévus. Un planning peut et doit surtout changer, nous devons d'ailleurs y être préparé et réagir en conséquence.

Maintenant si le planning est sans importance, pourquoi le réaliser ? Eh bien parce que la planification est essentielle ! Elle nous permet de rester focalisés sur ce qui importe le plus au vu des circonstances environnantes, c'est ce qui nous permet de considérer les changements et de réajuster « le tir » continuellement. Selon Kent Beck dans « Planning Extreme Programming », la planification nous permet de :

- Nous assurer constamment de la prise en charge des priorités ;
- Coordonner nos efforts avec d'autres intervenants et/ou équipes ;
- Analyser la répercussion d'imprévus sur les 2 précédents critères.

Une planification agile n'est efficace que si elle est récurrente et régulière. Il reste donc seulement à déterminer la fréquence qui nous convienne. La plupart des processus agiles proposent 2 créneaux réguliers :

- Un créneau journalier porté par chaque équipe projet : ce créneau sert à faire le point sur les avancées de la veille ainsi que les objectifs du jour. Le format « SpeedBoat » initialement présenté par « Luke Hohmann » comme outil de rétrospective, est parfaitement adapté à la tâche, il met en œuvre les axes suivants :
  - Ancres : Les problèmes non encore résolus ;
  - Rochers : Les risques pouvant engendrer de futurs problèmes ;
  - Les voiles : Les boosters permettant à l'équipe d'avancer dans de meilleures conditions ;
  - Les palmiers : Les objectifs à atteindre.



Crédit Photo :
http://www.morisseauconsulting.com/2015/08/17/pratiques-pour-manager-agile/

- Un créneau itératif rassemblant l'ensemble des équipes projets : ce créneau sert en général à faire le point sur l'avancement des travaux et à planifier une partie du « reste à faire », pour la plupart des méthodes agiles, ce créneau correspond au bilan d'itération. Il est très important de veiller à l'aspect fédérateur de ces séances qui contribuent à consolider l'esprit d'équipe et le « Collective Ownership ». Une orientation interactive au travers d'activités **ludiques** peut considérablement renforcer le sentiment d'appartenance des individus et favoriser leur bien-être au sein des organisations. Voici quelques exemples pouvant facilement être mis en pratique :
  - Rétribution des meilleurs intervenants d'un projet (Immunité « nerfique », attribution du meilleur fauteuil, café gratuit durant tout le prochain sprint...) ;
  - Châtiment des pires intervenants d'un projet (croissants pour toute l'équipe, prise en charge de certaines tâches détestables tel que la saisie des CRA, rédaction des Wiki en attente...) ;
  - Illustrations comiques pour relater les problèmes rencontrés, l'état d'avancement des travaux et toute autre actualité projet ;
  - Déroulement de certains jeux agiles pour réorganiser le backlog, redéfinir les priorités ou même pour réaffecter les équipes tel quel le « Product Box » ou encore « Remember the Future ».

Il est bien entendu possible de rajouter d'autres créneaux si cela conforte l'équipe dans sa planification ; bilan de mi-itération, Point de suivi de Release, etc.

Maintenir une planification récurrente et régulière est le secret d'une bonne conduite de projet.

Estimer est l'art de recenser le passé, analyser le présent et imaginer l'avenir.

L'estimation, cette tâche si compliquée mais tout autant importante pour le business constitue l'enfer de tout développeur !

« Quand est-ce que vous comptez terminer cette action ? », « à quel pourcentage jugez-vous votre avancement ? », « Allons-nous honorer notre engagement de livraison ? »...

On a tous été confronté à ces questions et la plupart d'entre nous ont réagi de la même manière.

Cette image illustre bien le sentiment d'un développeur quand on lui demande des projections qu'il n'est pas capable de fournir ; c'est un sentiment désagréable et extrêmement frustrant qu'on a tous dû ressentir un jour ou l'autre. Comment faire alors pour produire une estimation rationnelle et pertinente sans avoir le sentiment de déraciner un arbre de 100 ans ?

La solution est simple et applicable à tout problème complexe : Il faudra, simplement, le diviser en sous-problèmes. Pour cela, nous procèderons de la sorte :

- Nous devons d'abord recenser nos précédentes expériences assimilables à celle-ci. Cet exercice nous permet de mettre en évidences d'éventuels pièges et/ou problèmes déjà identifiés et donc anticiper leurs prises en charge.
- Nous devons analyser tous les paramètres contextuels impactant le périmètre concerné pour anticiper d'éventuels imprévus non encore identifiés à ce jour mais qui peuvent constituer des risques opérationnels.
- Une fois nos anciennes expériences remémorées et l'état des lieux effectué,

nous pouvons imaginer le travail à effectuer et mesurer alors son poids. Le « spike » est une bonne pratique si l'on ne maîtrise pas les moyens techniques de la solution à produire.

Il est vrai que toutes les unités de mesures sont viables du moment qu'on utilise la même unité tout le long du projet. Cela dit, il est préférable d'éviter l'emploi du Jour-Homme car cela combine inéluctablement des effets psychologiques secondaires induisant de l'incertitude, de l'imprécision et de la peur de l'engagement. Il existe plusieurs autres substitutions qui ne sont pas mal adoptées par les équipes agiles : Le point de complexité, Le jour idéal, La monnaie, La taille de T-Shirt, Le nombre de tomates...

Le « jour idéal » est une bonne unité de mesure assez granulaire et permet de mettre en évidence les problèmes et les imprévus rencontrés lors de l'implémentation (Productivité vs Vélocité).

Pour ce qui est du déroulement, la pratique du « Poker planning » est bien adaptée à la plupart des contextes projets. Issue de l'univers XP et largement adoptée dans les équipes SCRUM, cette pratique permet des interactions et un échange collaboratif de l'ensemble des intervenants d'un projet.

Ceci permet d'une part une manœuvre collégiale renforçant la confiance des développeurs, et d'autre part, une meilleure compréhension des besoins par l'ensemble des intervenants.

Dans intégration continue, il y a "intégration"... et "continue"

La fusion de code et les livraisons de releases constituent pour nombre de personnes un triathlon olympique long, fatigant et dont l'issue est incertaine.

Effectivement, nous avons tous connu le fameux « Code-Freeze » et la semaine infernale qui le suivait servant à regrouper, stabiliser, tester et déployer la dernière version du code.

Nous avons tous également gardé de bons souvenirs des longues nuits blanches passées aux bureaux à déboguer des erreurs de configuration suites aux livraisons précipitées.

Le remède est simple et existe depuis plus de 25 ans : l'intégration continue ! Il est malheureusement regrettable de voir qu'il reste encore pas mal d'équipes « agiles » qui n'emploient pas encore cette discipline (ou du moins croient l'adopter mais il n'en est rien).

L'intégration continue, « Continuous Integration », correspond à une association de pratiques et d'outils servants à garantir la capacité d'une équipe à livrer à tout moment. Elle constitue également une condition sine-qua-non de la pratique de livraison continue (« Continuous Delivery »).

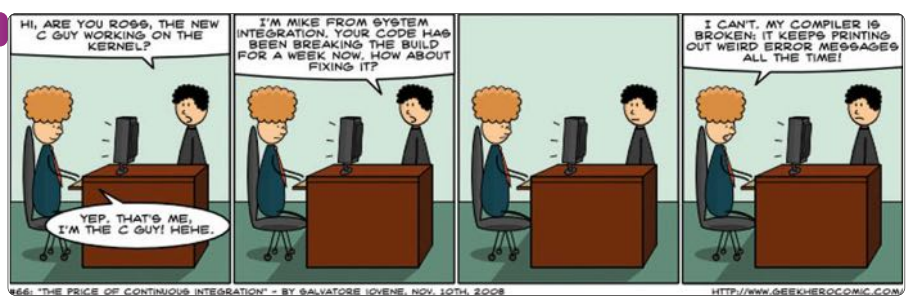
L'adjectif « Continue » illustre l'aspect ininterrompu de la pratique et c'est là que réside toute la subtilité de la discipline. Il faut impérativement que la mécanique tourne en continue et sans interruptions.

Installer un Dépôt de code source centralisé, un serveur de « Build » ou un serveur d'analyse statique ne veut pas pour autant dire que l'on fasse de l'IC. Malheureusement, nombreuses sont les organisations qui confondent usines logicielles et Intégration Continue.

Un projet qui se veut en intégration continue est un projet où :

- Tous les développeurs de l'équipe soumettent leurs changements de code source quotidiennement sur le dépôt central (voire plus d'une fois par jour) ;
- Des builds rapides et automatisées sont activées pour garantir en permanence la bonne construction du code ;
- L'équipe prend en charge et corrige immédiatement toute erreur causant l'échec des builds d'intégration ;
- Des tests automatisés sont activés pour éviter les régressions ;
- La qualité du code source est mesurée en permanence, des alertes automatiques notifient l'équipe en cas de baisse significative par rapport au niveau exigé ;
- Des chaînes de déploiement sont automatisées, préconfigurées et prêtes à l'emploi (Exemple : DEV -> UAT -> PPROD -> PROD) ;
- Des tableaux de bord de suivi et de monitoring de l'usine logicielle sont à disposition de tous les membres de l'équipe.

Fig.A



Un projet qui répond à l'ensemble de ces critères s'assurera une capacité de livraison optimale et ne souffrira point des effets d'une MEP douloureuse Fig.A.

Et si l'on offrait une vie à nos « specs »

Ah les fameuses « spécifications» ! Indispensables à tout projet informatique, elles permettent de décrire le fonctionnement d'un logiciel aussi bien d'un point de vue technique que fonctionnel.

Le problème avec les spécifications c'est qu'elles sont rarement mises à jour et rapidement négligées.

Nombreux sont les conflits qui ont opposé développeurs et experts métiers. Les uns soutenaient que les spécifications manquaient de détails et les autres affirmaient que les logiciels ne répondaient pas aux besoins exprimés dans ces mêmes « specs ».

Avec l'avènement des pratiques agiles, un nouveau format léger de spécifications a vu le jour et a contribué à relaxer ces conflits mais sans pour autant y mettre un terme : le « Backlog ». Exprimé sous forme de fonctionnalités et de récits utilisateurs, il a permis d'écarter les anciens formats papiers volumineux et peu efficaces de la table des négociations.

Cependant, le backlog souffrait des mêmes lacunes que les anciens formats et ne permettait pas de résoudre le problème des « DEFECT » - Un defect étant une fonctionnalité implémentée sans respect des spécifications énoncées.

C'est ainsi que la communauté agile s'est vu tournée vers une nouvelle approche complétement différente mais pas pour autant nouvelle : la substitution des spécifications par des tests exécutables (d'où l'appellation « Executable specifications » ou encore « Living documentation »).

Kent Beck fut l'un des premiers en 2001 à avoir proposé l'écriture des tests avant l'implémentation du code dans ses ouvrages « Planning Extreme Programming » et « TDD by Example », ce concept fut enrichi par Dan North en 2003 avec une syntaxe adaptée aux besoins métiers appelée « Gherkin » d'où l'avènement de l'approche BDD.

La pratique du BDD est une très bonne idée pour résoudre le problème des DEFECT et délivrer du logiciel pertinent et répondant aux attentes des organisations. Dans ce cas, les tests fonctionnels sont rédigés par les experts métiers, validés par toute l'équipe lors des séances de « Spec Review » et implémentées ensuite par les développeurs durant les sprints

ou au fil de l'eau. Il est crucial pour réussir une approche BDD de considérer la rédaction des scénarios « Gherkin » et leurs validation par l'équipe comme prérequis du DOR (Definition of Ready). Leur exécution, elle, sera comptée parmi les prérequis du DOD (Definition of Done). Il existe plusieurs outils intéressants pour accompagner une approche BDD dont l'un des plus connus est sans doute « Cucumber » (« SpecFlow » pour la communauté .NET). Il existe également plusieurs ouvrages illustrant des exemples métiers et décrivant les modes opératoires de certains produits comme par exemple « BDD In Action – John FERGUSON SMART».

Pour conclure, je dirais que s'il faut retenir une seule chose, ce serait sans doute de veiller à ne jamais implémenter d'incréments de code sans avoir au préalable rédigé et validé leurs scénarios de tests fonctionnels (Scénarios Gherkin). Cela garantirait l'actualisation des spécifications et évitera à l'équipe les problèmes de DEFECT.

Entre quatre yeux

Le « pair-programming » est une pratique très utile visant à augmenter la qualité du code et à partager la connaissance du logiciel au sein de l'équipe.

En apparence, cela semble trivial car il s'agit simplement de faire écrire le même code source par 2 personnes en même temps sur le même poste de travail. Toutefois, malgré les apparences, cette pratique n'est pas si simple à adopter car elle va au-delà du simple « binômage », elle met en œuvre des facultés sociales de collaboration, d'échange et de compromis.

Plusieurs parmi nous qui se sont prêtés au jeu, ont constaté la difficulté de la tâche, et ont dû renoncer au bout de quelques essais infructueux. Nombreux sont les développeurs qui n'aiment pas perdre leurs temps en regardant d'autres personnes travailler, beaucoup également n'aiment pas l'idée d'écrire du code sous le regard glacial et suspicieux d'autres personnes ou encore perdre du temps à débattre de ce qui leur semble évident...

Eh bien c'est précisément là que se situe l'erreur. Le « Pair-programming » est souvent vu comme une perte de temps ou encore comme une pratique dangereuse permettant aux uns de prendre de l'ascendant sur les autres :

- Beaucoup de développeurs assimilent cette relation à un apprentissage où le développeur le plus expérimenté dicte son commandement au développeur le plus « junior » ;
- D'autres développeurs souffrent de problèmes de confiance et ne souhaitent pas exposer ce qu'ils considèrent comme leurs « lacunes » à leurs collègues ;
- D'autres encore croient dur comme fer qu'ils perdront leurs temps avec des personnes « moins qualifiées » ou « moins compétentes » ;
- Finalement, il y a également ceux qui pratiquent l'hyper-concentration et qui sont habitués aux tunnels de programmation. Ceux-là ne peuvent pas se permettre des interruptions fréquentes dès lors qu'ils commencent à produire du code.

Pour réussir une séance de « pair-programming », il faudra d'abord s'affranchir de tous ces freins psychologiques et supposer que chaque membre d'un binôme est propriétaire du code source en question à hauteur de 50% - cette considération facilite beaucoup l'échange et place les partenaires sur un même pied d'égalité. Voici quelques conseils permettant de résorber cette frustration et d'offrir aux développeurs un cadre de travail positif favorisant la collaboration et permettant de tirer profit de cette pratique si avantageuse :

- Une séance de pair-programming commence à la réunion matinale (où sont fixés les objectifs du jour) et finit par un code qui compile où tous les tests passent au vert ;
- Le temps d'écriture/observation doit être équitablement réparti entre les individus, un gadget de type « pomodoro » peut beaucoup simplifier la répartition du temps ;
- Le conducteur (celui qui tient le clavier) doit exercer de la programmation à haute voix pour permettre à son binôme de mieux comprendre son approche et anticiper les points de blocage ;
- Chaque individu dispose d'un seul droit de véto à employer quand il se retrouve en situation de désaccord, ce droit lui confère la possibilité d'inverser les rôles et de prendre la main sur le clavier ;
- Prévoir des pauses régulières au rythme du binôme (toutes les heures, toutes les 90 min ...) ;
- Changer de pair au moins une fois par jour ;
- Ecouter attentivement ce qu'un partenaire essaie de dire même s'il y a une profonde divergence. Si aucun terrain d'entente n'est trouvé alors recourir à une tierce personne (arbitre) pour délibérer ;
- Ne jamais élever la voix ou user d'agressivité verbale.

Conclusion

L'agilité n'est pas quelque chose que l'on fait, c'est un état d'esprit qu'on essaye d'incarner chaque jour (sans toujours réussir d'ailleurs...) Etre agile c'est **capitaliser** sur l'humain, **produire** de la valeur et **réagir** face aux changements. Toute discipline s'alignant sur ces principes fondamentaux est bonne à adopter. ⊞

Vorlon.JS : adoption d'une démarche DevOps sur un projet de développement web/node.js

Quelle que soit la nature, la taille, le contexte d'un projet, adopter une démarche DevOps peut être bénéfique. Trop souvent, on entend dire que DevOps ne concerne que les gros projets, car cela peut être long à mettre en place. C'est à la fois vrai et faux... En effet, « faire du DevOps » n'a pas vraiment de sens en soit, il s'agit avant tout d'une culture qui prône le partage de la connaissance et l'entraide entre les équipes (dev, ops, mais pas que !), le fait de se mettre au service des autres, et nécessite aussi une bonne dose d'automatisation !

Julien Corioland
Technical Evangelist
Microsoft France

Il existe de nombreuses pratiques, pas forcément récentes d'ailleurs, que l'on englobe aujourd'hui sous le chapeau de DevOps. Vous avez certainement entendu parler d'intégration continue, de tests de charges, de tests fonctionnels, de release management, de déploiement continu (la liste est longue, mais je vais m'arrêter là...). Tout ça pourquoi ? Pour être capable de délivrer des fonctionnalités en continu sur un projet, tout en assurant la meilleure qualité possible. Tout simplement parce que c'est ce que veulent nos utilisateurs.

Le cas Vorlon.JS

Vorlon.JS est un outil de debug à distance pour les applications Web, qui a été conçu principalement pour répondre aux problématiques des développeurs d'applications Web mobiles qui rencontrent des difficultés pour debugger leur code client sur différents types de périphériques. Fig.1.
Concrètement, il s'agit d'une application Node.JS, qui utilise socket.io. Plutôt simple sur le papier. C'est vrai ! Mais ça n'empêche que lorsqu'on a commencé à vouloir livrer de plus en plus fréquemment des nouvelles versions ou de nouveaux plugins, on s'est vite rendu compte qu'il y avait moyen d'automatiser plein de choses : build/packaging, exécution des tests unitaires, création d'une image Docker et envoi dans le hub public, création du package NPM... Jusqu'au déploiement automatisé dans Microsoft Azure pour avoir toujours une instance de debug disponible.
Tout ceci s'étant passé une fois convaincu qu'il était nécessaire d'automatiser un certain nombre de tâches pour ne pas avoir à les refaire continuellement, manuellement, à chaque fois. Et pour ça, il fallait décider d'utiliser un outil qui soit capable de nous accompagner dans la mise en place d'intégration continue, de déploiement automatisé, etc. Nous avons choisi d'utiliser **Visual Studio Team Services**, la version SaaS/hébergée de Team Foundation Server.

Pourquoi avoir choisi Visual Studio Team Services ?

Visual Studio Team Services (VSTS) est un outil en ligne, ultra-complet, qui permet d'outiller vos développements logiciels et de mettre en place des usines de builds, tests et déploiements de la plus simple à la plus complexe qui soit.
L'outil propose également tout un système de suivi et gestion de projet, permettant la centralisation de toutes les informations nécessaires pour suivre l'évolution du projet dans le temps (user stories, tasks, bugs, tableau de bord Kanban...). Fig.2. Ce qui nous a surtout plu dans VSTS, c'est sa capacité à s'intégrer avec énormément d'outils existants que ce soit au niveau de la build ou du déploiement.

Intégration continue avec Visual Studio Team Services

Vorlon.JS est une application Web, développée en Node.JS. Nous utilisons des outils comme **NPM** pour la gestion des dépendances, **Gulp** pour la compilation des fichiers **TypeScript** et des fichiers **LESS** ou encore **Mocha.JS** pour la réalisation des tests unitaires.
Visual Studio Team Services gère nativement ces différents outils, et il suffit de quelques minutes et quelques clics dans l'interface Web pour mettre en place de l'intégration continue : Fig.3. Un des autres critères pour choisir l'outil était sa capacité d'intégration avec GitHub. En effet, Vorlon.JS est un projet open source hébergé sur GitHub (http://github.com/microsoftdx/vorlonjs). VSTS

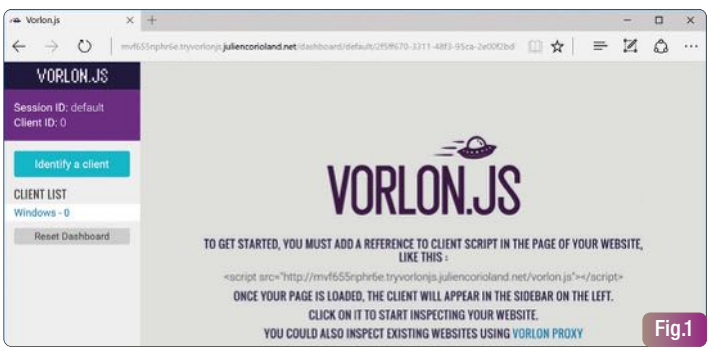


Fig.1

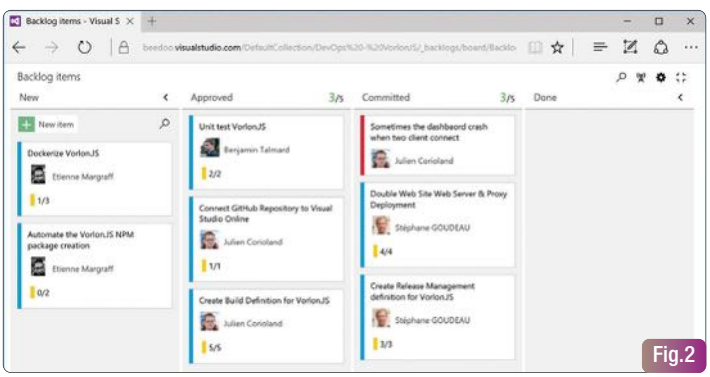


Fig.2

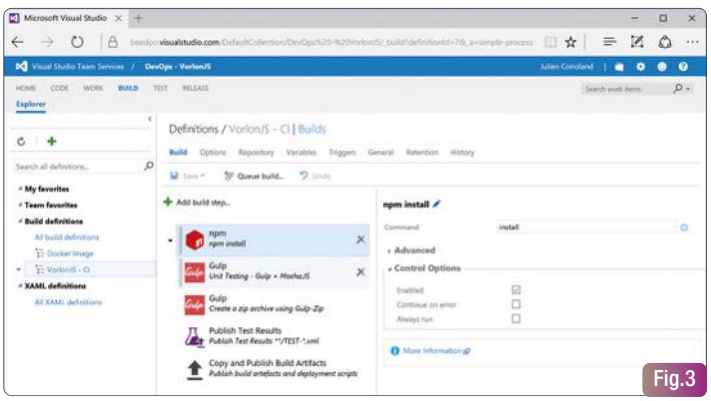


Fig.3

intègre nativement la connexion avec GitHub, aussi il est tout à fait possible de réaliser de l'intégration continue sur du code hébergé dans GitHub, il s'agit d'une simple option : Fig.4. L'un des autres points intéressants est la capacité de VSTS de s'intégrer avec des systèmes de tests unitaires tiers. En l'occurrence, pour Vorlon.JS nous utilisons le Framework de tests Mocha.JS qui est capable de générer un fichier de résultats au format **JUnit**. Il suffit ensuite d'ajouter une étape de type **Publish Test Results** pour publier les résultats directement dans VSTS : Fig.5. Ceux-ci sont alors visibles directement dans le rapport d'exécution : Fig.6.

Release Management et déploiement continu

L'une des finalités de la mise en place d'une démarche DevOps est d'être en capacité de délivrer des fonctionnalités en continu sur un projet. Visual Studio Team Services propose une fonctionnalité de **Release Management** qui permet de gérer le déploiement d'une application, automatiquement, sur divers environnements. L'intérêt d'avoir plusieurs environnements est de pouvoir déclencher, automatiquement ou manuellement, un ensemble de tests (fonctionnels ou de charge, par exemple) afin de valider la qualité de l'application avant que celle-ci n'arrive en production ! On peut par exemple envisager d'avoir un environnement de développement sur lequel l'application soit déployée directement après une build, puis sur un environnement de recette et de production après une approbation manuelle.
Le système de Release Management de VSTS est similaire au système de build et fonctionne avec des tâches qui s'enchaînent les unes à la suite des autres : Fig.7. VSTS fourni nativement un certain nombre de tâches pour déployer des applications vers **Azure**, **Docker** ou en utilisant des outils de configuration comme **Chef**, par exemple. Ce qui est également intéressant, c'est que pour certaines tâches que vous souhaitez effectuer et qui ne sont pas supportées nativement, il est possible d'exécuter un script **PowerShell**, ou **Bash** pour les traitements non Windows !

Fournir ses propres agents de build et release

Microsoft fournit des agents de build sous Windows nativement avec Visual Studio Team Services. Ceux-ci sont configurés pour fonctionner avec la majorité des outils de développement et tâches disponibles dans VSTS. Cependant, dans certains cas il est nécessaire d'utiliser des technologies qui ne sont pas forcément supportées nativement. Pour cela il est possible de configurer son propre agent et de le connecter à son compte Visual Studio Team Services. Il existe une version de l'agent pour Windows, Linux et OS X (notamment pour build **Xamarin** ou **XCode**, par exemple).
Les agents Linux et OS X sont disponibles directement sur GitHub : https://github.com/Microsoft/vso-agent

Conclusion

Visual Studio Team Services se positionne donc comme un excellent outil pour gérer l'intégration et le déploiement continu de vos applications, même si celles-ci ne sont pas développées avec des technologies Microsoft, qu'il s'agisse des applications développées en node.js (comme Vorlon.JS), en Java, ou encore à destination des plateformes iOS et Android. Pour ces applications mobiles justement, les rachats récents de **HockeyApp** (store pour les applications en béta à destination d'Android, iOS et Windows) et **Xamarin** permet à VSTS d'offrir l'une des plateformes « Mobile DevOps » la plus complète qui soit : du contrôle de code source, en passant par l'intégration continue, les tests automatisés avec **Xamarin Test Cloud** jusqu'aux déploiements automatisés et à la remontée de feedbacks pour alimenter son backlog !
Pour tester Visual Studio Team Services, rendez-vous sur cette page : https://www.visualstudio.com/en-us/get-started/setup/sign-up-for-visual-studio-team-services.

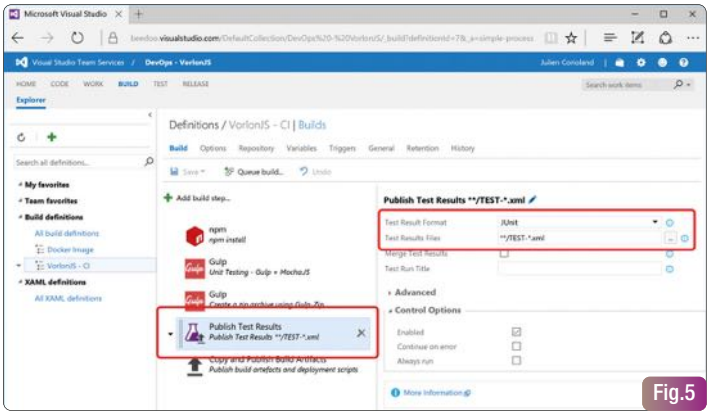


Fig.5

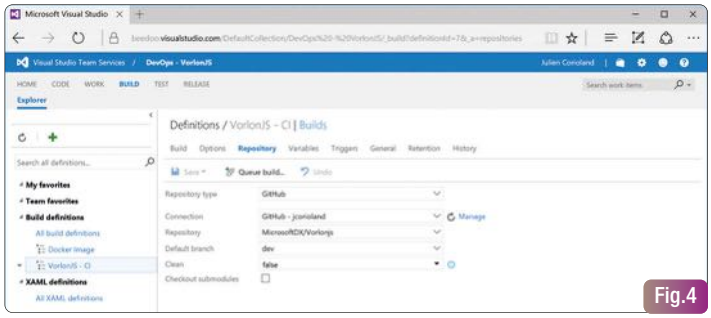


Fig.4

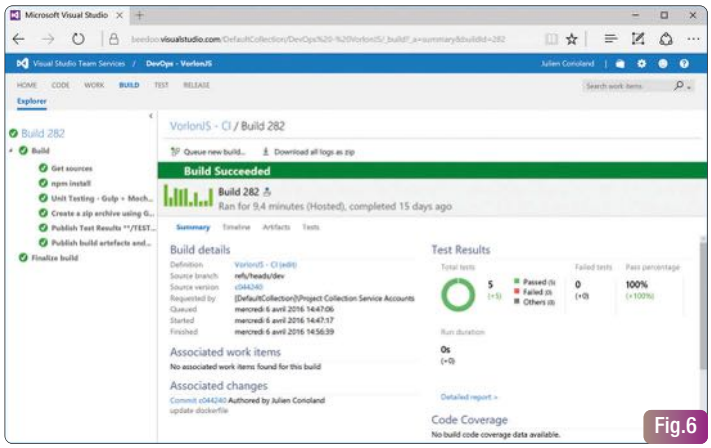


Fig.6

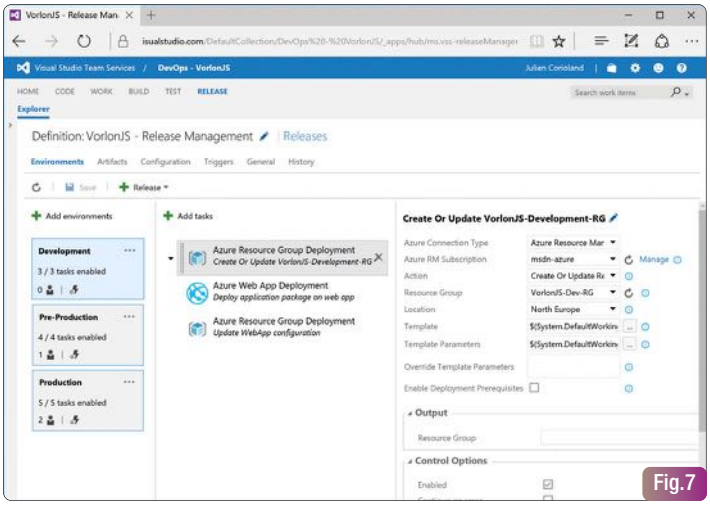


Fig.7

En quoi les conteneurs aident le développeur et prennent leur place dans le DevOps

Le « DevOps » est un terme inventé par Patrick Desbois en 2009. John Allspaw et Jesse Robbins sont également à l'origine de ce mouvement, né d'un constat, celui de la faible synergie des équipes de développement d'applications et d'administration des systèmes.

Cedric Leblond
Consultant DevOps chez MERITIS, MVP ALM.
@leblond_c

Pierre-Henri Gache
Consultant ALM chez Cellenza, MVP ALM.
@phgache
www.pierrehenrigache.com

Vincent Thavonekham
Microsoft Practice Manager VISEO, MVP Azure.
@vthavo. www.thavo.com

Aujourd'hui, cette approche est plus que jamais présentée pour diminuer le « Time to Market » des applications et ainsi être compétitif, en limitant le manque de communication entre Dev et Ops, origine du « wall of confusion ». Cela est accentué par le fait que ces deux services n'ont pas les mêmes objectifs ; livrer toujours plus vite pour les Dev, et garantir la stabilité et la sécurité des applications en production pour les Ops.
Aussi, est-il nécessaire de trouver des processus et des outils capables de répondre aux attentes de tous. C'est en cela que Docker est l'outil idéal. Simple et léger, il permet aux Devs de tester et déployer des conteneurs aussi bien lors de leurs développements, que lors des releases. Scalable et facile à déployer, il sera vite adopté également par les Ops.

Ce que sont les conteneurs ?

Il y a quelques années, les applications fonctionnaient sur des machines physiques. Généralement provisionnées de manière unitaire, elles étaient compliquées à mettre en place et la configuration variait souvent d'une génération à une autre. Conscient de ce problème et du manque de souplesse, le choix se porte aujourd'hui sur la virtualisation pour provisionner des environnements destinés à héberger des applications.
Ces machines virtuelles (VM) déployées de manières automatisées et standardisées ont permis d'apporter une réelle amélioration dans le déploiement des applications. On a ainsi aujourd'hui la possibilité avec le Cloud de déployer des environnements de Dev & Tests en quelques clics.
Mais avec les méthodologies agiles, qui ont accéléré les cycles de développement, l'utilisation de VM a atteint ses limites en termes de souplesse. C'est alors qu'interviennent les conteneurs qui représentent les dernières évolutions permettant aussi bien aux Devs qu'aux Ops de fournir un environnement modulaire, déployable en quelques minutes, et ce sur n'importe quel OS. Voyons donc plus en détails ce que sont ces conteneurs ; nous verrons en quoi les « Docker-files » seront clef.

Les VM sont « lourdes » et par opposition, les conteneurs font une isolation au niveau système d'exploitation (OS). Ils permettent d'isoler les ressources du système hôte et des autres conteneurs entre eux. Cela signifie que le système hôte et les autres conteneurs n'ont pas accès aux différents fichiers, mémoire et service. Des liens peuvent être créés pour partager des points d'entrées et permettre aux conteneurs de travailler ensemble. Les conteneurs partagent les instructions du noyau du système d'exploitation, ce qui implique qu'un conteneur créé sur une plateforme Linux ne pourra malheureusement pas être utilisé directement sur Windows.
L'image d'un conteneur est constituée de différentes couches qui se superposent. Nous démarrons un conteneur à partir d'une image. Une couche supplémentaire est alors automatiquement ajoutée dans laquelle toutes les modifications peuvent être réalisées. Les couches inférieures restent quant à elles inchangées. Si nous lançons un second conteneur, celui-ci sera instancié depuis la même image mais complètement isolé du précédent. Les conteneurs sont donc « idempotents » grâce à leurs images « immutables » ; ce sont des notions critiques dans les architectures modernes Fig.1.

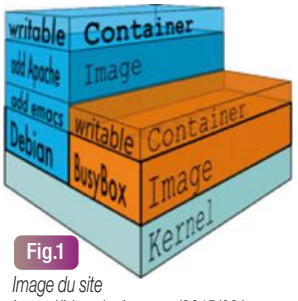


Fig.1
Image du site https://blog.docker.com/2015/08/docker-demo-faq/

Les conteneurs fournissent une abstraction au niveau système. Ils sont différents des VM, car ils se lancent directement sur le système hôte. Leur démarrage est ainsi quasi-instantané, presque aussi rapide que le lancement d'une application. En outre, l'image d'un conteneur est aussi plus légère, car elle embarque uniquement les « deltas », i.e. les différences par rapport à l'image sur laquelle elle se base. Elle « pèse » de quelques mégas à quelques centaines de méga-octets, selon le contenu. L'image d'une VM est souvent de plusieurs giga-octets, car elle inclue l'OS.
Docker est un projet Open Source apparu en mars 2013. Il s'est appuyé au départ sur les outils Linux (LXC, Linux Container) qui existent depuis longtemps. Docker utilise maintenant un service « daemon » pour gérer les conteneurs de manière simplifiée. Il propose une série d'outils « client » en ligne de commande très cohérente et facile à apprendre. Ces outils sont cross-plateforme « X-Plat ». Ils couvrent l'ensemble du cycle de vie d'une application pour développer, builder, distribuer, configurer, surveiller et mettre en production. Tout cela facilite beaucoup son adoption et celle des conteneurs.
Le succès important de Docker est à l'origine de l'investissement dans la technologie des conteneurs de tous les principaux acteurs de la virtualisation et des services Cloud. Ils ont créé ou soutiennent l'organisme Open Container Initiative (OCI). En juillet 2015, Docker a fait donation de composants afin de promouvoir l'émergence d'un format d'image et d'outils d'exécution standards. Ils constituent la base des travaux de standardisation de l'organisme. La communauté autour des conteneurs est très active avec beaucoup de projets importants comme le portage sur la prochaine version de Windows Server, la création d'OS dédiés à l'exécution de conteneur (RancherOS, PhotonOS).

Les éléments clés

Démarre sur l'OS directement (noyau)

Il n'y a aucune couche de virtualisation et d'abstraction comme sur les VM. Il faut donc que le conteneur soit « compatible » avec l'OS hôte. Concrètement un conteneur d'une image Linux x64 ne fonctionnera pas sur un Raspberry PI car l'architecture est différente entre les deux. En revanche

l'absence de la couche de virtualisation permet d'avoir un gain important en termes de performance.

Les conteneurs sont isolés, ils ne partagent que l'OS.

Comme évoqué précédemment, les conteneurs partagent l'OS hôte pour s'exécuter. En revanche, ils sont complétements isolés les uns des autres. Il n'y a pas non plus d'accès au système sous-jacent.

Instanciées à partir d'images

Chaque conteneur à besoin de son propre système d'exploitation (sans le noyau) pour s'exécuter. C'est-à-dire tous les binaires système. Pour cela Docker se base sur un système d'empilement d'images. Chaque Dockerfile, à l'exception du premier, fera référence à une autre image. On aura ainsi des images pour le système d'exploitation sur lequel on ajoutera, par exemple, un serveur applicatif, puis pour finir les binaires de l'application.

Par rapport aux VM, gain considérable en taille (OS) et en empreinte mémoire

Chaque conteneur étant dépourvu du noyau du système, il n'est donc pas nécessaire d'allouer de l'espace disque et mémoire pour l'instancier. On peut donc démarrer beaucoup plus de conteneurs que de VM sur une même machine.

Les conteneurs comme clef de l'agilité et de la simplicité dans le DevOps

Nous allons voir en quoi les conteneurs contribuent au DevOps.

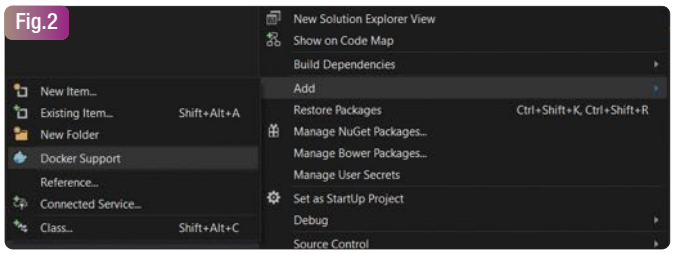
Des conteneurs pour gérer des environnements, des plateformes et des processus hétérogènes

Qui n'a jamais entendu un développeur s'exclamer « oui, mais ça marche sur mon PC... » ? ... Eh bien il y a des circonstances dans lesquelles cela se fait mieux que d'autres: une application est complexe et comporte de nombreux facteurs :

- L'environnement : Linux, Windows XP / 7 / 10, Raspberry PI
- Les processeurs : Intel vs. ARM ; 32 bit vs 64 bit,
- Les plateformes :
  - PC : Poste local / Dev / Test / Intégration / Recette / Pré-prod / Prod
  - Mobile : Android, iOS...
- Les processus : les processus de livraisons du service de Développement sont et doivent être complètement différents de ceux de la Production, où ce dernier intègre parfois des notions ITIL v3, où les problématiques et les enjeux sont complètement différents de ceux du développement.
  La durée des « sprints » est souvent en totale contradiction :
  - Les Devs veulent des Sprints de plus en plus courts (par exemple 15 jours afin d'être davantage Agile).
  - Les Ops : fonctionnent parfois en Kanban et/ou ont des cycles de validation bien plus longs, avec des critères de « no-Go » (i.e. de refus de mise en production) différents de ceux des développeurs.

Du coup, nous allons voir comment les conteneurs comme Docker peuvent concilier ces problématiques.

Fig.2



Docker : un outil aussi pour les Devs

Lors de nos développements, nous sommes souvent amenés à utiliser plusieurs éléments pour qu'une application puisse fonctionner (base de données, service de cache, ...). Parfois longs et complexes à installer, tous ces modules requièrent une configuration précise pour fonctionner de concert, avec des fichiers de configuration qu'il est parfois long et fastidieux de documenter pour le service d'Exploitation.

Avantages des conteneurs :

- Ils sont lancés quasi instantanément sur le poste de développement, car construits en quelques minutes à partir de fichiers de configuration (Dockerfile) ou téléchargés depuis un repository.
- Lorsqu'un nouveau développeur arrive sur le projet, il peut préparer son environnement et être opérationnel en quelques minutes ; terminée la longue liste de prérequis à installer (souvent pas ou mal documentés).

Comme les conteneurs sont isolés, ils ne modifient pas l'environnement du poste, là où, par le passé nous passions plusieurs jours à réinstaller notre PC pour travailler sur un nouveau projet. Pour basculer vers une version différente de l'application, il est possible de lancer en parallèle une autre version des conteneurs sans risque que des données ou configurations ne soient mélangées.

Le conteneur représentant l'environnement cible, l'idéal est donc de coder et tester son application directement à l'intérieur.

La fonctionnalité de volume des conteneurs est essentielle. Les volumes permettent de partager des répertoires locaux avec le conteneur. Dès qu'un code est modifié sur le PC, celui-ci est également modifié dans le conteneur. En adaptant les outils de l'environnement de développement, il est alors possible de développer et de déboguer directement dans un conteneur. La garantie d'un comportement identique de l'application sur d'autres environnements est ainsi assurée. Les problématiques liées aux environnements hétérogènes sont par conséquent éliminées.

Pour les développements ASP.NET Core, les outils Visual Studio 2015 Tools for Docker Preview donnent la possibilité de déboguer avec "Edit and Refresh" sur le projet par défaut (1).

Cette extension permet de piloter Docker directement sur le poste de développement en ajoutant au projet le support nécessaire, comme le montre la capture ci-dessous : Fig.2.

En combinant cette extension avec « Docker for Windows », il est possible d'avoir en local l'ensemble des ressources nécessaires Fig.3.

Bien que ces deux éléments soient encore en « preview », ils montrent bien que Docker n'est plus un outil réservé aux Ops. Maintenant chaque développeur peut en tirer parti.

Déploiements sur des environnements hétérogènes

Le DevOps facilite l'accès « à la demande » des environnements pour tous :

- Le développeur pour vérifier la bonne intégration de ses modifications ;

Fig.3

(1) https://visualstudiogallery.msdn.microsoft.com/0f5b2caa-ea00-41c8-b8a2-058c7da0b3e4

- Le testeur pour valider les nouvelles fonctionnalités ;
- L'IT pour s'assurer que le dernier patch des machines hôtes n'introduit aucune régression.

Chacun sera libre de l'utiliser selon ses besoins cibles spécifiques ; les conteneurs sont déjà pleinement supportés sur nombre d'environnements. En voici, quelques exemples :

- Système local (ou dans une machine virtuelle locale) ;
- Azure : machine virtuelle ou Azure Container Services ;
- AWS : machine virtuelle ;
- Docker Cloud gérant le déploiement vers votre Cloud public ;
- Raspberry PI : un support intéressant pour un scénario distribué ou IoT.

Malgré cette hétérogénéité, une fois en production, un critère important que de nombreuses sociétés recherchent est la « réversibilité ». Ceci pour éviter notamment les problématiques de « vendor-lock ». Imaginons que l'on ait choisi, par exemple AWS, il sera quasi-impossible de migrer vers Microsoft Azure ou un hébergement « On Premise », et vice-et-versa. Or, avec des conteneurs cela devient facilement possible.

La simplicité : clé du succès de Docker

La simplicité de Docker est la raison principale de son succès, cela nous permet redécouvrir les conteneurs et leurs avantages. Pour créer un conteneur et une image de son application existante, il suffit de quelques minutes (quelques heures s'il s'agit de vos premiers pas avec Docker). L'utilisation du Dockerfile avec la commande docker build permettent de construire une image prête à l'emploi de notre application. Le Dockerfile est constitué de quelques commandes de base qui permettent de déployer notre application à l'intérieur d'un conteneur et d'en sauvegarder son état. Pour un site Web ASP.Net Core 1.0, nous pouvons utiliser le suivant :

```
FROM microsoft/aspnet:1.0.0-rc1-update1

ADD . /app
WORKDIR /app

ENTRYPOINT ["dnx", "-p", "./approot/src/PartsUnlimitedWebsite/project.json", "web", "--server.urls", "http://0.0.0.0:5000"]
```

La première ligne « FROM » indique que nous partons d'une image de base, nous indiquons ici utiliser le runtime ASP.NET en version RC1 pour Linux. Docker tire cette image depuis le répertoire public. A chaque commande, Docker enregistre (commit) le conteneur, créant ainsi une nouvelle couche. L'ensemble de celles-ci constituera l'image de notre site. La commande « ADD » transfère le contenu du répertoire courant local, notre site Web, dans le conteneur. La commande « WORKDIR » configure le répertoire courant. Enfin, la commande « ENTRYPOINT » indique la commande principale lancée au démarrage d'un conteneur ; ici nous lançons notre site Web pour écouter sur le port 5000.

Pour construire l'image et démarrer celle-ci, nous publions d'abord le site Web en local sur le système de fichier, puis nous exécutons les commandes suivantes :

```
docker build -t monsite -f MonSiteDockerfile .
docker run -t -d -p 8080:5000 --name monSiteDocker monsite
```

Le site Web est alors disponible sur http://localhost:8080 (ou http://docker:8080 si vous utilisez la beta Docker for Windows/MAC). La commande build crée une image « monsite » à l'aide du Dockerfile placé dans le répertoire courant. La commande build lance un conteneur au nom « monSiteDocker » à partir de l'image « monsite », le « -p 8080:5000 » précise de rediriger les appels sur le port 8080 du système hôte vers le port 5000 du conteneur sur lequel le site écoute.

DevOps : simplification des processus « Ops » ITIL

Avant tout passage vers la pré-production et production, de nombreux processus sont à mettre en place ; notamment dans un contexte ITIL, où il est important de considérer qui valide et quand et quelle version ? Les rôles et les responsabilités de chacun sont à définir précisément, toujours dans le but de limiter le phénomène de « wall of confusion ».

« RACI matrix » : exemple de rôles et de responsabilités avant une livraison en production en production

ID	Responsible (qui réalise l'action ?)	Accountable (qui est responsable ?)	Consulted	Informed
10 Développement et Build en localhost	Développeur	Développeur	N/A	Chef de projet / Autres développeurs
20 Développement et déploiement sur le serveur d'intégration fonctionnel	Développeur	Chef de projet	Equipe de développement	Responsable Produit
30 Déploiement sur le serveur de pré-production	Développeur	Chef de projet	Equipe d'Exploitation (Ops)	Responsable Produit
... ...	...	...	...	...

Par exemple, ce mécanisme de garantie de qualité via un processus clair peut être assuré avec l'outil Microsoft Visual Studio Team Services (VSTS) – Release Management.

Intégration dans le processus de déploiement continu

L'intégration dans le processus de déploiement continu est un élément clé pour l'adoption de Docker. Cela diffère sensiblement d'une usine logicielle habituelle ; en voici les grandes étapes :

Build

Traditionnellement la première étape consiste à produire les binaires de l'application, puis la seconde à générer un package les contenant. Il existe plusieurs moyens pour générer ce package : MSI, WebDeploy, Zip pour ne citer que les plus utilisés. C'est alors qu'interviennent les problèmes car ces packages ne sont pas standardisés et il faut donc expliquer aux Ops comment les déployer. Les outils d'automatisation des déploiements résolvent en partie ce problème en réduisant les interventions manuelles et donc les mauvaises manipulations. Avec Docker ce problème est derrière nous car l'application sera packagée dans un conteneur. Prenons comme exemple la build suivante : Fig.4.

La première étape consiste à compiler l'application. La tâche suivante génère une image Docker contenant la sortie de build. Pour finir, cette image est uploadée vers Docker Hub où elle est stockée et versionnée.

Docker Hub

Contrairement à un déploiement classique, les conteneurs sont stockés dans Docker Hub. Ce dernier permet d'héberger un ensemble de packages et également d'obtenir ceux qui sont mis à disposition par la communauté. Docker Hub est un repository public à l'instar de GitHub pour le code source. Mais il est également possible de stocker de manière privée ses images ou encore d'installer un service propre à ses besoins.

Déploiements

Maintenant que les images Dockers sont disponibles, il reste à les assembler lors du déploiement de l'application. Dans ce cas Release Management de VSTS va uniquement gérer le paramétrage et l'ordonnancement de l'installation des conteneurs sur la machine cible. Il est également possible d'utiliser docker-compose et de lancer cette commande via la tâche qui porte le même nom Fig.5.

Remarque : la tâche permettant d'effectuer ces opérations dans VSTS est disponible dans le marketplace. Bien penser à l'installer avant de démarrer votre projet.

Production

Dès lors que le stade du déploiement en production est validé, il est possible de publier l'application en production. Pour cela l'application sera certainement déployée sur un cluster afin de garantir sa capacité à monter en charge. Plusieurs technologies sont disponibles pour effectuer cette opération, voici la liste des outils les plus populaires :

- Kubernetes
- Swarm
- Mesos

Conclusion

Docker est donc une solution pour réaliser du DevOps plus simplement. Cela dit, quelques limitations sont à considérer, comme les problématiques de licences, notamment sur les produits comme les bases de données Oracle, qui imposent une licence même sur des environnements éphémères de développement. Ou encore des « briques » PaaS « Platform as a Service » qui sont très pratiques, mais non « containerisable » et imposent alors d'être « vendor-lock ».

Liens

Linux Containers : https://linuxcontainers.org/fr/

Windows Containers : https://msdn.microsoft.com/en-us/virtualization/windowscontainers

"Old" operating-system-level virtualization :

- LXC : https://en.wikipedia.org/wiki/LXC ,
- Solaris Zones https://en.wikipedia.org/wiki/Solaris_Containers,
- BSD Jails https://en.wikipedia.org/wiki/FreeBSD_jail

Docker: https://www.docker.com/what-docker

https://www.opencontainers.org/about

Beginner friendly on containers and docker : https://medium.freecodecamp.com/a-beginner-friendly-introduction-to-containers-vms-and-docker-79a9e3e119b

Marc Russinovitch : https://azure.microsoft.com/en-us/blog/containers-docker-windows-and-trends/

Stephane Goudeau : https://blogs.msdn.microsoft.com/stephgou/2015/05/11/clusters-de-containers-docker/

Julien Corioland sur TP5 : https://blogs.msdn.microsoft.com/jcorioland/2016/04/28/getting-started-with-containers-and-docker-on-windows-server-2016-technical-preview-5/

Article Xspirit : https://xpirit.com/wp-content/uploads/2016/03/Xpirit-magazine-2-Docker-Microsoft.pdf

Fig.4

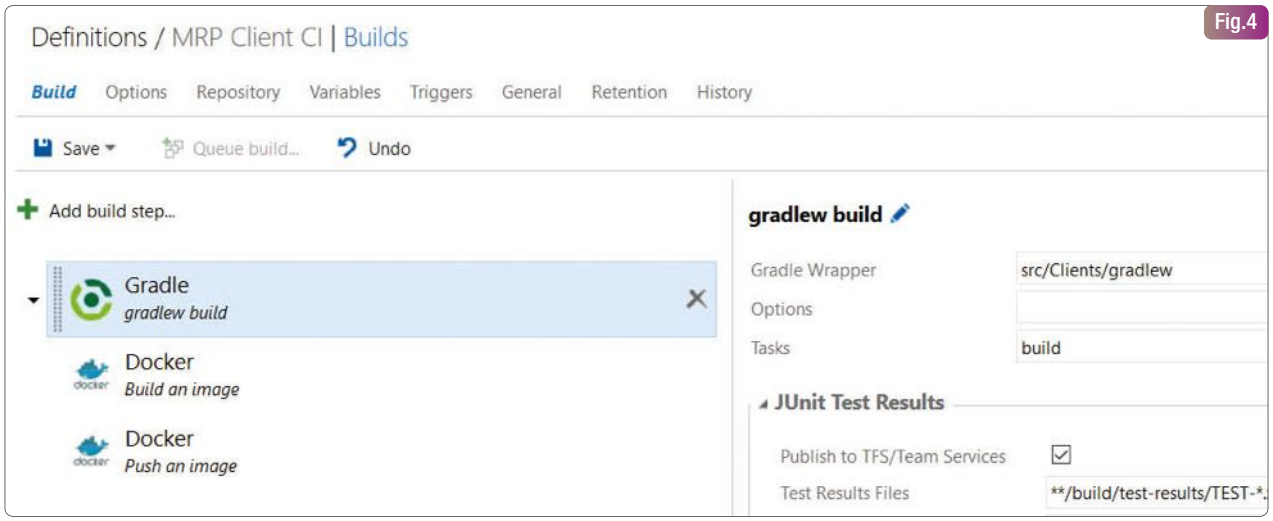
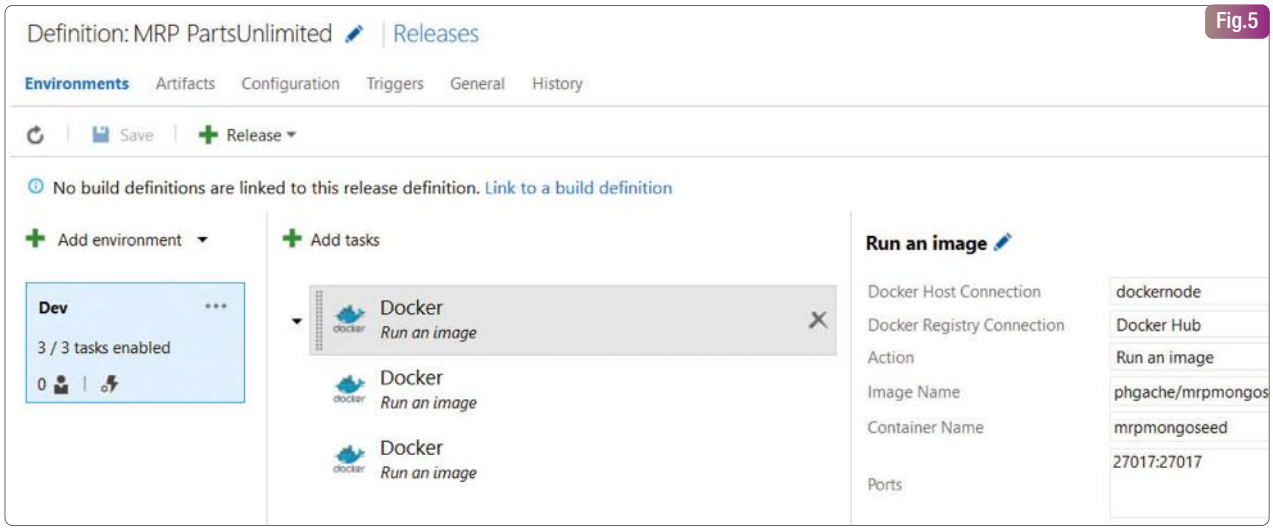


Fig.5



Accélérer, automatiser, standardiser

DevOps est partout. Tout le monde dit en faire. De quoi parle-t-on réellement ? Sylvain Cailliau (Directeur technique de Serena Software, groupe Micro Focus) revient avec nous sur les notions fondamentales du DevOps et évoque les avantages qu'il peut vous apporter.

Des enquêtes évoquent presque la moitié des entreprises françaises faisant du DevOps. Est-ce réel, n'y a-t-il pas un problème de compréhension du DevOps par les entreprises ?

Il y a effectivement un problème de définition. Le DevOps n'est pas une solution sur étagère que l'on peut acheter. Le DevOps a été une « solution » qui a été proposé (le terme a été inventé par Patrick Debois) suite à un constat : la spécialisation des équipes et des profils (la création de silos de compétences) pour gérer la complexité des architectures et des applications informatiques à entrainer une sclérose dans le processus du changement du SI. Des bénéfices évidents ont été obtenus en matières de robustesse et de qualité mais au prix de la flexibilité et de l'adaptation aux besoins des métiers. Les différentes équipes ont perdu de vue l'objectif commun et se concentrent sur l'excellence dans leur domaine. À l'origine du DevOps, il s'agissait donc de remettre tous les acteurs dans une même dynamique, avoir un objectif commun, partager la vision d'ensemble et remettre de la fluidité entre les équipes.

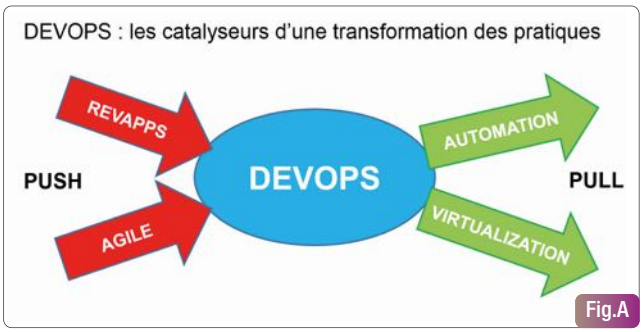


Fig.A

Quels mots clés pour définir le DevOps ?

Jez Humble a défini en 6 points ce qui est nécessaire à une démarche DevOps :

- Créer un processus fiable et répétable
- Automatiser presque tout
- Tout doit être versionné et contrôlé (y compris les scripts d'automatisation)
- Si ça fait mal, faîtes le souvent
- La qualité fait partie intrinsèque du processus
- « Finit » veut dire « En Production »
- Tout le monde est responsable du processus de déploiement

Je voudrais, pour ma part souligner que « l'effet DevOps » est boosté par 4 catalyseurs : Fig.A.

Deux qui permettent le DevOps (effet « Pull ») :

- L'automatisation : automatiser tout ce qui peut l'être comme si c'était du code
- La virtualisation qui va permettre une plus grande flexibilité

Deux qui poussent le DevOps (effet « Push ») :

- L'adoption de l'agilité par les équipes de développement qui conduit à la multiplication des livraisons d'applications
- La mise à disposition des applications non plus aux employés de l'entreprise mais directement aux clients.

Ce dernier point en particulier change de manière très importante la manière de penser ces applications : leur utilisation impacte directement le chiffre d'affaires de l'entreprise et elles participent aux stratégies de développement des nouvelles offres et des activités marketing qui les accompagnent : elles doivent donc plaire, évoluer au rythme des besoins du marketing digital, s'avérer performantes) et être d'une robustesse à toute épreuve. L'impact sur le nombre de versions qui seront livrées en tests (fonctionnels, performances, robustesse, sécurité ...) est considérable.

Mais ce besoin de qualité n'est-il pas contradictoire avec la notion d'un time to market toujours plus court et des développements plus complexes. L'accélération des développements et donc du déploiement s'accommode-t-elle de la qualité ?

L'accélération est forcément un facteur de risque et pour pallier les possibles problèmes de qualité il n'y a qu'un seul moyen : accélérer aussi tout ce qui contribue à la qualité. Comme évoqué plus haut, la seule façon de réussir est d'automatiser tout ce qui est possible. D'autre part, j'évoquais le rôle de la virtualisation, les conteneurs apportent une souplesse supplémentaire. Ils sont prêts à l'emploi et faciles à déployer et à maintenir. Le conteneur est un vrai avantage et jouera un rôle de plus en plus important dans la perspective de cette accélération.

En 2014, Gartner propose une vision bimodale de l'informatique (Bimodal IT), l'un séquentiel et traditionnel, l'autre non linéaire et adapté à l'agilité et à la rapidité. Est-ce toujours d'actualité et est-ce que le DevOps favorise un nouveau modèle IT et applicatif ?

En réalité, il existe plus que deux modes comme proposé par Gartner : la vitesse de déploiement d'une Application est déterminée par l'architecture et le type de ainsi que par les exigences de conformité à des normes. L'objectif est de faire progresser les équipes sur tous le gradient des applications qui couvrent les systèmes d'innovation, de différentiation et d'enregistrement. Dans une telle perspective de « Multimodal IT », la visibilité sur l'avancement des différentes des composantes du SI est très importante : il faut mettre en place une « tour de contrôle » des processus pour garantir la cohérence et donné la visibilité à tous les niveaux de l'organisation.

DevOps modifie forcément l'organisation des équipes. Il faut communiquer, simplifier et standardiser pour que tout le monde se comprenne. Nous ne pouvons, dans une entreprise, avoir x équipes avec leur propres spécificités, ni 50 méthodes pour faire des configurations et des déploiements.

Le DevOps est-il « rentable » ?

Prenons une entreprise qui pousse en production 8 versions majeures par an et 90 versions mineures. Il s'agit déjà d'un bon rythme. Imaginez que cette entreprise veuille doubler les versions déployées, le DevOps va apporter des pratiques, des outils, de l'automatisation. Les équipes vont accélérer et automatiser les projets et les provisionnements d'infrastructure, au final les mêmes équipes absorberont la charge supplémentaire (des entreprises avec lesquelles nous travaillons indiquent une productivité jusqu'à 8 ETP pour le profil indiqué ci-dessus).

N'utilise-t-on pas un peu n'importe comment le terme DevOps ?

Il s'agit d'un vrai problème. Si une entreprise veut des outils DevOps, cela ne veut rien dire. DevOps est une démarche, ça ne s'achète pas sur étagère ! Mais les outils peuvent accélérer les projets en apportant de l'automatisation, des bonnes pratiques, un cadre plus formel.

SQL Server 2016

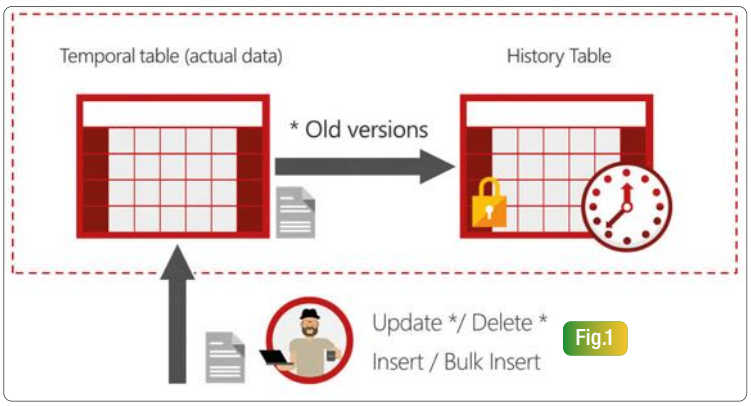
Deux ans après la sortie de SQL Server 2014, Microsoft nous livre cette année une nouvelle version de son système de bases de données relationnelles star, devenu le leader du marché. S'il fallait encore convaincre les DSI de prendre la software assurance avec leurs licences, le rythme soutenu de sortie des releases devrait finir de les persuader. Cet article vous propose de découvrir les nouvelles fonctionnalités de cette monture.

David Joubert
Leader Cercle Data Factory chez **Cellenza**
MVP Data Platform
Microsoft P-Seller
@dj_uber

Michel Hubert
Directeur Technique chez **Cellenza**
MVP Azure, Regional Director
@michelhubert

Microsoft Most Valuable Professional

Microsoft Regional Director



Nouveautés côté moteur

Un système de bases de données relationnelles est d'abord caractérisé par son moteur relationnel. Réputé mature sur SQL Server, on a pu compter sur l'arrivée du InMemory (Column Store Index et moteur Hekaton) en 2012. Pas de révolution pour la version 2016, mais quelques améliorations tout de même bienvenues.

Vous allez notamment pouvoir aujourd'hui utiliser les column store index sur des tables InMemory. Jusqu'à maintenant, vous deviez choisir entre les deux : typiquement, on utilisait des tables InMemory (moteur Hekaton) plutôt pour le transactionnel, tandis que les column store index avaient les faveurs des datawarehouses. Cette limitation est désormais révolue. On gagne en plus la possibilité de créer des index sur une table possédant un CSI. Il s'agit d'un gain notable en flexibilité.

Faisons le tour de nouvelles fonctionnalités.

Temporal tables Fig.1

Première nouveauté intéressante, les temporal tables. Il s'agit d'un nouveau type de tables permettant de garder un historique complet des modifications de données de ladite table. Avantage en découlant, on peut retrouver le contenu de la table à une date donnée.

Comment ça marche ? A la création de la table, on spécifie les colonnes qui correspondent à la date de début et à la date de fin de validité et une table d'historisation :

```
CREATE TABLE Employee
(
  [EmployeeID] int NOT NULL PRIMARY KEY CLUSTERED
  , [Name] nvarchar(100) NOT NULL
  , [Position] varchar(100) NOT NULL
  , [Department] varchar(100) NOT NULL
  , [Address] nvarchar(1024) NOT NULL
  , [AnnualSalary] decimal (10,2) NOT NULL
  , [ValidFrom] datetime2 (2) GENERATED ALWAYS AS ROW START
  , [ValidTo] datetime2 (2) GENERATED ALWAYS AS ROW END
  , PERIOD FOR SYSTEM_TIME (ValidFrom, ValidTo)
)
WITH (SYSTEM_VERSIONING = ON (HISTORY_TABLE = dbo.EmployeeHistory));
```

Ensuite, toute la gestion est transparente :

- Si vous insérez une nouvelle ligne (comprendre une ligne avec une nouvelle valeur de clé primaire), la ligne est insérée dans la table principale avec une date de départ correspondant à la date d'exécution de la requête et une date de fin à 31/12/9999.
- Si vous mettez à jour/supprimez une ligne, elles sont déplacées dans la table d'historique avec les dates mises à jour. Dans la table principale, la ligne est supprimée dans le cas d'un DELETE, et les valeurs mises à jour dans le cas d'un UPDATE avec les dates de validité adéquates.

Pour requêter une telle table, vous disposez d'un nouveau mot-clé FOR SYSTEM_TIME avec 5 clauses suivant les résultats souhaités :

- AS OF <date_time>
- FROM <start_date_time> TO <end_date_time>
- BETWEEN <start_date_time> AND <end_date_time>
- CONTAINED IN (<start_date_time> , <end_date_time>)
- ALL

Cela permet de récupérer les données actives à une date précise (AS OF) ou sur une plage donnée (FROM TO et BETWEEN AND), les données qui ont changé de statut sur une plage de données (CONTAINED IN), ou enfin la totalité des données (ALL).

Exemple :

```
SELECT * FROM Employee
FOR SYSTEM_TIME
BETWEEN '2014-01-01 00:00:00.0000000' AND '2015-01-01 00:00:00.0000000'
```

Row Level Security

Autre nouveauté intéressante, le row level security qui va permettre de filtrer les données dans un SELECT de manière transparente pour un user donné et sans avoir à créer de procédure stockée ou autre, à l'instar de ce qui existe dans les cubes OLAP.

Concrètement, les cas d'usage sont nombreux :

- Un commercial qui ne pourrait voir que les ventes qu'il a lui-même effectuées ;
- Un médecin qui ne doit voir que les données de ses patients.

La configuration se fait en deux temps :

- On crée la fonction qui va récupérer le login de la connexion ;
- On crée ensuite une stratégie de sécurité sur la table que l'on souhaite filtrer en spécifiant la colonne sur laquelle on s'appuie pour valider l'accès pour ne retourner que les lignes qui matchent avec notre login.

```
CREATE FUNCTION dbo.userAccessPredicate(@UserName sysname)
    RETURNS TABLE
    WITH SCHEMABINDING
AS
    RETURN SELECT 1 AS accessResult
    WHERE @UserName = SUSER_SNAME()
GO

CREATE SECURITY POLICY dbo.userAccessPolicy
    ADD FILTER PREDICATE dbo.userAccessPredicate(UserName) ON dbo.MyTable
GO
```

Cette fonctionnalité vous permet de gérer l'accès aux données au sein des stratégies de sécurité et non plus dans différentes procédures stockées. Cela évite également de créer de nouvelles vues juste pour gérer la sécurité. Bref cela simplifie la maintenance en permettant aux utilisateurs de requêter directement les tables.

Dynamic Data Masking

On continue sur la sécurité avec une nouvelle fonctionnalité : le Dynamic Data Masking.

Elle vous permet de crypter des données sensibles de manière simplifiée. Imaginons que votre base de données contiennent des numéros de cartes bancaires, des emails,... que vous ne souhaitez pas divulguer à votre équipe de développement. Précédemment, vous deviez créer une solution d'anonymisation à la main. Avec SQL Server 2016, il suffit de spécifier une propriété MASKED WITH sur la colonne désirée à la création de la table :

```
CREATE TABLE Membership
(MemberID int IDENTITY PRIMARY KEY,
FirstName varchar(100) MASKED WITH (FUNCTION = 'partial(1,"XXXXXXX",0)') NULL,
LastName varchar(100) NOT NULL,
Phone# varchar(12) MASKED WITH (FUNCTION = 'default()') NULL,
Email varchar(100) MASKED WITH (FUNCTION = 'email()') NULL);
```

Comme nous l'observons dans notre exemple de script, cette propriété accepte plusieurs fonctions pour obtenir le masque voulu :

- default() : crypte toute la valeur ;
- email() : remplace le mail en ne gardant que la première lettre et l'extension du domaine ;
- partial(prefix,[padding],suffix) : remplace la chaîne selon le pattern défini ;
- random([start range], [end range]) : retourne un entier situé dans la plage spécifié ;

```
INSERT Membership (FirstName, LastName, Phone#, Email) VALUES
('Roberto', 'Tamburello', '555.123.4567', 'RTamburello@contoso.com');

SELECT * FROM Membership;
```

Exemple : Le résultat obtenu :

1	RXXXXXXX	Tamburello	xxxx	RXXX@XXXX.com

Qui respecte bien les fonctions définies. Les droits des utilisateurs à lire des colonnes masquées se gèrent facilement au travers de privilèges :

```
GRANT UNMASK TO TestUser;
```

Query Store

Le Query Store est une feature qui va ravir les DBA. Il permet en effet de monitorer les exécutions des requêtes. Il va simplifier le dépannage des performances, en les aidant à trouver rapidement les différences de performance causées par les changements de plan d'exécution.

Le Query Store capture automatiquement un historique des requêtes, des plans et des statistiques d'exécution, et les conserve. Il sépare les données par fenêtre temporelle de sorte que vous pouvez voir les patterns d'utilisation des bases de données et ainsi comprendre quand les changements de plan de requête ont eu lieu sur le serveur.

L'activation est facile :

```
ALTER DATABASE AdventureWorks2012 SET QUERY_STORE = ON;
```

Exemple d'écran : Fig.2.

Pour aller plus loin : https://msdn.microsoft.com/en-us/library/dn817826.aspx

Support JSON Fig.3.

Le JSON est supporté nativement dans SQL Server, à la manière de la gestion du XML dans SQL Server (FRO XML AUTO, par exemple). Alors ce n'est pas, à proprement parler, un nouveau type disponible, les données étant stockées au format NVARCHAR. Mais on a désormais à disposition tout un jeu de fonctions permettant d'effectuer les tâches courantes :

- ISJSON : teste si la valeur est au format JSON
- JSON_VALUE : retourne une valeur du JSON
- JSON_QUERY : retourne un objet ou un scalaire du JSON
- JSON_MODIFY : met à jour la valeur d'un propriété d'un JSON
- OPENJSON : fonction qui va vous permettre de transformer votre JSON en rowset
- FOR JSON : va vous permettre de renvoyer un JSON à partir de vos données

Exemple :

```
SELECT TOP 3 [BusinessEntityID]
      ,[PersonType]
      ,[NameStyle]
      ,[Title]
      ,[FirstName]
      ,[MiddleName]
      ,[LastName]
  FROM [AdventureWorks2016CTP3].[Person].[Person]
  FOR JSON AUTO
```

Renvoie le JSON suivant :

```
[{"BusinessEntityID":1,"PersonType":"EM","NameStyle":false,"FirstName":"Ken","MiddleName":"J","LastName":"Sánchez"},
{"BusinessEntityID":2,"PersonType":"EM","NameStyle":false,"FirstName":"Terri","MiddleName":"Lee","LastName":"Duffy"},
{"BusinessEntityID":3,"PersonType":"EM","NameStyle":false,"FirstName":"Roberto","LastName":"Tamburello"}]
```

La requête ci-dessous :

```
SELECT *
FROM OPENJSON( N'[ null, "string", 1, true, false, ["a","r","r","a","y"], {"obj":"ect"} ]')
```

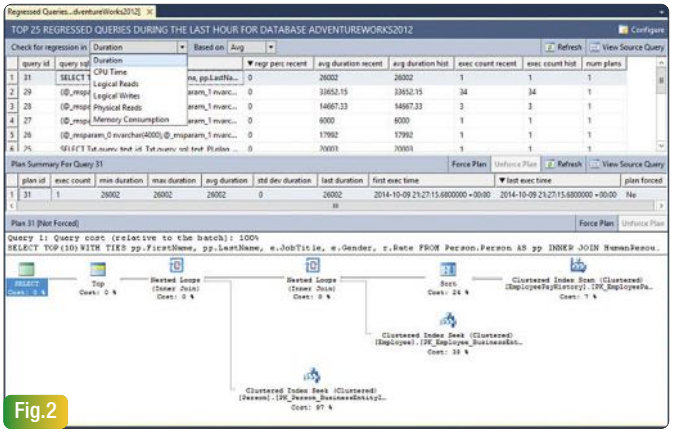


Fig.2

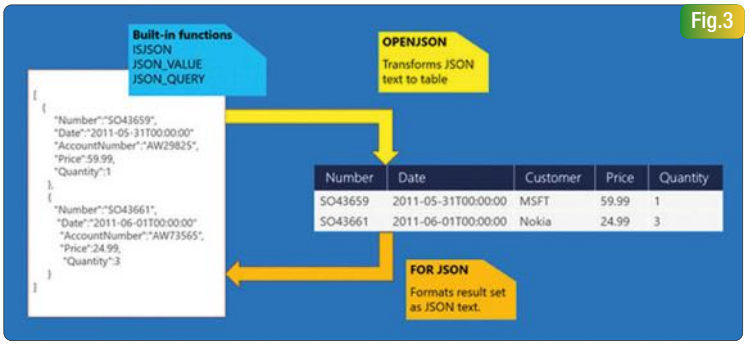


Fig.3

	key	value	type
1	0	NULL	0
2	1	string	1
3	2	1	2
4	3	true	3
5	4	false	3
6	5	["a","r","r","a","y"]	4
7	6	{"obj":"ect"}	5

Renvoie la table suivante :
Pour aller plus loin :
https://msdn.microsoft.com/en-us/library/dn921897.aspx

Mais aussi...

Cette présentation de nouvelles fonctionnalités ne se veut pas exhaustive. Il existe en effet encore d'autres nouveautés, je pense notamment au **Always Encrypted**, option qui permet de sécuriser les transferts de données entre une base de données sous SQL Server 2016 et un applicatif. On peut également parler du **live query statistics** qui vous permet de connaître l'état d'avancement en temps réel de l'exécution de votre requête.

Les architectures hybrides

La tendance forte aujourd'hui est aux architectures hybrides. Les questions sous-jacentes sont les suivantes : comment faire communiquer mes données issues de bases relationnelles avec mes données issues de bases non relationnelles ? Comment utiliser du on-premise et du Cloud de manière transparente ?
Pour répondre à ces questions, Microsoft implémente deux nouvelles fonctionnalités dans SQL Server 2016 : Polybase et les Stretch database.

Polybase

Polybase, moteur déjà disponible depuis quelques années dans Parallel Data Warehouse (PDW, l'appliance MPP de Microsoft), arrive dans SQL Server. Il permet d'utiliser du Transact-SQL pour accéder à des données stockées sur un cluster Hadoop ou sur des blob storages. Concrètement, à quoi cela va nous servir ? Outre l'accès simplifié à nos données Big Data, on va en plus, pouvoir les mixer dans des requêtes, à nos données relationnelles. Cela va également rendre plus simples nos tâches d'importation et d'exportation (passage des données de Hadoop ver SQL Server et vice-versa). Le gros avantage est que tout ça reste transparent pour l'utilisateur, aucune connaissance Hadoop n'est requise pour récupérer ses jeux de données. Et si c'est transparent pour l'utilisateur, ça l'est également pour les outils tiers (Microsoft ou non), tout logiciel acceptant du SQL est compatible. Dernier point intéressant, et pas des moindres, Polybase repose sur un principe fort d'Hadoop, on envoie le calcul vers la donnée : le moteur pousse donc des jobs à la volée vers vos données. Bien sûr, cela est vrai si vous avez des clusters, un peu moins vrai s'il s'agit de blob storage Fig.4.

Stretch database

On a parlé d'une architecture hybride entre données relationnelles et non relationnelles, voyons ce que SQL Server 2016 propose pour mixer technologies on-premise et Cloud Fig.5. L'idée, ici, est de stocker une partie d'une table sur un serveur on-premise, et une autre partie sur une base SQL Azure, le tout de manière totalement transparente pour les utilisateurs. On pense à une table avec une grosse volumétrie due à un historique important : on garde alors les données « chaudes » sur notre serveur local et les données « froides » sur le Cloud.
Les avantages sont évidents :

- Vous bénéficiez du coût de stockage moindre dans Azure, plutôt que de devoir mettre à l'échelle votre instance locale ;
- Vous bénéficiez de performances accrues si vous travaillez sur vos données « chaudes » ;
- Vous réduisez vos tâches de maintenance sur votre serveur on-premise ;
- Et bien sûr, cela reste transparent : vous n'avez pas à modifier vos requêtes.

L'implémentation se fait au travers d'un assistant, et vous avez la possibilité de migrer la totalité d'une table ou de configurer un filtre (typiquement sur une date) Fig.6.

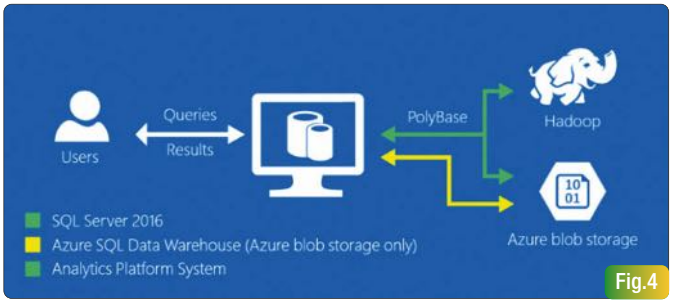


Fig.4

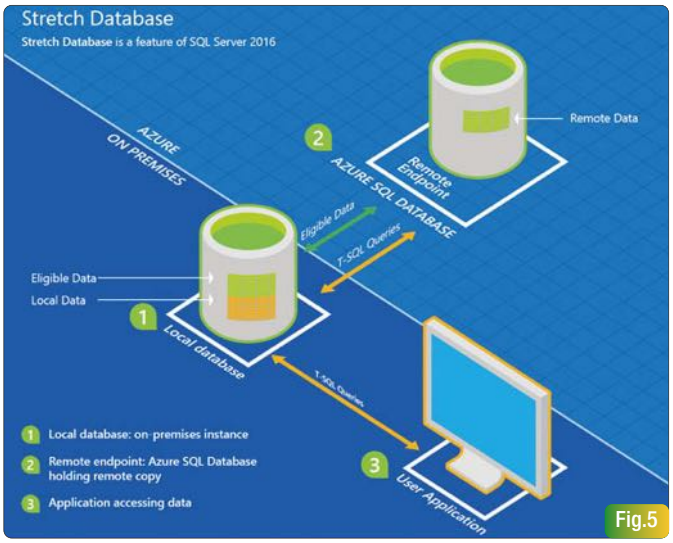


Fig.5

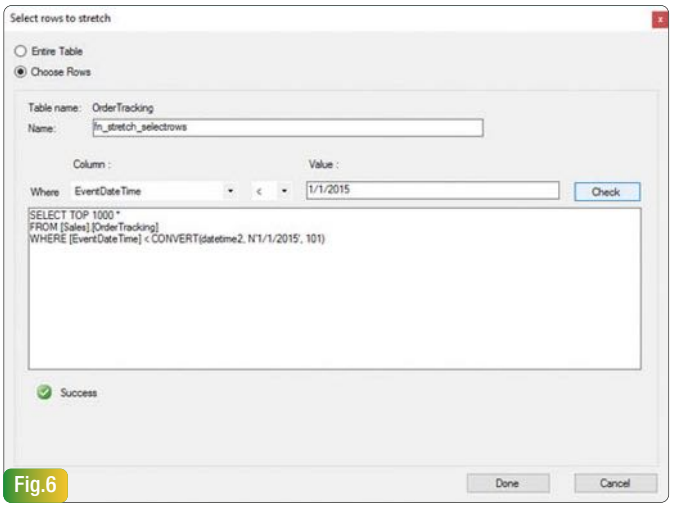


Fig.6

Et l'analytics dans tout ça ?

SQL Server n'est pas seulement un moteur de base de données. Depuis de nombreuses années maintenant, l'outil est aussi réputé pour la suite Business Intelligence qu'il propose. Loin de se reposer sur ses acquis, Microsoft continue de développer cette branche, notamment avec le rachat de Revolution Analytics l'année dernière.

R services

Depuis ce rachat, on savait que Microsoft voulait associer R (langage particulièrement prisé des data-scientists) à SQL Server, on attendait juste de voir comment cela se ferait. On a la réponse à cette question avec SQL Server 2016 et sa fonctionnalité R Services.
R Services, qu'est-ce que c'est ? C'est à la fois un enrichissement des packages R pour fonctionner avec SQL Server, et la possibilité dans SQL Server d'exécuter du script R (A noter que cette possibilité sera étendue à d'autre langages dans le futur, comme le Python ou le JSON).

De ce fait que va nous apporter R Services ?

Cela va permettre à SQL Server d'industrialiser des solutions créées en R : Fig.7.

La possibilité d'exécuter du script R dans SQL Server permet d'utiliser directement les données stockées en base. De la même manière cela autorise à intégrer vos résultats dans vos applications. Tout cela pour gagner en interopérabilité entre R et vos applications métiers.

Cela va également autoriser des modèles plus performants : Fig.8.

Ici, c'est la fonctionnalité qui permet d'utiliser SQL Server directement dans ses scripts R qui va faire gagner en performance. D'une part parce qu'on pousse le calcul vers les données à la manière de ce que l'on peut voir dans l'écosystème Hadoop, ce qui élimine les mouvements de données inutiles, d'autre part parce qu'on profite d'une infrastructure serveur pour exécuter ce calcul. En conclusion, une belle feature qui montre la volonté de Microsoft d'aller jouer dans le monde de la data science.

Nouveautés BI

La Business Intelligence plus traditionnelle n'est pas en reste, et la firme de Redmond apporte son lot de nouveautés à ses outils historiques. On pense notamment à Reporting Services, qui depuis sa version 2008R2, n'avait pas forcément beaucoup évolué.

Integration Services

Plus de nouveautés qu'il n'y parait pour l'ETL de Microsoft.

D'un point de vue développeur, on pense au rajout de connecteurs essentiellement basés sur des sources Cloud : l'Azure Blob Storage ou le HDFS. En plus des connecteurs, cela signifie qu'on a maintenant les composants source et destination. Fig.9. D'autres connecteurs arrivent également : OData v4, SAP BW, ... Au niveau des composants, on peut noter la disponibilité en natif du Balanced Data Distributor qui permet de distribuer un buffer SSIS à travers plusieurs threads. Ce composant nécessitait précédemment un téléchargement séparé. Fig.10.

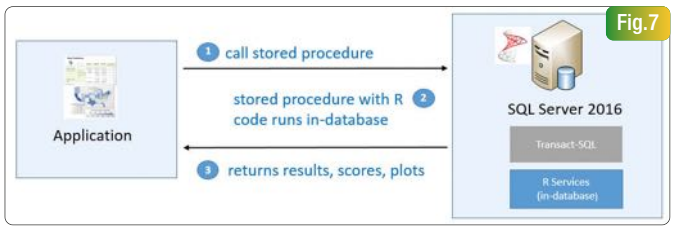


Fig.7

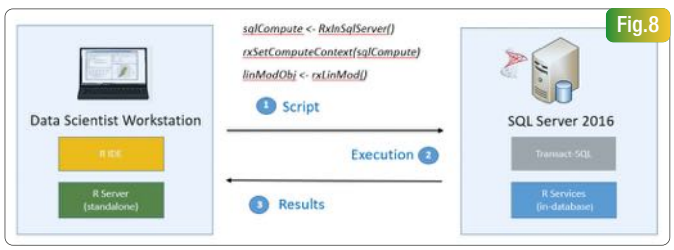


Fig.8

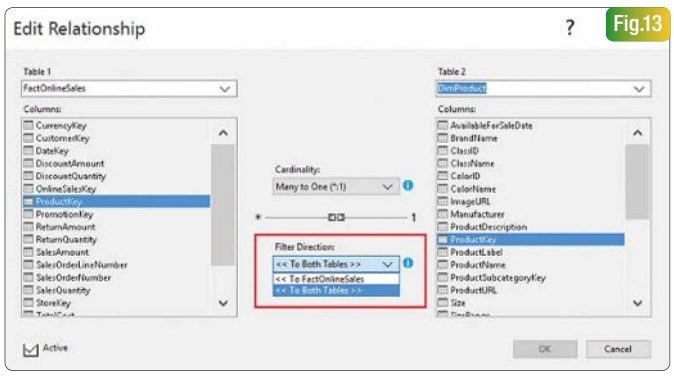


Fig.13

Arrivent aussi des tâches Hadoop pour exécuter du Pig ou du Hive.

Arrivent aussi des tâches Hadoop pour exécuter du Pig ou du Hive. Fig.11.

Dernière nouveauté intéressante : les templates. On peut désormais réutiliser des control flow à travers nos différents packages pour un gain non négligeable en temps de développement et en maintenance. Fig.12.

Pour la partie administration, on peut parler du déploiement incrémental des packages (c'est peu de dire que c'était attendu). Pas mal d'évolution côté SSIS catalog aussi avec le support du AlwaysOn et du Always Encrypted. Egalement un nouveau rôle et la possibilité de paramétrer ses logs. Bref une mise à jour tout de même conséquente pour Integration Services. Pour tout savoir : https://msdn.microsoft.com/en-us/library/bb522534.aspx

Analysis Services

Le moteur OLAP de Microsoft qui se décline en deux versions depuis 2012 avec une version multidimensionnelle et une version tabulaire a également le droit à son lot de nouveautés. Soyons clair, la majorité des évolutions se fait côté tabulaire. On peut néanmoins citer l'arrivée du DBCC côté moteur multidimensionnel, qui fera le bonheur des administrateurs. La majeure partie des nouveautés sur le moteur tabulaire permet en réalité de rattraper des choses déjà existantes en multidimensionnel. On pense, par exemple, aux traductions qui ne sont gérées pour l'instant qu'au niveau des métadonnées, aux display folders qui permettent de regrouper des mesures ou des attributs de dimension dans un répertoire et au parallel processing pour le rafraîchissement des données de nos cubes. Il en est de même au niveau des relations au sein de notre modèle tabulaire, sur lesquelles on peut désormais configurer un filtre bidirectionnel qui permet de créer la relation de type many-to-many d'un modèle multidimensionnel Fig.13 et 14. Dans les nouveautés propres au modèle tabulaire, on peut citer un nouveau langage de script. Jusqu'à maintenant, les deux versions d'Analysis Services partageaient le même langage, le XMLA et plus précisément l'ASSL (Analysis Services Scripting Langage). SQL Server 2016 introduit le TMSL : Tabular Model Scripting Langage. Basé sur du JSON, il gagne en lisibilité, et par extension en maintenance. On pense naturelle-

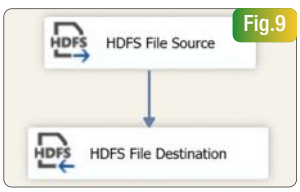


Fig.9

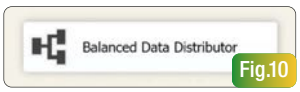


Fig.10

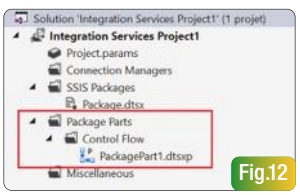


Fig.12

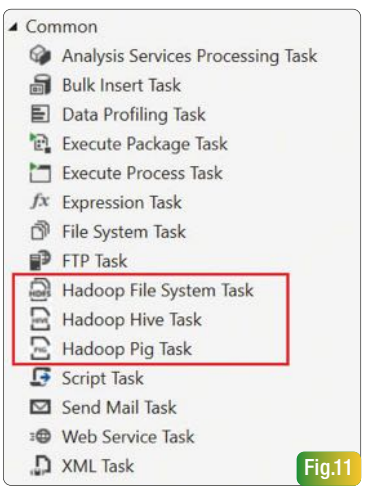


Fig.11

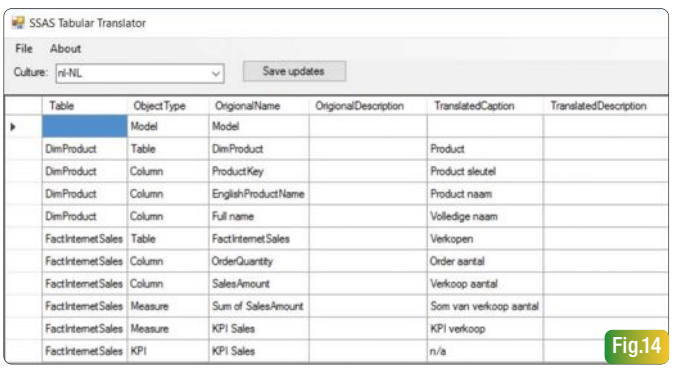


Fig.14

ment à la facilité accrue de manipuler ce type de format pour le travail collaboratif (notamment avec Team Foundation Server) Fig.15. D'un point de vue purement développement, quelques ajouts bienvenus sont à noter :

- Nouvelles fonctions DAX, surtout mathématiques et statistiques ;
- Possibilité de créer des tables calculées ;
- Interface de saisie de formules plus ergonomique.

Pour aller plus loin : https://msdn.microsoft.com/en-us/library/bb522628.aspx

Reporting Services

L'outil de reporting de Microsoft a souvent été délaissé lors des dernières sorties de SQL Server. On attendait donc pas mal de nouveautés pour cette release, en partie parce que Microsoft a racheté Datazen, société spécialisée dans les rapports mobiles, l'année dernière.

L'évolution la plus importante, et elle saute directement aux yeux, est le nouveau portail SSRS : Fig.16.

L'apport de Datazen est directement vérifiable avec l'intégration des KPI (indicateurs performance clé) et des rapports mobiles dans ce portail. Les rapports paginés (paginated reports) correspondent aux rapports SSRS standards. On peut également uploader des fichiers Excel ou des fichiers Power BI (on parle ici des fichiers .pbix).

Il est possible aujourd'hui de personnaliser le portail, notamment aux couleurs de son entreprise de manière simple : il suffit de spécifier un logo et un fichier JSON contenant les couleurs Fig.17.

Les rapports mobiles sont créés à travers **SQL Server Mobile Report Publisher**, qui permet entre autres de choisir un rendu par type de device (PC, tablettes, téléphone) Fig.18.

Si l'on parle de Reporting Microsoft, on pense bien évidemment aussi à Power BI. Reporting Services permet aujourd'hui d'épingler un rapport vers Power BI avec un rafraîchissement paramétrable. Si vous cliquez sur le rapport épinglé, vous êtes alors redirigé vers SSRS.

Il est évident que Microsoft essaie de mettre en place un lien fort entre ses deux solutions de reporting, il n'est donc pas utopiste de penser qu'il sera bientôt possible d'épingler un rapport Power BI sur le portail SSRS.

Dernières nouveautés :

- Le portail bénéficie maintenant d'un rendu HTML5.
- Désormais, on peut également exporter ses rapports au format PowerPoint.
- En termes de développement pur, on gagne deux nouveaux types de graphiques avec les tree map et les sunburst charts.

En conclusion, une vraie évolution de SSRS qui permet de centraliser tout le reporting d'une entreprise au sein d'un même portail.

Pour aller plus loin : https://msdn.microsoft.com/en-us/library/ms170438.aspx

La liste des nouveautés est encore bien longue. On peut citer également des améliorations et nouveautés autour de la brique de MDM de Microsoft (Master Data Services), notamment au niveau de l'add-in Excel mais également sur le cœur de produit.

Et demain ?

Début Mars, Microsoft surprenait tout le monde en annonçant le portage de SQL Server 2016 sur Linux. Ne vous voilons pas la face, il s'agit ici d'une attaque frontale envers Oracle. D'ailleurs Microsoft propose des licences gratuites pour migrer vers SQL Server (évidemment soumises à condition). La disponibilité générale est annoncée pour mi-2017. La question que l'on peut légitimement se poser est la suite que va donner Microsoft à cette version : peut-on espérer une version open source pour faire concurrence à MySQL, l'avenir nous le dira. Pour le moment, nous avons très peu d'informations, nous savons cependant qu'une version preview tourne sur un Linux Ubuntu, ou sous la forme d'un container. Fig.19.

L'avenir passera aussi par les containers qui sont l'avenir de la virtualisation, on pense notamment aux investissements Microsoft vers Docker. Les équipes internes ont récemment rédigé un article dédié au déploiement de SQL Server en mode container. Aucun doute qu'il s'agit de l'évolution logique de la plateforme, et qu'on n'a pas fini d'en entendre parler.

Pour en savoir plus : https://blogs.msdn.microsoft.com/sqlserverstorageengine/2016/03/21/sql-server-in-windows-containers/

Fig.18

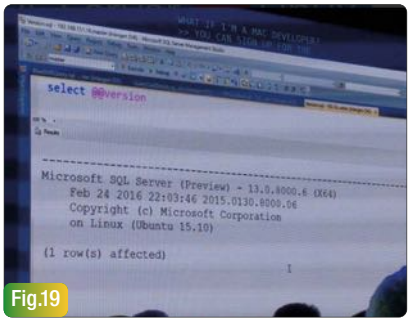


Fig.19

Fig.15

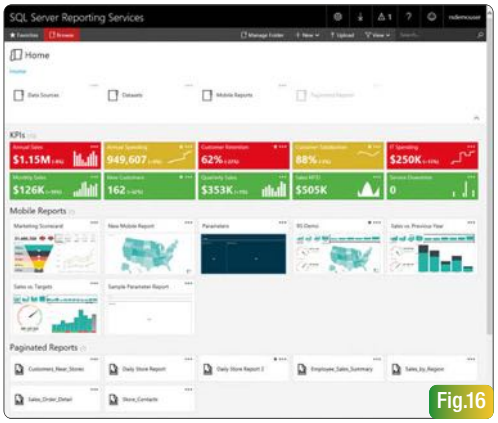


Fig.16

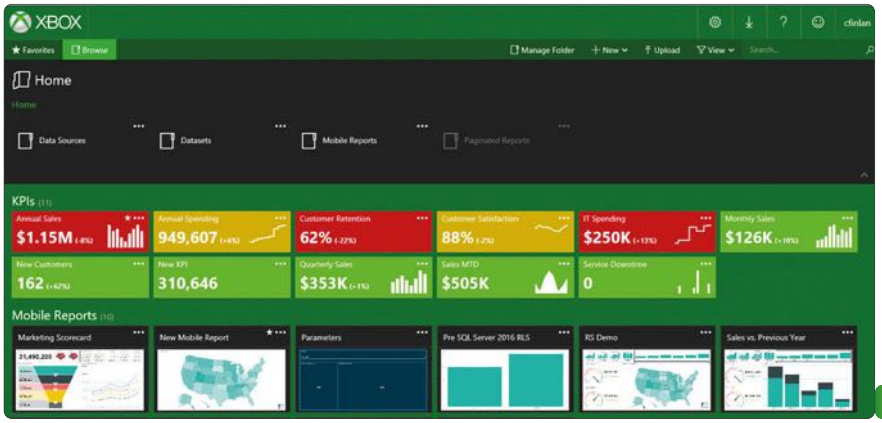


Fig.17

Les microservices, le régime tendance

Aujourd'hui, nombreuses sont les entreprises qui ne seraient rien sans leur SI. Depuis des années, l'informatique est au centre de nos activités professionnelles. C'est pourquoi, mes chers contributeurs, depuis tout ce temps vous me chérissez, MOI, votre application. Je suis votre bébé et vous m'élevez de la meilleure des façons. Pour mon développement, vous êtes très organisés et êtes à mon écoute. Vos méthodes d'éducation sont agiles et vous prenez bien soin de moi en me nourrissant régulièrement. Vous m'entretenez bien et me nettoyez souvent.

Yann Danot
Software gardener Arolla

La complainte d'une application monolithique

Force est de constater que j'ai bien grandi ! Et je dois vous avouer que j'ai mal à chaque nouveau projet. Certains de mes composants sont trop vieux et trop enfouis pour que vous puissiez faire quoi que ce soit. Je commence à douter de moi et j'ai du mal à me débrouiller en production. Je me sens lourde, lente et je ne m'aime plus. Cela fait des années que je m'engraisse et cela commence à se voir. Je vous entends parler de moi comme d'un monstre et certains d'entre vous pensent même à démissionner pour ne plus me voir, tellement il est devenu difficile de me moderniser. Tout s'emmêle en moi et mes entrailles ressemblent à un plat de spaghetti. Mes chers fondateurs, ça ne va plus, je suis devenue un immonde **monolithe**.

Mes copines, réactives, résilientes et scalables (que je jalouse !) m'ont parlé d'un régime très efficace : les **microservices**. Je ne vais ni vous parler d'un nouveau concept, ni d'un nouveau framework à la mode. L'approche microservices n'est pas vraiment innovante dans ses principes et ses racines remontent à loin. On peut par exemple voir que l'architecture Linux a été conçue avec cette approche. Cependant, il me semble que la mise en place de microservices va vous guider de façon naturelle dans le respect des principes SOLID, particulièrement sur le côté SRP, mais je vais y revenir plus tard.

Un régime payant

Pour faire bref, l'architecture en microservices est une approche pour développer une application en un ensemble de petits services, chacun s'exécutant dans son propre processus et communiquant avec les autres via des mécanismes légers (REST, AMQP, etc...). Ces services doivent être caractérisés par un besoin business ou technique (envoi de notifications par ex.), et déployables de façon indépendante et un minimum automatisés. Il sera possible de les écrire en différents langages adaptés à chaque besoin. De même, les services les plus sollicités pourront être «scalés» de manière plus aisée. Les ressources seront allouées spécifiquement pour ce qui est indispensable au bon fonctionnement du système. Mais laissez-moi expliciter plus en détails pourquoi l'approche microservices fera de moi une bien meilleure collaboratrice !

Des temps de développement diminués

La raison qui m'amène à penser que cette architecture serait meilleure est principalement l'indépendance des éléments. En effet, aujourd'hui, chaque modification de votre part amène une compilation de tous mes organes, et le chemin pour la voir arriver en production est long et fastidieux. Je suis un bloc indivisible, alors que si j'étais décomposée en services, un changement quelque part n'entraînerait que la compilation et le déploiement d'un seul de mes membres. Plus un service serait unitaire, plus le temps perdu à compiler et à déployer serait court.

Cependant, certains développements entraîneraient des modifications d'interface et vous demanderaient un peu de coordination. Il existe néanmoins des mécanismes permettant de minimiser ces effets, la communication par « messaging » par exemple, mais un bon découpage devrait atténuer la cohésion et les liens entre services. Tout changement d'une interface devrait vous faire réagir, et devrait vous amener à vous poser des questions et peut-être à revoir mon découpage. Les évolutions seront plus simples à suivre qu'actuellement.

Des responsabilités isolées

Chaque service doit avoir une responsabilité identifiée. Ceci réduira la complexité et améliorera la vision du métier de l'entreprise. Cet aspect est essentiel au bon déroulement d'une mise en place de microservices : si un composant correspond à plusieurs domaines, vous allez vous compliquer la tâche et ne pas voir de réels gains. Le respect du principe SRP est fondamental. Il faut être le plus granulaire possible sans pour autant tomber dans l'obsession de créer un nouveau service pour tout et n'importe quoi. Toute la difficulté résidera dans le découpage, comment trouver le juste milieu. Cette question reviendra sans cesse et il n'existe malheureusement pas de recette miracle. Nous verrons un peu plus tard quels sont les défis qui vous attendent lors d'une refonte en microservices.

Une fluidité accrue au quotidien

Comme nous l'avons vu précédemment, un service s'exécute dans un processus indépendant. Du coup, tout changement dans le code d'un service implique le déploiement de celui-ci uniquement. Chaque service a son propre cycle de vie et ceci entraînera une meilleure fluidité dans vos réalisations. Grâce à cette isolation, un problème (retard des développements, changement fonctionnel, etc.) lors de l'implémentation d'un projet n'entraînera pas le blocage de la réalisation d'une fonctionnalité différente. Si vous isolez en plus de la couche applicative la couche data, les données relatives à un service pourront être mises à jour, migrées ou traitées sans craindre d'effets de bord. Chaque service va également avoir sa propre durée de vie, il sera plus pratique de supprimer une fonctionnalité si celle-ci est isolée et n'interagit que très peu avec le reste de l'application. En effet, si les communications sont faites de façon assez souple (messaging, push d'event, etc...), il suffira de le débrancher.

La mise en place d'A/B Test sera fluidifiée. Vous pourrez essayer plus de choses pour comprendre le comportement de mes utilisateurs. Il suffira de déployer plusieurs versions différentes d'un service et d'ajuster au fil de l'eau. En y réfléchissant bien, la

composition en microservices vous permettra d'être beaucoup plus agiles et vous pourrez collecter le feedback de nos clients plus rapidement.

La scalabilité facilitée

Il est assez facile d'imaginer le gain d'une architecture en microservices pour la scalabilité d'une application. Aujourd'hui, la production est conçue de manière horizontale. Pour augmenter mes capacités, vous me divisez en plusieurs instances et me faites tourner sur plusieurs serveurs derrière un load-balancer. Les microservices étant des processus isolés, l'augmentation ou la diminution des ressources allouées par service pourra se faire de manière indépendante. Aujourd'hui, vous ne vous en apercevez peut-être pas, mais mes performances sont médiocres et ceci est dû à un petit nombre de classes qui, à mon sens, devraient être isolées et scalées pour éviter l'effet bottleneck. La scalabilité d'un monolithe est plus compliquée car scaler un service, c'est scaler l'ensemble. Certains composants sont plus complexes que d'autres et peuvent rendre impossible la scalabilité de l'ensemble. C'est le principal problème qui m'a fait prendre conscience du bien-fondé d'une structure en microservices.

Un métier mieux maîtrisé

Plus besoin de chercher pendant des heures pour trouver les informations relatives à une fonctionnalité, la taille limitée et l'isolation des services vous permettront d'être plus précis dans vos recherches et de limiter le scope, puisque toutes les informations liées à un domaine seront englobées dans un composant indépendant.

Possibilité d'externalisation

Aujourd'hui, notre société est « au top », vous avez développé des fonctionnalités géniales qui, à mon sens, pourraient être réutilisées par d'autres. Vous avez une expertise dans un domaine clé et je pense que cette approche pourrait être une source de revenus supplémentaire si vous décidiez de vendre l'utilisation d'un service en particulier.

Un futur prometteur

L'innovation se fera sans risque : un composant à changer ? Même pas peur ! Du fait de leur taille raisonnée, les microservices sont plus maintenables. La mise à jour des différents frameworks, par exemple, sera beaucoup moins douloureuse car les impacts seront bien mieux maîtrisés. Finis les chantiers de six mois pour combler une faille de sécurité dont vous n'êtes même pas responsable. Vous pourrez explorer de nouvelles pistes technologiques sans remettre en cause l'existant, l'échec sera enfin permis et vos idées pourront voir le jour.

Fig.1

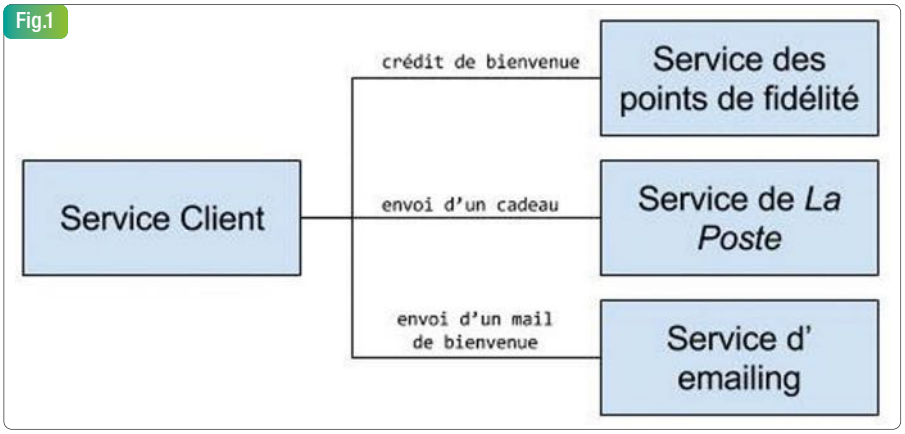
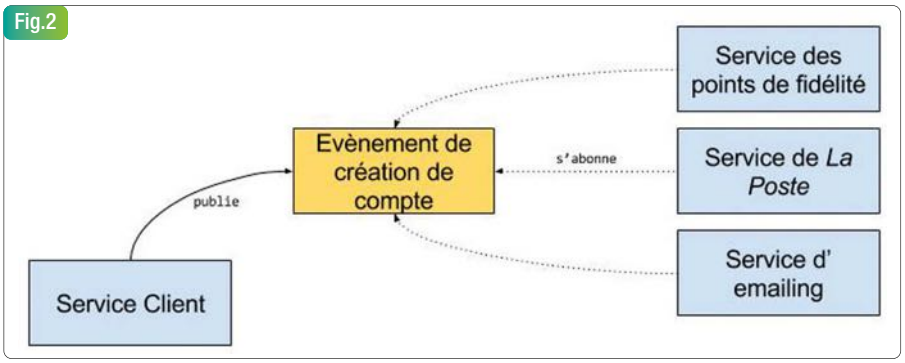


Fig.2



Comment devient-on un micro-développeur ?

Mes chers collaborateurs, je vous ai présenté une possibilité de me rendre heureuse. Je pense sincèrement qu'une telle refonte est nécessaire, tant je suis devenue lourde. Cependant, pour être de bons micro-développeurs, vous allez devoir vous familiariser avec certaines notions. La principale difficulté réside dans les communications inter-services et voici quelques recommandations pour mener à bien votre mission :

- Evitez à tout prix les intégrations par base de données. Si plusieurs microservices communiquent avec la même base de données, vous n'aurez plus du tout d'indépendance. Chaque modification de schéma impliquera des modifications de tous les µ-services y étant liés.
- Privilégiez les appels REST sur les mécanismes de RPC (Remote Procedure Call). Les intégrations de services via REST sur HTTP sont un bon choix pour débuter.
- N'exposez pas les détails de vos implémentations dans vos APIs, gardez vos APIs technologiquement agnostiques. Il faut éviter les intégrations qui imposent la technologie ou le langage à utiliser.
- Un microservice doit être simple d'utilisation.
- Préférez toujours la chorégraphie à l'orchestration de service. Prenons un exemple simple : sur notre site e-commerce, lors d'une création de compte, nous souhaitons créditer des points de fidélité au client, lui envoyer un cadeau par la poste et lui envoyer un mail de bienvenue.

L'**orchestration** consisterait en : Fig.1.
La **chorégraphie** en : Fig.2.

- Optez pour une collaboration asynchrone par évènement. Les différents brokers de messages comme Apache QPid, OpenAMQ, Windows Azure Service Bus ou encore RabbitMQ vont vous permettre de publier et consommer des évènements assez simplement. Seulement il faut savoir qu'il y aura un coût supplémentaire pour la mise en place et la maintenance d'une telle solution.
- Publiez tous vos messages sur un seul bus, lisible par tout le monde.
- Il faut intégrer dans votre travail le fait que chaque appel à un microservice peut échouer. Il va donc falloir être beaucoup plus attentif à la gestion d'erreur.
- Privilégiez toujours l'**autonomie** plutôt que l'**autorité**.

Un service autonome doit pouvoir assurer sa responsabilité seul. Dans le cas d'une

récupération de données, une configuration en autonomie pourrait être : Fig.3.
Ici, le µ-service appelant conserve une copie des données retournées par le µ-service appelé, dans ce cas le µ-Service 1 devient autonome après le premier appel.
Alors qu'une configuration **autoritaire** serait : Fig.4.

Ici le µ-service appelant utilise les données retournées par le µ-service appelé pour renvoyer un résultat à son appelant. Dans ce cas, le µ-Service 1 est dépendant du bon fonctionnement du µ-service 2.
Lorsqu'un service a l'autorité sur un autre, les données sont toujours à jour, alors que lorsqu'il est autonome, il existe une latence. Cependant, dans un laps de temps plus ou moins court, les données seront mises à jour. Notez tout de même que pour rester autonome, la gestion par évènement est parfaite.

- Prenez en compte que les données ne sont pas tout le temps à jour dans des systèmes distribués : on parle "**d'eventual consistency**".
- Pensez à des mécanismes de compensation pour contrer les problèmes d'eventual consistency.
- Acceptez la panne de vos services, mais prévenez haut et fort tous leurs utilisateurs.

Les développeurs de Netflix sont partis du principe que la panne surviendrait forcément. Ils ont créé un outil la provocant volontairement pour tester en production la résilience de leur système, le Chaos Monkey. Ils ont aussi mis à disposition une librairie très utile pour gérer les interactions dans un système distribué : Hystrix.

- Supervisez ! Le monitoring va devenir beaucoup plus complexe dans une architecture distribuée.
  - Monitorez les temps de réponse utilisateurs. Le RUM (Real User Monitoring) est un outil idéal, il existe des solutions clés en main sur le marché comme New Relic, mPulse de SOASTA ou encore Pingdom.
  - Utilisez des Time Series Database qui permettent de stocker des métriques à intervalle régulier, comme InfluxDB (rétrocompatible avec Graphite) ou OpenTSDB.
  - Traquez toutes les communications, Hystrix fournit ce service nativement.
  - Standardisez vos logs et trouvez un moyen de les agréger.
  - Utilisez des identifiants pour les différentes requêtes (correlation ids).
- Formez des équipes pluridisciplinaires pour être autonome.

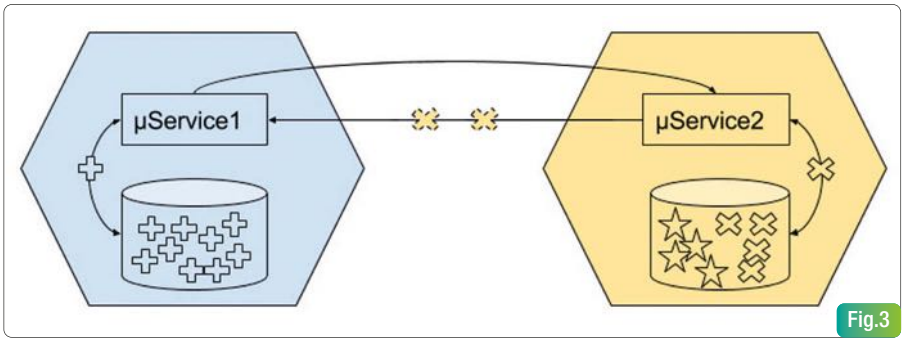


Fig.3

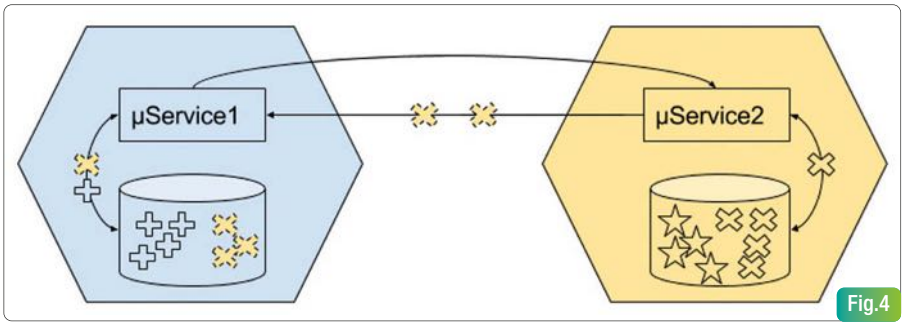


Fig.4

- Découpez finement mais intelligemment vos services. Les microservices doivent pouvoir être réécrits intégralement en peu de temps. De ce fait, ils doivent être de très petite taille et compréhensibles rapidement.

- Renseignez-vous sur l'offre du marché. Il existe plusieurs outils qui vous aideront, en voici une liste non exhaustive, bien sûr :
  - **Vert.x** : boîte à outil polyglotte facilitant l'implémentation d'application réactive. Utilisée notamment par RedHat, Bosch, Cyanogen, Groupon...
  - **Hystrix** : librairie (portée dans plusieurs langages) développée par les ingénieurs de Netflix, permettant d'être facilement tolérant à la panne et notamment d'éviter les effets domino.
  - **Docker** : docker va permettre de "conteneuriser" vos microservices, de les tester et de les déployer plus facilement.
  - **Apache Kafka** vous aidera si vous partez sur une solution event-sourcing.
  - **Apache Flume** vous permettra de collecter et d'agréger les logs de vos différents services.
  - **Microsoft Azure Service Fabric** est une solution complète pour orchestrer des services distribués.

Ces différentes recommandations vous serviront certainement à la mise en place d'une architecture en microservices. Mais vous l'aurez compris, ceci est une philosophie à adopter et pas une suite d'outils à utiliser. Ne tombez pas dans le piège consistant à vouloir profiter à tout prix de tous les avantages d'une telle architecture ! Restez pragmatiques en vous focalisant sur les points qui répondent à nos problématiques.

Modélisation d'un microservice

Il est indispensable de connaître certaines notions avant de se lancer dans le découpage de son application. Nous allons les aborder rapidement ici, mais je vous invite à vous documenter davantage pour bien comprendre les enjeux.

Couplage faible

Quand deux services sont à "couplage faible", un changement dans l'un n'entraîne pas nécessairement de changement dans l'autre. L'essence même d'un microservice est d'être capable d'y effectuer une modification et de la déployer sans avoir à modifier le reste de l'application. Un service faiblement couplé ne connaît que le strict minimum des services avec lesquels il interagit. Il faut aussi faire attention à la fréquence d'échanges entre deux services, parce qu'en plus des potentiels problèmes de performance, les communications récurrentes peuvent mener à un couplage fort. Le couplage concerne donc les contrats entre services, d'un point de vue schéma (JSON par exemple) et temporel (disponibilité à tout moment au runtime).

Cohésion forte

Nous voulons regrouper toutes les fonctionnalités liées entre elles au même

endroit car si nous voulons changer un comportement, nous devons être capables de le changer à un seul endroit. Ceci dans le but de déployer de façon indépendante tous nos services.
Je voulais attirer votre attention sur ces deux concepts fondamentaux à respecter pour faire des microservices. Ces notions sont très présentes dans le monde de la programmation objet mais il est indispensable de les rappeler et de les garder en tête lors d'un découpage en microservices. Nous voulons donc trouver des "limites" ou des "frontières" dans notre domaine métier qui vont nous assurer que les fonctionnalités liées se trouvent au même endroit et qu'elles ne sont pas fortement couplées avec d'autres.

Bounded Context

Le livre d'Eric Evans « Domain-Driven Design » explique comment créer des systèmes qui modélisent les différents domaines du métier d'une entreprise. Ce livre propose des principes comme l'utilisation par tous les acteurs d'un même vocabulaire (ubiquitous language), l'abstraction de technologies d'un point de vue métier au travers d'un Repository, et bien d'autres conseils utiles.
Le concept qui va le plus nous importer ici est celui des **Bounded Contexts**, ou Contextes Bornés en Français. L'idée est que n'importe quel domaine peut se décrire comme un ensemble d'interactions de plusieurs sous-domaines. Ces sous-domaines sont limités de façon explicite et possèdent des caractéristiques propres :

- Un vocabulaire propre à chacun ;
- Des services exposés ;
- Des fonctionnalités homogènes ;
- Des notions métier bien comprises et clairement modélisées.

Certaines de ces caractéristiques vont être exposées à l'extérieur par l'intermédiaire d'une sorte de "frontière". C'est précisément cette "borne" (boundary) qui va renforcer l'indépendance des "Bounded Context" et assurer les différents échanges. Eric Evans utilise une analogie avec les cellules, où "les cellules existent car leurs membranes définissent ce qui est dedans, ce qui est dehors et ce qui peut y pénétrer" Fig.5.

En conclusion, si nos microservices représentent littéralement ces Bounded Contexts, nous avons de grandes chances d'assurer le couplage faible et la cohésion forte. Cependant, un microservice doit être, par définition, un service de très petite taille. Nous devons être capables de le réécrire dans son intégralité en peu de temps. Cette notion étant propre à chaque entreprise, je vous laisse définir la durée maximale, mais l'essentiel est que la réécriture d'un microservice ne doit pas être trop engageante : par exemple, elle doit pouvoir se faire sans signature du management.
La première étape est donc d'identifier les différents Bounded Contexts de notre métier, pour ensuite les implémenter sous forme d'un ou plusieurs microservices. Identifier des Bounded Contexts n'est pas chose évidente, et ce n'est pas l'objet de mon propos. Il s'agit avant tout de penser aux fonctionnalités métier et à qui elles sont destinées. Il ne s'agit surtout pas de découper de façon naïve : "tout ce qui parle de client" vs. "tout ce qui parle de commande", mais plutôt de découper par zone fonctionnelle cohérente : "tout ce qui parle de facturation" vs. "tout ce qui parle de réductions".

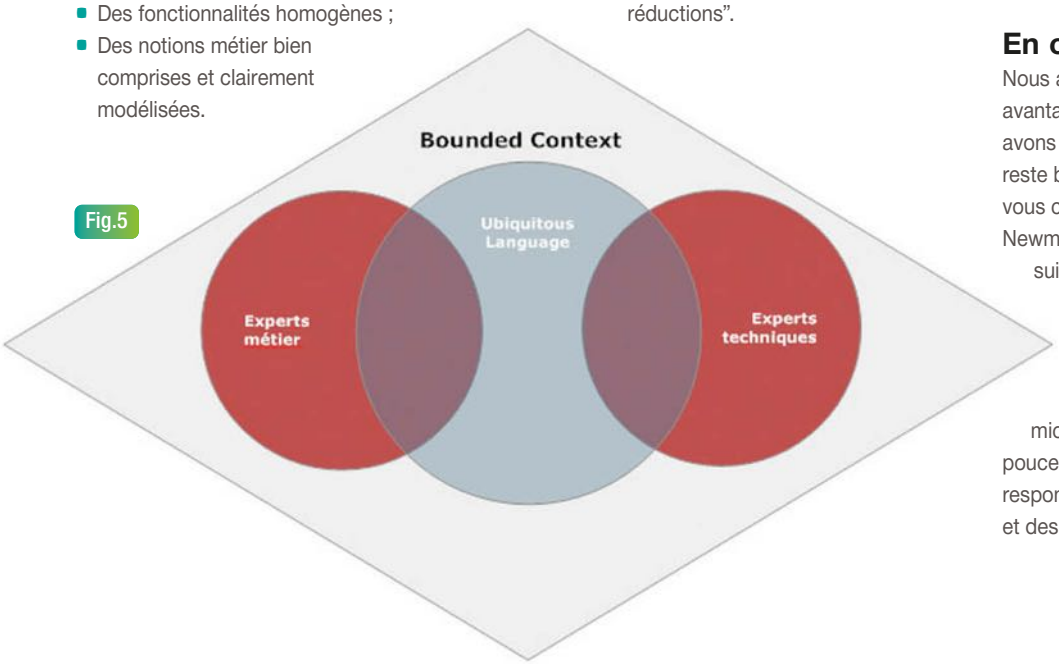


Fig.5

Regarder de près les comportements métiers par rapport aux données qu'ils utilisent peut donner des pistes, tout comme poser la question "A qui s'adresse cette zone fonctionnelle : au directeur marketing ? à la responsable de prévention des fraudes? au département livraison ?
Un µ-service ne devrait donc pas dépasser le périmètre fonctionnel d'un Bounded Context, au plus gros. Mais rien n'oblige un microservice seul à couvrir tout un Bounded Context, qui peut alors être couvert par plusieurs microservices de granularité plus petite.
La collaboration de plusieurs bounded contexts implique aussi des relations, certes réduites au minimum, mais néanmoins bien présentes. La question est alors de définir quel service impose sa vision du monde à son partenaire, à moins que tous deux n'adhèrent à un contrat tierce. Toutes ces questions sont essentielles à une co-évolution heureuse du système dans son ensemble, et Domain-Driven Design apporte l'outillage méthodologique du Context Mapping.
Même si vous n'allez pas jusqu'au bout de la démarche, avec un process par microservice, vous pouvez néanmoins commencer par soigner le découpage et maximiser l'indépendance des modules entre eux au sein d'un même process, dans une approche qu'on peut qualifier de "microservices-ready". L'effort principal, c'est-à-dire le découpage logique du code en termes de dépendances, contrats sous forme d'interfaces et de structures de données est ainsi déjà effectué, sans pour autant payer immédiatement le coût de la distribution physique et du monitoring d'une grande quantité de services.

En conclusion

Nous avons vu ensemble les différents avantages d'une solution microservices et avons évoqué le travail que cela implique. Il reste beaucoup de domaines à couvrir et je vous conseille vivement le livre de Sam Newman : "Building Microservices" dont je me suis beaucoup inspiré pour écrire cet article.
Certains détracteurs diront (à juste titre), qu'il n'y a rien de nouveau dans cette approche, mais l'architecture en microservices nous donne un coup de pouce pour toujours mieux isoler les responsabilités de chacune des fonctionnalités et des composants d'une application.

Création d'un cluster Docker Swarm avec Azure Container Service

Azure Container Service (ACS) est un service qui permet de simplifier le déploiement et la gestion d'un cluster de machines qui permettent l'hébergement d'applications à base de conteneurs.

Julien CORIOLAND
@jcorioland
Evangéliste Technique Azure
Microsoft France

Azure Container Service prend en charge deux types de clusters :

- Basés sur DC/OS (issu du projet Apache Mesos), avec la possibilité d'utiliser des frameworks comme Marathon (long-running) ou Chronos (cron) pour lancer des applications à l'intérieur de conteneur Docker, mais pas uniquement !
- Basés sur Docker Swarm, l'outil de « clusterisation » créé par Docker, qui permet de faire passer un pool de machines hôtes comme un seul hôte, c'est-à-dire utiliser les commandes Docker traditionnelles, mais distribuées sur un cluster.

Dans cet article, vous allez découvrir comment utiliser Azure Container Service pour déployer en quelques clics un cluster basé sur Docker Swarm dans Microsoft Azure.

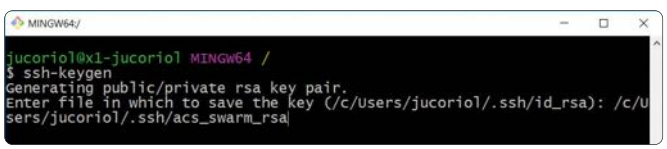
Génération d'une clé SSH RSA

Microsoft travaille actuellement au support des conteneurs pour Windows Server 2016, la future version du système d'exploitation serveur. Cependant, à ce jour, Azure Container Service ne supporte que des machines virtuelles Linux, aussi il est nécessaire de créer une clé SSH RSA pour se connecter au cluster, une fois celui-ci créé. Cette opération ne nécessite que quelques étapes.

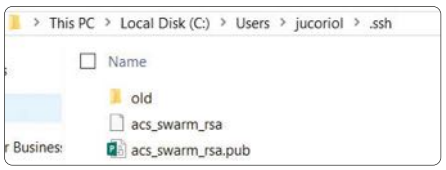
Si vous êtes utilisateur Linux ou OS X, vous devez être plutôt familier avec ce type d'opération, et vous pourrez trouver une documentation détaillée sur le site de GitHub : https://help.github.com/articles/generating-an-ssh-key/

Sous Windows, le plus simple est de télécharger et installer Git pour Windows (https://git-for-windows.github.io/), qui vous installera tout le nécessaire pour générer une clé SSH, mais aussi vous connecter en SSH à un serveur ou créer un tunnel SSH. Une fois installé, lancez l'outil Git Bash, tapez **ssh-keygen** et tapez Entrée :

Par défaut, l'outil **ssh-keygen** créé un couple de clé publique/privée dans le dossier .ssh de votre profil utilisateur. Les noms par défaut pour ces fichiers sont id_rsa (clé privée) et id_rsa.pub (clé publique) mais il est tout à fait possible d'utiliser le nom de votre choix pour ces fichiers (dans le cas où vous souhaiteriez créer plusieurs clés, pour différents services, par exemple) :



Appuyez sur la touche Entrée et entrez une « passphrase » lorsque cela vous est demandé. Celle-ci sera utilisée pour protéger vos clés et elle vous sera demandée à chaque connexion. Une fois l'opération terminée, vous devez retrouver les deux fichiers à l'emplacement souhaité :



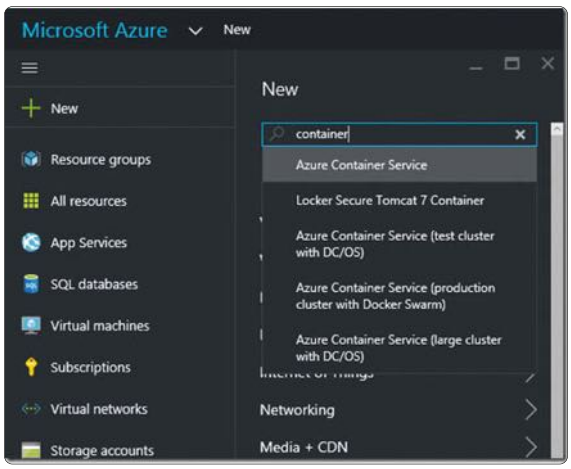
Vous êtes maintenant prêt pour créer votre premier cluster Azure Container Service !

Note : si vous êtes sur une des dernières build Insider de Windows 10, la fonctionnalité Bash on Ubuntu on Windows vous permet d'utiliser directement les commandes SSH, sans passer par l'outil Git Bash.

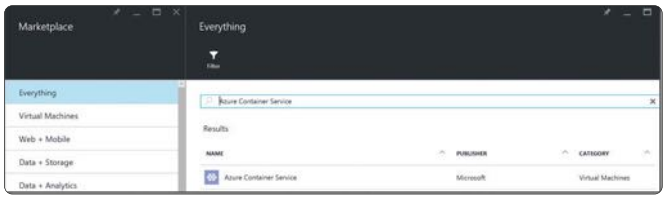
Création d'un cluster Docker Swarm

Il existe plusieurs méthodes pour créer un nouveau cluster Docker Swarm avec Azure Container Service : via le portail d'administration Azure, Azure CLI ou PowerShell. Dans cet article, nous allons utiliser le portail Azure. Connectez-vous à http://portal.azure.com et authentifiez-vous avec votre compte Azure.

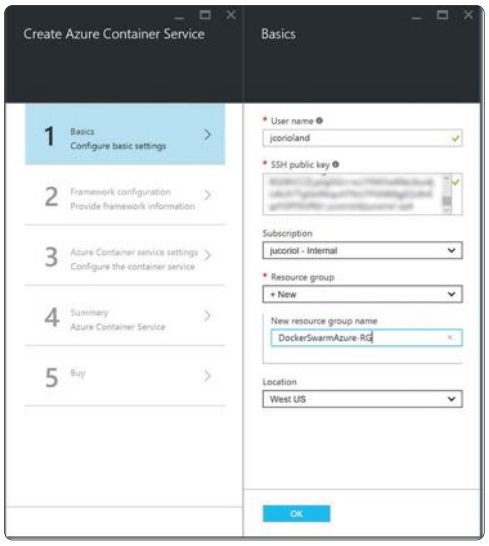
Cliquez sur le bouton + NOUVEAU en haut à gauche et recherchez « container » dans la zone de recherche :



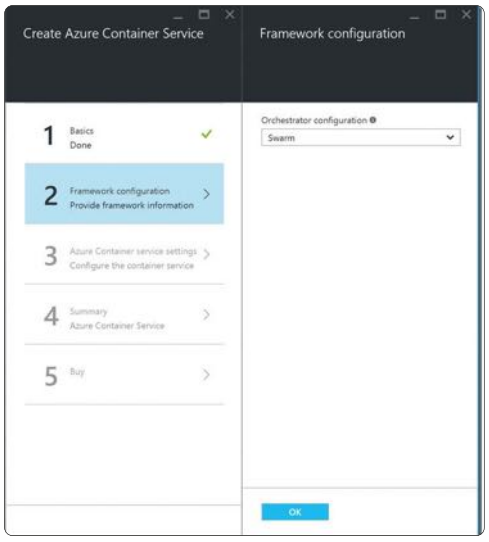
Cliquez sur Azure Container Service, puis sur la première ligne de résultat :



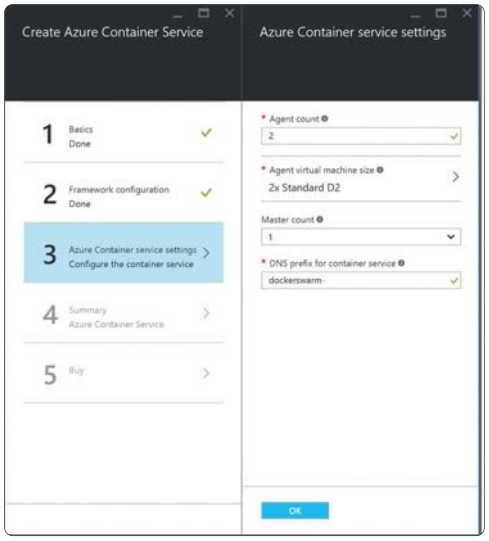
L'assistant de création du cluster s'ouvre alors. La première étape vous permet de configurer le nom d'utilisateur administrateur du cluster, d'entrer votre clé publique SSH générée plus tôt et de choisir le groupe de ressources et la localisation de votre nouveau cluster (il est fortement recommandé de créer un nouveau groupe de ressources) :



Cliquez sur OK pour passer à l'étape 2 dans laquelle vous pourrez choisir l'orchestrateur à utiliser : Docker Swarm ou DC/OS. Dans le cas présent, choisissez Docker Swarm, et cliquez sur OK.

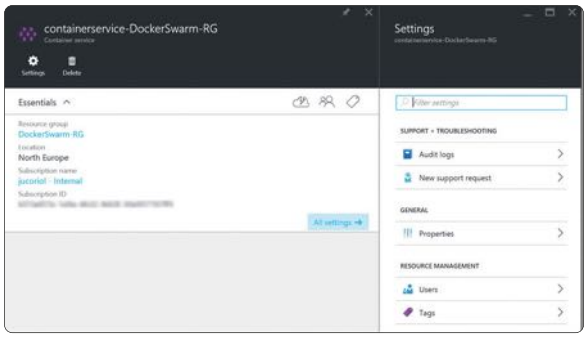


L'étape suivante vous permet de configurer le nombre de machines master et le nombre de nœuds que vous souhaitez pour votre cluster, ainsi que la taille de ces machines. Vous devez également fournir un préfix de DNS qui sera appliqué à toutes les ressources créées.

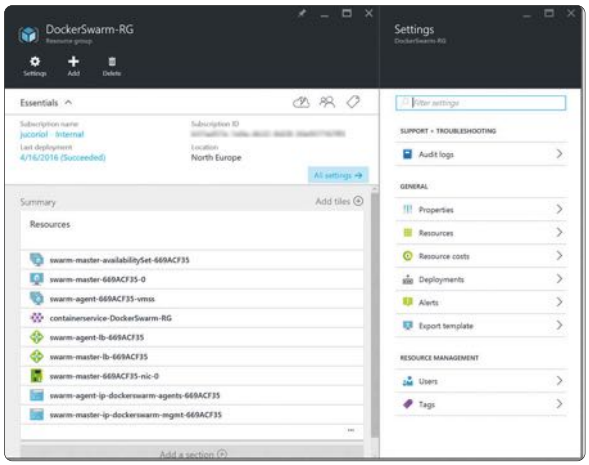


Cliquez sur OK pour passer à l'étape de validation, puis à l'étape d'acceptation des conditions pour lancer la création du cluster. En fonction du nombre de machines que vous avez paramétré, la création peut prendre un peu de temps.

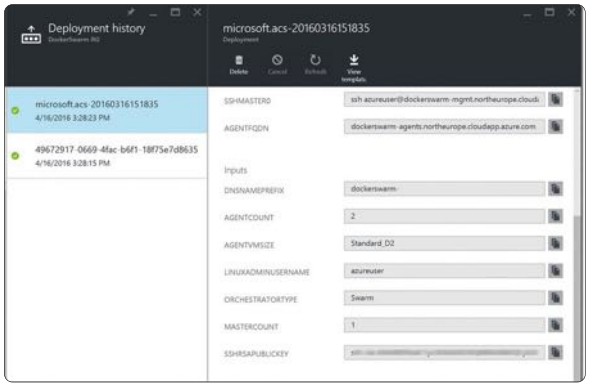
Une fois terminé, vous pouvez accéder à votre nouvelle instance d'Azure Container Service :



Cliquez sur le nom du groupe de ressource en haut à gauche pour accéder à celui-ci. Vous pouvez alors visualiser toutes les ressources qui ont été créées :



En cliquant sur le déploiement, vous pourrez accéder aux valeurs de sortie pour récupérer la commande vous permettant d'accéder au master en SSH



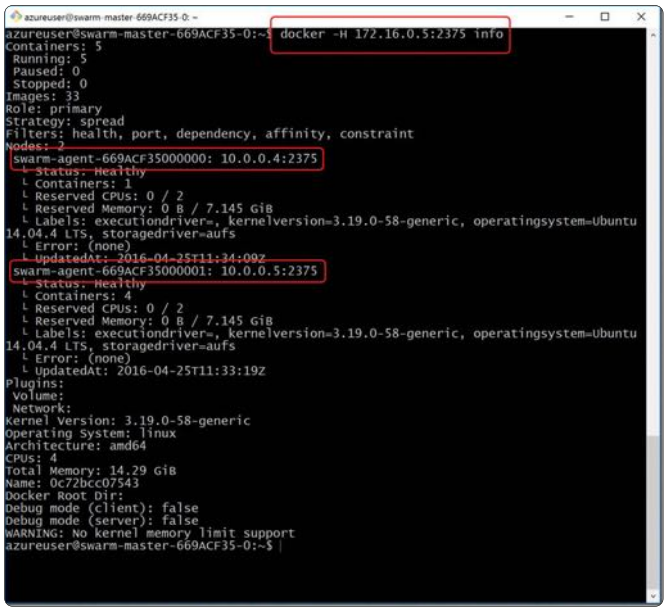
Connexion au master Swarm

Il est maintenant très simple d'accéder au cluster, en se connectant en SSH à l'un des master Swarm qui a été déployé. Si vous êtes sur Linux ou Mac OS, ouvrez un terminal. Sous Windows, vous pouvez continuer à utiliser Git Bash et utilisez l'outil SSH comme ci-dessous :

NB : Si vous avez choisi de personnaliser le nom ou l'emplacement de la clé privée que vous avez générée à l'étape 1, il est possible de préciser ce fichier grâce à l'option « -i ».

Lorsque cela vous est demandé, entrez votre « passphrase ». Vous devez alors être connecté au master Swarm qui écoute sur le port 2200 :

L'adresse IP locale du premier master Swarm est 172.16.0.5 et l'agent Docker écoute sur le port 2375 (toujours cf. schéma du cluster plus haut), il est donc possible de récupérer les informations liées au cluster à l'aide de la commande Docker info, comme dans ce qui suit :



Déploiement d'un premier conteneur Docker

Dans cette dernière étape, je vous propose de déployer un premier conteneur Docker, basé sur l'image officielle Nginx disponible dans le Docker Hub Repository.

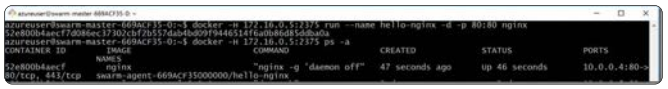
Pour démarrer un conteneur basé sur cette image, il suffit d'exécuter la commande suivante sur le master Swarm :

```
docker -H 172.16.0.5:2375 run --name hello-nginx -d -p 80:80 nginx
```

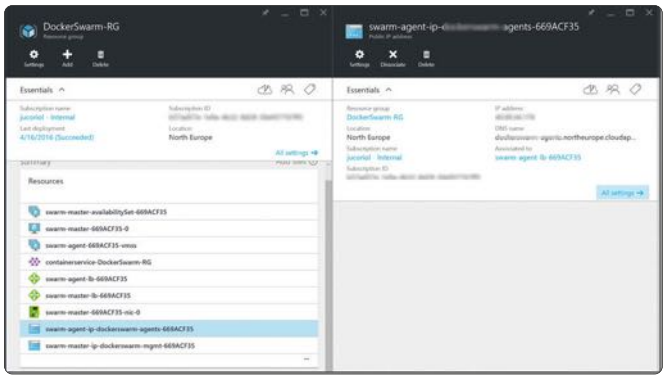
NB : l'opération peut prendre quelques instants la première fois, le temps que l'image soit téléchargée.

Vous pouvez ensuite valider que le conteneur est bien en cours d'exécution, à l'aide de la commande suivante :

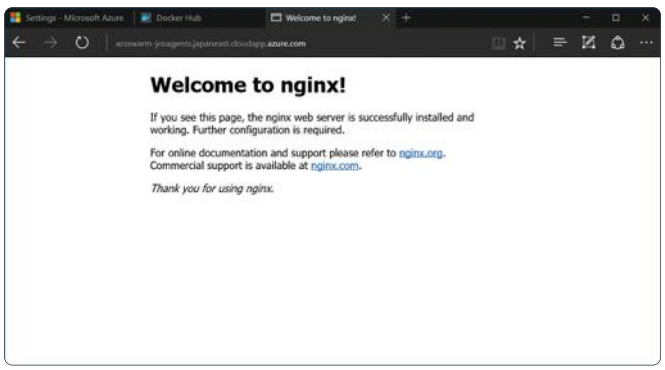
```
docker -H 172.16.0.5:2375 ps –a
```



Enfin, il ne reste plus qu'à vous connecter sur le DNS associé au load balancer des agents Swarm pour valider le bon fonctionnement du serveur Nginx. Pour récupérer l'adresse, sélectionnez l'IP publique liée aux agents dans le groupe de ressources, depuis le portail Azure :



Parcourez l'adresse disponible dans le volet des propriétés avec votre navigateur préféré. Vous devriez retrouver l'interface d'accueil de Nginx :



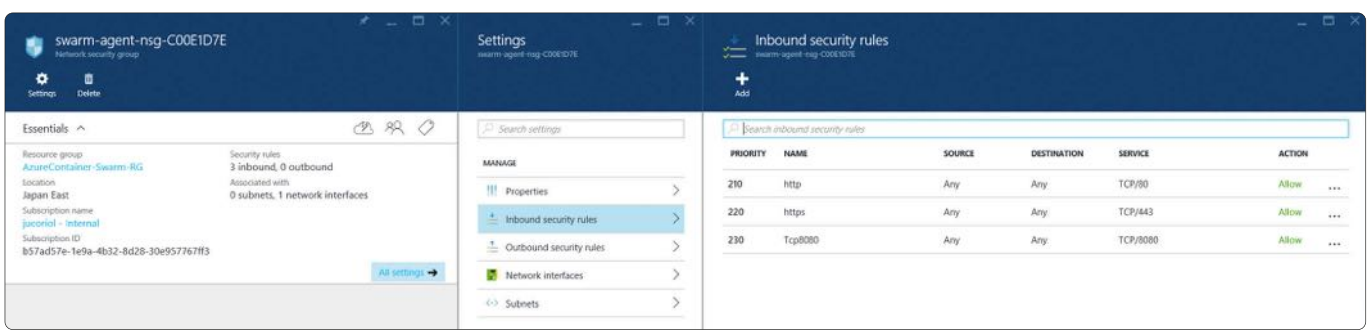
Si vous souhaitez exposer le serveur sur un autre port, il est possible de configurer les règles d'entrée au niveau du groupe de sécurité lié aux agents (par défaut, le port 80 est autorisé).

Conclusion

Dans cet article, nous avons vu comment il est possible de déployer en quelques clics un cluster Azure Container Service basé sur Docker Swarm dans Microsoft Azure, puis de s'y connecter en SSH et d'y déployer un premier conteneur Docker ! L'intérêt de Docker Swarm étant, comme évoqué en introduction, de pouvoir considérer un ensemble de machines comme un seul hôte Docker (virtuel) et d'utiliser les commandes Docker traditionnelles pour le piloter.

Si vous souhaitez déployer une version utilisant DC/OS, il suffit juste de le choisir à l'étape 2, mais le reste se passe de la même manière !

Bons déploiements !



Les solutions de communication inter-applications du SDK UWP

Avec Windows 10, les développeurs ont accès à un nouveau SDK. Nous commençons à connaître les diverses fonctionnalités que celui-ci nous offre vis-à-vis de l'ancien SDK 8.1, mais cet article se concentre sur un aspect en particulier : la communication entre les applications au sein d'un même appareil (PC, téléphone, etc.). Nous abordons ce sujet car le concept d'applications qui communiquent entre elles est très intéressant d'un point de vue client, mais nous pensons surtout aux applications dans un contexte LOB (Line of Business).

Jonathan Antoine
http://blogs.infinitesquare.com/b/jonathan
Daniel Djordjevic
http://blogs.infinitesquare.com/b/ddjordjevic
Consultants **Infinite Square**

Le modèle de bac à sable des applications Windows Store

Dans un premier temps, il est nécessaire de rappeler que les applications Windows Store fonctionnent dans un modèle de « bac à sable », ou plus connu sous le terme anglais « sandbox ».

La sécurité a toujours été une problématique majeure dans le développement et l'univers dans lequel nous sommes. C'est en partie pour ces raisons que les applications Windows Store ont leur propre espace virtuel (la sandbox), ce qui veut dire que, quoiqu'il arrive, les autres applications ou l'OS ne devraient pas être impactés.

La conséquence de ce mécanisme est que les applications n'ont pas d'accès direct aux données des autres applications qui ne sont pas dans la même sandbox. Cette limitation apparaît aussi au niveau des processus, on parle de « process isolation », et donc la communication inter-processus est bloquée entre les différentes applications Windows Store. Cependant les applications vont pouvoir communiquer avec l'OS via des APIs ou bien des brokers, puis entre elles grâce à des contrats Fig.1.

Les possibilités du Framework Windows 8.1

Avec le SDK 8.1, nous pouvions utiliser l'association de protocoles. Concrètement, on lie une application à un protocole, une url, et lorsqu'on accède à l'url depuis une application ou notre explorateur, cela nous permet d'activer l'application associée tout en lui passant des arguments.

```
var uri = new Uri("InfiniteSquare://123456789/ernsteinSugar", UriKind.Absolute);
Windows.System.Launcher.LaunchUriAsync(uri);
```

Les limitations de cette solution sont notamment les suivantes :

- Impossible de passer beaucoup de données ;
- Le protocole peut être le même pour plusieurs applications, et l'utilisateur peut choisir laquelle il veut utiliser, on perd donc du contrôle ;
- Il n'y a pas de moyen built-in pour savoir si l'application cible a fini le traitement de notre appel ;
- L'application cible ne peut pas renvoyer de données en retour facilement. Il faudrait implémenter un nouveau protocole pour que l'application cible puisse rappeler l'application source... ;
- Il n'y a aucune garantie que l'appel soit géré correctement par l'OS (la méthode LaunchUriAsync retourne un booléen indiquant le succès ou non de l'opération).

Une autre alternative est d'utiliser l'association de fichiers, on se sert d'un fichier pour faire transiter des informations entre les applications.

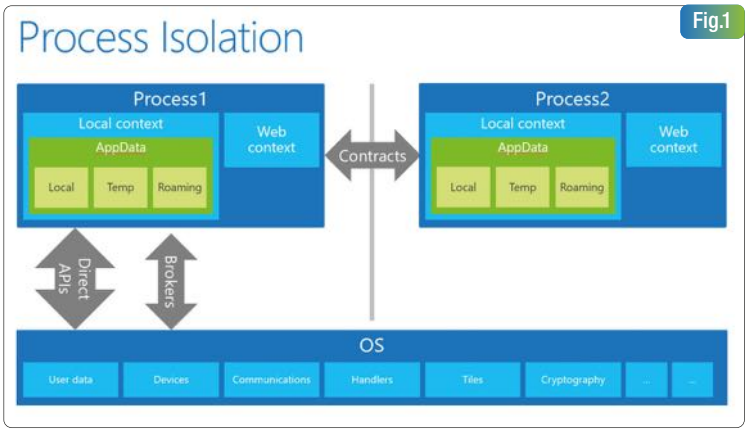


Fig.1

Fig.2

Nouveautés UWP

Un effort a été fourni pour lever les limitations que nous venons de citer. Cela passe par plusieurs nouveautés du SDK :

- Lancer une application bien spécifique par protocole en fournissant son nom de package ;
- Lancer une application bien spécifique (ou non) par protocole en fournissant un fichier ;
- Lancer une application bien spécifique (ou non) par protocole et attendre qu'elle nous retourne un objet ;
- Lancer une application bien spécifique (ou non) via extension de fichier ;
- Vérifier qu'une application gérant un protocole est bien installée sur le PC.

Avec ce nouveau kit, les scénarios de communication inter-applications peuvent être bien plus robustes et complets. Chez Infinite Square, une de nos fonctionnalités préférées est la possibilité d'appeler une application spécifique en lui fournissant un fichier, tout en attendant un retour suite à cette action. Voyons comment l'implémenter dans une application Fig.2.

Côté client

Le code client, c'est-à-dire l'application A envoyant des informations, va se charger d'activer une autre application :

```
// Définition de variable d'aide
var protocolUri = new Uri("InfiniteSquare://", UriKind.Absolute);
var packageFamilyName = "MonIdDePackage";
var fileUri = new Uri("ms-appx:///NousAInfiniteSquare.file");
```

```
var file = await StorageFile.GetFileFromApplicationUriAsync(fileUri);

var result = await Launcher.QueryUriSupportAsync(
    protocolUri, LaunchUriType.LaunchUri, packageFamilyName);

if (result != QueryUriSupportStatus.Success)
{
  // opération non concluante
  return false;
}

// On lance une app spécifique
LauncherOptions options = new LauncherOptions();
options.TargetApplicationPackageFamilyName = packageFamilyName;

// On lui donne un fichier
ValueSet inputData = new ValueSet();
var fileToken = SharedStorageAccessManager.AddFile(file);
inputData.Add("FileToken", fileToken);

// On attend le résultat
var launchResult = await Launcher.LaunchUriForResultsAsync(
  protocolUri, options, inputData);

if (launchResult.Status != LaunchUriStatus.Success)
{
  // opération non concluante
  return false;
}

//lecture du résultat
var retourDeLautreApp = launchResult.Result
  .FirstOrDefault(k => k.Key == "Resultat").Value;
```

Côté serveur

Côté serveur, c'est-à-dire l'application recevant les informations, nous avons deux étapes ; la première consiste à surcharger la méthode OnActivated de l'application pour identifier si l'application est activée par le biais d'un protocole. Si c'est bien le cas, on rajoute en paramètre le IActivatedEventArgs dans notre navigation.

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

    // Créer la frame si elle n'existe pas

    // navigation classique en passant le paramètre
    // d'activation
    rootFrame.Navigate(typeof(MainPage), args);

  }
  base.OnActivated(args);
}
```

Puis, dans un second temps, nous allons lire et traiter le paramètre qui est de type ProtocolForResultsActivatedEventArgs.

```
private ProtocolForResultsActivatedEventArgs _activationArgs;

protected override void OnNavigatedTo(NavigationEventArgs e)
{
  base.OnNavigatedTo(e);

  // Stockage du paramètre dans le bon type
  _activationArgs =
    e.Parameter as ProtocolForResultsActivatedEventArgs;
}

public void ReturnResult()
{
  // Création du résultat, on pourrait aussi renvoyer
  // un fichier via un token (comme à l'aller)
  var returnedData = new ValueSet();
  returnedData.Add("MonResultat", "Un objet ici");

  // On retourne le résultat à l'application originale
  _activationArgs.ProtocolForResultsOperation
    .ReportCompleted(returnedData);
}
```

Enfin, n'oublions pas, il est nécessaire de spécifier dans le manifest de l'application cible la gestion du protocole en question :

```
<Extensions>
 <uap:Extension Category="Windows.protocol">
  <uap:Protocol Name="InfiniteSquare">
   <uap:Logo>Assets\InfiniteSquare.png</uap:Logo>
  </uap:Protocol>
 </uap:Extension>
</Extensions>
```

AppService

Dans les nouveaux outils mis à disposition pour la communication inter-applications, nous avons AppService. Celui-ci propose une toute nouvelle façon de communiquer avec un concept intéressant : une application héberge un pseudo service Web local, qui expose ses données aux autres applications Windows Store Fig.3. Pour mettre en place AppService, on implémente l'interface IBackgroundTask de la sorte :

```
public sealed class AppServiceBackgroundTaskImpl : IBackgroundTask
{
```

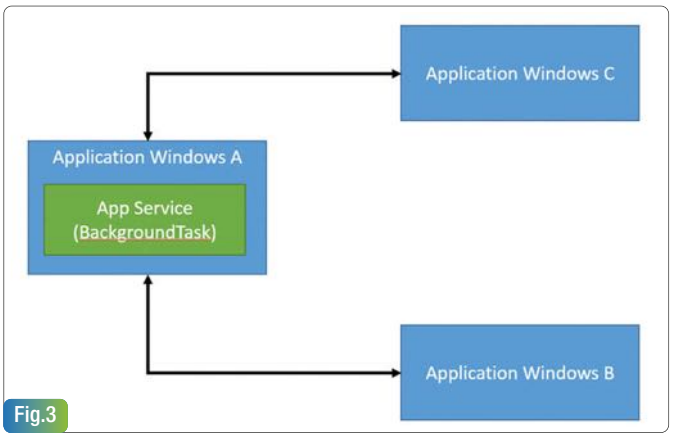


Fig.3

```
public void Run(IBackgroundTaskInstance taskInstance)
{
  var appServiceInfo = taskInstance.TriggerDetails as AppServiceTriggerDetails;
  if (appServiceInfo != null)
  {
    //nom du service lancé
    var name = appServiceInfo.Name;

  }
}
}
```

On est aussi habitué à renseigner les informations nécessaires dans le manifest de l'application :

```
<Extensions>
 <uap:Extension Category="Windows.appservice"
         EntryPoint="AppServiceBackgroundTask.AppServiceBackgroundTaskImpl"/>
 <uap:AppService Name="NousAInfiniteSquareAppService"/>
</Extensions>
```

AppService côté client

On retrouvera les étapes similaires à un appel réseau depuis notre application source :

- On essaye d'ouvrir la communication. L'API nous dit si c'est un succès, un échec, ou même si l'application n'est pas présente sur le device, nous permettant de demander à l'utilisateur d'installer l'application requise ;
- On prépare le message de type key/value ;
- On envoie le message ;
- On ferme la connexion.

Voici le code correspondant :

```
using (var appServiceConn = new AppServiceConnection())
{
  //Configuration pour cibler la bonne app
  appServiceConn.AppServiceName = "NousAInfiniteSquareAppService";
  appServiceConn.PackageFamilyName = "PackageFamilyNameDeLAppli";

  //Ouverture de la connexion
  var status = await appServiceConn.OpenAsync();

  //On peut savoir si l'app est installée aussi
  if (status != AppServiceConnectionStatus.Success)
  {
    return;
  }

  //Définition du message
  var message = new ValueSet();
  message.Add("Command", "Dis bonjour");

  //Envoi du message
  var resp = await appServiceConn.SendMessageAsync(message);

  if (resp.Status != AppServiceResponseStatus.Success)
  {
    //Message d'erreur
  }
}
```

AppService côté serveur

Côté serveur, dans la background task donc, nous utilisons le principe de « deferral ». Lors du premier Run du service, on récupère un deferral qui sera utilisé sur la durée de vie de la background task/du service, tandis qu'on utilise un nouveau deferral pour chaque requête.
On ajoute alors dans notre méthode Run :

```
//Abonnement à la réception d'une requête
appServiceInfo.AppServiceConnection.RequestReceived
  += AppServiceConnection_RequestReceived;

//Récupération du deferral pour le service
_seviceDeferral = taskInstance.GetDeferral();
```

Et pour finir, chaque requête va être implémentée de la sorte :

```
private async void AppServiceConnection_RequestReceived
  (AppServiceConnection sender, AppServiceRequestReceivedEventArgs args)
{
  var requestDeferral = args.GetDeferral();

  //lecture de la commande
  var commande = args.Request.Message.FirstOrDefault(k => k.Key == "Command");

  switch (commande.Value as string)
  {
    case "Dis bonjour":
      var response = new ValueSet();
      response.Add("response", "bonjour");
      await args.Request.SendResponseAsync(response);
      break;
    case "Quit":

      //Fin du service
      _seviceDeferral.Complete();
      break;
  }

  //Fin de la requête
  requestDeferral.Complete();
}
```

Il est intéressant de savoir qu'aucune limitation CPU n'est annoncée sur ce type de service, à l'opposé d'une background task classique. La taille des données qui peuvent être transmises par ce biais est limitée uniquement par les ressources systèmes. Il faut donc prendre en compte que le système se permette de ne pas lancer la background task correspondante l'AppService, faute de ressources système.
Dernière précision : les APIs correspondant à AppServices sont disponibles et utilisables dans n'importe quelle application .Net sur Windows 10. Un article décrit cette mise en place sur le blog d'Infinite square.

Au final

Comme on peut le constater, un gros manque est enfin comblé sur cette partie du SDK, et nous pouvons désormais établir des scénarios complets de communications inter-applications, sans devoir mettre en place des techniques et workflow sournois ! Aussi, on parle ici d'application UWP, cela est donc disponible autant sur les devices mobiles que desktop, et cela est une très bonne nouvelle pour toutes les applications grand public, mais surtout pour les applications LOB.

Valider ses formulaires avec JQuery

"La simplicité est la sophistication suprême." disait Léonard de Vinci au 15e siècle, et, force est de constater que proposer une expérience utilisateur simple à un internaute est une tâche souvent bien délicate. En particulier lorsque celui-ci est sollicité pour remplir un formulaire. Un gros travail doit être fait pour faciliter le remplissage de ce dernier. Dans cet article, nous allons présenter comment grâce à un plugin JavaScript nos formulaires vont prendre une forme bien plus moderne et agréable.

Gautier Deruette
Ingénieur développement
Osaxis

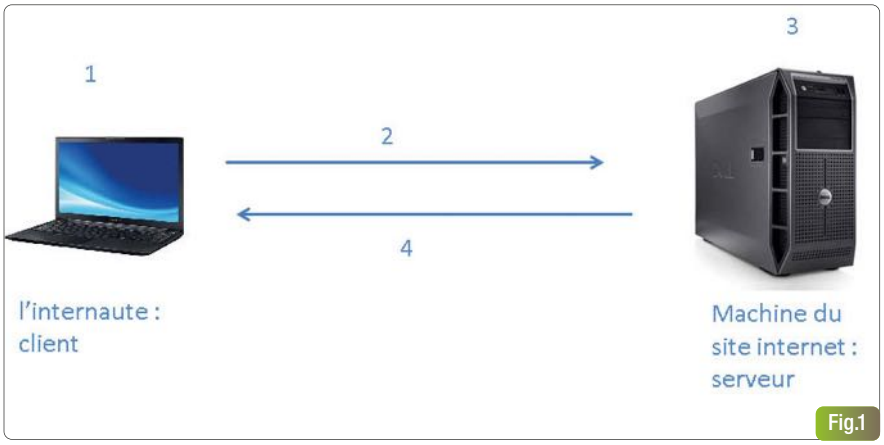


Fig.1

L'utilisateur n'attendra pas...

Sur la toile, il y a de plus en plus de formulaires intelligents, c'est-à-dire qu'ils nous signalent, aussitôt détectées, toutes erreurs syntaxiques qui seront refusées à la validation. C'est extrêmement pratique, car, en plus de signaler à l'internaute que son champ n'est pas syntaxiquement correct, cette pré-validation limite les allers-retours avec le serveur. Ceux-ci sont forcément plus longs et procurent donc une expérience utilisateur moins agréable. Nous rappelons tout de même que, pour des raisons de sécurité, tous les contrôles effectués du côté du client (sur son navigateur) doivent tout de même être effectués côté serveur Fig.1.

Les différentes étapes de validation du formulaire sont dans cette situation (figure 1) :

1. Validation côté client des erreurs syntaxiques (ex : email erroné) ;
2. Soumission du formulaire ;
3. Validation serveur qui vérifie les erreurs de données (ex : email déjà existant) ;
4. Réponse du serveur : succès ou erreur de soumission.

Nous essayons de faire un maximum de validations à l'étape 1 de notre schéma, les exécutions des étapes 2, 3 et 4 pouvant être coûteuses en temps.

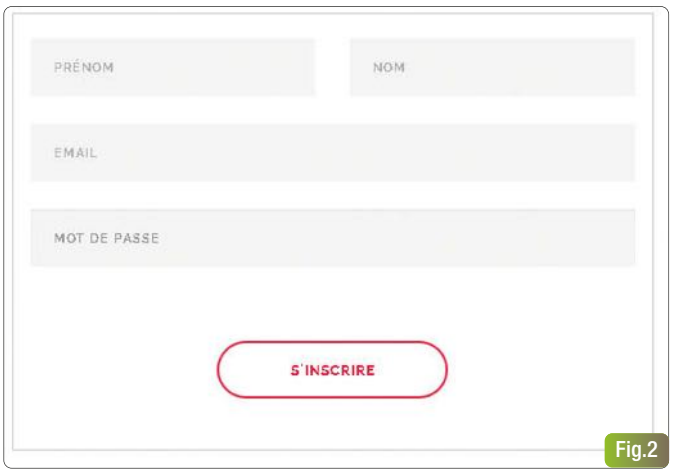


Fig.2

Deux minutes pour moderniser son formulaire

Nous vous proposons d'utiliser « jQuery Validator Plugin », une bibliothèque sous licence libre permettant de mettre en place très rapidement notre validation côté client.

Considérons le formulaire suivant, simple et sans fioritures : Fig.2.

Voici son code HTML :

```
<form method="post" id="mainForm">
  <input type="text" name="firstName" placeholder="PRÉNOM" required>
  <input type="text" name="lastName" placeholder="NOM">
  <input type="email" name="login" placeholder="EMAIL">
  <input type="password" name="password" placeholder="MOT DE PASSE">
</form>
```

Ici, remarquez que le champ « firstName » porte l'attribut « required » et que le champ email est de type email. Ce sont des attributs HTML 5 qui, sans ajout particulier, demandent au navigateur de signaler à l'utilisateur si le champ « firstName » est manquant ou si le mail est mal formé. Cependant, chaque navigateur le fera différemment. Intégrer le plugin va permettre d'uniformiser l'affichage.

Mettons le plugin en place. Il faut en premier lieu inclure « jQuery » ainsi que la bibliothèque du plugin :

```
<script src="http://code.jquery.com/jquery-1.12.0.min.js"></script>
<script  src="http://cdn.jsdelivr.net/jquery.validation/1.14.0/jquery.validate.min.js">
</script>
```

Et voilà les quelques lignes de code mettant en place le plugin :

```
$().ready(function() {
  $("#mainForm").validate(
  {

  });
});
```

Voilà ! C'est tout ! C'est simple non ? La seule « difficulté » ici est de mettre l'identifiant du formulaire dans notre code.

Le plugin fonctionnera correctement et vous guidera sur les champs « first-

Name » et email afin qu'ils soient correctement remplis. Il vous empêchera de valider le formulaire tant que les conditions indiquées ne sont pas respectées. Mais à part ça, il ne fait pas grand-chose. Il ne se fiera qu'aux informations connues dans le HTML et affichera ses messages d'erreur en anglais.

Par ailleurs, pour des raisons de clarté, il est déconseillé de procéder ainsi, c'est-à-dire de mettre les informations dans le HTML. Car au fur et à mesure que nous allons enrichir le formulaire et le personnaliser, celui-ci va grossir de manière désagréable, mais surtout certaines choses ne pourront se mettre que dans le JavaScript. Nous allons donc gérer toutes les conditions de validation dans le JavaScript, cela sera plus lisible.

Personnalisons

Nous allons donc personnaliser le message pour le mettre en français, et déplacer l'information expliquant que le champ « firstName » est requis et que le champ email doit être un email bien formé. Et nous ajoutons la contrainte que le nom doit comporter au moins trois lettres :

```
$().ready(function() {
 $("#mainForm").validate({
 rules : {
  firstName : {
   required : true
  },
  lastName : {
   minlength : 3
  },
  login : {
   required : true,
   mail : true
  }
 },
 messages : {
  firstName : "Veuillez fournir un prénom",
  lastName : "Veuillez fournir un nom d'au moins trois lettres",
  login : "L'email est incorrect"
 }
});
});
```

Remarquez que ce qui identifie le champ concerné est l'attribut « name ».

Lors de certains événements, le plugin va réagir et créer des alertes. Cela peut être la perte du focus d'un champ mal rempli ou tout simplement lorsque nous essayons de valider le formulaire. Dans l'exemple ci-présent, l'utilisateur a tenté de valider le formulaire alors que celui-ci est incomplet Fig.3.

Le plugin fait deux choses : premièrement, il ajoute un label sous chaque « input » incriminé avec le message d'erreur que nous avons paramétré. Deuxièmement, Il ajoute une classe CSS « error » à l' « input » permettant de customiser l'affichage. Ici un fond jaune a été choisi pour mieux se focaliser sur les champs mal remplis (Fig.4, HTML généré par le plugin lors d'une erreur sur le champ « firstName »).

```
<input id="firstName" class="error" type="text" placeholder="Prénom"
name="firstName" aria-required="true" aria-invalid="true">
<label id="firstName-error" class="error" for="firstName">Veuillez
fournir un prénom</label>
```

Fig.4

Customisons encore un peu

Nous allons maintenant ajouter une fonction que nous allons écrire nous-même. Nous allons forcer l'utilisateur à entrer un mot de passe sécurisé, à savoir qu'il devra avoir :

- Au moins une lettre minuscule
- Au moins une lettre majuscule
- Au moins un chiffre
- Au moins huit caractères

L'expression régulière exprimant ce besoin est celle-ci : ^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)[a-zA-Z\d]{8,}$

C'est typiquement le genre de contraintes syntaxiques qui fait fuir l'internaute s'il n'est pas bien guidé. Nous allons donc ajouter la fonction « addMethod » à notre code :

```
jQuery.validator.addMethod(
 "password",
 function(value, element) {
  return /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)[a-zA-Z\d]{8,}$/.test(value);
 },
 ' Le mot de passe doit contenir
  • Au moins une lettre minuscule
  • Au moins une lettre majuscule
  • Au moins un chiffre
  • Au moins huit caractères'
);
```

Nous remarquons ici :

- Le nom (« name ») du champ que nous voulons protéger constitue le premier argument de la méthode ;
- La fonction qui nous retourne un booléen. Elle teste si « value » respecte l'expression régulière donnée ;
- Le message d'erreur à afficher.

Une autre fonctionnalité souvent utile pour les mots de passe est la possibilité de comparer deux champs : il est souvent souhaité que l'utilisateur entre à nouveau son mot de passe afin de s'assurer qu'il a bien saisi ce qu'il souhaitait. Le plugin l'a prévu. Il suffit pour cela d'ajouter la règle « equalTo » au second champ mot de passe :

```
password-confirmation: {
```

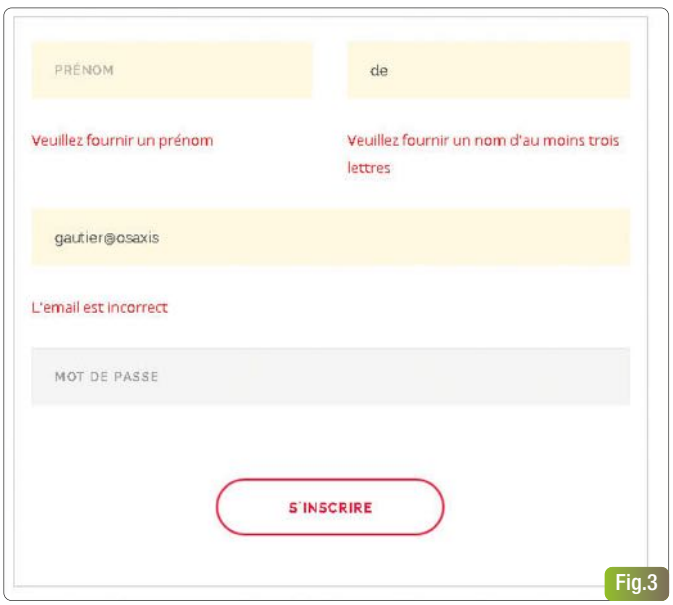


Fig.3

```
  required: true,
  equalTo: "#mainForm input[name='password]'"
}
```

Deux évènements utiles

Pour rendre plus agréable encore notre formulaire, nous allons voir que deux évènements peuvent être détectés. Ils permettront le déclenchement d'une réaction de notre part. L'API de notre plugin nous fournit deux méthodes qui sont appelées, l'une lorsque le formulaire est soumis alors qu'il est invalide et l'autre alors qu'il est valide. La première s'appelle « invalidHandler » et permet par exemple d'afficher une bulle d'aide supplémentaire à l'utilisateur. La seconde méthode a pour nom « submitHandler ». Elle permet par exemple de soumettre le formulaire en Ajax.

```
$("#mainForm").validate({
  rules : {
    //La liste des règles
  },
  messages : {
    //La liste des messages d'erreur
  },
  submitHandler: function(form) {
    ajaxSubmit(form);
    //appel d'une fonction qui va lancer
    //la soumission du formulaire en ajax
  },
  invalidHandler: function(form){
    //appel d'une fonction qui va
    //apporter de l'aide à l'utilisateur
    help(true);
  }
});
```

Nos formulaires sont vivants

Nous donnons enfin l'ensemble du code qui nous a permis de faire une série de contrôles basiques avant la soumission physique du formulaire (qui fera un aller-retour serveur). Remarquez que l'ensemble est simple et que les possibilités de personnalisation existent.

```
$().ready(function() {
 $("#mainForm").validate({
  rules : {
   firstName : {
    required : true
   },
   lastName : {
    minlength : 3
   },
   login : {
    required : true,
    mail : true
   },
   password-confirmation : {
    required: true,
    equalTo: "#mainForm input[name='password]'"
   }
  },
  messages : {
   firstName : "Veuillez fournir un prénom",
   lastName : "Veuillez fournir un nom d'au moins trois lettres",
   login : "L'email est mal formé"
  },
  submitHandler: function(form) {
   ajaxSubmit(form);
   //Appel d'une fonction qui va lancer
   //la soumission du formulaire en ajax
  },
  invalidHandler: function(form){
   //Appel d'une fonction qui va
   //apporter de l'aide à l'utilisateur
   help(true);
  }
 });
 jQuery.validator.addMethod(
  "password",
  function(value, element) {
   return /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)[a-zA-Z\d]{8,}$/.test(value);
  },
  ' Le mot de passe doit contenir
   • Au moins une lettre minuscule
   • Au moins une lettre majuscule
   • Au moins un chiffre
   • Au moins huit caractères'
 );
});
```

Grâce à ce code, nous minimisons les appels serveur, forcément plus coûteux. L'utilisateur aura une sensation d'être guidé en temps réel. Nous diminuons ainsi le risque qu'il ne s'en aille, par lassitude, avant la fin du processus.

Aller plus loin

Nous avons vu, grâce à un plugin facile d'utilisation, comment rendre un formulaire plus agréable. Les créateurs de ce plugin ont pensé à plein de choses. Par exemple, nous pouvons déplacer nos erreurs et toutes les afficher en haut de notre page dans une balise ayant pour id « errors » avec un style que nous aurons choisi. Il suffit pour cela d'ajouter les options suivantes dans notre plugin :

```
errorContainer: $("#errors"),
errorLabelContainer: $("ul", $("#errors")),
wrapper: 'li'
```

Une fois les possibilités du plugin bien exploitées, nous pourrions aller plus loin et proposer à l'utilisateur de le guider encore plus. Par exemple, il serait agréable de lui afficher précisément quels critères ne sont pas respectés dans son mot de passe par un jeu de couleur. De plus, Pour rendre la vue plus confortable nous pourrions également ajouter, grâce à d'autres bibliothèques, un formatage de certains champs. Ce formatage insèrera des espaces dans les champs qui contiennent un numéro de téléphone ou des symboles dans un champ contenant une date. Ceci rendra les erreurs de saisie plus évidentes.

Le formulaire est sans doute un des points clés d'un site internet efficace. Il convient d'y passer du temps afin que l'utilisateur en passe, lui, le moins possible. ⊞

Sources

Site officiel : http://jqueryvalidation.org

Développement d'une application AppleWatch avec watchOS 2

Depuis 2014 avec la sortie d'Android Wear, et en 2015 de l'AppleWatch, les applications pour montres connectées se multiplient. J'en ai réalisé une permettant d'afficher le top 10 des applications payantes sur l'AppStore. L'idée est de récupérer le contenu d'un webservice Apple, d'en 'afficher le résultat dans une liste, puis après sélection d'une application par l'utilisateur, d'en fournir le descriptif dans un second écran. Il est pour le moment obligatoire de développer une application iOS en parallèle, la montre doit représenter une ou plusieurs fonctionnalités de votre projet.

Benoit Briatte
Président Digipolitan
Professeur à l'ESGI

L'application sera réalisée sous watchOS 2. Cette nouvelle version du système d'exploitation apporte de nombreuses nouveautés par rapport à la version 1. Le code de l'extension s'exécute directement sur la montre sous watchOS2 et non plus sur l'iPhone. Les applications sont plus réactives et permettent aussi l'accès à certains composants matériel Fig.1.

L'Apple Watch aura toujours besoin de l'iPhone pour fonctionner mais l'application pourra être exécutée y compris si l'iPhone n'est plus à portée. De la même façon, pour l'accès aux réseaux Wi-Fi, l'Apple Watch pourra s'y connecter automatiquement, à conditions de s'être déjà connectée à l'iPhone au moins une fois. Cela permet un gain de performance et de réactivité des applications, et permet ainsi d'utiliser l'Apple Watch le plus souvent possible.

Sous watchOS 2, il est désormais possible de créer des applications visibles sur le cadran. Elles peuvent évoluer en fonction de l'heure et permettent un accès rapide à l'information pour les utilisateurs. Ceci est très utile pour des applications qui se mettent à jour en continu : Météo, Calendrier, Évènement sportif, etc.

De nombreux acteurs du marché des applications mobiles conçoivent une extension Apple Watch : *Shazam* permet de reconnaître une musique directement depuis votre montre, plus besoin de rechercher son téléphone pour savoir quelle musique est en train d'être jouée, *Runtastic,* une application de running, permet depuis l'Apple Watch de lancer directement des programmes et de voir l'avancement de son activité au poignet, très pratique lors d'une activité physique.

Fig.2

Voici les captures d'écran de ma réalisation : Fig.2.

Comment la réaliser ?

- Création du projet ;
- Apprentissage de l'interface graphique ;
- Récupération des données avec utilisation de webservices ;
- Affichage du résultat dans une liste ;
- Gestion des évènements et affichage de la page de détails ;
- Déploiement.

Fig.3

Création du projet

Depuis Xcode 7.0+, pour créer un projet iOS, il faut suivre la procédure habituelle (File/New/Project), choisir dans l'onglet watchOS la sous-rubrique application et sélectionner iOS App with WatchKit App, ce qui génère les dossiers et fichiers suivants :

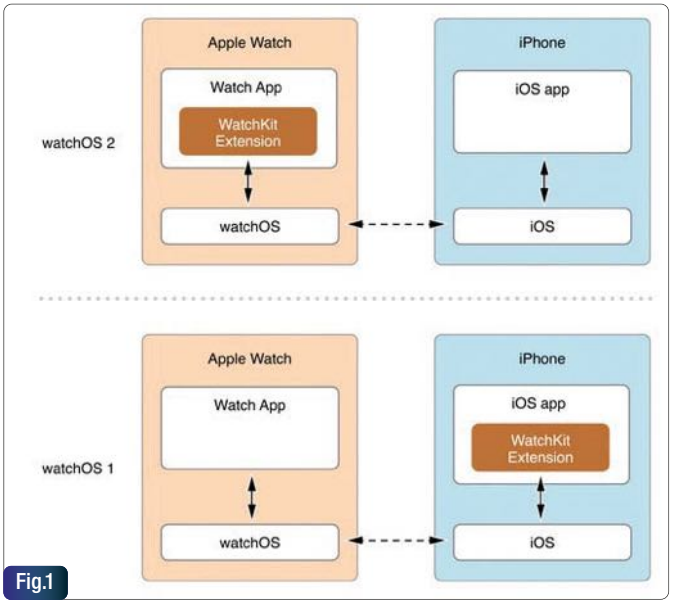


Fig.1

- ***TopTenWatch*** : dossier spécifique à l'application iPhone ;
- ***TopTenWatch WatchKit App*** : dossier des ressources de l'application contenant le fichier ***Interface.storyboard***, permettant de générer l'interface graphique à l'aide d'un builder ;
- ***TopTenWatch WatchKit Extension*** : dossier contenant le code source de l'extension AppleWatch contenant les fichiers :
  - ***InterfaceController.h*** et ***.m*** : controleur initial de l'extension (le premier écran).
  - ***ExtensionController.h*** et ***.m*** : classe permettant de récupérer les évènements du système (démarrage de l'application, mise en background ou foreground, ...) Fig.3.

L'interface graphique Fig.4.


Fig.4

L'interface graphique de l'AppleWatch est gérée par le fichier ***Interface.storyboard***, et suit le pattern MVC (Model View Controller) : la classe controller permet de faire le lien entre vues (les composants) et modèles de données. Elle a été repensée pour être intégrée dans une montre et doit être complètement définie dans le storyboard. Il sera donc impossible d'ajouter dynamiquement à partir du code des vues en dehors de celui-ci.

Si nécessaire, il suffira de définir les éléments dans l'interface graphique depuis le storyboard, et de les cacher / afficher en utilisant la property ***hid-***

den. Un simple Drag&Drop suffit à définir l'ensemble des vues dans l'interface, automatiquement insérées les unes sous les autres. Evidemment, si l'affichage dépasse la taille de la montre, la vue sera automatiquement scrollable.

Apple propose trois techniques pour gérer la largeur et la hauteur des éléments dans le storyboard : Fig.5.

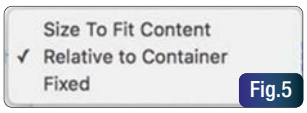


Fig.5

Size to fit content : la vue va prendre la largeur ou hauteur nécessaire pour se dessiner mais sera limitée par les dimensions de la vue parent.

Relative to Container : la vue s'adapte à la vue parent, avec un ajustement possible au champ suivant (valeur entre 0 et 1, 0 pour non visible, 1 pour 100% de la largeur ou hauteur du parent et 0,5 pour 50% de la taille du parent).

Fixed : les dimensions en pixels sont fixées par le développeur.

A noter : le nouveau composant AppleWatch, ***WKInterfaceGroup***, qui permet de regrouper plusieurs sous-vues dans un conteneur de manière horizontale ou verticale. Un ***WKInterfaceGroup*** peut contenir des sous ***WKInterfaceGroup*** pour construire des interfaces plus complexes. C'est la seule classe de WatchKit permettant d'ajouter des sous vues.

Pour modifier les propriétés d'une vue dynamiquement, il suffit de la rattacher à une variable du code. Trois étapes : appuyer sur la touche Ctrl, drag&drop la vue sur la ligne de code concernée, puis attribuer un nom à la variable, elle sera par la suite accessible dans la classe.

Récupération des données

Voici l'URL permettant de récupérer le top des applications payantes sur l'AppStore : https://itunes.apple.com/fr/rss/toppaidapplications/limit=10/json

Nous allons utiliser la classe NSURLSession pour faire la request HTTP qui attaquera le webservice, Cette classe utilisable depuis iOS 7 permet d'effectuer des requêtes HTTP très simplement sur iOS et AppleWatch. Le format de sortie du webservice sera le suivant : Fig.6.

Voici le code nécessaire au déclenchement du webservice dans la classe ***InterfaceController*** générée par défaut lors de la création du projet.

```
- (void) awakeWithContext:(id)context {
   [super awakeWithContext:context];
    [[[NSURLSession sharedSession] dataTaskWithRequest:[[NSURLRequest alloc] initWith
URL:[NSURL  URLWithString:@"https://itunes.apple.com/fr/rss/toppaidapplications/limit
=10/json"]] completionHandler:^(NSData * _Nullable data, NSURLResponse * _Nullable response,
 NSError * _Nullable error) {
     if(error == nil && data != nil) {
       NSDictionary* rssDictionary = [NSJSONSerialization JSONObjectWithData:data options:
0 error:&error];
        if(!error) {
          dispatch_async(dispatch_get_main_queue(), ^{
          NSArray<NSDictionary<NSString*, id>*>* entries = [[rssDictionary objectForKey:
@"feed"] objectForKey:@"entry"];
             [self reloadInterfaceTableWithEntries:entries];
          });
        }
     }
   }] resume];
}
```

Plusieurs points sont importants dans ce code :

- Le ***completionHandler*** est un bloc, un équivalent d'une fonction déclenchée lorsque la requête HTTP sera terminée (aboutie ou non, c'est pour cela qu'il y a le test ***error == nil && data != nil***).
- La classe ***NSDictionary*** (table de hachage type clé-valeur, équivalente à la classe ***HashMap*** de Java) est utilisée pour stocker le résultat du webservice (une chaîne de caractères au format JSON).
- La récupération de la liste des applications dans la clé ***feed*** puis la clé ***entry*** et l'appel de la méthode ***reloadInterfaceTableWithEntries*** pour afficher le contenu dans la liste.

Cette méthode est exécutée depuis le main thread, grâce à la fonction dispatch_async sur la main_queue, pour interagir avec l'interface graphique.

Affichage du résultat dans la liste

Sachant qu'il est impossible d'ajouter des vues dynamiquement, la partie graphique de la liste est différente de l'iPhone. Tout devra donc être construit dans le storyboard. L'ajout de vues se fait par drag&drop depuis la liste déroulante d'éléments situés en bas à droite d'Xcode

Le composant ***WKInterfaceTable,*** élément pour créer une liste est nommé ***Table*** dans le storyboard Fig.7.

Il existe dans ce composant un ***RowController*** (en bleu sur le screenshot), contrôleur utilisé pour chacune des lignes de la liste, et pour affecter un contenu aux Labels, Images, etc.

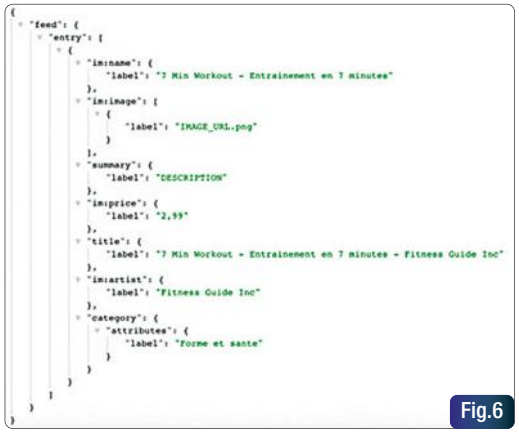


Fig.6

```
#import <Foundation/NSObject.h>

@interface GameRowController : NSObject

@property (readonly, nonatomic) NSDictionary<NSString*, id>* gameDictionary;

@property (readonly, nonatomic) NSUInteger rank;

- (void) setGameDictionary:(NSDictionary<NSString*, id>*)gameDictionary
                  rank:(NSUInteger)rank;

@end
```

Fig.7

Pour le construire, créer une classe supplémentaire (File/New/File, catégorie watchOS, Source WatchKitClass), appelée ici **GameRowController**. Sélectionner **NSObject** dans le menu déroulant SubclassOf.
Cette classe va permettre de remplir les ***WKInterfaceLabel*** et les ***WKInterfaceImage*** avec le contenu du NSDictionary récupéré à partir du webservice. Le ***rank*** permet de générer le numéro dans le classement de l'application que l'on souhaite afficher dans le ***RowController*.**
Dans l'implémentation de la classe ***GameRowController,*** des properties internes ont été rajoutées pour relier les variables de l'interface graphique au code à l'aide d'une extension de classe Objective-C.

```
@interface GameRowController ()
@property (strong, nonatomic) IBOutlet WKInterfaceLabel *rankInterfaceLabel;
@property (strong, nonatomic) IBOutlet WKInterfaceImage *interfaceImage;
@property (strong, nonatomic) IBOutlet WKInterfaceLabel *titleInterfaceLabel;
@property (strong, nonatomic) IBOutlet WKInterfaceLabel *categoryInterfaceLabel;
@property (strong, nonatomic) IBOutlet WKInterfaceLabel *priceInterfaceLabel;
@end
```

Voici le code de la méthode ***setGameDictionnary:rank:*** permettant d'alimenter les labels à l'aide du dictionnaire et de télécharger l'image rattachée à l'application grâce à la classe ***NSURLSession*** puis de l'afficher dans la liste.

```
@implementation GameRowController

- (void) setGameDictionary:(NSDictionary *)gameDictionary rank:(NSUInteger)rank {
   _gameDictionary = gameDictionary;
   _rank = rank;
   [self.rankInterfaceLabel setText:[NSString stringWithFormat:@"%lu", (unsigned long)
_rank]];
   [self.titleInterfaceLabel setText:[[_gameDictionary objectForKey:@"im:name"] object
ForKey:@"label"]];
   [self.categoryInterfaceLabel setText:[[[_gameDictionary objectForKey:@"category"] object
ForKey:@"attributes"] objectForKey:@"label"]];
   [self.priceInterfaceLabel setText:[[_gameDictionary objectForKey:@"im:price"] object
ForKey:@"label"]];
   NSArray<NSDictionary*>* images = [_gameDictionary objectForKey:@"im:image"];
   if([images count]) {
      NSString* imageString = [[images objectAtIndex:0] objectForKey:@"label"];
      NSURL* imageURL = [NSURL URLWithString:imageString];
       [[[NSURLSession sharedSession] downloadTaskWithRequest:[NSURLRequest request
WithURL:imageURL] completionHandler:^(NSURL * _Nullable location, NSURLResponse
* _Nullable response, NSError * _Nullable error) {
         if(!error && location) {
            NSData* data = [NSData dataWithContentsOfURL:location];
            if(data) {
               dispatch_async(dispatch_get_main_queue(), ^{
                  [self.interfaceImage setImageData:data];
               });
            }
         }
      }] resume];
   }
}

@end
```

Cette méthode remplit le contenu des Labels avec le informations du dictionnaire. Si une image est présente, je vous montre dans la suite de la méthode comment la télécharger, toujours grâce à la classe ***NSURLSession***. Il reste maintenant à coder la méthode ***reloadInterfaceTableWithEntries:*** *depuis* ***InterfaceController.m****, pour remplir la liste* ***WKInterfaceTable*** *des applications récupérées depuis le webservice.*

```
- (void) reloadInterfaceTableWithEntries:(NSArray<NSDictionary<NSString*, id>*>*)entries {
   NSUInteger count = [entries count];
   [self.interfaceTable setNumberOfRows:count withRowType:@"Game"];
   for(NSUInteger i=0; i<count; i++) {
      GameRowController* gameRowController = [self.interfaceTable rowControllerAtIndex:i];
      [gameRowController setGameDictionary:[entries objectAtIndex:i] rank:i+1];
   }
}
```

Cette méthode récupère les ***GameRowController*** depuis l'objet ***WKInterfaceTable*** et les alimente avec le contenu du dictionnaire de l'application ainsi que son classement dans l'AppStore (par l'exécution de la méthode ***setGameDictionnary:rank***)
La liste est maintenant visible et le top 10 des applications scrollable.

Gestion des évènements et affichage de la page détail

Reste la gestion du clic sur les éléments de la liste. Pour cela il suffit d'implémenter la méthode suivante dans le fichier ***InterfaceController.m*** :

```
- (void) table:(WKInterfaceTable *)table didSelectRowAtIndex:(NSInteger)rowIndex {
   if(self.interfaceTable == table) {
      GameRowController* gameRowController = [self.interfaceTable rowControllerAtIndex:
rowIndex];
      [self pushControllerWithName:@"GameDetailInterfaceController" context:gameRowController.
gameDictionary];
   }
}
```

L'index de la ligne sélectionnée donné en paramètre permet de récupérer le ***GameRowController*** qui lui est associé. Il est transmis au controller suivant grâce à la méthode ***pushControllerWithName:content:*** qui attend également qu'on lui fournisse l'identifiant de l'écran auquel on veut accéder. Le contexte correspond toujours aux données d'une application de l'AppStore
A noter que cet identifiant doit être inscrit dans ***Interface.storyboard***. Fig.8.

Fig.8

Pour l'étape suivante, créer une nouvelle classe (File/New/File, catégorie watchOS, Source/WatchKitClass) appelée ***GameDetailInterfaceController.*** Sélectionner ***WKInterfaceController*** dans le menu déroulant SubclassOf .
Elle servira à remplir dynamiquement le contenu de l'interface à l'aide du dictionnaire correspondant à l'application de l'AppStore, (titre et description).

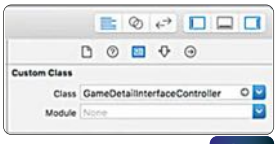


Fig.9

Il faut ensuite mettre dans ***Interface.storyboard*** le nom de la classe correspondante au controller afin que la bonne classe soit instanciée par le système. Fig.9.
Il est maintenant possible de récupérer le dictionnaire à un autre endroit du code, pour éviter de télécharger deux fois les informations de l'application. Le dictionnaire est transmis au controller d'interface suivant grâce au paramètre ***context*** de la méthode ***pushControllerWithName:context:*** .
La méthode ***awakeWithContext:*** devra être codée de façon à ce que le

context soit récupéré en paramètre pour qu'à la création de la classe, l'affichage puisse être effectué avec le contenu du dictionnaire.

```
@implementation GameDetailInterfaceController

- (void) awakeWithContext:(id)context {
   [super awakeWithContext:context];
   _gameDictionary = context;
   NSString* gameContent = [[_gameDictionary objectForKey:@"summary"] objectForKey:
@"label"];
   [self.contentInterfaceLabel setText:gameContent];
   NSString* gameTitle = [[_gameDictionary objectForKey:@"im:name"] objectForKey:
@"label"];
   [self.titleInterfaceLabel setText:gameTitle];
}

@end
```

Le déploiement

Il ne reste plus qu'à déployer l'application en connectant l'AppleWatch et l'iPhone par des câbles USB à l'ordinateur à condition de disposer d'un compte iTunes developer.

L'exécution demande de sélectionner la cible TopTenWatch WatchKit App et de choisir le device. Fig.10.

Voila, c'est fini ! Vous avez découvert la programmation classique AppleWatch. Les principaux problèmes ont été évoqués, il devrait être possible de passer à des applications plus complexes : utiliser les capteurs, jeux vidéo ou personnaliser des notifications.

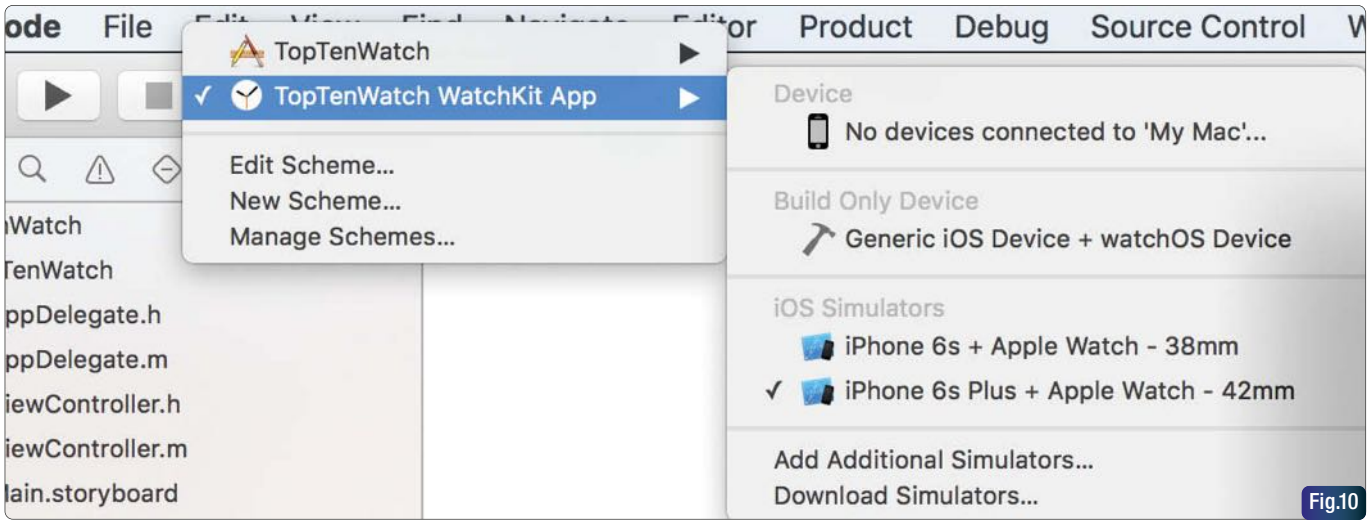


Fig.10

L'INFORMATICIEN + PROGRAMMEZ
versions numériques

OFFRE SPÉCIALE DE COUPLAGE

PDF

2 magazines mensuels, 22 parutions / an
+ accès aux archives PDF

PRIX NORMAL POUR UN AN : 69 €
POUR VOUS : 49 € SEULEMENT*

Souscription sur www.programmez.com

* Prix TTC incluant 1,01€ de TVA (à 2,10%).

Android N : tour d'horizon du millésime 2016 de l'OS mobile de Google

1ère partie

Comme il est de coutume chaque année désormais, Google a dévoilé une nouvelle version majeure de son OS pour mobile Android. Néanmoins, le planning est bien différent cette année puisque le géant de Mountain View aura dévoilé la première pré version d'Android N dès le mois de mars via un tout nouveau programme de preview à destination des développeurs. Dans cet article, nous vous proposons un tour d'horizon des nouveautés de ce millésime 2016 d'Android côté développeurs.

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

Le timing aura finalement surpris la plupart des développeurs du monde Android. En effet, Google ne nous avait pas habitué à dévoiler la nouvelle version d'Android si tôt dans l'année. Avec son nouveau programme de preview à destination des développeurs, Google a donc décidé de changer la donne et de donner plus d'importance aux développeurs afin d'obtenir leurs retours au plus tôt sur Android N. Annoncé le 9 mars 2016, ce programme de preview se veut ambitieux avec un planning prévoyant une nouvelle mise à jour tous les 4 à 6 semaines d'intervalle (**Fig.1**) pour une mise à disposition grand public durant le 3ème trimestre 2016.

Google semble d'ailleurs prendre ce programme très au sérieux puisque la première mise à jour prévue en Avril est bien sortie dans les temps le 13 avril 2016. Pour les développeurs souhaitant d'ores et déjà tester cette nouvelle version d'Android dont le nom définitif reste encore inconnu, il suffit de lancer le SDK Manager via Android Studio puis de récupérer toutes les mises à jour nécessaires que ce soit pour obtenir le SDK et les outils ou encore les images systèmes de l'émulateur. L'opération sera à répéter à chaque nouvelle mise à jour de la developer preview d'Android N.

Passage à OpenJDK et support de Java 8

Comme annoncé en toute fin d'année 2015, Android N sera bien la première version d'Android basée sur OpenJDK, l'implémentation open source de Java SE. Sûrement facilité par le procès toujours en cours opposant Oracle à Google concernant l'usage des APIs Java, ce choix ouvre la voie à un support de nombreuses fonctionnalités de Java 8 sous Android. Ces nouvelles fonctionnalités peuvent être classifiées en deux catégories : celles qui seront supportées depuis Android Gingerbread (API 9) et celles qui ne seront supportées qu'à partir d'Android N. Dans la première catégorie, on va ainsi retrouver les fameuses Lambda expressions ouvrant la voie à de la programmation fonctionnelle en Java et les nouvelles APIs utilitaires contenues au sein du package java.util.function. Pour la seconde catégorie, cela concerne les méthodes par défaut et statiques d'interfaces, les annotations répétables, les Streams, les APIs de Reflection ou encore les interfaces fonctionnelles via java.lang.FunctionalInterface. Pour bénéficier du support des nouveautés de Java 8 sous Android N, il est nécessaire d'utiliser la nouvelle chaîne de compilation Android nommée Jack pour Java Android Compiler Kit. Cette nouvelle chaîne possède son propre format (.jack) visant à remplacer le format .class. Elle propose également un certain nombre d'outils supplémentaires permettant d'optimiser et de réduire la taille des APKs tout en gérant au mieux le problème du multidex. Pour activer Android N et le support Java 8 au sein d'un projet de test, il est nécessaire de configurer le fichier build.gradle de votre module de la sorte :

```
android {
   compileSdkVersion 'android-N'
   buildToolsVersion '24.0.0 rc1'

   defaultConfig {
      applicationId "com.ssaurel.sample"
      minSdkVersion 'N'
      targetSdkVersion 'N'
      versionCode 1
      versionName "1.0"

      jackOptions{
         enabled true
      }
   }
   compileOptions {
      targetCompatibility JavaVersion.VERSION_1_8
      sourceCompatibility JavaVersion.VERSION_1_8
   }
   //...
}
```

Planning du programme de preview Android N

A présent, nous pouvons créer une interface avec une méthode par défaut en son sein :

```
public interface Thermometer {

   void setCelsius(final float celsiusValue);

   float getValue();

   String getSign();

   default String getFormattedValue(){
      return String.format(Locale.getDefault(),
         "Température : %.2f %s", getValue(), getSign());
   }
}
```

Puis, une classe classique l'implémentant :

```
public class FahrenheitThermometer implements Thermometer {
```

```
  private float fahrenheitDeg;

  public FahrenheitThermometer(float celsius) {
    setCelsius(celsius);
  }

  @Override
  public void setCelsius(float celsius) {
    fahrenheitDeg = celsius * 9 / 5 + 32f;
  }

  @Override
  public float getValue() {
    return fahrenheitDeg;
  }

  @Override
  public String getSign() {
    return Constants.DEGREE + "F";
  }
}
```

Enfin, il ne reste plus qu'à l'utiliser au sein d'une fonction Lambda permettant d'écouter un click sur un bouton :

```
buttonFahrenheit.setOnClickListener(view -> {
  fahrenheitThermometer.setCelsius(currentCelsius);
  String text = fahrenheitThermometer.getFormattedValue();
  makeText(MainActivity.this, text, Toast.LENGTH_SHORT).show();
});
```

On s'aperçoit clairement que le support de Java 8 apporte un énorme plus aux développeurs Android et va leur offrir un gain de productivité notable durant leurs développements.

Support Multi-Fenêtres

Proposé aux possesseurs d'appareils Samsung, via leur surcouche TouchWiz, depuis longtemps maintenant, le support Multi-Fenêtres était attendu de longue date par les autres utilisateurs d'Android. Android N vient combler ce manque en proposant un support en standard de cette fonctionnalité permettant aux utilisateurs d'afficher deux applications à l'écran en même temps (Fig.2).

Une fois l'écran divisé en deux parties, l'utilisateur peut déplacer la barre de séparation centrale afin d'agrandir ou de réduire à sa guise chaque application affichée. Mieux encore, Android N propose une fonctionnalité nommée "Freeform Window" permettant de bénéficier de véritables fenêtres multi-tâches déplaçables à souhait à l'écran. Las, cette fonctionnalité est désactivée par défaut et son activation sera laissée à la discrétion de chaque constructeur de smartphones ou tablettes Android ...

Du point de vue du développeur, le passage au Multi-Fenêtres ne s'accompagne pas de nouvelles APIs dédiées. Il va plutôt s'agir de renseigner de nouveaux attributs de configuration XML pour définir de quelle manière le Multi-Fenêtres sera supporté et de quelques méthodes au niveau de la classe Activity permettant de vérifier si l'on est ou non en mode Multi-Fenêtres. Pour le reste, le système de ressources proposé par Android permet une gestion simple et quasiment transparente pour le développeur puisqu'il se chargera de fournir les ressources alternatives nécessaires aussi bien au niveau des dimensions, layouts, drawables ou autres menus suivant les qualifiers définis. La principale donnée prise en compte étant

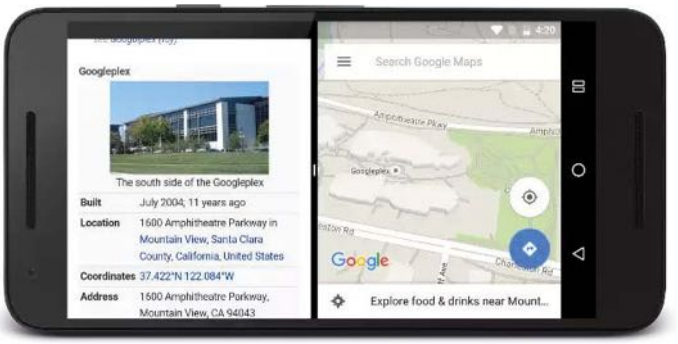


Fig.2 *Support Multi-Fenêtres sous Android N*

bien entendu la taille de l'écran, mais la taille minimum (smallestWidth) et l'orientation ont également un rôle à jouer.

Le support Multi-Fenêtres ne change pas le cycle de vie des activités. Petite précision toutefois, il faudra prendre garde de ne plus interrompre l'exécution du traitement dans une application au sein de la méthode onPause() puisque même en pause, elle pourra toujours être affichée à l'écran. Ainsi, en prenant l'exemple d'une application diffusant une vidéo, il ne faudra interrompre la vidéo que dans la méthode onStop() et la redémarrer dès l'appel à la méthode onStart(). Pour être informé du passage au Multi-Fenêtres, il faudra demander à ce que l'activité soit notifiée des changements de configuration à l'exécution. A minima, il faudra donc renseigner l'attribut configChanges comme suit :

```
<activity
 android:name=".MyActivity"
 android:configChanges="screenSize|smallestScreenSize|screenLayout|orientation"
/>
```

L'autre nouveauté induite par le mode Multi-Fenêtres concerne l'obligation désormais pour le développeur de supporter l'orientation paysage en plus de l'orientation portrait afin de garantir un affichage optimal de son application. En effet, le redimensionnement d'une fenêtre contenant votre application peut tout à fait la faire basculer du mode portrait au mode paysage tant bien même que l'appareil Android serait toujours en mode portait !

Au niveau configuration XML, il va être possible de spécifier qu'une activité ou une application ne peut être lancée en mode Multi-Fenêtres ou en mode Freeform en positionnant à false l'attribut android:resizeableActivity. L'élément layout est introduit avec Android N pour positionner de nouveaux attributs permettant de définir la taille par défaut en largeur et en hauteur d'une activité lancée en mode Freeform ainsi que la taille minimum à respecter en mode Multi-Fenêtres. Ainsi, on pourra configurer une activité comme suit au sein du manifest Android :

```
<activity android:name=".MyActivity">
  <layout android:defaultHeight="500dp"
      android:defaultWidth="600dp"
      android:gravity="top|end"
      android:minimalHeight="450dp"
      android:minimalWidth="300dp" />
</activity>
```

Au niveau de la classe Activity, quelques méthodes ont donc été ajoutées parmi lesquelles isInMultiWindowMode() qui renvoie true si l'activité est en mode Multi-Fenêtres ainsi que la méthode callback onMultiWindowModeChanged() qui est appelée par le système lorsque l'activité rentre ou sort du mode Multi-Fenêtres. En outre, il est possible d'indiquer au système que l'on souhaite lancer une activité en mode Multi-Fenêtres lors de l'appel

à la méthode startActivity() en ajoutant le nouveau flag Intent.FLAG_ACTIVITY_LAUNCH_TO_ADJACENT.
Très attendu des utilisateurs Android, ce nouveau mode est amené à connaître un grand succès et donc une grande utilisation. Les développeurs d'applications Android devront prendre garde à supporter au mieux les différentes configurations d'écrans et les différentes orientations pour construire des interfaces utilisateurs responsives.

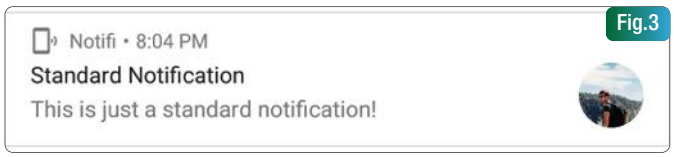
Amélioration des Notifications

Avec Android N, l'API liée aux Notifications se voit dotée de nombreuses nouveautés. Les développeurs ont désormais une plus grande liberté pour contrôler le look and feel des notifications tout en bénéficiant de nouvelles possibilités visant à améliorer l'expérience utilisateur. En premier lieu, le visuel a été revu pour assurer plus de clarté et une plus grande simplicité. Les nouveaux templates sont utilisés automatiquement par le système et il n'y a donc pas besoin de modifier son code. La construction d'une notification de base se faisant toujours de la sorte :

```
NotificationCompat.Builder builder = new NotificationCompat.Builder(context)
    .setSmallIcon(R.drawable.ic_drawable)
    .setContentTitle(title)
    .setContentText(message)
    .setLargeIcon(largeIcon)
    .setAutoCancel(true);
```

Le résultat présenté dans la Fig.3 laisse apercevoir le nouveau rendu visuel proposé aux utilisateurs Android.
Afin d'éviter de submerger l'utilisateur de notifications en tous genres dans sa barre de statut, les notifications liées pourront désormais être groupées au sein d'un composant extensible. Ainsi, lorsque vous souhaitez grouper plusieurs notifications, il vous faudra utiliser la nouvelle méthode setGroup() définie au sein de la classe NotificationCompat.Builder en passant en entrée un ID de type String qui permettra de regrouper entre elles toutes les notifications ayant le même ID. Prenons l'exemple d'une application affichant à l'utilisateur les nouveaux mails reçus, il va être possible de grouper chaque notification de nouveau mail entrant au sein d'une notification parent de type sommaire comme suit :

```
final static String GROUP_KEY_EMAILS = "group_key_emails";

// Construction des notifications avec mise en place du groupe
Notification notif = new NotificationCompat.Builder(mContext)
    .setContentTitle("Nouveau message de " + sender1)
    .setContentText(subject1)
    .setSmallIcon(R.drawable.new_mail)
    .setGroup(GROUP_KEY_EMAILS)
    .build();

NotificationManagerCompat notificationManager =
    NotificationManagerCompat.from(this);
notificationManager.notify(notificationId1, notif);

Notification notif2 = new NotificationCompat.Builder(mContext)
    .setContentTitle("Nouveau message de " + sender2)
    .setContentText(subject2)
    .setSmallIcon(R.drawable.new_mail)
    .setGroup(GROUP_KEY_EMAILS)
    .build();

notificationManager.notify(notificationId2, notif2);

// Ajout de la notification sommaire parent
Bitmap largeIcon = BitmapFactory.decodeResource(getResources(), R.drawable.ic_large_icon);

Notification summaryNotification = new NotificationCompat.Builder(mContext)
    .setContentTitle("2 nouveaux messages")
    .setSmallIcon(R.drawable.ic_small_icon)
    .setLargeIcon(largeIcon)
    .setStyle(new NotificationCompat.InboxStyle()
        .addLine("Sylvain Saurel  Hello")
        .addLine("Rachida Saurel  Coucou")
        .setBigContentTitle("2 nouveaux messages")
        .setSummaryText("sylvain.saurel@gmail.com"))
    .setGroup(GROUP_KEY_EMAILS)
    .setGroupSummary(true)
    .build();

notificationManager.notify(notificationId3, summaryNotification);
```



Nouveau template pour les Notifications

La notification de type sommaire créée via la méthode setGroupSummary() sera ainsi la seule à être affichée dans la barre de statut. Une fois étendue, on ne verra par la suite plus que les notifications simples du groupe. En s'appuyant sur l'API RemoteInput, il est désormais possible d'ajouter des actions de réponse directement au sein d'une notification. Une fois l'objet de type RemoteInput créé, il suffit de l'affecter à une action de notification puis de l'ajouter à la notification durant sa construction. Cela se fait comme suit :

```
private static final String KEY_TEXT_REPLY = "key_text_reply";

String replyLabel = getResources().getString(R.string.reply_label);
RemoteInput remoteInput = new RemoteInput.Builder(KEY_TEXT_REPLY)
    .setLabel(replyLabel)
    .build();

// Création de l'action de réponse
Notification.Action action =
    new Notification.Action.Builder(R.drawable.ic_reply_icon,
        getString(R.string.label), replyPendingIntent)
        .addRemoteInput(remoteInput)
        .build();

// Ajout de l'action durant la construction de la notification
Notification newMessageNotification =
    new Notification.Builder(mContext)
        .setSmallIcon(R.drawable.ic_message)
        .setContentTitle(getString(R.string.title))
        .setContentText(getString(R.string.content))
        .addAction(action))
        .build();

// Envoi de la notification
```

```
NotificationManager notificationManager = NotificationManager.from(mContext);
notificationManager.notify(notificationId, newMessageNotification);
```

Ce nouveau type d'action au sein des notifications (Fig.4) sera forcément apprécié des utilisateurs du fait du gain de temps quotidien que cela leur offrira. Au sein de l'activité cible de l'action, la récupération du contenu de la réponse se fait à partir de l'Intent et de la méthode statique getResultsFromIntent() de la classe RemoteInput :

```
private CharSequence getMessageText(Intent intent) {
  Bundle remoteInput = RemoteInput.getResultsFromIntent(intent);

  if (remoteInput != null) {
    return remoteInput.getCharSequence(KEY_TEXT_REPLY);
  }

  return null;
}
```

Dernière amélioration notable au niveau des notifications, la possibilité de définir ses propres vues tout en bénéficiant encore des décorations du système via la méthode setStyle() de la classe Notification.Builder. Pour construire une notification au contenu custom, nous pouvons recourir à la classe RemoteViews :

```
RemoteViews remoteViews = new RemoteViews(context.getPackageName(), R.layout.notif
_custom_view);
remoteViews.setImageViewResource(R.id.image_icon, iconResource);
remoteViews.setTextViewText(R.id.text_title, title);
remoteViews.setTextViewText(R.id.text_message, message);
remoteViews.setImageViewResource(R.id.image_end, imageResource);
```

Il reste ensuite à créer la notification en appliquant cette instance de RemoteViews via la méthode setCustomContentView :

```
Notification.Builder builder = new Notification.Builder(context)
    .setSmallIcon(R.drawable.ic_drawable)
    .setCustomContentView(remoteViews)
    .setStyle(new Notification.DecoratedCustomViewStyle());
    .setAutoCancel(true);
```

Economiseur de données

Après de gros efforts réalisés pour améliorer l'utilisation de la batterie avec Android Marshmallow, Android N s'attaque à l'utilisation des données cellulaires. Jusqu'à présent, l'utilisateur manquait de contrôle sur l'utilisation des données cellulaires par les applications de son appareil Android. Cette nouvelle mouture propose de donner plus de contrôle aux utilisateurs qui pourront définir au sein de l'application Paramètres les restrictions d'usage à appliquer lorsqu'un appareil est sur un réseau limité en termes de données. Il sera par exemple possible de bloquer l'utilisation des données lorsqu'une application tourne en arrière-plan. Tout ceci pouvant se définir par application. Côté développeur, cela implique de prendre en compte ces nouveaux choix offerts aux utilisateurs.

Ainsi, l'API ConnectivityManager a été étendue pour permettre aux développeurs de connaître les préférences de l'utilisateur dans ce domaine quant à leur application mais également de monitorer les éventuels changements de préférences. La méthode getRestrictBackgroundStatus() permet de connaître le mode d'économie des données choisi par l'utilisateur avec 3 valeurs possibles en retour :

- RESTRICT_BACKGROUND_STATUS_DISABLED indiquant que l'économiseur de données est désactivé ;
- RESTRICT_BACKGROUND_STATUS_ENABLED indiquant que l'économiseur de données est activé pour l'application appelante. L'application doit donc limiter l'utilisation des données lorsqu'elle est au premier plan et ne pas en consommer en arrière-plan ;
- RESTRICT_BACKGROUND_STATUS_WHITELISTED indiquant que l'économiseur de données est activé mais que l'application appelante a été placée en liste blanche ce qui implique qu'elle peut utiliser les données en arrière-plan.

Désormais, il est donc conseillé de respecter le pattern suivant lors des appels réseaux réalisés depuis une application Android :

```
ConnectivityManager connMgr = (ConnectivityManager)
      getSystemService(Context.CONNECTIVITY_SERVICE);

// Connexion de données mesurées (3G, 4G, ...)
if (connMgr.isActiveNetworkMetered()) {

  // Vérification préférences de l'utilisateur
  switch (connMgr.getRestrictBackgroundStatus()) {
    case RESTRICT_BACKGROUND_STATUS_ENABLED:
    // Economiseur de données activé pour cette application.
    // Appels réseaux en arrière-plan bloqués.
    break;

    case RESTRICT_BACKGROUND_STATUS_WHITELISTED:
    // App en liste blanche
    break;

    case RESTRICT_BACKGROUND_STATUS_DISABLED:
    // Economiseur de données désactivé, faire attention tout de même
    break;
  }
} else {
  // Connexion de données non mesurées (Wi-Fi par exemple)
  // Utilisation sans restrictions
}
```

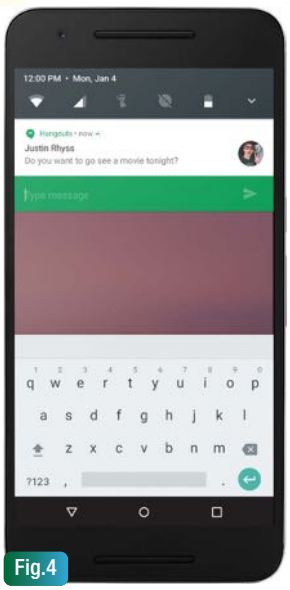


Fig.4 *Notification avec action de réponse*

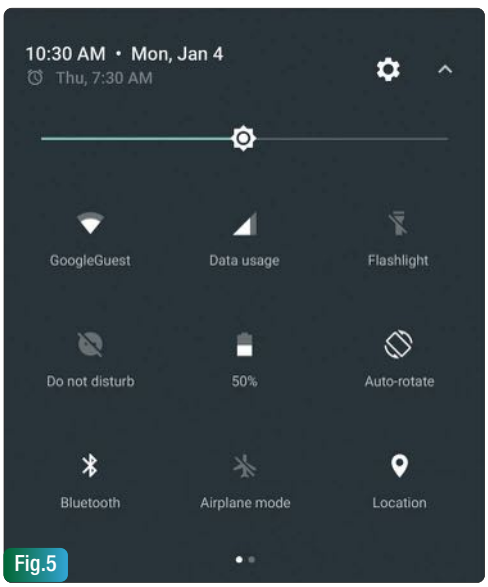


Fig.5 *Nouvelle API Quick Settings Tile*

Enfin, pour monitorer les changements dans les préférences de l'utilisateur quant à l'économie des données, il faudra créer un BroadcastReceiver à l'écoute de l'action "android.net.conn.RESTRICT_BACKGROUND_CHANGED" et l'enregistrer dynamiquement via l'appel à Context.registerReceiver(). Lorsque l'application recevra ensuite ce broadcast, il sera alors temps de faire un appel à ConnectivityManager.getRestrictBackgroundStatus() pour vérifier l'impact du changement sur son application.

Divers

Outre les nouveautés majeures présentées en détails dans cet article, Android N apporte tout un tas de petites évolutions qui bénéficieront aux développeurs, et, de fait, aux utilisateurs de l'OS. Parmi celles-ci, la nouvelle API Quick Settings Tile qui va permettre aux développeurs de définir leurs propres Quick Settings Tiles pour donner aux utilisateurs un accès direct et simple à des fonctionnalités de leurs applications (Fig.5). Le point central de l'API étant la classe android.service.quicksettings.Tile.

Côté Unicode, Android N intègre un sous-ensemble des APIs de la très populaire bibliothèque open source ICU4J. L'API est regroupée au sein du package android.icu. Certaines classes du SDK fournissent déjà des fonctionnalités équivalentes mais il est préférable de passer à ICU4J qui fournit un meilleur support pour les standards Unicode et la gestion des langues. Cela va par exemple concerner les classes android.icu.text.DecimalFormat, android.icu.util.Calendar ou encore android.icu.lang.UCharacter.

Au niveau graphique, Android N intègre la nouvelle API de rendu 3D Vulkan. Comme OpenGL ES, Vulkan est un standard ouvert pour le rendu des graphiques 3D qui est maintenu par le groupe Khronos. Contrairement à OpenGL ES, Vulkan a été conçu directement pour minimiser les overheads CPU et permettre aux applications de contrôler de manière plus directe les opérations GPU. Ce nouveau standard offre également un meilleur support de la parallélisation. Tous les outils et bibliothèques nécessaires à son utilisation ont été intégrés à Android NDK. Toutefois, il faut noter que seuls les smartphones supportant Vulkan au niveau hardware pourront bénéficier de cette API côté applicatif. Néanmoins, c'est un pas important pour le futur qui bénéficiera bien évidemment aux développeurs de jeux 3D sous Android.

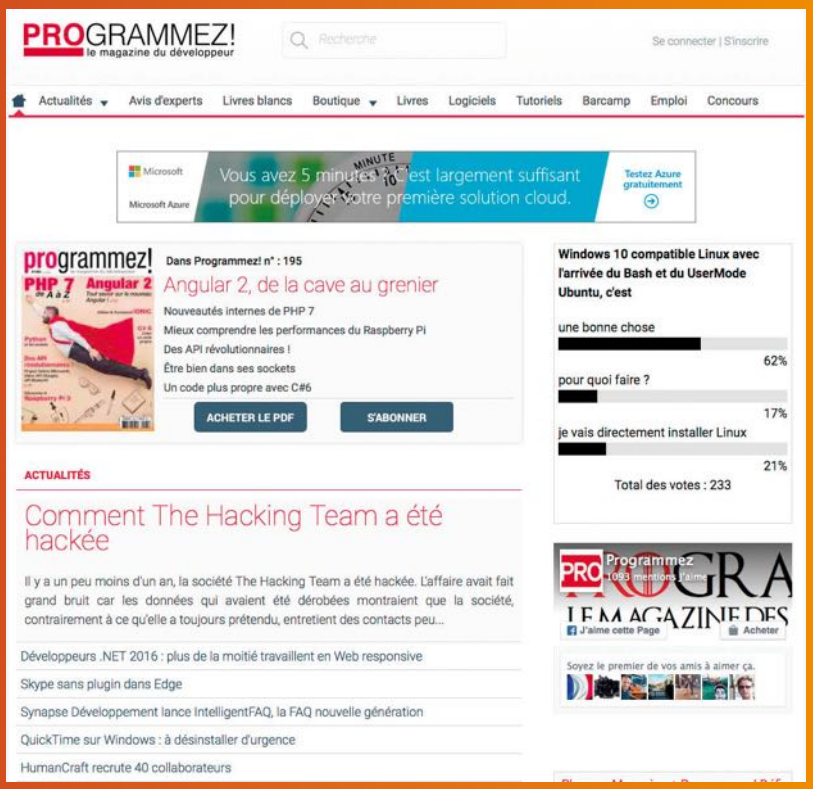
Conclusion

Annoncé plus tôt qu'à l'accoutumée par Google, le millésime 2016 d'Android s'annonce comme une étape importante pour le futur de la plateforme avec le passage à OpenJDK et le support de plusieurs fonctionnalités majeures introduites par Java 8. Pour le reste, pas de réelle révolution comme cela avait pu être le cas il y a 2 ans avec Android Lollipop et l'introduction du Material Design mais plutôt des touches d'amélioration avec une prise en compte des retours des utilisateurs de l'OS. Ainsi, le support Multi-Fenêtres arrive en standard pour répondre à une forte attente des utilisateurs et constituera un enjeu pour les développeurs qui devront gérer au mieux les changements de configuration d'écran au sein de leurs applications en s'appuyant sur le système de gestion des ressources alternatives d'Android. Enfin, la création d'un programme developer preview avec un suivi qui semble sérieux est une bonne chose qui permettra aux développeurs d'anticiper au mieux les nouveautés d'Android N. Pour le reste, il faudra encore être patient pour connaître le nom qui sera donné à cette nouvelle mouture ...

Suite dans le n°198.

Restez connecté(e) à l'actualité !

- L'**actu** de Programmez.com : le fil d'info **quotidien**
- La **newsletter hebdo** : la synthèse des informations indispensables.
- **Agenda** : Tous les salons, barcamp et conférences.



Abonnez-vous, c'est gratuit ! **www.programmez.com**

React, le starter kit pour bien démarrer

Depuis ces dernières années, l'utilisation de frameworks tels qu'AngularJS, Backbone ou EmberJS, pour le développement d'applications Web est devenue presque incontournable. Mais ces derniers mois, un nouveau nom commence à se faire une place dans le monde du développement Web, il s'agit de React.

Marina Avataneo
Développeuse front end chez **Yestudent**

Mais React c'est quoi ?

React est une librairie Javascript open source développée par Facebook, dont la première release a vu le jour au début de l'année 2013. Aujourd'hui, elle a été mise en place sur des plateformes célèbres telles que Netflix, Yahoo, AirBnb, Sony, Altassian et tout récemment Wordpress. Depuis, son niveau de popularité ne cesse d'augmenter Fig.1.

Une des particularités de React est qu'il ne s'occupe que de l'interface de l'application, c'est pourquoi on le qualifie parfois de moteur de rendu Web. En effet, à la différence de son principal concurrent, AngularJS, basé sur un modèle MVW (Model View Whatever) le plus souvent MVC (Model View Controller), React s'occupe essentiellement de la partie Vue. Il est par ailleurs possible de l'intégrer dans un framework basé sur du MVC, mais ce n'est pas le but premier puisque l'écosystème React permet aisément de se passer de ce genre de framework, notamment grâce à Flux.

Une architecture très souple

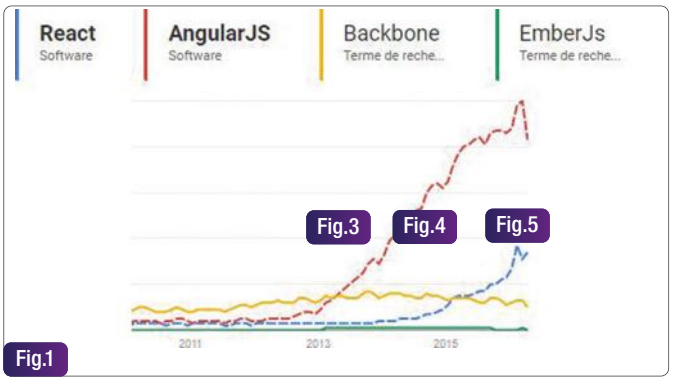
React est basé sur un système de composants. Le but est de décomposer au maximum son application en une hiérarchie de sous-composants, afin de les rendre les plus simples et réutilisables possibles.

Un composant React est défini par une state c'est à dire un état à l'instant t. Pour initialiser l'état d'un composant, on utilise la méthode getInitialState({prop: x}). Et pour modifier les propriétés de cet état, on utilise la méthode de setState({prop: y}). Un composant peut également posséder des propriétés en dehors de ses states, accessibles via l'objet this.props, qui peuvent être créées à la volée dans la classe ou bien passées en paramètres à l'instanciation du composant.

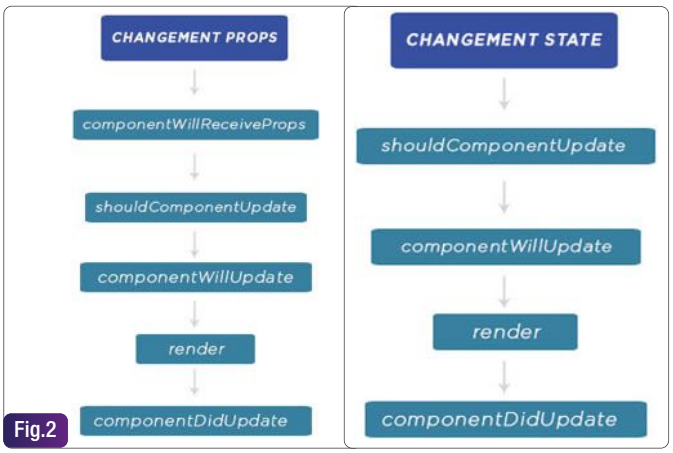
Il existe plusieurs méthodes de cycles de vie à utiliser dans la classe d'un composant. Ces méthodes permettent de gérer les états et les propriétés du composant au fil du temps. Par exemple lors d'un premier rendu, componentWillMount() est invoqué une fois, immédiatement avant le rendu initial. Mais il en existe d'autres comme componentWillReceiveProps() , appelé quand un composant reçoit de nouvelles propriétés Fig.2.

Il existe également la méthode componentWillUnmount, appelée immédiatement avant que le composant ne soit enlevé du DOM.

Dans une application utilisant React et Flux (dont nous allons parler un peu plus tard), on différencie deux types de composants. Les Smart Components (composants intelligents) et les Dumb Components (composants idiots). La principale différence entre ces deux types est leur besoin d'accès aux états externes. En effet, les composants intelligents sont moins réutilisables en dehors d'une application, car ils sont dépendants de l'état de celle-ci. Au contraire, les dumb components n'ont pas d'états propres et fonctionnent uniquement via les propriétés qui leur sont passées en paramètres. Par exemple on peut créer un composant Input, qui prend en paramètre un type (checkbox, radio, text...), une value, une fonction OnChange etc. Ce composant sera donc un dumb component car il n'est pas relié à une state externe. Il pourra donc être réutilisé au sein de différentes applications. En revanche, un composant permettant l'affichage des utilisateurs pour une application d'administration par exemple, sera lié à la state des utilisateurs, qui est propre à l'application, ce qui le rendra dépendant de celle-ci.



Courbe Google Trend illustrant l'évolution de l'intérêt de la requête "React" dans Google.



Les différentes méthodes et leur cycle de vie

Un virtual DOM pour gagner en performances

Mais React ne s'arrête pas là. Une autre caractéristique clé est sa performance, due à l'utilisation d'un virtual DOM. En effet, les accès au DOM en lecture mais surtout en écriture sont très lents. Avec cette méthode, on travaille avec une abstraction du DOM, ce qui permet de ne pas mettre directement à jour le véritable DOM. De ce fait, les mises à jour se font uniquement sur certaines parties de la vue, via la méthode render() de React. De cette manière, les mises à jour deviennent très efficaces et performantes. Pour cela, l'utilisation de la méthode shouldComponentUpdate vue plus haut, est très intéressante. En effet, elle est appelée avant le rendu quand de nouvelles propriétés ou états sont reçus. Grâce à nextState et nextProps, il n'y a plus qu'à faire des conditions comme sur cet exemple :

```
20  shouldComponentUpdate (nextProps, nextState){
21    let stateBody = this.props.body !== nextPops.body;
22    return stateBody;
23  }
```

Exemple d'utilisation de la méthode shouldComponentUpdate

Le return de ShouldComponentUpdate va donc renvoyer un booléen (true/false) qui déclenchera ou non la mise à jour du composant.
Par ailleurs, React n'utilise pas de système de templates et ne fonctionne qu'avec du Javascript. L'avantage est que cela permet d'encapsuler entièrement chaque composant de notre vue au sein d'une même classe, mais cela rend sa lisibilité plus difficile. C'est pourquoi, React fait appel à JSX. Il s'agit d'une syntaxe développée par Facebook, proche du XML, qui permet d'écrire du HTML augmenté au sein d'un code Javascript. De cette manière, l'écriture de la vue devient plus claire, et la création des composants plus intuitive. Il faudra cependant ajouter une étape de compilation JSX vers JS au système grâce à des outils comme Gulp ou Grunt.
De plus, en 2015, une nouvelle version du langage ECMAScript6 (ES6) a fait son apparition. Il était en effet nécessaire de mettre à jour ce langage, afin d'alléger sa syntaxe et de rattraper un certain retard qui a souvent desservi l'image du JavaScript chez de nombreux développeurs, en comparaison à des langages plus classiques. React est donc compatible avec ES6, mais malheureusement beaucoup de navigateurs ne le sont pas encore. C'est pourquoi il existe des outils comme Babel, qui permettent de transpiler le code ES6 en ES5, largement mieux supporté.

Flux et redux, un nouveau modèle d'architecture Web

Pour contrer les difficultés issues du modèle MVC et développer une nouvelle architecture, les équipes de Facebook ont mis au point Flux, caractérisé par son flux de données unidirectionnel **Fig.3**.
Dans cette architecture, en cas d'une action de l'utilisateur, le composant React appelle une méthode d'un objet Action qui appelle le dispatcher. Dans un même temps, les stores, qui écoutent le dispatcher, se mettent alors à jour et émettent à leur tour des notifications de mises à jour auprès des composants React. Cependant, bien que cette architecture soit efficace et robuste, sa mise en place nécessite une quantité de code assez importante. Plusieurs implémentations de Flux comme Fluxxor ou Reflux ont alors vu le jour, parmi lesquelles Redux tend à s'imposer. Redux est donc une évolution de Flux, ayant pour but de simplifier le mécanisme de dispatching des actions. Le concept consiste à remplacer le dispatcher de Flux - appelé dans les actions et chargé de faire les modifications dans un ou plusieurs stores - par un reducer **Fig.4**.
A l'opposé de Flux, le reducer de Redux écoute toutes les actions qui ont été déclarées et effectue les modifications sur un store unique et non mutable, c'est à dire dont l'état ne peut plus être modifié après sa création. Là encore, comme pour les composants, le but est de décomposer les reducers afin de les hiérarchiser et de rendre l'application plus maintenable.

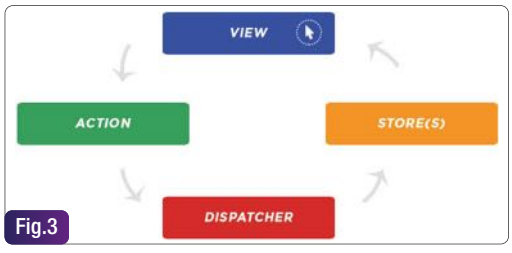


Fig.3

Illustration de l'architecture Flux

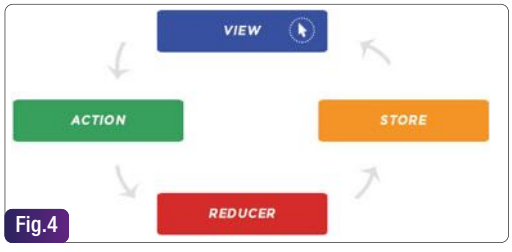


Fig.4

Illustration de l'architecture Redux

Router son application avec React Router

L'utilisation d'un routeur pour configurer une application React est indispensable. Et à ce jour, React Router est la bibliothèque la plus largement utilisée. Proche du système de routing d'Ember, ce routeur se présente sous la forme d'un composant React. Généralement, les routes sont définies dans un fichier global nommé routes.jsx, que l'on place à la racine du projet.

routes | index.jsx

```
ReactDOM.render((
  <Router>
    <Route path="/" component={Home} />
    <Route path="/users" component={Users} />
    <Route path="/products" component={Products} />
  </Router>
), document.getElementById('app'));
```

Exemple simple d'utilisation de React Router

Comme vous pouvez le voir sur cet exemple, Router contient toutes les routes qui vont être définies dans le projet. Chaque route est définie par le composant Route, qui prend deux propriétés : path et component. Lorsque la route définie dans path est appelée, le composant associé sera donc affiché.

Et pour compiler tout ça ?

Il existe plusieurs solutions pour la compilation d'applications, telles que Gulp ou Grunt, mais celle que nous allons aborder est Webpack. Il s'agit d'un outil permettant de faciliter l'organisation d'applications en modules. C'est notamment pour cela qu'il est très populaire dans la communauté React, caractérisé par l'utilisation de composants. L'avantage de Webpack est que son système de loaders permet de gérer d'autres format (css, jade, png etc.) en les transformant en modules Javascript. Voici un exemple de loaders issus du fichier de configuration de Webpack nommé par défaut webpack.config.js :

webpack.config.js | routes.jsx | index.jsx

```
{
    module: {
        loaders: [
            {
              test: /\.jsx?$/, // On prend tous les fichiers .js ou .jsx
              exclude: /node_modules/, // On exclue le fichier node_modules
              loader: 'babel', // On applique le loader babel
            },
            {
              test: /\.jade$/,
              loader: "jade"
            },
              test: /\.scss$/,
              loaders: ['style', 'css', 'sass']
        ]
    }
}
```

Exemple de configuration des loaders de Webpack dans le fichier webpack.config.js

Il faudra ensuite configurer les points d'entrée et de sortie du build ainsi que les plugins qui permettent de personnaliser un peu plus le système de compilation en minifiant le code par exemple.

Développer sa première appli step by step

Tout d'abord, pour l'installation de React, vous devrez installer Node.js car Facebook préconise de passer par npm (Node Package Manager). Pour ceux qui ne sont pas familiers avec Node.js, npm est son gestionnaire officiel de paquets. Il permet d'installer de nouveaux modules directement en ligne de commande, tout en gérant les dépendances. Dans un premier temps, lancez donc la commande :

```
npm install --save react react-dom
```

Par défaut, React se place dans un environnement de développement. Pour le basculer en mode production, il faudra modifier la variable

NODE_ENV. Ensuite, pour builder l'application React, il est recommandé de passer par des outils tels que Webpack ou Browserify. Dans la suite de ce tutoriel nous utiliserons également les syntaxes JSX et ES6.
Concernant l'architecture de votre projet, plusieurs solutions s'offrent à vous, voici un exemple

```
- src
  - /config
  - /actions
  - /assets
  - /components
  - /reducers
  - routes.jsx
  - index.html
```

Nous allons maintenant pouvoir créer une application React toute simple, ayant pour but d'afficher "Hello Programmez". Nous utiliserons pour cela les syntaxes ES6 et JSX. Commençons par créer un fichier index.html, à la racine de /src :

index.html
```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Hello Programmez</title>
  </head>
  <body>
    <div id="app"></div>
  </body>
</html>
```

Ce fichier va permettre de faire le lien avec la vue de la home page que nous allons créer par la suite. Celle-ci sera ensuite injectée dans l'application grâce à la balise . Créons maintenant la vue de la home page. Dans /components on crée un dossier /home, dans lequel on va placer un fichier index.jsx. Cette vue sera un composant React, qui contiendra les autres composants Hello et Programmez, vous suivez toujours ?

index.jsx
```
import React from 'react';
import ReactDOM from 'react-dom';

class HomePage extends React.Component {
  render() {
    return <h1>Home page</h1>
  }
}

ReactDOM.render(<HomePage/>, document.getElementById('app'));
```

Comme vous pouvez le voir, ReactDOM nous permet de faire le lien avec le fichier index.html créé un peu plus tôt. On injecte notre composant <HomePage /> via notre .
Par ailleurs, dans notre exemple, l'application est très simple puisqu'elle ne compte qu'une vue principale et deux composants mais dans des applications plus complexes, il est recommandé d'utiliser React Router, qui se chargera de router toutes les vues. Nous allons maintenant pouvoir créer nos composants permettant d'afficher "Hello" et "Programmez". Pour cela on crée un fichier hello.jsx et programmez.jsx dans /components/home.

hello.jsx
```
import React from 'react';

class HelloComponent extends React.Component {
  render() {
    return <span>Hello</span>
  }
}
export default (HelloComponent)
```

programmez.jsx
```
import React from 'react';

class ProgrammezComponent extends React.Component {
  render() {
    return <span>Programmez</span>
  }
}
export default(ProgrammezComponent)
```

En ES6, export permet de rendre les composants accessibles depuis l'extérieur. On peut donc désormais faire appel à ces composants dans celui de la home page.

index.jsx
```
import React from 'react';
import ReactDOM from 'react-dom';

import HelloComponent from 'hello.jsx';
import ProgrammezComponent from 'programmez.jsx';

class HomePage extends React.Component {
  render() {
    return (
      <div>
        <h1>Home page</h1>
        <p>
          <HelloComponent />
          <ProgrammezComponent />
        </p>
      </div>
    )
  }
}
ReactDOM.render(<HomePage/>, document.getElementById('app'));
```

On importe donc les composants et on les envoie dans la méthode render() du composant de la home page. A noter qu'il faut toujours que les différents composants soient imbriqués dans une div afin de ne former qu'un seul bloc à la sortir du render. On obtient ensuite le résultat suivant :

Voilà notre exemple est terminé, cependant vous devez vous dire qu'il n'est pas très utile de créer plusieurs composants pour du texte. Et vous avez raison, au lieu de cela il faudrait plutôt réaliser un composant de type Text, qui puisse être réutilisable dans toute l'application.

index.jsx
```
import React from 'react';
import ReactDOM from 'react-dom';

import TextComponent from 'text.jsx';

class HomePage extends React.Component {
  render() {
    return (
      <div>
        <h1>Home page</h1>
        <TextComponent value="Hello programmez" />
      </div>
    )
  }
}
ReactDOM.render(<HomePage/>, document.getElementById('app'));
```

On passe donc le texte souhaité dans la propriété "value" du composant, que l'on récupère ensuite de cette manière à l'intérieur du composant Text :

text.jsx
```
import React from 'react';

class TextComponent extends React.Component {
  render() {
    return <p>{this.props.value}</p>
  }
}
export default (TextComponent)
```

Nous avons maintenant le même résultat en sortie, mais avec un code plus léger et dynamique. Cependant, il y a une fois de plus pas de grande utilité à réaliser un composant qui affiche simplement du texte, puisqu'on pourrait le faire directement dans le code. C'est là que React devient intéressant. Car si on implémente un peu plus notre composant Text, on peut imaginer lui ajouter des propriétés comme un type (h1, h2, h3), une position (text-align: center, left, right, justify....), une couleur et bien d'autres encore. De cette manière on peut gagner beaucoup de temps en développement car une fois que le composant est réalisé, il suffit d'ajouter la balise Text avec ses différents paramètres.

Et le SEO (Search Engine Optimization) dans tout ça ?

React se démarque aussi par le fait qu'il peut être exécuté coté serveur (en utilisant Node.js par exemple) à des fins de performance mais surtout de référencement. En effet, le problème majeur de frameworks tels qu'Angular ou Backbone réside dans le fait que les contenus sont injectés côté client, après le chargement de la page côté serveur. De ce fait, Googlebot ne peut pas crawler et indexer ces contenus ce qui est catastrophique d'un point de vue SEO. Avec React, le rendu de l'application est le même que l'on soit coté client ou coté serveur, c'est ce qu'on appelle l'isomorphisme.

Une librairie en constante évolution

Bien qu'elle soit encore jeune et en pleine expansion, la communauté React est très active. On trouve actuellement une bonne majorité de composants traditionnels sur Internet, bien que pour les plus spécifiques il faille encore un peu patienter et les implémenter soi-même. Cependant le Web évolue vite et la nouvelle version 15.0 de React a vu le jour le 7 avril 2016. A noter que Facebook a décidé de changer son système de notation de versions puisque la 15.0 fait suite à la 0.14. La notion de stabilité indiqué par le 0 au début a donc été déplacée à la fin de la notation.

Un des changements majeurs est celui de la méthode utilisée pour le premier rendu. En effet, avant la version 15.0, React générait un long string de HTML qu'il envoyait via node.innerHTML. Désormais React utlise document.createElement, ce qui lui permet d'être encore plus rapide. Ainsi, l'attribut data-reactid qui permettait d'identifier de manière unique chaque élément du DOM et de faire le lien entre la partie serveur et client ne sera plus visible dans les balises côté client.

Aperçu de la console d'une application React avant la version 15.0

De cette manière il est plus difficile de savoir si un site utilise React mais cela permet d'avoir un DOM beaucoup plus léger et cela améliore donc les performances. Un autre changement majeur est la suppression des balises autour des éléments de texte. En effet, avant cette version, il y avait beaucoup d'ajouts inutiles de balises, ce qui alourdissait le code au rendu. Par exemple pour le composant suivant :

```
<div> Hello {this.props.value} </div>
```

Le rendu était visible sous cette forme :

```
<div> <span>Hello></span><span>Programmez</span></div>
```

Désormais il sera tout simplement affiché de cette manière :

```
<div> Hello Programmez </div>
```

Enfin une autre amélioration importante est la gestion des SVG. Désormais tous les tags SVG seront pris en charge ainsi que les attributs SVG implémentés par les navigateurs. De manière générale cette nouvelle version permet donc de simplifier le code en supprimant les éléments superflus afin de gagner en performances.

Du mobile avec React Native

React Native est un nouveau framework développé par Facebook en 2015, permettant de créer des applications mobiles natives.

"Learn one, write everywhere" (Apprendre une fois, écrire partout) est la devise de React Native. En effet, il est basé sur JavaScript et React, ce qui facilite considérablement son apprentissage. La syntaxe étant la même, le passage de React a React Native est très simple. Seuls les composants changent, ils sont donc natifs. Par exemple sous iOS, une div devient une View, un paragraphe un composant Text, etc.

En résumé

- React n'est pas un framework, mais une librairie, que l'on peut associer à la Vue d'un modèle MVC.
- Redux est une librairie issue de Flux et créée par Dan Abramov, pour palier le besoin d'architecture de React. Redux est plus léger que Flux et son concept repose sur un store unique et immutable.
- React est un système orienté composants qui sont définis par des propriétés et des états pouvant changer au cours du temps.
- Bien que ce ne soit pas une obligation, il est conseillé d'utiliser de nouvelles syntaxes telles que JSX et ES6 (à compiler avec Babel pour la conversion avec ES5).
- React Router est le système le plus utilisé pour le routing d'applications sous React.
- React est SEO friendly grâce à son isomorphisme. Les composants React peuvent être rendus simplement côté serveur sans PhantomJS.
- React dispose d'un virtual DOM, ce qui lui permet d'être très performant puisque les mises à jours se font uniquement sur les éléments qui en ont besoin.
- Pour builder votre application il vous faudra passer par des outils tels que Webpack ou Browserify.
- React Native est basé sur React et permet de développer facilement de applications mobiles.

React propose donc une nouvelle vision de la conception d'applications Web. Son approche orientée composants permet de hiérarchiser et de découper intelligemment son application afin de la rendre plus scalable et performante. De plus, la courbe d'apprentissage de React est très rapide. L'avantage de l'utilisation de JSX est que cela permet aux développeurs, même débutants, de comprendre rapidement comment fonctionne le code. Et c'est également le cas pour le passage à React Native, facilité par la similarité du code avec React, et permettant ainsi de rendre le développement d'applications natives iOS et Android plus simple et accessible.

React est donc une librairie avec un énorme potentiel, qui a su se développer très rapidement depuis sa parution en 2013, tout en proposant aujourd'hui une version stable et sans break changes entre les releases. De plus, la communauté très active de React ne cesse de s'agrandir, au milieu d'un écosystème déjà très riche, en marche pour peut-être révolutionner le monde du développement Web. ⊞

Créer son Add-In Excel et l'utilisation d'un ribbon.

La création d'un Add-In permet de traiter des données en utilisant toutes les fonctionnalités d'Excel mais aussi de les étendre grâce au .NET. A travers cet article vous verrez comment créer un Add-In Excel. Je mettrai aussi en place un nouvel onglet dans le Ribbon. Je donnerai également toute une série de trucs et astuces qui m'ont été très utiles pour développer un Add-In.

Cédric Michel
Analyste développeur chez **Trasys.**
Spécialiste .NET
Membre (MEET – Microsoft Extended Expert Team)
http://www.microsoft.com/belux/meet/

Pour pouvoir créer votre Add-In, vous devrez vous munir de Visual Studio 2008 ou supérieur. Selon la version d'Office que vous voulez atteindre, vous devrez avoir une version compatible de Visual Studio. Voici un tableau reprenant les versions de Visual Studio et la version office compatible.

Visual studio / Office – Version	2003	2007	2010	2013	2016
2008	x	x			
2010		x	x		
2012			x	x	
2013			x	x	
2015			x	x	x

X : Il s'agit du même type de projet et donc compatible dans les 2 sens.

Si vous créez un Add-In pour Office 2007 celui-ci reste compatible, jusqu'à la dernière version (2016).
Par contre un Add-In 2003 n'est pas compatible avec les versions suivantes. Vous pouvez retrouver le tableau complet des possibilités d'exécution des solutions dans les différentes versions ici (Vous trouverez également la version du Framework nécessaire pour chacune des versions). https://msdn.microsoft.com/fr-fr/library/bb772080.aspx

1 - Création de l'Add-In

Faites donc bien attention à la version que vous allez choisir. (Ici, il s'agit d'un Add-In pour office 2013 créé avec Visual Studio 2013 et donc compatible Office 2016) Fig.1. Lorsque votre projet est créé vous pourrez en voir la structure dans l'explorateur. Il s'agit d'une structure très simple. Le projet comprend un container Excel dans lequel vous trouverez un fichier ThisAddIn.cs. Ce fichier contient une classe partielle ThisAddIn avec deux méthodes. La première, ThisAddIn_Startup, est une méthode qui est appelée dès l'ouverture d'Excel. La seconde, ThisAddIn_Shutdown est appelée sur la fermeture d'Excel Fig.2. Dans le fichier ThisAddIn.Designer.cs (appuyé sur show all files 1*) vous trouverez une méthode Initialize() qui fait appel à base.Initialize(); qui vient de la classe AddInBase. C'est lors de cet appel que les autres composants sont chargés. (exemple le Ribbon).

2 - Ajout du Ribbon

Afin de définir une zone avec vos boutons et autres contrôles que vous désirez ajouter vous pouvez ajouter un nouvel onglet dans le Ribbon.
Pour ce faire, il faut ajouter un ribbon. Faites un clic-droit sur le nom du projet et ajoutez nouvel élément. Choisissez Office/Sharepoint puis Ribbon, vous avez le choix entre XML ou Designer. J'ai préféré prendre Designer. Ainsi pour ajouter vos contrôles, il suffira de faire du drag&drop et définir les propriétés Fig.3. Ensuite via la fenêtre des propriétés donnez un nom (Démo Tab1) au RibbonTab. Si vous exécutez maintenant votre application vous aurez ceci Fig.4. Dans le mode design vous pouvez ajouter autant de RibbonTab que besoin. Il suffit pour cela de faire un clic droit sur le ribbon et dans le menu vous verrez la possibilité d'ajouter un ribbonTab Fig.5.
Dans ce designer appuyez sur la touche F7 pour voir le code ou cliquez sur View Code dans le menu. Vous arriverez dans la classe de votre Ribbon.
Une méthode Ribbon_Load est présente et est appelée au chargement du Ribbon qui vient juste après l'ouverture d'Excel.

3 - Astuces

3.1 - Gestion des Add-In dans Excel.

Si cette popup (Excel a rencontré un problème grave avec le complément «) apparaît, cela veut dire que lors de la dernière exécution, votre Add-In a rencontré un problème (exception,...).
Le oui/non aura pour effet de désactiver ou non votre Add-In.
Une fois celui-ci désactivé, il vous faudra le réactiver dans les options d'Ex-

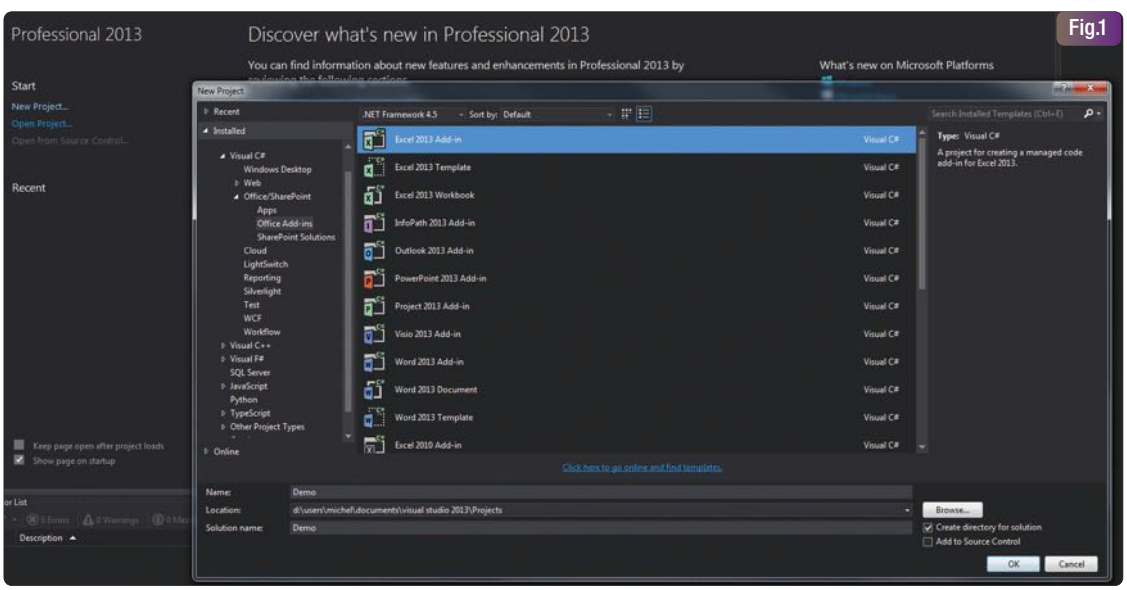


Fig.1

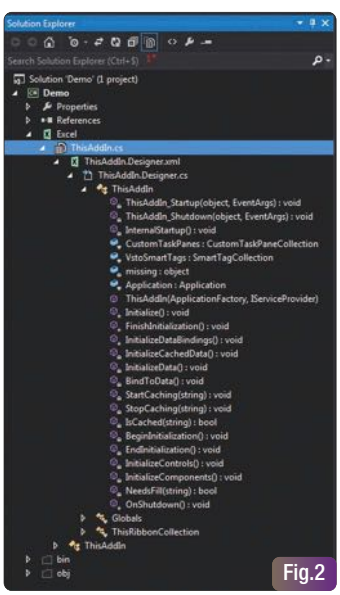


Fig.2

cel sinon celui-ci n'apparaîtra plus. Pour ce faire dans Excel allez sur Fichier puis Options choisissez l'onglet Compléments. Vous aurez la liste de tous les Add-In présents. Pour les gérer, il faut utiliser le menu déroulant et le bouton atteindre tout en bas. Choisissez dans ce menu « éléments désactivés » puis cliquez sur atteindre. Dans l'écran suivant sélectionnez le composant puis cliquez sur Activer Fig.7.
Vous pouvez également constater que le chemin vers l'Add-In pointe sur le fichier VSTO dans le répertoire Bin de notre application. Donc à chaque compilation celui-ci est remplacé.

3.2 – Gestion des using

```
using Excel = Microsoft.Office.Interop.Excel;
using Tool = Microsoft.Office.Tools.Excel;
```

Pour le premier, il s'agit du namespace utilisé pour gérer les objets dans un Add-In (.VSTO) Le second est un namespace qui contient des fonctionnalités supplémentaires pour Excel. De ce fait, quand vous faites des recherches vérifiez bien dans quel namespace se trouve ce dont vous avez besoin. Ici nous l'utiliserons pour la génération d'un graphique.

3.3 - Lecture et écriture dans une feuille

Pour lire les données d'une cellule, il faut connaître sa référence (exemple « B3 ») :

```
Excel.Worksheet sheet = Globals.ThisAddIn.Application.ActiveSheet;
    var value = sheet.get_Range("B3").Value;
```

Vous pouvez également retrouver la valeur via l'index de la ligne et de la colonne :

```
Excel.Worksheet sheet = Globals.ThisAddIn.Application.ActiveSheet;
    int rowIndex =3;
    int columnIndex =2;
    var value = sheet.Cells[rowIndex, columnIndex].Value;
```

Pour écrire des données, il suffira de faire :

```
Excel.Worksheet sheet = Globals.ThisAddIn.Application.ActiveSheet;
    sheet.get_Range("B3").Value = "une valeur";
```

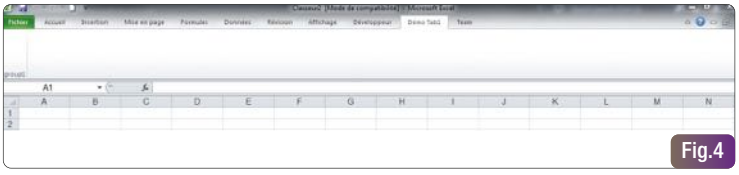


Fig.4

Si vous désirez connaître les index depuis une référence :

```
Excel.Worksheet sheet = Globals.ThisAddIn.Application.ActiveSheet;
    int rowIndex = sheet.get_Range("B3").Row;
    int columnIndex = sheet.get_Range("B3").Column;
```

Si vous désirez connaître la référence depuis les index :

```
Excel.Worksheet sheet = Globals.ThisAddIn.Application.ActiveSheet;
    int rowIndex = 3;
    int columnIndex = 2;
    Excel.Range range = sheet.Cells[rowIndex, columnIndex];
    string cellReference = range.AddressLocal.Replace("$", string.Empty);
```

3.4 - Charger un template

Si votre template utilise des sources de données existantes censées se rafraîchir à l'ouverture, celles-ci ne se réactualiseront pas. Pour ce faire il faut lancer manuellement un refresh.

```
Excel.Workbook workbook = Globals.ThisAddIn.Application.Workbooks.Open(@"C:/temp/
template.xltx");
workbook.RefreshAll();
```

3.5 - Génération d'un graphique

Pour le graphique, il vous faudra définir plusieurs choses. Pour créer un graphique, il faut définir la zone de données utile pour le graphique. Ensuite pour ajouter le graphique dans une feuille, il faut définir la zone dans laquelle on veut que le graphique soit présent. On vérifie si le contrôle était déjà présent sur la feuille, et, enfin, on génère le graphique. Attention que pour que cela se fasse, il faut avoir le using Tool Microsoft.Office. Tools. Excel car c'est dans celui-ci que réside la méthode .Controls.AddChart et le Chart.

```
/// <summary>
/// Génération d'un graphique
/// </summary>
public void GenerateChart()
{
  //Définir le nom du contrôle pour le graphique
  string chartName = "Chart name";

  //Nouvelle feuille pour dessiner le graphique
  Excel.Worksheet newWorksheet;
  newWorksheet = (Excel.Worksheet)Globals.ThisAddIn.Application.Worksheets.Add();
  newWorksheet.Name = "Sheet name";
```

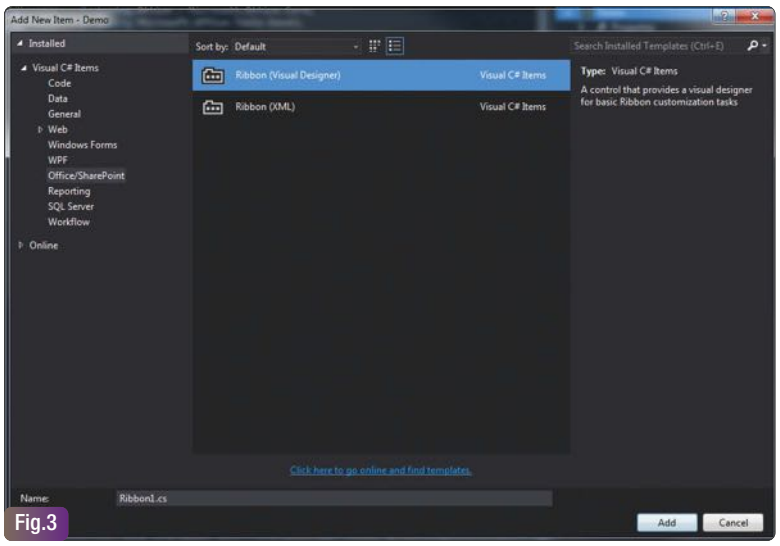


Fig.3

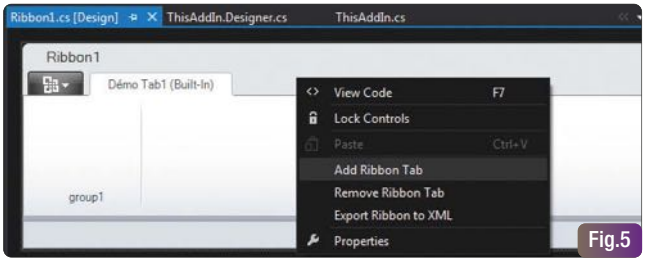


Fig.5

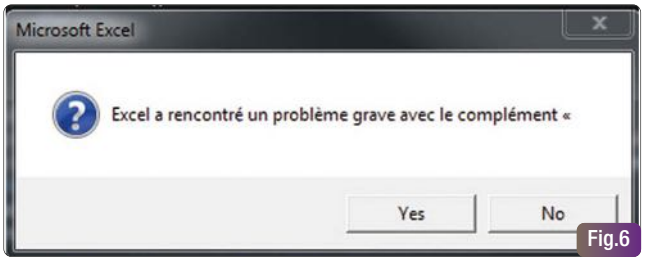


Fig.6

```
        //Range où se trouvent les données
        Excel.Range dataRange = Globals.ThisAddIn.Application.ActiveSheet.get_Range("A1","C3");

        //Transformation de l'objet vers Microsoft.Office.Tools.Excel voir (3.2)
        //qui permet d'accéder à la méthode permettant de gérer le graphique
        Tool.Worksheet worksheet = Globals.Factory.GetVstoObject(newWorksheet);

        //Range où se trouvera le graphique
        Excel.Range chartRange = newWorksheet.get_Range("A1", "L40");

        //On vérifie qu'il ne soit pas déjà présent
        if (worksheet.Controls.Contains(chartName))
        {
          worksheet.Controls.Remove(chartName);
        }

        //Ajout du graphique
        Tool.Chart chart = worksheet.Controls.AddChart(chartRange, chartName);

        //Définition du type de graphique
        chart.ChartType = Excel.XlChartType.xlColumnStacked;

        //Attribution des données du graphique
        //Choix des données sur l'axe X et Y
        chart.SetSourceData(dataRange, Excel.XlRowCol.xlRows);

    }
```

Pour définir le type du graphique, il faut attribuer la valeur de l'Enum que l'on désire à chart.ChartType. Les valeurs possibles se trouvent ici :
https://msdn.microsoft.com/en-us/library/microsoft.office.interop.excel.xlcharttype.aspx
Pour le SetSourceData, il faut passer en paramètre le range où se trouvent les données. En second paramètre, il faut définir si la série de données est en ligne ou en colonne afin de définir à quel axe correspondent les données. Pour ce faire les valeurs de l'Enum possibles sont soit xlColumns ou xlRows. Vous pouvez les retrouver ici :
https://msdn.microsoft.com/fr-fr/library/office/ff834335.aspx

3.6 – RibbonDropDown

Vous pouvez ajouter des contrôles dans votre Ribbon. Pour ce faire, ouvrez la boîte à outils, sélectionnez le contrôle que vous désirez et faites un drag & drop là où vous le désirez.

Exemple pour remplir une dropDown

```
IEnumerable<Car> cars = GetCar();
        foreach (Car car in cars)
        {
          RibbonDropDownItem item =  Globals.Factory.GetRibbonFactory().CreateRibbon
DropDownItem();
            item.Label = string.Format("{0} - {1}", car.ID, car.Color);
            item.Tag = car;
            this.dropDown1.Items.Add(item);
        }
```

RibbonDropDownItem étant une interface, il faudra passer par le Factory du Ribbon pour créer un item.
Dans la propriété, Tag étant de type object, je copie l'objet. Ainsi, il sera facile de retrouver quel est l'objet sélectionné :

```
Car selectedCar = this.dropDown1.SelectedItem.Tag as Car;
```

3.7 – Déploiement

Le déploiement se fait via clickOnce.
Dans mon cas nous avons utilisé un répertoire partagé où seuls les utilisateurs ayant besoin de l'Add-in auront accès.
Donc pour ce faire, faites un clic droit sur le projet et choisissez publish.
Dans le premier écran, il vous faudra renseigner le chemin du répertoire partagé accessible par les utilisateurs Fig.8.
La méthode CD-Rom – DVD-ROM suffira pour notre déploiement simple Fig.9. Ensuite, il vous suffira de cliquer sur finish Fig.10.
Pour que l'utilisateur puisse installer l'Add-In, il devra se rendre dans le répertoire partagé et cliquer sur le .exe.
L'installation se lancera automatiquement et une fenêtre demandera confirmation à l'utilisateur. Ensuite celui-ci préviendra également l'utilisateur lorsque l'installation sera terminée.
En utilisant ce type de déploiement, lorsque vous allez devoir publier une nouvelle version, l'utilisateur n'aura rien à faire. Lors du redémarrage d'Excel celui-ci se chargera de vérifier si une nouvelle version est disponible et l'installera. Cela est donc entièrement transparent pour l'utilisateur.
Pour voir toutes les possibilités sur le déploiement :
https://msdn.microsoft.com/en-us/library/bb772100.aspx
A travers ces trucs et astuces, vous pourrez plus facilement démarrer le développement d'un Add-In Excel.

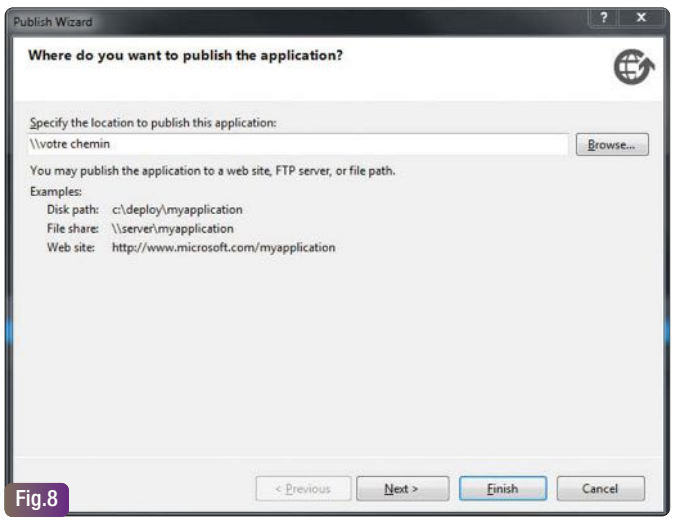


Fig.8

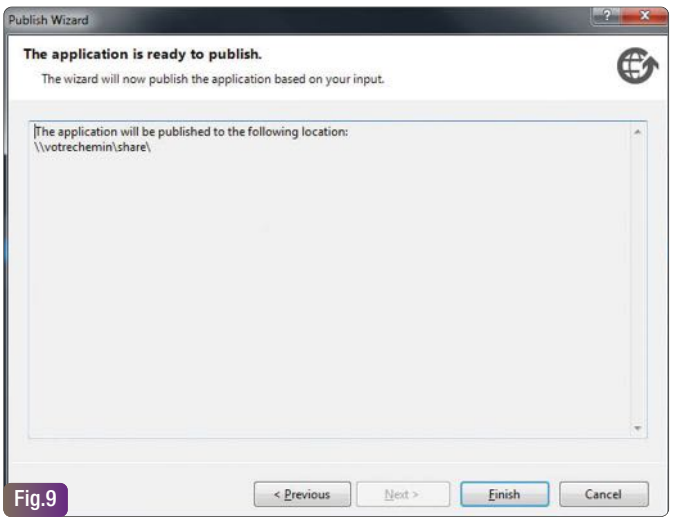


Fig.9

Scapy : un outil de sécurité incontournable

Scapy est un logiciel libre. Il est l'outil incontournable pour travailler en sécurité informatique, en protocole réseau ou pour l'étudiant qui souhaite comprendre et décortiquer les trames réseaux. Il faut une bonne compréhension du modèle TCP/IP car tout est bien entendu basé dessus. Il a été conçu par Philippe Biondi, la dernière version est la 2.3.2 du 15 janvier 2016.

Franck Ebel
Expert R&D et Formations
Serval-concept / serval-formation
Responsable Licence CDAISI , UVHC
Commandant de Gendarmerie réserviste,
cellule Cyberdéfense

Scapy est capable, entre autres, d'intercepter le trafic sur un segment réseau, de générer des paquets dans un nombre important de protocoles, de réaliser une prise d'empreinte de la pile TCP/IP, de faire un traceroute, d'analyser le réseau informatique...
Scapy n'est pas un outil clé en main (comme Nmap ou autre) mais un Framework basé sur Python fournissant un ensemble de fonctions pour interagir avec le réseau. Il a donc comme avantage d'avoir une grande liberté d'action car chaque paramètre peut être modifié et il décode mais n'interprète pas les paquets reçus.

Modèle TCP/IP

Le modèle TCP/IP est décomposé en 4 couches : la couche accès réseau, la couche Internet, la couche transport et la couche application.

- La couche accès réseau est typique de la technologie utilisée sur le réseau local, par exemple Ethernet, qui sera le plus fréquent. Ethernet est un protocole de réseau local à commutation de paquets. C'est une norme internationale : ISO/IEC 8802-3.
- La couche Internet réalise l'interconnexion des réseaux (hétérogènes) distants sans connexion. Son rôle est de permettre l'injection de paquets dans n'importe quel réseau et l'acheminement de ces paquets indépendamment les uns des autres jusqu'à destination (pour nous IP) .
- La couche transport permet à des entités paires de soutenir une conversation. Cette couche n'a que deux implémentations : le protocole TCP (Transmission Control Protocol) et le protocole UDP (User Datagram Protocol). TCP est un protocole fiable, orienté connexion, qui permet l'acheminement sans erreurs de paquets issus d'une machine d'un Internet à une autre machine du même Internet. Son rôle est de fragmenter le message à transmettre de manière à pouvoir le faire passer sur la couche Internet.
  A l'inverse, sur la machine destination, TCP replace dans l'ordre les fragments transmis sur la couche Internet pour reconstruire le message initial. TCP s'occupe également du contrôle de flux de la connexion. UDP est en revanche un protocole plus simple que TCP : il est non fiable et sans connexion. Son utilisation présuppose que l'on n'ait pas besoin ni du contrôle de flux, ni de la conservation de l'ordre de remise des paquets. Par exemple, on l'utilise lorsque la couche application se charge de la remise en ordre des messages.
- La couche application contient tous les protocoles de haut niveau, comme par exemple Telnet, TFTP (trivial File Transfer Protocol), SMTP (Simple Mail Transfer Protocol), HTTP (HyperText Transfer Protocol) ...

Donc si nous résumons, une trame sera de la forme, par exemple : Ethernet/IP/TCP/FTP ou Ethernet/IP/UDP/DNS.
Ceci a une importance pour comprendre la suite de l'article.

Prise en main de scapy

Scapy va permettre de travailler sur de nombreux protocoles, il suffit pour connaître ces différents protocoles de lancer Scapy et dans le prompt scapy de lancer la commande ls().

```
franck:~ franckebel$ sudo scapy
WARNING: No route found for IPv6 destination :: (no default route?)
Welcome to Scapy (2.3.1)
>>> ls()
AH       : AH
ARP      : ARP
ASN1_Packet : None
BOOTP    : BOOTP
CookedLinux : cooked linux
DHCP     : DHCP options
DHCP6    : DHCPv6 Generic Message)
DHCP6OptAuth : DHCP6 Option - Authentication
DHCP6OptBCMCSDomains : DHCP6 Option - BCMCS Domain Name List
DHCP6OptBCMCSServers : DHCP6 Option - BCMCS Addresses List
DHCP6OptClientFQDN : DHCP6 Option - Client FQDN
DHCP6OptClientId : DHCP6 Client Identifier Option
DHCP6OptDNSDomains : DHCP6 Option - Domain Search List option
....
```

Le résultat donné est bien sûr tronqué mais vous pourrez aisément vous rendre compte sur votre ordinateur personnel de tous les protocoles supportés et des possibilités immenses de Scapy.
De la même manière nous pouvons regarder les commandes de base intégrées à Scapy grâce à la commande lsc().

```
>>> lsc()
arpcachepoison     : Poison target's cache with (your MAC,victim's IP) couple
arping             : Send ARP who-has requests to determine which hosts are up
bind_layers        : Bind 2 layers on some specific fields' values
bridge_and_sniff   : Forward traffic between two interfaces and sniff packets exchanged
corrupt_bits       : Flip a given percentage or number of bits from a string
corrupt_bytes      : Corrupt a given percentage or number of bytes from a string
defrag             : defrag(plist) -> ([not fragmented], [defragmented],
defragment         : defrag(plist) -> plist defragmented as much as possible
dyndns_add         : Send a DNS add message to a nameserver for "name" to have a new "rdata"
dyndns_del         : Send a DNS delete message to a nameserver for "name"
etherleak          : Exploit Etherleak flaw
fragment           : Fragment a big IP datagram
```

Si nous souhaitons avoir une aide sur les commandes par exemple, nous pouvons utiliser help().

```
>>> help(arpcachepoison)
Help on function arpcachepoison in module scapy.layers.l2:

arpcachepoison(target, victim, interval=60)
```

```
Poison target's cache with (your MAC,victim's IP) couple
arpcachepoison(target, victim, [interval=60]) -> None
```

Le résultat de la commande lsc() est lui aussi tronqué. Nous utiliserons certaines de ces commandes dans un autre article. Nous allons dans celui-ci nous concentrer sur la base de Scapy et la façon de forger des paquets. Pour obtenir des informations (méthodes associées, utilisation) sur un protocole, par exemple IP, nous utiliserons la commande dir().

```
>>> dir(IP)
['__class__', '__contains__', '__delattr__', '__delitem__', '__dict__', '__div__',
'__doc__', '__eq__', '__format__', '__getattr__', '__getattribute__', '__getitem__',
'__gt__', '__hash__', '__init__', '__iter__', '__len__', '__lt__', '__metaclass__',
'__module__', '__mul__', '__ne__', '__new__', '__nonzero__', '__rdiv__',
'__reduce__', '__reduce_ex__', '__repr__', '__rmul__', '__rtruediv__', '__setattr__',
'__setitem__', '__sizeof__', '__str__', '__subclasshook__', '__truediv__', '__weakref__',
'_do_summary', 'add_payload', 'add_underlayer', 'aliastypes', 'answers', 'build', 'build
_done', 'build_padding', 'build_ps', 'canvas_dump', 'clone_with', 'command', 'copy', 'decode
_payload_as', 'default_payload_class', 'delfieldval', 'display', 'dissect', 'dissection_done',
'do_build', 'do_build_payload', 'do_build_ps', 'do_dissect', 'do_dissect_payload', 'do_
init_fields', 'explicit', 'extract_padding', 'fields_desc', 'firstlayer', 'fragment', 'from_hexcap',
'get_field', 'getfield_and_val', 'getfieldval', 'getlayer', 'guess_payload_class', 'hashret',
'haslayer', 'hide_defaults', 'hops', 'init_fields', 'initialized', 'lastlayer', 'libnet', 'lower_bonds',
'mysummary', 'name', 'ottl', 'overload_fields', 'payload_guess', 'pdfdump', 'post_build',
'post_dissect', 'post_dissection', 'pre_dissect', 'psdump', 'raw_packet_cache', 'remove_
payload', 'remove_underlayer', 'route', 'self_build', 'send', 'sent_time', 'setfieldval', 'show',
'show2', 'show_indent', 'sprintf', 'summary', 'underlayer', 'upper_bonds', 'whois']
>>>
```

Nous voyons que par exemple IP comporte send, show, summary, haslayer entre autres. Nous verrons certaines de ces commandes dans divers exemples dans cet article et les suivants.

Fabrication de trames

Nous avons vu à la fin de la partie "Modéle TCP/IP" que la trame serait de la forme Ethernet/IP/TCP/FTP. Nous allons voir que nous aurons exactement cette "forme" avec Scapy. Nous allons déjà regarder les options des différentes couches Ethernet, IP et TCP.

```
>>> ls(Ether)
dst       : DestMACField      = (None)
src       : SourceMACField    = (None)
type      : XShortEnumField   = (36864)
```

Des valeurs par défaut sont déjà incluses dans la configuration et en utilisant la commande ls() nous pouvons en voir les différentes options. Par exemple pour l'Ethernet, Ether sous Scapy, le champ type est renseigné à 36864 et les champs dst et src à None. Nous pouvons bien sûr mettre les valeurs que nous désirons dans ces champs.

```
>>> ma_trame=Ether(dst='00:01:02:03:04:05')
>>> ma_trame.show()
###[ Ethernet ]###
  dst= 00:01:02:03:04:05
  src= 00:00:00:00:00:00
  type= 0x9000
>>>
```

Nous avons ici imposé l'adresse MAC 00:01:02:03:04:05 au champ dst. Et nous pouvons nous assurer que celui-ci a bien été pris en compte grâce à la commande show(). Regardons ce que nous avons pour IP et TCP :

```
>>> ls(IP)
version   : BitField          = (4)
ihl       : BitField          = (None)
tos       : XByteField        = (0)
len       : ShortField        = (None)
id        : ShortField        = (1)
flags     : FlagsField        = (0)
frag      : BitField          = (0)
ttl       : ByteField         = (64)
proto     : ByteEnumField     = (0)
chksum    : XShortField       = (None)
src       : Emph              = (None)
dst       : Emph              = ('127.0.0.1')
options   : PacketListField   = ([])
>>> ls(TCP)
sport     : ShortEnumField    = (20)
dport     : ShortEnumField    = (80)
seq       : IntField          = (0)
ack       : IntField          = (0)
dataofs   : BitField          = (None)
reserved  : BitField          = (0)
flags     : FlagsField        = (2)
window    : ShortField        = (8192)
chksum    : XShortField       = (None)
urgptr    : ShortField        = (0)
options   : TCPOptionsField   = ({})
>>>
```

Construisons maintenant une trame complète.
Comment ferions-nous si nous voulions envoyer une commande GET à un site Web ?
Il faut que nous encapsulions les différentes couches du modèle TCP/IP.
Pour cela nous utiliserons le symbole "/".
Nous allons donc encapsuler la couche Ethernet, IP, TCP et la commande (Application) "GET HTTP/1.0\r\n\r\n".
Pour connaître les commandes associées aux différents protocoles, il suffit d'aller lire la RFC correspondante, par exemple ici HTML (RFC 2616 : https://tools.ietf.org/html/rfc2616)

```
>>> ma_trame=Ether()/IP()/TCP()/"GET HTTP/1.0\r\n\r\n"
>>> ma_trame.show()
###[ Ethernet ]###
WARNING: Unable to guess datalink type (interface=lo0 linktype=0). Using Raw
WARNING: Unable to guess datalink type (interface=lo0 linktype=0). Using Raw
WARNING: more Unable to guess datalink type (interface=lo0 linktype=0). Using Raw
WARNING: Mac address to reach destination not found. Using broadcast.
  dst= ff:ff:ff:ff:ff:ff
  src= 00:00:00:00:00:00
  type= 0x800
###[ IP ]###
    version= 4
    ihl= None
    tos= 0x0
    len= None
    id= 1
    flags=
```

```
    frag= 0
    ttl= 64
    proto= tcp
    chksum= None
    src= 127.0.0.1
    dst= 127.0.0.1
    \options\
###[ TCP ]###
      sport= ftp_data
      dport= http
      seq= 0
      ack= 0
      dataofs= None
      reserved= 0
      flags= S
      window= 8192
      chksum= None
      urgptr= 0
      options= {}
###[ Raw ]###
        load= 'GET HTTP/1.0\r\n\r\n'
>>>
```

Nous avons bien entendu des Warning car nous n'avons renseigné aucune option, et par ailleurs, dst et src de Ether sont à None.
Complétons donc notre trame en indiquant le site de destination, ici www.programmez.com.

```
>>> ma_trame=Ether()/IP(dst="www.programmez.com")/TCP()/"GET HTTP/1.0\r\n\r\n"
>>> ma_trame.show()
###[ Ethernet ]###
  dst= ec:9b:f3:f5:50:82
  src= 3c:15:c2:d9:6a:ce
  type= 0x800
###[ IP ]###
    version= 4
    ihl= None
    tos= 0x0
    len= None
    id= 1
    flags=
    frag= 0
    ttl= 64
    proto= tcp
    chksum= None
    src= 192.168.43.189
    dst= Net('www.programmez.com')
    \options\
###[ TCP ]###
      sport= ftp_data
      dport= http
      seq= 0
      ack= 0
      dataofs= None
      reserved= 0
      flags= S
      window= 8192
      chksum= None
      urgptr= 0
      options= {}
###[ Raw ]###
        load= 'GET HTTP/1.0\r\n\r\n'
>>>
```

Nous constatons maintenant que, contrairement à tout à l'heure certains champs se sont remplis d'eux-mêmes dans IP (la source) et dans Ether (la source, la destination et le type est passé à 0x800 puisque la couche du dessus est IP qui correspond à ce numéro).
Nous pouvons maintenant réellement envoyer cette trame que nous avons formée vers le site grâce à la commande send().

```
>>> send(ma_trame)
WARNING: Mac address to reach destination not found. Using broadcast.
.
Sent 1 packets.
>>>
```

Nous voyons qu'un paquet est bien parti. Nous n'avons pas de retour pour l'instant. Appliquons la même chose à une requête ICMP.

```
>>> p=Ether()/IP(dst="www.programmez.com")/ICMP()
>>> p.summary()
"Ether / IP / ICMP 192.168.43.189 > Net('www.programmez.com') echo-request 0"
>>> send(p)
WARNING: Mac address to reach destination not found. Using broadcast.
.
Sent 1 packets.
>>>
p.summary() nous permet de savoir comment nous avons forgé notre paquet.
```

Conclusion

Nous venons de découvrir Scapy et la base de son utilisation. Nous verrons dans de prochains articles la façon de recevoir la réponse des trames envoyées et de créer des boucles. Nous pourrons ensuite appliquer notre apprentissage en créant des applications réelles pour les tests réseaux, pour ensuite s'attaquer à des scripts spécifiques à la sécurité et finir par l'utilisation des commandes de scapy telles que traceroute ou arpcache-poison.

À lire dans le prochain numéro n°198 en kiosque le 30 juin 2016

Retour sur la conférence
Devoxx France 2016

Les devoirs d'été
- Réalité virtuelle avec CardBoard et Hololens
- Créer son jeu vidéo avec BabylonJS
- Hacker sa Tesla
- Comment créer un langage de programmation ?

.Net devient (presque)
open source

Totalement indispensable

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 : François Tonic
Rédacteur en chef : François Tonic

Secrétaire de rédaction : Olivier Pavie

Ont collaboré à ce numéro : S. Saurel

Nos experts : F. Botte, P. Batty, E. Manuguerra, Y. Danot, M. Amokhtari, O. Laplace, F. Dibot, R. Squelbut, W. Ammar, J. Corioland, C. Leblond, P-H. Cache, V. Thavonekham, J. Antoine, D. Djordjevic, G. Deruette, B. Briatte, M. Avataneo, C. Michel, F. Ebel, D. joubert, M. Hubert.

Couverture : © StockFinland

Maquette : Pierre Sandré

Publicité : PC Presse,
Tél.: 01 74 70 16 30, Fax : 01 40 90 70 81
pub@programmez.com

Imprimeur : S.A. Corelio Nevada Printing, 30 allée de la recherche, 1070 Bruxelles, Belgique.

Marketing et promotion des ventes :
Agence BOCONSEIL - Analyse Media Etude

Directeur : Otto BORSCHA oborscha@boconseilame.fr
Responsable titre : Terry MATTARD
Téléphone : 09 67 32 09 34

Contacts
Rédacteur en chef :
ftonic@programmez.com
Rédaction : redaction@programmez.com
Webmaster : webmaster@programmez.com
Publicité : pub@programmez.com
Evenements / agenda :
redaction@programmez.com

Dépôt légal : à parution - Commission paritaire : 1220K78366 - ISSN : 1627-0908

© NEFER-IT / Programmez, mai 2016
Toute reproduction intégrale ou partielle est interdite sans accord des auteurs et du directeur de la publication.

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

Jusqu'au 4 Juillet 2016

COMMANDEZ WINDEV MOBILE 21

OU WEBDEV 21 OU WINDEV 21

ET RECEVEZ LE NOUVEL iPhone 6S

Ou choisissez parmi:

- 1x **iPad Pro** 9,7"WiFi 128GB
- 1x **iPad Pro** 12,9" WiFi 32GB
- 1x **MacBook Air** 11,6" 128GB
- 2x **iPhone SE** 16GB
- 2x **iPad Mini 4** Wi-Fi 64GB
- 2x **iPad Air 2** Wi-Fi 16GB

D'autres matériels sont proposés sur le site www.pcsoft.fr
Aucun abonnement à souscrire.

iPhone 6S **128GB**.
Choix de la couleur sur le site

ou

iPhone 6S Plus **64GB**.
Choix de la couleur sur le site

OPÉRATION POUR 1 EURO DE PLUS

Pour bénéficier de cette offre exceptionnelle, il suffit de commander WINDEV Mobile 21 (ou WINDEV 21, ou WEBDEV 21) chez PC SOFT au tarif catalogue avant le 4 Juillet 2016. Pour 1 Euro de plus, vous recevrez alors le ou les magnifiques matériels que vous aurez choisis. Offre réservée aux sociétés, administrations, mairies, GIE et professions libérales, en France métropolitaine. **L'offre s'applique sur le tarif catalogue uniquement**. Voir tous les détails et des vidéos sur : **www.pcsoft.fr** ou appelez-nous (04.67.032.032).
Le Logiciel et le matériel peuvent être acquis séparément. Tarif du Logiciel au prix catalogue de 1.650 Euros HT (1.973,40 TTC). Merci de vous connecter au site www.pcsoft.fr pour consulter la liste des prix des matériels et les dates de disponibilité. Tarifs modifiables sans préavis.

Elu
«Langage le plus productif du marché»

www.pcsoft.fr

Document non contractuel. Logiciel professionnel. Apple® iPhone® iPad® iPad Air® iPad Mini™ sont des marques déposées de la société Apple. Apple n'est pas un organisateur ou un sponsor de cette opération.