PROGRAMMEZ.COM

PROGRAMMEZ!

Mensuel n°190 - Novembre 2015

le magazine du développeur

App Store : qui veut gagner des millions ?

Faut-il passer à Drupal 8 ?

Office 365

Les développeurs à la fête !

Carrière Cobol,

un vieux langage toujours vivant !

Photo by Pictorial Parade/Archive Photos

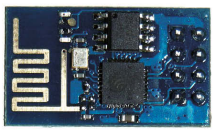
Mouvement Craftsmanship

Pour un code de qualité et bien documenté !

Mission : MAKER

ESP8266 : puissant, petit, WiFi inclus pour 2€

Test de la carte MicroView



M 04319 - 190 - 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

ikoula
HÉBERGEUR CLOUD

TROPHÉES DUCLOUD by EuroCloud
MEILLEUR SERVICE CLOUD D'INFRASTRUCTURE 2015

LE CLOUD GAULOIS, UNE RÉALITÉ ! VENEZ TESTER SA PUISSANCE

EXPRESS HOSTING

Cloud Public
Serveur Virtuel
Serveur Dédié
Nom de domaine
Hébergement Web

sales@ikoula.com
01 84 01 02 66
express.ikoula.com

ENTERPRISE SERVICES

Cloud Privé
Infogérance
PRA/PCA
Haute disponibilité
Datacenter

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

EX10

Cloud Hybride
Exchange
Lync
Sharepoint
Plateforme Collaborative

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

PROGRAMMEZ!
le magazine du développeur

“La seule voie pour faire un grand travail est d’aimer ce que l’on fait. Si vous ne l’avez pas trouvée, continuez à (la) chercher”

Steve Jobs

À lire dans le prochain numéro n°191 en kiosque le 30 Novembre 2015

Dossier école
Quelle école informatique choisir ?
Quels cursus ? Quels critères ?

Open Data
Comment utiliser les données ouvertes ?
Comment les intégrer dans son application ?

Mobile :
Découvrons le développement mobile avec Xamarin

Surface Book,

même les fanboys Mac sont impressionnés. Et vous ?

30 ans.

L'âge du premier compilateur C++

57% des terminaux iOS

ont migré sur la v9 du système en 3 semaines. iOS 8 pèse 33 %

87%.

Le % de terminaux Android vulnérables ! (étude de Cambridge)

Surprise ! Zend,

un des leaders des outils PHP, a été racheté par Rogue Wave, connu pour ces outils C++ et HPC...

Rumeur : Intel

pourrait fournir la future puce réseau des prochains iPhone

Salaires 2015

Urban Linker publie son nouveau rapport sur les salaires du monde web et informatique.

langage/profil	-2 ans	2 – 4 ans	4-6	Lead/architecture
PHP avec compétences Zend/CMS	32-40 +	40-45 =	45-50 =	48-60+ +
.net	33-42 =	42-46 =	47-55 +	50-70 =
Java	33-42 =	42-47 +	47-54 +	50-70 =
Python	33-40	38-44	45-55	50-60+
Ruby	34-46 +	47-52 +	52-57 +	53-70
Mobile natif	35-45 +	45-52 +	52-62 +	60-85 +
Mobile hybride	32-45	42-45	45-55	
intégrateur HTML	28-36 -	36-42 -	42-45 =	
fullstack JS	36-45	46-52 +	52-60	53-70+

Salaire moyen en milliers €/an/brut. Attention : certains salaires estimés sont uniquement sur la région Île-de-France.
+/– /= : tendance générale du salaire donnée par Urban Linker
Tous les détails : http://www.urbanlinker.com/salaire-moyen/salaire-technique-2015/

INDEX TIOBE :
Objective-C s'écroule, vive Swift ?

Oct 2015	Oct 2014	évolution	langage	%	Changement
1	2	↑	Java	19.543 %	+6.04 %
2	1	↓	C	16.190 %	-1.47 %
3	4	↑	C++	5.749 %	+0.88 %
4	5	↑	C#	4.825 %	+0.08 %
5	8	↑	Python	4.512 %	+2.18 %
6	7	↑	PHP	2.561 %	-0.38 %
7	13	⇈	Visual Basic .NET	2.462 %	+0.71 %
8	12	⇈	JavaScript	2.292 %	+0.52 %
9	9		Perl	2.247 %	+0.13 %
10	16	⇈	Ruby	1.825 %	+0.70 %

Depuis plusieurs semaines, Objective-C tombe au classement. Il est sorti du top 10 et se retrouve à une piteuse 14e place. Il y a 1 an, le langage était 3e... Swift, le nouveau langage de la Pomme est désormais 15e et pourrait très rapidement dépasser son « ancêtre ». Java retrouve la 1ere place. C# fait un retour en force et Python confirme sa bonne tendance auprès des développeurs.
Rappel : l'index TIOBE donne des % selon les recherches des développeurs sur 25 moteurs de recherche. L'index fournit une popularité, les recherches à instant précis, mais ne donne pas l'utilisation réelle des langages.

Et vogue le kernel...

Octobre a été un peu agité dans le monde du noyau Linux. Matthew Garret, un des développeurs du noyau, avait annoncé qu'il avait créé sa branche du noyau. Mais pour nos confrères de ZDNet.com, lorsque Matthew a réalisé ce travail, il ne s'agissait pas de faire un fork Linux. L'origine du problème concerne le secure boot (démarrage sécurisé), présent dans les firmware UEFI. Matthew voulait implémenter un système identique au BSD securelevel dans Linux pour créer une fonctionnalité identique au Secure Boot utilisé par Microsoft. Fork ou pas fork ? Visiblement Matthew n'a jamais présenté son travail comme un fork Linux. Cette « affaire » fait suite au départ de Sarah Sharp de la communauté des contributeurs au noyau et son long post sur les raisons de son départ. Elle évoque l'attitude de certains développeurs et de l'ambiance. À lire ici : http://sarah.thesharps.us/2015/10/05/closing-a-door/
Le « fork » de Matthew : https://github.com/mjg59/linux

42 % des nouveaux développeurs sont des femmes

Voilà un titre qui nous a intrigués. C'est ainsi que démarre l'article publié par Le Figaro Madame du 7 septembre dernier. Un graphique de l'étude confirme cela et les analystes s'attendent à un accroissement même. Mais cette étude est au niveau mondial sans fournir les % par pays. Le problème est que nous ne voyons pas cette tendance dans les écoles et les conférences auxquelles nous assistons. Dans les cursus informatiques/développement, les femmes sont toujours très minoritaires, 10-15 %.

2016 : un clone de l'école 42

ouvrira en Afrique du Sud. Une jeune Française en est à l'origine.

Des Français

ouvrent une école informatique à San Francisco

COMMANDEZ WINDEV 21

OU WEBDEV 21 OU WINDEV MOBILE 21

ET RECEVEZ UN SUPERBE MATÉRIEL

OPÉRATION POUR 1 EURO de plus

Jusqu'au 11 Décembre 2015

TV SAMSUNG

140cm 4K
3.840 x 2.160
Wi-Fi

Réf: UE55JU6000

Ou choisissez :

- TV HD 3D Wi-Fi 140 cm Réf UE55H6400

ORDINATEUR DELL

17'' Core i7 tactile 2To 12Go

Ou choisissez parmi:

- Station de travail **DELL Précision T1700** Mini-tour Disque 500Go Mémoire 8Go Windows 7 Pro + 8.1 Pro (MAJ 10 incluse)
- «All in One» PC **DELL 23" Inspiron 5348** Windows 8.1 (MAJ 10 incluse) Intel Core i5 Disque 1 To - tactile

D'autres matériels sont proposés sur le site www.pcsoft.fr

Aucun abonnement à souscrire. Compatible tous opérateurs

SAMSUNG GALAXY S6 EDGE

5,1'' 64Go
Bords incurvés
Android

Ou choisissez parmi:

- 1x **Galaxy S6 Edge+** 32 Go
- 2x **Galaxy S5 New**
- 2x Tablettes **Galaxy Tab 2S** 9,7"

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 11 Décembre 2015: 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.

www.pcsoft.fr

Elu **«Langage le plus productif du marché»**

Fournisseur Officiel de la Préparation Olympique

Document non contractuel. Logiciel professionnel. Version annoncée.

Belgique

Dev Day : 1er décembre (Mons, Belgique)

L'événement développeur belge se tiendra le 1er décembre à Mons. L'événement sera très orienté sur les technologies Microsoft et Windows 10 mais aussi Raspberry Pi (+ Windows 10). A cette occasion des speakers de renom seront présents : David Rousset, Sébastien Warin, Samuel Blanchard...
L'agenda est alléchant et la journée bien équilibrée entre le web, le mobile et les IoT.
Pour les étudiants, l'événement est gratuit.
Pour en savoir plus : http://www.devday.be

TechDays Camps : octobre / novembre

Microsoft organise une série de conférences dans toute la France. Ces journées s'organisent autour de 4 thèmes : Windows 10, développement web, Données et IT, avec notamment Azure, Office 365, Windows Server.
Les dates à venir : Strasbourg : 5 novembre, Paris : 26 novembre.
Pour en savoir et inscription :
https://techdays.microsoft.fr/camp/default.aspx#pg1

DevFest : 6 novembre à Nantes

Le GDG Nantes organise, le 6 novembre, le DevFest 2015 à la Cité des Congrès de Nantes. Pour cette 4eme édition, l'accent sera mis sur les nouvelles technologies et l'innovation : il sera question des dernières avancées en termes de Cloud, Web, Mobile & IoT. Des experts locaux et internationaux seront présents :

- Francesc Campoy Flores, équipe Go chez Google
- Raphaël Goetter, expert web et créateur d'Alsacréations
- Cyril Mottier, Google Developer Expert Android chez Capitaine train
- Goeffrey Dorne, Expert UX chez Human & Design

Ce DevFest est le rassemblement français autour des technologies Google avec plus de 700 participants. Cet évènement soutenu par Google, Intel... et plus d'une vingtaine de partenaires locaux est labelisé NantesTech. Site Web : http://devfest.gdgnantes.com

Droidcon Paris 2015 : 9 & 10 novembre

Dans quelques jours, la Droidcon Paris sera de retour au Tapis Rouge pour sa troisième édition consécutive ! Les 3 thèmes majeurs de cette année seront Android Development, Android Everywhere avec tous les objets connectés et enfin UX/UI pour ne pas oublier le design. 50 speakers experts Android et plus de 600 participants seront présents pour cette conférence, la plus grande sur Android en France.
L'événement se rapproche à grands pas et bientôt, le programme complet sera en ligne sur le site droidcon.fr. En attendant, voici un petit aperçu de quelques experts très attendus cette année et de leur conférence :
Svetlana Isakova (JetBrains) travaille sur le langage Kotlin (http://kotlinlang.org/) Elle est l'un des auteurs de « Kotlin en Action » et prend part à de nombreuses conférences. Elle présentera d'abord les avantages du langage Kotlin comparé aux autres langages JVM puis comment certaines structures de langage/ language constructs pourront être utilisés pour rendre Android plus agréable. Elle présentera également un DSL for type-sage dynamic layouts et un plugin permettant d'éviter la répétition systématique de «findViewById ()'.
Corey Latislaw abordera le thème Katas, la meilleure façon de s'entraîner au développement piloté par les tests (TDD). Petits programmes ou exercices à répéter sans cesse, cela vous permettra d'améliorer votre rapidité de codage d'applications mais aussi vous maîtriserez entièrement les principaux types de tests. Florian Mierzejewski, développeur Android a récemment rejoint l'équipe de Novoda. Volker Leck, quant à lui, est un vétéran de Java et un pionnier de la plateforme Android. Il travaille sur de nombreuses applications Android chez Novoda.
Leur conférence fournira quelques astuces et techniques poussées des plugins Gradle. Quatrième tête d'affiche pour l'événement, Alexis Fogel, co-fondateur de Dashlane et responsable des produits. Alexis Fogel est expert en interface utilisateur (UI) et expérience utilisateur (UX).
Sa conférence présentera la nouvelle fonctionnalité « Experiements » du Play Store et comment en tirer le meilleur parti. Il expliquera comment cette fonctionnalité va permettre un retour sur investissement plus important comparé aux tests in-app.
Enfin, Cyril Mottier, développeur Android chez Capitaine Train, traitera des techniques avancées de scroll sur Android.
Les scrolls containters View, ListView, RecyclerView et ScrollView, livrés dans le SDK Android et assez simples d'utilisation, deviennent plus difficiles à gérer lorsqu'on les imbrique. Cyril Mottier expliquera donc comment d'une manière générale, Android gère le défilement ainsi que la façon de maîtriser le mécanisme de défilement afin d'assurer, sur mobile, une expérience utilisateur convaincante. Vous souhaitez y participer ? Vite, prenez votre place à 50 euros (offre exceptionnelle à durée limitée) et saisissez le code promo « Programmez_droidcon2015 » sur droidcon.fr !

Paris Open Source Summit : 18 & 19 novembre (Paris)

Solutions Linux, le salon Linux et Open Source, et Open World forum fusionnent pour donner un unique grand événement Open Source français : Paris Open Source Summit. Il s'agit de concentrer les efforts et de faire de Paris une place incontournable du monde Open Source et Linux en Europe.

Forum PHP : 23 & 24 novembre (près de Paris)

Le monde PHP fête les 20 ans du langage et de la sortie de 7 PHP. Le Forum PHP est la référence française des développeurs et entreprises utilisant ce langage open source ! Cette année, 2 formats de conférences seront proposés : format 40 minutes et format 20 minutes.
Un des thèmes abordés sera 7 PHP qui sortira presque en même temps. Une track dédiée sera proposée et le sujet promet d'être très largement abordé ! Site : http://www.afup.org/pages/forumphp2015/

LabVIEW Developer Days (National Instrument), du 17 au 26 novembre dans 8 villes de France.

Cette journée est dédiée aux programmeurs et aux développeurs LabVIEW. Objectif : renforcer vos compétences techniques et vos connaissances de la plate-forme NI au travers de nombreuses sessions techniques : des principes de conception logicielle pour créer du code modulaire aux modèles de conception, en passant par les bonnes pratiques de programmation qui vous assureront la stabilité et la maintenabilité de vos applications à long terme.
17 novembre - Paris, 17 novembre - Rennes, 19 novembre - Rouen, 19 novembre - Strasbourg, 24 novembre - Marseille, 24 novembre - Toulouse, 26 novembre - Bordeaux, 26 novembre - Grenoble
Inscription gratuite et infos sur http://france.ni.com/labview-developer-days

Vous croyez **encore que le** *Cloud Computing* **est cher ?**

Avec ArubaCloud,

Vous avez accès à une large gamme de solutions de Cloud Computing, avec choix du pays du datacenter utilisé, solution packagée ou flexible avec facturation adaptée..
Vous pouvez dès maintenant créer votre serveur Cloud SMART à partir de 1€ht / mois.

*Prix Hors Taxe, vérifiez la liste des prix pour plus d'informations

**MON PAYS, MON CLOUD.

Hyperviseur vmware

Contrôle des coûts

6 datacenters en Europe

APIs et connecteurs

Linux & Windows

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

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

aruba CLOUD

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

MY COUNTRY. MY CLOUD.**

Cobol, un langage toujours vivant !

Cobol fait partie des dinosaures de l'informatique. Son origine remonte à la fin des années 1950 ! Dès l'origine, il était dédié à la programmation d'applications de gestion. Depuis la première version en 1959, il n'a cessé d'évoluer et fut rapidement standardisé et normalisé. La dernière version est Cobol 2014. Le langage est même 20e au classement TIOBE ! Cobol n'est pas un langage aussi mort que ce que l'on pourrait croire. Il demeure le langage historique, et incontournable de la finance/banque/assurance, et des administrations publiques. Une compétence Cobol peut-elle vous servir ? Faut-il se former ? Qui sont les acteurs majeurs du monde Cobol ? Comment évoluent les outils ? Faisons le point.

François Tonic
Programmez!

En préambule, quelques chiffres parlent d'eux-mêmes. Cobol c'est avant tout un héritage étalé sur 50 ans :

- + de 220 milliards de lignes de code
- 70 – 80 % des transactions bancaires
- + de 1,5 milliard de nouveaux codes (et des millions de lignes disparaissent chaque année).

Pour maintenir les applications et continuer l'exploitation de ces logiciels, il faut chaque année, supprimer et créer du code. Le monde bancaire demeure fortement Cobol/Mainframe. Cobol peut se comparer à d'autres langages : Fortran dans le monde scientifique, LISP/Haskel dans l'intelligence artificielle.

Une image vieillotte du coboliste et un désintérêt des écoles !

Dans la dernière étude Cobol de MicroFocus, la démographie des développeurs Cobol est très marquée :

- 45 % des développeurs ont 45 ans et +
- 54 % des développeurs ont – 45 ans

Les jeunes cobolistes (-35 ans) sont minoritaires : 11 % en moyenne des effectifs. Dans certains pays, l'âge moyen est plus élevé ce qui pose un défi sans précédent aux entreprises pour trouver de nouvelles compétences. Les départs en retraite ont un impact direct sur la connaissance et la bonne gestion des applications Cobol. Cependant, il faut nuancer. Comme le rappelle Stéphane Croce de Cobol-IT, cela fait 25 ans que l'on parle de ce problème. « C'est un sujet réchauffé. Le développement Cobol a évolué » rajoute Stéphane. Se focaliser sur ce problème serait retourner dans les années 1980-1990 et le développement sur mainframe et les profils de développeurs, d'analystes-développeurs. « Le métier a beaucoup évolué », poursuit Stéphane. Aujourd'hui, un « coboliste » doit avoir une culture multiforme et de nombreuses compétences, car l'univers IT est devenu extrêmement hétérogène.

Une double compétence plutôt qu'un profil purement cobol

Nous ne parlons pas de devenir coboliste et de faire carrière dans le Cobol/mainframe. Même si le marché existe et existera encore longtemps, ce profil est peu recherché. Par contre, une compétence Cobol peut, dans certains secteurs, être un plus. Le monde Mainframe n'est plus isolé. Il y a 15 ans, on parlait de Web to Host, aujourd'hui la modernisation des applications mainframes s'accélère, mais il ne s'agit pas de tout réécrire. Beaucoup de projets consistent à créer des frontaux modernes avec des langages et des technologies actuelles ; en revanche le back-end demeure en Cobol/mainframe. « Les applications sont plus ouvertes. Il s'agit aussi de savoir comment se rapprocher des nouvelles applications » précise Stéphane Croce. La modernisation est une réalité ainsi que la rénovation applicative. Nous avons fait une recherche de développeurs et responsables Cobol/mainframe début octobre sur différents sites d'emplois (offres en CDI, CDD, intérim) :

- lesjeudis.com : environ 12 annonces
- Pôle Emploi : impossible à déterminer
- Monster.fr : - de 18 offres
- Indeed.fr : ce site permet d'avoir une vue large des offres. Plusieurs centaines d'offres récentes ou anciennes répertoriées par de nombreux sites d'emplois.

Par rapport à notre précédent dossier Cobol (décembre 2013), les chiffres ont peu évolué. Le secteur bancaire est le principal demandeur de profils et compétences Cobol/mainframe.

À titre de comparaison, sur monster.fr, + de 300 offres concernent Java, + de 400 profils Java sur lesjeudis.com.

Les formations Cobol ne manquent pas. Une simple requête Google le prouve. La plupart sont proposées par des formateurs ou sociétés de services informatiques. Quelques formations étaient/sont proposées par des écoles/universités (ex. : Université de Nantes). Mais les écoles d'informatique et les universités, en France, sont très discrètes sur ces technologies. Ces postes ne sont pas forcément remplacés et surtout, il y a un risque réel de perte de connaissances. Comme nous l'a indiqué Compuware, si une entreprise manque de compétences en interne, elles seront externalisées en offshore.

Impact du mainframe en France

Une étude menée par IDC (décembre 2013), « l'impact économique du Mainframe en France », montre le poids de ces systèmes et il demeure incontournable :

- 1,5 milliard € en 2013 : dépense pour les services, matériels et logiciels
- 13 % : pourcentage estimé de l'économie française qui repose sur du Mainframe
- Un réseau de + 300 partenaires
- 30 000 personnes travaillent sur et autour du Mainframe
- 59 % des sociétés travaillant dans le Mainframe disent avoir du mal à trouver des compétences
- 65 % des applications critiques sur Mainframe

À noter que le marché Mainframe évolue finalement assez peu : 1,531 milliard en 2012, 1,579 milliard estimé en 2017. Les services et les logiciels représentent 89 % des dépenses (2013), à peine 11 % pour le matériel. Ce sont des plateformes qui ont un cycle de vie très long par rapport aux serveurs et PC classiques (et les tarifs ne sont pas tout à fait les mêmes).

Des formations plus nombreuses ?

Une forte majorité d'écoles d'informatique et d'universités n'ont pas de cours Cobol. Une étude datant de 2013, menée par Micro Focus, évoque même que 73 % des universités (améri-

caines) ne font rien sur notre vénérable Cobol. Toutefois depuis quelques années, les éditeurs du monde Cobol/Mainframe, et certaines entreprises, cherchent à sensibiliser entreprises, écoles, étudiants et développeurs. L'un des objectifs est d'ouvrir des formations dédiées et de former des développeurs à la compétence Cobol. Attention, nous ne parlons pas de cobolistes, mais de compétences Cobol. Que ce soient les éditeurs, les SSII, les entreprises, des initiatives existent et se mettent en place pour proposer des formations et des cursus, notamment dans les écoles. Ainsi Compuware travaille avec des universités pour former au Cobol en fournissant ces outils. Ce programme arrivera en Europe dans les prochains mois. IBM a lancé un programme mondial « System Z Academic Initiative » qui doit faciliter l'accès aux ressources pour les enseignants et les étudiants. Ce programme a été déployé dès 2004 en France. En France, les écoles référencées sur le site du programme sont : Telecom Nancy, Epita, Epsi (Montpellier). Atos a lancé une académie Cobol.

Des outils plus modernes

Les éditeurs Cobol/Mainframe ont énormément fait évoluer leurs outils. Il n'est pas question de travailler directement sur le mainframe, ni en écran vert. Les technologies les plus récentes sont intégrées. Par exemple, Micro Focus fournit un outil graphique Visual Cobol qui s'utilise dans Visual Studio ou Eclipse, Compuware s'appuie sur la plateforme Eclipse pour ces environnements, Cobol-IT fait de l'open source, Metrixware utilise lui aussi Eclipse pour son outil Cobos, Raincode utilise la plateforme .Net. Ces outils récents facilitent la modernisation, la rénovation, voire, la migration des applications Cobol, et, surtout, ils offrent un environnement de développement très moderne et bien plus agréable. La génération des API permet aussi de proposer des développements plus rapides, même le cœur Cobol changera finalement peu.

Observatoire Cobol 2015 : quelques chiffres

Micro Focus a publié son observatoire Cobol 2015, pour la première fois, l'étude est internationale. Les données clés :

- 24,4 % des applications Cobol tournent sur un système zOS mais Windows et Linux s'imposent très largement (44,9 % et 25,9 %), les systèmes Unix pèsent 27 %.
- Les codes Cobol pèsent parfois très lourd dans l'informatique des entreprises. 30 % des entreprises disent avoir au moins 5 millions de lignes de codes Cobol.
- Le poids de l'existant, et souvent l'importance du Cobol, se perçoivent dans l'évolution ou non du parc applicatif : 46 % des sondés disent prévoir le même périmètre Cobol, 35 % pensent l'augmenter, et (seulement) 18 % parlent de le baisser... Les entreprises ayant le moins de ligne de codes sont celles qui pensent à décroître la part du Cobol.
- Sans surprise, pour la majorité des entreprises, ce sont des codes stratégiques et très stratégiques.
- Ce poids se retrouve bien évidemment dans le budget alloué au Cobol. Pour une majorité, c'est minimum 15 % du budget IT (finance/assurance, distribution, ISV).
- 61 % des applications sont maintenues en l'état, mais 35 % sont en phase de modernisation. Seulement 15 % sont migrées vers des systèmes plus ouverts et 21 % sont ou seront réécrites.

Conclusion : Cobol est là pour durer...

C'est dans l'R

Si vous feuilletez ce magazine à la recherche d'une courte introduction à R, essayez ceci : R est un langage de programmation et un environnement pour compiler analyses statistiques et représentations graphiques associées. R est donc logiquement privilégié par les statisticiens et data scientists, mais il a vu sa popularité exploser ces dernières années. Si vous n'aviez jamais entendu parler de R, cela devrait suffire à vous en donner une première idée.

Dr. Christoph P. Freier
Pharmacien et chercheur de formation, d'une nature curieuse pour l'outil informatique et l'open source, aimant à programmer l'analyse des ses données de laboratoire grâce à R et Python.

Vous pourriez néanmoins vous poser un certain nombre d'autres questions. Par exemple, pourquoi R ? R a été initialement développé par Ross Ihaka et Robert Gentleman. Le langage nommé R reprend ainsi les initiales de leurs deux prénoms respectifs, mais présente aussi un clin d'œil évident à S. Ou encore R, pourquoi ? R et ses librairies codent pour un très grand nombre de modèles statistiques. Il présente l'énorme avantage d'être facilement modulable via un certain nombre de fonctions implémentées et extensions mises à disposition par une communauté de professionnels et de passionnés active et réactive. Cela sonne certainement prometteur si vous avez besoin de tester la signifiance de résultats expérimentaux ou de représenter un set de données.

Si R présente déjà de nombreuses possibilités de calcul et une variété de solutions élégantes pour visualiser des données sous formes d'histogrammes, boîtes à moustaches, diagrammes circulaires, sa grande force réside dans les nombreux modules, ou *packages*, disponibles. Les plus simplistes et malgré tout déjà très performants sont inclus dans l'installation de base, de nombreux autres étant disponibles sur le Web sous forme de modules à installer dans votre librairie locale avant utilisation. Le tout peut se faire aisément via le CRAN, qui rassemble presque tous les packages disponibles. Même si tout ou à peu près est déjà possible sans avoir recours à ces packages supplémentaires, pour certaines solutions spécifiques, comme avoir accès à un certain nombre d'options graphiques ou bien de calcul supplémentaires, ceux-ci s'avèreront utiles. Typiquement, toutes les fonctions offertes par un package sont liées entres elles. Par exemple, le package `stats` contient les fonctions requises pour faire des analyses statistiques. Pour utiliser un package, il vous faut au préalable le charger dans R via la fonction `library(…)`. Dans cet article, nous nous contenterons des packages `stats` et `graphics`, tous deux présents et chargés lors du démarrage de R, si bien que vous n'avez à vous soucier de rien dans un premier temps.

Si vous êtes arrivé jusqu'ici, c'est probablement que vous avez saisi l'intérêt de R et pensez qu'il serait possiblement un outil intéressant pour vous. Pour le vérifier, rien de mieux que d'en faire l'essai. Selon votre système d'exploitation, après installation, l'interface graphique de R sera composée d'une fenêtre de commandes et éventuellement de quelques barres de menu. À l'heure actuelle, l'environnement le plus intéressant pour développer en R est RStudio. Celui-ci est multiplateforme, gratuit, open source et présente de plus l'avantage d'être directement accessible via un navigateur Web. Une fois RStudio lancé, l'interface graphique utilisateur présente quatre fenêtres incluant une console au sein de laquelle une invite de commande « > » fait suite à un court message de bienvenue. Cette console représente l'outil le plus important pour utiliser R. C'est ici que seront tapées les commandes - aussi nommées *expressions* - et qu'apparaîtront les résultats (ou messages d'erreur en cas de malheur) après compilation. Le langage R, en plus d'être très fortement orienté objet, est interactif. Toute expression sera traitée et le résultat vous en sera immédiatement retourné. Si vous n'avez jamais utilisé d'environnement interactif, pas de panique, cet article a vocation de vous donner les bases nécessaires pour travailler avec R.

Par défaut, l'invite de commande est un signe plus grand que « > » et paraîtra en début de ligne à chaque fois que R attend de votre part une expression. Cet article inclut des exemples d'expressions à entrer dans R (et vous invite à faire l'essai chez vous). Par la suite, l'invite « > » permet de différencier commandes et réponses, mais ne doit pas être entrée dans votre console (éventuellement juste une fois, pour expérimenter et vous familiariser avec un message d'erreur). Ainsi :

```
> 19 + 23
[1] 42
```

Signifie qu'en entrant « 19 + 23 » après l'invite de commande dans votre console, R devrait vous répondre « [1] 42 ». De même :

```
> 20 + 20 + 2
[1] 42
> 14 * 3
[1] 42
> (10 + 4) * 3
[1] 42
```

Notez le récurent « [1] » qui accompagne chaque réponse. R interprète chaque nombre en tant que vecteur, correspondant à une collection ordonnée de nombres. Ici « [1] » nous indique que l'index de nos réponses est toujours de 1 (au sein d'un vecteur, lui même de longueur 1). Pour créer un vecteur de longueur différente, le plus aisé est d'utiliser la fonction combiner c(...) :

```
> c(1, 2, 3, 4, 5)
[1] 1, 2, 3, 4, 5
```

Ou encore, plus simple dans le cas présent, l'opérateur « : » qui produit toutes les valeurs entre 1 et 5 (les utilisateurs de Python noteront ici une différence de taille) :

```
> c(1:5)
[1] 1, 2, 3, 4, 5
```

Évidemment, R devient vraiment intéressant si l'on commence à combiner les opérations :

```
> c(1:5) * 5
[1] 5, 10, 15, 20, 25
```

Enfin, il est important de noter qu'aucun de nos résultats n'a été pour le moment sauvegardé. Ils ont simplement été affichés dans la console. Recompiler le code en naviguant via les flèches haut et bas risque de rapidement se révéler insuffisant. Mieux vaut assigner et sauvegarder sous forme de variables via l'opérateur « <- » :

```
> x <- c(1:5)
>
```

Qui ne génère pas de réaction de la part de R. Par contre, la variable x a bien été créée et peut être rappelée ou bien servir dans de nouvelles opérations (sans être elle même affectée !) :

```
> x
[1] 1, 2, 3, 4, 5
> x * 5
[1] 5, 10, 15, 20, 25
> x
[1] 1, 2, 3, 4, 5
```

La puissance de R, c'est de faire appel à de nombreuses fonctions qui vont pouvoir travailler avec ces variables. À travers les exemples précédents, nous avons déjà utilisé une fonction implémentée dans R : c(…), qui permet de combiner des valeurs en un vecteur ou une liste. De manière générale, les fonction dans R prennent la forme f(argument 1, argument 2, …) à l'exception des opérateurs (« + », « - », « * », « / », ...) qui sont néanmoins traduits par R en fonctions. Les fonctions permettent de réaliser un grand nombre d'opérations sur nos variables. Admettons par exemple que nous cherchions à savoir si la valeur 15 se trouve indexée dans notre variable x :

```
> x == 15
[1] FALSE FALSE FALSE FALSE FALSE
```

Non, ce n'est pas le cas. Par contre 15 se trouve dans notre variable « x * 5 » :

```
> x * 5 == 15
[1] FALSE FALSE TRUE FALSE FALSE
```

La fonction which(…) peut même nous donner la position exacte de la valeur 15 dans « x * 5 » :

```
> which(x * 5 == 15)
[1] 3
```

Dans le cas présent, la réponse est évidente. La fonction which(…) se révèle par contre très utile si votre vecteur contient plusieurs dizaines, voir centaines de valeurs et potentiellement plusieurs fois la valeur recherchée. Pour bien débuter dans R, la plus importante de toutes les fonctions lors de l'apprentissage est la fonction help(…) . Celle-ci vous renvoie directement à la documentation de chaque fonction. Essayez help(c) ou help(which) pour voir par vous même. La documentation contient systématiquement une courte description de la fonction, de son usage, de ses arguments et même un ou plusieurs exemples.

Dans R, vous pouvez organiser vos données dans des structures plus complexes que les vecteurs seuls : un *tableau de données* ou *data frame* en anglais. Un tableau de données rassemble plusieurs vecteurs de même longueur. Pour rendre le tout un peu plus parlant, construisons un exemple à partir de données fictives sur des souris de laboratoire : 20 souris témoins nourries avec une alimentation standard et 20 souris nourries avec une alimentation riche. De plus, dans chaque cohorte, nous avons inclus dix souris dont le pelage est blanc, dix dont le pelage est noir. Le poids moyen et la couleur des souris peuvent être simulé de la manière suivante :

```
> souris_norm <- rnorm(20, 19)                    # Alimentation standard
> souris_rich <- rnorm(20, 21)                    # Alimentation riche
> souris_coul <- c(rep(c("blanc", "noir"), 10))   # Couleur des souris
```

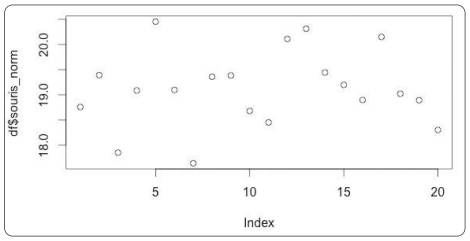
La fonction rnorm(…) permet de simuler des données de distribution normales. Ici, de manière relativement arbitraire, les 20 souris témoins pèsent en moyenne 19 grammes, les souris richement nourries en moyenne 21 grammes. La fonction rep(…) permet de répéter une variable, ici la couleur blanche ou noire de la souris, 10 fois dans chaque cas. De manière occasionnelle, il est utile d'écrire du texte à l'attention des humains qui vont lire le code, mais pas des ordinateurs qui vont l'exécuter. Pour inclure des *commentaires* dans votre code, utilisez le symbole #. Tout ce qui vient par la suite sur la ligne sera ignoré par R et ne servira qu'à clarifier un point important pour vous ou pour une tierce personne qui, jetant un œil sur votre code, pourrait avoir besoin d'une information importante.

Ces informations n'étant utiles que pour les humains, pas pour R, vous pouvez sereinement les oublier lors de vos premiers essais de code. Par contre, elles se révèleront bien vite indispensables, même si vous ne montrez votre code à personne, simplement pour vous souvenir de ce que vous avez codé le vendredi soir, si vous y revenez le lundi matin. Enfin, pour construire notre tableau de données, la fonction data.frame(…) se révèle utile :

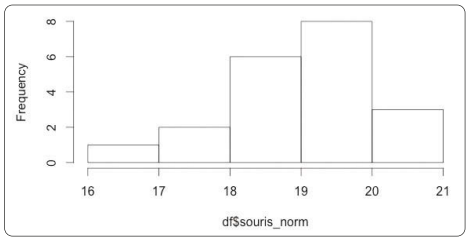
```
> df <- data.frame(souris_norm, souris_rich, souris_coul)  # Construire le tableau
> df                                                       # Imprime le tableau
   souris_norm souris_rich souris_coul                     ###
1    18.75768   20.87528      blanc                       # Attention ! Ne reste
2    19.39169   20.00118       noir                       # lisible que dans le cas
3    17.85296   20.71500      blanc                       # d'un petit tableau. Sinon
4    19.08678   21.31159       noir                       # préférez la fonction
5    20.45429   18.05614      blanc                       # View(…)
6    19.09789   22.12341       noir                       ###
7    17.64126   19.86626      blanc                       # Notez également que vos
8    19.36024   21.87872       noir                       # données de poids générées
9    19.38423   21.09137      blanc                       # sont différentes car
10   18.68007   19.62770       noir                       # générées par vous avec la
11   18.45230   21.20415      blanc                       # fonction rnorm(…)
12   20.11152   20.07638       noir
13   20.31215   19.31553      blanc
14   19.44303   20.02577       noir
15   19.19723   21.23953      blanc
16   18.89781   21.15122       noir
17   20.14992   21.21244      blanc
18   19.02156   19.87921       noir
19   18.89328   22.50774      blanc
20   18.30369   22.08841       noir
> View(df)                                                # Visualiser le tableau
```

Pour vous referer à certains composants du tableau seulement, il faut spécifier le nom du tableau et le nom du vecteur séparés par l'opérateur « $ » :

```
> df$souris_norm            # Valeurs pour les souris témoins seulement
> summary(df$souris_norm)   # Résumé (min, max, médiane, etc.)
> plot(df$souris_norm)      # Nuage de point pour les souris témoins
```

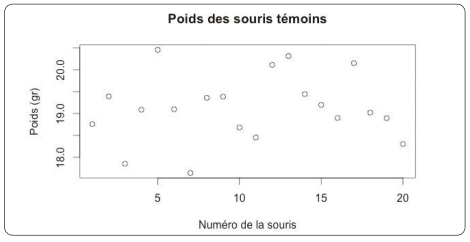



```
> hist(df$souris_norm)    # Distribution sous forme d'histogramme
```



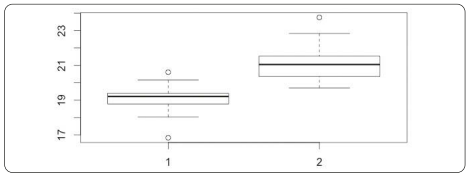
Le nuage de points et l'histogramme sont générés, sans informations de votre part, de manière relativement brute : sans légendes. Ceci ne facilite pas la lecture. Heureusement, help(boxplot) permet d'obtenir la liste des arguments que peut prendre la fonction boxplot(…). Par exemple, l'argument xlab permet de rajouter une légende sur l'axe des abscisses (de même pour ylab sur l'axe des ordonnées). Le plot(df$souris_norm) gagne énormément en lisibilité après l'ajout de la légende "numéro de la souris" en abscisse et "poids (gr)" en ordonnée. Pour ajouter un titre à la figure, c'est l'argument main qu'il faut remplir. Ainsi :

```
> plot(df$souris_norm,                  # Le saut de ligne pour la lisibilité.
+xlab= c("Numéro de la souris"),        # Le + signifie que R attend le reste
+ylab= c("Poids (gr)"),                 # des arguments avant de compiler mais
+main= c("Poids des souris témoins"))   # ne l'entrez pas dans votre console !
```



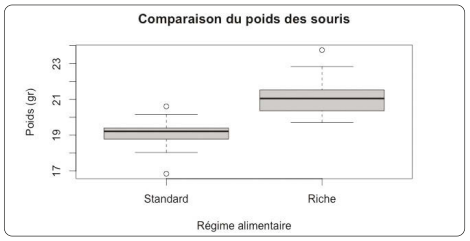
La fonction sort(…) permet de classer les souris par ordre croissant de poids. Essayez ! Evidemment, nous pourrions procéder de même pour les souris nourries avec une alimentation riche. Cependant le plus adéquat pour comparer souris témoins et alimentées richement est la représentation sous forme de boîte à moustaches :

```
> boxplot(df$souris_norm, df$souris_rich)
```



Là encore, la liste des arguments accessibles via la fonction help(boxplot) peut nous aider à améliorer l'esthétique et à gagner en lisibilité :

```
> boxplot(df$souris_norm, df$souris_rich,
+xlab= c("Régime alimentaire"),
+names= c("Standard", "Riche"),                  # Donne un nom aux deux groupes
+ylab= c("Poids (gr)"),
+main= c("Comparaison du poids des souris"),
+col= c("grey"))                                 # Colore les boîtes à moustaches
```

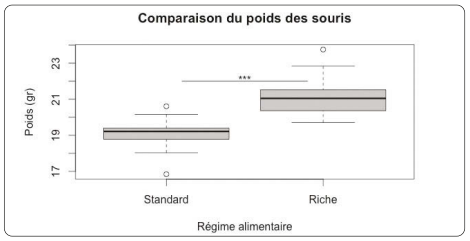


Mais R, c'est aussi un formidable outil statistique ! La fonction t.test(…) teste pour nous la validité de l'hypothèse d'égalité entre les deux populations de souris :

```
> t.test(df$souris_norm, df$souris_rich)
```

La valeur retournée dépendra des données de poids des souris générées. Mais elle sera significative. Nous pourrions donc rajouter à la figure précédente une légende du type :

```
> lines(x=c(1.1,1.9), y=c(22, 22))
> text(1.5, 22.2, "***")
```

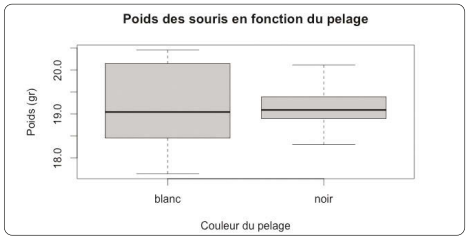


Pour comparer, par exemple le poids des souris blanches et des souris noires, il suffit de modifier légèrement les arguments de la fonction boxplot(…) :

```
> boxplot(df$souris_norm~df$souris_coul,    # Poids en fonction de la couleur
+xlab= c("Couleur du pelage"),              # via l'opérateur « ~ ».
+ylab= c("Poids (gr)"),
+main= c("Poids des souris en fonction du pelage"),
+col= c("grey"))
```

Là encore, la fonction t.test(…) teste l'hypothèse d'égalité entre les populations :

```
> t.test(df$souris_norm~df$souris_coul)
```



Mais dans ce cas, la couleur du pelage des souris ne semble pas avoir d'influence sur le poids de nos souris témoins.

Tout ceci ne représente qu'un aperçu des fonctions de R. Il ne faut pas oublier que de nombreux packages disponibles viennent démultiplier les possibilités. Néanmoins, les bases sont toujours les mêmes, et ceci, que les données concernent des souris de laboratoire, la pureté et le prix de diamants, ou bien encore les passagers du Titanic. Pareillement, R compilera figures et analyses pour 20 souris tout comme pour 200.000 avions au départ de l'aéroport de Houston. Enfin, votre *script* R tout comme votre environnement peuvent être sauvegardés via RStudio et réouverts ultérieurement pour continuer le travail sur vos données.

Bienvenue dans l'R ! ⊞

Tous les mois, faites le plein de technologies

Abonnez-vous à PROGRAMMEZ! le magazine du développeur

Nos classiques

Tarifs France métropolitaine

1 an ..... 49 €
11 numéros

2 ans ..... 79 €
22 numéros

PDF ..... 30 €(*)
1 an - 11 numéros

(*) Souscription sur le site internet

Etudiant ..... 39 €
1 an - 11 numéros

Nos goodies

Clé USB ..... 29,90 €

Commandez-la sur www.programmez.com

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

Nos offres spéciales

Abonnement +  + 

1 an → 64,90 € *
2 ans → 94,90 € *

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

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

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

Toutes nos offres sur www.programmez.com

Oui, je m'abonne

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

☐ ***Abonnement 1 an au magazine :*** 49 €

☐ ***Abonnement 2 ans au magazine :*** 79 €

☐ ***Abonnement étudiant 1 an au magazine :*** 39 €
Photocopie de la carte d'étudiant à joindre

Offre spéciale :
Programmez + kit CPL dLAN 550 + Clé USB

☐ ***Abonnement 1 an :*** 64,90 € (au lieu de 158,80 €) *

☐ ***Abonnement 2 ans :*** 94,90 € (au lieu de 188,80 €) *

☐ M. ☐ Mme ☐ Mlle Entreprise : Fonction :

Prénom : Nom :

Adresse :

Code postal : Ville :

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

E-mail : @

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

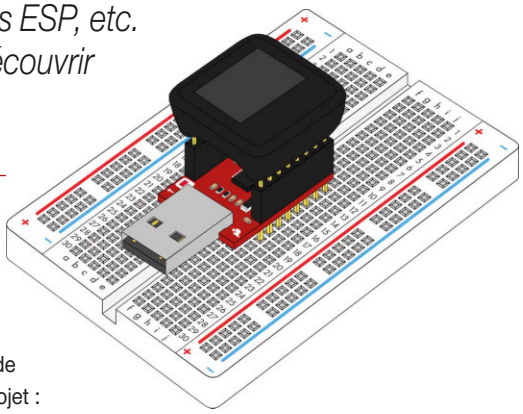
☐ Je souhaite régler à réception de facture

** Tarifs France métropolitaine*

PROG 190
***Offre limitée**, valable jusqu'au 30 novembre 2015*

Sparkfun MicroView : microcontrôleur + OLED = super fun !

Les cartes Arduino Uno ou Raspberry Pi, voire Gadgeteer, sont trop grandes pour vos projets et objets connectés ? Les mini-cartes se multiplient : Arduino Nano/Mini, AirBoard, LimiFrog (ex-BlueFrog), MicroView, Photon, les cartes ESP, etc. Nous vous proposons de découvrir la MicroView de Sparkfun.



François Tonic
Programmez!

Nous avions besoin d'une nouvelle carte de petite taille pour concevoir un nouveau projet : une montre. Les dimensions des composants étaient donc primordiales, surtout pour la carte et l'intégration des différents composants et de l'écran OLED. Nous avions fait quelques essais avec une Arduino Nano puis une Mini mais le résultat n'était pas toujours très satisfaisant. C'est alors que nous avons regardé les modèles plus compacts et les plateformes intégrant plusieurs composants tels qu'un écran OLED. Après plusieurs heures de comparaisons, notre choix s'arrêta sur la MicroView de Sparkfun et son starterkit.

Aperçu de la MicroView

Basiquement, la MicroView est plus qu'une carte compatible Arduino. Elle est à la fois un microcontrôleur ATmega328P et un écran OLED 68x68, supportant 3,3 V et 5V. Elle fonctionne de manière autonome. Elle exige uniquement une alimentation. Elle possède 18 broches de connexions (12 pour les I/O numériques et 8 pour l'analogique). Les broches sont compatibles Arduino même si la configuration déconcerte un peu. D'autre part, elle nécessite un programmeur externe pour être reliée au poste de développement. Sparkfun propose un module USB pour assurer la liaison. Ce qui étonne quand on le voit pour la première fois, c' est l'aspect très compact : à peine 2,5 cm de côté et autant en hauteur. C'est une des qualités de la MicroView. On peut facilement l'embarquer dans son projet ; un objet.

Première prise en main

La taille étonne beaucoup ainsi que la légèreté du MicroView. La fabrication est de qualité avec une bonne finition. Petit détail : il n'y a aucune indication sur les broches, juste un point (pas toujours facile à voir) indique le sens de connexion sur le module USB et il permet ainsi de déterminer l'ordre des broches.

Sur les broches, un conseil : imprimer le schéma de connexion et les équivalents Arduino – MicroView, sinon vous allez perdre beaucoup de temps. Pour le développement, nous utilisons très simplement Arduino IDE. Nous n'avons eu aucun problème de pilotes ou de reconnaissance sous OS X 10.11. Dommage que le module USB ne comporte pas un connecteur USB plus standard (le câble est fourni). La board est alimentée par l'USB, bien entendu, vous pouvez, une fois le montage terminé, utiliser une batterie de type pile ou LiPo, via la broche VIN (16 V maximum en entrée). Pour les montages, malgré la petitesse du matériel, on dispose des broches nécessaires pour monter des objets et projets avec les Entrées/Sorties analogiques et numériques. Vous pouvez utiliser des composants 3,3 V et 5 V, mais attention à ne pas trop charger la board.

Pour faciliter nos montages, nous utilisons le module USB et la planche à pain du kit. Vous ne serez pas déconcerté : il s'agit d'une Arduino classique, même si on dispose de moins d'extensions.

Côté programmation, là encore, aucune surprise, à un (gros) détail près : microview.h. En effet, pour contrôler la MicroView, Sparkfun a conçu une librairie dédiée. Ainsi, on manipule facilement les fonctions de l'écran, notamment la partie graphique. Il est par exemple capable d'afficher des graphiques, des animations 3D... Pour ce faire, il suffit d'inclure MicroView.h dans votre code. Dans le code, on utilisera uView, suivi de la fonction. L'écran fonctionne avec des notions de page et de widgets (graphiques) que l'on peut facilement utiliser (widget = new MicroViewGauge) et comme toujours, on positionne les différents objets sur l'écran. La librairie propose de nombreuses fonctions : affichage, contraste, clear, circle, etc. La documentation en ligne permettra d'en savoir plus : http://microview.io/doco/html/index.html

Il ne faut pas hésiter à tester les possibilités graphiques et les différentes fonctions de l'écran qui est de bonne qualité et très lisible,

MONTREGEEK

Depuis le printemps 2015, nous souhaitons réaliser une montre. Nos premiers prototypes furent réalisés avec des Arduino Nano, mais la partie graphique posait problème : instabilité de l'affichage sur un écran OLED générique avec la librairie graphique U8glib. L'usage d'un OLED de type Grove OLED a été envisagé, mais, un peu trop volumineux. Après quelques semaines de recherches, nous décidons de choisir la MicroView pour ses dimensions réduites et l'intégration board et OLED. Dans le prototype actuel, nous avons intégré les capteurs suivants :

- Horloge temps réel DS1302
- Température TMP36
- Altimètre, avec pression
- 2 boutons poussoirs
- Bonus : capteur de flamme (utilisation réelle à définir j'avoue)

Le composant altimètre remplit une triple fonction : altimètre, baromètre et thermomètre. Dans une future itération, nous supprimerons le capteur TMP, qui fait désormais doublon. Les boutons poussoirs permettent de changer de pages d'affichage selon les données à afficher. Une batterie externe sera intégrée, durant la phase de finalisation.

Pour le moment, nous n'avons pas choisi le capteur pour détecter le mouvement du poignet pour activer l'affichage de l'OLED et économiser l'énergie.

Une fois tous les composants intégrés (et retrait de la planche de prototypage), le code finalisé et fonctionnel, nous nous occuperons de la partie design du boîtier et du bracelet.

malgré sa taille. Cependant, on aura moins de liberté qu'avec un OLED générique et la librairie U8glib. Le site officiel propose quelques docs, une prise en main rapide. Mais cette documentation reste relativement pauvre et très peu de docs sont disponibles en Français. Et la MicroView étant encore peu utilisée, les projets et tutoriels sont peu fréquents.

Conclusion

La MicroView est une board sympathique et idéale pour les objets connectés et les projets makers. Si on est parfois un peu dérouté au début, la board se révèle efficace et stable, avec de bonnes performances graphiques. Ce n'est pas l'unique carte intégrée du marché et elle est relativement chère et pas toujours facilement trouvable en France. Les shields adaptés sont peu nombreux et chers, par rapport aux shields Arduino classiques. Et le kit complet « inventeur », bien que de qualité, est trop cher. Dommage. Espérons aussi que la communauté se développera. Malgré tout, MicroView reste un excellent choix.

LES TARIFS

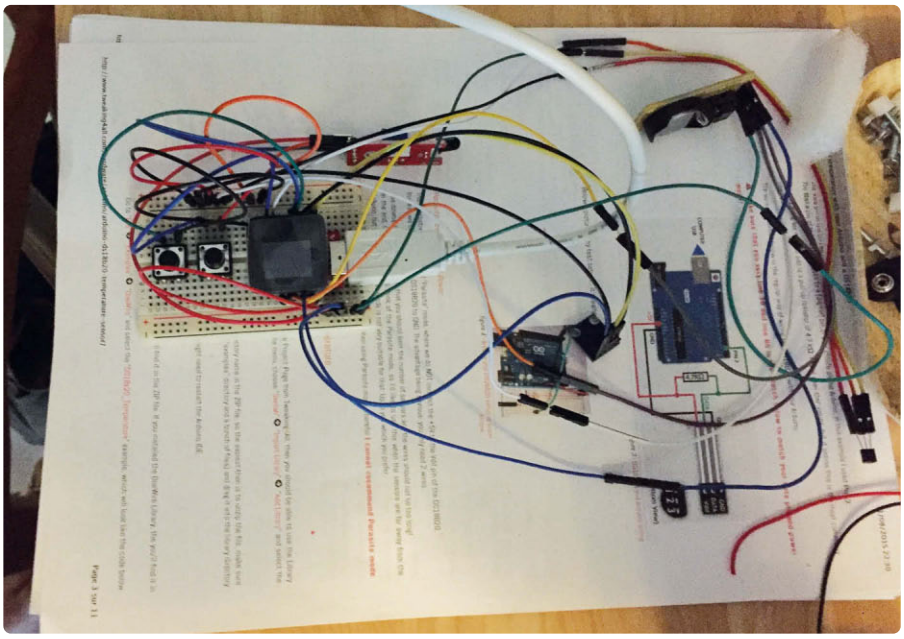
La MicroView seule est vendue en France dans certaines boutiques en ligne telles que Lextronix. Il est possible de la trouver à – 40 €. Le module USB coûte – 15 €. Le kit complet Inventor est vendu généralement – 85 €.
Comparé à des cartes Nano ou Mini, la MicroView est plus chère même en y rajoutant l'ensemble des composants. L'avantage de la plateforme Sparkfun est la qualité de fabrication, la compacité de l'ensemble et l'aspect intégré. Mais elle est aussi moins facile à trouver que les autres cartes, Arduino ou équivalente.

LES +

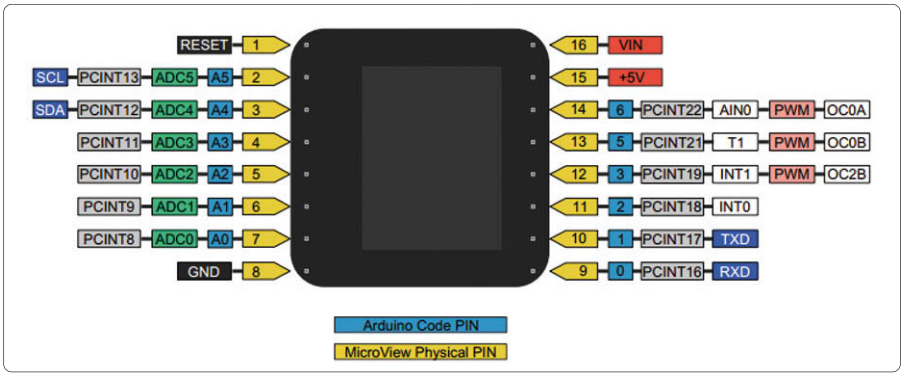
- Dimension réduite
- Intégration OLED
- Facilité d'intégration
- Librairie dédiée
- Compatibilité Arduino
- Qualité du boîtier
- Stabilité

LES –

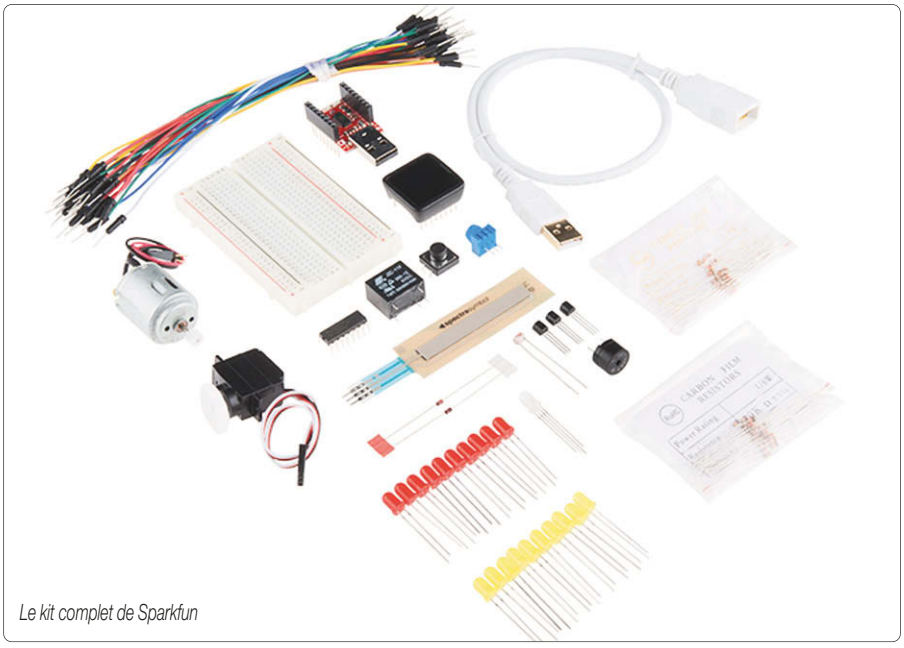
- Brochage déroutant
- Librairie parfois peu souple
- Taille de l'écran
- Pas toujours facile à trouver en France
- Documentation trop faible
- Communauté
- Prix



Prototype montregeek



Correspondance des broches



Le kit complet de Sparkfun

Une partie des classes de la librairie dédiée

Je pratique le C++

Partie 4/5

Level 300

Nous vous proposons une série d'articles sur la pratique de C++ pour que vous puissiez tous vous y mettre. Ce mois-ci on aborde la librairie standard du C++ nommé STL : Standard Template Library.

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

Le langage C possède son runtime et le C++ possède aussi son runtime, appelé STL. Dans le monde Microsoft, le runtime du C se nomme MSVCRTxxx.dll et celui du C++ se nomme MSVCPxxx.dll. Cependant, la STL ne se comporte pas comme une liste de fonctions que l'on peut appeler – exemple fopen en C.

La STL est constituée de templates. SI vous avez lu l'article précédent, nous y avons étudié les templates en douceur. Cela veut dire plusieurs choses. La STL est organisée via des fichiers d'entêtes.

Les fonctionnalités offertes par la bibliothèque standard peuvent être classées comme suit :

- Support du runtime du langage (par exemple, pour l'allocation et les informations de type runtime) ;
- La bibliothèque C standard (avec des modifications vraiment mineures pour minimiser la violation du système de type) ;
- Les chaînes (avec le support des jeux de caractères internationaux et localisation) ;
- Support pour la correspondance des expressions régulières ;
- Les flux d'entrée/sortie sont un framework extensible pour l'entrée et la sortie vers lequel les utilisateurs peuvent ajouter leurs propres types, flux, stratégies de buffering, locales et jeux de caractères ;
- Un framework de conteneurs (comme vector et map) et d'algorithmes (comme find(), sort(), et merge()). Ce framework, appelé conventionnellement la STL, est extensible pour que les utilisateurs puissent y ajouter leurs propres conteneurs et algorithmes ;
- Support pour le calcul numérique (comme les fonctions mathématique standard, les nombres complexes, les vecteurs avec des opérations arithmétiques, et les générateurs de nombres aléatoires) ;
- Support pour la programmation parallèle, incluant les threads et les verrous. Le support parallèle est fondamental pour que les utilisateurs puissent ajouter le support aux nouveaux modèles parallèles dans des bibliothèques ;
- Des classes utilitaires pour le support de la méta programmation à base de templates (par exemple, type traits, la programmation générique avec le style STL (par exemple, pair), et la programmation générale (par exemple, clock) ;
- Les ''Pointeurs Intelligents'' pour la gestion des ressources (par exemple, unique_ptr et shared_ptr) et une interface pour la libération des ressources ;
- Des conteneurs spéciaux, comme array, bitset, et tuple.

Les critères principaaux pour inclure une classe dans la bibliothèque étaient que :

- Cela pourrait être utile à presque tous les programmeurs C++ (les novices ou les experts) ;
- Cela pourrait être fournie sous une forme générale qui n'ajoute pas de surcharge importante par rapport à une version simplifiée de la même fonctionnalité ;
- Que l'utilisation doit être simple à apprendre (par rapport à la complexité de la tâche).

La bibliothèque standard de C++ fournit essentiellement les structures de données fondamentales les plus courantes ainsi que les algorithmes fondamentaux à utiliser avec. Nous allons balayer toute ou partie de cette liste exhaustive.

Sélection des fichiers d'entêtes de la bibliothèque standard:	
<algorithm>	copy(), find(), sort()
<array>	array
<chrono>	duration, time_point
<cmath>	sqrt(), pow()
<complex>	complex, sqrt(), pow()
<forward_list>	forward_list
<fstream>	fstream, ifstream, ofstream
<future>	future, promise
<ios>	hex,dec,scientific,fixed,defaultfloat
<iostream>	istream, ostream, cin, cout
<map>	map, multimap
<memory>	unique_ptr, shared_ptr, allocator
<random>	default_random_engine, normal_distribution
<regex>	regex, smatch
<string>	string, basic_string
<set>	set, multiset
<sstream>	istrstream, ostrstream
<stdexcept>	length_error, out_of_rang e, runtime_error
<thread>	thread
<unordered_map>	unordered_map, unordered_multimap
<utility>	move(), swap(), pair
<vector>	vector

Le type string

Ce type est défini dans le fichier d'entête <string> et c'est le type qui permet de faire la liaison entre les type char du C et le type string d'un niveau plus haut qu'est le type string.

Le type string permet de concaténer des strings entre elles. Ce n'est pas un pointeur de char. C'est un template.

```
typedef basic_string<char, char_traits<char>, allocator<char> >
  string;
```

Il est possible de fournir son propre allocateur mémoire. Mais qui va utiliser cela ? Et pourtant c'est prévu... Rappelez-vous dans l'article sur les classes lorsque je parlais de la casquette auteur de classe et de la casquette utilisateur de classe. On y est exactement sauf que c'est une classe template.

```
string msg = "Le nutty est coquine !";
string msg2 = "Maggie est la reine des coquines !";
string s = msg + " " + msg2;
cout << s << endl;
string nutty = s.substr(0, 9);
cout << nutty << endl;
msg[3] = toupper(msg[3]);
cout << msg << endl;
string temp("VS 2015 is here !");
```

```
size_t t = temp.find_first_of("here");
cout << temp << " " << t << endl;
```

La classe string contient de nombreuses opérations qui évitent le parcours des chaînes C à l'ancienne. Pour passer d'une string à un type C char *, il faut utiliser la fonction c_str() :

```
const char * psz = msg.c_str();
printf("String const char * = %s\n", psz);
```

Les expressions régulières (regex)

La fonctionnalité des regex est disponible dans le fichier d'entête <regex>. Le support est le même que dans d'autres langages ; on y trouve les fonctions regex_match, regex_search, regex_replace, regex_iterator, regex_token_ietrator.

Les IO/Streams

La STL utilise les flux pour gérer des I/O avec des buffers. La fonction la plus connue est cout et son opérateur '<<'.

```
cout << "On n'est pas que des Mickey !" << endl;
int i = 10;
float f = 2.5;
cout << i << "," << f << endl;
```

Le meilleur ami de cout est cin. Cela permet de capter les entrées du clavier :

```
int j = 0;
cout << "Give ma a Int !" << endl;
cin >> j;
cout << "Merci pour " << j << endl;
```

En plus de gérer les types standards et les strings, il est possible de tirer parti de la librairie pour afficher d'autres types. Exemple, considérons le type suivant :

```
struct CPersonne
{
  int age;
  string name;
};

ostream& operator<<(ostream& os, const CPersonne& p)
{
  return os << p.name << " " << p.age << " ans";
}

void TestIO()
{
  CPersonne p;0
  p.age = 5;
  p.name = "Audrey Maggie";

  cout << "La soeur de Lisa c'est " << p << endl;
}
```

La définition de l'opérateur << avec le type ostream permet de passer un type CPersonne à cout et cela sans forcer ! Autres aspects de la librairie I/O, c'est le formatage.
Pour afficher des types comme le fait une fonction printf en C avec les %d, %f, %x et %s ont leur équivalent dans la librairie.

```
void TestFormating()
{
  float f = 2.50;
  cout << f << ";"
     << scientific << f << ";"
     << hexfloat << f << ";"
     << fixed << f << ";"
     << defaultfloat << f << endl;
}
```

La gestion des fichiers est assurée au travers du fichier d'entêtes <fstream> :

- ifstream permet de lire un fichier,
- ofstream permet d'écrire un fichier,
- fstream permet de lire et d'écrire dans un fichier.

Exemple d'écriture de fichier :

```
ofstream ofs("c:\\temp\\MyGirls.txt");
ofs << "Edith" << endl;
ofs << "Lisa" << endl;
ofs << "Maggie" << endl;
ofs.close();
```

C'est vraiment très simple. La librairie I/O permet de gérer les buffers. Le fichier d'entête <sstream> permet de manipuler des chaînes :

- istringstream pour lire des chaînes,
- ostringstream pour écrire des chaînes,
- stringstream pour lire et écrire des chaînes.

```
int i = 20;
float f = 5.75;
string s = "Maggie est trop coquine !";
ostringstream oss;
oss << i << "," << f << "," << s;

string str = oss.str();
cout << str << endl;
```

Les containers

Le container le plus utilisé est vector<T>. Il est disponible dans le fichier d'entête <vector>. C'est une séquence d'éléments d'un type donné. Les éléments sont stockés de manière contiguë. Pour ajouter des éléments à un vector, il faut utiliser la méthode push_back().
Le parcours d'un vector peut se faire avec un range-for ou bien en utilisant un itérateur. Les containers de la STL sont tous accessibles au travers un itérateur. Les fonctions begin(), end(), operator++, operator–m operator* permettent de manipuler un itérateur. Exemple :

```
vector<CPersonne> v = { {12, "Edith"}, {9, "Lisa"}, {5, "Maggie"} };
CPersonne p;
p.age = 41;
p.name = "Papa";
v.push_back(p);
for (auto item : v)
{
   cout << item.age << " " << item.name << endl;
}
for (vector<CPersonne>::const_iterator it = begin(v); it != end(v); ++it)
{
   CPersonne p = *it;
   cout << p.age << " " << p.name << endl;
}
```

Voici la liste des opérations principales pour vector<T> :

Opération	Explication
v.empty()	Retourne true si v est vide. Sinon retourne false.
v.size()	Retourne le nombre d'élements dans v.
v.push_back(t)	Ajoute un élément de valeur t à la fin de v.
v[n]	Retourne une référence vers l'élément en position n dans v.
v1=v2	Remplace les éléments dans v1 avec une copie des éléments dans v2.
v1={a, b, c...}	Remplace les éléments dans v1 avec une copie des éléments de la liste.
v1==v2	v1 et v2 sont égaux s'il ya le même nombre d'él éments et de valeur.
v1!=v2	opposé de v1==v2
<, <=, >, >=	Suivant l'ordre des valeurs retourne un bool

Il existe un container qui représente une double liste chaînée au travers de list. Il est disponible au travers le fichier d'entête <list>.

```
list<CPersonne> l = { { 12, "Edith" },{ 9, "Lisa" },{ 5, "Maggie" } };
CPersonne p = { 41, "Papa" };
l.push_front(p);
CPersonne p2 = { 40, "Maman" };
l.push_back(p2);
for (auto item : l)
{
    cout << item.age << " " << item.name << endl;
}
```

Le container map<K,V> est très utile. Il est disponible dans le fichier d'entête <map>. C'est un container associatif.

```
map<string, int> family = { { "Edith", 12 },{ "Lisa", 9 },{ "Maggie", 5 } };
family["Papa"] = 41;
for (map<string, int>::const_iterator it = begin(family); it != end(family); ++it)
{
    string name = it->first;
    int age = it->second;
    cout << name << " " << age << endl;
}
```

Il existe d'autres containers dans la STL comme la hashtable nommée unordered_map. Le container hashtable et ses dérivées (voir tableau ci-dessous) ne contiennent pas le terme hashtable pour des soucis de nommage. Il y a surement du code existant qui a fait une classe ou un template hashtable, et c'est pour éviter la collision de nom.

Les containers disponibles dans la STL	
vector<T>	un vecteur de taille variable
list<T>	une liste doublement chaînée
forward_list<T>	une liste chaînée
deque<T>	une queue
set<T>	un set (une map avec une clé sans valeur)
multiset<T>	un set qui peut être en doublon
map<K,V>	un tableau associatif
multimap<K,V>	une map avec une clé qui peut être en doublon
unordered_map<K,V>	une map qui utilise un lookup hashtable
unordered_multimap<K,V>	une multimap qui utilise un lookup hashtable
unordered_set<K,V>	un set qui utilise un lookup hashtable
unordered_multiset<K,V>	un multiset qui utilise un lookup hashtable

Les algorithmes

La STL fournit des fonctions simples pour parcourir des ensembles, faire des copies, des insertions, des suppressions, des recherches simple ou complexes. Le fichier d'entête est <algorithm>. La force des algorithmes réside dans le fait qu'ils prennent pour la plupart un itérateur de début et un itérateur de fin afin de réaliser le parcours sur un ensemble fini. C'est un peu déroutant au début et puis finalement, on s'aperçoit que la plupart des parcours proposés dans ce fichier d'entête sont bien faits. Il faut maîtriser les itérateurs : c'est la seule contrainte. Dans le tableau ci-dessous, b vaut begin() et e vaut end().

Sélection d'algorithmes dans la STL	
p=find(b,e ,x)	p est le premier p dans [b:e) de telle manière que *p==x
p=find_if(b,e ,f)	p est le premier p dans [b:e) de telle manière que f(*p)==true
n=count(b,e ,x)	n est le nombre d'éléments *q dans [b:e) de telle manière que *q==x
n=count_if(b,e ,f)	n est le nombre d'éléments *q dans [b:e) de telle manière que f(*q,x)
replace(b,e ,v,v2)	Remplace les éléments *q dans [b:e) de telle manière que *q==v par v2
replace_if(b,e ,f,v2)	Remplace les éléments *q dans [b:e) de telle manière que f(*q) par v2
p=copy(b,e ,out)	Copie [b:e) dans [out:p)
p=copy_if(b,e ,out,f)	Copie les éléments *q de [b:e) de telle manière que f(*q) jusqu'a [out:p)
p=move(b,e ,out)	Déplace [b:e) vers [out:p)
p=unique_copy(b,e ,out)	Copie [b:e) vers [out:p); ne copie pas les duplicates adjacents
sort(b,e)	Trie les éléments de [b:e) en utilisant < comme critère de tri
sort(b,e,f)	Trie les éléments de [b:e) en utilisant la fonction de tri f
(p1,p2)=equal_range(b,e ,v)	[p1:p2) est la subséquence de tri [b:e) avec la valeur v; un binary search de v
p=merge(b,e ,b2,e2,out)	Merge deux séquences [b:e) et [b2:e2) dans [out:p)

Exemple avec find :

```
vector<string> v = { "Edith", "Lisa", "Maggie" };

//auto res = find(begin(v), end(v), "Maggie");
vector<string>::iterator res = find(begin(v), end(v), "Maggie");

if (res == end(v))
{
    cout << "Not found !" << endl;
}
else
{
    cout << "Found ! " << *res << endl;
}
```

Les templates utilities

Tout dans la STL n'est pas aussi simple que les containers ou la librairie I/O stream. Il existe des classes templates qui permettent de tirer parti de fonctionnalités avancées. Considérons la gestion des ressources, il existe deux templates qui sont unique_ptr<T> et share_ptr<T>. unique_ptr<T> représente la possession unique. shared_ptr<T> représente la possession partagée. Disponible dans le fichier d'entête <memory>, ces « smart pointers » ou pointeurs intelligents permettent de ne plus coder le delete, c'est le template qui s'en occupe. Le principal avantage d'utiliser les smart pointers est d'éviter les fuites mémoire.

```
unique_ptr<CPersonne> ptr(new CPersonne());
ptr->age = 41;
ptr->name = "Itchy";
```

```
// use ptr
// delete fait automatiquement
```

Voici la liste des opérations communes entre unique_ptr<T> et shared_ptr<T> :

Opération	Explication
shared_ptr<T> sp	Smart pointer null qui pointe sur un objet T
unique_ptr<T> up	Smart pointer null qui pointe sur un objet T
p	Utilise p comme une condition; true si p pointe sur un objet
*p	Déréférence p pour obtenir l'objet sur lequel p pointe
p->mem	Synonyme pour (*p).mem
p.get()	Retourne le pointeur dans p.
swap(p,q)	Swap les pointeurs dans p et q
p.swap(q)	Swap les pointeurs dans p et q

Le template share_ptr<T> possède quelques subtilités :

Opération	Explication
make_shared<T>(args)	Retourne un shared_ptr sur la mémoire allouée et initialise l'objet via args
shared_ptr<T> p(q)	p est une copie de q. Incrémente le compteur de référence interne
p=q	Incrémente le compteur de référence de q
p.use_count()	Retourne le nombre d'objets partagés avec p
p.unique()	Retourne true si p.use_count vaut 1 sinon false

Le template array<T, C> permet de gérer les tableaux aussi rapidement que les built-in arrays.

```
array<string, 3> ar;
ar[0] = "Edith";
ar[1] = "Lisa";
ar[2] = "Audrey";
for (auto element : ar)
{
    cout << element << endl;
}
```

Le template pair<T, U> représente deux éléments et est disponible dans le fichier d'entête <utility>. Utilisez make_pair pour remplir l'objet pair.

```
pair<string, float> p;
p.first = "The C++ Object Model";
p.second = 50.0;
cout << p.first << " " << p.second << endl;
pair<string, string> p2 = make_pair("Maggy", "t'es une coquine !");
cout << p2.first << " " << p2.second << endl;
```

Le template tuple<T...> représente une séquence de types variés et de types différents. Utilisez make_tuple pour remplir l'object tuple.

```
tuple<string, float, string> t;
t = make_tuple("C++ Primer", 50.0, "The best of all books !");
cout << get<0>(t) << ";" << get<1>(t) << ";" << get<2>(t) << endl;
```

Concurrency : le multithreading !

Il est possible de lancer une tâche en parallèle et d'en attendre la fin. On va utiliser la classe thread disponible dans le fichier d'entête <thread>. Il suffit de passer une routine en argument du constructeur de la classe thread. La méthode join() sur l'objet thread permet d'en attendre la fin.

```
void ThreadFunc()
{
  // ../..
}

void TestThread()
{
  thread t(ThreadFunc);
  std:thread::id id = t.get_id();
  t.join();
  cout << "TestThread: TID:" << id << ", " << "Main:" << GetCurrentThreadId() << endl;
}
```

Dans l'exemple ci-dessus, la fonction du thread ne prend pas de paramètres. Cependant, dans certains cas, on veut pouvoir passer des paramètres au thread. Il suffit de passer les paramètres au constructeur de l'objet thread :

```
class Param
{
public:
  string s;
  int i;
  float f;
};

void ThreadFunc2(Param param)
{
  cout << param.s << ", " << param.i << ", " << param.f << endl;
  // ../..
}

void TestThreadWithParam()
{
  Param param = { "C++", 14, 50.0 };
  thread t(ThreadFunc2, param);
  t.join();
}
```

Il existe des classes de type verrou comme mutex ou unique_lock<T> pour protéger l'accès aux données partagées. Vous pouvez remarquer que la gestion des threads est plutôt simple à utiliser.

Conclusion

Nous avons balayé les différentes composantes de la librairie standard et le constat est le suivant : le code source de la STL est complexe car c'est un framework extensible ; on l'a vu sur iostream et le passage d'une structure à cout ; qu'il faut apprendre à maîtriser. A partir des exemples simples présents dans cet article, vous n'avez plus qu'à vous lancer. Il y a des domaines que je n'ai pas couvert et c'est volontaire mais sachez que cet article couvre 90% de la STL. L'aspect le plus important est la maîtrise des containers. Ne construisez plus vos propres structures de list, hashtable ou autres, utilisez la STL ! La seule utilisation de vector<T> vous fait rentrer de plein pied dans la STL. Vous aurez des performances inégalables. N'utilisez plus les delete et passez aux smart pointers ! unique_ptr<T> et shared_ptr<T> justifient ; comme vector<T> ; une utilisation systématique. La fin des memory leak (fuite mémoire) en est une autre justification. Un conseil : laissez le code un peu ancien (legacy) avec un style à la papa, et sur le nouveau code, utilisez la STL. Le nouveau code doit être codé avec tous les artifices du C++ 11. Le code est plus lisible, les performances sont au rendez-vous. « Power & performance » comme disait Herb Sutter, chairman du standard C++ ISO. ⊞

Le mouvement Craftsmanship : un moyen de garantir un code de qualité et bien documenté.

Il est fréquent de constater que, quelle que soit la méthode choisie, la durée de vie des projets informatiques dépasse rarement cinq ans : code non maintenable, dette technique incontrôlable, connaissance perdue au gré des départs des "développeurs clés". Cet amer constat a poussé les premiers concernés, à savoir les équipes de développement, à agir. C'est ainsi qu'est né le mouvement Software Craftsmanship.

David Caramelo,

Marie-Laure Thuret

et
Fatiha Bouad

Consultants **Xebia**

Nous vous proposons de découvrir certaines pratiques issues du Craftsmanship qui vous permettront de documenter efficacement une application et vous garantiront une base de code saine et de qualité. Enfin, vous verrez comment transformer une équipe de développeurs en équipe de craftsmen accomplis.

Documenter ou ne pas documenter ?

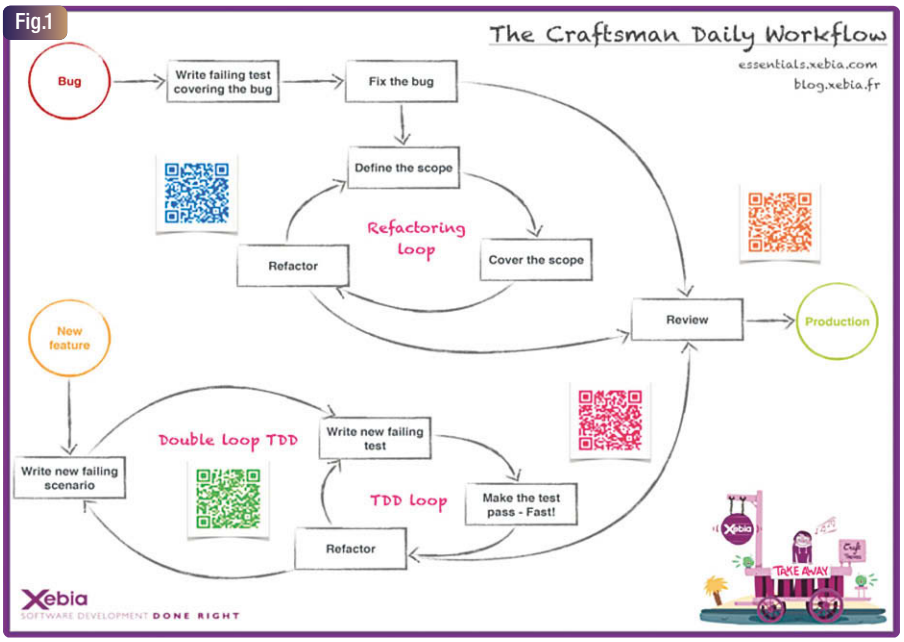
Traditionnellement, la réponse à cette question consiste à rédiger de la documentation. Elle se retrouve alors disséminée dans deux endroits :

- Dans les spécifications métiers afin qu'un développeur sache ce qu'il doit développer.
- Dans le code, à travers des commentaires supposés afin d'aider la compréhension.

Les problèmes de la documentation classique

Supposons qu'un développeur reçoive une spécification métier incomplète, puis un complément d'informations oral, ou bien qu'un bloc de code contenant un commentaire soit copié-collé puis son comportement modifié. Que se passe-t-il si la documentation n'est pas mise à jour au même moment ? Elle devient obsolète et la plupart du temps, cela se produit dès lors qu'on la considère comme achevée. On peut se dire qu'il suffit de faire attention afin de la garder synchronisée avec l'état actuel de l'application, mais il serait impossible de le garantir. Une documentation dans laquelle on ne peut pas avoir confiance et qui peut potentiellement devenir trompeuse, n'est pas une solution envisageable. Cependant, tout n'est pas perdu. Une application dispose en effet d'un élément qui reflète à n'importe quel moment la réalité : son code source. Après tout, pourquoi ne pas s'en servir directement à des fins documentaires ? Lire du code est une tâche difficile. Même s'il est évidemment possible pour un développeur de comprendre les fonctionnalités métiers d'une application en lisant son code, celui-ci contient de nombreuses abstractions qui peuvent rendre ce processus long et compliqué. On va plutôt chercher à s'appuyer sur des tests automatisés, qui sont plus directs et descriptifs que le code de production. Ils constituent un point d'entrée non négligeable pour comprendre un système. On appelle cela créer une documentation vivante, reflet du code actuel.

Fig.1



Comment créer une documentation vivante ?

Tous les types de tests sont concernés et seule la façon de les rédiger changera en fonction de la manière dont on décrit du comportement métier ou du comportement technique. Il faut cependant faire l'effort de rédiger ses tests avec attention et clarté. En effet, pour qu'ils aient une valeur documentaire, ils doivent être accessibles et faciles à comprendre. Lorsqu'il s'agit d'exprimer le comportement métier de l'application, ils doivent être écrits sous forme de spécifications et utiliser le langage du domaine.

Écrire une documentation revient à s'intéresser à deux aspects d'une application en particulier :

- Capturer son comportement,
- Capturer ses concepts métiers.

Cela tombe bien car il existe deux pratiques importantes du mouvement Craftsmanship qui vont vous permettre de couvrir ces deux points : le Behavior Driven Development et le Domain Driven Design.

Behavior Driven Development

Le Behavior Driven Development (BDD) est l'art de rédiger de façon collaborative les spécifications d'une application. On parle aussi de spécification par l'exemple, car les scénarios sont illustrés à l'aide de cas précis. Concrètement, il s'agit de faire en sorte que les trois acteurs types d'un projet (représentant du métier, développeurs et testeurs) travaillent ensemble pour faire émerger le comportement de l'application. Cette pratique vise à favoriser la compréhension des fonctionnalités par tous.

Les scénarios sont décrits en utilisant un langa-

ge spécifique appelé Gerkin. Il a pour particularité d'être lisible par n'importe qui et suit la syntaxe suivante :

- Given : décrit l'état initial du système ;
- When : décrit l'action effectuée ;
- Then : décrit le résultat attendu.

Le Gerkin impose un formalisme obligeant tout le monde à partager un langage commun.

Une fois ces scénarios déterminés, il existe de nombreux outils permettant aux développeurs de les "brancher" sur leur application afin de vérifier que celle-ci réponde correctement aux comportements attendus. Parmi les plus connus, citons par exemple Cucumber et JBehave.

Domain Driven Design

Le Domain Driven Design (DDD) est une approche initialement décrite par Eric Evans qui part du constat suivant : afin de réaliser un bon logiciel, on doit être en capacité d'en comprendre son domaine. Tout comme le BDD, il faut que les développeurs et les experts métiers se parlent et collaborent. L'ensemble du code doit être conçu, organisé et écrit, en vue de refléter le domaine pour lequel il existe (découpage et nom des classes, nom des packages, des méthodes, etc). Qui n'a jamais travaillé sur une application où un même concept se retrouve non seulement nommé de manières complètement différentes à l'intérieur du code lui-même, mais aussi dans les différents documents créés et uploadés sur le file sharing de l'entreprise ?

Qui n'a jamais travaillé sur une application où le domaine métier est relégué derrière les spécificités techniques, et où les premiers niveaux des répertoires sont "controllers", "services" ou bien encore "model" ? Le DDD fait émerger la nécessité de définir et figer un langage et un glossaire commun afin de mieux appréhender la complexité du domaine métier de l'application. On parle d'Ubiquitous Language. Il faut tout de même noter que le DDD est une pratique vivante. Les concepts et mots retenus initialement décrivant le domaine d'une application vont évoluer avec le temps et les nouveaux besoins métiers.

Ainsi, le BDD et le DDD nous permettent de créer une documentation vivante : notre application décrit son comportement et son domaine métier. Il faut maintenant être capable d'ajouter facilement de nouvelles fonctionnalités tout en garantissant la qualité de l'application.

Coder proprement : une boîte à outils

Le mouvement Craftsmanship prône l'écriture d'un code propre à travers l'adoption de bonnes pratiques dont la plupart sont extraites de l'eXtrême Programming (XP). Bien entendu, la mise en place de ces pratiques prend du temps, les développeurs doivent changer leurs habitudes de développement et adhérer à ces nouveaux changements.

Test First

Cette notion correspond à l'écriture d'un test unitaire avant l'implémentation. Il décrit le comportement du code que l'on veut rédiger, garantissant que le code produit par la suite sera de qualité, et cohérent avec les attentes. Cela permet de ne coder que ce qui est nécessaire et de rester simple comme le préconise le principe KISS (Keep It Simple Stupid). Écrire les tests avant l'implémentation permet aussi de faire émerger un design, de se focaliser sur les fonctionnalités et de repérer rapidement les régressions.

Test Driven Development(TDD)

Cette pratique découle de la précédente (Test First). La particularité du TDD est qu'elle est basée sur un cycle « Red/Green/Refactor » :

- Red : le développeur écrit un test automatisé qui teste le comportement attendu de la nouvelle fonctionnalité. Dans un premier temps, ce test doit échouer car le code permettant de le satisfaire n'est pas encore rédigé.
- Green : le développeur écrit le minimum de code nécessaire permettant au test précédent d'être exécuté avec succès.
- Refactor : le développeur améliore le code précédemment écrit tout en maintenant les tests en succès.

Double Loop TDD

C'est une double boucle TDD qui consiste à ajouter d'abord un test d'acceptance (par exemple un scénario BDD) qui va entraîner une boucle TDD jusqu'à l'implémentation complète de la fonctionnalité ajoutée par le test d'acceptance Fig.1.

Refactoring

Le refactoring consiste à revoir de manière continue, la conception et la lisibilité du code en vue de son amélioration. Lorsque l'on achève le développement d'une fonctionnalité, revenir sur le code pour éliminer des “Code Smell” et améliorer son design semble être une bonne approche. Dans cette situation, la présence de tests se révèle un atout indispensable. Intervenir sur du code existant devient alors possible, avec l'assurance de ne pas introduire de régressions.

Pair-Programming

L'utilisation d'une seule machine pour deux développeurs, favorisant le partage de la connaissance métier et technique, garantit la montée en compétence globale de l'équipe.

Cette pratique est souvent critiquée car de prime abord, on a l'impression que la fonctionnalité coûte plus cher que si elle avait été implémentée par une seule personne. Mettre en évidence les difficultés que présente la fonctionnalité à être développée et résoudre les problèmes via deux angles différents représentent un réel bénéfice. Les échanges et connaissances de chacun des développeurs permettent d'aboutir à des implémentations plus efficaces, à du code plus lisible, de meilleur qualité et donc d'éviter de générer de la dette technique.

Code review

La relecture du code peut se faire par un équipier ou l'ensemble de l'équipe. C'est un autre moyen qui, même s'il est moins dynamique que le pair-programming, permet de partager la connaissance avec toute l'équipe, et de faire émerger des critiques constructives afin de retravailler le code en conséquence.

Les outils

1) L'intégration Continue

Cette pratique permet de vérifier rapidement que les changements apportés par les membres d'une équipe restent cohérents et n'entrent pas en collision. À chaque fois qu'un développeur pousse un changement sur le dépôt de code, tous les tests sont exécutés pour vérifier qu'aucune régression n'a pas été introduite. Pour être efficace, cette pratique doit être combinée avec celle du "Stop & Fix". C'est à dire que tous les membres de l'équipe doivent corriger en priorité les erreurs remontées par l'intégration continue avant de reprendre leurs tâches en cours.

2) Indicateur de qualité

Mesurer la qualité du code est difficile. Le plus simple est de choisir des indicateurs permettant de dresser un « carnet de santé ». L'outil le plus connu dans ce domaine est Sonar. Beaucoup de ces indicateurs permettent de mesurer et d'identifier la dette technique. Nous pouvons citer :

- Les portions de code non utilisées ;
- Les erreurs mal gérées ;
- La couverture des tests qui est un ratio entre la quantité de code et les fonctionnalités testées.

Bien que toutes ces pratiques aient des bienfaits, il ne faut pas chercher à les appliquer systématiquement. Si elles ne sont pas nécessaires ou que de meilleures alternatives existent, il faut savoir rester pragmatique. C'est d'ailleurs l'une des qualités indispensables d'un craftsman. Il faut constamment se demander s'il n'existe pas des pratiques plus efficaces ou plus adaptées à sa situation. La meilleure façon de comparer ces pratiques est de comparer leurs valeurs ajoutées.

Les individus et leurs interactions au centre de vos préoccupations

Pour réussir un projet, l'équipe doit disposer des bonnes compétences, mais cela n'est pas suffisant. En effet, fonctionner de manière totalement isolée et sans réelle interaction avec les membres de l'équipe représente pour le projet des risques de ne pas aboutir. Le produit fonctionnera sûrement, mais il ne disposera d'aucune cohérence et sera difficilement maintenable.

Pour que votre projet soit couronné de succès, la recette à suivre est la suivante :

- Une complexité adaptée, pas "d'over-engineering " : Keep It Simple Stupid (KISS) ;
- Une documentation à jour via une documentation vivante ;
- Un code propre et lisible grâce au Pair Programming et à la Code Review.

Cependant, en ce qui concerne la mise en oeuvre, il manque un liant : "Les individus et leurs interactions" qui forment l'un des principes fondamentaux de l'agilité. En temps que Craftsman, cela doit être au centre de vos préoccupations. Les Code Review et le Pair-Programming ont pour but de favoriser les échanges durant les phases de développement mais cela reste limité. En complément, il faut être capable d'insuffler un état d'esprit dans l'équipe ; pour cela, soyez moteur afin de créer d'autres points de rencontre.

Les sessions d'échanges ou chapters

Les sessions d'échanges ne doivent pas être prises à la légère, il faut vous réserver une plage horaire dédiée afin d'échanger sur vos dernières trouvailles et sur les outils qui vous font gagner du temps. Les informations échangées durant ces sessions n'ont pas de prix.

Sessions de veille technologique

Prenez d'assaut les salles de réunion le midi afin de regarder les dernières conférences Web ou d'organiser un Brown Bag Lunch (BBL). Ces sessions au format conférence ou table ronde, voient généralement la présence d'un speaker désigné, interne ou externe à l'entreprise, qui présentera pendant 1 à 2 heure(s), un sujet qu'il maîtrise.

Les coding dojos

Puisque c'est en forgeant que l'on devient forgeron, quoi de mieux que de coder pour s'améliorer ? Lors d'un Coding Dojo, regroupez-vous pour vous entraîner sur un problème en un temps limité. À la fin du temps imparti, le code est effacé, puis on recommence à coder avec un autre partenaire de travail. Le but est ainsi de se perfectionner et de découvrir de nouvelles techniques de conception ou de programmation. Ce simple moment de partage de connaissances peut s'avérer très bénéfique et enrichissant.

Échanger avec la communauté

Les User Groups, Meetups et Tech Events

Ces dix dernières années ont vu l'apparition de nombreuses guildes modernes : les "User Groups". Ces groupes de travail réunissent périodiquement des utilisateurs passionnés autour d'une même thématique technologique.
Ouverts à tous, ils permettent des sessions d'échange riches et vivantes. Que vous soyez simple spectateur ou bien acteur de ces communautés, les craftsmen pourront continuer à apprendre au contact de pairs.
Au delà de ces réunions mensuelles, la communauté s'est aussi organisée sur une échelle plus large en créant des conférences annuelles partout en France (Devoxx Paris, Mix-It, Breizh-Camp, XebiCon) et en Europe (Devoxx Anvers, GeeCon, Jax). Elles réunissent des milliers de passionnés et, sur un ou plusieurs jours, abordent des thèmes variés. Elles permettent de cumuler à grande échelle tous les bienfaits de la veille technologique :

- Aborder et défricher de nouveaux sujets,
- Confronter sa vision avec une vision extérieure,
- Se bâtir un réseau de "sachants",
- Apprendre des retours terrain sur la mise en place des nouvelles technologies.

Les MOOC

Autrefois plébiscités par des étudiants de tous âges, l'Université de Tous les Savoirs (des cours universitaires gratuits et accessibles à tous) se retrouve aujourd'hui massivement en ligne à travers les Massive Open Online Courses. Pour une fois, les cordonniers ne sont pas les plus mal chaussés car une grande partie de ces cours en ligne traite d'informatique. Si vous avez la volonté d'apprendre, il y a forcément une ressource en ligne qui vous permettra d'assouvir votre curiosité.

Les hackathons

Vous avez sûrement entendu parler des hackathons. Sur un jour ou un week-end entier, un hackathon est un challenge d'innovation vous proposant de développer un logiciel en équipe, dans un délai imparti. À la fin du concours, chaque équipe présente son travail à un jury, constitué le plus souvent de dirigeants. Le produit le plus convaincant, le plus innovant ou le plus fun aura une récompense.
Ces évènements offrent l'opportunité de progresser collectivement via divers biais :

- Renouveler les équipes : de par leur format, les hackathons proposent souvent de faire collaborer des personnes qui d'ordinaire se trouvent dans des pôles différents. C'est une bonne occasion de favoriser la mixité.
- Proposer un cadre plus libre : généralement, la seule règle imposée par le hackathon est le thème. Le choix des frameworks de développement et l'idée précise est laissée aux équipes. Il s'agit souvent d'une belle occasion pour expérimenter de nouvelles technologies, voire mettre en avant des idées innovantes d'un point de vue fonctionnel.
- Faire émerger des produits et des cas d'usages innovants : c'est souvent le but recherché par les hackathons.

Pour résumer la réussite d'un projet passe par :

- La documentation qui, à l'image d'un mode d'emploi, ne doit pas différer du produit : il faut que la documentation vive avec le produit.
- La mise en oeuvre des techniques telles que le Double Loop TDD, le pair-programming ou encore les Coding-Dojo permet de produire un code de qualité et ceci de manière interactive.
- Favoriser la collaboration et la communication entre les différents acteurs afin d'assurer la cohérence de votre projet.

Le développement logiciel n'est qu'un maillon de la chaîne qui conduit au succès d'un produit. Il existe également des pratiques pour bien définir son besoin et gérer l'exploitation du produit de façon efficace ; jetez un oeil aux pratiques XP ou Agiles telles que Scrum, Kanban.
Fini le temps où il fallait essayer de tout prévoir à l'avance et de cadrer l'ensemble du projet, qui, de plus, enlève la souplesse et toute possibilité d'innovation. Cette devinette est insoluble, il y a trop de facteurs inconnus ou variables. Aujourd'hui, le produit, l'équipe, le code, la documentation doivent vivre et évoluer ensemble. Et vous, vous commencez quand ? ⊞

Office 365 : Delve, graph, vidéo

Il n'aura certainement échappé à personne que la stratégie de Microsoft a grandement évolué ces dernières années. Très centrée sur Windows par le passé, Microsoft mise aujourd'hui beaucoup sur les services, notamment autour du Cloud, qui gagnent chaque jour des parts de marchés dans le monde entier. Parmi les services sur lesquels mise énormément la firme de Redmond, il y a ceux tournant autour du social et de la collaboration avec comme fer de lance Yammer et Office 365. C'est sur ce dernier que nous allons nous concentrer car encore assez peu utilisé en Europe, et pourtant si riche et pouvant répondre aux besoins de bons nombre de personnes, qu'il s'agisse de particuliers ou bien d'entreprises (petites ou grandes).

Stéphane Cordonnier
MVP Office Development
http://blog.beecomedigital.com

Quels services sont offerts par Office 365 ?

Lancé en 2009 sous l'appellation *Business Productivity Online Services*, à l'époque à destination des entreprises uniquement, Office 365 a depuis évolué pour offrir un large choix de services parmi lesquels :

- Messagerie électronique (Outlook / Exchange)
- Messagerie instantanée / Vidéoconférence (Skype for Business)
- Espaces collaboratifs (SharePoint / OneDrive for Business)
- Réseaux sociaux (Yammer)
- Outils bureautiques (Office)

Il s'agit d'une offre en mode *Software as a Service* (SaaS) qui permet à qui y souscrit, de choisir quels services lui sont utiles afin de ne payer que ce qui est vraiment nécessaire à son besoin. Par exemple, il est possible de s'abonner pour bénéficier d'une boîte à lettre (50 Go de stockage par défaut) accompagnée de la suite Office (Word, Excel, PowerPoint...) pour quelques euros mensuels. Si l'ensemble des services vous intéresse, des packs tout compris à des tarifs plus avantageux sont également proposés. Tous les services mentionnés précédemment peuvent être utilisés aussi bien sur son ordinateur, au travers d'un simple navigateur Internet, que depuis ses périphériques mobiles (téléphone et tablettes) grâce aux très nombreuses applications mises à disposition par Microsoft (aucun coût supplémentaire à prévoir pour les utiliser).

Et pour les développeurs ça permet de faire quoi ?

Au vu de la typologie du magazine que vous êtes en train de lire, vous vous demandez probablement ce qu'Office 365 peut également apporter à un développeur ? Pour chacun des services évoqués, Microsoft met à disposition des APIs permettant aux développeurs de pouvoir concevoir des applications tirant partie des fonctionnalités exposées par chacun des services. Cela permet par exemple de pouvoir embarquer dans vos propres applications, souvent des applications professionnelles mais pas nécessairement, des fonctionnalités comme du stockage documentaire, de l'envoi de mail, de la consultation de fichiers Office, etc...

Les APIs en questions peuvent être utilisées de différentes manières :

- Via les SDKs mis à disposition par Microsoft (.NET, Objective-C, Java...)
- Via des appels REST (simples requêtes HTTP)

Les applications que vous pouvez développer grâce à ces APIs peuvent s'exécuter directement dans Office 365 (ex : un plugin à Outlook Web App pour interagir avec les mails reçus), comme applications Web classiques (ex : une interface simplifiée de partage de documents stockés dans SharePoint) ou bien encore comme applications mobiles (ex : un client iOS mélangeant Yammer et Skype for Business). Mais au-delà du développement Office 365, cela ouvre de nouvelles perspectives autour d'autres produits comme la suite Office pour laquelle Microsoft a fait évoluer le modèle de développement ces dernières années. Pour ceux qui ont connu les fameuses macros VBA (Visual Basic for Applications), il faut désormais parler des Office Add-ins, qui permettent d'enrichir Word, Excel, PowerPoint, etc... de nouvelles fonctionnalités qui peuvent, pourquoi pas, interagir avec Office 365. Mais le développement de ces Add-ins n'est pas nécessairement connecté à la plateforme Office 365 car aujourd'hui de nombreux éditeurs proposent des connecteurs vers des plateformes CRM (ex : Salesforce), des réseaux sociaux professionnels (ex : Linkedin), de la cartographie (ex : Bing Maps / Google Maps), etc.

Comme vous pouvez le voir, de très nombreuses possibilités sont offertes aux développeurs qui souhaitent s'investir dans cette voie, que ce soit pour développer des applications professionnelles, ou bien des add-ins à la suite Office. Si vous souhaitez de plus amples informations sur les APIs et SDKs disponibles pour Office 365, Microsoft a mis à disposition ceux-ci sur GitHub (https://github.com/OfficeDev). Un portail dédié aux développeurs a également été créé (http://dev.office.com) avec notamment de nombreux tutoriaux et exemples de code.

Focus sur Delve / Office Graph et Office 365 Vidéo

Comme nous l'avons évoqué ci-dessus, de très nombreux services sont offerts par Office 365 et nous pourrions consacrer un magazine entier à présenter les possibilités de développement autour de la plateforme.

Dans la suite de cet article, nous allons nous concentrer sur 2 services particuliers, parmi les derniers à avoir été lancés sur Office 365 et qui sont offerts par le biais de la brique collaborative (SharePoint) :

- Delve / Office Graph qui permet d'étendre et d'enrichir les capacités et fonctionnalités du moteur de recherche
- Office 365 Vidéo qui propose un service streaming vidéo construit sur Azure Media Services (utilisé lors de la diffusion des derniers Jeux Olympiques)

Pour chacun de ces services, nous présenterons ce qu'ils offrent, comment ils fonctionnent et surtout comment les utiliser dans nos propres applications à l'aide des APIs REST mises à disposition par Microsoft.

Présentation de Delve / Office Graph

Comme tout outil collaboratif, Office 365 par l'intermédiaire de sa brique SharePoint, possède des fonctionnalités de stockage documentaire et un moteur de recherche permettant théoriquement de retrouver simplement et rapidement un fichier. Le souci est généralement qu'après plusieurs mois ou années d'utilisation, des centaines de milliers de documents se retrouvent stockés, sans forcément avoir été correctement classés. Il devient donc difficile de retrouver l'information pertinente pour un utilisateur. C'est sur ce terrain que va venir se positionner Delve, qui permet au travers d'interfaces dédiées à cet effet, de pousser l'information pertinen-

te vers l'utilisateur, plutôt que d'attendre que celui-ci exécute la bonne requête pour retrouver un document. Comment cela fonctionne ? Grâce au moteur Office Graph de Microsoft, pas spécifique à Office 365 puisque Microsoft compte l'utiliser dans d'autres produits, mais dont Office 365 est l'un des premiers à tirer parti. Office Graph est un moteur de « Machine Learning », sujet très à la mode ces derniers temps qui se base sur de nombreux critères (centres d'intérêts de l'utilisateur, consultation de documents, édition de documents, relations et interactions avec ses collègues, etc.) afin de déterminer quels contenus seraient les plus à même d'intéresser tel ou tel utilisateur. Ces données sont ensuite exploitées par Delve pour permettre une présentation en adéquation avec les services disponibles sur Office 365 (documents, vidéos, billets de blogs...) Fig.1. Là où la chose devient intéressante, c'est qu'il est possible d'intégrer au sein de ses propres applications, la puissance offerte par Office Graph, Delve et les APIs mises à disposition par Microsoft.

Authentification et utilisation des APIs Office 365

Pour pouvoir utiliser les APIs disponibles, et comme sur toutes les plateformes disponibles à ce jour (Facebook, Twitter, DropBox...), il est nécessaire de s'authentifier pour pouvoir manipuler les données stockées dans Office 365. Différents scénarios sont utilisables sur la plateforme, en fonction des besoins des entreprises notamment, mais nous nous concentrerons sur celui par défaut et le plus utilisé, à savoir l'utilisation d'Azure Active Directory (généralement nommé AAD) et du protocole OAuth2.

Lorsque vous souscrivez un abonnement Office 365, Microsoft met à votre disposition ce qu'on appelle un « tenant », qui sert à isoler vos données des autres clients utilisant la plateforme. L'authentification de chaque tenant est assurée par AAD, qui propose un console d'administration (https://manage.windowsazure.com) permettant d'accéder aux différentes informations tels que les comptes utilisateurs, groupes de sécurité, listes de distribution, etc. Pour pouvoir utiliser les APIs dans vos applications, vous devez accéder à cette console, déclarer votre application et définir de quelles autorisations celle-ci a besoin pour fonctionner lorsqu'elle accédera aux données Fig.2. Dans le cadre de l'utilisation de Delve / Office Graph, qui étendent les capacités du moteur de recherche, de simple droits de lecture sur les données suffisent à pouvoir utiliser les APIs.

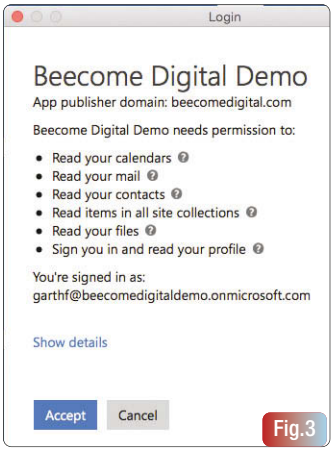


Fig.3

Une fois votre application déclarée dans la console AAD, vous êtes à même de pouvoir utiliser les APIs, mais pour cela encore faut-il que l'utilisateur s'authentifie. C'est là que les SDKs dont nous parlions en introduction, vont prendre tout leur sens. Comme nous l'évoquions juste avant, l'authentification sur Office 365 repose sur OAuth2. Pour les personnes non familières avec ce protocole, il se base sur la génération d'un jeton (qui a généralement une durée de vie limitée) qui doit ensuite être transmis à chaque requête afin que le serveur valide l'identité et les droits de l'utilisateur. Il est bien évidemment possible de développer soi-même une couche d'authentification OAuth2, mais Microsoft met à disposition dans ses SDKs pour différentes plateformes, une bibliothèque nommée « Azure AD Authentication Library » (également appelée ADAL). Cette bibliothèque permet de gérer pour vous tout le mécanisme d'acquisition et de mise en cache de jetons d'authentification, en vue de leur utilisateur avec les APIs. Cette bibliothèque est disponible pour différentes plateformes (.NET, JavaScript, Objective-C, Java...) et est relativement simple à manipuler. Ci-dessous un exemple de la version Objective-C pour iOS et OS X.

```
NSString *clientId = @"12345678-abcd-1234-abcd-1234567890ab";
NSURL *redirectUri = [NSURL URLWithString:@"https://beecomedigitaldemo"];

   ADAuthenticationContext *authenticationContext = [ADAuthenticationContext
authenticationContextWithAuthority:@"https://login.windows.net/common" error:nil];
   [authenticationContext acquireTokenWithResource:@"https://tenant.sharepoint.com/"
clientId:clientId redirectUri:redirectUri completionBlock:^(ADAuthenticationResult *result) {
     if (result.status == AD_SUCCEEDED) {
       // Authentication succeeded
     }
     else if (result.status == AD_USER_CANCELLED) {
       // Authentication cancelled by the user
     }
     else {
       // Authentication failed
     }
  }];
```

Lorsque l'utilisateur exécute l'application, celui-ci est invité à s'authentifier via le formulaire Web mis à disposition par Microsoft, s'il ne l'a jamais fait ou si son jeton d'authentification est expiré. Une fois authentifié, il suffit de récupérer le jeton d'authentification et de l'ajouter à sa requête lors d'un appel REST.

```
NSURL *serviceURL = [NSURL URLWithString:@"https://tenant.sharepoint/_api/site"];
NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:serviceURL];
[request setValue:[NSString stringWithFormat:@"Bearer %@", result.accessToken]
forHTTPHeaderField:@"Authorization"];
 [httpClient dataWithRequest:request completionHandler:^(NSHTTPURLResponse *response,
 NSData *data, NSError *error) {
   if (error == nil) {
     id jsonObject = [NSJSONSerialization JSONObjectWithData:data options:0 error:nil];
     NSLog(@"%@", jsonObject);
   }
}];
```

Pour renforcer la sécurité et afin de s'assurer qu'une application ne s'exécutera pas avec plus de droit que cela est nécessaire, l'utilisateur doit donner son consentement lors de la phase d'authentification. Ainsi lors de la

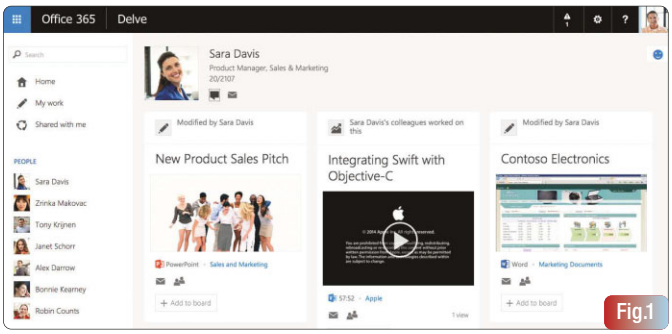


Fig.1

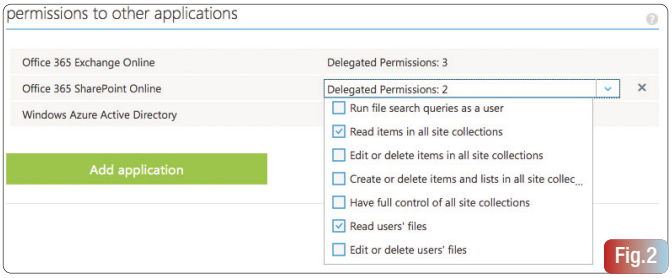


Fig.2

récupération initiale du jeton, l'utilisateur se verra proposer une boîte de dialogue comme celle-ci, qui récapitule ce que l'application en question aura comme permission si on l'autorise à s'exécuter Fig.3.

Utilisation des APIs Office 365 pour Delve / Office Graph

Maintenant que nous sommes à même de nous authentifier et d'exécuter des appels REST, voyons en détail le fonctionnement des APIs Office 365 pour interroger Delve / Office Graph. Afin de simplifier l'utilisation, Microsoft s'est reposé sur les APIs de SharePoint, et notamment les APIs d'accès au moteur de recherche pour permettre de récupérer les données souhaitées. Il existe 2 manières de les utiliser :

- Soumettre une requête GET à https://tenant.sharepoint.com/_api/search/query
- Soumettre une requête POST à https://tenant.sharepoint.com/_api/search/postquery

Les 2 méthodes permettent d'arriver au même résultat. Toutefois dans le cas d'une requête très longue et très complexe, la méthode GET comporte une limite, puisqu'une URL ne peut comporter plus de 4000 caractères. Nous utiliserons donc la seconde dans notre exemple. Pour interroger le moteur de recherche de SharePoint, il suffit de créer un message JSON comme ci-dessous et de le soumettre à l'URL évoquée précédemment.

```
{
  "request": {
    "Properties": [
      {
        "Name": "GraphQuery",
        "Value": {
          "QueryPropertyValueTypeIndex": 1,
          "StrVal": "ACTOR(ME,action:1021) "
        }
      },
      {
        "Name": "GraphRankingModel",
        "Value": {
          "QueryPropertyValueTypeIndex": 1,
          "StrVal": "action:1021,weight:1,edgeFunc:weight,mergeFunc:max"
        }
      }
    ],
    "QueryTemplate": "({searchterms}) AND ((NOT HideFromDelve:True) AND (FileExtension
:doc OR FileExtension:docx OR FileExtension:xls OR FileExtension:xlsx OR FileExtension:ppt
 OR FileExtension:pptx OR FileExtension:one OR FileExtension:pdf OR ContentTypeId:0x010
100F3754F12A9B6490D9622A01FE9D8F012*))",
    "Querytext": "*",
    "RankingModelId": "0c77ded8-c3ef-466d-929d-905670ea1d72",
    "RowLimit": 20,
    "StartRow": 0,
    "Timeout": 30000,
    "TrimDuplicates": true
  }
}
```

Attardons nous sur les parties importantes de cette requête :

- QueryText è Nous voulons tous les documents possibles via une recherche wildcard, mais il est possible de soumettre une recherche plus précise (ex : Title:Demo pour les documents dont le titre contient la valeur Demo) ;
- RankingModelId è Le modèle de classement spécifique à Delve qui permet d'utiliser les informations venant d'Office Graph pour calculer la pertinence des résultats ;
- QueryTemplate è En plus de la recherche wildcard, nous filtrons quels documents sont renvoyés (Word, Excel, PowerPoint, PDF et Videos) ;
- GraphRankingModel è Sur quel critère le classement est-il basé ? Dans cet exemple nous prenons le poids du résultat mais il est possible de se baser sur la date si cela est préférable ;

Reste la partie la plus importante de la requête, à savoir celle nommée « GraphQuery » qui comme vous le constaterez, utilise une syntaxe particulière. En effet, afin de permettre une écriture relativement simple pour les requêtes, Microsoft a créé un nouveau langage de requête nommé GQL (Graph Query Language). Ce langage repose sur 3 éléments fondamentaux à savoir :

- Un acteur è La source qui nous intéresse (généralement l'identifiant d'un utilisateur) ;
- Un objet è La destination (un document, un tag...) ;
- Une action è Pourquoi/Comment la source est-elle liée à la destination ?

Ces 3 éléments ne sont pas toujours présents dans les requêtes (nous n'en avons que 2 dans notre exemple), mais ils influent sur les résultats renvoyés. Actuellement dans Delve / Office Graph, différents types d'actions sont supportées. Ci-dessous la liste de celles qui peuvent être utilisées dans les requêtes.

ID	Action	Visibilité
1021	Mur personnel de l'utilisateur	Privé
1003	Les documents modifiés par l'utilisateur dans les 3 derniers mois	Public
1015	Les collègues de l'acteur	Public
1014	Les subordonnés de l'acteur	Public
1013	Le responsable hiérarchique de l'acteur	Public
1016	Le N+2 de l'acteur	Public
1019	Les personnes avec qui l'acteur communique ou travaille régulièrement	Privé
1020	Les données populaires pour l'acteur	Public
1001	Les documents vus par l'utilisateur dans les 3 derniers mois	Privé
1003	Une version publique de l'action 1019	Public

L'écriture de requêtes tirant parti d'Office Graph repose sur la combinaison de ces différents éléments au sein du paramètre GraphQuery évoqué précédemment. Voyons quelques exemples de requêtes et de combinaisons possibles. Dans les requêtes qui suivront, la valeur ME correspond à l'utilisateur courant (celui qui s'est authentifié) tant que la valeur 2962 correspond à l'identifiant d'un utilisateur nommé Carl.

- ACTOR(ME) è Les 10 premiers éléments (modifiable via la valeur RowLimit) me concernant ;
- ACTOR(2962) è Les 10 premiers éléments concernant Carl ;
- ACTOR(ME, action:1013) è Qui est mon responsable hiérarchique ;
- ACTOR(ME, OR(action:1001,action:1003)) è Les 10 premiers éléments que j'ai vus ou modifiés durant ces 3 derniers mois ;
- ACTOR(ME, AND(action:1003, time:datetime(2015-10-01))) è Les 10 premiers éléments que j'ai modifiés le 1er Octobre 2015 ;
- ACTOR(ME, AND(action:1003, time:range(datetime(2015-08-01),max))) è Les 10 premiers éléments que j'ai modifiés entre le 1er Août 2015 et aujourd'hui ;
- AND(ACTOR(ME), ACTOR(2962)) è Les 10 premiers éléments que j'ai en commun avec Carl
- AND(ACTOR(ME, action:1001), ACTOR(2962, action:1003)) è Les 10 premiers éléments que j'ai vus ces 3 derniers mois et qui ont été modifiés par Carl durant la même période.

Lorsque vous exécutez ces requêtes, le moteur de recherche de SharePoint vous renvoie les résultats sous forme d'un flux JSON contenant de très nombreuses informations.

Pour simplifier le résultat et rester concis, ci-dessous une extraction des données concernant un document :

```
{
Key: "Rank",
Value: "6.60530138015747",
ValueType: "Edm.Double"
},
{
Key: "DocId",
Value: "13124569",
ValueType: "Edm.Int64"
},
{
Key: "Title",
Value: "Contoso Seattle Expansion",
ValueType: "Edm.String"
},
{
Key: "Path",
Value: "https://tenant.sharepoint.com/sites/contoso/Resources/Document Center/Documents/
Contoso Seattle Expansion.pptx"
ValueType: "Edm.String"
}
```


Il ne reste ensuite qu'à manipuler le contenu du résultat JSON renvoyé par le serveur afin de pouvoir déterminer quoi faire des données trouvées par le moteur de recherche en se basant sur les calculs de pertinence effectués par Office Graph. Delve et Office Graph sont des briques sur lesquelles Microsoft travaillet activement et autour desquelles de nouvelles fonctionnalités voient le jour tous les mois. Les APIs qui vont de concert avec ces services évoluent évidemment en conséquence.

C'est pourquoi il est fortement recommandé, pour être sûr d'être en permanence au fait des dernières évolutions, de consulter régulièrement la documentation officielle mise à disposition par Microsoft sur son portail pour les développeurs MSDN. L'article de référence concernant l'utilisation des APIs et de GQL se trouve à cette adresse si vous désirez en savoir plus : https://msdn.microsoft.com/en-us/office/office365/howto/query-Office-graph-using-gql-with-search-rest-api

Présentation d'Office 365 Vidéo

Depuis l'avènement des plateformes telles que YouTube, Vimeo, DailyMotion, etc., la vidéo est devenue un vecteur important dans la communication de tous les jours. Or, le souci, notamment dans le cadre d'une entreprise, concerne le fait de pouvoir diffuser des vidéos en interne pour ses salariés alors que la plupart des plateformes citées sont avant tout accessibles à tous sur Internet (même s'il existe des fonctionnalités de restriction d'accès). Microsoft a donc décidé de proposer dans Office 365, un service destiné uniquement aux entreprises (du moins pour l'heure), de stockage et de diffusion de vidéos, mais surtout totalement intégré à la plateforme utilisée quotidiennement par leurs utilisateurs.

Pour cela, un service a été créé de toutes pièces, se basant sur les fonctionnalités collaboratives de SharePoint (création de sites, stockage de fichiers, moteur de recherche, restriction d'accès...) et tirant parti d'Azure Media Services pour l'encodage et la lecture des vidéos.

Ainsi, par l'intermédiaire du portail Office 365 Vidéo, les utilisateurs de l'entreprise pourront créer des canaux pour classer et stocker leurs vidéos, télécharger et visualiser en streaming les supports de l'entreprise, ainsi et interagir de manière sociale via l'intégration de Yammer. Les services de communication/marketing de l'entreprise trouveront également un nouveau moyen de mise en valeur des produits/services de l'entreprise entre la possibilité de mettre en évidence (spotlight) des vidéos importantes de manière globale sur le portail, ou bien de manière spécifique à un canal.

De nouvelles fonctionnalités sont en cours de développement au sein des équipes de Microsoft, qui étend les capacités de la plateforme de manière très régulière (tous les mois généralement) et qui est également très à l'écoute des demandes clients pour faire évoluer son service Fig.4.

Utilisation des APIs Office 365 Vidéo

Bien évidemment, qui dit service intégré dans Office 365, dit ensemble d'APIs pour pouvoir intégrer celui-ci au sein de son système d'information. Office 365 Vidéo ne déroge pas à la règle et propose des interfaces REST pour manipuler (lecture / écriture) les médias stockés dans le portail.

L'utilisation de ces APIs repose sur le même principe que pour les APIs Delve/Office Graph, aussi bien au niveau de l'authentification (basée sur Azure AD/OAuth2) que sur les appels REST. Reportez-vous aux éléments précédemment évoqués si nécessaire.

Récupération de la liste des canaux disponibles

Chaque client possède, par l'intermédiaire de son tenant, un portail Office 365 Vidéo accessible via une URL sous la forme https://tenant.sharepoint.com/portals/hub/. L'accès à toutes les APIs en rapport à Office 365 Vidéo, se basera sur cette URL à laquelle sera ajouté le chemin des appels REST. Comme nous le disions précédemment, les vidéos sont organisées et stockées dans des canaux. La première chose à faire est donc de récupérer cette liste de canaux. Pour cela, il suffit de soumettre une requête à l'URL suivante :

https://tenant.sharepoint.com/portals/hub/_api/VideoService/Channels

L'appel de cette URL permet de récupérer tous les canaux sur lesquels l'utilisateur actuellement identifié (via le jeton OAuth2) possède des autorisations de lecture ou d'écriture. Cette liste de canaux est retournée sous la forme d'un flux JSON tel que celui-ci :


```
{
odata.metadata: https://tenant.sharepoint.com/portals/hub/_api/$metadata#SP.ApiData.VideoChannels,
value: {
odata.type: "SP.Publishing.VideoChannel",
odata.id: "https://tenant.sharepoint.com/portals/hub/_api/VideoService/Channels(guid'ac
12504d-f0a3-40bc-bc30-0aca60dcc8e2')",
odata.editLink: "VideoService/Channels(guid'ac12504d-f0a3-40bc-bc30-0aca60dcc8e2')",
Description: "",
Id: "ac12504d-f0a3-40bc-bc30-0aca60dcc8e2",
ServerRelativeUrl: "/portals/community",
TileHtmlColor: "#2A8DD4",
Title: "Community"
}
}
```

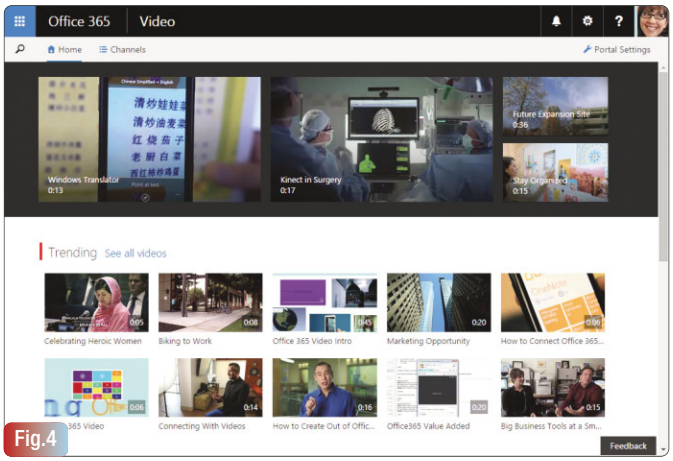



Fig.4

Comme nous le voyons, chaque canal est identifié par une URL stockée dans le champ « odata.id ». C'est à partir de cette URL que nous pouvons ensuite manipuler le contenu d'un canal.

Récupération des vidéos d'un canal

Une fois l'URL d'un canal connue, deux types d'informations peuvent nous intéresser :

- La liste de toutes les vidéos stockées dans le canal ;
- La liste des vidéos mises en avant (spotlight) dans le canal.

Pour cela rien de plus simple, il suffit d'appeler les URLs ci-dessous :

- https://tenant.sharepoint.com/portals/hub/_api/VideoService/Channels(guid'ac12504d-f0a3-40bc-bc30-0aca60dcc8e2')/Videos
- https://tenant.sharepoint.com/portals/hub/_api/VideoService/Channels(guid'ac12504d-f0a3-40bc-bc30-0aca60dcc8e2')/SpotlightVideos

Comme pour la liste des canaux, la réponse retournée au format JSON se présentera sous la forme (ci-dessous l'exemple de la liste des vidéos du canal) :

```
{
odata.metadata: "https://tenant.sharepoint.com/portals/hub/_api/$metadata#SP.ApiData.VideoItems"
value: {
odata.type: "SP.Publishing.VideoItem"
odata.id: "https:// tenant.sharepoint.com/portals/hub/_api/VideoService/Channels(guid'ac
12504d-f0a3-40bc-bc30-0aca60dcc8e2')/Videos(guid'd48666c4-09e1-49af-94ce-deb8e10a66ff')"
odata.editLink: "VideoService/Channels(guid'ac12504d-f0a3-40bc-bc30-0aca60dcc8e2')/
Videos(guid'd48666c4-09e1-49af-94ce-deb8e10a66ff')"
ChannelID: "ac12504d-f0a3-40bc-bc30-0aca60dcc8e2"
CreatedDate: "2015-08-29T14:54:34Z"
Description: ""
DisplayFormUrl: "https:// tenant.sharepoint.com/portals/community/pVid/Forms/DispForm.aspx?ID=5"
FileName: "Celebrating Heroic Women.mp4"
OwnerName: "Anne Wallace,#i:0#.f|membership|annew@ tenant.onmicrosoft.com,#AnneW
@ tenant.onmicrosoft.com,#AnneW@ tenant.onmicrosoft.com,#Anne Wallace,#https:// tenant
-my.sharepoint.com:443/User%20Photos/Profile%20Pictures/AnneW_ tenant_onmicrosoft
_com_MThumb.jpg,#Executive Management,#President"
ServerRelativeUrl: "/portals/community/pVid/Celebrating Heroic Women.mp4"
ThumbnailUrl: "https:// tenant.sharepoint.com/portals/community/pVid/Celebrating
Heroic Women.mp4.PNG?VideoPreview=1"
Title: "Celebrating Heroic Women"
ID: "d48666c4-09e1-49af-94ce-deb8e10a66ff"
Url: "https:// tenant.sharepoint.com/portals/community/pVid/Celebrating Heroic Women.mp4"
VideoDurationInSeconds: 5
VideoProcessingStatus: 2
ViewCount: -1
YammerObjectUrl: "https:// tenant.sharepoint.com/portals/hub/_layouts/15/videoplayer
.aspx?v=https%3A%2F% tenant%2Esharepoint%2Ecom%2Fportals%2Fcommunity%2F
pVid%2FCelebrating%20Heroic%20Women%2Emp4"
}
}
```

Comme nous le voyons, nous récupérons de nombreuses informations sur la vidéo (titre, auteur, date de publication, URL du fichier source, statut d'encodage, etc.).

Récupération de l'URL du flux streaming

Maintenant que nous avons récupéré les données d'une vidéo, nous allons vouloir jouer celle-ci au sein d'un lecteur vidéo (qu'il s'agisse d'une application mobile native ou bien d'un site Web en HTML5). La solution qui viendrait tout de suite à l'esprit serait d'utiliser l'URL de la vidéo telle que renvoyée dans le flux JSON. Cela fonctionnerait sans soucis mais poserait quelques problèmes car il s'agit de l'URL du fichier source, non encodé et non optimisé pour être joué en streaming. L'utiliser reviendrait à télécharger une vidéo de plusieurs dizaines ou centaines de mégaoctets avant de pouvoir commencer la lecture. Microsoft a pensé à cela, et expose une API permettant de récupérer l'URL de la version en streaming de la vidéo qui a été stockée dans le portail. Pour cela il faut repartir de l'URL REST de la vidéo (contenue dans le champ odata.id du flux JSON précédent), et lui ajouter un paramètre comme vous pouvez le voir ci-dessous :

https://tenant.sharepoint.com/portals/hub/_api/VideoService/Channels(guid'ac12504d-f0a3-40bc-bc30-0aca60dcc8e2')/Videos(guid'd48666c4-09e1-49af-94ce-deb8e10a66ff')/GetPlaybackUrl(1)

L'utilisation de la méthode GetPlaybackUrl en lui passant le paramètre 1 permet de récupérer l'URL de streaming de la vidéo au format HTTP Live Streaming (HLS) qui est compatible avec la plupart des lecteurs vidéos mobiles et des lecteurs vidéos HTML5 (Safari, Chrome, Firefox...).

Télécharger / supprimer des vidéos d'un canal

Lire une vidéo est très utile mais si l'on souhaite mettre à disposition des utilisateurs une application la plus complète possible, il serait logique de pouvoir ajouter et supprimer des vidéos d'un canal. Là aussi, les APIs mises à disposition par Microsoft vous permettent de réaliser ces opérations en repartant de l'URL du canal en question. Ainsi si vous souhaitez télécharger (uploader) une vidéo dans le canal, il suffit de soumettre une requête POST à une URL sous la forme suivante :

https://tenant.sharepoint.com/portals/hub/_api/VideoService/Channels(guid'ac12504d-f0a3-40bc-bc30-0aca60dcc8e2')/Videos(guid'd48666c4-09e1-49af-94ce-deb8e10a66ff')GetFile()/SaveBinaryStream

Cette méthode est très utile pour les petites vidéos (quelques mégaoctets) mais échouera en cas de vidéos imposantes. Dans ce cas, vous pouvez télécharger vos vidéos par petits morceaux en utilisant 2 autres URLs :

https://tenant.sharepoint.com/portals/hub/_api/VideoService/Channels(guid'ac12504d-f0a3-40bc-bc30-0aca60dcc8e2')/Videos(guid'd48666c4-09e1-49af-94ce-deb8e10a66ff')/GetFile()/StartUpload(...)

https://tenant.sharepoint.com/portals/hub/_api/VideoService/Channels(guid'ac12504d-f0a3-40bc-bc30-0aca60dcc8e2')/Videos(guid'd48666c4-09e1-49af-94ce-deb8e10a66ff')/GetFile()/ContinueUpload(...)

Si vous souhaitez à l'inverse supprimer une vidéo présente au sein d'un canal, il suffit de soumettre une requête POST sur l'URL de la vidéo en précisant dans les en-têtes HTTP de votre requête la valeur « X-HTTP-Method: DELETE ». Bien d'autres possibilités sont offertes par les APIs Office 365 Vidéo, comme la possibilité de mettre à jour les métadonnées (titre, description, etc..) liées à une vidéo ; aussi, le plus simple pour voir l'intégralité des capacités offertes par la plateforme, est de vous référer à la documentation officielle disponible sur le site MSDN : https://msdn.microsoft.com/en-us/office/office365/api/video-rest-operations

Conclusion

Comme nous l'avons vu dans cet article au travers des deux services que sont Delve et Office 365 Viéeo, les capacités d'intégration de la plateforme collaborative de Microsoft au sein de vos applications, via les APIs REST et les SDKs mis à disposition par l'éditeur de Redmond, sont très importantes. Les services offerts sont en constante évolution pour proposer de nouvelles fonctionnalités, et de nouveaux services sont également en préparation. Un des derniers exemples en date est Sway qui permet de créer des présentations interactives de manière simple et efficace.

Si vous souhaitez en savoir plus sur ce que prépare Microsoft dans les mois qui viennent, autour de sa plateforme Office 365, vous pouvez vous référer au site présentant la roadmap de celui-ci :

http://success.office.com/en-us/roadmap

Une nouvelle API unifiée pour Office 365 : les enjeux et la mise en pratique de cette API

Il y a maintenant près d'un an, la première mouture des API Office 365 atteignait le stade de la disponibilité générale (General Availability). L'objectif de ces API était d'offrir un accès simplifié aux services Cloud et à leurs données (emails, contacts, calendriers, documents...) à la fois sans nécessiter de connaître les modèles de développement de chacun des produits sous-jacents (Exchange, SharePoint, AD...) en proposant cette surcouche, mais en plus de ne pas se restreindre aux développeurs de l'écosystème Microsoft par l'adoption de standards connus et reconnus tels que REST et OData pour le transport et l'interrogation des données, Open ID et OAuth pour les parties authentification et autorisation.

Gaëtan Bouveret
Consultant **Infinite Square**

INFINITE SQUARE

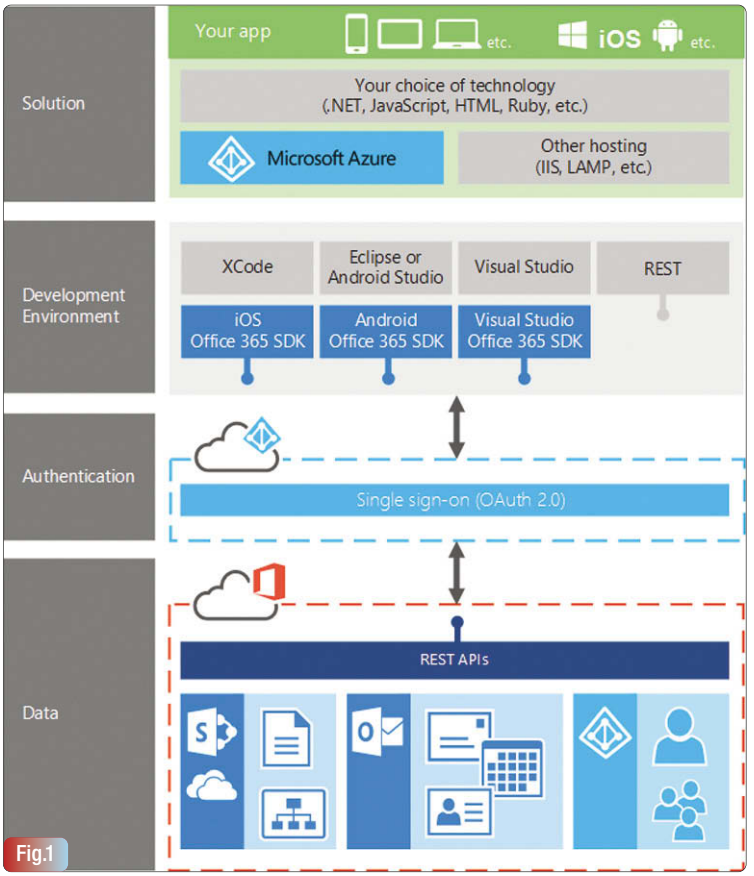


Fig.1

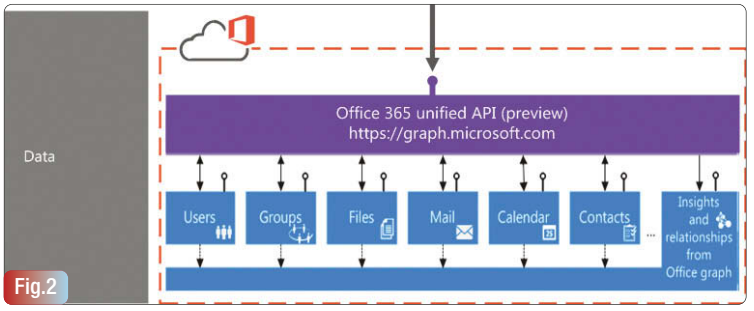


Fig.2

Cette interopérabilité permet son utilisation dans n'importe quel langage (.NET, Java, PHP, Ruby, Python...) et sur n'importe quel environnement (Azure, LAMP, iOS, Android...). Et Microsoft ne s'est pas arrêté à cela, et n'a pas cessé de fournir de nombreux exemples de projets et de code pour ces différentes plateformes, notamment sur le repository GitHub « OfficeDev » (http://github/OfficeDev), mais aussi des SDKs pour XCode (iOS) et Eclipse / Android Studio (Android) en plus de sa cible traditionnelle qu'est Visual Studio, SDK permettant de gagner du temps puisque proposant déjà la majorité des actions sans avoir à coder les appels REST. C'est donc résolument un message d'ouverture : qui aurait cru il y a quelques années qu'il serait possible d'intégrer aussi facilement des services Microsoft sous plateforme LAMP, iOS ou Android ? **Fig.1.**

Mais malgré cette première avancée, Microsoft ne comptait pas en rester là. Depuis quelques mois une nouvelle génération de cette API est proposée en preview, appelée « Office 365 Unified API », ou « Microsoft Graph », que je vais vous présenter ci-après.

Un Endpoint pour les gouverner tous

Une des évolutions tout d'abord, c'est que nous n'avons plus à nous préoccuper que d'un seul point d'accès pour tous nos appels : « https://graph.microsoft.com/{n°version}/... ». Et mine de rien, cela simplifie grandement les appels car auparavant, il nous fallait tout d'abord appeler le service « Discovery» pour connaître les différents Endpoints disponibles pour notre tenant, puis les appeler selon nos besoins. Nous avions par exemple « https://api.office.com/discovery/v1.0/ » pour la découverte des services, « https://graph.windows.net » pour la partie Azure AD, « https://outlook.office365.com » pour tout ce qui touche à Exchange et « https://[tenant]-my.sharepoint.com/ » pour OneDrive.

C'est désormais plus simple puisque tout se résume à spécifier le bon chemin pour, au choix, récupérer des utilisateurs (« https://graph.microsoft.com/beta/domain.onmicrosoft.com/users »), obtenir mes derniers emails (« https://graph.microsoft.com/beta/me/messages ») ou encore mes fichiers (« https://graph.microsoft.com/beta/me/files ») **Fig.2.**

Des entités et leurs relations

Mais ce n'est pas tout. En effet, comme vous l'avez peut-être deviné avec les exemples précédents, nous avons des entités reliées entre elles, ce qui permet de naviguer au sein de ces informations. On peut accéder ainsi aux emails d'une personne en rajoutant « /messages » à la suite d'une entité « utilisateur », « /files » pour parcourir ses fichiers OneDrive ou encore obtenir le détail de son manager avec « /manager ». Il est possible de la même manière d'effectuer des actions selon l'entité en ajoutant la commande à celle-ci : /send pour un message électronique par exemple. Cette façon de

parcourir nos entités devient plus intuitive et c'est ce qui représente le fameux Graph.

Je vous parlais d'entités, en voici un rapide tour non exhaustif avec quelques-unes des actions disponibles à ce jour :

- « Users » : ce sont toutes les informations propres aux utilisateurs : leur nom, fonction, photo, numéro de téléphone etc. Il est possible de modifier certaines propriétés, ou encore de désactiver ou réactiver un compte
- « Groups » : c'est une des nouveautés d'Office 365 apparues il y a quelques mois : ils regroupent à la fois les fonctions de listes de distribution et de groupes de sécurité, avec partage de conversations, de calendrier ou encore de fichiers. Vous pourrez les créer, éditer, supprimer.
- « Files » : correspond à tout ce qui touche aux fichiers situés dans OneDrive For Business. On peut parcourir ainsi les fichiers et répertoires d'un utilisateur, en récupérer le contenu, créer de nouveaux documents ou encore les modifier.
- « Mail » : ce sont les courriers électroniques bien sûr, que vous pourrez requêter, créer, envoyer, supprimer.

« Calendar » : calendriers et événements sont accessibles en lecture.

« Contacts » : ce sont les contacts (internes pour l'instant) que vous pourrez récupérer et gérer.

Concernant les relations entre les entités, sachez qu'elles seront à peu près partout. Un petit exemple pour illustrer ceci : vous venez de récupérer un des fichiers situés dans le répertoire OneDrive de l'utilisateur courant qui a été partagé avec un de ses collègues qui vient de le modifier, et vous voulez en savoir sur ce dernier. Voici ce que cela vous donne :

https://graph.microsoft.com/<version>/me/files/<idFichier>/lastModifiedBy/user/displayName

C'est une approche qui vous semblera rapidement beaucoup plus naturelle que celle où l'on récupère un identifiant à réutiliser ailleurs pour obtenir le détail auprès d'un service dédié.

Vos premiers pas avec cette API

Afin de se familiariser avec cette API, je vous invite à consulter (et utiliser !) les deux sites suivants :

- Graph Explorer (https://graphexplorer2.azurewebsites.net) tout d'abord vous permet d'effectuer vos requêtes REST et de consulter leurs réponses. Il suffit de se connecter avec un compte valide Office 365 et consentir les permissions pour interroger votre tenant
- API Sandbox (https://apisandbox.msdn.microsoft.com/) propose quant à lui des exemples d'appel et un mini-environnement de développement en ligne dans lequel vous pourrez essayer vos portions de code avant de les mettre en pratique dans vos projets.

Nous allons utiliser un peu le Graph Explorer. Après s'être connecté avec un compte entreprise et consentit les autorisations de lecture sur votre tenant, tapez l'adresse suivante :

(GET) https://graph.microsoft.com/beta/me/files (beta sera remplacé prochainement par la version release, sans doute « 1.0 » ou « 1 ») **Fig.3**.

On observe en retour tout un tas de propriétés définissant les dossiers et répertoires présents dans votre répertoire OneDrive For Business (bien sûr, pensez à le fournir un peu avant de faire vos tests !). L'entité de la capture est un dossier, il est donc possible de l'utiliser pour obtenir les fichiers qui y résident à l'aide de l'URL suivante :

« https://graph.microsoft.com/beta/me/files/<idRepertoire>/children ».

Et étant donné le support d'OData, vous pouvez aussi limiter les informations récupérées en ajoutant une clause « $select », par exemple seuls l'identifiant et le nom du fichier :

« https://graph.microsoft.com/beta/me/files/<idRepertoire>/children?$select=id,name ».

On retrouve d'autres paramètres tels que « $orderBy » pour trier, « $filter » pour filtrer « $skip » ou « $top » pour passer un certain nombre de résultats ou afficher les n premiers.

Bref, vous l'avez compris, c'est à la fois simple et puissant, et sans réinventer la roue de par la prise en compte de standards reconnus.

Et pour aller plus loin ?

Comme je vous l'ai évoqué, Microsoft a mis à disposition de nombreux exemples de code et tutoriaux. Je vous conseille de parcourir les labs sur l'API unifiée disponibles dans le repo GitHub OfficeDev (https://github.com/OfficeDev/Unified-API-Getting-Started-Labs). Vous y trouverez des exemples avec Angular, ASP.NET MVC, Javascript, Node.js et PHP, ainsi que prochainement sous Android, Python, iOS ou Windows 10.

Les packages NuGet sont déjà prêts (https://www.nuget.org/packages/Microsoft.Graph), cependant restez à l'affut des éventuelles prochaines mises à jour pour la General Availability.

Mais s'il n'y avait qu'un lien comme point de départ à vous donner, ce serait sans aucun doute le Dev Center Office http://dev.office.com, et plus spécifiquement la section dédiée à cette API http://dev.office.com/unifiedAPIs.

J'espère vous avoir donné envie d'intégrer Office 365 dans vos projets, et comme Satya Nadella l'a expliqué lors des événements Build et Ignite 2015 : « The most strategic developer surface area for us is Office 365 » ! Vous n'avez plus de raison de passer à côté de cette opportunité d'enrichir vos applications avec Office 365, pas même celle de ne pas travailler dans un environnement de développement Microsoft !

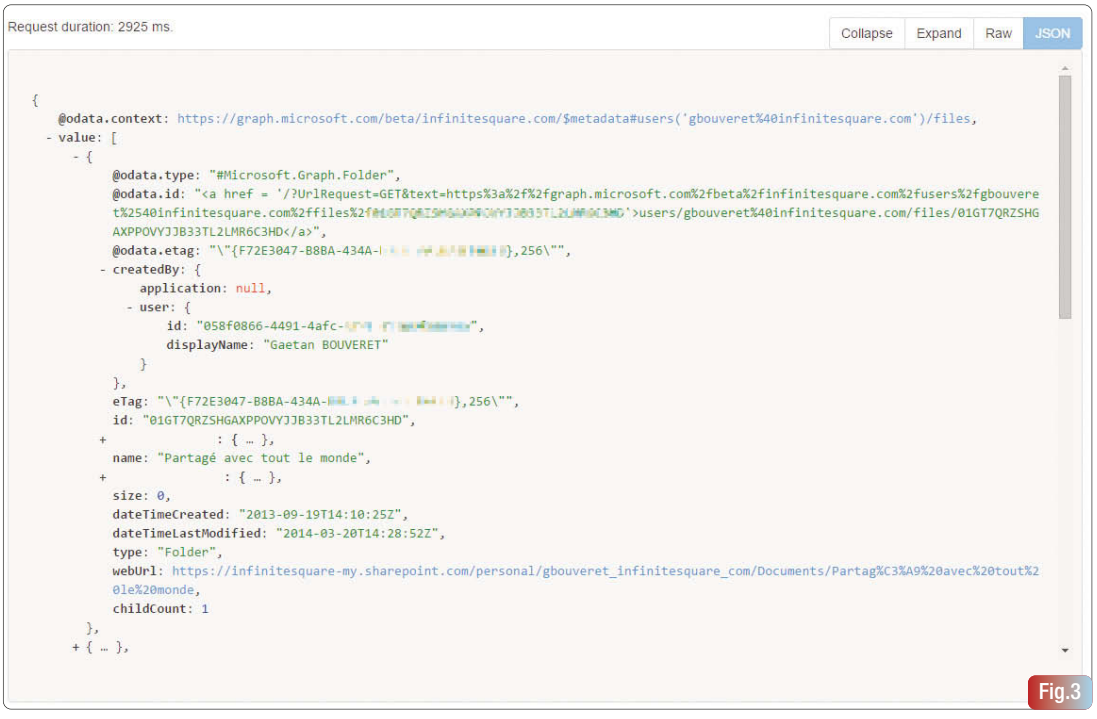


Fig.3

Les add-ins Office ou comment réaliser une application Office disponible en mode desktop, Web et tablettes !

Microsoft a introduit avec Office 2013 le concept d'Apps pour Office, renommé depuis peu en Add-ins Office. Ceux-ci permettent d'enrichir Word, Excel, PowerPoint ou encore Outlook à l'aide d'applications tierces directement intégrées au sein des produits et capables d'interagir avec leurs contenus. Mieux encore, l'objectif avoué est de pouvoir proposer la même application quel que soit l'environnement d'utilisation: votre add-in vous suivra en mode Desktop, Web ou sur vos devices mobiles.

Gaëtan Bouveret
Consultant **Infinite Square**

Un add-in Office est composé de 2 éléments :

- Un fichier XML (manifest) qui décrit votre application : quel est son nom, son type, son logo, les permissions nécessaires à son bon fonctionnement, l'URL de la page Web de démarrage etc. En bref, sa fiche d'identité.
- Une application Web qui contiendra la ou les pages Web qui seront appelées par votre add-in. Point important : il n'y a aucun prérequis technologique ici et vous pourrez tout autant utiliser ASP.NET MVC que Node.js, PHP ou de simples pages HTML Fig.1.

En effet, afin d'assurer une portabilité sur les différentes versions d'Office, votre add-in est au final un simple site Web qui est encapsulé au sein d'une version sandboxée d'Internet Explorer (pour isoler le code et éviter toute intrusion non désirée) en mode Desktop, ou tout simplement intégré sous forme d'iframe en mode Web. L'API qui vous servira à communiquer avec Office et le document est un Framework JavaScript : vous pouvez ainsi récupérer la sélection courante effectuée dans Word, modifier le sujet d'un email, détecter si vous êtes en mode présentation ou édition dans PowerPoint, etc. L'intérêt, outre bien évidemment la portabilité, est aussi la facilité de développement grâce à la réutilisation des compétences connues de tous les développeurs Web : HTML, JavaScript, CSS. Il existe 3 types d'add-ins :

- Volet de tâche (« Task Pane ») : un panneau en regard du document,
- Contenu (« Content ») : directement affiché au sein du document, comme une image,
- Messagerie (« Mail ») : spécifiquement pour emails et rendez-vous.

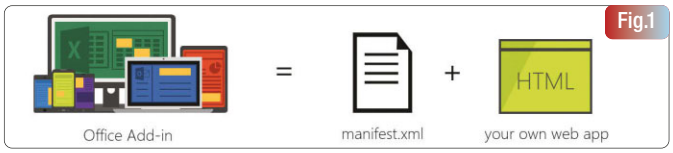


Fig.1

Prérequis

Avant de vous lancer dans le développement de votre add-in, il vous faudra auparavant télécharger la dernière version des « Office Developer Tools » (http://aka.ms/getlatestofficedevtools) pour Visual Studio qui vous permettront de débuguer facilement vos applications et de les publier. Vous pourrez ensuite mettre à jour les références à l'API JavaScript via le package NuGet « Office JavaScript API » (« Microsoft.Office.js »).

Développement de l'add-in

Pour notre premier développement, nous allons partir sur un add-in pour Excel, celui-ci étant disponible quelle que soit la plateforme (Desktop, Web, iOS). Nous y verrons comment lire / écrire le contenu d'une cellule, ainsi qu'une plage de données.

Initialisation du projet

Tout d'abord, commencez par lancer Visual Studio et créer une nouvelle « App pour Office » (le renommage en add-in/complément n'est pas encore répercuté partout) Fig.2.

Sélectionnez ensuite le type « Task Pane » Fig.3.

Et gardez seulement « Excel » coché Fig.4.

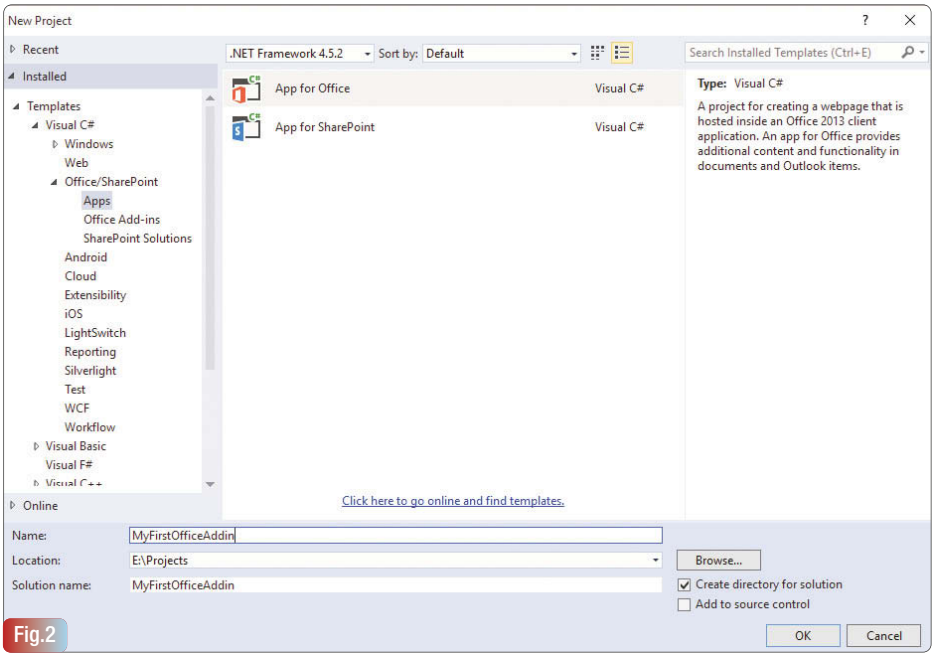


Fig.2

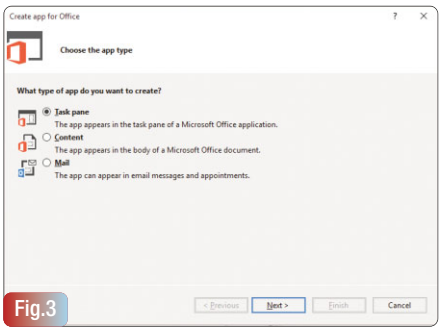


Fig.3

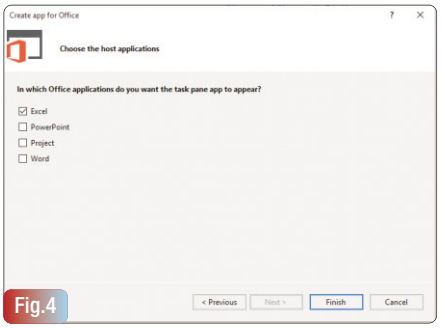


Fig.4

Il est évidemment possible de créer une application disponible pour plusieurs logiciels, mais il conviendra alors de gérer soi-même le support ou non de telle ou telle partie de l'API selon celui qui exécute actuellement l'add-in.

Anatomie de la solution Visual Studio

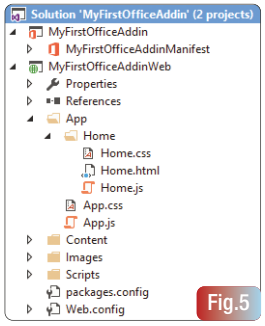


Fig.5

La solution est composée de deux projets : le premier ne sert qu'à contenir le Manifest, la définition de notre add-in, le deuxième est un projet Web contenant principalement une page HTML des feuilles de styles et des scripts Fig.5. En cliquant sur le Manifest, nous pouvons l'éditer au travers d'un assistant. Dans le premier onglet (« Général ») nous retrouvons les informations générales telles que le nom de l'add-in, sa version, son logo ou encore le niveau de permission requis : ce dernier définit le contrat que vous passez avec vos futurs utilisateurs en stipulant si vous désirez accéder au document en lecture ou écriture. Y est défini aussi l'emplacement source, c'est-à-dire l'URL de la page qui sera appelée lors du chargement du complément. Cette URL sera automatiquement mise à jour avec votre URL locale lors du debug avec IIS ou IISEXPRESS, ou avec l'URL de déploiement pour les environnements de recette ou de production Fig.6. Dans le deuxième onglet (« Activation ») se trouvent les différents prérequis : quels jeux d'API sont requis, quelles applications (Word, Excel, ... mais aussi version : SP1, Web App...) et le support d'Intellisense Fig.7. Le dernier onglet (« App Domains ») permet de spécifier les domaines d'éventuelles autres pages qui seraient ouvertes dans l'add-in. En effet, si le domaine n'est pas listé, la page sera forcément ouverte dans une nouvelle fenêtre de navigateur. Si vous ouvrez directement le manifest, vous obtiendrez sa description XML :

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Created:cb85b80c-f585-40ff-8bfc-12ff4d0e34a9-->
<OfficeApp xmlns="http://schemas.microsoft.com/office/appforoffice/1.1"
  <Id>0cd7fc23-1e64-4e0a-9a70-9a25a62461d5</Id>
  <Version>1.0.0.0</Version>
  <ProviderName>GBOUVERET</ProviderName>
  <DefaultLocale>en-US</DefaultLocale>
  <DisplayName DefaultValue="MyFirstOfficeAddin" />
  <Description DefaultValue="Ceci est mon premier addin pour Office">
    <Override Locale="fr-FR" Value="Mon texte localisé"/>
  </Description>
  <Hosts>
    <Host Name="Workbook" />
  </Hosts>
  <DefaultSettings>
    <SourceLocation DefaultValue="~remoteAppUrl/App/Home/Home.html" />
  </DefaultSettings>
  <Permissions>ReadWriteDocument</Permissions>
</OfficeApp>
```

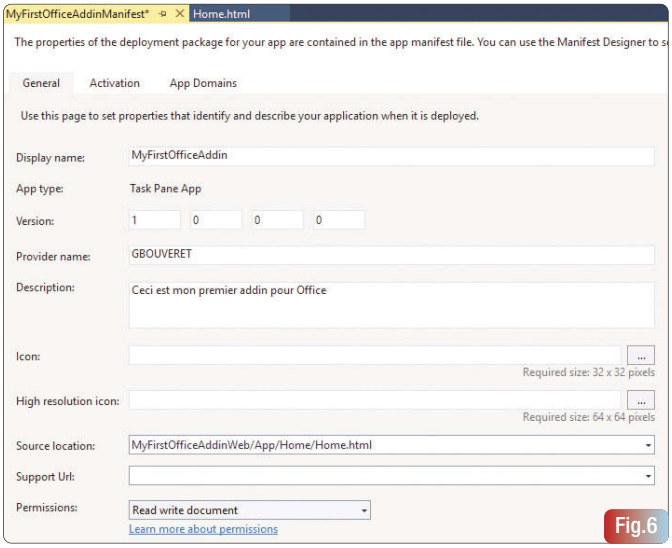


Fig.6

On y retrouve les différentes propriétés éditables dans l'assistant, et il est possible de surcharger le titre ou la description pour le support d'autres langues (nœud « Override »). Vous noterez aussi l'utilisation d'un jeton particulier représentant l'URL du site hébergeant l'application : « ~remoteAppUrl ». C'est elle qui est remplacée automatiquement par Visual Studio en debug ou une fois publiée.

Maintenant que le Manifest est prêt, nous allons pouvoir nous focaliser sur l'autre projet qui contient réellement la logique : le fichier « Home.html » sera notre page principale, et « Home.js » le fichier JavaScript avec notre code appelant l'API.

Deux points importants : tout d'abord, toutes les pages doivent référencer l'API Office via le CDN, ou en local pour le debug en mode offline.

```
<script src="https://appsforoffice.microsoft.com/lib/1/hosted/office.js" type="text/javascript"></script>

<!-- To enable offline debugging using a local reference to Office.js, use:                    -->
<!-- <script src="../../Scripts/Office/MicrosoftAjax.js" type="text/javascript"></script>  -->
<!-- <script src="../../Scripts/Office/1/office.js" type="text/javascript"></script>  -->
```

Ensuite, il faut impérativement implémenter la méthode « Office.initialize ». Celle-ci est appelée une fois que la page est correctement chargée dans l'add-in et signifie que l'API peut être utilisée.

```
// The initialize function must be run each time a new page is loaded
Office.initialize = function (reason) {
    $(document).ready(function () {
        app.initialize();

        $('#get-data-from-selection').click(getDataFromSelection);
    });
};
```

Passons aux choses sérieuses (ou pas) : place au code !

Commencez par modifier le contenu de notre page HTML comme ceci afin d'y ajouter les boutons qui appelleront nos fonctions :

```
<div id="content-main">
    <div class="padding">
        <p><strong>Mon premier addin Office avec Programmez</strong></p>
        <p>Actions simples:</p>
        <button id="get-data-from-selection">Lire</button>
        <button id="set-data-current-date">Ecrire la date du jour</button>
        <p>Sélection de valeurs:</p>
        <button id="set-data-power">Révisez les puissances de 2</button>
    </div>
</div>
```

Passez ensuite au fichier « Home.js », et ajoutez à la suite de la fonction « getDataFromSelection » le code suivant qui permet d'écrire la date du jour dans la cellule sélectionnée :

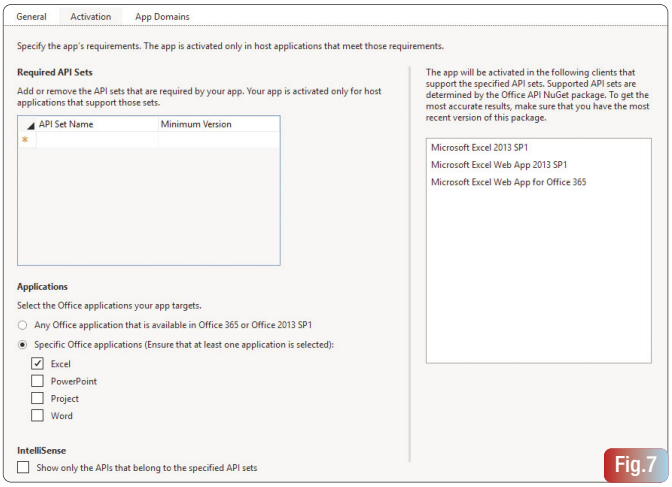


Fig.7

```
function setDataCurrentDate() {
    var curDate = (new Date()).toLocaleDateString();
    Office.context.document.setSelectedDataAsync(curDate, { coercionType: Office.CoercionType.Text },
        function (result) {
            if (result.status === Office.AsyncResultStatus.Succeeded) {
                app.showNotification('Date du jour ajoutée :', curDate);
            } else {
                app.showNotification('Erreur:', result.error.message);
            }
        });
}
```

Petite explication : nous commençons par récupérer la date du jour, puis nous appelons la méthode « setSelectedDataAsync » afin d'écrire dans la sélection courante dans le document. Nous lui passons la valeur à recopier (la data), des options facultatives (ici on spécifie le coercionType qui correspond au type de donnée, ici du texte brut, mais cela peut-être une matrice, tableau...) ainsi qu'un callback récupérant le résultat de l'opération que nous afficherons sous forme d'un petit panneau (app.showNotification est défini dans app.js). On peut ainsi s'assurer que notre action s'est bien déroulée ou non, et l'indiquer à l'utilisateur.

Ajoutez ensuite cette autre fonction qui va afficher les puissances de 2 sous forme de matrice dans une sélection sur 2 colonnes :

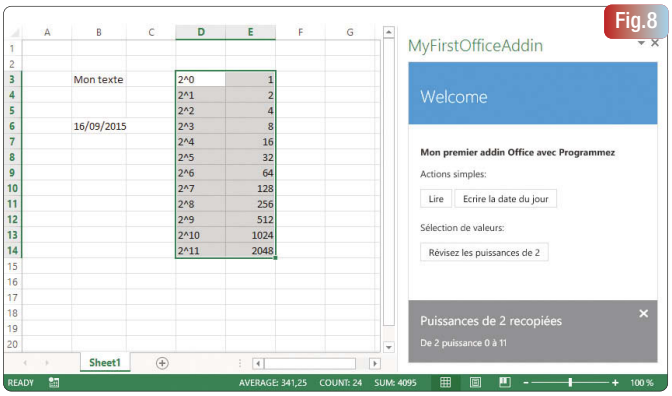
```
function setDataCalculate2Power() {
    Office.context.document.getSelectedDataAsync(Office.CoercionType.Matrix, function (getResult) {
        if (getResult.status == Office.AsyncResultStatus.Succeeded) {
            if (getResult.value.length > 0 && getResult.value[0].length == 2) {
                var p2 = [];
                for (var i = 0; i < getResult.value.length; i++) {
                    p2[i] = ["2^" + i, Math.pow(2, i)];
                }
                Office.context.document.setSelectedDataAsync(p2, { coercionType: Office.CoercionType.Matrix },
                    function (setResult) {
                        if (setResult.status === Office.AsyncResultStatus.Succeeded) {
                            app.showNotification('Puissances de 2 recopiées', 'De 2 puissance 0 à '
                                + (getResult.value.length - 1));
                        } else {
                            app.showNotification('Erreur:', setResult.error.message);
                        }
                    });
            }
            else {
                app.showNotification('Plage de valeur invalide', 'Sélectionnez 2 colonnes de cellules vides.');
            }
        }
    });
}
```

Cette fois-ci nous lisons d'abord la sélection via « getSelectedDataAsync » en spécifiant un type Matrix, puis nous testons si nous avons deux colonnes et au moins une ligne avant de générer son contenu sous la forme libellé / valeur. Nous réutilisons ensuite le « setSelectedDataAsync » pour écrire le résultat.

Enfin, modifiez le contenu de l' « Office.initialize » comme suit :

```
Office.initialize = function (reason) {
    $(document).ready(function () {
        app.initialize();

        $('#get-data-from-selection').click(getDataFromSelection);
        $('#set-data-current-date').click(setDataCurrentDate);
        $("#set-data-power").click(setDataCalculate2Power);

        Office.context.document.addHandlerAsync(
            Office.EventType.DocumentSelectionChanged,
            getDataFromSelection);
    });
};
```

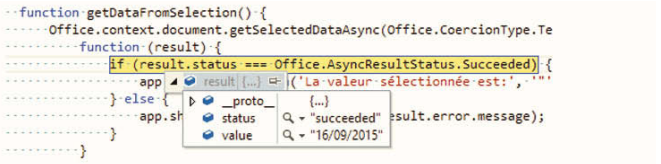
On y associe les clicks sur les boutons à nos fonctions, et nous ajoutons aussi l'événement « DocumentSelectionChanged » qui appellera la méthode affichant le contenu de la cellule sélectionnée. Il est ainsi possible de réagir automatiquement à un certain nombre d'événements qui seront levés par l'API Office lorsque l'utilisateur sélectionne du contenu, modifie des valeurs liées etc.



Exécution !

Hop, un petit F5 pour lancer l'exécution en mode debug et Excel se lance avec l'add-in chargé et ouvert. Vous pouvez tester en ajoutant la date du jour, en affichant automatiquement le contenu des cellules lorsque vous les sélectionnez (via l'événement), et afficher les puissances de 2, Fig.8.

Vous pouvez tout à fait poser quelques points d'arrêts dans vos fonctions JavaScript et observer en détail les valeurs des différentes variables et du contenu courant.

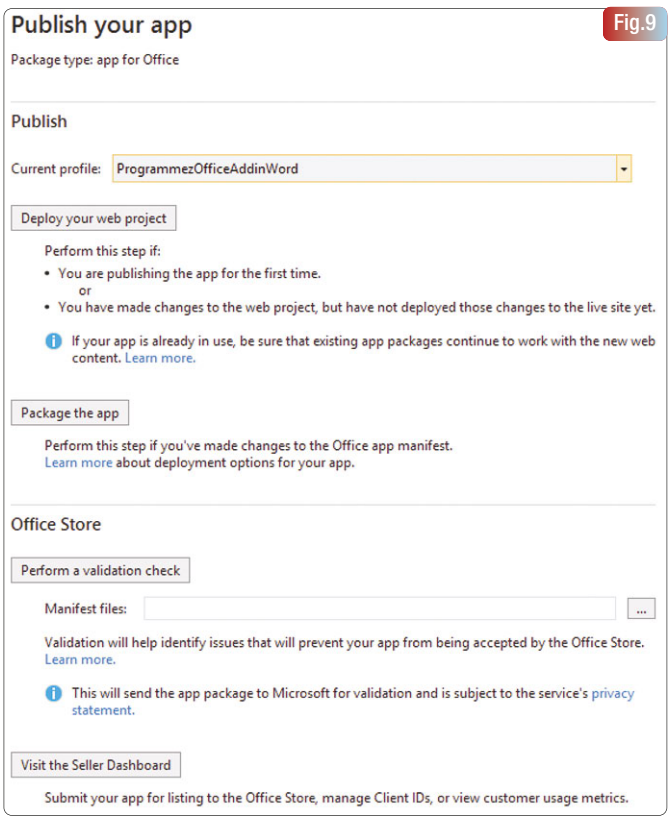


Et la publication ?

Quand vous franchirez la première phase de développement et que vous devrez mettre à disposition votre application, il vous suffira de passer par l'action « Publish » du projet contenant le manifest. Vous trouverez des assistants pour déployer votre projet Web (dans Azure ou ailleurs), packager votre application (au final remplacer le jeton d'URL par l'URL de destination), mais aussi tester et publier votre add-in sur le Store Office si vous le désirez ! Fig.9.

Pour aller plus loin

Vous avez désormais les bases pour développer vos add-ins Office. Pour aller plus loin, je vous conseille de parcourir le Dev Center Office http://dev.office.com qui vous guidera vers de nombreux exemples de codes. Vous y trouverez aussi les références à la documentation des API. Bon développement Office !



App Store : qui veut gagner des millions ?

Comment faisions-nous avant les App Store pour installer des logiciels ? Le plus souvent, on les achetait en boîte. Depuis le premier App Store sur iOS, le logiciel est désormais dématérialisé. En quelques secondes, on lance l'installation. Les App Store sont désormais partout : mobiles (App Store sur iOS, Google Play sur Android, Windows Store sur Windows 10), desktop et les App Store privés se multiplient dans les entreprises.
Les apps des stores pèsent très lourd dans l'économie informatique et le modèle économique. Et c'est aussi l'espoir de gagner de l'argent même si finalement, les heureux gagnants ne sont pas forcément très nombreux. Des éditeurs ont explosé grâce aux App Store et aux centaines de millions de terminaux. Mais avec une bonne idée, une bonne app, le développeur peut espérer se faire une petite place.
Mais l'App Store n'est pas un eldorado facile. Il y a beaucoup de monde et aussi beaucoup de mauvaises apps. Développer pour les App Store n'est pas très compliqué, mais vous devrez respecter certaines règles et parfois vous désespérer devant les lenteurs et les incohérences de certains stores, ou des règles qui changent, sans avoir été prévenus !
La sécurité est un autre point sensible à ne jamais négliger. La récente mésaventure d'Apple avec XCodeGhost illustre qu'aucun App Store n'est à l'abri. Il y a les failles systèmes, mais aussi la légèreté de certains développeurs / éditeurs qui ne respectent pas les bonnes pratiques. Apple a même été obligé de rappeler qu'il faut télécharger les outils de développement depuis le site officiel, car signés et validés...
Bienvenue dans le merveilleux monde des App Store !

La rédaction

Quelques rappels

	App Store	Google Play	Windows Store
Coût inscription store	99 $*	25 $	env. 14 € (individuel) env. 75 € (entreprise)**
Prix mini/maxi	oui	oui	
Apps gratuites/payantes	oui/oui	oui/oui	oui/oui
% revenant à la plateforme	30 %	30 %	30 %***
% du prix des apps versés aux développeurs	70 %	70 %	70 %***

Pour certains pays, il n'est pas possible de proposer des apps gratuites et payantes. Ainsi dans les pays listés par Google Play, pas d'apps payantes en Chine, Iran et au Soudan par exemple.

Attention à la TVA et aux règles fiscales des pays.

* Ce coût est celui de l'abonnement annuel à un des programmes développeurs Apple permettant l'accès aux bêtas, au support et aux outils d'Apple.

** Le compte entreprise donne accès à toutes les fonctionnalités.

*** Sur Windows Store, il y a bien le taux de 30 % de commission (sur les recettes nettes), mais il existe aussi un autre taux, appelé « ajustement du développement commercial », clause 6-b du contrat développeur : « La Redevance du Store que vous devez à Microsoft pour la mise à disposition de vos Applications et Produits intégrés à l'application par l'intermédiaire du Store sera déduite des recettes nettes pour calculer les Revenus de l'application en votre faveur conformément à cette Section 6. La Redevance du Store pour les Applications et les Produits intégrés à l'application mis à disposition dans le Store s'élève à trente pour cent (30 %) des Recettes nettes, sauf pour les transactions soumises à l'Ajustement du développement commercial, pour lesquelles la Redevance du Store est actuellement de quarante-trois virgule neuf pour cent (43,9 %) des Recettes nettes. »

Quelques succès, mais peu d'élus !

Un des gros succès depuis 18 mois est le superbe jeu Monument Valley qui a été téléchargé par des millions de joueurs, majoritairement sur iOS. L'éditeur a réalisé (janvier 2015) presque 6 millions $ (81,7 % sur iOS) grâce au jeu (payant) et à son extension (payante aussi). Mais attention, le coût de développement n'est pas négligeable : les estimations (techrepublic.com) se montent à 852 000 $ (1 an de développement !) pour le jeu, l'extension aurait coûté 549 000 $ et 6 mois de développement !

Ces chiffres sont assez impressionnants, on peut aussi se rappeler l'aventure incroyable d'Angry Birds ou Candie Crush, mais ces éditeurs ont du mal à trouver une autre « killer app ». Ces succès ne doivent pas faire illusion : le cimetière des apps est bien rempli. Entre les apps non maintenues et celles qui sont téléchargées à quelques reprises, la vie des développeurs et d'éditeurs d'apps mobiles n'est pas simple. Il faut la bonne idée, les développeurs/designers, le modèle économique qui va bien et bien entendu, trouver son public ! Une mise en avant de son app dans un Store est souvent un élément clé.

Que cette réalité ne vous empêche pas de vous lancer dans l'aventure. Mais faites-le en parfaite connaissance. Vous devez avoir une stratégie avant tout développement, car une bonne idée ne suffit pas. Et il n'est jamais inutile de regarder si votre idée n'existe pas déjà et si elle existe, ce que vous apportez comme nouveauté.

Un conseil parmi d'autres : regarder les domaines émergents comme l'écologie, l'éducation.

Des plateformes de développement diverses

	App Store	Google Play	Windows Store
Langage de référence	Objective-C Swift	Java	C#
Machine de développement	Mac + OS X	Mac/PC/ Chrome Book	de préférence PC sous Windows
IDE de référence	XCode	Android Studio	Visual Studio
Support des solutions de type Xamarin, Cordova...	oui	oui	oui

La guerre des chiffres et poids économiques

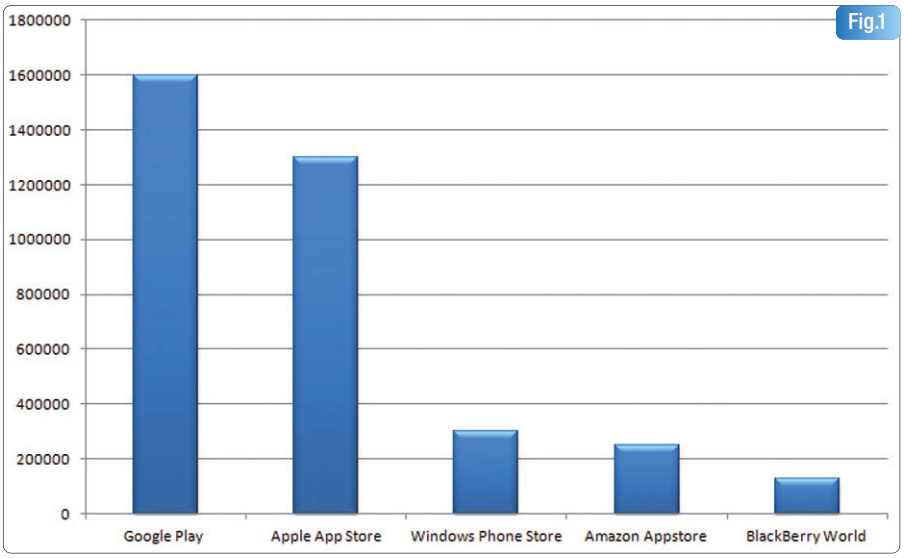
Chaque store fait une guerre des chiffres : Google a le plus d'apps, Apple va dire qu'il distribue le plus d'argent, Microsoft met en avant le store unique pour tous les terminaux... Google Play reste largement devant pour le nombre de téléchargements, mais toujours en retrait sur les revenus générés, iOS restant premier. En janvier 2015, Apple affirmait avoir versé + 10 milliards de $ aux développeurs. Google Play verserait + 7 milliards aux développeurs/éditeurs. Play continuera à croître avec une base installée bien plus grande. Il dépasse déjà assez largement iOS en nombre de téléchargement d'apps.

Google Play Store : visite guidée du store applicatif Android de Google

L'émergence des appareils mobiles a entraîné une véritable révolution au niveau applicatif avec une modification en profondeur du mode de consommation des applications. La mise en avant d'un unique store applicatif au sein des OS mobiles leaders comme Android aura eu pour effet de les ériger en point d'accès obligatoire pour rendre une application disponible au plus grand nombre. Dans cet article, nous vous proposons une visite guidée du store applicatif de Google du point de vue du développeur.

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

Très largement leader dans l'univers mobile, Android représente désormais une part de marché de plus de 70%. En imposant son propre store applicatif, le fameux Google Play Store, comme point d'entrée obligatoire pour le téléchargement d'applications sous Android, Google a eu le nez creux. Son Play Store dépasse ainsi largement le milliard d'utilisateurs actifs à mi-2015 pour un total de plus de 1,6 millions d'applications disponibles (figure 1). Plus de 180 millions de téléchargements ont ainsi été recensés sur le 2ème trimestre 2015. Aussi faramineux que cela puisse paraître, ces chiffres sont en croissance perpétuelle. Cette croissance continue devrait propulser les revenus globaux générés par le Play Store de 6 milliards de dollars pour 2015 à plus de 15 milliards de dollars à l'orée 2020 ! Fig.1.



Prédominance du Play Store sur le marché des Apps

Ces chiffres montrent clairement qu'il est aujourd'hui impossible de se passer du Google Play Store pour distribuer des applications en espérant toucher un maximum d'utilisateurs. Et ce, même si la nature de la relation entre Google et les développeurs Android penche clairement du côté du géant Américain. En effet, les alternatives représentées par le store d'Amazon ou le store indépendant SlideMe restent trop confidentielles par rapport au Play Store. Ce dernier est donc bel et bien indispensable pour les développeurs Android.

Inscription

La première étape pour pouvoir publier une application sur le Play Store consiste à s'inscrire auprès de Google en tant que développeur ce qui implique d'avoir un compte Google actif, de saisir son identité de développeur et d'accepter les règles de distribution édictées par Google. Il restera en outre à s'acquitter de frais d'inscription de 25 dollars permettant de publier autant d'applications que souhaité sur une durée de temps illimitée.

Pour aller plus loin que la publication d'applications gratuites et proposer aux utilisateurs des applications payantes ainsi que des achats in-app, il est nécessaire de créer un compte marchand Google qui est gratuit. Il faudra enfin renseigner une adresse postale dans son profil développeur pour permettre à d'éventuels acheteurs de vous contacter par cette voie s'ils le souhaitent.

Bonnes pratiques

En publiant une application sur le Google Play Store, les développeurs s'engagent à respecter les règles édictées par Google en matière de contenus. Ces règles sont accessibles à l'adresse suivante : https://play.google.com/about/developer-content-policy.html. Très précises, elles vont de l'interdiction de proposer des applications incitant à la haine ou présentant des scènes de violence gratuite, par exemple, à l'interdiction de violer la propriété intellectuelle d'autrui. La partie du règlement relative à l'usurpation d'identité ou aux comportements trompeurs est très importante et doit être lue attentivement. En outre, toutes les pratiques visant à augmenter illicitement le nombre de téléchargements ou les revues sont assimilées à du spam par Google et peuvent être punies sévèrement.

Enfin, depuis le milieu de l'année 2015, Google impose de compléter un formulaire pour chaque application afin de lui permettre d'avertir les utilisateurs sur le public visé par l'application. Attention car des contrôles sont réalisés et des erreurs dans ce formulaire peuvent entraîner des suppressions d'applications. Le dernier règlement concerne l'accord de distribution donné par le développeur à Google pour une distribution sur le Play Store et peut être consulté ici : https://play.google.com/about/developer-distribution-agreement.html .

Mauvaises expériences

A la lecture de ces règlements, il est aisé de comprendre que de nombreux développeurs

ont déjà eu affaire à des suppressions d'applications pour des raisons plus ou moins compréhensibles. Car le problème ne tient pas tant dans la suppression d'applications que dans l'impossibilité d'échanger avec Google afin de comprendre les raisons des suppressions et autres avertissements pour pouvoir corriger le tir. Pire encore, des comptes développeurs peuvent être supprimés si plusieurs infractions aux règlements sont constatées. Ces cas les plus extrêmes sont autant d'alertes rappelant aux développeurs la prudence dont il faut faire preuve vis-à-vis des règlements de Google. Pour bien comprendre le type de malentendu pouvant s'installer entre un développeur et Google, imaginons une application donnant les résultats de la Coupe du Monde de Rugby 2015. Le développeur pourrait la nommer Coupe du Monde de Rugby 2015. Eh bien, ce titre est en infraction avec le règlement sur le contenu puisqu'il pourrait faire croire aux utilisateurs qu'il s'agit d'une application officielle... La bonne pratique ici consiste à nommer l'application : App pour la Coupe du Monde de Rugby 2015. Compte tenu de la taille limitée des titres, cela devient rapidement compliqué d'autant plus que des vérifications systématiques sont désormais réalisées avant publication des applications. Prudence donc !

Modèles économiques

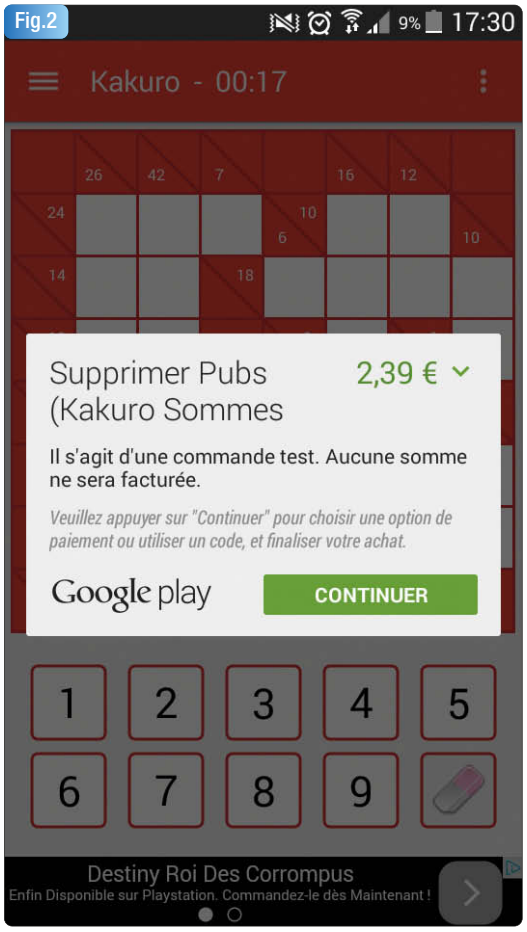
Fort heureusement, tout n'est pas noir avec le Google Play Store et il demeure possible pour des développeurs indépendants de tirer leur épingle du jeu en générant des revenus. Voici les différents modèles économiques existants :

- Premium : le développeur fixe un prix pour une application qui est facturée à l'utilisateur lors de l'achat. Ce modèle est en perte de vitesse et s'adresse à des applications de niches.
- Freemium : les utilisateurs sont désormais plus enclins à télécharger des applications gratuites. Partant de ce constat, il est avantageux de leur proposer des achats in-app pouvant être durables ou consommables. Un achat durable pourra concerner l'achat d'une fonctionnalité supplémentaire (figure 2) alors qu'un achat consommable pourra être un nombre de vies dans un jeu.
- Souscription : la souscription consiste à faire payer à intervalles réguliers les utilisateurs et s'adresse aux applications proposant du contenu fréquemment mis à jour. Il est réservé à une catégorie bien particulière d'applications comme les magazines.
- Publicité : la dernière solution revient à proposer de la publicité aux utilisateurs en étant rémunéré au nombre d'affichages et de clics générés. C'est le modèle le plus répandu. Les publicités pouvant être de type bannières, interstitielles ou vidéos. AdMob, la solution publicité de Google, est la référence sur le marché puisqu'elle est déployée sur plus de 650 000 applications !

Ces modèles peuvent être combinés et c'est justement le combo freemium et publicité qui représente la solution la plus rentable à l'heure actuelle puisqu'elle génère près de 90% des revenus sur le Google Play Store ! Fig.2.

Gigantesque, le gâteau à se partager ne cesse de croître et Google ne laisse bien évidemment pas sa part du lion en appliquant des frais de transactions de 30% sur tous les achats réalisés sur le Play Store. Au niveau du partage entre AdMob et les développeurs, peu de chiffres sont à disposition, malheureusement.

Fig.2



Achat in-app durable pour supprimer la Publicité

Sécurité

Avec son Play Store, Google ne se contente pas de prendre sa part du gâteau. Il fournit aux développeurs Android une plateforme de distribution clé en main facilitant la distribution d'applications au plus grand nombre. Il garantit aux utilisateurs une sécurité maximale lors du téléchargement d'applications ou de jeux depuis le Play Store. Les différentes affaires liées à des malwares sous Android ou plus récemment sous iOS rendent cette sécurisation toujours plus importante tout en démontrant, s'il le fallait, qu'aucun système n'est infaillible. Pour garantir des APKs sains, Google a développé un service spécifique, répondant au nom de Bouncer, chargé d'analyser les applications publiées sur son store en quête d'éventuels logiciels malicieux.

Ce service permet à Google de rendre la publication d'applications sur son store quasiment instantanée en évitant un long processus d'approbation. Google Bouncer peut également être utilisé lors de l'installation d'APKs provenant de sources inconnues. En sus, Google a développé un ensemble de programmes scannant automatiquement les nouvelles applications en cours de publication afin d'alerter des validateurs humains d'applications potentiellement contraires aux règles du Play Store.

Conclusion

Porte d'entrée obligatoire pour le téléchargement d'applications depuis les appareils Android, le Google Play Store est le passage obligatoire pour tout développeur souhaitant distribuer ses applications au plus grand nombre dans l'espoir d'en tirer un quelconque revenu. Comme tout géant, il répond à des règles bien précises qu'il faut suivre à la lettre sous peine de se voir exclu du jeu.

Au niveau de la monétisation, il convient de suivre les bonnes pratiques de Google en adoptant le modèle freemium avec de la publicité pour maximiser ses chances de générer des revenus.

Cette visite guidée du Play Store terminée, il ne reste plus qu'à vous souhaiter bonne chance pour vos applications ! ⊞

GUYZMO

App Store : témoignage

Stéphane Sibué, directeur technique chez GUYZMO, nous évoque les différents stores et particulièrement l'App Store d'Apple. Il possède une solide expérience avec 16 apps sur App Store, 14 sur Google Play et 9 sur Windows Store.

Quelles différences vois-tu entre les 3 stores ?

Vue du coté utilisateur

Les 3 stores se ressemblent beaucoup et offrent en gros les mêmes possibilités de recherche. Les fiches des applications sont bien finies, avec beaucoup d'images et pas mal de textes d'explication.

Vue du coté développeur

Les stores sont un peu différents et ne sont pas tous au même niveau. Le store Apple est le plus ancien et fonctionne très bien. Il a évolué pour prendre en compte les différents types de supports (iPhone, iPad, Apple Watch) et aussi toutes les versions de tailles d'écrans. Au début il n'y avait qu'un seul type d'appareil (iPhone) et une seule taille d'écran possible, ce qui facilitait bien la tâche, tant au niveau développement qu'au niveau publication sur le store.

Le store Android me semble le plus abouti. Il est simple d'utilisation et permet très simplement de publier une nouvelle application. En cas de problème (omission d'informations importantes la plupart du temps), il nous guide efficacement pour résoudre le problème. Il gère assez bien le fait qu'il existe plusieurs types de supports (smartphone, tablette, tv, montre), et une quantité incroyable de tailles d'écrans différentes.

Le store Windows vient récemment d'être unifié entre le store Windows (PC) et le store Windows Phone. Depuis cette unification il faut bien admettre que son fonctionnement est parfois un peu erratique. A part ça il permet lui aussi de prendre en compte les différentes plateformes (PC, tablettes, smartphone) et les différentes tailles d'écrans dans un confort très correct.

Sur la partie App Store, Comment se déroule une soumission d'une app et le processus à suivre durant la soumission... et après ?

Etape 1 : Créer la fiche de l'application

Cette étape se passe sur le site Internet de iTunes Connect et permet de créer une nouvelle fiche pour une nouvelle application ou une fiche modifiée pour la mise à jour d'une application déjà publiée.

Il faut pour chaque langue supportée saisir les textes de présentation et d'explications, préparer des copies d'écrans (pour chaque langue, chaque support et chaque taille d'écran). C'est une opération qui peut prendre pas mal de temps de préparation.

Etape 2 : Uploader le binaire de l'application

Depuis les outils de développement, on doit ensuite uploader le binaire de l'application que l'on souhaite publier. Cette opération n'est possible que si vous avez au préalable préparé vos certificats (il y en a plusieurs suivant les tâches que vous souhaitez activer au sein de votre application). Dès cette étape, une série de vérifications automatiques est effectuée. Il se peut que le binaire soit refusé. Il faut corriger et de nouveau uploader.

Etape 3 : Activer le binaire uploadé pour lancer la soumission

Retour sur le site iTunes Connect, dans la fiche de l'application. On doit alors sélectionner le binaire que l'on vient d'uploader et activer la soumission qui peut, suivant les cas, prendre entre 3 et 10 jours tout de même !

Etape 4 : Prier pour que l'application soit acceptée

Etape 5 : Application acceptée ou non ?

Application acceptée :

C'est fini, jusqu'à la prochaine version.

Application refusée :

Accéder au compte rendu de soumission qui explique le motif du refus, qui donne quelques conseils sur comment arranger les choses, et un ou des liens vers la documentation et les exemples liés. Ce compte rendu s'est énormément amélioré avec le temps. Il permet généralement de corriger sans trop galérer le problème.

Il faut alors reprendre à l'étape 2 après avoir bien sûr corrigé le problème.

On parle souvent des règles parfois obscures pour qu'une app soit acceptée ou refusée, notamment en respectant les guidelines officielles, qu'en penses-tu ?

Il y a des règles qui sont clairement définies car très « lourdes », comme par exemple, l'interdiction de placer dans l'application un lien ou un système qui permet de payer sans passer par le système d'achat InApp d'Apple (qui vous prend au passage 30% du montant).

Il y a d'autres règles qui ne sont parfois pas évidentes à connaître car il faudrait se palucher une documentation très conséquente, qui change de version en version d'iOS, et qui n'est pas toujours très claire (c'est valable pour les 3 plateformes). J'ai eu une application iOS refusée car ma manière d'implémenter un cas de figure lié justement aux achats InApp ne leur convenait pas. J'avais eu le toupet d'automatiser une opération qu'Apple voulait exclusivement manuelle (comme pas mal d'actions sous iOS, ce qui souvent n'est pas un mal). Donc dans ce cas, application refusée, explications du support sur la raison avec un lien direct sur la partie de la fameuse documentation qui en parle. Pour le coup, ça a été assez simple de rectifier le tir.

Quels conseils donnerais-tu à des dévs pour soumettre sur App Store ?

Il faut impérativement lors d'une soumission garder une version des sources correspondante à cette soumission précisément, car parfois, en cas de problème, on doit revenir sur ces sources-là, et si entretemps vous les avez modifiés c'est vite la galère. Autre conseil, prévoyez du temps pour la préparation de la soumission, car la rédaction des textes (les taper, pas les inventer, ça c'est encore une autre histoire), la préparation de toutes les ressources graphiques (les écrans dans toutes les tailles, toutes les langues, tous les supports) vont vous prendre beaucoup de temps, et généralement cette étape est sous-estimée, d'où une mauvaise surprise (si vous ne pouvez facturer ce temps à votre client car non prévu ou sous-estimé).

On évoque aussi souvent les contraintes de localisation des apps (langues), les vidéos, les screenshot ; la partie App Store est-elle plus longue et contraignante qu'une autre ?

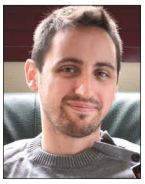
Non, en gros c'est le même délire ! Mais pour tout dire je ne vois pas trop comment il pourrait en être autrement. Il faut fournir ces ressources et on n'a pas le choix, à moins de dire que la fiche sera exclusivement en anglais et que les utilisateurs se débrouilleront bien pour comprendre, mais là ce n'est franchement pas une bonne idée.

Globalement quel est ton ressenti sur App Store ?

Franchement, à mon niveau, je n'ai pas grand-chose à lui reprocher. Peut-être que parfois il refuse de valider une fiche application et ne donne pas assez d'informations pour corriger le problème (contrairement au store Android que je trouve très bien fait pour ça).

Le nouveau Windows Store

Le Windows Store a fait peau neuve ; suite à l'arrivée de Windows 10, Microsoft a eu la délicate idée d'unifier les deux anciens Stores (Windows Store et Windows Phone) dans un seul et unique endroit, pour le plus grand bonheur des développeurs !



Stéphane Graziano
@poppyto
Webrox SARL
2,5M de downloads sur le Windows Store

Un compte unique

Un seul et unique compte Microsoft peut se connecter au Store, il faudra donc partager vos identifiants (gloups) avec toute l'équipe, et donc toutes les informations potentiellement confidentielles qui y sont entreposées.

Publication

Il faut réserver un ou plusieurs noms pour votre application (internationalisation) avant de pouvoir créer une fiche, et respecter le fait que ce nom ne peut être déjà pris : exit les doublons sur le Store. Ensuite vous devez envoyer vos packages : il faudra au préalable associer le package au nom d'application réservé dans le Store via Visual Studio. Une fois vos packages envoyés, les fiches de descriptions internationalisées correspondant aux langues prises en charge par les packages seront créées. Il n'est pas possible de rajouter manuellement une fiche dans le store (exemple : votre app est uniquement en anglais mais vous voulez traduire les fiches de Store : pas possible).

La bonne nouvelle avec le nouveau Store, c'est que les fiches de description sont désormais unifiées. La précédente mouture nous obligeait à refaire les fiches par langue, et même par plateforme. Ainsi pour une application avec 20 langues sur 5 plateformes (WP7/WP8/WP8.1/W8/W8.1), il fallait remplir 100 fiches, un calvaire! L'équipe de Microsoft n'a probablement jamais vu les dashboards sur les plateformes concurrentes, ainsi le concept de "par défaut" n'existe pas ; il faudra pour chaque fiche (chaque langue donc) renvoyer le texte, les adresses de contacts, les images promotionnelle internationalisées (sous-titres des images traduits également au risque de vous faire refuser l'app) et là ça commence à chiffrer : (cas typique : 5 captures d'écrans + 3 images promotionnelles * 20 langages = 160 fichiers à uploader 1 par 1 car pas d'envoi multiple - pourtant présent sur l'ancienne version du store ! Attention à la taille des images qui ne doivent pas dépasser 2 Mo (Astuce : renommez vos JPG en PNG si vos photos sont trop lourdes).

Passons à la grosse zone d'ombre du Store, à savoir la réglementation concernant les jeux-vidéos, vous devrez vous débrouiller pour fournir les certificats d'évaluation (PEGI, ESRB, etc.) au risque de ne pas être distribué dans les pays correspondants. La procédure est pénible car il faut les récolter un peu partout sur le Web, certains certificats étant vraiment difficiles à obtenir : les validateurs sont intransigeants avec ça, méfiez-vous ! Une fois vos achats InApp rajoutés (ou éventuellement vos bandeaux de pub rajoutés) votre app sera prête à être publiée : vous pourrez d'ailleurs préciser si elle sera visible ou non dans le Store lors de la publication, et donner une date éventuelle de lancement.

Publicité

PubCenter, le média de Publicité made by Microsoft a aussi été remanié mais il souffre toujours du même problème ; les taux de remplissages sont très bas (très peu d'annonceurs) et les eCPM qui en découlent sont par conséquent catastrophiques, difficile de monétiser son app avec de la publicité. Les utilisateurs sont habitués aux app gratuites sur les autres plateformes. La pub ne rapportant pas grand-chose sur celle de Microsoft, il faudra plutôt passer par la case achats intégrés si vous souhaitez en vivre ! Aussi étonnant vu le contexte, le backoffice ne permet pas de faire des campagnes avec des gros budgets !

Validation

C'est le moment tant redouté, à raison. Vous devrez patienter de 1 heure à 5 jours si vous n'avez vraiment pas de chance ; les validations les plus longues sont généralement pour les toutes nouvelles apps. De ma propre expérience, la validation dure plus longtemps quand elle échoue (sic!). En effet, vous recevrez un joli message non personnalisé avec un numéro de règle, c'est souvent général et frustrant (exemple : "Un des certificats d'évaluation n'est pas bon" - on ne vous dira pas lequel...). Le côté "presse bouton" manque également un peu de finesse, mais vous pourrez néanmoins rajouter un message dédié aux validateurs lors de vos soumissions. Toutefois, il vous sera impossible d'échanger avec eux par mail, même en cas de grosse tuile. N'y pensez même pas. Il semblerait que le process soit de plus en plus automatisé et que les véritables validations soient faites plus tard : ne soyez pas étonné de vous voir retirer une app plusieurs jours après sa validation pour une raison obscure, certains validateurs peuvent être très tatillons (un conseil, renvoyez tel quel, vous aurez peu de chance de retomber sur le même...)

Dashboard

Le dashboard vous permet de voir vos téléchargements journaliers et achats (non cumulés malheureusement), vous trouverez aussi un rapport de crashs et d'exceptions qui fait souvent peur ; de nombreux crashs indépendants de votre bon développement apparaissent, une bonne partie serait bien mieux chez les équipes de Microsoft... en farfouillant un peu et en examinant les stack-traces vous pourrez trouver vos bugs. Vous retrouverez aussi les évaluations auxquelles vous pourrez répondre, et vous pourrez télécharger tous les rapports sur une plage de 3 mois maximum.

Paiements

Un site dédié a été attribué aux paiements. Pas de surprise, Microsoft prend 30% du montant HT, et vous rend le reste HT. Suite aux nouvelles lois européennes, vous n'avez plus la pénible tâche de collecter la TVA (Mauvais plan pour les auto-entrepreneurs !) !

En vendant une app 1€ (TTC) en Europe, 17 centimes seront collectés par MS pour la TVA, 25 centimes pour la redevance Microsoft, et vous

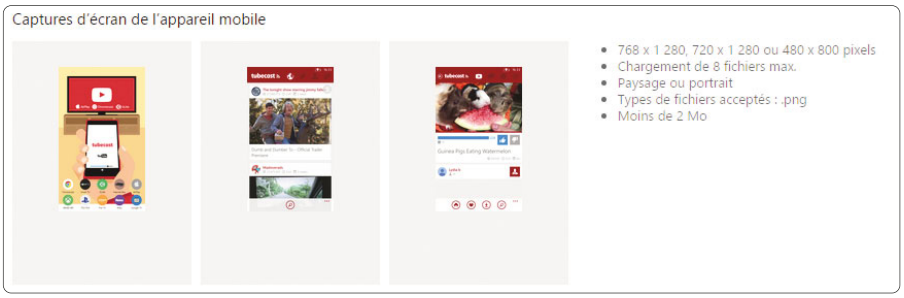
toucherez net 58 centimes. Depuis l'unification des Stores, plus besoin de récupérer les fichiers XLSX pour calculer ces différentes parties, un gros soulagement dans votre comptabilité ! On aurait quand même apprécié des graphiques sur les sommes globales, ce défaut étant déjà présent dans la partie statistiques du Store, la fonction SUM n'est visiblement pas du goût de l'équipe de dev ! Vous recevrez l'argent autour du 15 du mois.

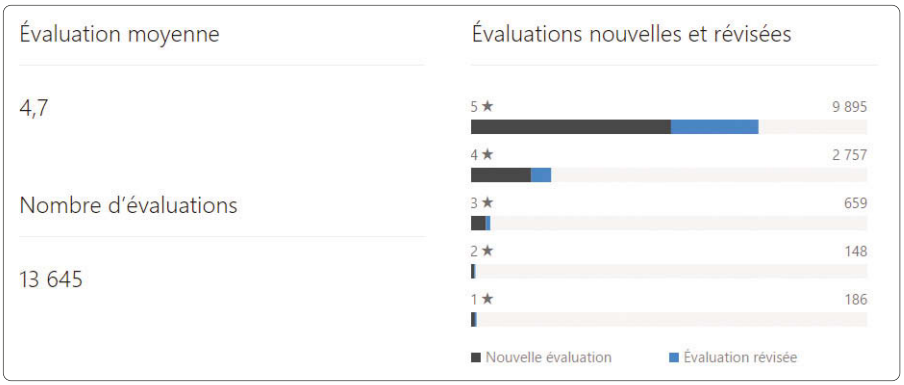
Support

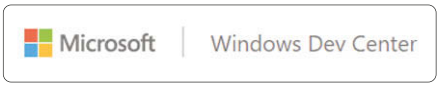
Microsoft fournit un support de niveau 1, uniquement en anglais. En gros, une personne avec les mêmes droits que vous sur vos fiches... en cas de problèmes plus technique, votre ticket sera transféré aux équipes qui vont bien, et s'ils ont envie d'y répondre peut-être que votre problème sera réglé... La patience est une vertu, comptez plus en semaines qu'en jours ! Un conseil, passez plutôt par le twitter @LumiaHelp : rapide et efficace !

Client Store

Pour l'utilisateur final, le nouveau Store a lui aussi un arrière-goût d'inachevé, vous ne l'aurez pas noté quand je parlais des fiches, mais vous n'aurez pas la possibilité de mettre des vidéos sur vos fiches... Le moteur de recherche (Bing is inside) a une pertinence discutable sur les mots-clés concurrentiels, et de nombreuses applications plus maintenues sont encore en haut des classements... Si vous faites partie des nombreux utilisateurs qui réinstallent de temps en temps leurs apps, ne soyez pas surpris de devoir repayer les apps déjà achetées (soucis de synchronisations) ou, au pire des cas, vos téléchargements resteront bloqués avec des messages d'erreurs explicites "0x80240017","0x80073CF9"...



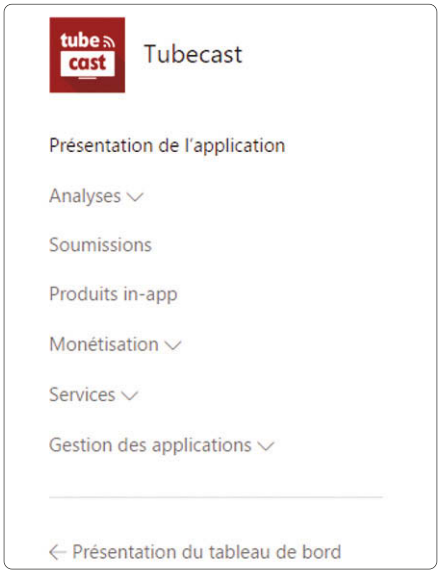


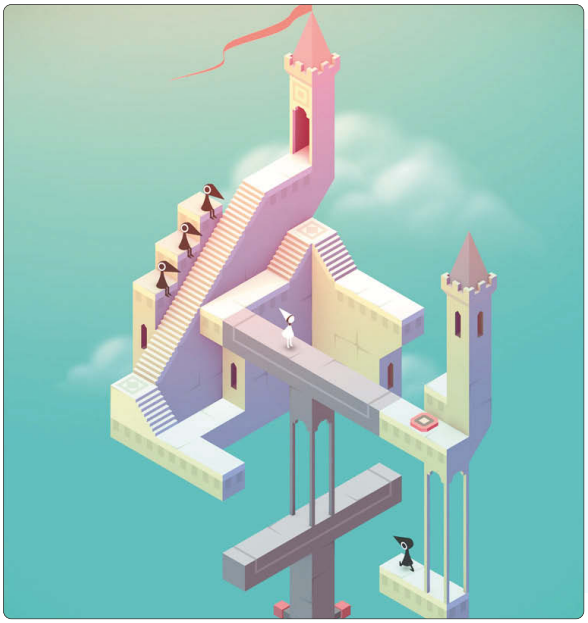


Impression générale

Difficile de pardonner les erreurs de jeunesse en 2015, l'équipe du Store n'a pas la fibre du Web applicatif, et ne vous aidera visiblement pas à gagner du temps en remplissant vos fiches (surtout si vous internationalisez), ni à gagner de l'argent avec PubCenter.

La bonne nouvelle, c'est que les utilisateurs ne vivent pas sur les Stores, et que passé cette étape, vous aurez encore tout le temps de faire le support de vos propres applications car, en plus de les acheter, vos utilisateurs, en contrepartie, cherchent souvent du réconfort !





Les Store « Desktop »

Si Windows Store unifie désormais les stores mobiles et desktop, ce n'est pas le cas sur les autres plateformes. Sur Mac, nous disposons du Mac App Store et sur ChromeBook du Chrome Web Store (applications pour Chrome).

Ces stores sont moins connus que les stores pour mobiles. Régulièrement, le Mac App Store connaît des problèmes et les développeurs / éditeurs s'en plaignent. Mi-octobre, l'éditeur de jeux Feral a retiré 22 jeux suite à des problèmes de téléchargement ou d'installation depuis le Mac App Store.
Si la logique ressemble à celle de l'App Store, le succès est plus mitigé.

Gérer le retour de navigation sous Windows 10 : une galère… mais réalisable

Franck N'Guessan
CeriBoo - AutoEntreprise de développement d'applications Windows
ceriboowp@yahoo.com

OS: UWP (Universal Windows Program) bref Windows 10 !

Comme certaines personnes, j'ai attendu la sortie de Windows 10. Par curiosité j'ai souhaité vérifier le comportement de certaines de mes applications en mode PC (Laptop) et tablette. J'ai dans le même temps découvert que le passage du mode tablette au mode Laptop pouvait se faire, même durant le lancement d'une application. Une chose est sûre: mes applications n'aiment pas le bouton retour du mode tablette. En effet lorsque l'on appuie actuellement sur mes applications sur le bouton retour du mode tablette, l'application disparaît au lieu de naviguer vers la page précédente. Aussi, en mode Laptop, si l'on n'affiche pas de bouton retour afin de naviguer vers une page antérieure, il est impossible de faire marche arrière. Nonobstant ces constats, il était pour moi urgent de comprendre, étudier, et trouver une solution un moyen de gérer cette nouveauté..... D'autant plus que le mode Smartphone (Windows 10 Mobile) s'ajoute à la danse.

Quel est le sujet du jour ?

L'objectif de ce tutoriel est de montrer comment gérer le bouton retour (ou d'en ajouter un) suivant le mode du contenant Windows 10 (PC/Laptop, tablette et smartphone). Les prérequis afin de réaliser cette manipulation sont :

- Visual Studio 2015 ;
- Un PC sous Windows 10 (pour pouvoir passer du mode tablette au mode Laptop et donc valider ce qui suit) ;
- Un émulateur W10M.

Le programme

Windows 10 nous permet de réaliser un seul code pour tous les supports (PC tablette smartphones iOS, etc.) Le problème (enfin pas vraiment) est de savoir sur quel support notre code "tourne" . Pour cela Microsoft nous a mis à disposition plusieurs outils afin de le détecter.

Le fichier App.xaml.cs

Pour savoir si notre programme est en train de fonctionner sur smartphone, j'ai mis ceci dans le fichier app.xaml.cs :

```
public static bool isHardwareButtonsAPIPresent =
  Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.UI.Input.
HardwareButtons");
public static UIViewSettings viewSettings = UIViewSettings.GetForCurrentView();
```

Bon je vous l'accorde certains WP comme le 630 n'ont pas de bouton "physique", mais il faut le considérer en tant que tel.
Le booléen isHardwareButtonsAPIPresent nous permet de savoir si oui ou non nous avons des boutons physiques. Viewsettings nous permettra plus tard de savoir si on est en mode tablette ou Laptop.

Le fichier MainPage.xaml.cs

On va gérer dans ce fichier le bouton retour afin de ne pas ressaisir un peu partout ce bout de code. Le voici :

```
  if (App.isHardwareButtonsAPIPresent)
      {
        Windows.Phone.UI.Input.HardwareButtons.BackPressed += HardwareButtons
_BackPressed;
      }
      else
      {

       Windows.UI.Core.SystemNavigationManager.GetForCurrentView().BackRequested
 += MainPage_BackRequested;
      }
private void MainPage_BackRequested(object sender, Windows.UI.Core.BackRequested
EventArgs e)
    {
      if (this.Frame.CanGoBack)
      { this.Frame.GoBack(); }
      else
      {
        Application.Current.Exit();
      }
      e.Handled = true;
    }
    void HardwareButtons_BackPressed(object sender, Windows.Phone.UI.Input.Back
PressedEventArgs e)
    {
      if (this.Frame.CanGoBack)
      { this.Frame.GoBack(); }
      else
      {
        Application.Current.Exit();
      }
      e.Handled = true;
    }
```

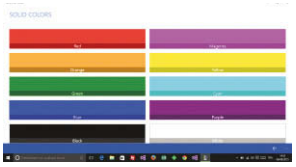
En résumé si on est sur Windows 10 mobile, on active la gestion du bouton physique du smartphone. Dans le cas contraire, on "active" la gestion du bouton retour en mode tablette. Sachant que c'est la première fenêtre de l'application, dans notre cas de figure si vous copiez/collez ce code, l'application se fermera normalement en mode tablette et smartphone via leur bouton retour respectif.

Le fichier [un fichier de navigation].xaml.cs

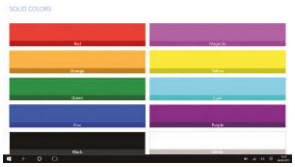
Lorsque nous naviguons de la page principale vers une autre, il est peut être utile pour l'utilisateur de pouvoir faire marche arrière. Pour cela en mode tablette il y a un bouton retour, sur smartphone aussi, mais sur laptop...rien. Personnellement j'ajoute (créé dans le cas de ce tutoriel) dans la barre d'application un bouton retour dans le fichier xaml qui à l'évènement clic contient Frame.GoBack() (code qui permettra de naviguer vers la page précédente).
Aussi, pendant que votre application est ouverte, on peut supposer que l'utilisateur passe du mode tablette au mode Laptop et vice versa sans forcément crier gare. Raison pour laquelle scruter l'événement layoutupdated d'une page permet de détecter tout changement de mode.
Voici le code mis en place:

```
public mapage()
    {
        this.InitializeComponent();
        this.LayoutUpdated += mapage_LayoutUpdated;
    }

    private void mapage_LayoutUpdated(object sender, object e)
    {
        if (App.isHardwareButtonsAPIPresent)
        {


            //suppression du bouton retour



        }
        else
        {

            if (App.viewSettings.UserInteractionMode == Windows.UI.ViewManagement.
UserInteractionMode.Touch)
            { //suppression du bouton retour }
            else
            { //maintien du bouton retour }
        }
    }
```

En résumé si on est en mode Windows 10 mobile ou en mode tablette, on supprime le bouton retour. Sinon on maintient le bouton retour (Laptop).
En voici la preuve en image sur l'application My Screen Tester



Mode Laptop



Mode Tablette

Conclusion

Voici le premier tutoriel pour Windows 10. Je m'étais engagé à ne pas trop "bidouiller" avant octobre et Windows 10 mobile, mais un désagrément de taille m'a contraint à faire le contraire.
Et c'est à ce moment-là que je me suis demandé comment gérer cette nouveauté. Il ne m'a pas fallu longtemps pour y parvenir, mais il est clair que cette petite manipulation met en lumière une évidence : si vous avez des applications Windows 8.x, il est urgent de les porter vers Windows 10 afin d'implémenter cette fonctionnalité.
Sinon vous risquez d'affronter la grogne de l'utilisateur lambda qui ne comprendra pas pourquoi le bouton retour du mode tablette ne fonctionne pas avec votre application.
De plus le portage de certaines applications s'est réalisé en douceur. On peut donc conclure que jusqu'à présent, à part le bouton retour, aucun pépin ou désagrément n'est à déplorer de ce nouvel OS.

chronique

Docker : le point de vue des développeurs

Depuis plusieurs mois, Docker fait régulièrement la une de l'actualité. L'outil est apparu il y a environ deux ans et s'est rapidement propagé auprès des développeurs, pour s'imposer aujourd'hui comme un produit phare de la mouvance DevOps.

Pierre-Alexande Bardina
et Yann Cézard
experts Docker chez **Objectif Libre**
www.objectif-libre.com
@objectiflibre

Docker : de quoi parle-t-on ?

Commençons par rappeler en quelques mots ce qu'est Docker : Docker est un outil Open Source, développé en Go par l'entreprise Docker Inc., qui facilite la gestion de conteneurs sous GNU/Linux. Pour que l'explication soit complète, précisons également la notion de conteneur : un conteneur est une méthode de virtualisation légère, qui isole les applications et les rend indépendantes les unes des autres sur les plans des binaires, des bibliothèques et des environnements. Pourquoi légère ? Contrairement à une solution de virtualisation « classique » à base d'hyperviseurs, les conteneurs sont de simples processus exécutés sur leur hôte, hôte dont ils partagent le noyau. L'idée principale est d'empaqueter une application et ses dépendances dans un conteneur virtuel qui pourra être exécuté sur n'importe quel serveur GNU/Linux – et prochainement sur Windows (qui disposera de ses propres images).

Docker : un outil pensé par et pour des développeurs.

Les avantages des conteneurs logiciels pour faciliter le travail des développeurs sont nombreux et ont permis une adoption fulgurante de la technologie par cette communauté.

Comment la portabilité et l'outillage améliorent la productivité des développeurs ?

La portabilité des conteneurs est le principal attrait de la technologie : il est possible d'avoir le même environnement en local et en production. Finies les erreurs de déploiement dues à une mauvaise version de bibliothèque ou à une autre dépendance : tester ses applications sur différents OS devient un jeu d'enfant. Côté outillage, Docker peut être considéré comme le GIT des conteneurs. Il propose un outil de gestion des images, versionnées, sur lequelles les actions proposées reprendront la même terminologie que celle de GIT : une image se récupère en faisant un « pull », un conteneur se transforme en image en effectuant un « commit », l'image est étiquetée avec un « tag », et sera publiée à l'aide d'un « push ».
Dans la pratique, grâce à la gestion des versions d'images de Docker, il est facile de déployer une nouvelle image en production, et il est tout aussi facile de revenir, si nécessaire, sur une image antérieure. Il est également possible d'utiliser les images officielles présentes sur le Hub de Docker telles que proposées, et d'ajouter ainsi des services à son application. Ces images sont préconfigurées de manière à permettre leur utilisation immédiate, tout en autorisant un paramétrage plus précis si besoin : l'ajout d'une base PostgreSQL ou d'un serveur Redis à sa

stack se fait ainsi en seulement quelques minutes.

Chaque développeur peut ainsi très rapidement mettre en place un environnement de développement identique à celui de son équipe : grâce aux conteneurs, les développeurs deviennent beaucoup plus productifs.

Docker et l'intégration continue

Bien qu'antérieure à la technologie des conteneurs, l'intégration continue est indissociable de Docker, et doit être prise en compte pour bien en comprendre l'intérêt. La mise en place de nouveaux environnements devient si évidente avec Docker que les applications d'intégration continue l'embarquent désormais dans leurs processus, à l'instar de Jenkins ou de la nouvelle application Drone entièrement construite autour de Docker.

Jenkins utilisera Docker pour lancer ses slaves, avec des images types pour effectuer les builds ou les tests. Et Docker pourra à son tour utiliser Jenkins afin de créer automatiquement des images qu'il mettra à disposition dans un registre (un dépôt d'images Docker) en cas de succès.

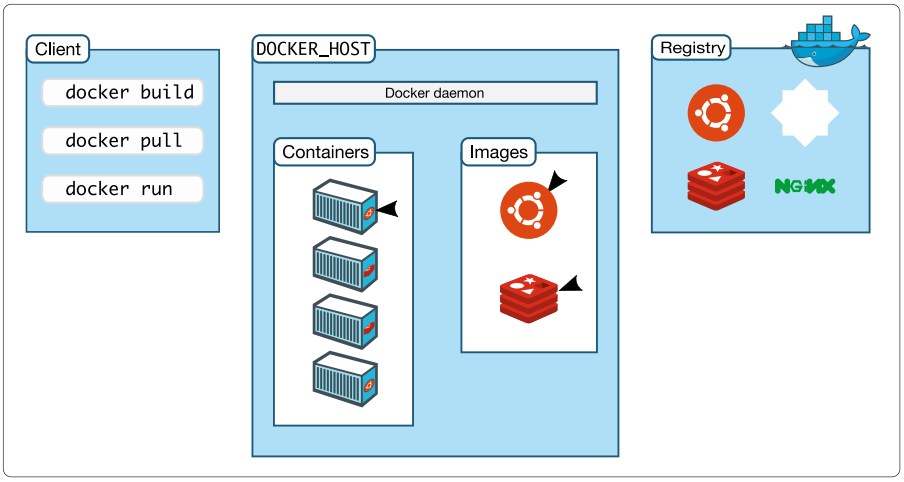
L'approche micro-services : une nouvelle manière de développer

L'émergence de Docker va modifier les habitudes de développement car l'outil incite à la construction d'applications moins monolithiques. De plus en plus, les architectures sont construites à partir de micro-services : les applications, plus complexes, sont composées de services plus petits répondant chacun à un besoin précis et qui communiquent entre eux via leurs APIs. L'intérêt de cette architecture est de pouvoir piloter, superviser et faire évoluer chaque micro-service indépendamment du reste de l'application. Si un des composants de l'application nécessite un peu plus de capacité de traitement, il suffit d'augmenter le nombre d'instances de ce composant.

L'utilisation de Docker dans un tel contexte pourrait se schématiser par des constructions de type « Lego », les briques étant des micros services (hébergés chacun dans son propre conteneur) pour lesquels les ressources allouées vont être gérées suivant les besoins.

Ce point est particulièrement intéressant pour créer des applications « élastiques », à capacité variable et adaptative. Sur une architecture virtuelle traditionnelle, la création de l'élasticité via l'isolation des services dans des machines virtuelles serait bien sûr possible, mais consommerait plus de ressources et serait donc plus coûteuse.

Ainsi, Docker apporte des changements dans les méthodes de développement, en privilégiant les micro-services et l'intégration continue ; et les applications gagnent en qualité. Des équipes réduites peuvent, à moindre effort, maintenir des architectures de très grande complexité, proches de celles de géants comme Twitter.



Création des images dans Docker.

Docker : une (r)évolution du métier de développeur

Si les méthodes de développement des applications changent, les méthodes de mise en production évoluent aussi et modifient fortement les métiers du développement et de l'administration système.

Encore récemment, le développeur avait pour tâche essentielle de coder l'application, ou une partie de celle-ci, et avait peu de liberté d'intervention sur les applications en production ; les serveurs de production restant la chasse gardée des administrateurs systèmes. Les développeurs devaient se lancer dans de longues négociations s'ils voulaient « toucher à la prod' »... sans garantie de succès.

Docker facilite le travail du développeur pour concevoir une architecture ou mettre en œuvre un environnement d'exécution complet. La palette des outils à la disposition des développeurs s'élargit et ouvre la porte à une nouvelle répartition des responsabilités au sein des équipes. Les administrateurs systèmes vont être amenés à se spécialiser dans le déploiement, la supervision et le maintien en conditions opérationnelles de l'architecture, et en particulier des composants hébergeant les conteneurs.

Est-ce la fin des accrochages entre développeurs et administrateurs systèmes pour des problèmes de versions de Python différentes en développement et en production ? Docker offre la possibilité d'une nouvelle forme de collaboration et d'un rapprochement des deux métiers... si toutefois les entreprises arrivent à accompagner ces changements organisationnels importants.

Perspectives : adoption massive, nouveaux acteurs et standardisation

Docker, malgré son jeune âge, est un outil déjà largement adopté par les développeurs et par les grands acteurs de l'hébergement en ligne : de nouveaux services spécifiques ont été créés chez Google Cloud Engine, AWS ou encore sur OpenStack pour l'hébergement des conteneurs. Côté clients, de belles « user stories » avec de grands logos comme ebay, Uber, Spotify, BBC News montrent que l'outil est suffisamment mature pour une industrialisation des déploiements. Son adoption plus massive par les entreprises est sans doute une simple question de temps : le temps que les métiers et méthodes de travail s'adaptent, et que les solutions clé-en-main d'industrialisation gagnent la maturité, à l'instar des projets Swarm ou Kubernetes qui visent à orchestrer des clusters de machines exécutant des conteneurs.

Mais l'outil n'est pas seulement intéressant que pour les grandes entreprises. Les micro-entreprises et les développeurs freelance qui l'utilisent peuvent libérer le temps auparavant alloué à la configuration de serveurs ou à la mise à jour du code en production. Ce temps peut maintenant être intelligemment réinvesti dans de la gestion du projet, davantage de développement, ou de la veille technologique. S'intéresser à Docker est incontournable. Des concurrents sérieux de Docker émergent, comme Rkt de CoreOs, confirmant que la technologie des conteneurs est bien en plein essor et pleine de perspectives. Afin d'éviter l'écueil des exclusivités technologiques, le projet « Open Container Initiative » a vu le jour et compte standardiser le domaine. Les conteneurs de Docker deviendraient compatibles avec ceux de la concurrence, les outils pour les exploiter seraient eux-aussi modulaires et interchangeables. La révolution des conteneurs est en marche !

Les tests Android démystifiés

1ère partie

Le développement d'applications Android a vu son écosystème évoluer et gagner en qualité. Le nombre de bibliothèques facilitant et accélérant les temps de développement a explosé grâce notamment à la communauté open source. Les différents outils à disposition des développeurs ont aussi grandement évolué, à l'image d'Android Studio, l'IDE officiel développé par Google.

Configuration

- Android Studio v1.0 ou plus
- Android SDK v22 ou plus
- Android SDK Tools v22.0.0 ou plus
- Android Support Repository v15 ou plus
- Plugin gradle: v1.1.0 ou plus

Florent Noël
Genymobile – Ingénieur Expert Android
@heb_dtc

Cette professionnalisation du développement sur Android s'explique surtout par le fait que la plateforme et la communauté ont toutes deux gagné en maturité. De nouveaux challenges émergent auxquels nous, développeurs, nous retrouvons confrontés au quotidien.

Parmi ces nouvelles préoccupations, la nécessité de tester correctement et efficacement les applications Android semble être au cœur de toutes les discussions. De nombreux frameworks et bibliothèques ont fait leur apparition afin de palier le manque d'outils officiels et de contourner les difficultés inhérentes au système Android lui-même. Cet article explique comment mettre en place des tests dans une application Android en utilisant les outils officiels fournis par Google. Cependant, afin de pouvoir être testée, une application doit avoir été pensée, architecturée et écrite pour effectuer les tests. Cet article aborde donc également les différentes stratégies à adopter afin d'écrire des applications modulaires et testables. Avant d'aborder la technique pure, il est essentiel de comprendre la nécessité d'écrire des tests, de comprendre ce qu'ils apportent concrètement, et quelles sont leurs conséquences sur les cycles de développement.

De l'importance des tests

"Écrire des tests prend du temps" ou encore "écrire des tests est difficile", sont les excuses le plus souvent invoquées par les équipes de développement. Si la deuxième affirmation peut se défendre, la première est clairement biaisée. S'il est vrai qu'écrire des tests demande du temps, il est aussi vrai que sur la durée ce sont ces mêmes tests qui vont permettre d'en gagner. Tester son code s'inscrit dans une démarche de qualité : qualité du produit fini duquel de nombreux bugs ont pu être éliminés et l'expérience utilisateur validée, mais aussi qualité des cycles de développement. Il est en effet beaucoup plus agréable pour une équipe de travailler sur une base de code saine et en laquelle elle peut avoir confiance.

Entreprendre une refonte complète d'une fonctionnalité ou venir greffer un nouveau module devient beaucoup moins risqué et peut s'entreprendre plus sereinement. C'est là l'une des leçons les plus importantes : si l'on écrit des tests, c'est avant tout pour soi-même. C'est bien parce qu'ils participent à rendre le travail quotidien plus serein qu'écrire des tests apporte une réelle valeur ajoutée. Les tests sont une sorte de boussole sur laquelle l'équipe peut s'appuyer pour faire évoluer son produit dans la bonne direction. Les tests sont donc aussi un outil de mesure et peuvent renseigner sur la bonne santé d'un projet.

C'est aussi à travers les tests que l'équipe peut capturer les besoins du client et assurer à ce dernier que le code écrit accomplit ce pourquoi il a été développé. Écrire des tests permet aussi de limiter au maximum les effets de bord et d'identifier les bugs le plus tôt possible. Enfin, les tests ayant une influence sur la façon dont les applications sont architecturées, ils tendent à forcer les développeurs à écrire du code plus propre, plus modulaire et moins interconnecté.

Quand tester ?

Comme évoqué plus haut, les tests font partie intégrante du cycle de développement et doivent donc être écrits en même temps que les fonctionnalités. Une bonne pratique consiste à ne livrer une nouvelle fonctionnalité que si celle-ci est accompagnée de tests. La plus grosse erreur à faire serait de développer de nombreuses nouvelles fonctionnalités et ensuite d'écrire tous les tests. Ainsi, s'atteler à tester de nombreuses choses différentes d'un seul coup est fastidieux, décourageant et résulte souvent en l'abandon pur et simple de la tâche d'écriture des tests. Les tests influencent la façon de coder. Seul un code modulaire et pensé pour les tests permettra une vaste couverture du projet. Et il sera parfois nécessaire de réécrire certaines parties du code afin de découpler le plus possible les différents modules d'un programme et ainsi rendre le code facilement testable. Cela nécessite des efforts de la part des développeurs mais devient vite une habitude. Impossible de parler de tests sans évoquer le Test-Driven Development. Cette technique consiste à écrire les tests avant même d'écrire du code. Les spécifications sont capturées et directement traduites en tests et ensuite le code vient s'articuler autour.

Que tester ?

Afin d'être efficace, il est important d'identifier quels sont les composants à tester au sein d'une application. Au début, il est en effet très facile de partir dans toutes les directions et de se retrouver à tester des choses hors de notre responsabilité. Écrire des tests est déjà un exercice fastidieux en lui-même, autant cibler correctement ce qui nous intéresse. Dans le cas d'une application Android, trois grandes catégories se distinguent :

- L'interface et le parcours utilisateur,
- La logique métier,
- Le support des différents types de terminaux.

Encore une fois, lors de la mise en place des tests, il ne s'agit pas de se mettre à écrire tous les tests d'un coup. Par exemple, il faut commencer par identifier le parcours utilisateur le plus utilisé, ou le code métier le plus critique de l'application. Ensuite, petit à petit, la base de tests grossira jusqu'à couvrir idéalement l'intégralité de l'application. Commençons par identifier les différents types de tests applicables. On distingue principalement deux grandes familles :

- Les tests unitaires,
- Les tests d'intégration / les tests fonctionnels.

Tests unitaires

Les tests unitaires s'appliquent purement au code. Il s'agit de tester la logique métier d'une unité de code. Ce sont des tests indépendants les uns des autres qui viennent vérifier qu'un bout de code, qu'une méthode ou qu'un objet fait précisément ce qu'il est censé faire. Les tests unitaires sont idéalement les plus détachés possibles du framework Android, et sont coupés de toute dépendance vers l'extérieur (appels réseau, accès disque ou autre). Ce sont des tests qui peuvent être exécutés extrêmement rapidement et permettent de vérifier que l'ensemble des blocs logiques d'une application font chacun exactement ce pourquoi ils ont été écrits.

Test d'intégration

Les tests d'intégration ont eux vocation à venir tester un parcours utilisateur. Ils peuvent servir à vérifier que les différentes Activity s'enchaînent correctement, qu'un menu est bien présent, ou encore valider que les informations affichées à l'écran sont bien celles attendues. Noter qu'ici les interactions avec le framework Android sont beaucoup plus marquées. Ces tests doivent donc être exécutés sur des terminaux (physiques ou non) et prennent donc infiniment plus de temps à être exécutés.

De la difficulté des tests sous Android

Pendant longtemps, tester son application Android a relevé du parcours du combattant. Fort heureusement, de nombreuses solutions, officielles ou non, ont vu le jour ces derniers mois et rendent enfin la chose beaucoup plus abordable. La suite de cet article démontre comment mettre en place ces solutions, mais auparavant, il est intéressant de comprendre pourquoi tester une application Android peut être si difficile.

Terminaux physiques, tests unitaires, JVM et mocking

Le premier obstacle vient du fait qu'initialement, il était impossible d'exécuter les tests unitaires sur une JVM. Il fallait absolument faire tourner les tests sur une machine virtuelle Dalvik et donc avec un vrai terminal. Cela implique de devoir compiler, indexer, packager et déployer un APK à chaque fois que quelqu'un souhaite exécuter la suite de tests. Cette approche est extrêmement lente et va totalement à l'encontre de la philosophie des tests unitaires, à savoir avoir un retour très rapide sur l'état de la base de code. Le problème vient du fait que le fichier android.jar utilisé lors de la compilation ne contient en fait que des classes « stubbées » et donc inutilisables. Un autre aspect compliqué des tests sous Android vient du fait que les classes du framework sont finales (au sens Java) et ne peuvent donc pas être « mockées » facilement. C'est ce constat qui a mené à la naissance du projet Robolectric qui offre une solution sérieuse à ce problème et propose des classes « Shadow » permettant de mocker le comportement des objets du framework.

Depuis la sortie du plugin Gradle 1.1.0, il est possible de générer un MockableJar pour exécuter les tests unitaires sur une JVM en local sur la station de travail du développeur. Cette approche est la même que celle répandue dans la communauté Java depuis des années via le framework JUnit.

Préparation du terrain

La première étape de la mise en place de tests unitaires dans une application consiste à mettre à jour le fichier build.gradle afin d'ajouter les dépendances nécessaires :

```
android {
        [...]
}

dependencies {
        [...]
        testCompile 'junit:junit:4.12'
        testCompile 'org.mockito:mockito-core:1.10.19'
}
```

JUnit et Mockito sont deux bibliothèques de l'univers Java très utilisées pour écrire des tests. Il est maintenant possible d'utiliser la version 4 de JUnit ce qui offre encore plus de flexibilité pour écrire des tests notamment grâce aux nouvelles annotations et API. Le code source de votre application se trouve généralement dans : app/src/main

De la même manière, les classes de tests vont venir se glisser dans un dossier test : app/src/test. Maintenant que le projet est correctement configuré, place à l'écriture des tests unitaires. Avec les tests unitaires, c'est bien le comportement d'un composant de l'application que l'on souhaite tester. Bien souvent, une classe de notre application aura son pendant dans le dossier test. Ainsi, imaginons que l'on souhaite tester la classe UserCookieHandler, chargée de garder en mémoire les cookies de l'utilisateur, il nous faudra alors créer une classe UserCookieHandlerTest.

```
@RunWith(MockitoJUnitRunner.class)
public class UserCookieHandlerTest {
   @Before
   public void init() {
   }
   @Test
   public void testThatStoredCookiesCanBeRetrieved() {
   }
}
```

Cette nouvelle classe se compose de plusieurs méthodes toutes annotées. Soulignons que les noms des méthodes ou de la classe importent peu, ce sont les annotations qui permettent au « test runner » de déterminer quoi et dans quel ordre exécuter les méthodes.

@RunWith

Permet de définir le « test runner ». Le « test runner » est celui qui va exécuter les tests. Ici, on précise que celui à utiliser est celui de « Mockito ».

@Before

Définit la méthode à exécuter avant chaque test. C'est dans cette méthode qu'il faut placer les différentes initialisations nécessaires au bon déroulement des tests.

@Test

Identifie un test

Il existe de nombreuses autres annotations en plus de ces trois-là, toutes répondant à des besoins particuliers. Voici quelques exemples des plus communément utilisées :

@Test(expected=IllegalArgumentException.class)

Permet de vérifier qu'une exception est bien lancée.

@After

Définit une méthode à lancer après chaque test, pour libérer des ressources par exemple.

@BeforeClass et @AfterClass

Définit les méthodes à exécuter avant et après tous les tests de la classe.

Structure d'un test

Un test unitaire peut se décomposer en trois parties. L'initialisation, l'appel à l'opération que l'on souhaite tester et l'assertion. Un test unitaire est binaire : il "passe" ou ne "passe" pas. Idéalement, un test unitaire n'a qu'une seule et unique assertion. Cette bonne pratique permet de garantir l'unicité du test et assure que si le test échoue, il est facile d'identifier pourquoi.

Quelques conseils pratiques :

- Ecrire des tests aux noms évocateurs. L'idée est de décrire l'intention. Pour faire un parallèle, les tests peuvent presque être considérés comme de la documentation du code. En lisant les tests, un nouveau développeur dans l'équipe doit pouvoir comprendre l'intention globale de la classe testée.
- Ecrire des tests succincts, ne tester qu'une fonctionnalité par test. Chaque test est indépendant des autres. Il faut pouvoir comprendre facilement pourquoi un test échoue.
- Faire autant d'efforts, si ce n'est plus, que lorsqu'on écrit le code de production. Les tests ne doivent pas être bâclés ou alors ils deviendront contre-productifs, ajoutant de la difficulté au lieu d'en retirer. Il ne faut pas oublier que les tests doivent aussi être maintenus. ⊞

La suite dans Programmez! 191

Vorlon.js : le debug Web simplifié

Etienne Margraff *(@meulta)*
Evangéliste Web et mobilité
Microsoft

Qu'est-ce que Vorlon ?

Imaginez ! Vous êtes dans votre canapé chez vous, vous êtes sur Facebook et un ami à vous poste un lien vers un article que vous voulez lire. Vous l'ouvrez et là, déception, le site ne fonctionne pas, ne s'affiche pas correctement et votre expérience de lecture est plus que mauvaise. Ça vous rappelle quelque chose ? Nous, ça nous arrive tout le temps. Et la raison est simple : **les développeurs n'ont pas les bons outils pour identifier et corriger les problèmes d'affichages liés au mobile**. Donc c'est complexe pour eux de vous fournir un site Web parfait !

Quand un développeur Web se retrouve face à un problème lié à l'affichage sur mobile, il n'a pas beaucoup de choix : il doit utiliser Chrome pour débugger le site sur Chrome sur mobile, il doit utiliser Safari pour faire de même sur Safari Mobile et Visual Studio pour Windows Mobile. Et très franchement, jongler avec tout ça n'est pas évident. Et sinon il y a Weinre (http://people.apache.org/~pmuellr/weinre-docs/latest/), un outil qui permet de se connecter à un site Web qui tourne sur un mobile via une interface Web. Malgré l'interface peu ergonomique et le fait que l'extensibilité n'est pas au top cela fonctionne plutôt bien. Par contre, cela nécessite d'utiliser Chrome (on ne peut pas choisir son navigateur préféré !).

David Catuhe, David Rousset, Pierre Lagarde et moi-même travaillons tous les 4 chez Microsoft. Lors de nos expériences de développement Web et de debug sur mobiles aucune de ces solutions ne nous satisfaisait vraiment. C'est pour cette raison que nous avons décidé de créer **Vorlon.js** (http://vorlonjs.io).

L'objectif de Vorlon.js est de proposer une expérience la plus proche possible de celle qu'on peut avoir avec les outils de débug présents dans tous les navigateurs (F12) mais à distance et avec un effort le plus petit possible. Notre mojo : **keep it simple**.

L'idée est simple : on collecte des informations depuis le site Web que vous voulez tester et on les affiche dans un tableau de bord Web. On peut également exécuter des commandes sur le site que vous testez pour effectuer des modifications et voir le résultat en direct. Parfait pour débuguer un site sur mobile mais aussi sur desktop.

Quand on l'a créé, on ne voulait pas que Vorlon.js soit notre projet à tous les 4, mais le projet de toute la communauté Web. C'est pour ça que c'est un projet Open Source disponible sur GitHub : https://github.com/microsoftdx/vorlonjs. C'est aussi pour cette raison que

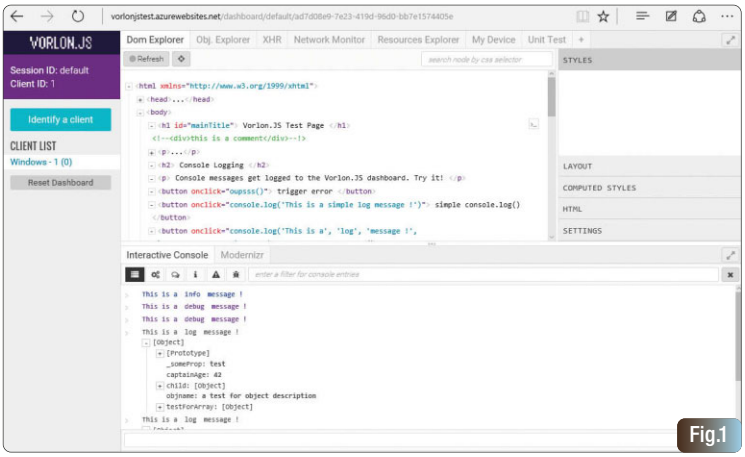


Fig.1

nous avons choisi un modèle qui permet d'ajouter facilement de nouvelles fonctionnalités à travers un système de **plugins**. Un plugin est constitué d'une partie client qui collecte les informations et effectue des actions, et d'une partie serveur qui indique au tableau de bord ce qu'il faut afficher et comment l'afficher **Fig.1**.

A l'heure actuelle Vorlon.js contient 8 plugins disponibles dont 5 ont été réalisés par la communauté ! Guillaume Leborgne, par exemple, est l'auteur de l'Object Explorer et de nombreuses améliorations du DOM Explorer qu'il a réalisé avec Mehdi Lahlou. L'inspecteur Angular.js a été créé par Sébastien Ollivier et Pierre-Alexandre Gury. Le futur plugin Babylon.js est réalisé par Julian Chenard. Bref : Vorlon.js est enrichi par de plus en plus de développeurs, et on en est ravis. :)

Comment l'utiliser ?

La première question qu'on peut se poser c'est : comment utiliser Vorlon.js ? Concrètement, Vorlon.js est un site développé en node.js. Celui-ci sert une page Web qui correspond au tableau de bord (développé en Express.js) et expose un server WebSocket (réalisé à l'aide de Socket.io) Vous avez trois possibilités :

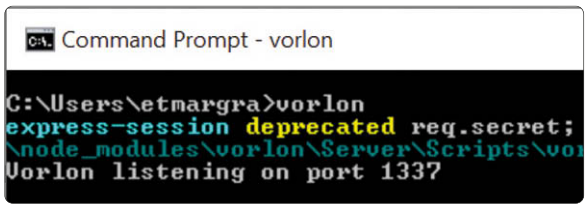
- L'installer via Node.js Package Manager ;
- Cloner l'entreprôt GitHub et compiler votre propre version en local ;
- L'installer automatiquement dans une application Web Microsoft Azure.

Pour le premier cas, le seul prérequis est d'avoir installé Node.js localement. Une fois que c'est fait, saisissez la commande :

```
npm install -g vorlon
```

Ceci a pour effet de télécharger la dernière version stable publiée sur l'entrepôt de NPM. Une fois installé, vous pouvez démarrer le serveur en tapant uniquement :

```
Vorlon
```



Note : la version actuellement déployée dans NPM contient une librairie obsolète. Ceci sera corrigé dans la prochaine version qui sort sous peu !

Vous pouvez alors accéder à Vorlon.js en naviguant sur l'url http://localhost:1337/dashboard/default

La dernière étape, pour que votre site Web puisse être débuggé via le tableau de bord, il faut ajouter une référence vers un script JavaScript qui est généré par le serveur :

```
<script src="http://localhost:1337/vorlon.max.js"></script>
```

A partir de ce moment, chaque navigateur qui accède à votre page devient un client pour le tableau de bord. Vous pouvez alors ouvrir la page de debug associée à chaque client et visualiser les informations remontées dans le tableau : autrement dit : let's debug !
Une autre manière d'installer la partie serveur de Vorlon.js est de cloner l'entrepôt GitHub. Pour cela vous devez installer un outil en ligne de commandes pour Git et taper ensuite l'instruction suivante pour récupérer les sources.

git clone http://github.com/microsoftdx/vorlonjs

```
C:\Users\etmargra>git clone http://github.com/microsoftdx/vorlonjs
Cloning into 'vorlonjs'...
remote: Counting objects: 4868, done.
remote: Total 4868 (delta 0), reused 0 (delta 0), pack-reused 4868
Receiving objects: 100% (4868/4868), 79.45 MiB | 1.13 MiB/s, done.
Resolving deltas: 100% (3392/3392), done.
Checking connectivity... done.
```

Pour compiler la version, rendez-vous à la racine du répertoire et tapez :

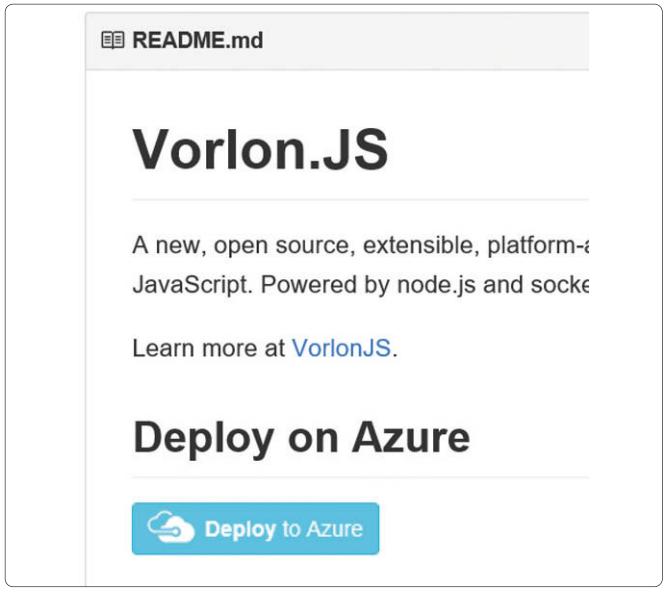
```
npm install
```

Cela aura pour effet d'installer les dépendances et de compiler les Plugins, puis le Server. Pour exécuter le serveur, saisissez :

```
npm start
```

```
C:\Users\etmargra\vorlonjs>npm start

> vorlon@0.0.15 start C:\Users\etmargra\vorlonjs
> npm run prepublish && node ./Server/server.js


> vorlon@0.0.15 prepublish C:\Users\etmargra\vorlonjs
> npm run build
```

La dernière option est de choisir d'installer une instance de Vorlon.js dans le Cloud. Vous pouvez choisir d'utiliser Microsoft Azure. Nous vous avons simplifié la tâche, il suffit de cliquer sur le bouton **Deploy on Azure** disponible dans le readme de la page d'accueil du projet GitHub.



Un assistant en 3 étapes créera une Azure Web App et déploiera automatiquement la dernière version de Vorlon.js !

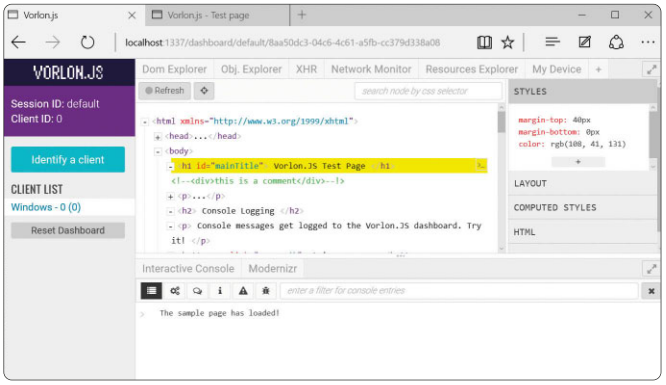
Quels plugins à l'heure actuelle ?

Chaque fonctionnalité de Vorlon.js est un plugin. Dans le tableau de bord, un plugin se matérialise par un onglet et un résultat qui s'affiche dans cet onglet. Le principe de fonctionnement est très simple. Le fichier vorlon.js que vous référencez dans votre site Web est généré par le serveur. Il contient le code qui récolte les données pour chaque plugin actif et également le code qui peut exécuter du code JavaScript dans votre site Web à la demande du tableau de bord.
Pour savoir quels plugins sont actifs et pour en modifier la liste, vous pouvez changer le fichier **config.json** qui se situe dans le répertoire **/Server** de votre installation (dans nodes_modules si vous l'avez installé à partir de NPM).

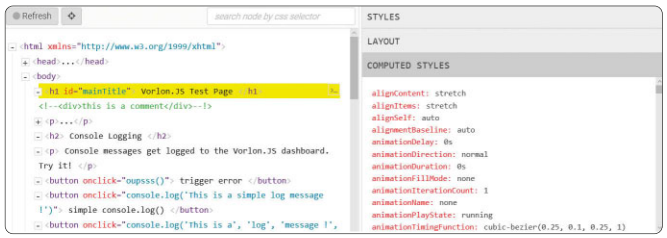
Note : C'est également dans ce fichier que vous pouvez modifier quelques paramètres de Vorlon.js tels que l'authentification ou la configuration d'un proxy. Vous pouvez vous référez à la documentation disponible sur http://www.vorlonjs.io
Faisons un tour des plugins pour comprendre ce que vous pouvez faire avec cet outil.

DOM Explorer

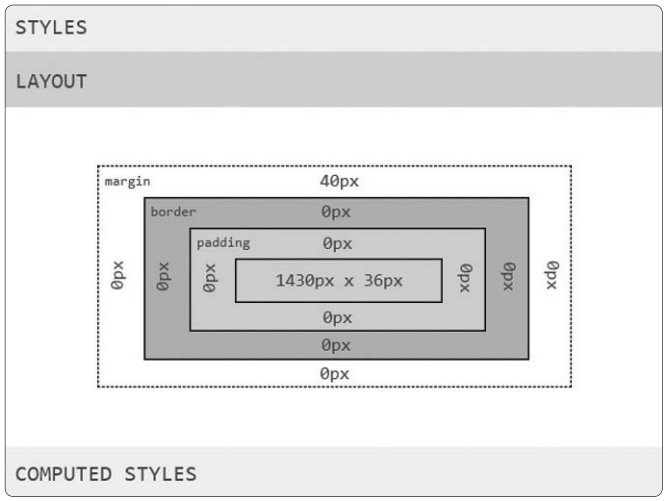
Le DOM explorer permet d'accéder au code HTML actuellement affiché sur le site que vous êtes en train de débuguer. L'ensemble des fonctionnalités du DOM Explorer classique disponible dans ce plugin. Vous pouvez modifier le contenu texte d'un élément, ajouter un attribut sur un élément, modifier le contenu CSS en ajoutant des attributs ou en modifiant ceux qui existent. Les modifications sont appliquées en temps réels sur le site que vous êtes en train de tester.
Quand vous survolez un élément dans le DOM Explorer, sa correspondance dans le site Web sur le client est mis en surbrillance pour que vous puissiez facilement l'identifier.



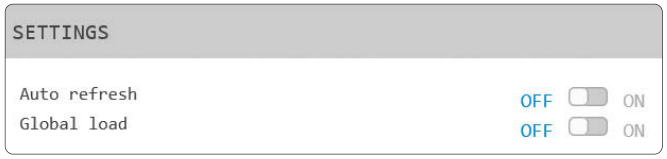
Les fonctionnalités avancées comme les styles CSS calculés (à l'opposé de ceux qui sont explicitement présents dans le fichier) sont aussi disponibles.



Vous pouvez également connaître la taille réelle des bordures et autres marges avec la section **layout**.

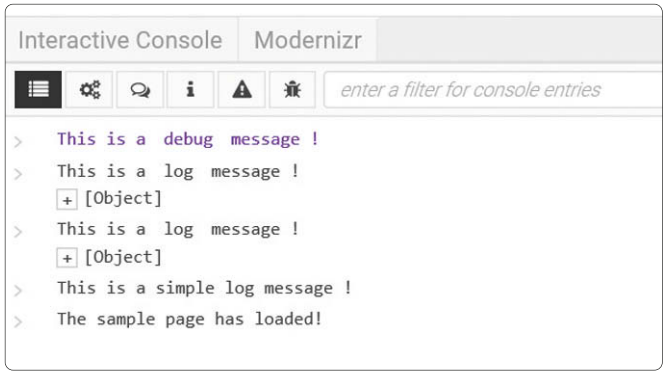


Deux paramètres permettent de modifier le comportement par défaut de rafraîchissement du DOM depuis le client. Par défaut, seul une petite pastille s'allume en rouge sur le bouton **refresh** pour indiquer que le DOM a changé et qu'il faut le rafraichir. Vous pouvez modifier ça pour que le DOM se rafraîchisse en temps réel.



Interactive Console

La console interactive est également un des plugins cœurs de Vorlon.js. Cela permet de visualiser tout ce qui est logué dans la console du client Web dans le tableau de bord. Les erreurs continuent à être loguées localement mais vous les voyez également via le plugin. Vous pouvez également visualiser des objets ou erreurs plus complexes et naviguer à l'intérieur de celles-ci.



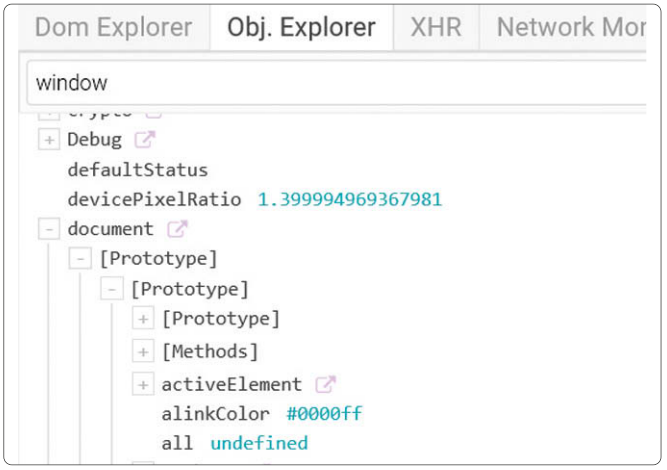
Dans l'affichage de la console, vous pouvez également filtrer par type de message et rechercher un message particulier.

Modernizr

Ce plugin vous simplifie la vie pour l'activation de l'outil modernizr. Il est référencé par le plugin et est automatiquement référencé dans votre client. L'interface graphique du plugin vous indique quelles fonctionnalités HTML et CSS sont activées sur le navigateur via lequel vous naviguez sur votre client. Idéal pour les navigateurs sur mobiles pour lesquels cette information est parfois complexe à obtenir.

Object Explorer

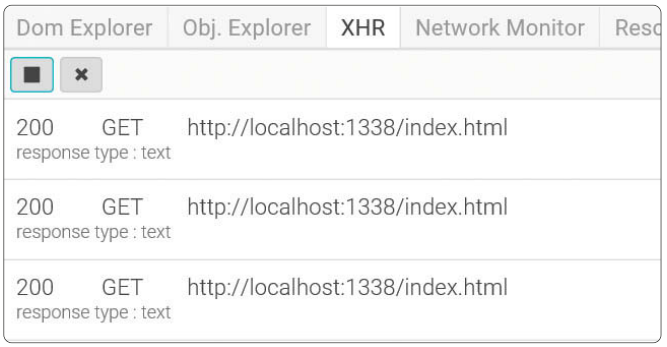
Un des enfers qui peut être créé par un code JavaScript mal conçu ou contenant des erreurs se retrouve dans les variables. Parfois on crée une variable dans un autre scope que celui qu'on voudrait. C'est exactement pour vérifier ça que l'Object explorer est conçu : vous avez accès à l'ensemble des objets qui sont stockés dans window et en dessous.



Vous pouvez donc voir facilement si ce que vous créez et manipulez est bien au bon endroit !

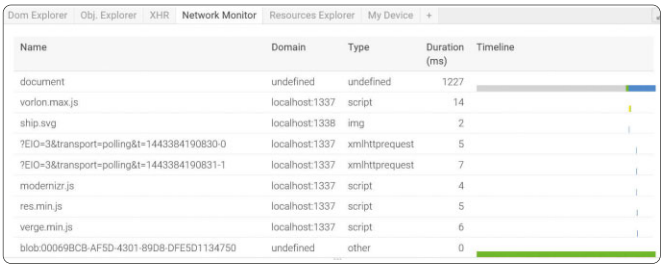
XHR

Ce plugin est présent par défaut mais n'est pas activé pour ne pas impacter en continu les performances du réseau. Vous pouvez démarrer l'écoute des requêtes XHR (autrement dit, les requêtes Ajax) en cliquant sur le bouton **play** en forme de triangle.



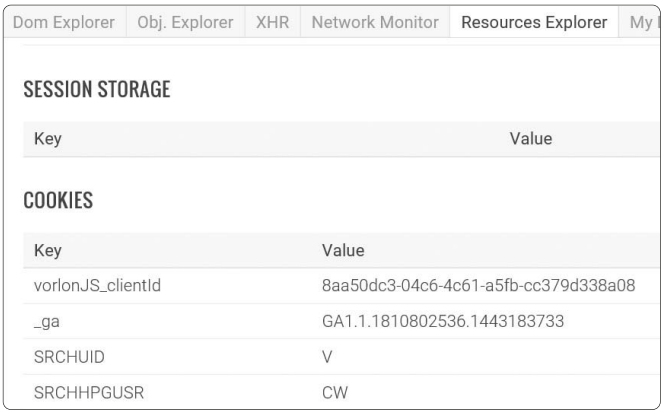
Network Monitor

Ce plugin est votre allié dans la quête de la performance de vos pages. Vous y retrouverez l'ensemble des requêtes effectuées sur les ressources dépendantes. Pour chaque requête, vous obtenez le domaine, le type, sa durée en millisecondes et une vue via une timeline pour vous représenter visuellement leurs temps relatifs les unes aux autres.



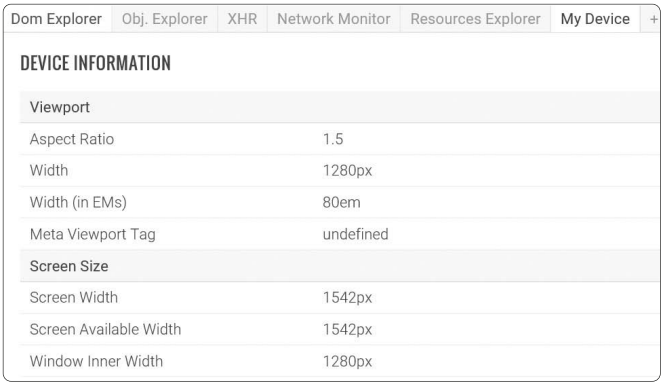
Resources Explorer

Votre page Web stocke des informations sur le poste de l'utilisateur. Ces informations peuvent être du Local Storage, des sessions ou encore des cookies. Ce plugin affiche la liste de ces ressources et les valeurs associées.



My Device

Quand vous testez votre site Web sur un mobile ou une tablette, vous avez parfois besoin d'avoir plus de détails sur les paramètres du navigateur. La largeur de l'écran, l'aspect ratio, les points par pouce ou encore le User Agent sont des informations que vous pouvez visualiser dans le plugin My Device.



NG Inspector

Ce plugin permet de visualiser les Scope dans une application Angular.js. Vous visualisez également les sous-scope et les valeurs qui sont associées. Pour l'instant seule la consultation est possible, mais son auteur nous indique travailler sur une version qui permet de modifier également les valeurs associées.

Note : Ce plugin n'est pas activé par défaut pour ne pas générer d'erreur quand votre site Web n'utilise pas Angular.js. Vous pouvez l'activer dans le fichier config.json.

Babylon.js inspector

Ce plugin est en cours de finalisation et est peut-être déjà disponible lors de la parution de cet article. Il permet de visualiser des informations de debug liées à l'utilisation du moteur webGL Babylon.js. Ces informations sont normalement disponibles dans une surcouche de Babylon : le debug layer. L'avantage d'utiliser Vorlon.js est de n'afficher rien d'autre dans votre jeu ou scène 3D que la scène elle-même. Cela vous permet de déporter ce debug layer.

Comment créer un plugin ?

La création d'un plugin est très simple. Si on reprend le principe d'une page Web moderne : elle contient généralement du code JavaScript qui interagit avec la page au sein du même navigateur. Par exemple, si vous vouliez créer l'équivalent du plugin **My Device**, il suffit de collecter les informations de taille d'écran etc. et de les affecter à des éléments HTML pour les afficher dans la même page. La transformation de ce concept en plugin consiste à séparer la collecte des données (ici les informations du device) et son affichage. L'architecture d'un plugin Vorlon.js vous aide largement dans ce travail.

Vous devez créer un premier fichier JavaScript qui correspond à la partie cliente. Elle a une organisation bien particulière et certaines fonctions seront automatiquement appelées par Vorlon.js, notamment pour démarrer la collecte.

Vous devez également créer un second fichier JavaScript qui contient tout le code qui a pour rôle d'afficher les données collectées dans le tableau de bord. L'affichage HTML et le transfert des données entre la partie cliente et serveur est très simple : nous avons créé des helpers pour vous faciliter la tâche.

Le détail de création d'un tel plugin sort du cadre de cet article. Si vous voulez vous lancer dans l'aventure, j'ai écrit un article de blog en anglais qui vous indique la marche à suivre : http://bit.ly/vorlonplugin.

Conclusion

Nous n'en sommes encore qu'au tout début de l'aventure Vorlon.js et déjà une vraie communauté s'est créée autour du projet. Notre objectif est de continuer à en faire l'outil de tous les développeurs Web. C'est pour ça que nous avons besoin de toutes les contributions et de tout ce que vous pourrez imaginer.

C'est déjà un outil très utilisé et nous avons hâte de voir la dizaine de plugins actuelle se transformer en dizaines, puis en centaines pour en faire la plateforme souple et incontournable pour le debug telle que nous l'avons imaginée.

REFACTORING TO FUNCTIONAL 2e partie

Les conférences d'Hadi Hariri, développeur et évangéliste technologique chez JetBrains, sont pour moi des conférences à ne pas manquer à Devoxx France, non seulement du fait de ses qualités indéniables d'orateur, mais aussi de la diversité et de la pertinence des sujets traités.

Jérôme Valloire
SOAT

Apprendre à réfléchir "fonctionnel"

Des fonctions comme primitives réutilisables du langage

Dans la programmation fonctionnelle, les fonctions sont au cœur du langage. Elles sont des briques essentielles qu'il est possible de combiner via des **Fonctions d'Ordre Supérieur,** qui prennent en entrée des fonctions et qui peuvent retourner d'autres fonctions :

- *filter* avec un prédicat (une condition)
- *map* avec une fonction de transformation
- *groupBy* avec une fonction de regroupement par index

Il est également possible de réaliser du ***pipelining*** pour mettre en place des chaînes de traitement :

```
1 albums.filter { x -> x.year >= 1976 }
2     .map { x -> Pair(x.title, x.year) }
3     .groupBy {x -> x.second }
```

A ce niveau, Hadi Hariri nous met en garde vis-à-vis de ce principe de *pipelining* afin d'éviter de multiplier à l'infini les fonctions dans le *pipeline,* ce qui aurait pour effet de limiter la lisibilité du code.
Il nous encourage à découper le pipeline en créant de nouvelles fonctions, aux noms explicites, pour extraire des regroupements de fonctions de base du pipeline. Un autre aspect proche du *pipelining* est la possibilité de réaliser des **compositions** de fonctions de type *f(g(x))* :

```
1 public fun sum(x: Int, y: Int): Int = x + y
2 public fun squared(x: Int): Int = x * x
3
4 val squaredSum = compose(::sum, ::squared)
```

Cela permet de respecter :

- Le principe de *Single Responsibility Principle* (SRP), appliqué aux fonctions en écrivant des fonctions simples, puis en les composant pour réaliser des traitements plus complexes,
- Le principe *Don't Repeat Yourself* (DRY), en créant une nouvelle fonction permettant d'extraire du code commun à plusieurs fonctions, puis en utilisant la composition pour appeler cette nouvelle fonction.

Les notions d'**application partielle** et de **curryfication** sont également très présentes en programmation fonctionnelle :

```
1 val sum = { (x: Int, y: Int) -> x + y }
2
3  // Partial Function Application
4 val sumNumberWith10 = sum.partial(10)
5 println(sumNumberWith10(5)) // -> 15
6
7 // Currying
8 val sumCurried = sum.curry()
9 println(sumCurried(3)(2)) // -> 5
```

La curryfication, du nom du mathématicien Haskell Curry, désigne l'opération qui permet, à partir d'une fonction à plusieurs arguments, d'obtenir une fonction à un argument retournant une fonction prenant le reste des arguments, permettant ainsi l'application partielle.

Des données à la recherche de l'immutabilité

Après s'être principalement intéressé aux fonctions, Hadi Hariri souhaite désormais revenir sur la notion de données.
Ces données peuvent être de différents types dans une approche fonctionnelle :

- Des **scalaires** : âge, date de naissance, montant...
- Des **collections de types abstraits** ("*Abstract Data Type*") : liste de clients, liste de factures...

Ces listes sont les entrées et les sorties des fonctions vues précédemment (*map, filter...*) et les scalaires peuvent être obtenus à partir de listes via d'autres fonctions (*first, last, find, aggregate...*).
Hadi Hariri nous présente alors plus particulièrement la fonction *fold,* qui permet d'appliquer une transformation sur les éléments d'une collection, en collectant les résultats au fur et à mesure via un accumulateur. Une fois que l'on a traversé toute la liste, seul l'accumulateur reste, c'est la valeur scalaire à laquelle on a réduit la liste :
Par exemple, pour déterminer le maximum dans une liste d'entiers :

```
1 public fun maximumFold(list: List<Int>) : Int {
2     return list.fold(0, {x, y -> Math.max(x, y)})
3 }
```

La programmation fonctionnelle dispose également de deux outils particulièrement intéressants : le "***pattern matching***" et la **récursivité**.

```
1 public fun maximumRecursive(list: List<Int>): Int {
2   when (list.count()) {
3       0 -> throw IllegalArgumentException("empty list!")
4       1 -> return list[0]
5       else -> return Math.max(list[0], maximumRecursive(tail(list)))
6   }
7 }
```

Comme on le voit dans la fonction ci-dessus, le "*pattern matching*" fournit une manière simple de construire des structures alternatives.
On est en effet bien loin des séries de *if ... else ... if* ou des*switch ... case,* que nous devons utiliser en Java pour répondre à ce type de problématique.
La récursivité n'est quant à elle pas limitée à la programmation fonctionnelle, mais l'utilisation de structures récursives est beaucoup plus courante dans ce type de programmation que dans le style de programmation impératif, où l'utilisation des boucles de type *for* est plus habituelle.
L'utilisation de la récursivité permet également de favoriser l'immutabilité des données, en évitant l'utilisation de variables d'état.
D'ailleurs, afin de limiter ces variables d'état, Hadi Hariri nous conseille également de traiter les listes comme des structures infinies, rendant possible l'évaluation paresseuse ("*Lazy evaluation*") et la programmation réactive.

Lorsque l'on fait de la programmation fonctionnelle, nous devons toujours rechercher à privilégier l'immutabilité des données, par exemple en créant une nouvelle liste plutôt qu'en modifiant une liste existante lors de l'application d'un traitement sur celle-ci.

Programmation fonctionnelle et performances

Hadi Hariri s'intéresse maintenant à l'aspect performance de la programmation fonctionnelle.

Mémoïzation

Il utilise tout d'abord l'exemple du calcul de la suite de Fibonacci, résolu ici dans un style purement récursif :

```
1 public fun fibonacciRecursive(n: Int) : Long {
2    if (n == 0 ||n == 1) {
3       return n.toLong()
4    } else {
5       return fibonacciRecursive(n -1) + fibonacciRecursive(n - 2)
6    }
7 }
```

Une première technique d'optimisation est l'utilisation de la mémoïzation, qui consiste à réduire le temps d'exécution d'une fonction en mémorisant ses résultats d'une fois sur l'autre.
Pour ce faire, on introduit tout simplement un cache qui va éviter le recalcul des valeurs déjà calculées lors de récursions intermédiaires :

```
1 val cache = hashMapOf<Int, Long>(0 to 0, 1 to 1)
2
3 public fun fibonacciMemoization(n: Int) : Long {
4    if (cache.containsKey(n)) {
5       return cache.get(n)!!.toLong()
6    } else {
7       val result = fibonacciMemoization(n -1) + fibonacciMemoization(n - 2)
8       cache.set(n, result)
9       return result
10      }
11   }
```

Récursion terminale

Comme on a déjà pu l'évoquer précédemment, la programmation fonctionnelle utilise de nombreuses structures récursives, ce qui peut poser des problèmes de gestion de la pile, la zone mémoire réservée à l'exécution d'un programme.
En effet, dans une procédure récursive, toutes les variables locales sont stockées dans la pile d'exécution et empilées autant de fois qu'il y a d'appels récursifs, avant d'être désempilées au fur et à mesure qu'on remonte les niveaux une fois la condition de sortie rencontrée.
Ainsi, si on ne fait pas attention à la profondeur de la récursivité, la pile se remplit progressivement jusqu'à atteindre sa limite de taille, ce qui entraîne le problème bien connu de débordement de pile : "*stack overflow*".
Afin de limiter ces impacts, on peut utiliser la technique de récursion terminale (*tail recursion*). Une fonction à récursivité terminale (*tail-recursive*) est une fonction où l'appel récursif est uniquement la dernière instruction à être évaluée.

```
1 public fun factorial(number: Int): Int {
2    when (number) {
3       0,1 -> return 1
4       else -> return number * factorial(number - 1)
5    }
6 }
```

La fonction de calcul de factorielle ci-dessus n'est pas *tail-recursive* car le résultat de l'appel récursif*factorial (number – 1)* est utilisé par la fonction *. À cause de cela, le résultat de *factorial (number – 1)* doit être conservé dans la pile d'exécution pour être utilisé par *. Cependant il est possible de modifier l'implémentation de cette fonction pour la rendre *tail-recurvive*:

```
1 public fun factorialTailCall(number: Int): Int {
2    return factorialTC(number, 1)
3 }
4
5 public fun factorialTC(number: Int, accumulator: Int): Int {
6    when (number) {
7       0 -> return accumulator
8       else -> return factorialTC(number - 1, accumulator * number)
9    }
10   }
```

Ici, la fonction *factorialTC* s'appelle elle-même uniquement lors de sa dernière instruction, ce qui permet d'économiser de l'espace mémoire car aucun état, sauf l'adresse de la fonction appelante, n'a besoin d'être sauvé sur la pile d'exécution.
Ainsi, alors que l'espace consommé sur la pile d'exécution augmente linéairement lors de l'exécution récursive, il reste constant après l'optimisation *tail-recursive*, diminuant ainsi nettement l'empreinte mémoire.
Le recours à un accumulateur (*accumulator* dans l'implémentation *tail-recursive* de factorielle) est une technique couramment utilisée pour l'écriture de fonctions *tail-recursive*. A noter qu'en Kotlin, il est nécessaire de préciser via une annotation *tailRecursive* que l'on souhaite utiliser une optimisation de type *tail-recursive,* appelée *Tail Call Optimization :*

```
1 tailRecursive
2 public fun factorialTC1(number: Int, accumulator: Int): Int {
3    when (number) {
4       0 -> return accumulator
5       else -> return factorialTC1(number - 1, accumulator * number)
6    }
7 }
```

Cela permet d'indiquer au compilateur que l'on souhaite faire cette optimisation et donc de vérifier si celle-ci est effectivement réalisable afin d'alerter le développeur si ce n'est pas le cas.
Attention néanmoins, car tous les langages fonctionnels ne prennent pas en charge ce type d'optimisation ! Par exemple, Java supporte les appels *tail-recursive*, mais il n'y apporte aucune optimisation car celle-ci n'est pas disponible dans la JVM, contrairement à Scala où le compilateur est capable de réaliser cette optimisation (via l'annotation *@tailrec*).

Fonctions *inline*

Enfin, Hadi Hariri avoue que les fonctions d'ordre supérieur peuvent avoir des impacts négatifs sur les performances.
En effet, en Kotlin, chaque fonction est un objet qui capture une *closure*, l'ensemble des variables qui sont accessibles dans le corps de la fonction. L'allocation mémoire pour ces fonctions et ces variables, ainsi que les appels virtuels, introduisent donc inévitablement un surcoût à l'exécution.

Afin de limiter cet impact, il est possible de déclarer ces fonctions comme *inline* ; dans ce cas, le code complet de la fonction (ainsi que des lambdas utilisés) est inséré dans le code de l'appelant à l'exécution.
Là encore, tous les langages ne disposent pas de cette possibilité de déclarer des fonctions comme *inline*. En Java, il n'est pas possible de suggérer au compilateur qu'une fonction doit être *inline,* mais la JVM réalise néanmoins ses propres optimisations à l'exécution, ce qui permet de limiter les impacts du paradigme fonctionnel.

Quid de tous les concepts effrayants de la programmation fonctionnelle ?

Comme vous avez pu vous en rendre compte, mis à part la **curryfication** et l'**application partielle**, aucun des termes habituellement utilisés par les puristes de la programmation fonctionnelle n'ont été cités dans cette présentation. Avant de conclure sa présentation, Hadi Hariri décide néanmoins d'évoquer rapidement certains de ces termes :

Foncteurs *("functors")*

Les foncteurs sont des éléments sur lesquels on peut réaliser une transformation, par exemple une collection de **a** où l'on peut appliquer une fonction **a -> b** pour retourner une collection de **b.**
La fonction *map* est donc un exemple simple de foncteur que nous avons déjà eu l'occasion de manipuler précédemment.

Monades

Une monade permet d'encapsuler une valeur, un traitement ou un résultat potentiel. Il est ainsi possible de voir une monade comme une boîte, vide ou ayant un contenu, qui nous fournit des abstractions et des opérations au dessus de la valeur éventuellement encapsulée.
On peut citer différentes monades usuelles :

- La **monade Option**, appelée aussi monade *Maybe* permettant d'enchaîner des traitements unitaires pouvant potentiellement ne pas retourner de valeur. Scala utilise le type *Option* permettant de caractériser la présence ou l'absence de valeur via deux sous-types, *Some[T]* ou *None.*
- La **monade List**, permettant d'appliquer des opérations unitaires sur des ensembles de valeurs. On peut ici citer le type *IEnumerable<T>* de C#.
- Les **Futures**, permettant d'encapsuler un résultat qui arrivera plus tard, qui sont utilisées pour réaliser des opérations asynchrones ou en programmation réactive.

Ces termes théoriques cachent des concepts mathématiques, mais comme le souligne Hadi Hariri, nul besoin de les maîtriser pour adopter les concepts de la programmation fonctionnelle présentés lors de sa conférence et commencer ainsi à refactoriser votre code. Si vous êtes intéressés par ces concepts, je vous invite à parcourir le Web où les ressources sont plus que nombreuses, mais encore une fois, le plus important, c'est de savoir les mettre en pratique de manière concrète.

Conclusion

En conclusion, Hadi Hariri nous encourage fortement à expérimenter la programmation fonctionnelle et ainsi à arrêter de réfléchir uniquement en termes d'objets en adoptant les fonctions comme des éléments primitifs du langage. L'utilisation de fonctions devrait nous permettre de nous concentrer sur différents points :

- Ecrire moins de code,
- Ecrire du code plus descriptif (le *what* plutôt que le *how*),
- Ecrire du code plus facile à comprendre en séparant données et traitements,
- Ecrire du code plus déterministe en favorisant l'immutabilité et en limitant les effets de bord.

It is not Rocket science !!! (**Hadi Hariri**)
Finalement, Hadi Hariri nous conseille différentes lectures pour nous aider à mieux appréhender les valeurs essentielles de la programmation fonctionnelle :

- *The little schemer* de Daniel P. Friedman et Matthias Felleisen
- *Learn you a Haskell for Great Good!* de Miran Lipova
- *Functional Programming in Java: Harnessing the Power Of Java 8 Lambda Expressions* de Venkat Subramaniam

Pour ma part, mes premiers pas en programmation fonctionnelle ont été liés à la découverte de Scala, et ont, aujourd'hui, considérablement modifié ma manière de coder en Java, notamment via l'utilisation de librairies telles que Guava, permettant d'apporter une touche de fonctionnel à Java et de se préparer ainsi efficacement à l'utilisation des nouvelles fonctionnalités offertes par Java 8.
En plus, pourquoi chercher continuellement à opposer programmation orientée objet et programmation fonctionnelle? Pour moi, les deux paradigmes ne sont pas là pour s'opposer mais pour se compléter, libre au bon développeur d'utiliser le bon outil pour résoudre le bon problème.
Tout comme Hadi Hariri, je ne peux que vous conseiller d'expérimenter la programmation fonctionnelle. C'est effectivement une manière différente de coder, mais avec le temps l'approche fonctionnelle devient de plus en plus naturelle ; et si cela peut faire de vous un meilleur codeur, alors pourquoi ne pas essayer ?
Enfin, pour ceux qui souhaitent s'initier à la programmation fonctionnelle, je vous recommande vivement le cours en ligne de Martin Odersky sur les Principes de la programmation fonctionnelle dans Scala, disponible via Coursera.

Node.js de A à Z

2e partie

Je ne vous apprends rien en vous disant que beaucoup de développeurs Web utilisent JavaScript pour créer des applications front. Node.js permet à ce langage très populaire d'être utilisé dans plusieurs autres contextes, par exemple sur un serveur Web. Il existe plusieurs fonctionnalités notables offertes par Node.js ce qui le rend sans nul doute digne d'un grand intérêt.

Wassim Chegham (@manekinekko)
Expert en Nouvelles Technologies Web chez Groupe SII
Google Developer Expert (GDE) en AngularJS

HTTP

Serveur HTTP

Node offre une API HTTP extrêmement simple à utiliser, tellement simple que réaliser un serveur HTTP avec Node est devenu le "Hello World" de Node. Voyez par vous-même : Fig 48

Exécutez votre script, puis accéder à l'adresse http://localhost:1337 et vous aurez votre "Hello, Programmez!" : Fig 48

Bravo ! Vous venez de créer votre serveur HTTP avec Node.

Voici quelques explications de ce que fait Node, sous le capot... Premièrement, nous chargeons le module **HTTP**, et nous utilisons la méthode **createServer()** pour créer notre serveur HTTP. Cette méthode prend un callback en paramètre qui sera invoqué à chaque nouvelle connexion d'un client HTTP à notre serveur. Ensuite nous appelons la méthode **listen(1337),** ce qui permet à notre serveur d'écouter toutes les connexions entrantes sur le port 1337. Lorsqu'un client se connecte, le callback est exécuté, et nous avons à notre disposition deux objets : un objet représentant la requête reçue — de type **HTTP.IncomingMessage** — et un objet représentant la réponse retournée au client — de type **http.ServerResponse**. La requête contient des données telles que l'URL demandée. La réponse est l'objet qui contient la réponse ainsi que d'autres données telles que le statut de la réponse ou encore des en-têtes HTTP tels que *Content-Type*. Dans notre exemple, nous avons ignoré l'objet **req**, et nous avons configuré l'objet **res** pour positionner le statut à 200, et le contenu en tant que "Hello, Programmez!" sous forme de texte. Vous noterez que nous avons utilisé la méthode **end()** qui permet d'envoyer un contenu dans la réponse juste avant de fermer la connexion.

Comprendre l'objet HTTP.IncomingMessage

Lorsque nous écoutons l'évènement **request**, le callback invoqué reçoit l'objet **HTTP.IncamingMessage** en tant que premier paramètre. Cet objet contient tout un tas de propriétés que nous pouvons exploiter, parmi lesquelles nous pouvons citer :

- **req.url** : cette propriété contient l'URL de la requête. Elle ne contient ni le protocole, ni le domaine, ni le port, mais tout ce qui suit. Par exemple : Fig 49 et 50
- **req.headers** : cet objet contient toutes les en-têtes qui sont envoyées avec la requête. Inspectons-les : Fig 51
- **req.method** : ceci contient la métode HTTP utilisée pour la requête : GET, POST, PUT, DELETE, HEAD, etc. Fig 52

Comprendre l'objet HTTP.ServerResponse

Le second paramètre qui est passé au callback invoqué lors de l'événement request représente la réponse utilisée pour répondre au client. Grâce à cet objet, nous pouvons définir des en-têtes et le corps de la réponse.

Envoyer des en-têtes

Pour envoyer des en-têtes, nous utilisons la méthode **writeHead(status, headers)**. Cette méthode permet en même temps de définir le statut de la réponse, et de renseigner un paramètre optionnel représentant les en-têtes.

Fig.48

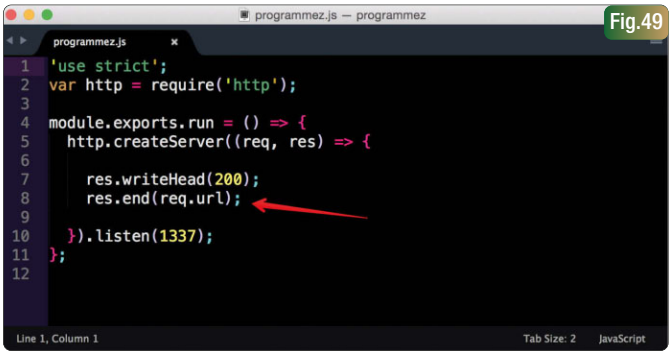


Fig.49

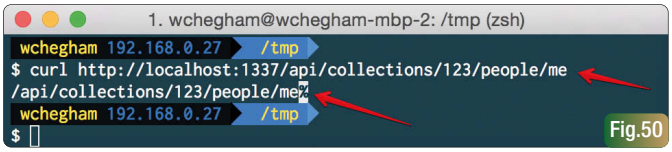


Fig.50

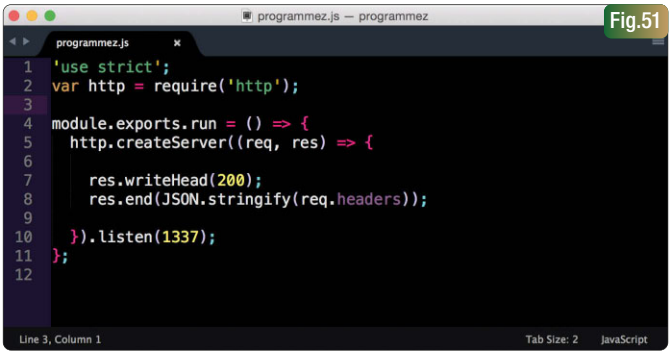


Fig.51

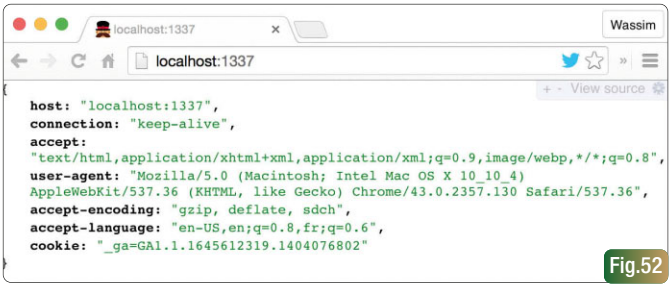


Fig.52

Fig 53 et 54. Il est également possible de manipuler des en-têtes que vous auriez déjà positionnées, soit en en ajoutant d'autres ou en en enlevant, grâce aux méthodes **setHeader(name, value)** et **removeHeader(name)**.

Envoyer le corps de la réponse

Bien évidemment, nous devons également envoyer une réponse en plus des en-têtes. Pour cela, nous utilisons la méthode **write()** qui prend en paramètre soit du texte ou bien un Buffer : Fig 55

Envoyer un flux de réponses

L'une des forces de Node, c'est sa capacité à consommer et produire des flux depuis différentes sources. L'encodage de transfert en bloc [13] (*chunked encoding*) est un mécanisme qui permet à un serveur de continuer d'envoyer des données au client sans envoyer la taille de la réponse. Node envoie donc l'en-tête suivante au client : **Transfert-Encoding: chuncked.** Un exemple d'utilisation de ce mécanisme avec Node, serait l'envoi d'un flux venant d'un autre processus ou de la lecture d'un fichier : Fig 56 et 57

Nous constatons dans cette capture d'écran que le client ne ferme pas la connexion, puisque le serveur lui envoie des données en continu.

Nous venons de voir qu'il est très simple de réaliser un serveur HTTP avec Node. Dans la section suivante, nous allons voir comment réaliser un client HTTP avec Node, et vous allez voir, ce sera tout aussi simple.

Client HTTP

Grâce à sa gestion performante des E/S, Node offre non seulement une API pour créer des services HTTP, comme nous l'avons vu précédemment, mais aussi un module de base — le module **HTTP** — pour interagir avec d'autres services HTTP. Dans cette section, nous allons voir comment envoyer des requêtes HTTP avec le module **HTTP** de base.

Requêtes GET

Voyons voir comment faire une simple requête GET avec le module **HTTP** de Node : Fig 58

Une fois exécuté, ce code crée une requête de type GET vers le host **wassimchegham.com**, port 80, en interrogeant l'URL **/programmez.txt**. Le résultat de cette requête est **200,** car j'ai pris soin de mettre ce fichier sur mon serveur. Fig 59

Fig.53

programmez.js — programmez

```
'use strict';
var http = require('http');

module.exports.run = () => {
  http.createServer((req, res) => {

    res.writeHead(200, {
      'X-Content': 'Programmez!',
      'Content-Type': 'plain/text'
    });
    res.end('Hello, Programmez!');

  }).listen(1337);
};
```

Line 11, Column 35 — Tab Size: 2 — JavaScript

Fig.54

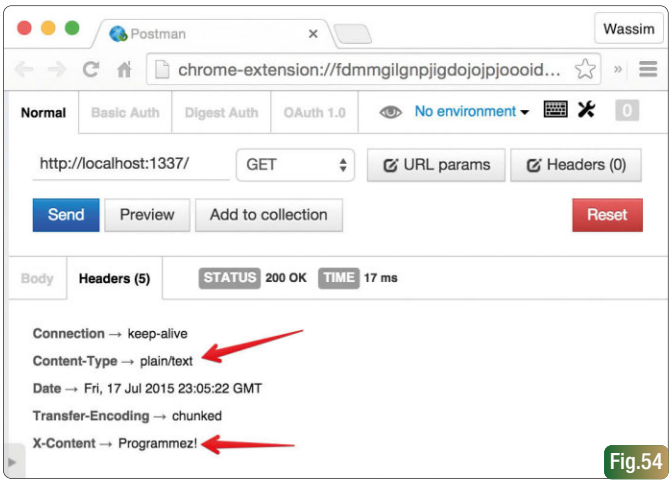


Fig.55

programmez.js — programmez

```
'use strict';
var http = require('http');

module.exports.run = () => {
  http.createServer((req, res) => {

    res.writeHead(200);
    res.write(new Buffer('Hello, '));
    res.write('Programmez');
    res.end('!');

  }).listen(1337);
};
```

Line 1, Column 1 — Tab Size: 2 — JavaScript

Fig.56

programmez.js — programmez

```
'use strict';
var http = require('http');
var spawn = require('child_process').spawn;

module.exports.run = () => {
  http.createServer((req, res) => {

    res.writeHead(200);
    var child = spawn('tail', ['-f', '/var/log/system.log']);
    child.stdout.pipe(res);
    res.on('end', () => child.kill() );

  }).listen(1337);
};
```

Line 1, Column 1 — Tab Size: 2 — JavaScript

Fig.57

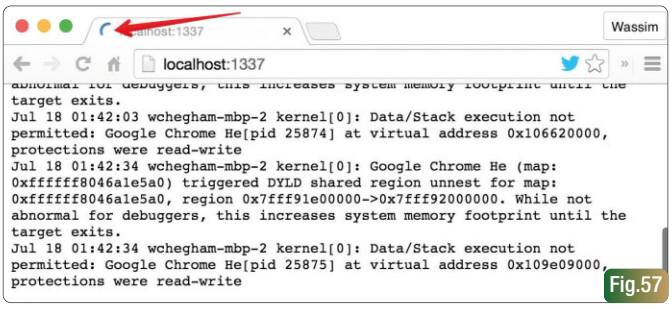


Fig.58

programmez.js — programmez

```
'use strict';
var http = require('http');

module.exports.run = () => {

  http.get({
    host: 'wassimchegham.com',
    port: 80,
    path: '/programmez.txt'
  }, (res) => {
    console.log('Got response', res.statusCode);
  });

};
```

Line 1, Column 1 — 4 misspelled words — Tab Size: 2 — JavaScript

L'objet **res** représentant la réponse HTTP, contient énormément d'informations, telles que la version du protocole HTTP, le statut de la réponse, les en-têtes de la réponse, etc. Je vous laisse inspecter tranquillement ces attributs et lire la documentation officielle pour en savoir plus.

Ce qui est important pour le moment est de savoir que cet objet de réponse implémente l'interface **Stream** de Node, et plus précisément celle concernant le flux de lecture. Ceci signifie que pour lire le contenu de la réponse, il suffit d'écouter l'évènement **data** (voir le chapitre sur les Streams). Voici un exemple plus parlant : Fig 60

Et comme par magie : Fig 61

Pour écrire le contenu du fichier http://wassimchegham.com/programmez.txt dans un fichier en local rien de plus simple ; vous l'aurez deviné, nous allons utiliser l'API des Streams. Fig 62

Dans cet exemple, nous avons créé un fichier **programmez.txt** en local, en utilisant la méthode **fs.createWriteStream()**, dans lequel nous avons écrit les données retournées par la réponse HTTP, grâce à la méthode **pipe()** de l'API Streams. La donnée est écrite au fur et à mesure qu'elles arrivent. Lorsqu'il n'y a plus de données, nous nous assurons de fermer la requête, ce qui ferme le flux de données, provoquant à son tour la fermeture du fichier local. Fig 63

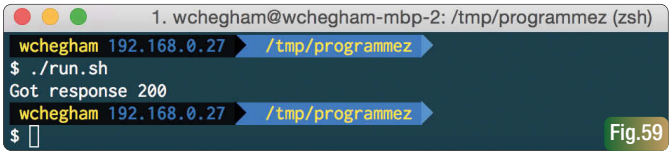


Fig.59

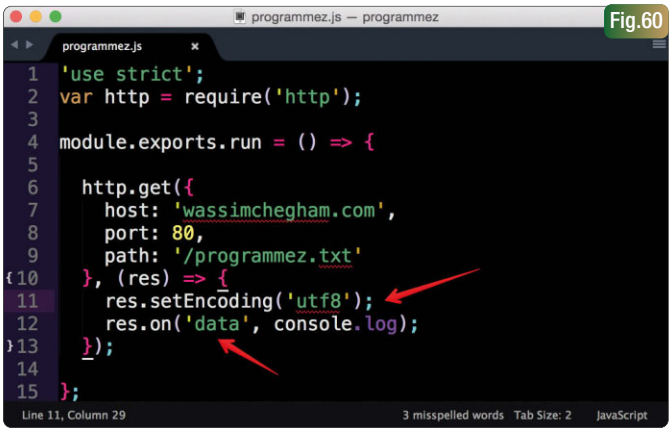


Fig.60

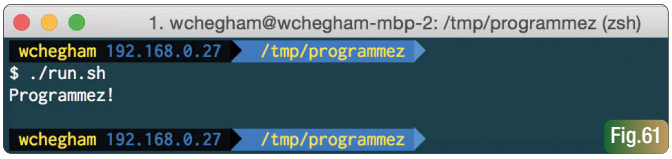


Fig.61

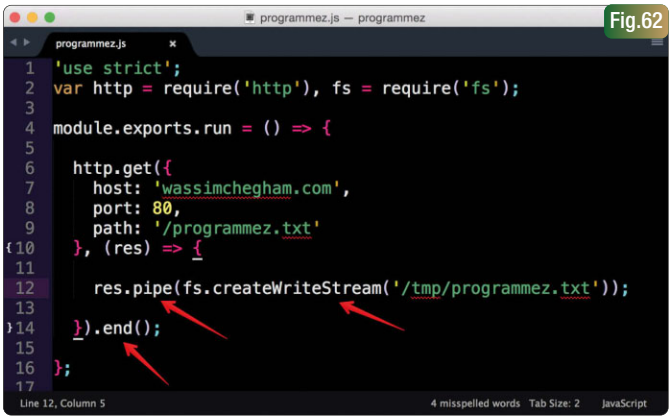


Fig.62

Utilisation d'autres verbes HTTP : POST par exemple

Dans l'exemple précédent, nous avons utilisé la méthode **http.get()** pour réaliser des requêtes de type GET. Sachez que cette méthode est un alias de la méthode **http.request()** avec l'option : **{ method: 'GET' }.**

La méthode **http.request()** retourne un objet **HTTP.ClientRequest** qui est un flux d'écriture. Nous pouvons utiliser ce flux pour envoyer des données dans le corps de la requête. Il suffit ensuite de fermer le flux pour terminer la requête. Fig 64

Pensez à bien fermer la requête (ligne 15), car sinon le serveur considérera cette requête comme invalide et ne va pas y répondre !

Voilà ! Avec le module **HTTP** de base, nous avons vu à quel point Node a simplifié la création et l'utilisation des requêtes HTTP, à travers deux exemples avec les verbes GET et POST. L'utilisation des autres verbes suit la même philosophie.

Ceci dit, même si le module **HTTP** est suffisant pour des petites applications, je vous recommande d'utiliser le module **request** [14], créé par Mikeal Roger, qui offre des fonctionnalités supplémentaires telles que la gestion des redirections HTTP, la gestion des cookies, l'envoi des données en tant que format JSON, etc.

Remerciements

Je tiens à remercier tout particulièrement mes deux collègues Benoit Colombani et Jaffar Boudad pour leur patience et leurs relectures à la fois orthographique et technique.

Icônes

Les icônes utilisées dans les schémas proviennent du site :
http://flaticons.net/

Suite et fin dans Programmez! 191.

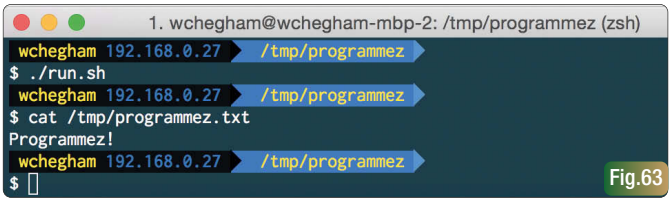


Fig.63

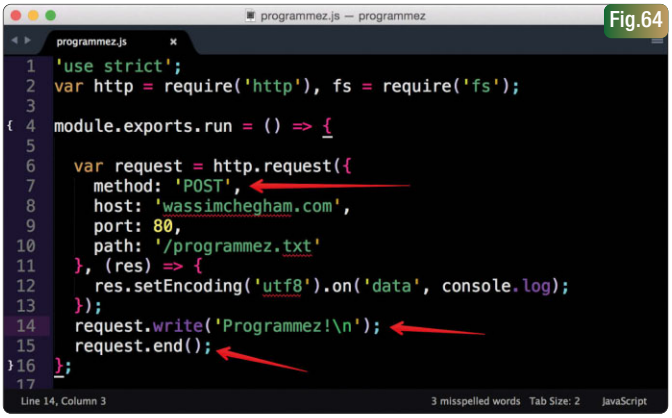


Fig.64

Références

[13] https://fr.wikipedia.org/wiki/Chunked_transfer_encoding

[14] https://github.com/request/request

Blackfire.io

Il est établi que plus de 40% des utilisateurs d'un service en ligne abandonnent après 3 secondes de temps de chargement. Et ces utilisateurs, potentiels générateurs de revenus, sont parfois perdus à jamais.

Christophe DUJARRIC
Chef de produit de Blackfire

Comment s'assurer que l'application offre une qualité de service optimale ? Peut-on éviter les soucis de performance en production en les prévenant en amont ? Peut-on automatiser des tests de performance comme on automatise aujourd'hui les tests unitaires ? C'est à ces questions que Blackfire répond, en proposant un service unique de mesure, d'analyse et d'optimisation de la performance tout au long du cycle de vie de l'application : en développement, testing/staging mais aussi en production. Lancée en Juillet 2015 après une beta de 6 mois rassemblant plus de 25.000 développeurs dans le monde, Blackfire propose à la fois un service gratuit ouvert à tous les développeurs et des offres payantes dédiées aux entreprises. Alors en quelques mots, comment ça marche ?

Une installation très simple et une documentation exhaustive

Commençons par la base : l'installation de Blackfire. Le guide utilisateur vous mènera pas à pas dans la configuration, qui revient pour l'essentiel à copier-coller une demi-douzaine de commandes dans le terminal, et laisser faire. Cela vous prendra 5 minutes maximum, que vous soyez sur une distribution Linux (Debian /Ubuntu, RedHat/Fedora/CentOS), OS X, ou encore Windows (dans un futur proche). Par ailleurs, si vous êtes utilisateur de plateformes telles qu'Heroku, Platform.sh ou autres, ce sera plus simple encore car l'agent et l'extension PHP Blackfire sont nativement embarqués dans leurs containers. Enfin pour compléter l'arsenal, Blackfire propose un container Docker et une recette Chef.

(Re-)Découvrez votre code en quelques secondes

Maintenant que vous avez installé Blackfire, vous êtes prêts à disséquer vos pages et requêtes. Blackfire n'ayant aucun overhead tant que vous ne lancez pas de requête de profiling, et n'impactant que la requête en cours lors du profil, vous pourrez tranquillement aller voir ce qu'il se passe en production aussi bien qu'en développement.

Le moyen le plus simple de profiler une page reste l'extension Chrome. En un clic sur le bouton « Profile ! » vous accédez au détail des métriques du chargement de la page, sous forme de la liste des appels de fonction et en visualisant un graphe d'appels Fig.1.

Si vous préférez travailler en lignes de commandes, l'utilitaire CLI vous apportera encore plus de flexibilité. Le préfixe blackfire run vous permettra de profiler vos scripts CLI, tandis que blackfire curl vous ouvrira les portes du profiling d'APIs et appels de Web services.

```
$ blackfire run ./my-script.php
$ blackfire run php ./my-script.php --your-script-options and arguments
$ blackfire curl http://symfony.com/
```

Pour l'heure, Blackfire vous permet d'analyser tous vos projets PHP. Mais il est aussi possible de lancer un blackfire upload, qui vous permettra de charger tout fichier au format callgrind et de bénéficier de l'UI de Blackfire pour résoudre vos problèmes de performance sur d'autres langages.

Vous voyez, là ?!

OK, maintenant que vous avez chargé votre profil, les choses sérieuses commencent. Mais Blackfire, là aussi, vous mâche le travail.

Le graphe d'appel permet non seulement d'identifier en un clin d'œil le chemin critique d'exécution (les nœuds avec une bordure de couleur), mais aussi les appels de fonctions qui, en propre, consomment le plus de ressources. A vous de vous balader dans les différentes dimensions de Wall Time, I/O time, CPU Time et Memory. Les éditions Premium et Enterprise offrent en plus la possibilité d'analyser les appels réseau et les requêtes SQL. Un des grands avantages de ce graphe, par rapport à Xhprof notamment, est que le graphe d'appel a été simplifié à l'affichage. Les appels représentant une consommation mineure ne sont par défaut pas affichés. Ceci étant, rien ne vous empêche de zoomer sur un appel de fonction pour les retrouver Fig.2.

Enfin, pour ceux que ces vues rebutent encore, l'onglet « Metrics » en haut à gauche vous permet de retrouver une liste succincte des appels les plus importants. Les librairies PHP les plus courantes sont automatiquement reconnues (Doctrine, Symfony, Twig, Smarty...). Dès que vous dépassez les limites du raisonnable, le nombre d'appels s'affiche en couleur. Mieux, vous avez la possibilité d'aller définir vos métriques personnalisées via le fichier .blackfire.yml, à stocker dans votre dépôt Fig.3.

Vos premiers tests unitaires de performance

On rentre dans le vif du sujet des tests de performance grâce à l'onglet Assertions. Le même fichier .yml va vous permettre de créer des tests sur l'ensemble des métriques disponibles, standard ou personnalisées. Ainsi vous aurez la possibilité de vous assurer qu'une page respecte vos objec-

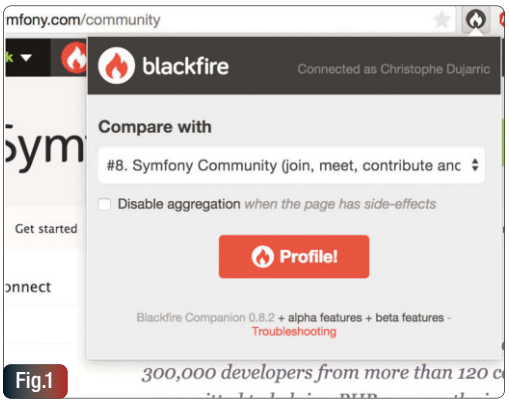


Fig.1

Fig.2

tifs de performance, en fonction de votre code et vos besoins métier. Le SDK PHP vous permet d'écrire des tests unitaires de performance grâce à son intégration native avec PHPUnit ou d'autres librairies Fig.4.

Et après ?

Maintenant que vous avez compris d'où venait votre souci de performance, et que vous avez écrit un patch, voyons s'il règle bien le problème identifié. Blackfire vous permet de comparer deux profils, et d'afficher un graphe d'appel résultant de cette comparaison. En bleu, ce qui est plus rapide, en rouge ce qui est plus lent par rapport au profil précédent Fig.5. Pour faciliter le travail en itérations, chaque profil généré peut devenir une référence de comparaison avec les prochains profils. Ainsi, en un seul clic, vous avez tout de suite la possibilité de voir l'impact de vos modifications.

La sécurité avant tout

L'équipe Blackfire communique régulièrement via son blog, et vous pourrez y retrouver une explication détaillée du modèle de sécurité adopté.
Le principe est d'assurer aux utilisateurs un contrôle précis des accès au profiling de leurs applis, même en production. Un couple d'identifiants client/serveur à signature cryptographique Ed25519 sécurise les échanges entre l'agent, le côté client (CLI ou extension Chrome), et les serveurs Blackfire. Ce système de signature réputé très rapide et très flexible tout en conservant un haut niveau de sécurité est notamment utilisé par OpenBSD.

Les « Teams », le moyen d'utiliser Blackfire dans votre entreprise

La version de base de Blackfire est gratuite, et le restera. Cela vous donne accès à la majorité des fonctionnalités décrites ci-dessus.
Ceci étant, pour aller plus loin, et en conséquence du modèle de sécurité que nous venons de voir, vous aurez vite besoin de permettre à plusieurs de vos collègues d'aller profiler le serveur sur lequel vous venez d'installer votre agent. En version gratuite, cela n'est pas possible.
La version payante vous donne accès à la fonctionnalité de gestion d'équipes (les « Teams »). Pour chaque équipe, vous avez des identifiants supplémentaires, que vous pourrez configurer à volonté sur une ou plusieurs machines pour profiler une ou plusieurs applications. Vous pourrez ensuite ajouter des membres à cette équipe, leur permettant ainsi de lancer des requêtes à leur tour sur les serveurs configurés Fig.5. L'ensemble des profils générés par le biais d'une équipe sont historiés dans un espace commun, auquel tout membre de l'équipe peut accéder à tout moment. Facile donc d'aller partager les informations récoltées avec un collègue.

Quel avenir pour Blackfire ?

Les concepts de Continuous Integration et Continuous Delivery ouvrent les prémisses d'un marché où des méthodologies répandues telles qu'Agile poussent à l'automatisation des tests. Chacun a conscience qu'il devrait y accorder plus d'attention, mais il reste difficile de percevoir le retour sur investissement, clé de l'approbation du budget et du lancement du projet. Blackfire va enfin vous permettre d'optimiser la performance de vos applications, en continu, quel que soit l'état de vos projets de mise en place de CI. Blackfire évolue en effet très rapidement : des évolutions sont proposées dans le cadre d'un cycle de release de 15 jours. Les prochaines étapes vont nous permettre de délivrer la promesse d'utilisation de Blackfire dans un contexte d'intégration continue, pour tous. Vous pourrez :

- Créer des scénarios de test ;
- Lancer les tests par le biais de planification ou sur la base d'évènements provenant d'outils externes (Jenkins, Travis,...) ;
- Être notifiés des résultats ou alertes dans vos outils actuels (GitHub, Bitbucket, Slack, e-mail...).

Pour résumer, le plus simple est d'aller essayer par vous-même, d'autant que l'édition Hack est et restera gratuite. Blackfire est un outil dont la vision est de faciliter la vie des développeurs, tout en répondant à des problématiques business fortes. ⊞

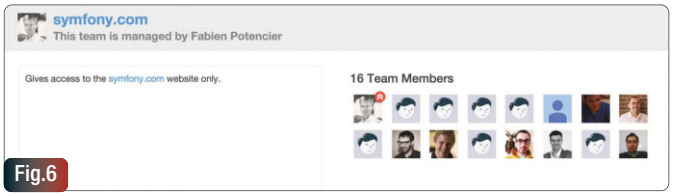


Fig.6

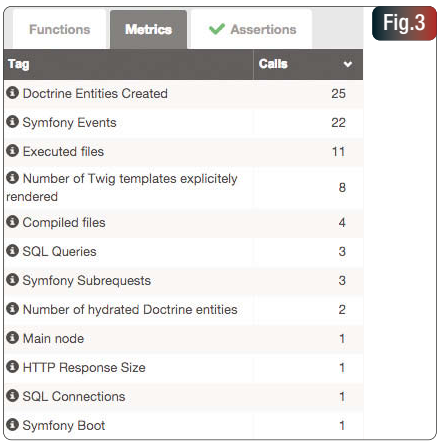


Fig.3

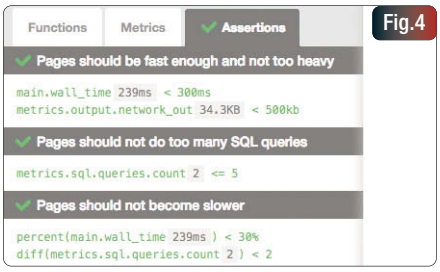


Fig.4

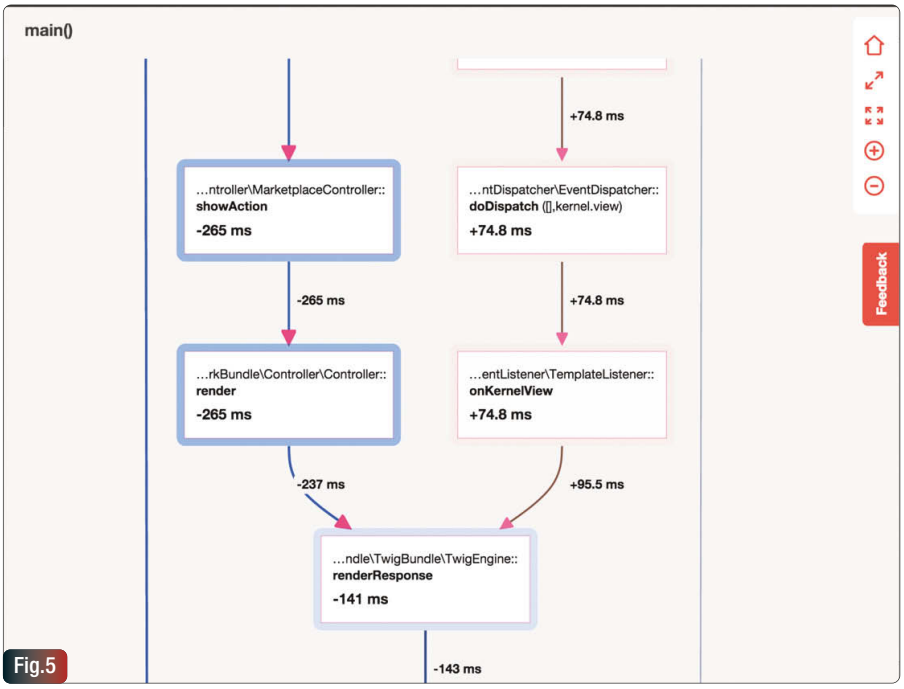


Fig.5

Amazon Mobile Analytics : mieux connaître ses utilisateurs pour améliorer son application !

Ces dernières années, le nombre d'applications disponibles a augmenté de manière exponentielle et les utilisateurs de smartphones et de tablettes en ont téléchargé de plus en plus sur leurs terminaux. La problématique qui se pose alors pour ceux qui proposent ces applications est la rétention de ces utilisateurs de plus en plus sollicités. En effet, pour les retenir et afin qu'ils reviennent sur l'application, il faut pouvoir leur proposer le contenu qui corresponde à leurs attentes. Ainsi, la connaissance du comportement des utilisateurs est aujourd'hui devenue essentielle car c'est fort de cette connaissance que vous serez en mesure d'adapter votre application pour qu'elle devienne incontournable.

Michael Garcia,
Solutions Architect AWS

Le Cloud Computing permet de récolter et de traiter rapidement les données de vos applications. Il est maintenant possible d'analyser le comportement des utilisateurs en fonction des segments tels que l'emplacement, la langue, la version de l'application ou de la plateforme, ainsi que le fabricant ou le terminal utilisé. Dans cet article, nous allons voir comment le service Amazon Mobile Analytics, disponible sur le Cloud Amazon Web Services (AWS), répond à vos besoins d'information et vous permet d'optimiser votre application en fonction des données collectées.

Les principales composantes du service Amazon Mobile Analytics

Le service Amazon Mobile Analytics bénéficie d'une grande élasticité, ce qui permet aux applications mobiles d'assurer la continuité de la capture des informations d'usage, en cas de croissance forte ou de pics importants.

De plus, l'analyse automatique des données se fait en 60 minutes ce qui vous permettra de prendre des décisions et de pouvoir réagir rapidement à un évènnement donné. Cette réactivité vous permettra de mieux répondre aux attentes de vos utilisateurs, d'augmenter ainsi leur satisfaction et cela leur donnera l'envie de revenir sur votre application. Avec Amazon Mobile Analytics vous êtes le seul maître des données collectées et vous avez la garantie qu'elles ne sont ni utilisées, ni partagées avec des tiers. Amazon Web Services vous permet de contrôler la confidentialité des données collectées, ce qui est un point critique pour vos utilisateurs.

Nous allons maintenant vous montrer comment faire usage d'Amazon Mobile Analytics, étape par étape.

Configuration du service Amazon Mobile Analytics

Pour commencer, nous allons utiliser la console Web AWS (https://console.aws.amazon.com/mobileanalytics/) afin d'accéder au service Amazon Mobile Analytics et pouvoir le configurer.

Une fois authentifié, vous pouvez accéder de manière gratuite à une démonstration sur la page du service, n'hésitez pas à vous en servir pour visualiser ce qu'il pourra vous apporter.

La première étape consiste à créer une application Amazon Mobile Analytics. Pour cela, il vous suffit de renseigner le nom de votre application (Fig.1).

Votre application Amazon Mobile Analytics est maintenant créée ! Pour faire le lien entre l'application Amazon Mobile Analytics et votre propre application, il faut intégrer un des différents SDK AWS (iOS, Android, Unity, Javascript) ; votre propre application sera ainsi capable d'envoyer des informations au service Amazon Mobile Analytics.

Pour cela suivez les instructions présentes à l'écran (Fig.2). N'oubliez pas de noter votre « App ID » affiché à l'écran car nous en aurons besoin plus loin lors de l'utilisation du SDK.

Dans la suite de cet article nous allons nous concentrer sur la partie mobile (iOS, Android), cependant il est tout à fait possible d'utiliser le service Amazon Mobile Analytics dans un contexte plus large comme pour une application Web avec le SDK Javascript ou dans un jeu vidéo avec le SDK Unity.

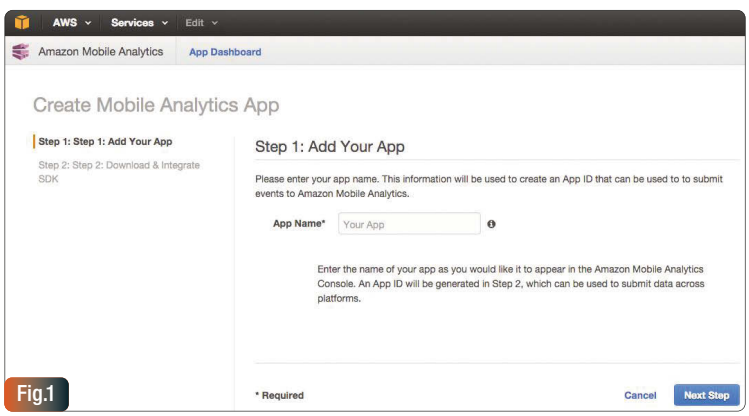


Fig.1

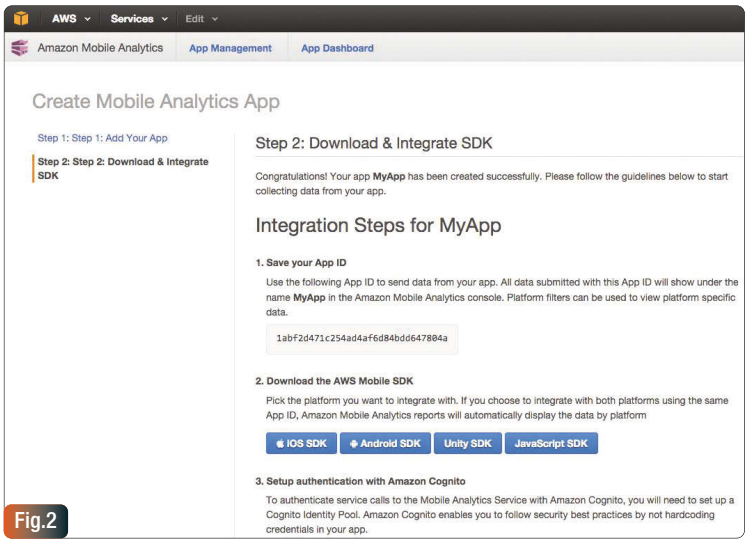


Fig.2

Intégration du SDK dans votre application mobile

Une fois le SDK téléchargé, vous pouvez continuer à suivre les instructions s'affichant à l'écran en fonction de la plateforme que vous avez retenue. Détaillons ensemble les parties de code les plus intéressantes qui permettent de faire usage d'Amazon Mobile Analytics sur iOS et Android. Le code complet à utiliser pour votre implémentation réelle est disponible directement dans la console Web AWS comme mentionné plus haut.

iOS

```
//**ETAPE 1**
//Initialisation du service Amazon Mobile Analytics
AWSMobileAnalytics* analytics = [AWSMobileAnalytics
  defaultAWSMobileAnalyticsWithIdentifier:@"someuniqueid"];

//**ETAPE 2**
//Creation d'un évènement personnalisé et envoi au service
id<AWSMobileAnalyticsEventClient> eventClient = analytics.eventClient;
id<AWSMobileAnalyticsEvent> someEvent = [eventClient
createEventWithEventType:@"someEvent"];
[eventClient submitEvents];
```

Android

```
//**ETAPE 1**
//Initialisation du service Amazon Mobile Analytics
private static MobileAnalyticsManager analytics;
analytics = MobileAnalyticsManager.getOrCreateInstance(this.getApplicationContext(),
  "yourCompany.yourAppId", Regions.YOUR_REGION, config, cognitoProvider );

//**ETAPE 2**
//Creation d'un évènement personnalisé et envoi au service
EventClient eventClient = analytics.getEventClient();
MobileAnalyticsEvent myEvent = eventClient.createEvent("myCustomEvent");
eventClient.recordEvent(myEvent);

//**ETAPE 3**
//Surcharge des méthodes onPause() et onResume() pour enregistrer les évènements
de //session.
analytics.getSessionClient().resumeSession();
analytics.getSessionClient().pauseSession();
analytics.getEventClient().submitEvents();
```

A noter : les étapes 1 et 2 sont identiques pour iOS et Android.

Etape 1 : Quelle que soit la plateforme utilisée, on retrouve un principe d'utilisation similaire, à savoir l'instanciation d'un objet représentant le service Amazon Mobile Analytics. Cette instanciation se base sur l'« App ID » que vous avez notée précédemment lors de la création de votre application Amazon Mobile Analytics.

Etape 2 : Les événnements personnalisés sont très utiles pour modéliser un événnement métier dans votre application. Par exemple dans le cadre d'une application de streaming de musique, il est intéressant de noter le nombre de chansons écoutées par l'utilisateur dans une session. Dans ce cas précis il faudrait placer le code qui gère la création d'événnements personnalisés au moment où la lecture d'une musique est déclenchée dans l'application.

Etape 3 : Cette étape concerne uniquement la plateforme Android. Afin qu'Amazon Mobile Analytics puisse matérialiser une session utilisateur, il est nécessaire que cette portion de code soit appelée dès que l'application ne se trouve plus au premier plan, ou qu'elle y revienne ; d'où la surcharge des méthodes onPause() et onResume().

Suivez l'usage de votre application avec le tableau de bord

Vous pouvez maintenant retourner dans la console Web AWS pour consulter le tableau de bord Amazon Mobile Analytics afin de pouvoir récupérer des informations sur les 5 axes suivants présents dans la console sous forme d'onglet (Fig.3) :

- Utilisateurs
- Sessions
- Monétisation
- Rétention
- Métriques personnalisées

N'hésitez pas à vous servir des filtres temporels pour contrôler la fenêtre de temps sur laquelle porteront ces différents graphiques. Vous pouvez également facilement télécharger les points de données présents dans un graphique au format .csv .

Aller plus loin dans l'analyse de vos données avec Amazon Redshift

Pour ceux qui souhaitent aller plus loin dans l'analyse des données collectées par Amazon Mobile Analytics, deux choix s'offrent à vous grâce à la fonctionnalité d'auto-export.

Vous pouvez récupérer les données collectées sur Amazon S3 afin de les exploiter dans un système tiers, ou utiliser le service de datawarehouse performant Amazon Redshift (Fig.4).

L'élasticité et la très grande capacité d'Amazon S3 et d'Amazon Redshift

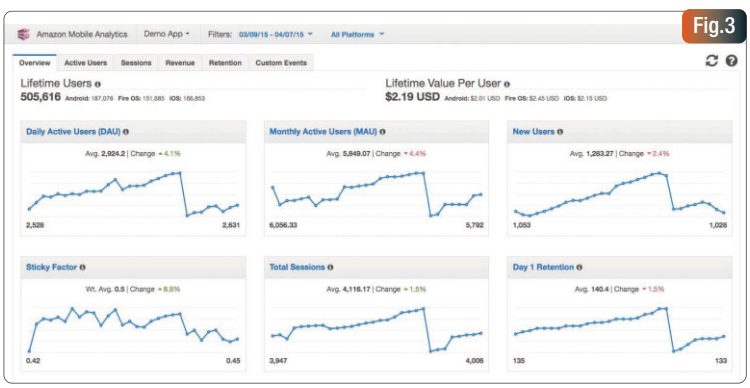


Fig.3

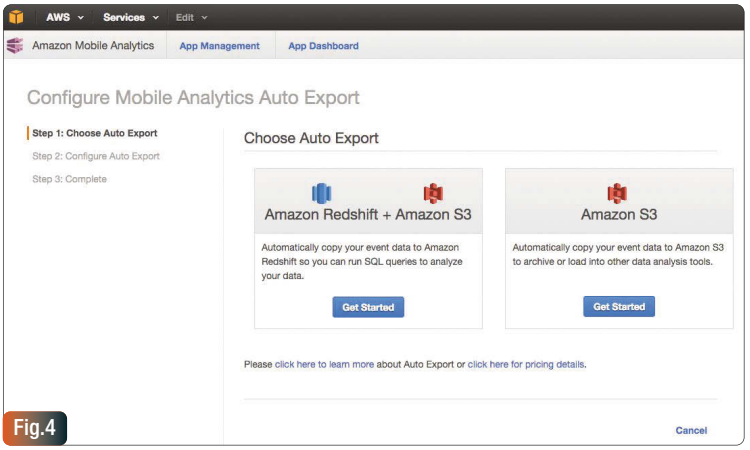
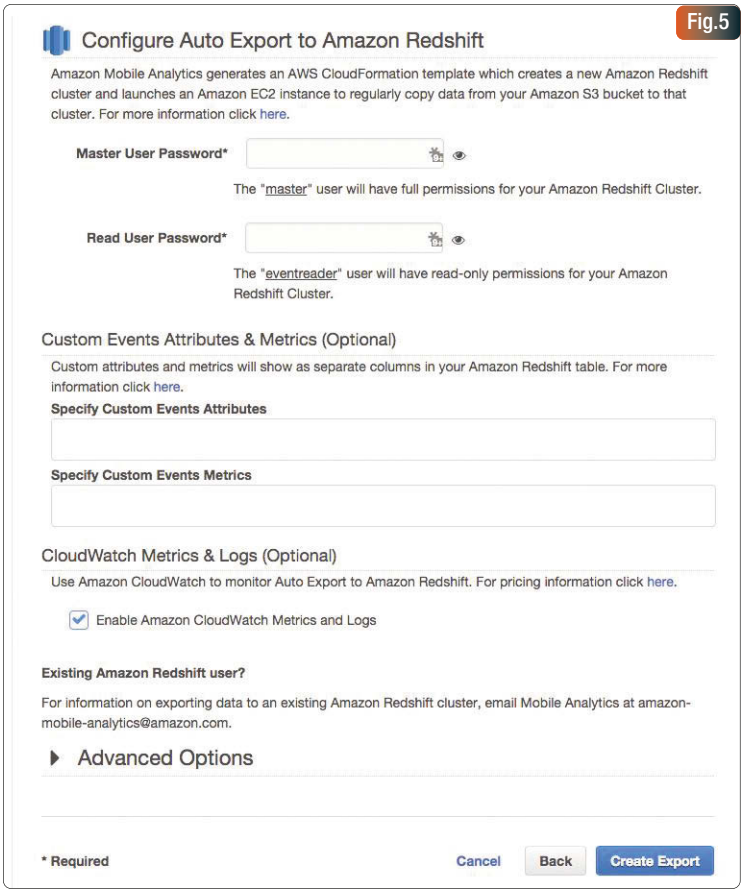


Fig.4

vous permettra de stocker une quantité de données importante, et de pouvoir les analyser de manière fine et performante. En utilisant Amazon Redshift pour analyser les données collectées par Amazon Mobile Analytics, vous pourrez récupérer des informations spécifiques à votre besoin à l'aide de requêtes SQL ou d'une solution commerciale de visualisation de données (http://aws.amazon.com/redshift/partners/).
Dans ce cas-ci, la fonctionnalité d'auto-export va d'une part exporter les données contenues dans Amazon Mobile Analytics vers Amazon S3, et va d'autre part lancer et configurer pour vous un cluster Amazon Redshift ainsi qu'une instance Amazon EC2 qui sera utilisée pour charger les données d'Amazon S3 vers Amazon Redshift de manière régulière. Ces éléments sont installés dans votre compte AWS, c'est pourquoi vous devez renseigner leurs configurations dans l'étape N°2 (Fig.5).

Une fois les éléments installés et les données chargées, vous pourrez créer des requêtes SQL vous permettant de récupérer des informations plus précises. A titre d'exemple, voici une requête vous permettant de récupérer le nombre d'utilisateurs et d'appareils actifs sur les 30 derniers jours, classés par version de l'application :
Utilisateurs actifs et appareils sur les 30 derniers jours, par version d'application.

Fig.5



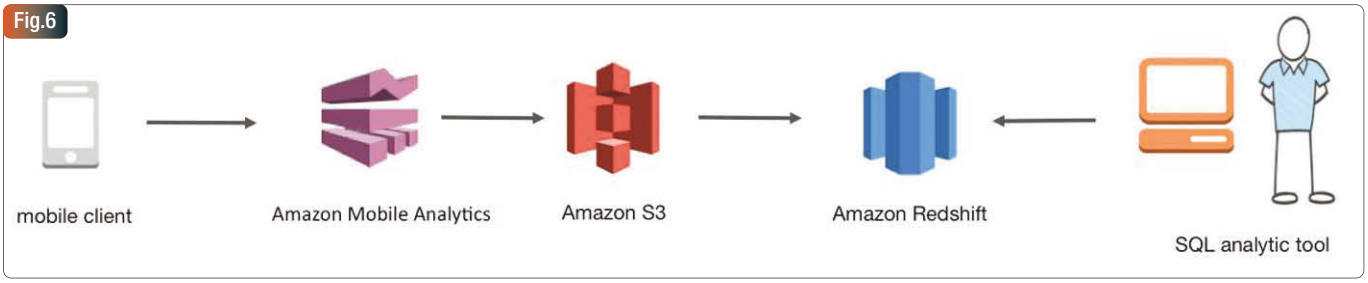

```
SELECT
  application_app_id AS "app id",
  device_platform_name AS "platform",
  application_version_name AS "version name",
  application_version_code AS "version code",
  COUNT(DISTINCT client_id) AS "devices",
  COUNT(DISTINCT client_cognito_id) AS "users"
FROM
  AWSMA.v_event
WHERE
  event_type = '_session.start' AND
  event_timestamp BETWEEN getdate() - 30 AND getdate() + 1
GROUP BY
  "app id",
  "platform",
  "version name",
  "version code"
ORDER BY
  "app id" ASC,
  "platform" ASC,
  "devices" DESC,
  "version name" DESC,
  "version code" DESC
;
```

Pour les personnes ne souhaitant pas écrire de requête SQL : vous pouvez passer par un outil partenaire comme mentionné plus haut qui vous permettra d'effectuer des requêtes sur Amazon Redshift de manière graphique. Voici un schéma simplifié de ce type d'intégration (Fig.6).
N'hésitez pas à consulter le Blog officiel Mobile pour en apprendre plus sur ce sujet (http://mobile.awsblog.com/).

Conclusion

Amazon Mobile Analytics vous permet de mieux appréhender le comportement de vos utilisateurs de manière rapide, avec un effort de développement réduit tout en gardant la maîtrise de vos données.
De plus, l'intégration native avec Redshift vous donnera la possibilité de mettre en exergue des tendances, et d'aller plus loin dans l'analyse de vos données.
En tant que développeur d'application, le plus important est de savoir écouter vos utilisateurs afin de pouvoir améliorer votre application et connaître l'impact réel de vos développements sur la popularité de votre application ainsi que sa monétisation.
N'attendez plus et testez dès aujourd'hui le service.

Fig.6



MEAN.IO (2/4)

Ce second article d'une série consacrée à MEAN.IO va nous permettre d'introduire les principaux concepts permettant de créer un nouveau module afin d'étendre le canevas. Le cas d'application que je vous propose d'étudier permettra également à la plupart d'entre vous de découvrir un autre domaine, celui de la visualisation de données géographiques sous forme de carte ou sous forme 3D. Nous souhaitons en effet créer une application permettant de visualiser un ou plusieurs itinéraires GPS (type randonnée VTT ou pédestre).

Luc CLAUSTRES
Consultant indépendant
Créateur d'Applications Numériques
http://www.digital-innovation.fr

Création d'un module

Tout d'abord un petit rappel, l'ensemble des modules d'une application MEAN.IO est stocké dans le dossier **packages**. Le premier sous-dossier **core** inclut les modules de base livrés avec le framework et le sous-dossier **custom** les modules spécifiques à la logique applicative, c'est donc ici qu'il faudra créer vos nouveaux modules. Grâce à l'outil en ligne de commande mean-cli, il est possible d'initialiser un module (scaffolding) via :

```
mean package package_name
```

Il est à tout moment possible de lister vos modules et d'effacer un module en particulier :

```
// Liste les modules présents
mean list
// Efface un module
mean uninstall package_name
```

La structure classique d'un module MEAN.IO vous a été présentée dans l'article précédent (un module nouvellement créé peut ne pas contenir tous les dossiers néanmoins).
A la racine on trouvera comme dans le cas de l'application les fichiers de configuration pour npm, bower et MEAN.IO (**mean.json**). Le plus important est le fichier **app.js** qui est le point d'entrée du module, à l'intérieur est réalisé l'enregistrement du module dans MEAN.IO. Créons un nouveau module nommé 'application' qui contiendra notre logique applicative, dans le fichier **app.js** généré par défaut nous trouverons ceci :

```
var Module = require('meanio').Module;
// Création du module, ceci va automatiquement charger tous les modèles du sous-dossier models
var Application = new Module('application');
// Enregistrement du module
Application.register(function(app, auth, database) {
 // Déclaration automatique des routes du sous-dossier routes
 Application.routes(app, auth, database);
 // Agrégation des dépendances front-end (voir article précédent)
 Application.aggregateAsset('css', 'application.css');

 return Application;
});
```

Si votre nouveau module dépend d'autres modules MEAN.IO ou AngularJS il vous faudra rajouter ceci lors du register :

```
Application.angularDependencies(['mean.users']);
```

En effet l'arbre des dépendances est construit côté serveur au lancement de l'application et est récupéré au chargement de la page via une requête sur l'URL */_getModules* côté client. Ceci permet de déclarer alors dynamiquement la liste de tous les modules AngularJS dans le code JS (pour les curieux, voir le fichier **bower_components/web-bootstrap/index.js**).
Par défaut MEAN.IO injecte trois dépendances en paramètre du register :

- **app** : l'application Express ;
- **auth** : un module proposant des fonctions d'authentification basiques ;
- **database** : la connexion Mongoose à la base de données.

Si votre module dépend d'autres modules (e.g. module1 et module2) ils pourront être injectés à la suite :

```
...
// Enregistrement du module
Application.register(function(app, auth, database, module1, module2) {
 ...
});
```

Partie serveur (back-end)

Création du modèle

Le premier travail lors de la création d'un module consiste à définir le modèle conceptuel du ou des objets qui seront manipulés par le module, ce modèle est ensuite exprimé au moyen d'un schéma Mongoose. Dans notre cas l'objet manipulé sera un itinéraire GPS ('track' en anglais). Un tel chemin est simplement décrit par une liste de positions GPS acquises par le capteur. Chaque point est repéré en coordonnées géographiques : longitude, latitude et altitude. A part l'utilisateur qui l'a créé et un descriptif, le chemin contiendra donc un tableau de coordonnées. Pour déclarer le modèle il suffit de créer un fichier **TrackModel.js** dans le dossier **models** du module, il contiendra le schéma suivant :

```
var mongoose = require('mongoose'),
  Schema = mongoose.Schema;
// Déclaration du schéma du modèle 'Track'
var TrackSchema = new mongoose.Schema({
 // Utilisateur ayant créé le chemin
 user : {
  type: mongoose.Schema.Types.ObjectId,
  ref: 'User' // Référence le modèle d'utilisateur de MEAN.IO
 },
 // Date de création
 created: {
  type: Date,
```

```
  default: Date.now
 },
 // Titre du chemin
 title: {
  type: String,
  required: true,
  trim: true
 },
 // Description associée au chemin
 description : {
  type : String, required : true
 },
 // Liste des points de passage constituant le chemin (coordonnées géographiques)
 waypoints : {
  type : [Number], required : false
 }
});
// Méthode utilisée pour récupérer un chemin via son ID,
// va récupérer certaines des informations de l'utilisateur via populate
TrackSchema.statics.getById = function(id, cb) {
 this.findOne({
  _id: id
 }).populate('user', 'name username').exec(cb);
};
mongoose.model('Track', TrackSchema);
```

Création des routes

Une fois le modèle créé, la seconde étape consiste à définir l'API REST qui permettra de le manipuler via les classiques opérations CRUD (création, lecture, mise à jour et destruction) et éventuellement d'autres opérations plus spécifiques à votre modèle. Une route consiste à associer un point d'entrée de l'API (i.e. une URL avec une méthode HTTP) à la fonction de traitement JavaScript associée au sein d'un contrôleur. Pour déclarer les routes il suffit donc de créer un fichier **TrackRoutes.js** dans le dossier **routes** du module, il contiendra le code suivant :

```
// Récupération du contrôleur
var track = require('../controllers/TrackController');
// Fonction pour vérifier les droits d'accès à l'API
var hasAuthorization = function(req, res, next) {
 // Un utilisateur ne peut modifier que ses propres chemins
 if (!req.track.user._id.equals(req.user._id)) {
  return res.status(401).send('User is not authorized');
 }
 next();
};
module.exports = function(Application, app, auth, database) {
  // Déclaration des routes
  app.route('/api/track')
    .get(auth.requiresLogin, track.list)
    .post(auth.requiresLogin, track.create);
  app.route('/api/track/count')
    .get(auth.requiresLogin, track.count);
  app.route('/api/track/:trackId')
    .get(auth.isMongoId, auth.requiresLogin, track.get)
    .put(auth.isMongoId, auth.requiresLogin, hasAuthorization, track.update)
    .delete(auth.isMongoId, auth.requiresLogin, hasAuthorization, track.destroy);

  app.param('trackId', track.findById);
};
```

On note l'ajout d'une fonction d'autorisation (middleware Express) spécifique à notre modèle car seul l'utilisateur qui a créé un chemin est autorisé à le modifier. On notera également l'appel à des fonctions de vérification d'authentification de l'utilisateur ou de validité de l'identifiant d'un chemin qui sont fournies par MEAN.IO.

Test de l'API

Afin de tester l'API j'utilise l'extension Chrome Postman. Cet outil permet d'envoyer des requêtes HTTP selon tout type de méthode (GET, POST, PUT, DELETE, etc.) et de créer des collections de templates de requête afin de pouvoir faire des tests rapides sur une API de type REST. Pour pouvoir l'utiliser avec MEAN.IO il faut tout d'abord récupérer un token JWT (voir article précédent) qui devra être adjoint à toutes les requêtes faites vers l'API. Pour cela le plus simple est de créer une requête de type POST vers l'URL */api/login* avec une charge utile (i.e. un body) contenant l'e-mail et le mot de passe de l'utilisateur de l'application préalablement créé (**Fig.1**).

La réponse renvoyée contient votre token (**Fig.2**). Il faudra ensuite introduire ce token dans le header de vos prochaines requêtes en respectant le standard Bearer.

Ainsi la requête GET vers l'URL */api/track* à la **Fig.3** va donner une réponse de status 200 ('OK') avec la charge utile [] (puisque nous n'avons créé aucun chemin à ce stade). Sans le token une réponse de status 401 'User is not authorized' aurait été obtenue. Vous pouvez essayer de créer un chemin à la main en fournissant simplement un descriptif dans le corps de

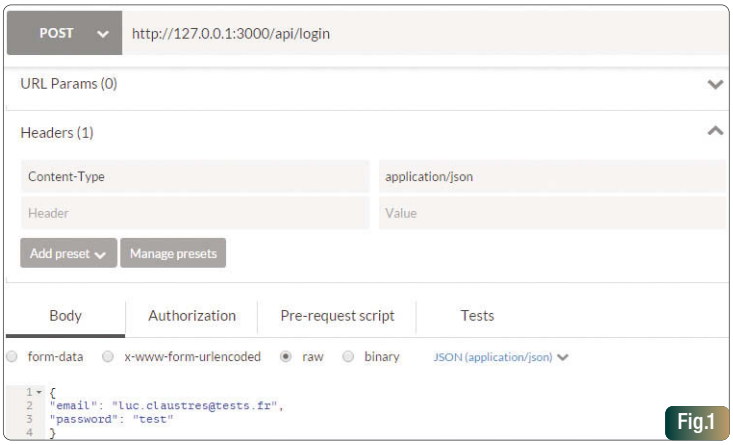


Fig.1

Fig.3

Fig.2

votre requête POST sur */api/track* puisque l'utilisateur sera automatiquement renseigné et que les coordonnées ne sont pas requises pour l'exemple.

Création du menu

Pour rajouter des entrées dans la barre de menu de MEAN.IO il suffit de les déclarer dans votre **app.js** en précisant pour chacune le titre, les rôles autorisés à la voir et l'état à activer côté client (voir ci-après dans la partie cliente). Dans notre cas nous aurons simplement deux entrées, une pour lister tous nos chemins et une pour créer un nouveau chemin :

```
// Ajout des entrées de menu pour les utilisateurs authentifiés
Application.menus.add({
  'roles': ['authenticated'],
  'title': 'Tracks',
  'link': 'list tracks'
});
Application.menus.add({
  'roles': ['authenticated'],
  'title': 'New Track',
  'link': 'create track'
});
```

Partie cliente (front-end)

Service

La partie service est réellement la plus simple étant donné que le service $resource d'AngularJS est spécifiquement fait pour encapsuler une API de type REST. Pour déclarer notre service d'accès à l'API des chemins il suffit de créer un fichier **TrackService.js** dans le dossier **services** de la partie publique, il contiendra la déclaration suivante :

```
// Service utilisé pour accéder à l'API REST des chemins
angular.module('mean.application').factory('TrackService', ['$resource',
 function($resource) {
  return $resource('api/track/:trackId', {
   trackId: '@_id'
  }, {
   update: {
    method: 'PUT'
   }
  });
 }
]);
```

Routes

Les routes côté front-end sont gérées via l'AngularUI Router. Ce module permet d'organiser la navigation sous la forme d'une machine à état. Dans la version simple à chaque état est associé une URL, une vue, et un contrôleur. Dans la version plus complexe il est possible d'imbriquer les machines à état. Pour déclarer les routes il suffit de créer un fichier **ApplicationRoutes.js** dans le dossier **routes** de la partie publique, il contiendra dans notre cas les déclarations permettant d'accéder aux pages pour lister nos chemins, créer un nouveau chemin, éditer un chemin et le visualiser (via son identifiant de base de données) :

```
// Definition des routes pour les chemins
angular.module('mean.application').config(['$stateProvider',
 function($stateProvider) {
  // Page listant tous les chemins de l'utilisateur
  $stateProvider
   .state('list tracks', {
    url: '/tracks',
    templateUrl: '/application/views/ListTracks.html',
    controller: 'TrackController',
    resolve: {
     loggedin: function(MeanUser) {
      return MeanUser.checkLoggedin();
     }
    }
   })
   ...
   // Page permettant de voir un chemin
   .state('view track', {
    url: '/track/:trackId',
    templateUrl: '/application/views/ViewTrack.html',
    controller: 'TrackController',
    resolve: {
     loggedin: function(MeanUser) {
      return MeanUser.checkLoggedin();
     }
    }
   });
 }
]);
```

Contrôleur

Notre contrôleur sera un contrôleur AngularJS classique mais nous allons faire en sorte qu'il gère les actions nécessaires à tous les états se rapportant aux chemins pour centraliser le comportement par type d'objet manipulé (dans notre cas il n'y a pour l'instant qu'un seul type d'objet). La méthode `find` récupère côté serveur la liste des chemins existants et `findOne` celui dont l'ID est présente dans l'URL. La méthode `create` est appelée lors de la soumission du formulaire de création d'un chemin, qui a préalablement rempli l'objet `track` du scope. De même pour la méthode `edit`, la seule différence étant que l'objet `track` est dans ce cas préalablement récupéré sur le serveur puis mis à jour. Enfin, la méthode `remove` détruit le chemin fourni en paramètre ou sinon celui en cours d'édition. Toutes ces méthodes seront appelées par les différentes vues de l'application.

```
// Contrôleur utilisé pour gérer les chemins
angular.module('mean.application').controller('TrackController', ['$scope', '$stateParams', '$location',
 'TrackService',
 function($scope, $stateParams, $location, TrackService) {
  // Objet par défaut pour le mode création
  $scope.track = { };
  // Crée un nouveau chemin
  $scope.create = function(isValid) {
   if (isValid) {
    var payload = {
     title: $scope.track.title,
     description: $scope.track.description
    };
    var track = new TrackService(payload);
    track.$save(function(response) {
     $location.path('tracks');
    });
   }
  };
```

```
  // Détruit un chemin existant
  $scope.remove = function(track) {
    track.$remove(function(response) {
      for (var i in $scope.tracks) {
        if ($scope.tracks[i] === track) {
          $scope.tracks.splice(i,1);
        }
      }
      $location.path('tracks');
    });
  };
  // Met à jour un chemin existant
  $scope.update = function(isValid) {
   if (isValid) {
    var track = $scope.track;
    track.$update(function() {
     $location.path('track/' + track._id);
    });
   }
  };
  // Liste tous les chemins de l'utilisateur
  $scope.find = function() {
   TrackService.query(function(tracks) {
    $scope.tracks = tracks;
   });
  };
  //Récupère un chemin via son ID
  $scope.findOne = function() {
   TrackService.get({
    trackId: $stateParams.trackId
   }, function(track) {
    $scope.track = track;
   });
  };
 }
]);
```

Vues

Concernant les vues nous avons tout d'abord besoin d'une présentation sous forme de liste de tous les chemins existants. Pour ce faire nous utiliserons des panels Bootstrap. L'en-tête (classe CSS *panel-heading*) contiendra le nom de notre chemin, le pied (classe CSS *panel-footer*) contiendra la date de création et le nom de l'auteur, et le corps (classe CSS *panel-body*) contiendra sa description. Ainsi nous créerons une page permettant d'afficher le contenu d'un chemin qui sera utilisée (via un ng-include) dans la vue listant tous les chemins (via un ng-repeat), mais aussi dans celle permettant d'afficher les informations d'un chemin en particulier connaissant son ID. Pour la création et l'édition la démarche est identique : nous créerons un formulaire permettant d'éditer le contenu d'un chemin qui sera utilisé (via un ng-include) dans la vue de création d'un chemin, mais aussi dans celle permettant d'éditer les informations d'un chemin en particulier connaissant son ID. La page de visualisation d'un chemin (**Fig.4**) créé dans **public/views/Track.html** se présente ainsi :

```
<!-- Nom du chemin -->
<div class="panel panel-default">
 <!-- Un clic sur l'en-tête provoque l'ouverture du corps du panneau -->
 <div class="text-center panel-heading" ng-click="track.isOpen = !track.isOpen">{{track.title}}
  <!-- Actions associées au chemin (delete/edit) -->
  <a data-ng-click="remove(track); $event.stopPropagation();"><i class="pull-right
glyphicon glyphicon-trash" tooltip="Remove track" tooltip-trigger="mouseenter" tooltip
-placement="top"> </i></a>
  <a href="/track/{{track._id}}/edit"><i class="pull-right glyphicon glyphicon-edit"
 tooltip="Edit  track"  tooltip-trigger="mouseenter"  tooltip-placement="top"> 
</i></a>
 </div>
 <!-- Description du chemin -->
 <div collapse="!track.isOpen">
  <div class="panel-body">
   {{track.description}}
  </div>
  <!-- Date/auteur du chemin -->
  <div class="panel-footer">
   <em >Created {{track.created | date:'medium'}} by {{track.user.name}}</em>
  </div>
 </div>
</div>
```

Le formulaire d'édition d'un chemin (**Fig.5**) créé dans **public/views/Track Editor.html** est un formulaire standard contenant un champ de type file afin de sélectionner un fichier sur le système de l'utilisateur.

Conclusion

Cet article a été l'occasion d'approfondir votre connaissance du framework MEAN.IO qui vous avait été présenté succinctement lors du précédent article. Tout d'abord par la mise en place d'un nouveau module, ensuite par la création d'un modèle de données et de l'API REST permettant de le manipuler, enfin par une partie cliente incluant des interfaces homme-machine (IHM) pour la présentation et l'édition des données. Lors du prochain épisode nous terminerons en beauté avec la visualisation de nos chemins sous formes de cartes à la façon "Google Maps" ou de vues 3D à la façon "Google Earth". ⊞

Fig.4

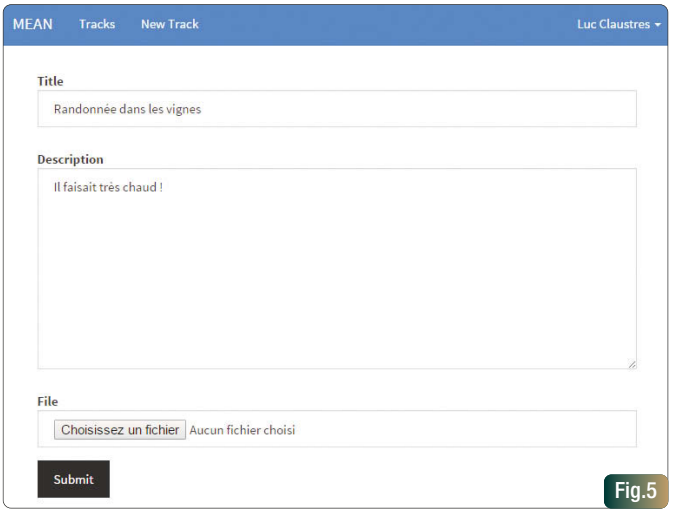


Fig.5

ESP8266 : le microcontrôleur connecté par Wifi pour 2€ au potentiel phénoménal.

Arrivé à l'été 2014, l'ESP8266 est un microcontrôleur produit par la société chinoise Espressif intégrant un module Wifi. Avec un prix avoisinant les 2€, il a très vite séduit les foules qui se sont empressées de créer une communauté dans laquelle on retrouve maintenant beaucoup de documentation, de bouts de code, des firmwares et projets en tout genre.

Sébastien Warin
Creative Technologist @ Publicis ETO
http://sebastien.warin.fr

Initialement cette puce était un « simple » module Wifi permettant à un microcontrôleur équipé d'une liaison série, tel un Arduino, de disposer d'une connectivité Wifi pour un prix dérisoire. Mais en Octobre 2014, Espressif publiait un kit de développement (SDK) permettant de reprogrammer la puce supprimant ainsi la nécessité d'un microcontrôleur séparé. Comme nous allons le découvrir dans cet article, les possibilités sont immenses !

Les caractéristiques

Ce microcontrôleur est cadencé à 80Mhz par un processeur 32bits RISC avec 96K de RAM et une mémoire flash de 512Ko à 4Mo selon les modèles. Il dispose d'une connectivité Wifi 802.11 b/g/n supportant le WEP, WPA/2/WPS et réseau ouvert et 16 GPIO dont le support du SPI, I²C, UART et un port ADC (pour les I/O analogiques).

Le module Wifi peut être également utilisé en tant que « soft-AP », c'est-à-dire configuré en tant que point d'accès Wifi. Il est alimenté en 3,3V et consomme entre 60mA et 200mA avec le Wifi activé **Fig.1**.

Il existe plusieurs modules d'ESP8266. Les plus connus sont ceux produits par AI-Thinkers, les ESP-XX : l'ESP-01 (le plus répandu), l'ESP-02, 03, ... jusqu'au 13 : **Fig.2**.

Gardez à l'esprit que tous ces modèles sont animés par le même contrôleur: l'ESP8266. Les différences sont principalement liées à la taille, au nombre de PIN « accessibles », à l'antenne (PCB, céramique ou socket RP-SMA), à la présence ou non d'un blindage électromagnétique, ou encore à la présence d'une LED. Pour ma part, je travaille essentiellement avec l'ESP-01 et l'ESP-07 **Fig.3**.

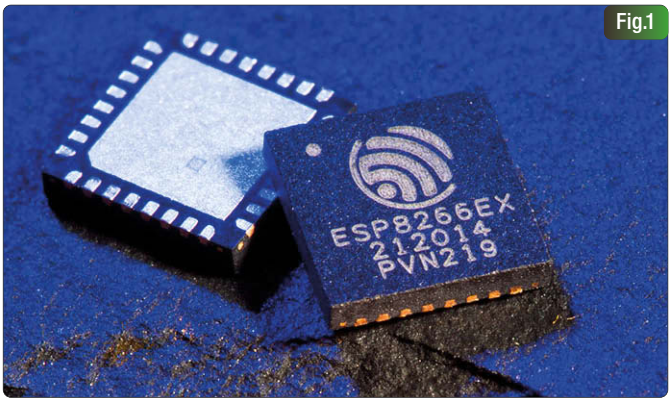


Fig.1

Figure 1 - ESP8266

L'ESP-01 à l'avantage d'avoir des pins sur un pas standard de 2,54mm très pratique pour le prototypage pour une dimension totale de 1,4cm sur 2,4cm avec une antenne intégrée sur la carte. L'ESP-07 quant à lui mesure 2cm x 1.6cm avec un blindage électromagnétique, une antenne céramique et un port RP-SMA pour y connecter une antenne externe et 10 GPIO disponibles, dont un port ADC pour les entrées analogiques

A noter qu'il existe d'autres sociétés créant des cartes animées par l'ESP8266 à l'image d'Olimex qui propose une carte de prototypage orientée « développement », la carte de SparkFun qui intègre un connecteur USB pour l'alimentation et un chargeur LiPo pour ajouter une batterie, sans oublier la ESP-WROOM-02 directement produite par Espressif

Fig.2

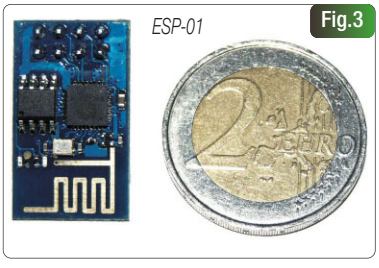
Différents modèles d'AI-Thinkers

Développer avec l'ESP

Avec la libération du SDK par Espressif fin 2014, sont apparues plusieurs façons de développer avec l'ESP8266. Flasher le firmware de l'ESP8266

Avant de démarrer nous allons aborder la façon de flasher le firmware de la puce pour la reprogrammer. Pour la suite de l'article, nous allons nous baser sur l'ESP-01, le plus simple pour prototyper **Fig.6**.

L'ESP-01 dispose de 8 PINs : deux pour l'alimentation en 3,3V (Vcc et Gnd), deux pour la connexion série (Tx/Rx), un « RST » (reset) pour réinitialiser la puce en le connectant à la masse, le « CH_PD » (chip power-down) qui doit être alimenté en 3,3V pour activer le Wifi et enfin deux GPIO pour vos I/O. Le GPIO a un double rôle : il sert également à entrer dans le « Flash



ESP-01

Fig.3

ESP-07

Fig.4

mode ». Pour cela, il faut le connecter à la masse. Ainsi l'ESP8266 ne démarrera pas son programme mais entrera dans le « UART download mode », c'est-à-dire qu'il écrira tout ce qu'il reçoit sur la liaison série dans sa mémoire. De ce fait, pour programmer la puce, il faut redémarrer l'ESP en « Flash mode » afin de pouvoir écrire un nouveau firmware.

Pour faciliter les opérations, je vous recommande de vous fabriquer une plaque de « programmation » : il s'agit de faire une carte avec un réceptacle dans lequel on pourra déposer notre ESP-01. Cette carte disposera de quelques headers pour y connecter l'interface FTDI/USB, une alimentation externe et deux boutons poussoirs pour respectivement faire un reset de la puce et l'autre pour démarrer en « Flash mode ». Cela vous évitera de vous perdre dans les câbles à chaque manipulation/programmation de l'ESP **Fig.7 et 8**.

Retrouvez tous les détails et le schéma de cette carte sur mon site Internet.

Pour flasher votre ESP, il existe plusieurs outils. Personnellement j'utilise par habitude « ESP Flasher Tool » : **Fig.9**.

Il faut sélectionner les binaires en précisant à quelles adresses ils doivent être écrits sur la mémoire de votre ESP. L'outil garde en mémoire les fichiers et adresses, il suffit donc de cocher/décocher les fichiers que l'on souhaite flasher, très pratique pour switcher entre différents firmwares.

Vous pouvez donc avec cet outil, et après avoir démarré votre ESP en « Flash mode », changer de firmware à souhait.

Le firmware original « AT »

Pour démarrer notre découverte des ESP, commençons par le firmware d'origine installé par Espressif.

L'idée est de se servir de l'ESP comme d'un simple module Wifi qu'on exploitera depuis un autre microcontrôleur comme un Arduino en liaison série. Une fois la communication série établie on dialogue avec l'ESP en utilisant des commandes Hayes, qu'on appelle aussi « commandes AT ».

Pour expérimenter ce firmware, vous pouvez utiliser Putty pour vous connecter directement sur la puce ou Esplorer : bien plus pratique !

Pour tester la connexion, envoyez la commande « AT ». Vous devriez recevoir « OK » de la part de l'ESP pour vous indiquer que la connexion série est opérationnelle. La commande « AT+GMR » renvoie les informations de version (ici le firmware AT 0.21 basé sur le SDK 0.9.5) :

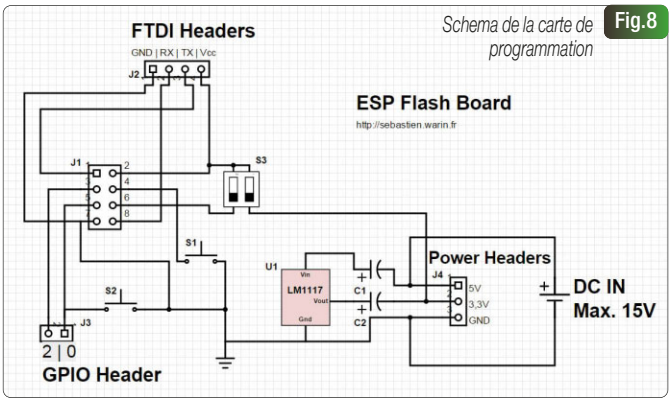
```
OK
AT+GMR
AT version:0.21.0.0
SDK version:0.9.5
OK
```

Vous pouvez ensuite configurer le mode Wifi : Station (client Wifi), AP (Access Point) ou les deux. Pour être client Wifi, entrez la commande « AT+CWMODE=1 ».

Puis, pour lister les réseaux Wifi, entrez « AT+CWLAP ». Vous obtiendrez en réponse la liste des AP avec le SSID, l'adresse MAC et le canal utilisé.

```
AT+CWMODE=1
OK

AT+CWLAP
+CWLAP:(4,"NUMERICABLE-7610",-80,"c0:d9:62:68:41:89",1)
+CWLAP:(3,"CARRE_NOIR",-73,"a8:9d:21:be:56:b6",6)
+CWLAP:(4,"Livebox-55c0",-85,"00:1d:6a:61:0b:ec",6)
+CWLAP:(3,"ETO-WEB2",-87,"a8:9d:21:43:ce:14",11)
OK
```

Pour se connecter, utilisez la commande « AT+CWJAP="MYSSID", "MYPASSWD" ». Si la commande retourne « OK » c'est que vous êtes connecté. Vous pouvez ensuite faire un « AT+CIFSR » pour connaître votre adresse IP. Une fois connecté, vous pouvez utiliser les commandes « AT+CIPSTART » et « AT+CIPSEND » pour initialiser et envoyer des don-



Schema de la carte de programmation **Fig.8**

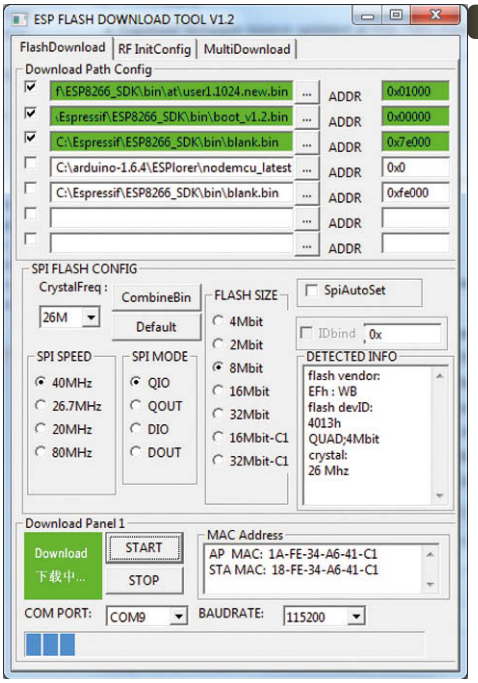
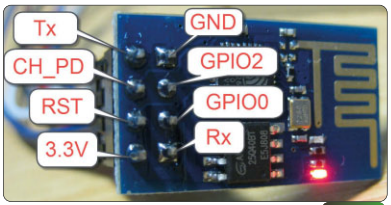


Fig.9

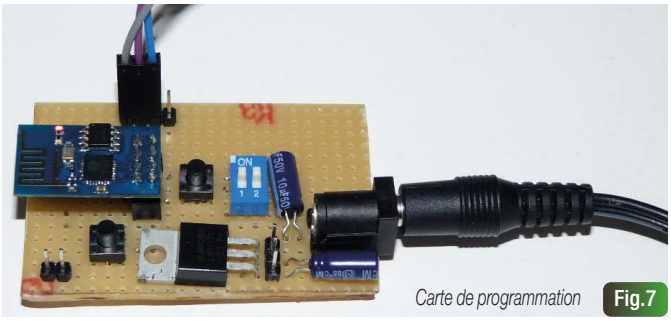
ESP Flash Download Tool

Olimex ESP8266 DEV

Fig.5



Connecteurs sur l'ESP-01 **Fig.6**



Carte de programmation **Fig.7**

nées dans une connexion TCP ou UDP. De ce fait, il devient possible de connecter n'importe quel contrôleur équipé d'une liaison série Fig.10.
Sur un Arduino, connectez votre ESP sur le port RX/TX (UART) de votre Arduino et alimentez l'ESP en 3,3V (sans oublier le CH_PD pour activer le Wifi). Attention, l'ESP8266 fonctionne sur une tension de 3,3V alors que la plupart des Arduino sont en 5V. Si vous utilisez la sortie 3,3V de votre Arduino n'oubliez pas de mettre un condensateur car cette sortie est généralement limitée à 50mA ou utilisez une alimentation externe. De plus, la liaison série (Rx/Tx) doit être également convertie en 3,3V (privilégiez l'usage d'un convertisseur de niveau). Une fois la liaison série opérationnelle, il vous suffira d'envoyer des commandes AT.

```
Serial1.begin(115200);
// Définir le mode Wifi "Station"
Serial1.println("AT+CWMODE=1");
// Se connecter au SSID avec le password
String cmd = "AT+CWJAP=\"" + SSID + "\",\"" + PASS + "\"";
Serial1.println(cmd);
delay(2000);

// On doit lire "OK"
if (Serial1.find("OK")) {
  Serial.println("OK, WiFi Connected.");

  delay(5000);
  // Afficher l'adresse IP
  Serial1.println("AT+CIFSR");
  Serial.println("ip address:");
  while (Serial1.available()) Serial.write(Serial1.read());
}
else {
  Serial.println("Can not connect to WiFi.");
}
```

Vous aurez compris le principe, libre à vous d'implémenter l'ensemble des commandes AT pour pouvoir connecter tout type de microcontrôleurs en exploitant la liaison série.

Développer son propre firmware avec le SDK Espressif

Jusqu'au mois d'octobre 2014 le firmware AT était la seule façon d'exploiter les ESP8266 avant que la société Espressif ouvre son SDK nous permettant de développer nos propres firmwares. Par ailleurs, le code du firmware AT a été ouvert en guise d'exemple.
Pour développer nativement, vous pouvez utiliser Eclipse ou Visual Studio. Personnellement j'utilise ce dernier correctement configuré avec le bon « tool chain » (MinGW, Extensa, PySerial pour flasher le firmware).
Il existe aussi des produits tels que VisualGDB, un plugin Visual Studio permettant de développer des firmwares pour plateformes embarquées, modules du kernel Linux, etc. En bénéficiant de toute la puissance de Visual Studio pour l'édition de code, le débogage et le déploiement. Et à ce titre, il permet de créer très facilement des projets pour ESP8266 Fig.11.
Il existe actuellement 3 SDK pour développer nativement sur les ESP8266: l' « ESP IoT SDK », l' « ESP IoT RTOS SDK » et l' « ESP Open RTOS ».
Le premier est fourni par Espressif, il est le plus utilisé. C'est le SDK « principal » qui ne repose pas sur un OS, on l'appelle le « non-OS SDK ». Il est mono-thread bien que vous puissiez utiliser un timer pour orchestrer des « tâches » (sans que celles-ci occupent trop le CPU sous peine d'avoir un reset). En parallèle, Espressif maintient sur Github un deuxième SDK, celui-ci basé sur FreeRTOS, un OS temps réel permettant de bénéficier de véritables threads.
Enfin, le dernier SDK est un projet alternatif open source, non supporté par Espressif, visant à créer un framework ouvert pour l'ESP8266 en se basant également sur FreeRTOS.
Grâce à la libération de ce SDK, il est désormais possible de développer des applications permettant d'exploiter toute la puissance du microcontrôleur 80Mhz en toute autonomie et également de créer des « middlewares » mettant un framework et/ou langage de plus haut niveau !

Développer en Lua avec NodeMCU

Développer des applications natives sur ESP est assez compliqué pour celui qui ne connait pas le développement bas niveau.
NodeMCU a eu l'idée de créer un firmware ESP8266 embarquant eLua, un interpréteur Lua très minimaliste pour système embarqué. Grâce à ce firmware vous pouvez développer des programmes pour votre ESP très facilement en Lua.
Concrètement, il faut commencer par flasher votre ESP avec le firmware NodeMCU en utilisant un outil comme ESP Flash Download présenté ci-dessus. Vous pouvez télécharger le binaire du firmware directement sur le GitHub de NodeMCU. La dernière release en date est la « 20150704 » du 04 juillet basée sur le SDK IoT 0.9.6 d'Espressif (c'est une vieille version !!).

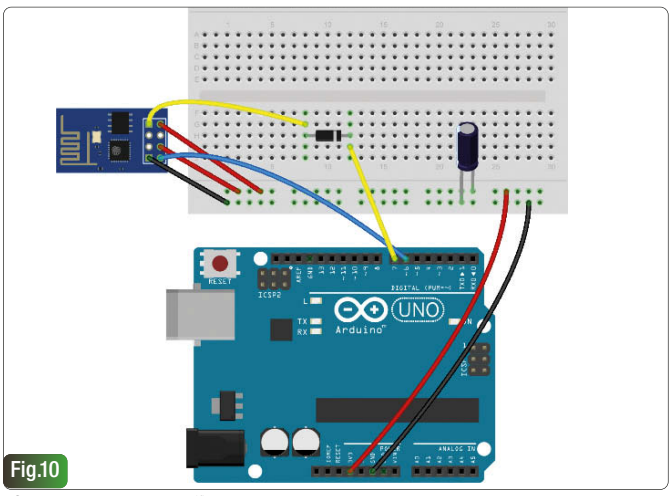


Fig.10 *ESP8266 comme module Wifi*

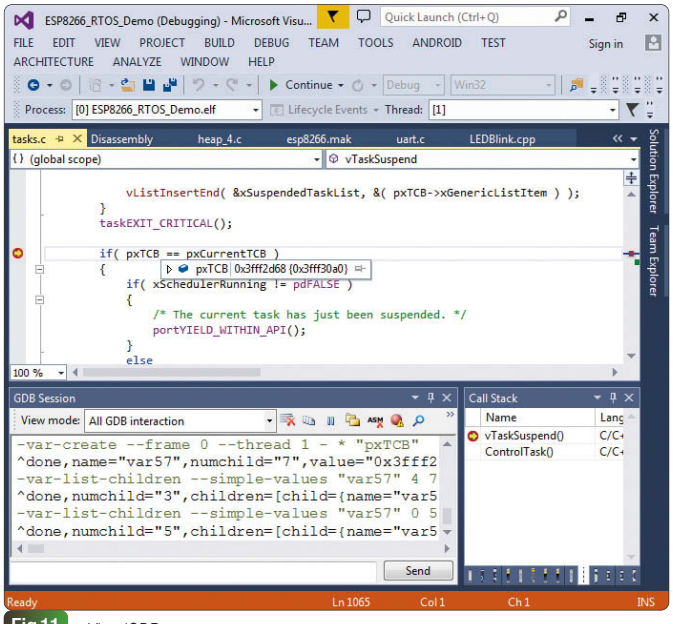


Fig.11 *VisualGDB*

Une fois installé, utilisez l'outil Esplorer, une application Java (compatible Linux/Windows et Mac) qui permet d'éditer et uploader des fichiers Lua dans le file system de votre ESP Fig.12.
A gauche, vous trouverez l'éditeur, à droite la sortie de votre ESP, et dans le menu complètement à droite vous pourrez gérer les fichiers présents dans la mémoire de votre ESP (éditer, supprimer, renommer, etc.).
Par exemple, pour se connecter au Wifi de la maison :

```
print('Démarrage')
wifi.setmode(wifi.STATION)
print('set mode=STATION (mode='..wifi.getmode()..')')
wifi.sta.config("MON RESEAU WIFI", "Ma clé Wifi")

tmr.alarm(0, 1000, 1, function()
  if wifi.sta.getip() == nil then
    print("Connecting to AP...")
  else
    print('Connected! IP: ',wifi.sta.getip())
    tmr.stop(0)
end)
```

On commence par définir le mode Wifi à Station (client) puis on se connecte en spécifiant le SSID et la clé Wifi. Ensuite, on lance une tâche qu'on exécute sur le timer « 0 » (vous pouvez utiliser 6 timers différents) avec une seconde d'intervalle (1000ms) de façon répétitive afin de récupérer l'IP. Si la connexion n'est pas encore effective on affiche le message « Connecting » sinon on coupe le timer « 0 » et on affiche l'adresse IP de l'ESP.
Avec NodeMCU, la gestion des connexions est basée sur des évènements. On va donc créer un socket puis s'abonner à l'évènement « connected » pour envoyer une requête http, et sur l'évènement « receive », on affichera sur la sortie standard la réponse du serveur :

```
local sk = net.createConnection(net.TCP, 0)
sk:on("disconnection", function(sck, c)
  print("Disconnected")
  end)
sk:on("receive", function(sck, c)
  print("Receiving answer : " .. c)
  sck:close()
  end)
sk:on("connection", function(sck, c)
  print("Connected")
  sck:send("GET / HTTP/1.1\r\n"
  sck:send("Host: " .. address .. "\r\n)
  sck:send(Connection: keep-alive\r\nAccept: */*\r\n\r\n")
  end)
sk:connect(port, address)
sk = nil
```

On peut bien sûr lire et écrire sur nos GPIO à la manière d'une Arduino :

```
gpio.mode(pin,gpio.OUTPUT)
gpio.write(pin,gpio.HIGH)
print(gpio.read(pin))
```

NodeMCU met à disposition plusieurs modules tel quel : i2c, pwm, spi, adc (pour les I/O analogiques), uart, donc de quoi s'interfacer avec tout type d'équipement. Bien que NodeMCU soit une alternative très intéressante pour développer très rapidement sur ESP8266, la plateforme souffre de gros problèmes de mémoire. Chaque fonction créée occupe une place importante sur la pile, et dès que vous avez un projet un peu plus compliqué qu'une simple acquisition de capteur pour envoyer une requête http, ça devient critique. Il n'est pas rare d'avoir des plantages avec le message « Not enough memory ».
Je vous conseille d'ailleurs de compiler vos scripts Lua pour générer un fichier LC, vous gagnerez quelques ko de mémoire ; privilégiez les variables locales, réduisez au maximum le nombre de fonctions et surveillez fréquemment la taille de la pile en invoquant la méthode « node.heap() ».
De ce fait, NodeMCU reste un détour incontournable dans la découverte

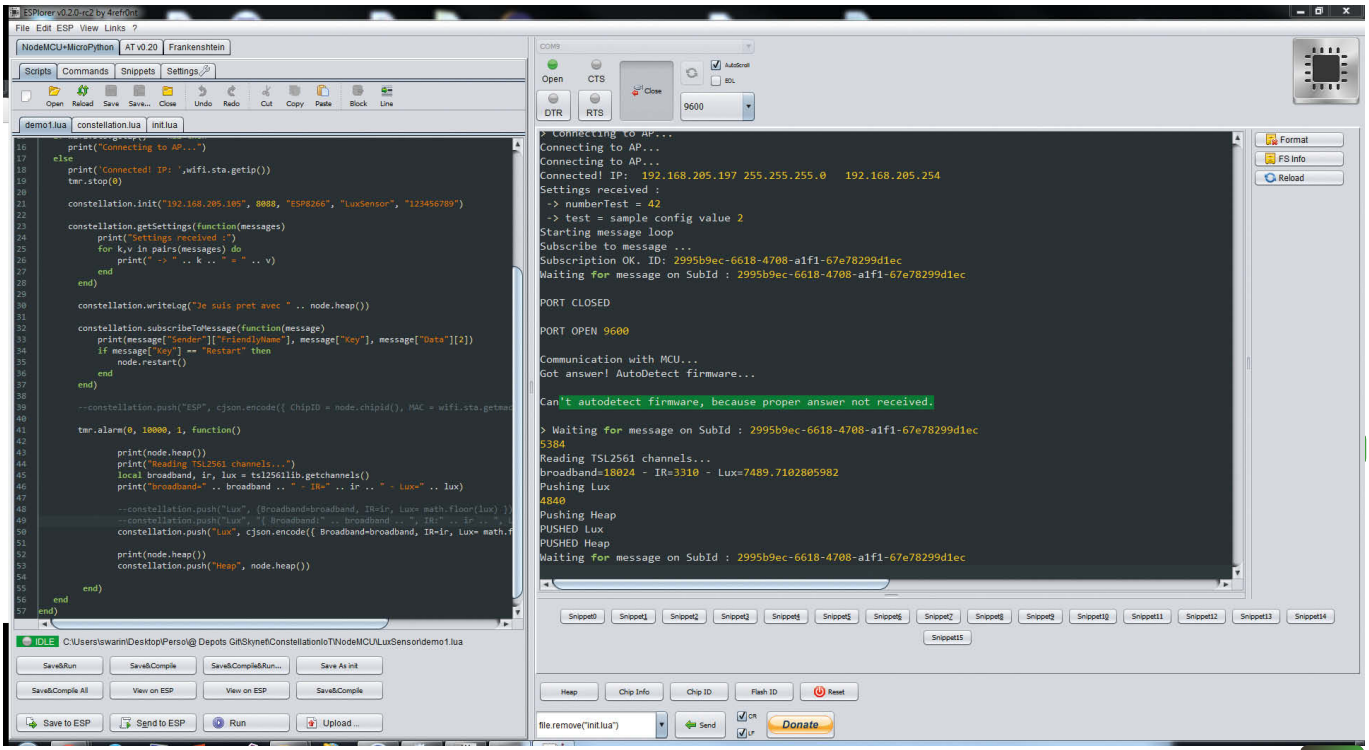


Figure 13 - NodeMCU via Esplorer Fig.12

de l'ESP8266, mais pour créer des projets plus fiables, je vous conseille vivement de développer des sketches Arduino qui offrent un bon compromis entre la fiabilité et performance du développement natif en C++ et la simplicité de NodeMCU.

Développer sur ESP8266 comme sur une Arduino

Le principe est très simple, avec l'IDE Arduino on ajoute simplement la board « ESP8266 » dans le « Additional Board Manager » Fig.13.
Ensuite, vous pouvez écrire votre sketche Arduino comme à votre habitude. Par exemple, pour se connecter sur le Wifi :

```
#include <ESP8266WiFi.h>

void setup() {
 Serial.begin(115200);

 WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
 }

 Serial.print("Connected ! IP address: ");
 Serial.println(WiFi.localIP());
}
```

Une fois connecté, exécutons une requête http avec affichage de la réponse sur l'interface série :

```
WiFiClient client;
if (!client.connect(host, httpPort)) {
  Serial.println("connection failed");
  return;
}
client.print(String("GET / HTTP/1.1\r\n" +
        "Host: " + host + "\r\n" +
        "Connection: close\r\n\r\n"));
delay(10);
while(client.available()) {
  String line = client.readStringUntil('\r');
  Serial.print(line);
}
```

Lors du télé-versement, il faut penser à mettre votre ESP8266 en « Flash board » car votre sketche est compilé comme un véritable firmware autonome (sans middleware à la NodeMCU). Comme une véritable Arduino vous pouvez accéder aux GPIO avec les classiques digitalRead/digitalWrite. Pour les entrées analogiques (analogRead sur A0) il faudra obligatoirement les connecter sur le port ADC. Vous pouvez également utiliser l'espace de stockage Flash de votre ESP grâce aux objets SPIFFS, Dir et File. Aussi, il est possible d'utiliser les fonctions natives du SDK ESP depuis vos sketches Arduino. On peut donc utiliser les nombreuses librairies Arduino que l'on retrouve sur Internet. Mais attention car certaines d'entre elles ne sont pas compatibles si trop dépendantes du hardware. Par exemple, la librairie Arduino « IRremote » permettant l'envoi et la réception de signaux IR ne compilera pas sur l'ESP8266 car elle utilise le service d'interruption (ISR) propre aux ATmegas (déclaré dans les librairies AVR). J'ai donc réécrit le code de cette librairie pour la rendre compatible avec l'ESP (vous retrouverez ce code sur mon Github). Pour ce faire, j'ai remplacé l'ISR de l'AVR par le système interruption propre au SDK de l'ESP8266. Par exemple, pour inclure des headers souhaités du SDK de l'ESP8266 depuis dans votre librairie Arduino :

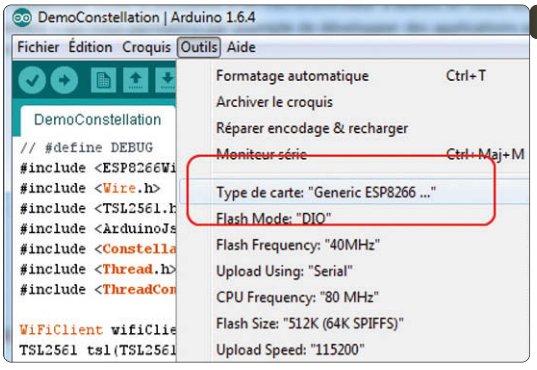


Fig.13 *ESP8266 dans l'Arduino IDE*

```
extern "C" {
  #include "user_interface.h"
  #include "gpio.h"
}
```

Grâce à l'accès aux fonctionnalités natives du SDK officiel d'Espressif, il est possible de porter des librairies trop spécifiques aux ATmegas sur l'ESP dans la mesure du possible. Mais rassurez-vous la majorité des librairies Arduino sont compatibles sur l'ESP8266 in fine !

Quelques autres firmwares

Dans la philosophie de NodeMCU : MicroPython, un moteur Python minimaliste pour système embarqué porté sur ESP8266. Il est possible d'exécuter du code Python sur l'ESP via un interpréteur accessible sur la liaison série. Cependant, le file system n'est pas supporté, donc pas de possibilité d'enregistrer son code dans la mémoire Flash de la puce. Actuellement, ce firmware est très expérimental et ne supporte le Wifi qu'en mode client (station). Vous avez toutefois accès aux GPIO et quelques modules de base comme les array, collection, struct, network et io. Dans la lignée du firmware AT, il existe sur Github le firmware « Frankenstein » beaucoup plus complet et « user-friendly » avec un jeu de commande digne d'un véritable shell. Par exemple, pour scanner les réseaux Wifi on utilisera la commande « iwscan » :

```
frankenstein > iwscan
BSSID b0:48:7a:d6:92:a8 channel 11 rssi -88 auth WPA2_PSK    TP-LINK_D692A8
BSSID c0:4a:00:c7:9d:8e channel 11 rssi -80 auth WPA_WPA2_PSK Home_TP-LINK
```

Un classique "ifconfig" permettra d'afficher les interfaces réseaux avec le détail. Et comme tout shell qui se respecte, la commande « help » vous donnera la liste complète des commandes disponibles. Pour finir, évoquons également Espduino toujours sur Github. Il permet d'utiliser l'ESP8266 comme un module Wifi depuis une Arduino via la liaison série. A la différence du firmware AT qui utilisait un jeu de commande ASCII, Espduino repose sur le protocole SLIP pour la communication Arduino<>ESP. Le SLIP pour Serial Line Internet Protocol est un protocole (RFC 10551) de liaison Internet en série qui encapsule le protocole IP. De ce fait, la liaison est plus performante et plus fiable qu'avec le firmware AT.

Connectez vos ESP8266 dans la Constellation et prenez de la puissance !

La plateforme Constellation que je vous ai déjà présentée dans des numéros précédents de ce magazine, permet l'interconnexion des objets et ser-

vices connectés et la création d'intelligence ambiante. Aujourd'hui, cette plateforme connecte ma domotique Z-Wave, mon thermostat Nest, mes média-centers Kodi, mes lampes Hue, mon alarme Paradox, ma station météo connectée, ma balance connectée, les TVs et amplis de la maison, mon agenda Exchange, etc. sans oublier tous mes projets DIY tel que S-Energy pour le comptage énergétique, S-Sound pour la diffusion audio multi-room, le miroir intelligent, ma porte de garage, etc. Bien sûr, les ESP8266 n'échappent pas à cette règle ! Il existe deux librairies : une librairie Constellation pour NodeMCU, écrite en Lua et une autre Arduino écrite en C++. Au sujet de la libraire Constellation Arduino, elle est basée sur la classe de base « Client » ce qui lui permet d'être compatible avec n'importe quelle interface réseau Arduino (comme le module CC3000, les shields Ethernet pour Arduino et bien sûr les ESP8266).
Il suffit d'initialiser la connexion vers la Constellation en spécifiant l'URI de la Constellation, le nom de la sentinelle, du package et la clé d'accès que vous aurez définie dans la configuration de votre Constellation.
En C++/Arduino :

```
#include <Constellation.h>
Constellation constellation(wifiClient, "192.168.205.105", 8088, "ESP8266", "LuxSensor", "123456789");
```

En Lua/NodeMCU :

```
local constellation = require("constellation")
constellation.init("192.168.205.105", 8088, "ESP8266", " MyPackage", "access_token")
```

On peut ensuite publier des « StateObjects ». Par exemple, un capteur de luminosité pourrait publier un StateObject contenant le nombre de Lux :

```
constellation.pushStateObject("Lux", value);
```

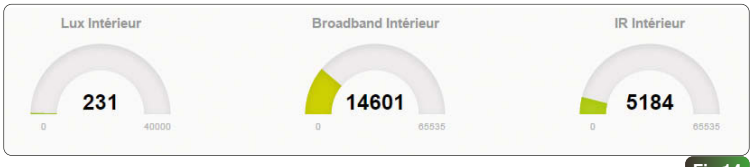
De même qu'il pourrait publier un objet complexe contenant plusieurs propriétés :

```
JsonObject& root = jsonBuffer.createObject();
root["Lux"] = lux;
root["Broadband"] = full;
root["IR"] = ir;
constellation.pushStateObject("Lux", &root);
```

En Lua on écrirait :

```
constellation.push("Lux", cjson.encode({ Broadband=broadband, IR=ir, Lux= math.floor(lux) }))
```

Dès lors, un autre package C#, Python, une page Web avec l'API JS de Constellation, un script Powershell, un Gadgeteer ou n'importe quel système connecté dans votre Constellation peut s'abonner à la mise à jour du StateObject publié par votre ESP8266.
Par exemple, en utilisant l'API Javascript de Constellation, on alimente une jauge dans une page HTML avec notre StateObject « Lux » : **Fig.14**.
On peut également envoyer un message dans la Constellation. Par exemple, si je veux que mon ESP8266 baisse le thermostat de mon Nest, il me suffit d'envoyer un message « SetTargetTemperature » avec la température de consigne au package Constellation « Nest » :



StateObjects dans une page Web **Fig.14**

```
constellation.sendMessage("Package", "Nest", "SetTargetTemperature", temperature);
```

On peut aussi recevoir des messages. Par exemple, depuis une page Web exploitant l'API JavaScript de Constellation, je peux envoyer un ordre de redémarrage. A l'initialisation, je vais m'abonner aux messages entrants et affecter un callback :

```
void messageReceive(JsonObject& json) {
 const char * key = json["Key"].asString();
 if (stricmp (key, "Restart") == 0) {
  ESP.restart();
 }
}

// Dans le setup :
constellation.setMessageReceiveCallback(messageReceive);
constellation.subcribeToMessage();
```

Côté Web, un bouton HTML et une ligne de code pour envoyer le message « Restart » sans paramètre vers la sentinelle "ESP8266" :

```
$('#btnRestart').click(function() {
   constellation.sendMessage({ Scope: 'Sentinel', Args: [ 'ESP8266' ] }, 'Restart', {});
});
```

Un simple clic sur le bouton HTML enverra le message « Restart » qui sera reçu sur votre ESP, lequel redémarrera instantanément !
Une autre fonctionnalité très intéressante : les « StateObjectLink ». C'est la capacité depuis un Arduino/ESP à s'abonner aux changements des StateObjects publiés par d'autre package de ma Constellation. Par exemple, je veux être informé de la T° de la chambre mesurée par mon capteur NetAtmo, ou bien savoir si la porte de garage est ouverte, ou encore connaître le volume de l'ampli, et tout cela en temps réel.
Prenons l'exemple de la consommation d'électricité publiée par le package S-Energy. L'idée sera d'afficher le nombre de Watts par heure de la maison en temps réel, dans la console de notre ESP/Arduino :

```
void soUpdate(JsonObject& json) {
  // On affiche la propriété 'WattPerHour' du StateObject
  Serial.print("Watt/heure : "); Serial.println(json["Value"]["WattPerHour"].asString());
}

// Dans le setup, on s'abonne au StateObject 'Electricity' du package 'SEnergy'
constellation.setStateObjectUpdateCallback(soUpdate);
constellation.subscribeToStateObjects("*", "SEnergy", "Electricity");
```

Tout comme les messages, il suffit d'affecter un callback et de préciser les StateObjects que l'on souhaite suivre ! Ainsi, à chaque variation de ma consommation électrique, mon Arduino/ESP a l'information en temps réel et l'enverra sur l'interface série ! Ainsi, les possibilités sont immenses car vos petites puces peuvent accéder à toutes les données et fonctionnalités de vos autres objets ou services connectés en quelques lignes de code sans aucune complexité apparente. Baisser le chauffage, allumer des lumières, changer la chaîne de votre TV, envoyer un SMS ou bien parler dans une pièce de la maison est accessible en une ligne de code depuis votre ESP/Arduino. Et vice-versa : les données et fonctionnalités embarquées de vos objets connectés à base d'ESP8266 sont à leur tour accessibles pour les autres objets et services connectés de votre Constellation. La seule limite sera votre imagination. ⊞

Des projets ESP8266 dans toute la maison avec la Constellation

De par sa taille, ses spécifications et son prix, le microcontrôleur Wifi ESP8266 offre une multitude d'usages. Couplé à une plateforme d'interconnexion des objets connectés telle que la plateforme Constellation, il devient possible en un rien de temps de faire des objets ultra connectés comme nous allons le découvrir.

Sébastien Warin
Creative Technologist @ Publicis ETO
http://sebastien.warin.fr

Objet #1 : mieux piloter l'éclairage intérieur avec un capteur de luminosité

A la maison, j'ai installé sur mon toit un capteur de luminosité TSL2561 dans une boite étanche. Ce capteur est relié en I²C à un Raspberry Pi lui-même connecté dans la Constellation. Le package « LuxSensor », écrit en Python, publie dans la Constellation un StateObject nommé « Lux » qui contient un objet JSON incluant le nombre de Lux mesurés !

Toujours dans ma Constellation, j'ai un package écrit en .NET nommé « MyBrain » abonné à ce StateObject, et qui, en fonction de plusieurs conditions liées à d'autres StateObjects (comme la présence ou non d'utilisateurs dans les pièces à vivre via les StateObjects des détecteurs de présence de l'alarme), pilote les luminaires de la maison.

Néanmoins, comme le capteur se situe sur le toit, je mesure les lux extérieurs. De ce fait, l'information n'est pas toujours proportionnelle à la luminosité du salon (variable en fonction des nuages par exemple). C'est pourquoi, afin d'améliorer l'algorithme il me fallait un capteur de luminosité à l'intérieur.

J'ai donc acheté un boitier/prise 220V (10€), un transformateur 220V vers 3,3V (2€), un capteur TSL2561 (2€) et un ESP8266 **Fig.1**.

Vous trouverez sur Github la librairie Arduino pour le TSL2561 parfaitement compatible avec l'ESP. Il suffit juste de connecter ce module I²C sur les GPIO0 et 2, et initialiser l'I²C en spécifiant les ports utilisés **Fig.2**.

Une fois mon ESP connecté dans Constellation, je récupère le setting « Interval », ce qui me permet de changer la période de rafraîchissement depuis ma Constellation.

```
JsonObject& settings = constellation.getSettings();
int interval = String(settings["Interval"].asString()).toInt();
```

Avec les méthodes « getFullLuminosity" et « calculateLux » je récupère le spectre complet, l'IR et je calcule les Lux.

Il ne reste plus qu'à publier le StateObject dans la Constellation :

```
// Push on Constellation
stateObjectLux["Lux"] = lux;
stateObjectLux["Broadband"] = full;
stateObjectLux["IR"] = ir;
constellation.pushStateObject("Lux", &stateObjectLux, "LightSensor.Lux");
```

Et voilà comment créer un capteur connecté dans votre Constellation très rapidement !

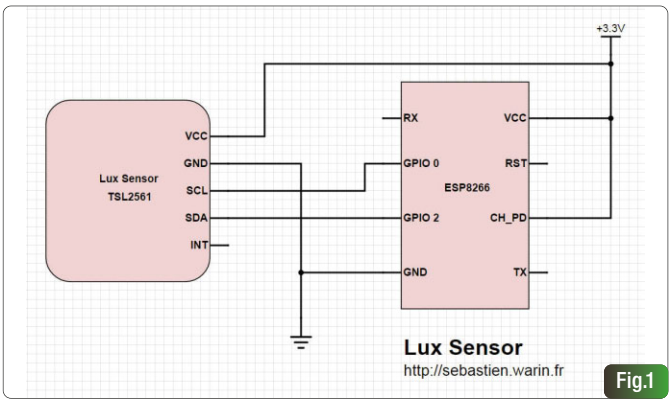
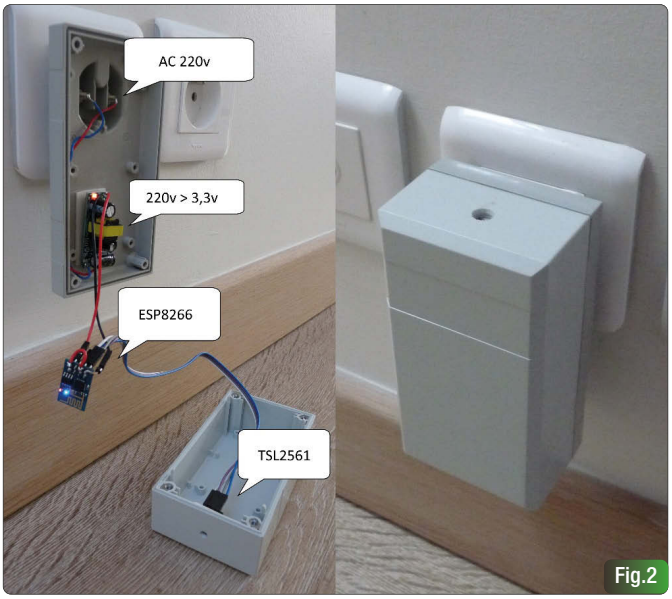


Schéma Lux Sensor



Lux Sensor

En C# je peux maintenant créer une propriété liée à ce StateObject, qui sera rafraîchi en temps réel :

```
[StateObjectLink("ESP-LightSensorSalon", "LightSensor", "Lux")]
public StateObjectNotifier IndoorLuxSensor { get; set; }
this.IndoorLuxSensor.ValueChanged += (s, e) =>
{
   Console.WriteLine("Nouvelle valeur de Lux : {0}", e.NewState.DynamicValue.Lux);
};
```

J'ai pu ainsi mettre à jour mon algorithme de gestion des lumières du salon en prenant en compte la luminosité intérieure.

Objet #2 : un anneau lumineux connecté

J'ai retrouvé dans mes cartons un NeoPixel Ring d'Adafruit, un anneau de 12 leds multi-couleurs pilotable individuellement par PWM.

J'ai donc simplement alimenté l'anneau en 3.3V et connecté le « Data In » de l'anneau sur le GPIO 0 de l'ESP ! Fig.3.
Côté code, j'ai utilisé la librairie Adafruit, là encore compatible ESP pour piloter mes LEDs. Par exemple, pour éteindre la 1ère LED et allumer en rouge la 2ème, on écrira :

```
pixels.setPixelColor(0, pixels.Color(0, 0, 0));   // 1ère OFF
pixels.setPixelColor(1, pixels.Color(150, 0, 0));   // 2ème en rouge
```

Maintenant, l'idée sera de piloter les LEDs depuis la Constellation. On s'abonne au message nommé « ShowLed » contenant le nombre de LEDs à allumer dans les paramètres du message. La méthode showLed(int) s'occupera de faire les setPixelColor.

```
void messageReceive(JsonObject& json) {
 const char * key = json["Key"].asString();
 if (stricmp(key, "ShowLed") == 0) {
   showLed(json["Data"].as<int>());
 }
}

// Dans le setup()
constellation.setMessageReceiveCallback(messageReceive);
constellation.subscribeToMessage();
```

On peut par exemple, à partir d'une simple page Web et avec la librairie Constellation Javascript, piloter en temps réel l'anneau lumineux. En utilisant un slider :

```
<input type="range" id="nbLed" min="0" max="12" />

$('#nbLed').on("slidestop",
  function( event, ui ) {
     constellation.sendMessage({ Scope: 'Sentinel', Args: [ 'ESP8266' ] }, 'ShowLed', $(this)[0].value);
  });
```

Fig.4.

Et voilà comment une simple page Web peut contrôler en temps réel votre ESP !

Objet #3 : un Hardware Monitor lumineux

Reprenons le même objet, un ESP avec un anneau lumineux. A l'exception que cette fois-ci, on contrôlera les LEDs en fonction d'un StateObject et non d'un message reçu.
Sur chaque machine Windows que je possède, est déployé le package Constellation « HWMonitor » qui publie une multitude de StateObject liés au hardware de la machine comme la consommation CPU, RAM, disque, réseau, etc... Il suffit donc de s'abonner au StateObject du CPU d'une de mes machines et d'afficher le nombre de LEDs proportionnellement à la consommation du CPU.

```
void soUpdate(JsonObject& json) {
 int cpu = json["Value"]["Value"].as<int>();
 showLed((cpu * ratio) + 1); // ratio = (nombre de LED / 100)
}
// Dans le setup()
constellation.setStateObjectUpdateCallback(soUpdate);
constellation.subscribeToStateObjects("SEB-PC", "HWMonitor", "/intelcpu/0/load/0");
```

Ainsi, mon anneau lumineux réagit maintenant en temps réel à la consommation CPU de mon ordinateur ! Fig.5.

Objet #4 : l'indicateur de zone

Partant du principe que l'on peut s'abonner à la mise à jour de n'importe quel StateObject d'une Constellation, on peut par exemple faire un indicateur d'état d'ouverture d'une porte ou d'une fenêtre.
Dans ma Constellation je peux retrouver ce genre de StateObject avec mon système d'alarme (capteur d'ouverture analogique) ou mon système domotique (capteur d'ouverture Z-Wave).
Pour l'affichage, prenons une matrice de Led bi-couleur pilotable en I C. Il suffit alors de la connecter sur la GPIO 0 et 2 pour le SDA et SCL de l'I C et de l'alimenter en 5V Fig.6.
Coté code, je récupère l'ID de la zone à surveiller depuis les settings Constellation :

```
JsonObject& settings = constellation.getSettings();
const char *zoneId = settings["ZoneId"].asString();
```

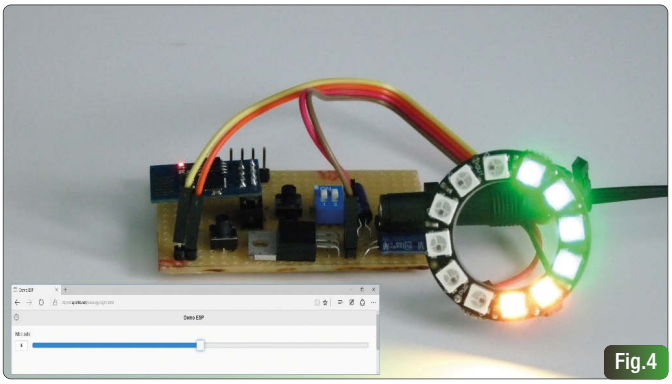


Fig.4

Contrôler l'anneau lumineux en Javascript

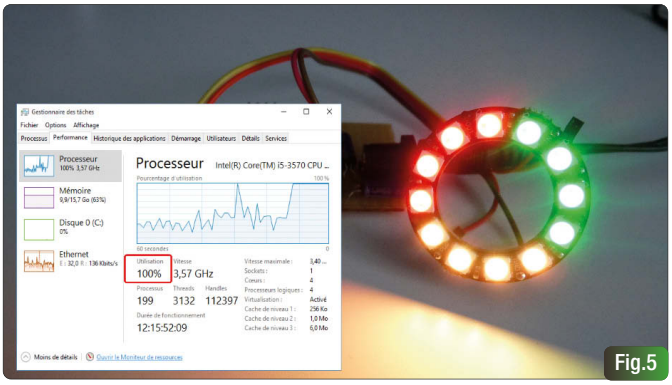


Fig.5

Hardware Monitor

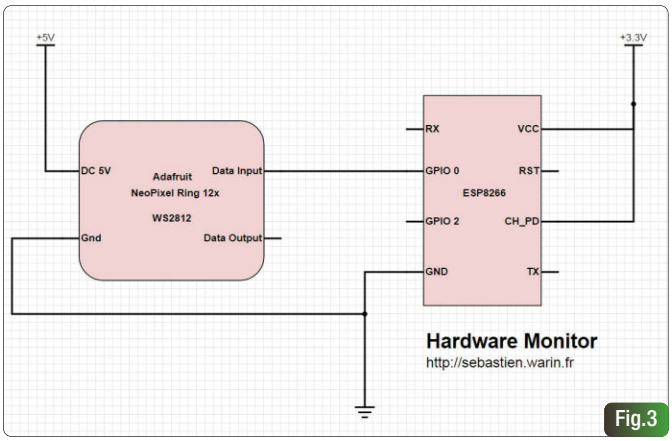


Fig.3

Schéma NeoPixel Ring

Grâce à cela, il me suffira depuis la console d'administration Constellation de changer le n° de la zone sans avoir à reprogrammer l'ESP.
Ensuite, on va demander l'état du StateObject de la zone à surveiller et on s'abonne à toute modification de cette zone :

```
// Request
JsonArray& zoneInfo = constellation.requestStateObjects("CEREBRUM", "Paradox", zoneId);
soUpdate(zoneInfo[0]);
// Subscribe
constellation.setStateObjectUpdateCallback(soUpdate);
constellation.subscribeToStateObjects("CEREBRUM", "Paradox", zoneId);
```

Dans les deux cas, on appellera la méthode "soUpdate" en passant en paramètre le StateObject de la zone. Ce StateObject publié par le package Constellation de l'alarme contient un objet JSON avec la propriété « IsOpen » qui nous renseignera sur l'état de la zone :

```
void soUpdate(JsonObject& json) {
  boolean isOpen = json["Value"]["IsOpen"].as<boolean>();

  Serial.print("La zone est ");
  Serial.println(isOpen ? "ouverte" : "fermée");
}
```

Il ne reste plus qu'à piloter la matrice de LEDs. La librairie d'Adafruit est très simple. En effet, il suffit d'appeler la méthode « drawBitmap » en passant en paramètre la couleur des LEDs et la matrice de points.
Dessinons alors un carré vert lorsque la zone est fermée, et une croix rouge lorsque la zone est ouverte. Les deux matrices seront décrites dans des tableaux nommés « x_bmp » et « box_bmp ». Il ne reste donc plus qu'à dessiner en fonction de l'état du StateObject :

```
if (isOpen) {
  // display a red X
  matrix.clear();
  matrix.drawBitmap(0, 0, x_bmp, 8, 8, LED_RED);
  matrix.writeDisplay();
}
else {
  // display a green box
  matrix.clear();
  matrix.drawBitmap(0, 0, box_bmp, 8, 8, LED_GREEN);
  matrix.writeDisplay();
}
```

Notre ESP pilotant une matrice de LEDs indique en temps réel l'état d'une zone, par exemple, la porte de ma cuisine Fig.7.

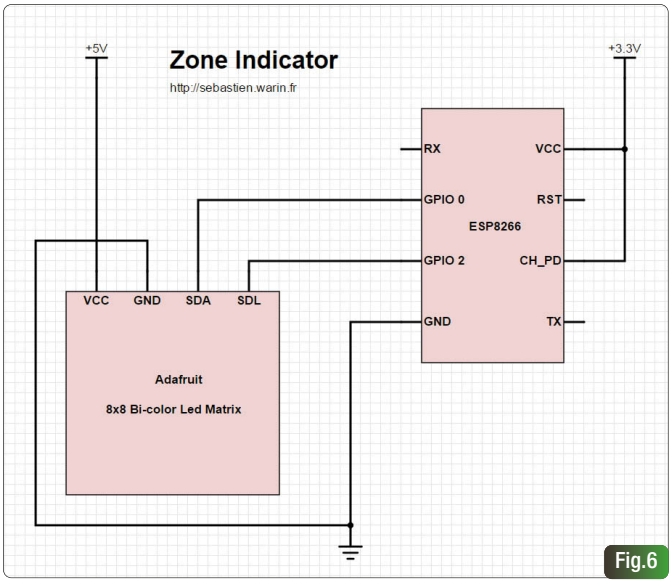


Fig.6

Schema du Led Matrix


Fig.7

Zone Indicator en action

Objet #5 : contrôler sa maison avec une télécommande infrarouge

Pour finir, voyons comment piloter sa maison avec une simple télécommande infrarouge.
Comme expliqué dans l'article précédent, j'ai porté une librairie IR sur l'ESP8266 me permettant ainsi d'envoyer des signaux infrarouges mais

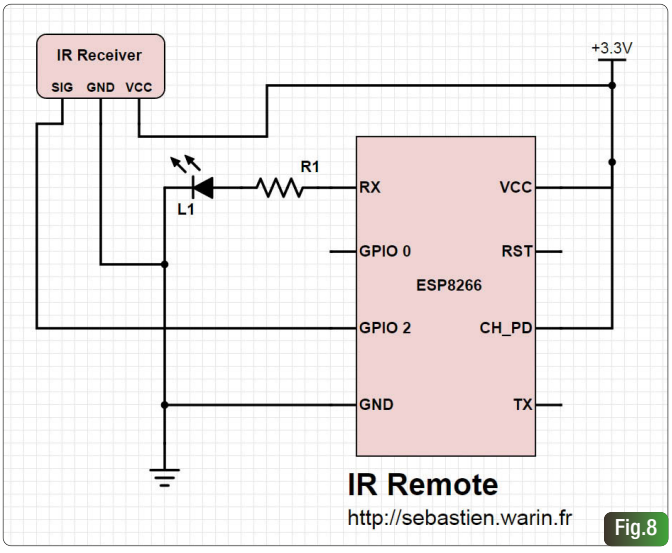


Fig.8

Schéma du IR remote

également de les décoder. Pour l'envoi : il suffit de piloter une LED infrarouge en PWM pour moduler le signal. En principe, on devrait connecter cette LED sur la GPIO 0 ou 2, mais comme la diode est reliée à la masse, l'ESP entrera en « Flash mode » au démarrage. Pour contrer cet effet, j'ai connecté la LED sur le GPIO1 (le RX) après l'avoir programmé.
Pour la réception : le récepteur IR 38kHz est alimenté en 3,3V et le signal reçu est envoyé sur la GPIO2 **Fig.8**.

J'ai donc packagé dans un petit boîtier en plastique l'ESP8266 avec un régulateur LD33V, la LED et le récepteur IR. Il suffit de connecter une alimentation de 5 à 15V et l'installer dans votre salon par exemple **Fig.9 et 10**.

Au niveau du programme Arduino, le principe est simple :
Lorsque l'on reçoit un message « SendCode » de la Constellation, on envoie le signal infrarouge dont le code se trouve dans les paramètres du message.
Lorsqu'on l'on décode un signal IR, on l'envoie dans un message dans la Constellation.
Ainsi n'importe quel système connecté dans la Constellation peut envoyer un signal IR comme un programme .NET, Python, un script Powershell, un Gadgeteer, etc.
Par exemple, une page Web peut allumer ma TV Samsung en envoyant simplement le bon code IR à notre ESP :

```
var power = function() {
  constellation.sendMessage({ Scope: 'Sentinel', Args: [ 'ESP8266' ] }, 'SendCode', { Code: 0xE0E040BF, Encoding: "Samsung" });
};
```

Ce qui est encore plus intéressant, est le fait qu'un signal IR décodé par notre ESP sera envoyé dans un message Constellation nommé « SignalReceive » à destination du groupe « IR ».
Ainsi, mon package .NET « MyBrain » peut s'abonner à ce groupe et écouter les messages « SignalReceive ».
En C# on écrira très simplement :

```
PackageHost.AddToGroup("IR"); // A appeler au démarrage pour s'attacher au groupe IR
[MessageCallback]
public void SignalReceive(string code)
{
  Console.WriteLine("L'ESP a décodé le signal IR : {0}", code);
}
```

Il reste simplement à définir ce que l'on veut faire en fonction des codes reçus. Pour ma part, j'ai repéré les codes des touches de mes télécommandes qui ne sont jamais utilisées afin de les affecter à d'autres commandes, comme contrôler les lumières Z-Ware, les lampes Hue, monter ou baisser le thermostat Nest, etc. Bref ! De quoi contrôler très facilement la maison avec la télécommande de la TV sans avoir à sortir un smartphone ou une tablette !

```
if (code == "E0E020DF") // Si touche "1"
{
  // Exemple : Mettre le thermostat à 21°
  this.SendMessage("Nest").SetTargetTemperature(21);
}
else if (code == "E0E0A05F") // Si touche "2"
{
  // Exemple : allumer lampe Z-Wave #21
  this.SendMessage("Vera").SetSwitchState(new { DeviceID = 21, State = true });
}
else if (code == "E0E0609F") // Si touche "3"
{
  // Exemple: Allumer lampe Hue #2 avec Hue:6479/Sat:252
  this.SendMessage("Hue").Set(2, true, 6479, 252, 254);
}
else if (code == "E0E010EF") // Si touche "4"
{
  // Exemple : lire du texte dans le salon
  this.SendMessage("SSound-Salon").Speech("Wajou, ca rox du poney !");
}
```

Conclusion

A travers ces quelques projets basés sur des ESP8266 et grâce à la plateforme Constellation, le potentiel de ces petites puces est immense. Elles servent à la fois de capteurs pour publier des données dans la Constellation qui seront utiles à d'autres programmes (par exemple un capteur de T°, de consommation, de mouvements, etc..), ou encore réagir à des messages pour pouvoir ajouter du contrôle à vos ESP (par exemple un robot, une prise commandée, etc.).
Plus encore, ces petites puces peuvent réagir en fonction de l'état de vos autres objets ou services connectés dans la Constellation grâce aux StateObjects. Ainsi, n'importe quel évènement se produisant dans votre maison ou au-delà, et peu importent les technologies mises en œuvre, peut être « capté » par vos ESP.
Du moment qu'elles sont connectées à la Constellation, elles accèderont à toutes les fonctionnalités et données de votre Constellation ! Que ce soit votre agenda, votre domotique, vos objets connectés, vos projets maison, vos ordinateurs et même votre douche !

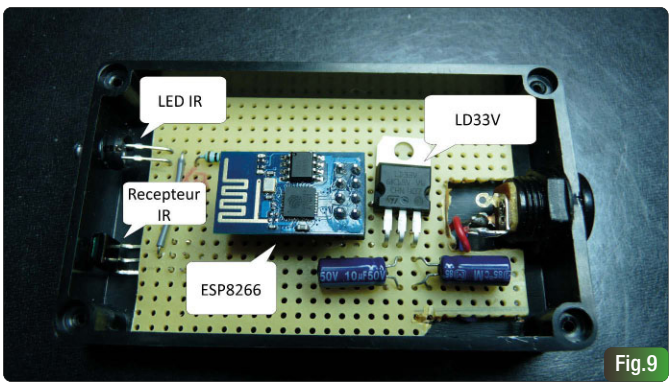


Fig.9

Intérieur du IR Remote


Fig.10

IR remote

Developer Assistant pour Visual Studio

Dans le numéro 158 de Programmez (décembre 2012) je vous présentais « All-In-OneFramework » un Sample Browser qui permettait de faciliter la recherche de codes sources selon plusieurs critères, suivant les technologies employées. Cet outil a maintenant évolué et est devenu une extension pour Visual Studio nommée «Developer Assistant».

Christophe Peugnet
MVP Windows Plateform Devlopment chez SodeaSoft / EBLM
http://www.sodeasoft.com
Blog : http://www.peug.net
Twitter : @tossnet1

Developer Assistant, édité par une équipe de Microsoft, va aider le développeur à trouver rapidement des extraits de codes via par exemple MSDN, GitHub, StackOverFlow, Code Project.com... Il se base entre autre sur Bing Code Search de Microsoft Research que vous pouvez retrouver et tester en ligne à cette adresse : http://codesnippet.research.microsoft.com/.

L'extension « Developer Assistant » est disponible pour les versions 2012, 2013 et 2015 de Visual Studio.

Pour l'installer, il vous suffit simplement de la rechercher via la fonction [Extension et mise à jour] du menu Outils : Fig.1.

Recherche dans GitHub

GitHub est l'une des communautés open source les plus populaires et on y retrouve un très grand nombre de projets .NET, Node.JS, les librairies comme HTMLAgilityPack, NewtonSoft.json, Xamarin... Actuellement les moteurs de recherche ne sont pas encore optimisés pour effectuer des recherches de codes dans les projets GitHub, mais grâce à « Developer Assistant » vous pouvez augmenter votre efficacité. Un explorateur de code permet de filtrer facilement les codes de GitHub, de les cloner dans un dossier local de votre choix et un autre onglet permet de récupérer des bouts de code (Snippet) : Fig.2.

L'expérience IntelliSense

L'équipe de développement de cette extension a soigné la partie IntelliSense. L'IntelliSense c'est l'auto-complétion du code dans l'IDE (environnement de développement ou IDE en anglais, pour Integrated Development Environment).

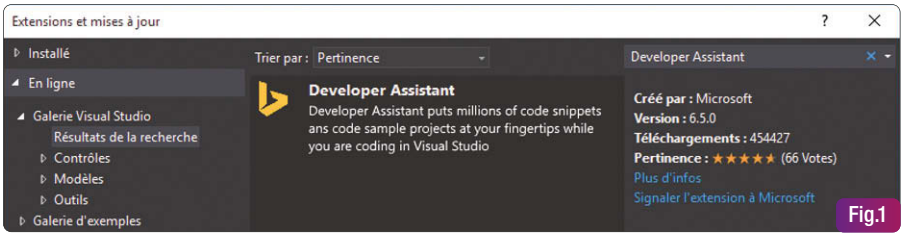


Fig.1

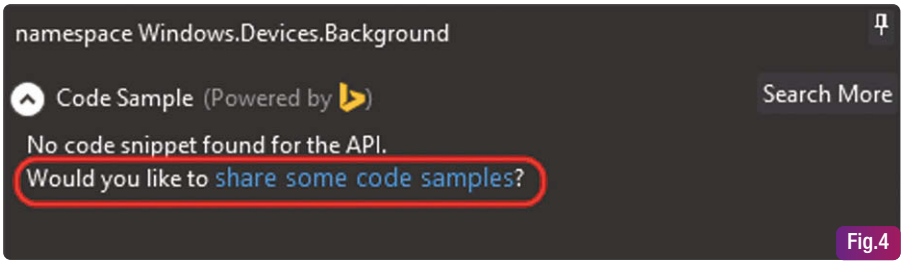


Fig.4

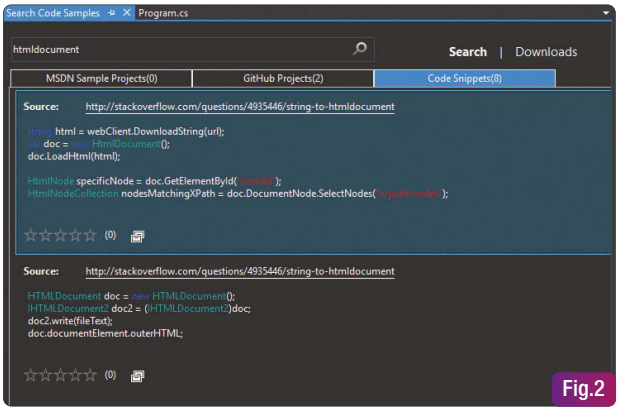


Fig.2

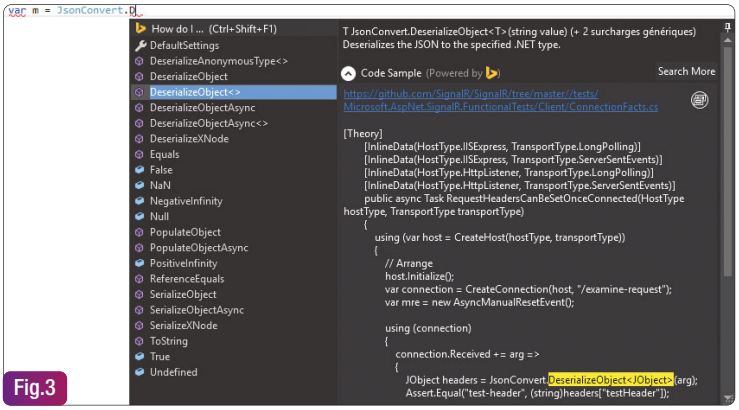
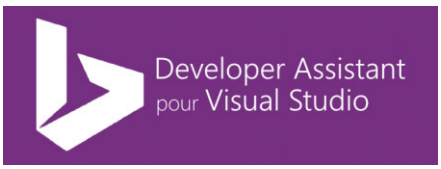


Fig.3



Globalement, cette invention géniale permet aux développeurs les plus fainéants comme moi de taper moins de caractères sur leur clavier. Dans Visual Studio, ce système s'appelle l'IntelliSense et il est particulièrement bien fait et c'est sur ce dernier que cette extension est venue se greffer : Fig.3. Depuis l'Intellisense, on peut visualiser un bout de code relatif à l'API que nous sommes en train de taper. La vitesse de recherche de Bing Code Search qui gère cette partie m'a juste surpris ! Un lien de cet extrait de code vous permettra d'aller sur la page de cette source, mais aussi de copier le code dans le presse-papier ou demander de rechercher d'autres propositions de codes via le bouton [Search more]

Proposer à MSDN votre exemple de code

Il arrive qu'aucun extrait de code ne soit trouvé ou disponible sur une API que vous utilisez. Donc là c'est tant pis pour vous, il va falloir faire sans mais ce qui est sympa c'est qu'une fois que vous aurez trouvé le fonctionnement de l'API, vous avez la possibilité de proposer à MSDN votre petit exemple très rapidement Fig.4. Le lien « share some code samples » vous ouvre dans Visual Studio le formulaire très simple pour y coller votre code. Si ce n'est pas directement fait, vous vous connectez avec mon compte Microsoft et il ne vous reste plus qu'à donner une petite explication et « bonjour la visibilité » ! La communauté de développeurs vous remercie par avance.

Une recherche locale

Précédemment, je vous disais que la recherche s'effectue via différents sites communautaires pour vous proposer des exemples. Eh bien l'extension permet aussi de faire une recherche en local sur votre disque dur. L'idée est intéressante car nous sommes beaucoup à avoir un dossier local d'exemples ; vous pouvez donc configurer Developer Assistant via le menu [Outils] puis [options] de Visual Studio pour paramétrer votre chemin. Ainsi lors des recherche et lorsque, par exemple, aucune connexion au réseau n'est disponible, eh bien l'Intellisense vous proposera tout de même des extraits Fig.5.

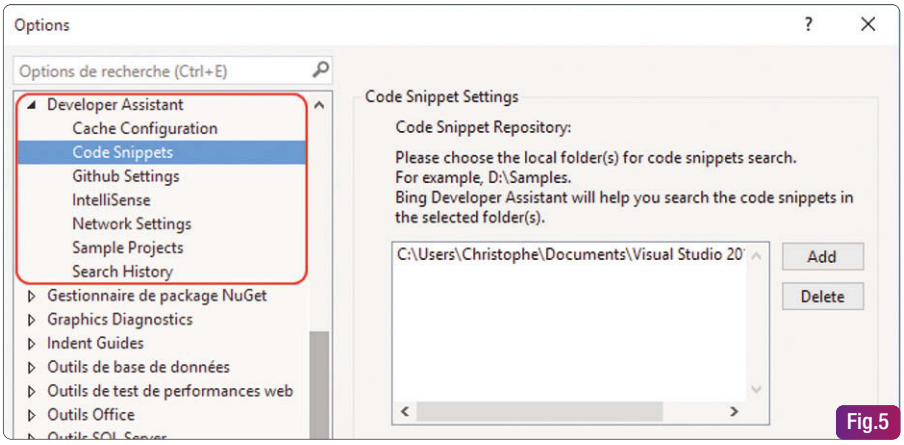


Fig.5

Une recherche omniprésente

Cette extension est en effet omniprésente et surtout de manière contextuelle, en plus d'une barre d'outils dédiée Fig.6. Par exemple nous retrouverons une recherche contextuelle sur l'ensemble des éléments de votre code où il suffit de sélectionner un texte de l'éditeur et dans le menu vous pourrez interroger Bing Search. Au niveau de la liste des erreurs c'est la même chose ; et en effet les codes d'erreurs ne sont parfois pas évidents à comprendre Fig.7. Vous l'aurez compris, Developer Assistant est un must-have qui s'intégre parfaitement à Visual Studio et nous aide dans ce que nous faisons constamment : touver des exemples d'utilisation d'API.

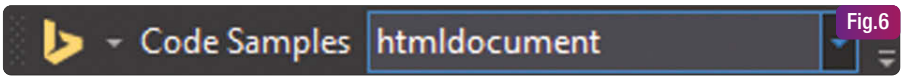


Fig.6

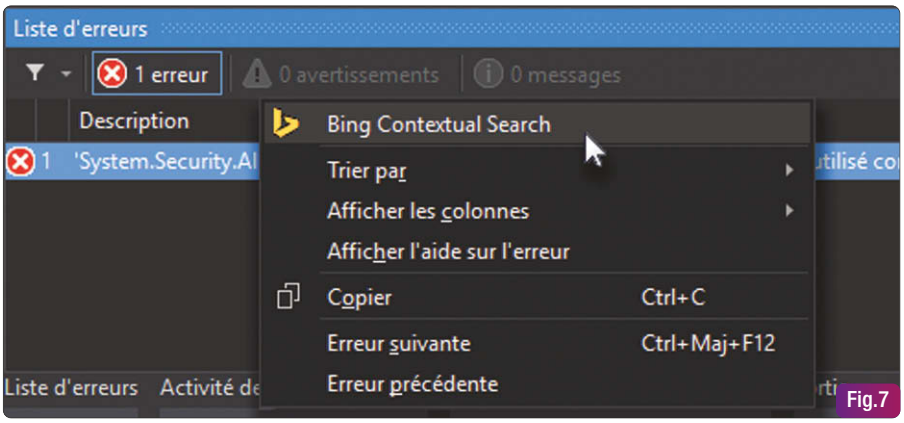


Fig.7

Des petits outils en ligne pour les dévs

Ce mois-ci, l'équipe de Bing a publié des outils très sympathiques pour nous faire gagner du temps pour certaines tâches. Notons que le mois dernier ils avaient ajouté des outils pour les musiciens (« guitar tuner »)

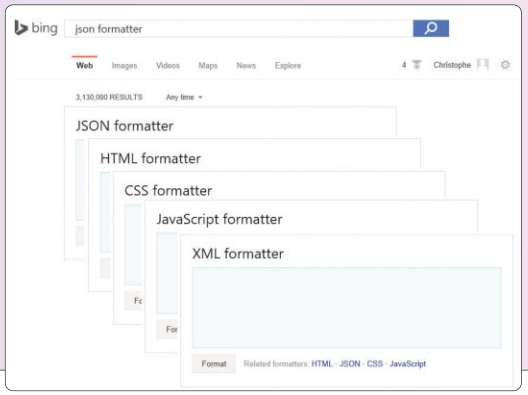
En premier lieu il vous faut changer la région de votre Bing ! Accédez à http://www.bing.com et en haut à droite cliquer sur l'icône représentant un engrenage, cliquez ensuite sur « paramètres ». Dans le nouvel écran cliquer sur [Région] et choisissez « États-Unis — Anglais » puis [Enregistrer]. Notez que si vous bloquez tous les cookies, ça risque de ne pas fonctionner.

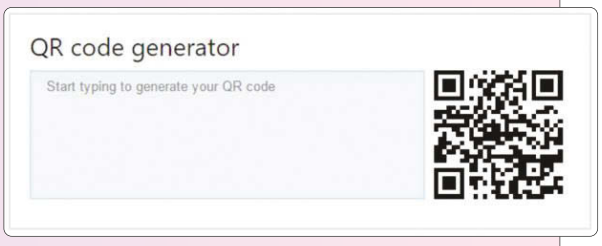
Au retour vous pourrez donc tester plein de petits outils ! Non ne tapez pas « guitar tuner », ici on parle « Dév » ^^, mais par exemple :

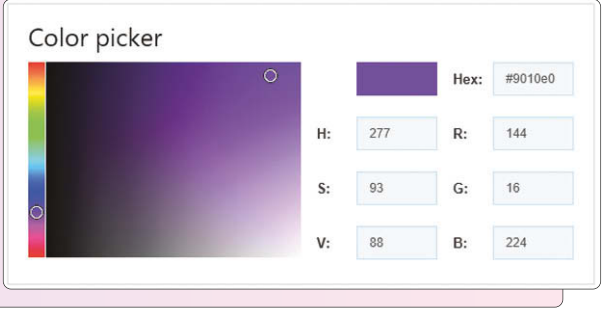
- Color picker : à droite
- XML formater
- Javascript formater
- JSON formater
- HTML formater
- CSS formater

Et aussi :

- Table ASCII
- QR code







Le Deep Learning pas à pas : l'implémentation (2/2)

Manuel Alves, **SQLi**
Co-écrit avec Pirmin Lemberger

Level 300

L'engouement actuel pour le Deep Learning ne repose pas sur les seules avancées conceptuelles de Hinton et al. Elle repose aussi sur des avancées technologiques. Après l'introduction aux concepts présentés dans la 1ere partie (Programmez ! n°189) nous abordons ici les questions liées à l'implémentation de ces réseaux.

L'enjeu majeur de la performance

Pour un informaticien, l'implémentation d'un DBN repose principalement sur le calcul de la formule (7)

$$-\frac{\partial}{\partial\theta}\log p_\theta(\mathbf{v}) = \frac{\partial F(\mathbf{v})}{\partial\theta} - \sum_{\mathbf{u}\in\mathcal{E}} p_\theta(\mathbf{u})\frac{\partial F(\mathbf{u})}{\partial\theta} \qquad (7)$$

Malgré les simplifications astucieuses obtenues en utilisant l'algorithme de Contrastive Divergence et en choisissant les RBM comme briques élémentaires, les expressions mathématiques à évaluer restent très coûteuses en temps de calcul. La question de la performance des algorithmes est donc vitale pour tout outil ou langage destiné à ce champ d'application. A ce titre, toutes les solutions sérieuses disponibles aujourd'hui sont capables d'exploiter l'immense réservoir de puissance de calcul que constituent les ordinateurs modernes, aussi bien en sollicitant le processeur principal (CPU) que les processeurs graphiques dédiés (GPU). Ces derniers sont fabriqués par des fondeurs spécialisés de cartes graphiques comme NVidia ou AMD. Ils sont à l'origine prévus pour le calcul 3D dans le monde du jeu vidéo ou de l'imagerie de synthèse, ou plus récemment sur le traitement vidéo. Quel lien alors avec le sujet du Deep Learning ?

Les GPU ont des fréquences d'horloge bien moindres que celles des CPU (20x) mais ils possèdent de nombreux cœurs (unités de calcul). A titre d'exemple, sur une carte graphique de dernière génération de marque NVidia le GPU intègre 5760 cœurs pour 12Go de mémoire. Ces composants sont parfaitement adaptés aux traitements parallélisables de données de grande dimension (multiplication de matrices, convolution etc.). Par ailleurs, les fabricants se sont attelés à repousser encore plus loin les possibilités offertes par ce type d'architecture. Les technologies SLI (NVidia) ou *Crossfire* (AMD) permettent utilisation jusqu'à 4 GPU dans la même machine, démultipliant quasi proportionnellement la puissance de calcul. Les librairies comme CUDA (chez NVidia), avec ses extensions *C ,C++, openAAC*..., permettent aux développeurs de coder des applications accélérées matériellement par les GPU compatibles, tirant parti du parallélisme offert par la plateforme. Lorsqu'il s'agit d'entraîner un réseau de neurones comportant plusieurs milliards de connexions ou d'essayer simultanément plusieurs modèles sur un jeu de données, cette aptitude à la parallélisation constitue un atout majeur Fig.1.

Des travaux récents produits en 2014, ont conduit à la construction d'un cluster de 64 GPU conçu spécifiquement pour l'entraînement d'un Convolutionnal Neuronal Network (CNN) à 18 couches de neurones pleinement interconnectées, soit 212.7 M de poids synaptiques (w). Appliqué au cas d'usage académique Image Net Large-Scale Visual Recognition Challenge (ILSVRC) cette nouvelle architecture a décroché le nouveau record de taux de reconnaissance sur les 1,2 million d'images classées dans 1000 catégories. Un résultat désormais supérieur à celui de l'œil humain !

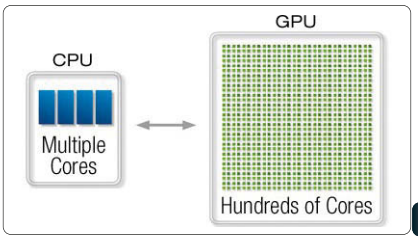


Fig.1 *Architecture GPU vs CPU*

Une offre pléthorique d'outils et langages

Pour un *data scientist* désireux de s'aventurer dans le codage d'un réseau de neurones ou d'un DBN, les frameworks et langages à disposition sont nombreux. Les principaux ténors du sujet sont : *Theano,* un projet de l'université de Montréal, *Torch,* une solution maintenue par des experts œuvrant chez de grands acteurs du Web (Google, Facebook ou Twitter) et *Caffe*, une initiative de l'université de Berkeley. De nombreuses autres alternatives sérieuses fonctionnent en surcouche. Citons *Pylearn2*, une librairie Python de machine learning reposant sur des fondations *Theano* et une alternative, en mode boîte blanche, à l'incontournable *sickit-learn*. Ajoutons la librairie Python *Keras,* sur base de Theano également, mais réellement spécialisée dans le Deep Learning. On trouve également des projets exploitant *Cuda-Convnet*, une implémentation performante *C++/Cuda* des réseaux de neurones classiques. Comme dans tant d'autres domaines, force est de constater qu'il n'y a pas de solution ultime qui surclasserait toutes les autres. Chaque solution aborde le sujet avec une approche spécifique qui conviendra à telle ou telle communauté sans pour autant créer l'unanimité.

Theano, outil par excellence pour coder un DBN ?

Theano est un compilateur, transformant du code Python en instructions *CUDA* et permettant de faire de la programmation symbolique. Il n'est donc pas exclusivement réservé aux DBN. La communauté est importante et active, et la documentation est des plus complètes pour un produit open-source. Le principe est d'écrire le code, non pas comme un informaticien, mais plutôt comme un mathématicien, et de déléguer à *Theano* le travail difficile et fastidieux de passer d'une formule mathématique à du code informatique efficace et performant.

Une somme ne s'écrit donc pas comme une boucle for mais comme une instruction Python **Theano.SUM()**. Le développeur ne calcule pas une dérivée numériquement *[f(x+h)-f(x)]/h* mais l'exprime naturellement avec une instruction déclarative du type **Theano.grad()**, en passant en paramètre un objet **Theano.function()**. C'est *Theano* qui, dans la phase de compilation, va opérer la transformation de ces déclarations Python en opérations informatiques optimisées. En outre *Theano* s'appuie entièrement sur le package scientifique standard *NumPy ;* aucune révolution du côté de la syntaxe et des manipulations de base. Enfin le code compilé, sur les parties pertinentes de calcul, est directement exprimé en instructions *CUDA.* L'intérêt majeur d'intervenir à ce bas niveau d'abstraction est la possibilité de spécifier intégralement, en boîte blanche, une architecture de réseaux de neurones. Les fonctions de coûts les plus exotiques ont toute leur place, et le calcul de leurs gradients/dérivés ne sera plus un casse-tête d'informaticien grâce à *Theano.* La liberté est donc maximale et particulièrement utile en cas d'expérimentation. Cependant cette liberté a un coût. Le travail de programmation symbolique est loin de faire l'unanimité et son accès n'est pas des plus aisés, a minima il nécessite une nouvelle façon de penser. Autre point non négligeable : la

latence induite par la compilation de son travail, et celle-ci peut prendre plusieurs minutes suivant les cas. L'expérimentation étant partie intégrante de la *datascience,* cela peut s'avérer frustrant. Ceci conduit certains développeurs de réseaux à envisager d'autres solutions spécialisées comme *Torch* ou *Keras.* Pour illustrer le propos, ci-dessous des extraits de code *Theano* pour instancier une classe RBM. Les variables symboliques de *Theano* sont nommées ici vbias, hbias en lieu et place des paramètres a, b et w de la partie I de cet article.

```
# Paramètres a,b,w de la formule 5 préalablement initialisés dans le code
self.params = [self.W, self.hbias, self.vbias]
[...]
# Energie effective (formule 8) définie par :
  def free_energy(self, v_sample):
    vbias_term = T.dot(v_sample, self.vbias)
    wx_b = T.dot(v_sample, self.W) + self.hbias
    hidden_term = T.sum(T.log(1 + T.exp(wx_b)), axis=1)
    return -hidden_term - vbias_term
[...]
# Implémentation de contrastive divergence pour obtenir un échantillon nv_samples
# distribués selon la probabilité souhaitée
 [...] nv_samples [...] = theano.scan(
    # fonction implémentant 1 step de deep sampling
    self.gibbs_hvh,
    # où chain start, une initialisation astucieuse par tirage d'une des
    # configurations au hasard
    outputs_info=[None, None, None, None, chain_start],
    # Nombre d'itérations limitant le temps d'attente de la convergence
    n_steps=k
)
[...]
# Déclaration du gradient de coût, où chain_end est le dernier élément de la chaîne
# nv_samples obtenu par l'échantillonnage de Gibbs précédent
cc = T.mean(self.free_energy(self.input)) - T.mean(self.free_energy(chain_end))
gparams = T.grad(cc, self.params, consider_constant=[chain_end])
cost = get_pseudo_likelihood_cost(gparams)
[...]
# Entraînement RBM , et calcul de la fonction de coût
train_rbm = theano.function(
    [index],
    # fonction de coût que Theano doit calculer
    cost,
    # nouvelles valeurs de toutes les variables partagées, ici self.params
    updates=updates,
    ,[...])
```

Caffe : une approche « packagée »

A un niveau d'abstraction totalement différent, nous trouvons *Caffe* et son approche très opérationnelle. Il s'agit d'un exécutable dédié aux réseaux de neurones (non profonds) et développé en *C++/Cuda*. L'approche est davantage celle d'un framework, c'est à dire un canevas général offrant une grande souplesse par paramétrage. *Caffe* est nativement utilisable en ligne de commande mais offre également des interfaces (wrappers) en *Python* ou même *Matlab.* Le cas échéant pour une personnalisation plus avancée, la librairie originale en *C++* est accessible et pourra être re-compilée. Ce « moteur » construit ainsi des réseaux de neurones suivant les spécifications définies dans des fichiers plats à l'extension douteuse .prototxt. La syntaxe adoptée, *Protobuffer* (alternative à JSON) s'attache à décrire efficacement des structures d'objets, en texte plein, et donc aisément lisibles par un humain. On y trouve 3 types de déclarations d'objets à choisir parmi un catalogue bien fourni :

- **Blob** : définition des données à manipuler dans le réseau. Nous y trouverons principalement le set de données d'entraînement, la liste de labels/catégories, les données en sorties du réseau de neurones.
- **Layer** : composant fondamental de *Caffe.* Définition du type d'opération à réaliser dans une couche donnée comme les convolutions, les filtres, l'application d'une fonction sigmoïde, le calcul de fonctions de coût usuelles ...
- **Solver** : objet exclusivement dédié au paramétrage de l'entraînement du modèle constitué de l'ensemble des objets « **Layers**« . On y trouve le choix de l'utilisation d'un SGD, le nombre maximal d'itérations autorisées, l'activation du calcul via GPU...

Tout le travail consiste donc avec cette syntaxe « light » à empiler sur un jeu de données (**Blob**) des opérations d'activations (eq. **Layers**) dont la dernière est *a fortiori* une fonction de coût, à savoir un layer de type particulier **EUCLIDEAN_LOSS, SOFTMAX_LOSS, HINGE_LOSS**... Le nombre de neurones par couche est ajusté à l'aide du paramètre **num_ouptut**. En lançant l'apprentissage par invocation du **Solver**, *Caffe* va alors automatiquement itérer des allers-retours sur cette pile de Layers, le retour / back-propagation, se faisant lors du calcul de la fonction de coût. Par la suite, le calcul de gradient assuré de façon transparente par *Caffe* lui permet de réajuster les poids synaptiques. Exemple de déclaration d'une couche d'un réseau de neurones avec *Caffe* :

```
layers {
  name: "ip1"
  type: INNER_PRODUCT  # déclaration d'une couche/layer de neurones totalement connectés
  blobs_lr: 1.  # coefficient de learning rate pour les paramètres « filters »
  blobs_lr: 2. # coefficient de learning rate pour les paramètres « biases »
 # paramètres spécifiques au type de layer
  inner_product_param {
    num_output: 1000  # nombre de neurones
    weight_filler {
    type: "gaussian"
    }
 }
```

A noter enfin que *Caffe* ne possède pas (encore ?) de layer RBM ou Autoencoder spécifique aux réseaux de neurones profonds. Nul doute alors qu'un *Caffe* pour DBN, ou équivalent, sur un mode très packagé, devrait voir le jour tant les espoirs sont grands concernant cette nouvelle classe d'algorithmes. Rendre leur implémentation plus accessible est une voie inéluctable.

Conclusion

Des débats passionnants existent aujourd'hui dans la communauté de l'IA et du machine learning au sujet de la pertinence des architectures profondes dans l'objectif de réaliser une authentique IA. L'avenir est-il à l'imitation de la nature comme semble le penser les partisans des RN ? Les grandes inventions du passé, comme l'électricité ou le moteur à explosion, suggèrent plutôt que les grandes innovations ont tendance à s'en démarquer. Les architectures profondes semblent adaptées à la reconnaissance de formes mais elles ne semblent pas être capables de raisonnement logique. Même s'il est peu vraisemblable qu'une seule classe d'idées comme celles des Hinton et al. puisse venir à bout d'un problème aussi colossal que la conception d'une IA, les réseaux de neurones profonds feront assurément partie des recherches dans cette direction ces prochaines années. ⊞

Créer son premier objet connecté avec Visual Studio

Objets connectés, Internet Of Things, IOT, Arduino, Netduino, etc., sont des mots que l'on entend beaucoup dans la sphère Microsoft et surtout depuis la dernière conférence //BUILD. Mais qu'est-ce que c'est que tout cela ?

Soriya THACH, Michaël FERY,
Jonathan ANTOINE
Consultants .NET - **Infinite Square**

Un objet connecté est un appareil électronique pouvant remplir le rôle de capteur ou d'actionneur. L'une de ses fonctions principales sera notamment l'échange de données avec d'autres appareils : objets connectés, services dans le cloud ou encore applications mobiles. C'est donc une définition englobant un très large spectre d'objets possibles, et vous en connaissez surement certains sans y avoir déjà attribué le nom d'objets connectés :

- Bracelets connectés de suivi de votre santé : Fitbit, Microsoft Band, etc.
- Capteurs intelligents : thermomètre à distance, caméra à distance, balance connectée en Wifi
- Déclencheur à distance : Nokia Treasure Tag, interrupteur à distance via une application mobile,
- Objets télécommandés : drones, voitures téléguidées, etc.
- Identifications : pass Navigo, tag NFC, etc.

En somme, il y a de plus en plus d'objets connectés nous entourant et c'est pour cela que l'on parle d'Internet Of Things (IOT) : un filet (net) d'objets interconnectés plus ou moins intelligents. L'institut Gartner parle d'environ 26 milliards d'objets connectés sur le marché en 2020 !

Le prototypage

Alors voilà, nous sommes inondés d'objets connectés mais en bons bidouilleurs que nous sommes, on souhaiterait créer les nôtres. Que l'on fasse cela dans son garage ou que l'on soit une startup qui souhaite commercialiser un nouveau produit, nous devons tous passer par une étape essentielle : le prototypage. Il s'agit de réaliser une preuve de concept de notre idée. Pas besoin que le produit soit beau ou qu'il ait sa forme finale, on veut juste vérifier que cela fonctionne. La solution est d'utiliser des cartes de prototypage. Il s'agit de plateformes matérielles et logicielles de développement permettant de réaliser des essais de circuits et de programmes que nous industrialiserons plus tard.

Pourquoi Visual Studio ?

Et la programmation de ces cartes dans tout ça ? Rien de plus simple si l'on est déjà un peu codeur. En effet, Visual Studio de Microsoft est compatible avec de nombreux écosystèmes de prototypage. Autre avantage, même en version Express et donc gratuite, l'IDE de Microsoft vous permettra de commencer à prototyper. Aucune raison de s'en priver donc !

Quelle carte de développement choisir ?

De nombreux modèles de cartes de développement sont aujourd'hui disponibles sur le marché : différentes plateformes, différents langages de programmation, différentes configurations hardware... Face à tant de choix et afin de vous aider à choisir votre carte, nous vous présenterons ici quelques modèles qui vous permettront de débuter rapidement et facilement dans le prototypage de vos premiers projets. Afin de vous permettre d'avoir un aperçu rapide de leurs possibilités, vous verrez comment faire clignoter une LED, un bon moyen d'avoir quelques bases pour votre premier objet connecté ! On vous rappelle que l'objectif de l'article est de vous faire un état des lieux des solutions existantes, pas de vous rendre expert. Nous avons donc choisi un scénario simple et reproductible sur toutes les plateformes présentées.

Arduino

L'Arduino est l'une des plateformes de référence dans le prototypage électronique, et se base sur un simple microcontrôleur qui vous permettra de débuter facilement dans le milieu du développement embarqué. Toute la plateforme, dont le hardware, étant open source, il existe de nombreuses cartes Arduino. Allant de la moins chère à 20€ à plus de 200€ pour les plus complexes, la carte que vous choisirez dépendra surtout de ce que vous en ferez. Celle que nous vous conseillons pour débuter est « l'Arduino Uno » dont le prix avoisine les 20€ sans compter les composants dont vous aurez besoin.

Le concept de Shield

Si vous n'êtes pas familier avec les circuits électroniques, vous pouvez vous procurer des shields (ou cartes d'interface) qui vous permettront d'ajouter des fonctionnalités à votre carte sans faire aucun branchement ! Ces cartes ont en général la même taille et le même nombre de ports que votre carte, et s'empilent directement dessus. Vous pouvez trouver par exemple des shields pour l'affichage comprenant un écran LCD et les composants nécessaires à son fonctionnement ; des shields pour la communication avec un port Ethernet ou un module Bluetooth ; et même des shields plus complets avec des joysticks et sorties vidéos pour diffuser un jeu sur un écran. Vous pouvez par ailleurs utiliser plusieurs shields sur une même carte mais il faudra faire attention aux branchements d'un shield à l'autre.

Faire clignoter une Led

Pour le circuit, vous aurez besoin de brancher une LED et sa résistance en série sur la carte Arduino. D'un côté vous relierez la LED au port 2 de votre carte et l'autre borne de la résistance à la masse de votre carte (GND).
Passons au code, commençons par allumer une LED :

```
[code]
  //Définition du port de la carte correspondant au branchement de la LED
  const int portLED = 2 ;
  //Configuration en sortie pour le port de la LED
  pinMode(LED, OUTPUT) ;
//Création d'une boucle qui va alterner l'état Haut (ON) et Bas (OFF) avec un délai de 500ms
entre chaque état et ainsi faire clignoter la LED
  while(true)
{
    digitalWrite(LED, HIGH) ;
    delay(500) ;
```

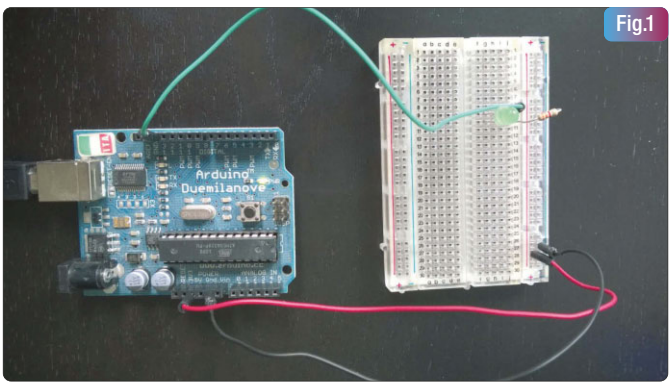


Fig.1

```
    digitalWrite(LED, LOW) ;
delay(500) ;
}
[/code]
```

Fig.1.

Netduino

Produite par la société SecretLabs, Netduino est une plateforme permettant le développement en C# avec le micro framework .NET. Plusieurs modèles de cartes Netduino sont aujourd'hui disponibles sur le marché. Entre autres, vous pouvez essayer la « Netduino Plus 2 » qui est compatible avec les shields Arduino et possède un port Ethernet ainsi qu'un port de carte micro SD. Pour ceux qui sont plus réticents à l'électronique, vous pouvez aussi tester la « Netduino Go » qui vous permettra d'éviter les branchements et circuits électriques en branchant directement des modules « préassemblés » sur la carte comme des LEDs, interrupteurs, écrans LCD...

Faire clignoter une LED

Le branchement à réaliser est le même que pour la carte Arduino. Eventuellement, si vous n'avez pas de LED/Résistance, vous pouvez utiliser la LED directement intégrée sur la carte. Pour le code :

```
[code]
//Définition du port de sortie correspondant à la LED : Pins.ONBOARD_LED pour la LED de la carte ou Pins.GPIO_PIN_D2 la LED branchée sur le port 2 de la carte
  var portLED = new OutputPort(Pins.ONBOARD_LED, false) ;
// Création d'une boucle qui va alterner l'état Haut (ON) et Bas (OFF) avec un délai de 500ms entre chaque état et ainsi faire clignoter la LED
while(true)
{
  portLED.Write(true) ;
  Thread.Sleep(500) ;
  portLED.Write(false) :
  Thread.Sleep(500) ;
}
[/code]
```

Fig.2.

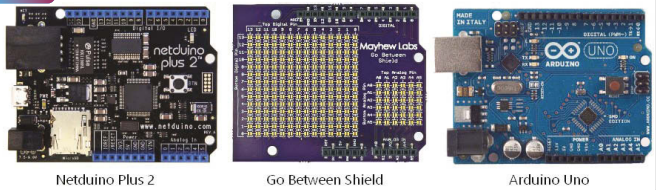
Gadgeteer Fig.3.

Gadgeteer est une partie open source et open hardware du projet .NET Micro Framework. Il s'agit d'un ensemble de cartes et de modules compatibles que chacun peut créer ou acheter. Avec ces cartes de prototypage, la promesse est de nous éloigner encore un peu plus des considérations électroniques pour se concentrer sur la partie code.

Aide au branchement et au montage

Son intégration dans Visual Studio est très complète puisqu'elle fournit des templates de projet mais surtout un designer automatisant la connexion des modules à la carte de manière visuelle Fig.4. Une fois vos composants branchés sur la carte dans la partie « surface de dessin » de Visual Studio, les objets correspondants seront automatiquement disponibles dans votre code. Aucun besoin de définir explicitement sur quelle entrée ou PIN les données devront transiter. De plus, avec l'IntelliSense très bien intégrée dans Visual Studio, vous découvrirez rapidement que les classes correspondantes possèdent des propriétés et méthodes très avancées.
Voyons justement dans la pratique comment faire clignoter notre LED.

Fig.2




```
[code]
// This method is run when the mainboard is powered up or reset.
void ProgramStarted()
{
   multicolorLED.BlinkRepeatedly(Colors.Blue);
}
[/code]
```

Fabricants

Plusieurs fabricants existent mais la référence reste GHI Electronics avec ses cartes FEZ et son Starter Kit comprenant de nombreux modules. Idéal pour débuter. Et qui sait, peut-être fabriquerez-vous aussi vos propres cartes et modules ?

Où acheter ?

Si vous vous êtes décidé à prendre une carte et que vous ne savez pas où l'acheter, voici une liste exhaustive de sites sur lesquels vous pourrez vous la procurer. En fonction des sites, vous trouverez les composants et/ou les cartes que vous cherchez :

- http://www.sparkfun.com : vous trouverez tout ce qui est relatif à Arduino sur ce site (attention aux frais de douane, cela vient de l'étranger certaines fois !)
- http://www.lextronic.fr : il y a des cartes mais sont cependant plus orientés composants.
- http://www.conrad.fr : c'est une référence pour la vente de composants électroniques.
- http://boutique.semageek.com : il s'agit d'un distributeur officiel d'Arduino en France.
- http://www.robotshop.com : vous trouverez plutôt des « kits » sur ce site.
- https://www.ghielectronics.com/ : il s'agit du fabricant de cartes et composants Gadgeteer.

Et demain ?

Nous avons fait ici un état des lieux des solutions de prototypage les plus communes sous Visual Studio mais rien n'est figé. En effet, de nouveaux acteurs veulent associer leur nom aux objets connectés. L'exemple marquant le plus récent est celui d'Intel et de sa carte Galileo compatible avec les shields Arduino mais hébergeant un véritable OS. Les annonces de la BUILD de Microsoft annonçaient également un retour de l'implication de la firme de Redmond dans le développement du Micro Framework. Les promesses sont tenues et des nouvelles versions du SDK ont été dévoilées ces dernières semaines. Le prototypage prend un nouvel essor, de nombreux appareils sont dévoilés et des APIs s'ouvrent chaque semaine. De quoi ravir les bidouilleurs que nous sommes.

Code des articles : https://github.com/claustres

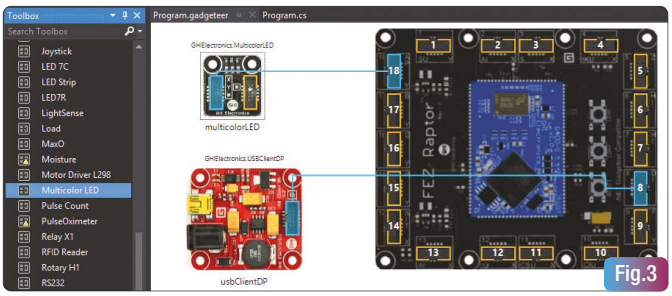


Fig.3

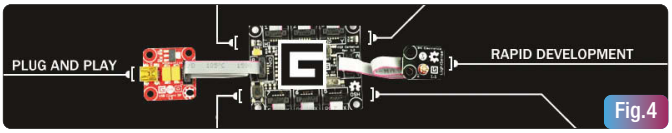


Fig.4

Faut-il ou non passer sur Drupal 8 ?
Des éléments concrets pour une prise de décision éclairée.

Alors que la Drupalcon de Barcelone s'achève enfin sur une date de sortie de la première Release Candidate au 7 octobre dernier, la question du choix de version entre Drupal 7 et Drupal 8 lors du lancement d'un nouveau projet est plus que jamais d'actualité.

Marine Soroko,
directrice associée de Core-Techs

En effet, pourquoi faire le choix du passé (Drupal 7) pour un projet qui démarre, quand on sait que la moyenne de vie d'un projet classique est de 2 à 4 ans ? Les quelques semaines / mois qui nous séparent d'une version opérationnelle en production sont-elles un obstacle réel quant à la réussite d'un nouveau projet, censé vivre plusieurs années ?

A première vue, un néophyte de Drupal et de sa communauté apporterait certainement une réponse négative à cette question.

Pourtant, la réponse est loin d'être triviale, voyons pourquoi.

Utiliser une version bêta en production fait courir un risque certain au projet déployé

Que vous décidiez de migrer un existant vers Drupal 8 ou lancer un projet ex-nihilo en Drupal 8 dans sa version bêta est source d'erreurs critiques et peut-être irrécupérables.

En effet, le lancement de la première « release candidate » n'est possible que lorsqu'aucun bug critique sur le cœur de Drupal ne subsiste.

Or, potentiellement, tout bug critique comporte un risque de perte ou d'altération des données. Dryes Buytaert lui-même déconseille tout usage de Drupal 8 en production tant que la première RC n'est pas lancée.

Cette recommandation est également valable par le fait que toutes les questions de sécurité sont publiquement discutées : aussi, toute faille de sécurité est potentiellement simplement exploitable par n'importe qui : c'est évidemment une source de vulnérabilité supplémentaire pour votre projet.

Cependant, des membres de la communauté, souvent considérés comme des experts, ont déjà réalisé ou réalisent en ce moment des projets sous Drupal 8, en considérant justement que la promesse technologique de l'outil et sa viabilité à long terme justifient les risques encourus. Il faut considérer que, dans la plupart des cas :

- Les projets lancés sont relativement simples ;
- Ils redéveloppent pour la plupart de grands pans fonctionnels actuellement couverts par des modules communautaires en Drupal 7 ;
- Ils sont surtout encadrés par d'excellents connaisseurs de la communauté.

Ces derniers :

- Sont ainsi capables d'appliquer des patches d'une version bêta à une autre (en effet, les outils de migration d'une version bêta – upgrade path - à une autre ne sont pas à la portée de n'importe qui) ;
- Suivent particulièrement les avancées de la communauté et leur veille active leur permet d'être vigilants sur d'éventuelles failles ou anomalies critiques qui compromettraient le fonctionnement du projet ;
- Peuvent directement intervenir dans le processus même de développement de ces versions bêta.

Drupal sans ses modules communautaires reste une coquille vide

Tout utilisateur ayant déjà déployé Drupal sait déjà que l'installation de base de Drupal n'est qu'une infime partie du travail d'installation. Pour faire fonctionner un site basique, un minimum d'une vingtaine de modules doivent être installés, paramétrés et configurés, et dans des sites plus conséquents, plus d'une centaine de modules sont fréquemment utilisés. Cette modularité, force de Drupal, selon une logique n tiers, repose sur l'intense participation d'une communauté particulièrement dynamique. Mais la richesse de cet écosystème a pour contrepartie une lente adaptation aux montées de version. En effet, les modules sont directement dépendants d'un numéro de version, et l'écosystème de ces modules ne peut évoluer que lorsque l'API d'une nouvelle version est stable. Pour Drupal 8, cette API n'a été stable qu'à partir de ...xxx. Par ailleurs, le développement ou l'adaptation des modules suit souvent le besoin des projets. Or, peu de projets étant actuellement développés en Drupal 8, les modules ne sont que très partiellement adaptés à cette nouvelle version. Ce cercle vicieux ne trouvera une issue que lorsqu'un nombre croissant de projets se lanceront sous Drupal 8.

Et, si l'on regarde l'état du développement de quelques-uns des modules principaux, le tableau reste particulièrement contrasté pour un projet existant ou à venir :

- Modules disponibles et réutilisables :
  - Features : module opérationnel, mais finalement moins indispensable que celui accessible dans sa version 7 en raison du fichier de configuration ;
  - Taxonomy menu : une version stable est actuellement disponible ;
  - Feeds ;
  - Simplenews ;
  - Rules est un des modules avancés sous D8. La dernière version de septembre 2015 apporte de nombreuses avancées mais il n'y a pas encore d'interface utilisateur disponible ;
  - Panels : une version alpha est disponible depuis août 2015 avec une dernière anomalie critique.
- Modules encore indisponibles, en cours de développement ou partiellement disponibles :
  - Media ;
  - Metatags ;
  - SearchAPI ;
  - Webforms : développements en cours, mais la communauté a quelques difficultés pour avancer suffisamment.

Le site http://www.bluespark.com/status-top-100-contributed-modules-drupal-8 donne un bon aperçu de l'état d'avancement des modules princi-

paux. Démarrer sur Drupal 8 alors que les modules que vous projetez d'utiliser ne sont pas disponibles ou insuffisamment développés vous laisse deux alternatives :

- Accepter d'utiliser des modules partiellement inutilisables et faire les mises à jour fournies par la communauté progressivement, sans planning fiable de disponibilité de ces modules,
- Redévelopper une partie de ces modules, avec exclusivement les fonctionnalités indispensables.

Dans les deux cas, votre planning projet en prend un sérieux coup.

Si votre commanditaire est familier de la communauté Drupal, connaît les enjeux de la version 8 et est prêt à passer plus de temps sur le développement de son projet avec la version 8 (pour certes au final épargner le coût d'une éventuelle montée de version D7 vers D8), alors le jeu en vaut certainement la chandelle.

Mais hélas pour nous, agences, développeurs et prestataires de service, nos clients sont souvent très éloignés des considérations de la communauté Drupal : il est difficile de faire entendre qu'une version est sur le point de sortir depuis 2 ans (!), et qu'en plus, quand elle sera prête, il faudra payer un peu plus cher, ou accepter de décaler le planning.

La situation est certainement pire dans le cas d'une migration vers Drupal 8 : vous aurez alors affaire à des contributeurs familiers de Media, exploitant des restitutions avec Panels, déployant à tout va des modules avec Webform, améliorant le référencement avec Metatags, ... et qui, sous prétexte d'une montée de version dont ils ne saisissent pas toujours la portée, ne pourront plus exploiter ces fonctionnalités ! Et l'amélioration ergonomique du back-office ou la contribution Responsive ne feront pas le poids dans l'analyse avantages / inconvénients de cette migration.

Et, si l'on compare la courbe d'adoption de Drupal 7 et de ses modules en 2011, on peut considérer que 6 mois sont nécessaires après la première Release Candidate pour obtenir un socle minimal de modules stables et utilisables.

L'évolution technologique de Drupal 8 nécessite de maîtriser de nouvelles compétences

Drupal 7 repose sur une logique n-tiers, essentiellement développée en PHP procédural, reposant sur un framework spécifique et dont l'extensibilité (via les hooks) repose sur une logique particulière.

Pour travailler avec Drupal 8, les développeurs devront être plus techniques et notamment :

- Maîtriser Symfony (quoique partiellement : l'usage de Symfony dans Drupal 8 reste très limité) ;
- Maîtriser le framework de templating Twig ;
- Maîtriser la programmation orientée objet et les derniers standards PHP ;
- Maîtriser la nouvelle API Twig.

La courbe d'apprentissage est conséquente pour certaines catégories de développeurs ou d'intégrateurs Drupal : la création d'un projet sous Drupal 8 ou sa migration, au-delà de la nécessaire fiabilité des supports logiciels (cœur et modules) nécessite des développeurs formés et aguerris.

Et les premiers projets sous Drupal 8 souffriront proportionnellement des lacunes des développeurs sur cette nouvelle mouture.

Aussi, lancer un projet Drupal 8 doit passer au préalable par un programme de mise à niveau ou de formation des équipes de développement, qui doivent être particulièrement motivées pour faire face aux nécessaires écueils qu'elles rencontreront.

Les avancées technologiques de Drupal 8 peuvent faire pencher la balance dans certains contextes projet

On l'a déjà dit, si votre projet est simple, que l'enjeu du projet n'est pas critique, et que vous pouvez souffrir de quelques retards, mais que votre équipe est motivée et accompagnée, le choix de Drupal 8, à quelques jours d'une release candidate, peut être envisagé.

Ce choix peut se trouver renforcé si les nouveautés technologiques de Drupal 8 constituent un élément différenciant dans votre projet :

- Votre projet est multilingue, avec une gestion complexe de différentes langues. Drupal 8 change radicalement l'approche du multilinguisme sous Drupal : si vous avez eu à mener un projet multilingue sous Drupal 7, vous avez connu les affres des multiples modules concurrents et non totalement satisfaisants. Avec Drupal 8 les modules liés à l'internationalisation sont intégrés au cœur : les interfaces et les vues sont nativement multilingues, les modules sont directement traduits depuis les dépôts de la communauté,
- Votre projet s'inscrit dans une stratégie de « continuous delivery », avec des cycles de déploiement très courts, nombreux et idéalement intégrant une démarche Devops. Avec Drupal 7, déployer une telle agilité était complexe, et la seule utilisation de Features n'y suffisait pas. Désormais, Drupal 8 embarque un gestionnaire de configuration s'appuyant sur le standard YAML : les déploiements automatisés s'envisagent donc beaucoup plus simplement.
- Votre projet est composé de nombreux développements spécifiques, fortement interconnectés avec des systèmes d'information externes. La suite de Web services fournie par Drupal 8 en RESTful permet désormais d'exposer les vues : au-delà d'être un simple fournisseur de données, Drupal devient désormais un fournisseur de services.

Si vous exploitez Drupal 6, cette version ne sera plus supportée par la communauté

La communauté Drupal ne maintient que 2 versions de Drupal en parallèle.

Aussi, l'arrivée de Drupal 8 signifie la fin du support de la version 6, dans les trois mois qui suivent la sortie de Drupal 8. La migration vers Drupal 8 s'impose donc rapidement, pour continuer à bénéficier du support de la communauté, essentiel dans le suivi du bon fonctionnement et de la sécurité du cœur de Drupal.

Pour conclure

Le choix de la migration ou de l'utilisation de Drupal 8 pour un nouveau projet est loin d'être simple. Si ce choix doit être déconseillé avant la première RC (sauf à disposer dans vos équipes d'un crack de la communauté et avoir un projet qui exploite peu les fonctionnalités classiques d'un CMS – mais pourquoi choisir Drupal alors dans ce dernier cas ?), la réponse est positive dès la sortie de la RC, dans le cas où vous répondez oui à toutes les questions ci-dessous :

- Vos équipes sont compétentes, formées et motivées pour passer sur Drupal 8 ;
- Votre projet n'est pas critique ou peut souffrir de quelques retards ;
- La durée de vie de votre projet est supérieure à 3 ans ;
- Votre projet ne nécessite pas impérativement certains modules dont la date de sortie est inconnue, lointaine ou ces modules peuvent être redéveloppés partiellement avec un impact acceptable en termes de coûts et le planning ;
- Votre projet est une refonte d'un site sous Drupal 6 ;
- Votre projet est particulier dans le sens où de nombreux développements métiers doivent être réalisés, en forte interconnexion avec des systèmes tiers et dans une logique multicanal : la programmation objet, l'introduction de Symfony et la nouvelle API RESTful sont des avantages certains dans ces contextes complexes

Quand Xamarin révolutionne le monde du développement multiplateforme

Cyril CATHALA
Expert Pôle Microsoft
et Microsoft MVP - SOAT

Très dynamique, le marché du développement multiplateforme et des applications mobiles connaît de nombreuses mutations pour répondre toujours plus rapidement aux attentes des utilisateurs. Dans ce contexte, il nous paraît important de faire un focus particulier sur les apports de la plateforme Xamarin qui est en train de créer une véritable petite révolution sur le marché.

Xamarin, une plateforme de développement multiplateforme permettant de lancer rapidement ses projets mobiles

Concrètement, Xamarin fournit une plateforme complète. Que ce soit pour mutualiser les développements, tester les interfaces graphiques ou diagnostiquer les bugs, Xamarin intègre les outils nécessaires pour mettre en œuvre une démarche qualité. Reste que le développement d'applications avec Xamarin peut prendre deux formes : l'approche traditionnelle et l'approche Xamarin Forms, qu'il faut bien distinguer.

L'approche traditionnelle de Xamarin

Elle permet de développer des applications natives au plus près de chaque plateforme (Android, iOS, Windows et OS X) et dans le langage C#, permettant productivité et partage de code.

Au même titre qu'une application développée avec les outils et langages "natifs", il est possible d'appeler les fonctions du système, débloquant ainsi toutes les possibilités et de communiquer par exemple avec une Apple Watch sur iOS, ou intégrer un widget sur l'écran d'accueil d'Android. L'interface graphique respecte le design et l'expérience utilisateur du système, mais nécessite la réalisation d'écrans spécifiques à chaque OS, pour s'adapter au mieux au design de l'OS.

Bien que les compétences .NET soient réutilisées, il ne faut pas négliger la montée en compétence pour maîtriser les concepts de chaque plateforme, même si de nombreux concepts généraux sont identiques.

Xamarin Forms

Xamarin propose une extension à son approche traditionnelle : un framework qui unifie le développement d'interfaces graphiques. On écrit ainsi le code régissant l'interface une seule fois, à un endroit unique, puis Xamarin Forms procède ensuite à la traduction afin qu'un élément graphique ait l'apparence adéquate sur chaque plateforme (iOS, Android et Windows).

Cette interface peut être écrite dans un format très similaire au XAML, bien connu des développeurs Windows (WPF, Windows Phone, etc.). Ce format apporte un énorme gain de temps en formation et en productivité.

Attention toutefois, l'intérêt de cette approche reste limité à des applications relativement simples ou orientées métier. Si l'application doit avoir une interface complexe, bien différente d'une plateforme à l'autre, ou si elle fait appel à de nombreuses fonctionnalités du système, il vaudra mieux privilégier l'approche traditionnelle.

Les autres composants de Xamarin

L'univers Xamarin est composé d'une multitude de produits contribuant à développer et renforcer une démarche qualité. Parmi ces dernières, nous pouvons mentionner :

Xamarin Test Cloud permet d'exécuter des tests automatisés des interfaces graphiques sur plus de 1 600 téléphones et tablettes compatibles iOS et Android.

Xamarin Insights permet de récolter des rapports de crash et d'analyser le parcours utilisateur. Le monitoring se fait via des rapports très détaillés et surtout adaptés au contexte Xamarin.

Xamarin University permet de se former en ligne et d'aborder tous les sujets fondamentaux autour de la mobilité. C'est également un moyen de valider ses connaissances au travers d'une certification.

Anticiper l'avenir

La plateforme Xamarin ne se contente pas de fournir des outils pour construire des applications mobiles, mais permet également d'adresser les "wearables" dont notamment Android Wear et Apple Watch.

Xamarin a su se projeter dans l'avenir, détecter les opportunités et investir sur un périmètre bien plus large que celui de la mobilité. On pense bien évidemment à celui des objets connectés, domaine dans lequel tous les grands acteurs investissent et sortent des OS adaptés.

Une solution pour tout ?

Xamarin peut-il répondre à toutes les attentes ? La solution idéale n'est malheureusement pas de ce monde et il est uniquement question de compromis. Bien sûr, des solutions pour applications hybrides Web telles que PhoneGap ou ionic existent, mais supposent des compromis en termes d'expérience utilisateur et surtout de performance. Cette approche est intéressante dans plusieurs cas : design strictement identique pour chaque OS, fonctionnalités limitées, peu d'exigences en termes de performance.

Tenter de mimer un design ou une expérience native en Web peut être dangereux et souvent source de frustration, tant pour le développeur que pour l'utilisateur. Il est donc important de bien jauger les différentes approches possibles.

Pour des applications de qualité sans compromis, Xamarin s'avère l'une des solutions les plus adaptées. Performance, fonctionnalités, expérience native, outillage, tout est là. Pour des applications orientées business ou à l'interface peu complexe, Xamarin Forms est une option favorable, dans la mesure où l'on peut atteindre jusqu'à 100% de partage de code sur toutes les plateformes.

En environnement Microsoft, Xamarin gagne énormément en intérêt puisqu'il devient alors possible de partager du code entre un projet ASP.NET MVC, une couche métier, ou une application WPF / Windows Universal App. En outre, Xamarin ne se limite pas au mobile et permet le développement d'applications bureau sur OS X.

Conclusion

Xamarin répond à des enjeux sur le long terme. Plutôt que d'avoir recours à une équipe mixte, la solution unifie les compétences, permet le partage de code et simplifie les phases de maintenance, tout en bénéficiant des avantages d'une approche native.

Xamarin devrait donc connaître un avenir radieux, notamment grâce à ses partenariats forts (Microsoft, SAP, IBM, etc.) et ses investissements dans les chaînons indispensables au bon développement d'applications.

Débuts de codeurs

Abonnement : Service Abonnements PROGRAMMEZ, 4 Rue de Mouchy, 60438 Noailles Cedex. - Tél. : 01 55 56 70 55 - *abonnements.programmez @groupe-gli.com* - Fax : 01 55 56 70 91 - du lundi au jeudi de 9h30 à 12h30 et de 13h30 à 17h00, le vendredi de 9h00 à 12h00 et de 14h00 à 16h30. **Tarifs** abonnement (magazine seul) : 1 an - 11 numéros France métropolitaine : 49 € - Etudiant : 39 € CEE et Suisse : 55,82 € - Algérie, Maroc, Tunisie : 59,89 € Canada : 68,36 € - Tom : 83,65 € - Dom : 66,82 € Autres pays : nous consulter.
PDF : 30 € (Monde Entier) souscription sur www.programmez.com

Une publication **Nefer-IT**
7 avenue Roger Chambonnet
91220 Brétigny sur Orge
redaction@programmez.com
Tél. : 01 60 85 39 96

Directeur de la publication & rédacteur en chef : François Tonic

Secrétaire de rédaction : Olivier Pavie

Ont collaboré à ce numéro : S. Saurel

Experts : C. Freier, C. Michaud, D. Caramelo, M-L Thuret, F. Boyad, S. Cordonnier, G. Bouveret, S. Sibué, S. Graziano, F. N'Guessan, P-A Bardina, Y. Cézard, F. Noel, E. Margraff, S. Warin, J. Valloire, W. Chegham, N. Creiche, M. Garcia, L. Claustres, C. DUJARRIC, C. Cathala, M. Alves, P. Lemberger, C. Peugnet, S. Thach, M. Fery, J. Antoine, M. Soroko.

Photos/illustrations : Photo by Pictorial Parade/Archive Photos, Apple, Microsoft, Drupal

Maquette : Pierre Sandré

Publicité : PC Presse,
Tél.: 01 74 70 16 30, Fax : 01 41 38 29 75
pub@programmez.com

Imprimeur : S.A. Corelio Nevada Printing, 30 allée de la recherche, 1070 Bruxelles, Belgique.

Marketing et promotion des ventes :
Agence BOCONSEIL - Analyse Media Etude

Directeur : Otto BORSCHA oborscha@boconseilame.fr
Responsable titre : Terry MATTARD
Téléphone : 09 67 32 09 34

Contacts
Rédacteur en chef : ftonic@programmez.com
Rédaction : redaction@programmez.com
Webmaster : webmaster@programmez.com
Publicité : pub@programmez.com
Evenements / agenda : redaction@programmez.com

Dépôt légal : à parution - Commission paritaire : 1215 K 78366 - ISSN : 1627-0908

Sur abonnement ou en kiosque

Le magazine des pros de l'IT

Mais aussi sur le web

Ou encore sur votre tablette

COMMANDEZ WINDEV 21

OU WEBDEV 21 OU WINDEV MOBILE 21

ET RECEVEZ

LE NOUVEL iPhone 6 S

Jusqu'au 11 Décembre 2015

iPhone 6 S Plus

iPhone 6S Plus **64GB**.
Choix de la couleur sur le site

ou

iPhone 6 S

iPhone 6S **128GB**.
Choix de la couleur sur le site

Ou choisissez parmi:

- 1x **iPad Air 2** Wi-Fi+Cellular 128GB
- 2x **iPhone 5S** 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. Compatible tous opérateurs

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 11 Décembre 2015. 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.

www.pcsoft.fr

Elu **«Langage le plus productif du marché»**

Fournisseur Officiel de la Préparation Olympique