PROGRAMMEZ.COM

PROGRAMMEZ!

Mensuel n°186 - Juin 2015

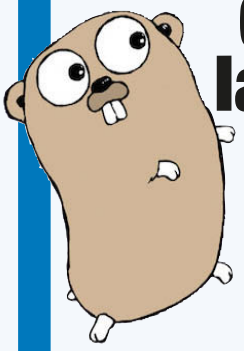
le magazine du développeur

Responsive Design

Créer des sites Web visibles partout

Stock vector © bagiuiani

Drupal & Responsive Design : *les meilleurs amis du monde*



Osez le langage Go

//build/ 2015

Maker Faire® Paris

Le meilleur des conférences développeurs du printemps

Open Source

*Les **licences libres** sont-elles réellement libres ?*

Un poste de développement sous Linux pour tout faire !

Carrière

Une certification est-elle utile ?

Apple Watch

Comment développer des apps avec le WatchKit ?

Microsoft Band

Du code et du sport

M 04319 - 186 - 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

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

Le choix de l'embarras, l'embarras du choix (1)

Depuis avril, les grandes conférences se bousculent : Devoxx France, Maker Faire Paris, F8, BUILD, Google I/O (fin mai) et enfin la WWDC (8-12 juin)... Arrêtez ! Le cerveau des développeurs va finir par exploser...

Rien qu'à la BUILD, Microsoft a définitivement entériné son nouveau visage autour des services, l'ouverture vers d'autres mondes, un Windows qui se veut universel, la mort (enfin !) d'Internet Explorer... La possibilité de générer rapidement (ne disons pas facilement) des apps Android ou iOS a beaucoup fait parler. Certains développeurs Windows Phone y ont vu un constat d'échec du modèle strict Windows Phone et de C# face aux concurrents, mais ces « imports » ne permettent pas de véritables projets mobiles multiplateformes, et c'est là une des grosses faiblesses de cette annonce. Xamarin, et tous les autres, conservent donc leur intérêt. Nous devrions en savoir plus durant l'été. Mais pour Microsoft, il s'agit d'inciter les développeurs à porter leurs apps sur Windows Phone ! De ce point de vue c'est à la fois un échec de la plateforme, car elle ne peut pas attirer à elle seule les développeurs, et aussi un pragmatisme totalement assumé par le nouveau patron : bon OK, on n'y arrive pas, on va changer de stratégie et proposer une migration pure et simple depuis les projets Java et Objective-C (SWIFT n'est pas supporté)...

Mais attention, Google et Apple ne vont pas attendre Microsoft et ses nouveaux outils miracles. Ils avancent rapidement et vont proposer de grosses nouveautés dans les prochains mois. Si Windows 10 est prometteur sur de nombreux aspects, notamment la continuité entre tous les terminaux (qui est partiellement possible chez les concurrents), Windows 10 a tout à prouver, surtout sur le mobile. L'équation est simple : soit Windows 10 version mobile réussit à s'imposer et à faire vendre des smartphones Win10, soit on continue sur la tendance actuelle avec une part de marché locale pouvant être forte, mais globalement faible... L'avenir nous le dira.

Nous ferons un gros retour sur Devoxx qui a été très riche, avec beaucoup de thèmes techniques.

Nous aborderons aussi la programmation Apple Watch avec WatchKit. La montre Apple est sortie depuis un mois. Les apps commencent à arriver, en attendant la possibilité de créer des apps natives (annonce au WWDC ?). Si le SDK est limité, on peut tout de même coder des choses sympathiques.

Ce mois-ci, beaucoup de sujets chauds :
- *Carrière : faire ou ne pas faire une certification ?*
- *Retour sur Maker Faire Paris : vive le maker et le « faire soi-même »*
- *Go : le langage de Google qui pourrait tout casser (ou pas)*
- *Responsive Design : faut-il y passer ? Comment faire du Responsive avec Drupal ?*
- *Un poste de développement sous Linux : non, nous ne sommes pas fous, bien au contraire ! Aurélie nous explique tout.*

Et tout plein d'autres choses !

Soyez développeur, nous avons des cookies.

(1) rayez la mention inutile

à lire dans le prochain numéro n°187 en kiosque le 29 Juin 2015

DRUPAL 8
Drupal 8 arrivera en version finale dès cet automne. Dans ce dossier, nous allons défricher le terrain : les nouveautés, comment préparer une migration de Drupal 7 vers Drupal 8.

CODING4FUN / MAKER
Pour cet été, Programmez! vous propose de nombreux projets makers à faire à la plage, en vacances ! Arduino, Gadgeteer, Raspberry Pi, des robots et des drones !

Coder en s'amusant avec Scratch

PGDay 2015 : 2 juin

Le 2 juin 2015 aura lieu le PGDay 2015 à Belfort, organisé par la communauté francophone de PostgreSQL, et soutenu par l'association PostgreSQL Europe.
Cet événement se déroulera à la Chambre de Commerce et d'Industrie du Territoire de Belfort. Cet événement sera orienté sur des retours d'expériences d'utilisation ou de migration vers PostgreSQL.
Plus d'informations sont disponibles sur le site :
http://select-2-6-2015-as-pgday.org/

Matinale Club (21 : 4 juin

Rupture (21 vous invite à sa nouvelle matinale Club (21, curieuse conférence-atelier, pour découvrir et vivre des ingrédients d'une forme de leadership agile essentiel pour réussir votre transition agile ou digitale. Pour plus d'informations, n'hésitez pas à nous contacter : event@zenika.com

Best of Web : 5 juin

Grâce aux nouvelles plateformes Web comme meetup.com, des communautés nouvelles ou existantes peuvent se retrouver et organiser des événements réguliers.
C'est d'autant plus vrai chez les professionnels du Web qui ont créé de nombreux groupes parlant des derniers Frameworks Javascript, des API REST, de CSS et de Design.
Ces meetups proposent toute l'année des conférences de qualité animées par des développeurs anonymes dans l'écosystème Web où se côtoient développeurs chevronnés, consultants en free-lance, étudiants et contributeurs open source.
Tellement d'événements ont lieu qu'il devient compliqué de pouvoir y assister, les rencontres se déroulant le soir en dehors des heures de travail.
Huit groupes meetups ont donc décidé de se fédérer et de proposer un « Best Of Web » où le meilleur de leurs conférences sera rejoué sur une journée. Ces groupes portent sur des technologies variées du monde du Web :

- AngularJS-Paris
- Backbone Paris
- Paris WebComponents
- EmberJs Paris
- Node.js Paris
- PhoneGap Paris
- D3.js Paris
- Paris JS

Les conférences qui ont été retenues comme « Best Of » sont déjà annoncées sur le site officiel de l'événement http://bestofweb.paris/ et le programme s'annonce très varié.
Citons par exemple l'utilisation D'ES6 avec Angular par Douglas Duteil, une présentation de Polymer par Martin Gorner ou un panorama de l'écosystème REST par Virginie Bardales.
Mais ce n'est pas tout : de nouvelles présentations transverses seront choisies grâce au call for paper ouvert à la communauté.
Best Of Web se déroulera le 5 juin 2015 à l'Hotel de ville de Paris où 500 participants sont attendus.
L'ambition des organisateurs est de proposer l'événement communautaire de l'année, participatif, sur un modèle différent de celui des conférences classiques où seules des stars du développement viennent s'exprimer. Cela sera à coup sûr le rassemblement d'une population de développeurs souvent expérimentés, toujours passionnés ayant en commun leur attachement aux technologies du Web.

Informations pratiques

site : http://bestofweb.paris/
Twitter : https://twitter.com/bestofweb2015
mail : bestofweb2015@gmail.com

Soirées JUG Paris

2 juin : design pattern vs lambda
23 juin : Tools in actions
Site :
http://www.parisjug.org/xwiki/bin/view/Main/WebHome

WWDC 2015 : 8 au 12 juin

La prochaine conférence développeur Apple sera très dense avec l'Apple Watch, des nouvelles de l'Apple TV, les nouvelles machines, les prochains iOS et OS X...
Plusieurs milliers de développeurs sont attendus, des centaines d'étudiants. Comme d'habitude, la Pomme mobilise un millier d'ingénieurs...
Site https://developer.apple.com/wwdc/

18 juin : Darkmira Tour

Viens faire le plein de compétences Zend à l'occasion du premier Darkmira Tour ! Pas de blabla, les experts du Darkmira Tour remplissent ton IDE de solutions pratiques. Conférences, ateliers, goodies... dans une ambiance conviviale !
Darkmira fête à sa manière les 20 ans de PHP et pour cette mémorable occasion organise son premier Darkmira Tour dans les locaux de l'ESGI le 18 juin 2015.
Dans cette première édition, Zend sera à l'honneur. Cette année, Zend est très présent dans l'actualité de Darkmira :

- Cyrille Grandval est devenu membre de la Z-team
- Le livre « Préparation à la certification Zend Certified PHP Engineer (ZCPE) » aux éditions ENI coécrit par Cyrille Grandval et Julien Charpentier (associés chez Darkmira)
- 100 % des développeurs Darkmira sont devenus certifiés ZCPE

Quand ? Le 18 juin de 13 h 30 à 19 h 30
Où ? Ecole ESGI — 242 Rue du Faubourg Saint-Antoine, 75012 Paris
Plus d'informations : http://tour.darkmira.fr

20-21 juin : attention, c'est la nuit du hack !!!

Ame sensible passez votre chemin. La nuit du hack revient les 20 et 21 juin, pour la 13e année... Comme chaque année, il y aura des ateliers, des conférences, la crash party, les challenges !
Préparez-vous dès maintenant :
https://www.nuitduhack.com

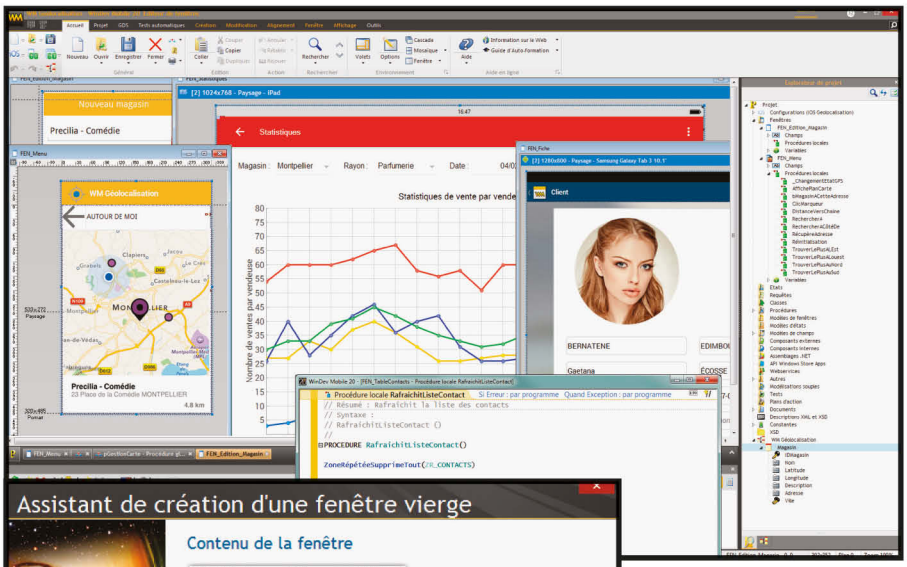
Conférence UNITE, 24-25 juin

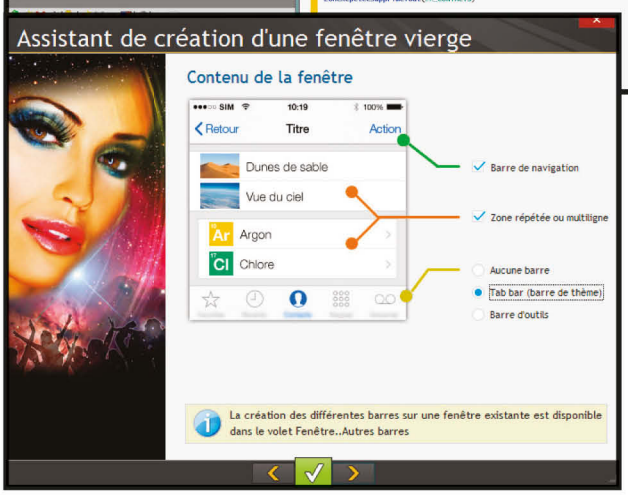
Unity, éditeur du moteur 3D, sera à Amsterdam les 24 et 25 juin pour sa grande conférence annuelle. Une occasion de parler techniques, de voir les dernières nouveautés et les futures évolutions, mais aussi de discuter avec les partenaires. Il n'est pas trop tard pour prendre sa place !
Site : http://unity3d.com/unite/europe

18-19 juin : Hackinparis (5e édition)

Le hacking revient en force à Paris pour un des événements phares de l'année : Hackinparis ! Cette année encore, la conférence s'annonce très riche avec des ateliers, des sessions (très) techniques, des intervenants prestigieux. On va y parler DDOS, bootkit, intrusions diverses et variées, X11, le hack de bracelets sportifs, etc.
Réservez dès maintenant votre agenda et votre place ! site :
https://www.hackinparis.com/home

WINDEV MOBILE 20
LE DÉVELOPPEMENT NATIF
POUR TOUS LES MOBILES





PORTABILITÉ DE VOS APPLICATIONS

ANDROID, IOS, WINDOWS PHONE, WINDOWS MOBILE & CE

Avec WINDEV Mobile 20, une même application peut fonctionner sous les différents OS mobiles: iOS (iPhone, iPad), Android, Windows CE & Mobile, Windows Phone... Recompilez !

TOUS LES TYPES DE MOBILES

Développez pour tous les mobiles: téléphones, smartphones, pocket PC, terminaux, terminaux durcis, terminaux industriels, tablettes, netbook,...

Un environnement de développement complet, intégré, adapté au monde du «mobile»

CRÉEZ DES APPLICATIONS NATIVES POUR TOUS LES SYSTÈMES MOBILES

WINDEV Mobile 20 permet aux professionnels du développement de créer facilement des applications natives pour tous les mobiles: smartphones, tablettes et terminaux industriels. Et si vous possédez un existant WINDEV ou WEBDEV, vous pouvez le ré-utiliser.

UN ENVIRONNEMENT DE DÉVELOPPEMENT AUTONOME

Quels que soient le matériel cible et le système d'exploitation, la méthode de développement avec **WINDEV Mobile 20** est similaire.
L'environnement de développement est intégré, puissant, complet, intuitif, et il est adapté aux spécificités des mobiles.
Avec ou sans base de données, avec ou sans connexion au S.I. il n'a jamais été aussi facile de développer professionnellement sur mobile.

LE CYCLE DE VIE COMPLET EST GÉRÉ

WINDEV Mobile 20 est livré en standard avec tous les outils qui permettent de gérer le cycle de vie des applications: Générateur de fenêtres, Langage L5G, Débogueur, Générateur de rapports, Générateur d'installations, mais aussi Générateur d'analyses Merise et UML, Tableau de Bord du projet, Gestion-naire de Sources collaboratif, Générateur de dossier de programmation, Suivi des plannings,...

PROGRAMMEZ EN L5G: 90% DE CODE EN MOINS

Le langage de 5ème génération WLangage permet de développer plus vite qu'avec un langage traditionnel.
Ses fonctions évoluées rendent le code facile à écrire et à lire, facilitent à la fois le développement et la maintenance.

VERSION EXPRESS GRATUITE
Téléchargez-la !

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

Fournisseur Officiel de la Préparation Olympique

www.pcsoft.fr
Des centaines de témoignages sur le site

Faut-il une certification pour être un développeur heureux ?

Ce n'est pas la première fois que nous parlons de certification dans Programmez !. Régulièrement, nous sommes interrogés sur cette question par des développeurs : faut-il une certification ?
La réponse est clairement non. La certification n'est pas un sésame miracle qui propulsera votre profil au sommet ni n'améliorera forcément votre carrière. Être un bon développeur (pas uniquement être bon codeur) sera votre meilleur atout. La certification est parfois vue comme nombriliste et inutile. Elle peut toutefois « rassurer » des entreprises ou dans des équipes. Et sont parfois exigées pour répondre à des appels d'offres…

En questionnant des développeurs, nous avons eu beaucoup de commentaires : comment juger les compétences d'un développeur sur des QCM, c'est juste une validation de la maîtrise de la théorie, quelle valeur a la certification, etc.

Certaines entreprises peuvent demander des certifications aux salariés ou demander à les passer régulièrement pour avoir un « beau » profil. Trop de certifications tuent-elles la certification ?

La certification sanctionne, la plupart du temps, une connaissance théorique et non une expérience. En clair, on peut dire qu'une certif' nous dit si on connaît l'outil ou un environnement. Ce n'est pas 100 questions qui vont dire si vous savez utiliser et coder tel langage ou tel framework.

Nous pensons donc qu'il est souvent plus intéressant de suivre des formations techniques si on veut réellement progresser ou renforcer une compétence. Cependant, la certification peut vous aider à évaluer vos connaissances, surtout quand la technologie évolue rapidement. Dans certains secteurs, notamment l'industrie, la certification est obligatoire, notamment à cause des normes de sécurité. *La rédaction*

Les avantages d'être certifié Zend Certified PHP Engineer

Cyrille Grandval
CEO Darkmira & Darkmira Brasil
Auteur de "Préparation à la certification ZCPE" (éditions ENI) et membre de la Z-Team de Zend, Cyrille est professeur, consultant et formateur en PHP et sécurité des applications web.

L'écosystème PHP propose des certifications pour la plupart des frameworks : Zend Framework, Symfony, Laravel et bien d'autres. Néanmoins, il n'existe qu'une certification pour le langage PHP lui-même qui est reconnue à travers le monde par les professionnels et les communautés. Cette certification c'est la Zend Certified PHP Engineer, alias ZCPE.

Pourquoi devenir un certifié PHP ?

Premièrement, avant de vous expliquer les rouages de la ZCPE, il est important d'expliquer les avantages indéniables qu'apporte cette certification à un développeur PHP. Créée par la société de Zeev Suraski et Andi Gutmans, La PHP Company, cette certification bénéficie de l'appui de Zend, mais également de la communauté PHP, de leurs visibilités et de leurs crédibilités. Elle permet ainsi de confirmer un haut niveau d'expertise technique chez un développeur PHP. En effet, être certifié, c'est démontrer son professionnalisme et ses connaissances du langage sur tous les thèmes de la certification. Elle apporte donc pour un développeur un réel gain en termes de visibilité et d'attractivité sur le marché du travail PHP. En effet, lorsque vous obtenez la certification, vous entrez ainsi dans le cercle fermé des quelques milliers de développeurs mondiaux à pouvoir apposer sur leur CV, leur site, leur carte de visite, réseaux sociaux, ... le titre de **Zend Certified PHP Engineer** ainsi que le logo de la certification.
Pour les recruteurs, cette certification devient de plus en plus un pré-requis à l'embauche puisqu'elle justifie de la qualité d'une candidature et rassure quant à la bonne possession de hautes connaissances en PHP.
Et bien sûr, concernant le salaire, il est indéniable que la certification apporte un certain poids dans la négociation.
En plus de cela, la société Zend fournit également plusieurs avantages professionnels et funs aux nouveaux certifiés.

- Une licence gratuite de Zend Studio
- Une page de profil sur le ZCE Directory (plus communément appelé les Yellow Pages). Cet annuaire met à disposition les pages profils des certifiés PHP et Zend Framework depuis les premières versions de ces 2 certifications.
  Cela permet ainsi de booster votre référencement, d'augmenter encore plus votre visibilité et de fournir vos contacts aux recruteurs et employeurs à la recherche de profils PHP experts.
- La possibilité d'entrer dans le groupe Linkedin très "select" des Zend Certified Engineer.
- Des autocollants du logo de la certification et votre diplôme cartonné de ZCPE directement envoyé depuis les bureaux de Cupertino.

DOMAINES | MAIL | **HÉBERGEMENT** | E-COMMERCE | SERVEURS

~~4,99~~ À partir de **2,99** € HT/mois (3,59 € TTC)*

PACKS 1&1 HOSTING UNLIMITED

NOUVEAU ! SANS LIMITE !

DES POSSIBILITÉS INFINIES POUR VOS PROJETS WEB

Complet

- Espace disque **illimité**
- Sites Web **illimités**
- Trafic **illimité**
- Comptes email **illimités**
- Stockage email **illimité**
- Bases MySQL **illimitées**
- Domaines **illimités** (1 inclus)

Fiable

- Géo-redondance
- Sauvegardes quotidiennes
- 1&1 CDN
- 1&1 SiteLock Basic
- Assistance 24/7

Facile à utiliser

- 1&1 Applications Click & Build : installez des applications comme WordPress et Joomla!® en un clic
- 1&1 Mobile Website Builder

1 MOIS POUR ESSAYER

1 CLIC POUR CHANGER DE PACK

1 APPEL UN EXPERT VOUS RÉPOND

0970 808 911
(appel non surtaxé)

1&1

1and1.fr

*Les packs 1&1 Hosting Unlimited sont à partir de 2,99 € HT/mois (3,59 € TTC) pour un engagement minimum de 12 mois. À l'issue des 12 premiers mois, les prix habituels s'appliquent. Certaines fonctionnalités citées ne sont pas disponibles dans tous les packs. Offres sans durée minimum d'engagement également disponibles. Conditions détaillées sur 1and1.fr.
Rubik's Cube® utilisé avec l'accord de Rubik's Brand Ltd.

Comment passer cette certification ?

Pour passer la certification ZCPE rien de plus facile.

Dans un premier temps, vous devez acheter sur le site officiel de Zend (http://www.zend.com) un voucher d'un montant de 195 dollars.

L'achat de ce bon vous permettra d'obtenir un numéro de licence *Certification - Zend PHP Certification Voucher* qui apparaîtra dans la partie Mon compte du site, onglet Licences. Ce numéro est valable 1 an.

Dans un deuxième temps, il faut vous rendre sur la page dédiée aux certifications Zend du site de Pearson Vue (http//www.pearsonvue.com/ZEND/). En effet, l'examen de la ZCPE est réalisé dans les centres de formation de Pearson Vue partout dans le monde et non dans les locaux de Zend.

Il est donc nécessaire de vous créer un compte et de choisir l'examen **200-550 : Zend Certified PHP Engineer** dans le catalogue.

Puis choisissez un centre près de chez vous et bloquez une date disponible pour ce centre.

Le jour du passage de l'examen, vous devez vous présenter au centre de formation choisi muni de deux pièces d'identité. Vous devrez remettre vos effets personnels comme votre téléphone avant de vous rendre dans la salle où vous passerez le test.

Pour effectuer vos calculs de conversion, vos réflexions sur des portions de codes ou toutes autres opérations qui peuvent vous être nécessaires pour répondre aux questions, le centre vous remettra soit une feuille quadrillée et plastifiée soit un mini tableau blanc.

La certification ZCPE

La certification Zend Certified PHP Engineer est basée sur **PHP 5.5**. C'est la quatrième version de cette certification PHP (ZCE PHP 4, ZCE PHP 5, ZCE PHP 5.3 puis ZCPE PHP 5.5).

En quelques chiffres, la certification c'est :

- **10 thèmes** avec 3 niveaux d'importances
  - Haute importance : Bases PHP, Programmation Orientée Objet, Sécurité,
  - Moyenne importance : Fonctions, Fonctionnalités Web, Tableaux, Chaînes de caractères et motifs,
  - Faible importance : Bases de données, Formats et types de données, Gestion des flux.
- Environ **70 questions en anglais**, de niveaux différents et sélectionnées aléatoirement parmi un pool de questions,
- **90 minutes** pour passer l'examen.

D'une manière générale, les questions de la certification sont plus pratiques que théoriques et présentent des portions de codes simplifiées à analyser pour connaître la sortie ou le contenu d'une variable, à compléter, à modifier, ... Ces questions font appel à l'expérience que vous avez acquise tout au long de votre expérience professionnelle plutôt qu'à une connaissance par cœur des fonctions et classes PHP.

Lors de l'examen, vous aurez :

- Des choix multiples avec une seule réponse attendue,
- Des choix multiples avec plusieurs réponses attendues. Le nombre exact attendu est indiquée dans l'énoncé de la question par *Choose X,*
- Une réponse libre (par exemple un nom de fonction ou de Design Pattern) dans laquelle vous ne devez pas mettre d'espaces, de parenthèses pour les fonctions, de commentaires, ... vous ne pouvez indiquer qu'un seul mot.

Quels sont les moyens pour réviser ?

Même si vous avez une grande expérience professionnelle, il est nécessaire de bien réviser chacun des 10 thèmes de la certification. Être expert seulement sur les 3 thèmes principaux par exemple, ne vous assurera pas d'obtenir la certification. Pour cela, vous pouvez choisir de réviser de manière autodidacte en plus de votre travail en révisant chaque concept PHP à l'aide de portions de codes pour vous entraîner : à comprendre le fonctionnement de PHP et à améliorer votre processing mental (une sorte d'exécution de PHP dans votre tête).

Pour cela, vous pouvez également vous appuyer sur la lecture d'un ouvrage pour vous guider dans vos révisions / votre apprentissage :

- L'utilisation du Study Guide de Zend (que vous pouvez acheter sur leur site) pour vous fournir un fil rouge à suivre pour votre révision à l'aide de points et de rappels. Celui-ci fournit également quelques questions (entre 5 et 10) par thèmes pour vous familiariser avec les questions de la certification.
- Une révision complète des 10 thèmes de la certification à l'aide de l'ouvrage Préparation à la certification Zend Certified PHP Engineer (ZCPE) aux éditions ENI dont je suis un des auteurs. Cet ouvrage organise chaque chapitre en 4 parties avec une définition d'objectifs, un cours théorique commenté par des exemples PHP, une validation des acquis sous forme de questions / réponses et des travaux pratiques vous permettant de manipuler des cours sources pour améliorer votre compréhension de PHP et être préparé à affronter les exemples de codes des questions de la certification.

Enfin, même si cela peut représenter un certain budget (souvent pris en charge par votre entreprise), une formation PHP certifiante de 2 / 3 jours, vous permet d'être en immersion PHP complète pendant ce délai et de profiter d'un formateur pour répondre à vos interrogations. Les quelques formations existantes vous permettent ainsi suivant l'organisme de formation :

- Soit de revoir sous formes de points succincts et d'exemples de questions (comme le Study Guide) les différents thèmes,
- Soit d'apprendre et de réviser à l'aide d'un cours théorique, des manipulations et des mini-tests blancs les 10 thèmes couverts par la certification.

Ces formations permettent ainsi d'acquérir la bonne logique de pensée pour passer et obtenir cette certification.

Une question pour l'exemple

Sans dévoiler complètement le contenu, voici un exemple de questions auxquelles vous aurez à répondre lors de l'examen (cette question, bien entendu, n'est pas une question de l'examen officiel).

Question 1

What will be the value of $var?

```
<?php
$var = strlen("I'm in São Paulo");
?>

1/ 0
2/ 16
3/ 17
4/ 18
5/ FALSE
```

La bonne réponse est bien sûr la réponse 3. En effet, l'environnement par défaut de la certification ne charge pas l'extension mbstring. Ainsi la fonction strlen va comptabiliser 2 octets pour le caractère ã et non un seul. Si l'extension mbstring était activée, il y aurait 2 possibilités :

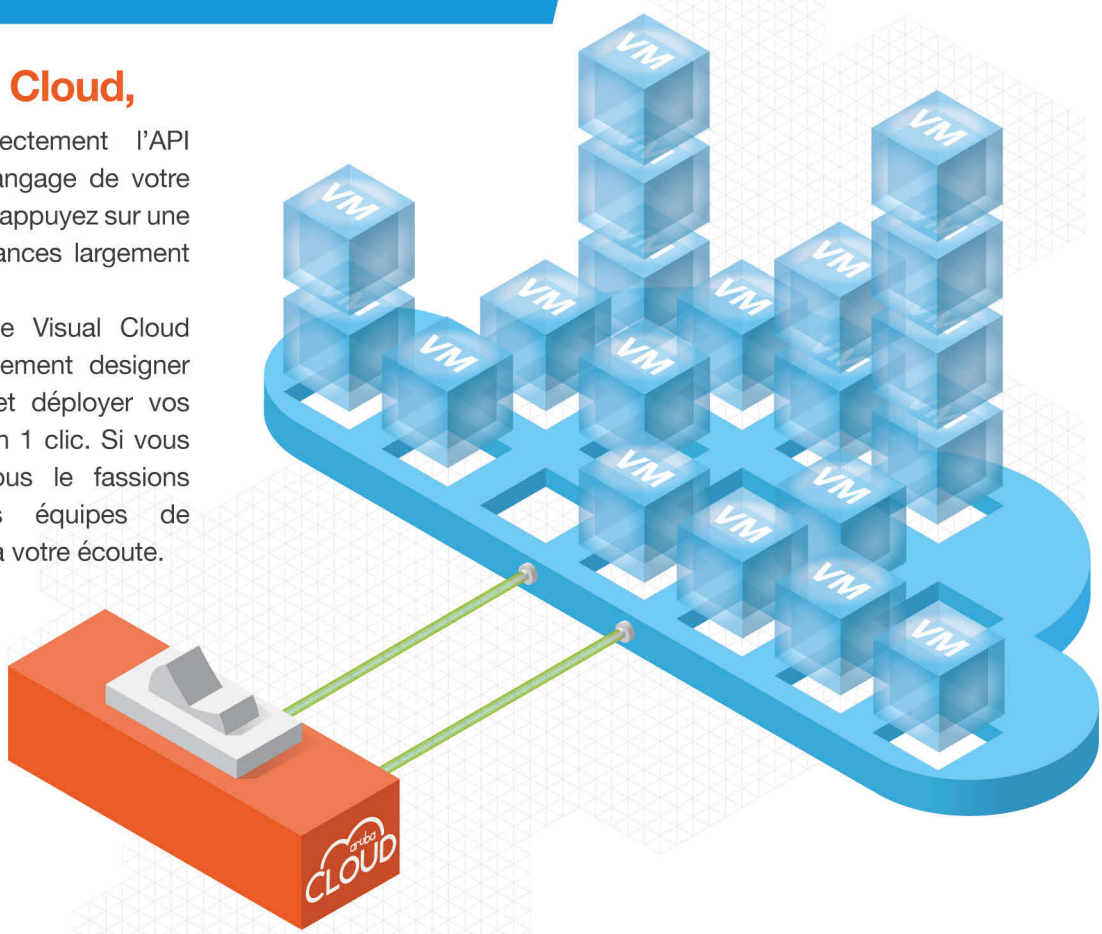
- Utiliser la fonction mb_strlen qui retournerait 16
- Ou renseigner la directive mbstring.func_overload avec la valeur 2 (http://php.net/manual/fr/mbstring.overload.php) du fichier de configuration php.ini pour remplacer automatiquement et à la volée les fonctions de traitements de chaînes par les fonctions mbstring. Ainsi, il ne serait pas nécessaire d'utiliser mb_strlen pour obtenir 16 sur cet exemple.

Bon courage pour la certification et n'hésitez pas à me contacter sur Twitter @CyrilleGrandval si vous avez des questions techniques ou de fonctionnement la concernant.

Je veux créer **20 serveurs cloud** *en un clic* **Comment faire ?**

Avec Aruba Cloud,

vous utilisez directement l'API Aruba depuis le langage de votre choix et vous vous appuyez sur une base de connaissances largement documentée.
Grâce à l'interface Visual Cloud vous pouvez également designer votre datacenter et déployer vos serveurs virtuels en 1 clic. Si vous souhaitez que nous le fassions pour vous, nos équipes de Consultance sont à votre écoute.



- 3 hyperviseurs
- 6 datacenters en Europe
- APIs et connecteurs
- 70+ templates
- Contrôle des coûts

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 (octobre 2014)

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

aruba CLOUD

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

MY COUNTRY. MY CLOUD.*

*MON PAYS, MON CLOUD.

Les certifications pour développeur, ça vaut quoi?

Pour mes 19 ans, j'ai intégré une école privée me promettant d'être « certifié » par les plus grandes entreprises de ce monde. Mon manque de reconnaissance fut comblé : enfin quelqu'un pourrait certifier mes talents de développeur. Enfin je pourrais accéder à des « diplômes » prouvant ma valeur.

Julien Dollon
Fondateur de Think as an Engineer

Ces certifications représentaient beaucoup pour moi. J'en suis devenu un vrai boulimique, j'ai passé plus d'une centaine de certifications, entre autres :

Scrum Master, Linux Professional, Sun Certified Java Programmer, Microsoft Certified Trainer and Solutions developers, Cisco Certified Security Professional, etc.

J'ai aussi tout fait pour recevoir des certifications honorifiques (Microsoft Student Partner, Most Valuable Professional)...

Bref, ridicule n'est-ce pas ?

Si j'attachais tous ces pin's sur mon manteau, on m'aurait appelé General Microsoft :). Mais en fin de compte, qu'est-ce que cela m'a apporté ? Qu'est-ce que cela vaut ? Voilà la question que je me suis longtemps posée... après les avoir toutes passées.

Qu'est-ce que les certifications certifient réellement ?

A moins d'aller au plus haut niveau dans les certifications (CCIE de Cisco par exemple, Microsoft Certified Master..), la plupart d'entre elles se passent dans des centres spécialisés devant un ordinateur. J'ai travaillé dans plusieurs centres de formations, et j'ai moi-même été un « surveillant » officiel de ces organismes, dans plusieurs pays. La première chose qui a démystifié la certification fut le fait que les surveillants n'étaient pas tous très professionnels. J'ai vu de tout, du surveillant prenant son métier à cœur, et qui restait derrière vous pour que vous sentiez son souffle chaud sur votre nuque, jusqu'à celui qui vous laisse libre. J'y ai vu pas mal de triche, entre autre grâce au téléphone.

Mais au-delà de cet aspect triche lors de l'examen, qu'on se le dise tout de suite : A MOINS DE PASSER L'EXAMEN EN BETA, TOUTES LES REPONSES SONT SUR INTERNET. Toutes. On appelle ça des « dumps » ou « testking ». Gratuits ou payants, ils sont trouvables en trois clics.

Personnellement, j'ai toujours travaillé comme un dingue mes certifications, au point de réécrire des livres entiers. Malgré tout ça, mes notes n'excédaient pas plus de 850/1000, avec des nombreux échecs sur un examen de plusieurs heures.

A l'inverse, en tant que surveillant, montre en main, plus de la moitié des candidats que j'ai vu passer réussissaient le test en moins de 10 minutes avec plus de 900/1000. Soit ce sont des prodiges, soit ce sont des tricheurs. Et ça, je ne supporte pas. Alors au-delà de toutes ces triches qui accompagnent les certifications, qu'en est-il ? Admettons que quelqu'un passe la certification sans tricher, qu'est-ce qu'il valide ?

Cela valide votre capacité à avaler des documentations très techniques. Point.

Mais les connaissances pures d'une technologie font-elles de nous un meilleur développeur ? Je ne pense pas.

Etre un bon développeur, c'est être capable de résoudre des problèmes complexes avec les solutions les plus simples et les plus optimales possibles. C'est être capable de comprendre de bout en bout la problématique et d'y répondre grâce à la science, l'ingénierie mais aussi tous les domaines pouvant être impactés (psychologie, économie..). La science d'une part, car le métier que nous exerçons possède souvent une grande part de science exacte, mais également l'ingénierie, car il existe souvent de nombreuses solutions et il faut savoir faire des compromis.

Un ingénieur n'est pas un expert qui donnera une solution magique ou apprise par cœur en un temps record. Non, c'est plutôt quelqu'un qui sait qu'il existe rarement une réponse miracle. Son éducation lui a enseigné qu'une réponse en ingénierie sera souvent « ça dépend ». La solution est miracle que si elle prend en considération un panel gigantesque de variables et il le sait.

Pour preuve, pour avoir été responsable du recrutement de plusieurs dizaines d'ingénieurs chez Microsoft et Amazon, aucune question technologique n'a été posée aux candidats pour un poste d'ingénieur logiciel.

Carnegie Institute of Technology a récemment annoncé que 15% de votre réussite personnelle est due à vos connaissances techniques pures, les restes sont dus à vos principes d'Homme, votre capacité à communiquer, analyser et diriger. Je pense vraiment que ceci s'applique aussi à notre métier.

Cela ouvre-t-il des portes ?

Clairement. Deux cibles potentielles : ceux qui n'y connaissent rien, et donc, qui seront épatés (clin d'œil au phénomène « Gilbert » pour ceux qui ont connu), et les entreprises type SSII qui vendent des ingénieurs à ceux qui n'y connaissent rien :).

Concrètement, ça marche. Un multi-certifié se fera plus facilement embauché, voire même négociera un meilleur salaire.

Au final, cela sert à quoi une certification ?

Je pense qu'elles ont trois utilités, la première est celle d'avoir un but à atteindre. Pratique pour se motiver à apprendre une nouvelle technologie. Cela vous permettra de « valider » le fait que vous l'avez bien apprise par cœur. La seconde est dans le cas où vous en avez besoin d'un point de vue « business ». C'est-à-dire que votre entreprise peut avoir des privilèges « spéciaux » si elle possède un certain nombre de certifiés.

Pour finir, c'est avant tout quelque chose qui doit vous faire plaisir. Si vous avez une passion, un attachement pour une technologie et que passer cette certification vous rend heureux, alors c'est une très bonne raison de la passer.

Q&A

3 questions que l'on me pose souvent :

- **Est-ce qu'une certification vaut un diplôme ?** Qu'on se le dise une fois pour toute : NON, NON, NON et NON. Un diplôme reconnu par l'état vous permettra de vous ouvrir de nombreuses portes, que ce soit pour un visa à l'étranger, un poste dans le public ou tout simplement un emploi. Certification et diplôme sont incomparables.
- **Que valent les certifications honorifiques type « Most Valuable Professional » ?** Il faut voir ces certifications comme un cadeau que fait une entreprise à ses contributeurs pour les remercier de leurs investissements. Cela prouve clairement une chose : leurs passions. La passion peut souvent être synonyme de compétence mais cela n'est pas systématique.
- **Une certification prouve-t-elle que je connais un produit ?** Oui, cela prouve que vous connaissez tout ce que propose le produit par cœur ! Mais pour imager un peu mon

discours, c'est comme connaitre toutes les lettres de l'alphabet arabe, mais que pourtant, vous n'y compreniez pas un mot.

Qu'aurait-il fallu valider?

Et c'est là que je vous vends un truc :). J'ai bien conscience qu'avoir un discours aussi sectaire sur les certifications et essayer de faire sa pub ne font pas franchement bon ménage. Mais j'ai depuis des années adopté ce discours sur mon blog et ceci m'a poussé à la réflexion.
Et si je créais une certification différente ? Une certification qui serait passée non pas sur ordinateur, mais avec des vrais ingénieurs en face à face, comme un oral avec un tableau blanc, pendant 1 à 4 heures suivant le niveau. Ces ingénieurs seraient les meilleurs ingénieurs de notre industrie (ingénieurs senior Microsoft, Amazon, Google, Apple, Facebook...). Le but serait de valider cette équation : ingénieur = connaissances de la science informatique + capacité à faire de l'ingénierie + expérience passées prouvant que vous maitrisez l'art des compromis.

J'ai donc créé la certification « Think as an Engineer »
(http://certifications.thinkasanengineer.com).
Cette certification n'est ni un diplôme, ni un papier reconnu, ni une clef magique qui vous ouvrira les portes des entreprises.
Cinq types de personnes peuvent être intéressés par ma certification :

- Ceux qui ont besoin d'un but pour se motiver et avancer. Voire même du coaching.
- Ceux qui veulent s'entrainer aux entretiens de haut vol des grandes entreprises.
- Ceux qui cherchent un feedback de professionnels.
- Ceux qui cherchent un award, une reconnaissance personnelle.
- Un recruteur qui souhaite valider un candidat.

Trois niveaux existent, de l'ingénieur junior accessible aux étudiants et employés de 0 à 8 ans d'expériences jusqu'au niveau senior avec plus de 10 ans de pratique (le temps de carrière est là à titre informatif, on ne se base pas sur ce point).
Les certifications ont un socle commun : le code, qui peut être préparé avant l'entretien. Pour le reste, seul votre expérience et votre talent vous permettront de réussir, surtout pour les certifications SDE II et Senior.
Alors, si cela vous tente de vous battre pour une certification où la triche n'est pas possible, où l'ensemble de ce qui fait de vous un développeur est analysé, contactez-nous !

Greg Madison : quels futurs pour nos interfaces graphiques ?

Peut-être avez-vous entendu parlé du projet 7th sens, finaliste à ImagineCup 2009. Ce projet était déjà un concentré de réalités virtuelles et augmentées. Depuis, Greg n'a jamais cessé de travailler sur ces nouvelles interfaces. Designer d'interaction, il est aussi un acteur actif des réalités alternatives. Nous avons voulu en savoir un peu plus sur sa vision et comment l'interface va évoluer.

Tu as présenté une session autour des interfaces, des tendances actuelles et futures durant les TechDays, pourquoi une telle session ?

Une des volontés de Microsoft avec cet événement, c'est d'informer et de prévenir les professionnels des grands changements à venir. Le thème de cette année était entre autres l'intelligence ambiante, les objets connectés et le machine Learning. Avec comme invité surprise, leur casque holographique : l'HoloLens. Je ne pouvais pas rêver meilleurs sujets pour m'exprimer sur cette édition, car j'ai travaillé dans une cellule de R&D sur des agents intelligents de type Cortana, je suis l'organisateur du plus grand Meetup français traitant de la réalité virtuelle et de la réalité augmentée (Paris réalités alternatives)! Cette session était aussi l'occasion de pointer quelques absurdités. Trop de produits misent sur leur coté innovant, plutôt que sur le besoin utilisateur: l'usage qui n'a pas de sens, c'est par définition un non-sens. Ma vision holistique de l'innovation c'est de penser l'humain comme un "périphérique" ou un "capteur" faisant partie intégrante de l'écosystème informatique.

Aujourd'hui, l'interface repose sur des concepts vieux parfois de 50 ans, comment expliquer cette longévité ?

L'homme est profondément kinesthésique, il a besoin de manipuler pour comprendre, puis il est visuel et enfin feignant!
Décortiquons le combo écran/clavier/souris : Kinesthésique et visuel : lorsque j'enfonce la touche "A" de mon clavier, j'ai un feed-back sensoriel instantané. Dans la foulée, mes yeux me confirment le bon déroulement de l'opération à l'écran. Ce "A" qui s'affiche, est une extension de ma pensée. C'est direct, tangible et prédictible, comme l'interrupteur d'une lumière. Feignant, j'ai juste eu à bouger le bout de mes doigts pour atteindre ce résultat. Pourquoi abandonnerions-nous un tel système ? Pour une kinect par exemple? Pour gesticuler 5 minutes afin de saisir une adresse mail? Une innovation qui fonctionne, née d'une rencontre entre une possibilité technique et un usage ou d'une simplification de l'usage existant (fainéantise, pensez à la télécommande ou aux capsules Nespresso).
Cela laisse à penser que ce qui remplacera le traditionnel clavier/souris/écran est une chose qui vous demandera encore moins

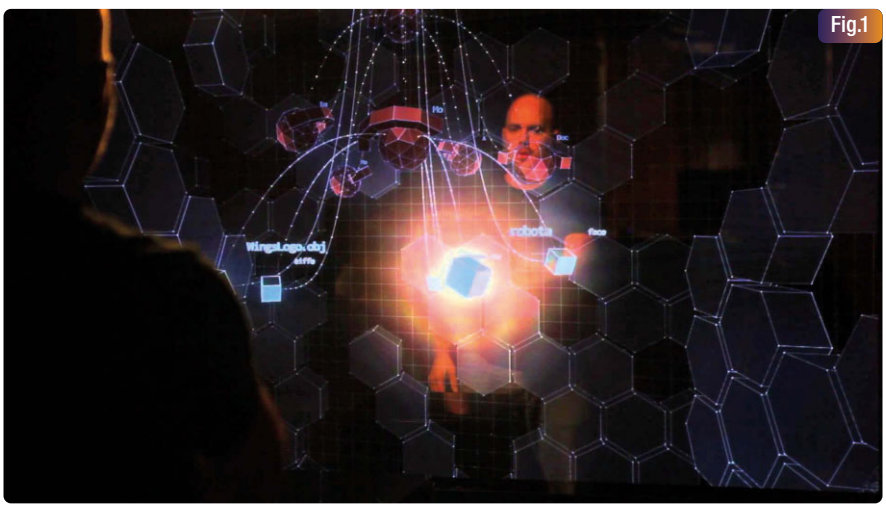


Fig.1

d'effort, certainement une entité intelligente qui réalisera vos tâches à votre place. Nous y arrivons !

Le tactile a changé beaucoup de choses, des technologies comme Kinect aussi, quel est ton ressenti par rapport à ces interfaces et ces nouvelles interactions ? **Fig.1**

Des technologies comme la Kinect, Leapmotion etc. sont magiques, mais je pense que l'on en fait n'importe quoi ! La promesse est de simplifier les interactions alors que dans les scénarios d'aujourd'hui on rajoute une couche de complexité. Comme nous sommes sur de l'expérientiel, il faut d'abord se trouver à un endroit bien précis, puis, les grammaires gestuelles changent tout le temps, et de ce fait, les interfaces manquent de prédictibilité. C'est-à-dire, savoir ce qu'il va se passer avant même que j'ai effectué une action. Pour bien comprendre regardez la vidéo "Microsoft envisioning the future" ou celle de http://seemove.net/. C'est peut-être très joli à l'écran, mais les interactions s'enchaînent et ne se ressemblent pas. Il serait impossible de se souvenir de toutes ces actions et de ce qu'il se cache derrière. C'est l'ordinateur qui doit s'adapter à nos routines de vie en devenant totalement invisible. Des technologies telles que "Witrack" ou "PointGrab" permettent d'imaginer des scénarios où c'est un mouvement de slide de la main vers la télé qui permettra de passer à la chaîne suivante, ou le même geste vers une ampoule qui l'allumera et l'éteindra.

Dans les années 1980 et surtout 90, plusieurs projets (avortés) ont circulé autour des interfaces dites 3D (ou du moins avec des effets 3D), je pense au système IBM, au Finder 3D d'Apple et au projet Hot Sauce (alias Project X) d'Apple, toutes ces tentatives furent avortées. Pourquoi revenir dessus aujourd'hui ?

Effectivement c'est une marotte dans le monde informatique, les interfaces spatialisées souffrent d'un seul problème pour être pertinentes: la technologie sur laquelle on les rend disponibles. Prenez par exemple BumpTop 3D Fig.2 (youtube - BumpTop 3D Multi-Touch Desktop) racheté par Google, jusqu'à présent ils n'en ont rien fait car passé l'effet "whaou" vous ne gagnerez pas en productivité avec un écran standard.
Maintenant, imaginez que vous puissiez rentrer "réellement" à l'intérieur de ce bureau 3D, et là, les interactions deviennent naturelles. Preuve en est Spacetop (MIT Medialab/Microsoft), un écran transparent qui permet d'avoir une véritable fenêtre sur son système d'exploitation

Fig.2

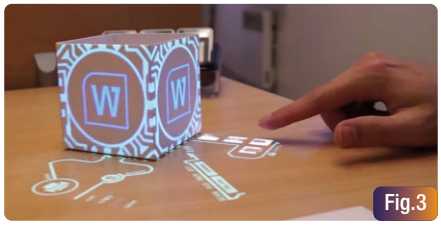


Fig.3

et qui autorise son utilisateur à interagir manuellement avec son bureau virtuel comme s'il était réel. Il y a une "skeuomorphismisation" logique de l'interface ; un bloc-notes ressemble à un bloc-notes et il se manipule comme tel. L'imitation du réel était peut-être absurde du temps de l'iPad mais il a permis de diminuer la courbe d'apprentissage de ce nouveau support et demain, elle prendra encore tout son sens lorsque nous chausserons nos lunettes de réalité alternative et que nous manipulerons autant d'objets réels que virtuels. Dans tous les cas, toutes ces recherches tendent à prouver que l'information est étriquée dans une fenêtre et que l'écran est condamné à disparaître plus vite que nous pourrions le penser.

Quelle est finalement ton approche sur les changements de paradigmes, d'approches pour le design et les interfaces ?

Mon approche c'est justement de spatialiser les interfaces et de fondre totalement l'informatique dans les objets du quotidien. Les objets réels seront liés à leurs objets virtuels. J'aborde notamment ce sujet dans ma conférence des Microsoft Techdays avec l'une de mes créations : la Pandora's box Fig.3. Ce démonstrateur est juste une simple boîte en carton qui, à l'aide de capteurs dissimulés dans la pièce, devient une extension de mon univers informatique. Concernant le plus gros changement de paradigme à venir c'est que, grâce ou à cause de l'informatique ubiquitaire, il n'y aura plus d'interfaces. Aujourd'hui nous disons à l'ordinateur ce qu'il doit faire, demain c'est lui qui nous commandera et l'on ne s'en rendra même pas compte. Exactement comme lorsque nous exécutons les ordres de notre GPS. Quand Watson par exemple, l'intelligence artificielle d'IBM, aura calculé toutes les probabilités pour organiser et optimiser votre journée, pris vos rendez-vous et commandé votre voiture autonome , vous n'aurez plus besoin de toucher à quoi que ce soit...

Tu travailles beaucoup autour des réalités alternatives. Est-ce là, l'un des futurs possibles de nos interfaces, de nos approches sociétales et des interactions homme - machine, homme - homme ?

Je vais être péremptoire... c'est LE futur des interfaces. Par interfaces, j'entends vecteurs de communication entre l'homme et la machine. Si je résume un peu, les écrans vont être remplacés par des lunettes, casques voir des lentilles et ce sont les objets eux-mêmes qui vont devenir les interfaces informatiques. Les interfaces plus compliquées seront remplacées par des holobots, des robots holographiques conversationnels comme dans le film HER.
Je sais qu'il y a des à priori à cause des Google Glass. Google a cannibalisé l'attention avec ses smartglass qui semblent à tort être un échec. Pourtant, si Sony, Epson, DAQRI, pour ne citer qu'eux, ont depuis plusieurs années des appareils comparables à l'HoloLens de Microsoft et que d'ailleurs Microsoft, Facebook et bien d'autres investissent massivement, c'est qu'il y a une raison. C'est un nouveau levier de croissance pour toutes les industries, software et hardware. Si vous n'avez pas encore inclus ces scénarios dans vos stratégies d'innovation, c'est que vous êtes passé à côté de quelque chose ! ⊞

Xavier Guillemane, créateur de Podcast Addict

Pour s'occuper durant ses trois heures de transport quotidien, Xavier écoute des podcasts et se découvre une passion. Déçu par l'offre existante, il développe son propre lecteur pour Android. Le succès ne se fait pas attendre. L'app Podcast Addict[1] est téléchargée plus de 2 500 000 fois sur le Play Store. Elle est mise-à-jour très fréquemment, en fonction des remarques et demandes des utilisateurs qui, en retour, lui accordent l'excellente note de 4,5[2].

Thierry Leriche-Dessirier
Twitter : @thierryleriche

Comment es-tu tombé dans l'informatique et plus spécialement dans le développement ?

Mon premier contact avec un ordinateur s'est déroulé à l'école primaire. Il s'agissait d'un Thomson TO7[3]. Ce nom évoque encore quelques souvenirs nostalgiques. Il y avait des jeux comme Lode Runner[4] et également la possibilité de « programmer » avec le langage Logo[5] : on pouvait déplacer une tortue à l'écran à l'aide de commandes, dessiner des formes, etc. A la même période mes parents ont acheté un Amstrad PC1512[6].

J'ai commencé par installer quelques jeux puis, en épluchant l'épais manuel (car les ordinateurs de l'époque étaient livrés avec des manuels), je me suis rendu compte qu'on pouvait faire autre chose que jouer… J'ai ainsi commencé à bidouiller, à recopier/modifier/tester des bouts de code et, surtout, à faire planter l'ordinateur familial...

Depuis cette période, j'ai su que je voulais travailler avec des ordinateurs et je n'ai jamais envisagé une autre profession.

Comment est née l'app Podcast Addict ?

En 2008 j'habitais en région Parisienne et j'ai changé de boulot. Mon trajet quotidien est passé de dix minutes à pieds à trois heures en RER. J'ai donc cherché un passe-temps : lecture, jeux vidéos, musique. Un jour, je suis tombé sur le blog de Scott Hanselman[7]. En plus de proposer des articles intéressants, j'ai découvert son podcast[8] « Hansel Minutes » dont j'ai téléchargé quelques épisodes sur mon PC, puis transféré sur mon téléphone pour les écouter dans le train. J'ai trouvé le concept du podcast très intéressant et, rapidement, j'ai passé mes trajets à écouter ce podcast, puis de nombreux autres ont suivi. Quelques années ont passé et j'ai remplacé mon téléphone par un HTC Desire, sous Android. J'ai donc découvert les apps, mais je n'ai pas été emballé par les gestionnaires de podcasts proposés. Après quelques mois, j'ai eu envie de tester le développement sous Android et j'ai commencé à développer mon propre gestionnaire. La première version a été publiée en janvier 2012 et ne proposait que les podcasts des regroupements NoWatch et Freepod. Puis, avec les retours des utilisateurs, l'appli a évolué à raison d'une nouvelle version par semaine.

Pour toi, qu'est ce qui fait que l'on aime toujours et encore le développement, la technique ?

L'informatique est un secteur en perpétuelle évolution. Il y a sans arrêt de nouveaux langages, frameworks, api qui sortent et que l'on peut explorer. De même, lorsqu'on développe un pro-

duit, il y a toujours des bugs à corriger, des performances à améliorer, de nouvelles fonctions à ajouter...

Tu as gardé un regard très geek : gadget, veille techno, c'est important pour ton job et ta passion ?

J'ai toujours été passionné par l'informatique et les nouvelles technologies. Je consacre donc pas mal de temps à m'informer mais il s'agit avant tout d'un loisir. Cette veille est toutefois essentielle lorsqu'on est développeur. Tout va très vite. Les nouvelles versions d'Android se succèdent, de nouveaux supports arrivent (Chromecast, Android Wear, TV, Auto, Google Glass, etc.), de nouveaux frameworks apparaissent tous les 6 mois... Il est donc indispensable de se tenir informé pour s'assurer que l'app continuera de fonctionner sur les nouvelles versions et aussi d'intégrer rapidement la compatibilité avec les nouveaux supports.

Être développeur n'est pas toujours facile : pression, évolution constante, frustration des projets et des « chefs », c'est quoi pour toi être développeur aujourd'hui ? Le job a-t-il changé depuis tes débuts ?

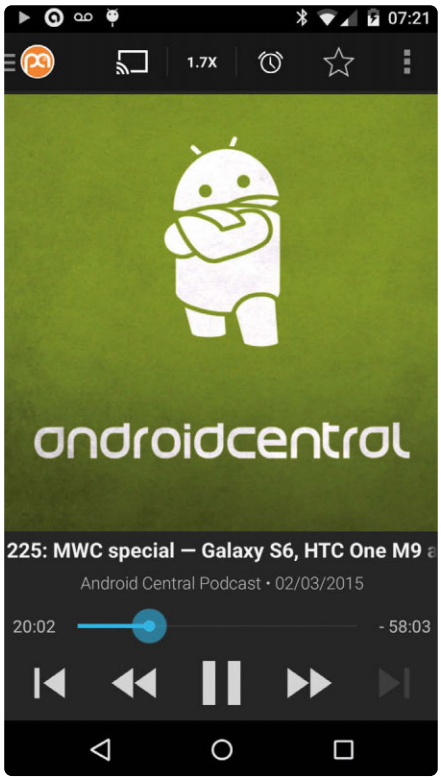
C'est justement ce qui fait tout l'intérêt du job ;-) On est constamment en train d'apprendre, de découvrir de nouvelles technos et de progresser. Etre développeur, c'est surtout être curieux, avoir envie d'apprendre et de progresser.

Ce qui a pas mal changé depuis que j'ai commencé à travailler, c'est l'arrivée progressive des méthodes agiles sur la plupart des projets, y compris sur de gros projets. Cela encourage, entre autres, à livrer de nouvelles fonctions régulièrement, plutôt qu'à travailler de longs mois sur une nouvelle version.

De plus je trouve que ça prend tout son sens lorsqu'on développe des applications mobiles sur Android. On est en contact direct avec les utilisateurs. Le déploiement d'une nouvelle version se fait en quelques minutes et la mise en ligne est effective sous deux heures.

Et en dehors du boulot, qu'aimes-tu faire ? Comment trouves-tu l'équilibre entre travail, vie privée, passion, famille ?

Mes loisirs tournent autour de l'informatique et des podcasts, mais j'aime aussi regarder des films et des séries, lire et jouer à des jeux vidéo lorsque je trouve un peu de temps. Comme je travaille désormais depuis chez moi, il est assez délicat de trouver un équilibre entre la famille et le travail. Les messages des utilisateurs et/ou les alertes serveurs peuvent arriver 24-7. Il faut que j'apprenne à détecter rapidement les urgences et à temporiser tout ce qui n'est pas critique afin de pouvoir passer du temps en famille.



« Un logiciel n'est jamais terminé, c'est pour cela que j'aime le développement... »

Peux-tu nous présenter ton quotidien en quelques mots ?

Durant les trois premières années du projet, mon quotidien a été rythmé. Au réveil, je vérifiais le nombre de crash, pour voir s'il n'y avait pas eu d'anomalies, la note moyenne de l'app, le nombre de ventes, les revenus publicitaires, etc. Pendant les transports, je répondais aux mails, commentaires, questions, etc. Je refaisais une passe midi et soir. Ensuite, soirée en famille avec un petit épisode d'une série TV pour faire une coupure puis je passais le reste de la soirée à développer l'application. J'y consacrais également du temps le weekend. Récemment la situation a changé. J'ai quitté la région parisienne et mon boulot pour me mettre à mon compte et, ainsi, libérer plus de temps pour travailler sur l'application... et pour la famille :)

Mon quotidien a donc évolué. Le matin, je me lève tôt pour répondre aux mails et commentaires au calme avant que les enfants ne se réveillent. Avec l'augmentation du nombre d'utilisateurs, le support prend de plus en plus de temps... Comme je travaille de chez moi, je peux m'occuper un peu des enfants le matin et amener ma fille à l'école. Ensuite, je suis au calme pour travailler sur les prochaines évolutions de l'app avant le retour des enfants en milieu d'après-midi. Là, comme ça devient de suite plus bruyant, je refais un peu de support.

La difficulté est d'arriver à s'arrêter alors que les messages, eux, continuent d'arriver. Pour l'instant, mes fins de soirée et mes weekends sont encore consacrés au développement de l'app... mais il va falloir que j'apprenne à couper...

Comment vois-tu ton job évoluer ?

Quand j'ai commencé à travailler sur Podcast Addict, il s'agissait d'un petit projet pour tester le développement Android, pour pouvoir accéder à mes podcasts plus simplement. Aujourd'hui les revenus engendrés par l'app me permettent de passer à mon compte.

Je vais consacrer les prochains mois à faire évoluer l'app, que ce soit pour corriger les bugs, améliorer les performances, mais aussi ajouter de nouvelles fonctionnalités. Avec les nouvelles versions d'Android qui se succèdent et la multiplication des supports (Android Wear, Android TV, Android Auto, Chrome, ...) ce n'est pas le travail qui manque.

Cependant il est assez difficile de se projeter à moyen/long terme tant le secteur est dynamique et évolue rapidement.

Des conseils aux étudiants et développeurs qui nous lisent ?

N'hésitez pas à travailler sur un projet perso. Le Web regorge de tutoriels sur tous les langages et frameworks existants.

Si le développement mobile vous intéresse, le coût d'entrée est très bas. De plus, c'est assez grisant de publier une app et de se dire qu'elle est accessible une heure plus tard par plus d'un milliard d'utilisateurs.

En travaillant sur un projet perso, on apprend pas mal de choses dans des domaines variés, comme le design, le support utilisateur, le SEO, le marketing, les analytics, etc. Il est important d'être sensibilisé à ces aspects, le plus dur étant de se lancer...

(1) Podcast Addict : http://podcastaddict.fr
(2) Moyenne de 4,5/5, pour 100 000 notes.
(3) Thomson T07 sur Wikipedia : http://fr.wikipedia.org/wiki/Thomson_TO7
(4) Lode Runner sur Wikipedia : http://fr.wikipedia.org/wiki/Lode_Runner
(5) Langage Logo sur Wikipedia : http://fr.wikipedia.org/wiki/Logo_%28langage%29
(6) Amstrad PC-1512 sur Wikipedia : http://fr.wikipedia.org/wiki/Amstrad_PC-1512
(7) Blog de Scott Hanselman : http://www.hanselman.com
(8) Podcast « Hansel Minutes » : http://www.hanselminutes.com

Retour sur Maker Faire Paris 2015

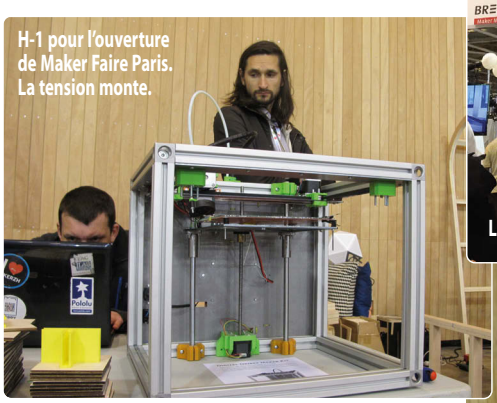
La seconde édition du Maker Faire s'est déroulée à Paris les 2 et 3 mai derniers, durant la Foire de Paris. Une mini Maker Faire s'était déroulée mi-avril à Saint Malo. Plus de 700 Makers étaient présents et ont exposé leurs projets dans tous les domaines : électronique, robotique, culture bio, tissu...

François Tonic
Programmez !

Maker Faire est le lieu par excellence du mouvement Maker et du faire soi-même (DIY en Anglais). Mais ne croyez pas qu'il s'agisse uniquement de technologies, même si elles prennent une part importante dans de nombreux projets. On parlera de hautes technologies, mais aussi de Low Tech (basse technologie), nous rappelant au passage que la technologie coûte cher et que tout n'est pas accessible partout. À côté des grands stands d'Intel, Leroy Merlin, Conrad, etc., l'honneur était aux Makers et à leurs projets, parfois montés à l'occasion du Maker Faire, parfois, issus d'un long travail de développement. Mais il y a toujours de l'enthousiasme, de l'énergie. Voilà pourquoi nous aimons ce mouvement d'envergure; dans cet esprit, nous retrouvons le côté bidouille des années 1980 et 1990...

Sur le stand Microsoft, plusieurs projets du Hackathon Zone61 étaient présentés durant les 2 jours. Et une vaste zone pour drones permettait des démonstrations. Nous avons aussi beaucoup aimé les ateliers de Lego Coding par Magic Makers, et bien entendu, les ateliers pour créer et souder son petit robot Maker Faire ! Dans la même idée, il y avait aussi VoLAB pour s'initier à la programmation et à la robotique. Autre coup de cœur (difficile de tous les citer) : le robot Buddy. Ce robot se veut un compagnon au quotidien pour aider les personnes. Vous avez été nombreuses et nombreux sur notre stand Programmez ! pour échanger sur le magazine, le découvrir et nos « projets » : la tasse connectée pour Geek vNext, la mini station médicale « mobile » et une boîte à meuh qui ne faisait pas meuh (merci les bugs du capteur). Comme chez d'autres

© François Tonic



© Valérie Turmel

© Valérie Turmel

© Valérie Turmel

35 000 visiteurs*
+ 250 projets
+ 50 conférences et ateliers
+ 789 makers

** chiffres donnés par leFaShop*

Makers, nous avons eu quelques bugs et un peu de casse matérielle : un écran OLED et plusieurs capteurs n'ont pas fini le salon ! Autre curiosité : les Robots Cubes. Ce projet explore les différentes manières de faire bouger des cubes tout seul...

La BlueFrog : une plateforme de prototypage ultra compacte

Voilà un projet comme on les aime à la rédaction. Le précieux tient dans une surface de 4,2 x 4 cm. L'objectif est de créer très rapidement des prototypes comme des objets connectés, des drones ou tout autre objet indépendant. Il propose un écran OLED (très lisible et de belle qualité), une batterie LiPo, une connexion USB et une minuscule carte contenant le processeur, le stockage, des capteurs et la connectivité ! Il fonctionne sur un ARM 32 bits (Cortex-M4) et embarque un stockage de 8 Mo utilisant un système de fichier FAT. Cette plateforme intégrée, vue sa taille, est impressionnante par les capteurs intégrés : pression, accéléromètre 3 axes, gyroscope 3 axes, magnétomètre, lumière ambiante, proximité, température, audio ! Pour la partie réseau, nous avons le port USB et le Bluetooth 4.0. L'extensibilité n'est pas oubliée avec des broches d'extensions (GPIO, timers, I2C, SPI, etc.). Côté programmation, on retrouve C et C++. Le tout fonctionne un mini-système (RIOT). Nous avons été impressionnés par cette nouvelle solution ! À découvrir : http://www.la-bluefrog.io

TheAirBoard : une mini Arduino que l'on attend !

Le monde Arduino accueille déjà de nombreuses cartes, plus ou moins grandes. L'Air Board sera bientôt une nouvelle arrivante avec de nombreux atouts : 100 % compatible Arduino, une taille compacte mais pratique, faible consommation et longue autonomie. Elle a été développée à Grenoble. Solution idéale pour les objets connectés, les objets embarqués. Les livraisons devraient débuter vers août – septembre... Sur kickstarter, plusieurs kits de la carte sont toujours disponibles. Un des coups de cœur de la rédaction.

Thingz ou comment construire en quelques minutes son objet électronique

Simple, ludique, amusant, voilà comment Thingz nous est apparu sur la table à la fin de la 2e journée du Maker Faire. Le principe est très

simple : des briques que l'on connecte les unes aux autres pour construire un circuit, un objet. On dispose d'une plaque d'accueil sur laquelle on pose les briques électroniques (bouton, led, etc.), puis on connecte le tout à son ordinateur pour écrire quelques lignes de codes... Trois kits sont proposés de 39 à 89 € !

Des kits Intel Edison réellement disponibles avec Smartliving

Il y a quelques mois nous avions présenté et testé la carte Edison conçue par Intel et taillée pour l'embarqué, le wearable. En attendant la carte Curie, Edison était prometteuse, mais pas toujours facile de démarrer en douceur avec la plateforme. Le constructeur propose aussi des kits pour Arduino et Raspberry Pi. Ces kits reposent sur les excellents modules et shields Grove (Seeedstudio). Chaque kit propose plus de 15 modules. Pratique pour démarrer rapidement et sans utiliser le fer à souder. Seul réel défaut de ces kits, leurs prix ! Le kit Edison est proposé à 212 €, celui dédié à la Pi (modèle Pi B+) est tout de même à 144,99 € ! Franchement cher même si les modules sont de qualité. Site : http://www.smartliving.io/ . Le stand Intel proposait plusieurs communautés et constructeurs pour démarrer: comme quoi, les cartes Intel (Edison ou Galileo) étaient parfaitement dans le mouvement des objets connectés et le Maker !

Vive les bornes d'arcade !

Souvenez-vous des bornes d'arcades des cafés et bars ! Des étudiants d'Epitech ont voulu recréer une borne de jeux sur le principe du jeu musical Osu (pour un premier jeu). Prototype après prototype, la borne s'affine ! On adore ! Twitter : https://twitter.com/jubeatreturn

e-nable the future : du Low Tech pratique et terriblement d'actualité

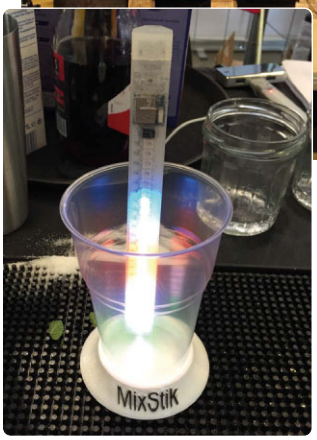
Voilà une communauté qui mérite notre soutien et notre admiration. À travers le monde, de nombreuses personnes amputées d'une main (plus ou moins sévèrement) n'ont pas accès à des prothèses : elles sont trop chères. Aujourd'hui, une prothèse se veut toujours plus technologique et technique, mais les tarifs s'envolent. Là, pour 50 € et grâce à l'impression 3D, on construit très rapidement une prothèse. Elle est personnalisable et surtout, la maintenance est simple et rapide. D'autre part, quand l'enfant grandit, changer la prothèse n'est plus un problème insurmontable ! Admiration ++ de la rédaction.

Kit PrintBot Evolution

Vous rêvez d'un objet mobile autonome ? Le distributeur Premium exposait le Kit PrintBot

Le groupe robotique s'installe...

© Valérie Turmel



© François Tonic

Impossible de rater un coktail...

Evolution. Le corps proprement dit est imprimé en 3D et le kit inclut dedans les éléments électroniques et mécaniques nécessaires. La réalisation est propre et de qualité, à condition de bien faire le montage. Surtout, le tarif proposé est très intéressant : 99,90 € ! La carte est disponible séparément. Elle est compatible Arduino. Sa construction est très propre et possède en standard une communication bluetooth (super pratique), de l'USB, même si elle est légèrement moins puissante en tension d'entrée.

MixStik : réussir son cocktail !

Voilà un petit objet connecté pratique et peu encombrant... Il se compose d'un stick embarquant toute l'électronique et d'une app mobile. Objectif : pour chaque cocktail, l'app donne la composition de celui-ci et surtout les hauteurs (quantité) pour chaque élément. Et le stick, servant de règle, affiche en temps réel les couleurs à suivre ! Pratique ! Un SDK sera proposé. L'objet devrait être lancé dans quelques mois. Site : http://mixstik.com/fr/

Impression 3D : il suffit de choisir !

Cette année encore, l'impression 3D est venue en force : marques françaises et étrangères, modèles « artisanaux » à monter ou à bas prix. Au-delà, il y avait plusieurs services d'impression à la demande et les accessoiristes et revendeurs. Petit tour :

- **Robox** (présentée par Le Comptoir 3D) est une petite nouvelle sur le marché français. Plusieurs atouts : buse d'impression changeable très facilement, mise en température automatique selon le filament (la bobine contient une puce d'identification), possibilité d'avoir deux buses d'impression et 2 bobines, température constante dans la zone d'impression. L'imprimante est plutôt compacte et autorise des impressions de 300 à 50 microns. Fonction intéressante : pause/reprise en cours d'impression.
- **MiniLab :** ce projet étudiant est de proposer de fabriquer des objets en Open Source avec des outils accessibles à tous, à des coûts les plus accessibles possible en utilisant des matériaux recyclés, des cartes de types Arduino. Ce projet permet de créer des découpeuses laser, des mini-serres, etc. On aime ++
- **Geekofyou :** le site propose des formations sur de nombreux sujets dont l'impression 3D, avec possibilité de monter soi-même son imprimante ! Mais le site propose de nombreux autres sujets ! www.geekofyou.fr
- **3D Slash :** outil de modélisation qui se veut grand public et surtout accessible à des utilisateurs non experts en 3D... Disponible en mode Web ou sur son desktop.
  Plusieurs services d'impression 3D étaient visibles durant les 2 jours, notons :
  Le service sculpteo qui permet d'envoyer ou de choisir un modèle 3D et de le recevoir quelques jours après chez soi
- **3D on Demand :** plateforme pour le design 3D et l'impression 3D. Ce projet est passé par du crowdfounding pour le financement.
- **Cults :** service français et communautaire. Il permet de télécharger et de vendre ces propres modèles 3D.

Pour découvrir ou redécouvrir les projets : http://www.makerfaireparis.com

Vivement Maker Faire Paris 2016.

L'intégration de l'outil de statistiques Flurry dans vos applications Android éditées via Xamarin Studio

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

J'ai débuté le développement des applications Android dans le but de les publier dans le store du Nokia X et ma première difficulté a été de choisir le support de développement. J'ai finalement opté pour Xamarin car cela me permettait de reprendre quasiment dans leur intégralité mes travaux réalisés sous Windows Phone. Bien sûr, si vous avez d'autres outils, rien ne vous empêche de les utiliser :)

1 Quel est le sujet du jour?

On va voir ensemble comment intégrer Flurry dans une application Android créée sous Xamarin. Les prérequis afin de réaliser cette manipulation sont :

- Visual Studio (celui que vous voulez tant que vous pouvez programmer) avec le module Xamarin ou Xamarin Studio,
- Un compte Flurry afin de constater le résultat. Compte que vous pouvez créer sur www.flurry.com et bien sûr sur lequel vous allez créer un projet pour votre application/jeu.

2 L'installation des fichiers

Avant de passer à la ligne de code il y un peu de téléchargements et d'installation de fichiers.

Premier fichier à télécharger, FlurryAgent.cs qui se trouve sur https://github.com/jamesmontemagno/TasksSimplified/tree/master/TasksSimplified/Helpers et à installer dans un répertoire Helpers (à créer) de votre projet.

Deuxième fichier qui est FlurryAgent.jar qui se trouve sur https://github.com/jamesmontemagno/TasksSimplified/tree/master/TasksSimplified/Jars et à installer dans un répertoire Jars (à créer) de votre projet.

3 Le programme

Il est très simple. Dans l'événement OnCreate de votre projet qui est dans le fichier MainActivity.s il faut ajouter le code suivant :

```
Helpers.FlurryAgent.OnStartSession(this, "le code de votre application");
```

Et enfin créer un événement OnStop dans ce même fichier :

```
protected override void OnStop ()
    {
      base.OnStop ();
      try{
      Helpers.FlurryAgent.OnEndSession(this);
      }
      catch{
      }
    }
```

4 Conclusion

Nous avons vu comment intégrer Flurry dans une application. Durant ma période d'essai Xamarin, j'ai également essayé la régie Airpush (un peu agressive à mon goût) et plusieurs contrôles. J'ai néanmoins pu tester et valider cette méthode par le biais d'une application que je ne peux pas encore publier.

Offres spéciales d'abonnement !

Nos formules classiques

PROGRAMMEZ!
le magazine du développeur

devolo
The Network Innovation

Pour un abonnement, Devolo et Programmez! vous offrent un kit complet CPL dLAN 550
d'une valeur de 79,90 €

Abonnement normal 49 € ou 79 €
+ 15,90 € de frais logistiques et postaux

Attention :
- *cette offre est strictement limitée à la France Métropolitaine et à la Corse.*
- *quantité limitée, jusqu'à épuisement des stocks. Cette offre est susceptible de s'arrêter à tout moment.*

1 an 11 numéros
64,90€
(au lieu de : 128,90 €, abonnement 1 an / 11 numéros : 49 € + 15,90 € de frais logistiques et postaux)

2 ans 22 numéros
94,90€
(au lieu de : 158,90 €, abonnement 2 ans / 22 numéros : 79 € + 15,90 € de frais logistiques et postaux)

1 an 11 numéros
49€
seulement (*)

2 ans 22 numéros
79€
seulement (*)

Spécial étudiant
39€(*)
1 an 11 numéros

(*) Tarifs France métropolitaine

ABONNEZ-VOUS !

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

Offre abonnement + kit CPL dLAN 550

☐ ***Abonnement 1 an :*** 64,90 € (au lieu de 128,90 €, kit CPL dLAN offert)
☐ ***Abonnement 2 ans :*** 94,90 € (au lieu de 158,90 €, kit CPL dLAN offert)
Tarifs France métropolitaine

☐ M. ☐ Mme ☐ Mlle Entreprise : Fonction :

Prénom : Nom :

Adresse :

Code postal : Ville :

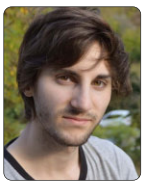
email indispensable pour l'envoi d'informations relatives à votre abonnement

E-mail : @

☐ Je joins mon règlement par chèque à l'ordre de Programmez !
☐ Je souhaite régler à réception de facture
** Tarifs France métropolitaine*

PROG 186
***Offre limitée**, valable jusqu'au 29 juin 2015*

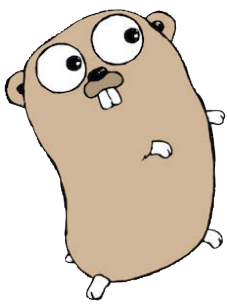
Go, c'est quoi ?



Vincent Nëel
Manager de l'innovation au sein du Hub d'innovation d'Epitech.

LES ORIGINES

Go est un langage de programmation créé par Google. Il est sorti en 2009 en open source, et la version stable Go 1 est arrivée en mars 2012. Son origine remonte à 2007 lorsque ses créateurs, Rob Pike, Robert Griesemer et Ken Thompson, à l'époque développeurs C++, travaillaient sur un projet chez Google qui mettait 45 minutes à compiler sur leur énorme cluster. Après avoir assisté à une conférence sur les nouveautés apportées par le C++ 11, Rob Pike ne comprenait pas pourquoi le *C++ Standards Committee* continuait à annoncer plein de nouveautés au langage alors qu'il estimait qu'il fallait le simplifier. Après la conférence, Rob Pike et Robert Griesemer ont commencé à réfléchir sur un nouveau langage, basé sur du C et s'appelant "go". Leur résultat final était loin du C et du C++ : garbage collector; pas de headers (fichiers .h), pas de dépendance circulaire, compilation par package, pas de templates, pas d'exceptions, utilisations de *struct* au lieu de *class,* pas d'arithmétique des pointeurs et encore bien d'autres différences Fig.1. Lors de son annonce en 2009, le langage a fait couler beaucoup d'encre, tant en bien qu'en mal, notamment à cause des partis pris (que nous verrons plus tard). Malgré tout, beaucoup de développeurs python ou ruby ont appréciés Go grâce à sa syntaxe simple et ses performances proches du C ; beaucoup diront que le Go est du C moderne. Aujourd'hui, le Go est en version 1.4.2 avec une popularité grandissante, notamment grâce à Docker qui est développé en Go.



Gopher, la mascotte de Go

COMMENT IL FONCTIONNE ?

Go est multiplateforme (Linux, OS X, FreeBSD, NetBSD, OpenBSD, Plan9 et Windows) et multi-architecture (32 bits, 64 bits et ARM). Comme dit précédemment, il est dans la lignée du C mais a opéré de grands changements visant à rendre le développement plus court, simple et sécurisé (pas de segfaults).

Voici une liste non exhaustive des différentes spécificités du langage :

- Une possible déclaration de variables via inférence de type (*x := 0* au lieu du classique *int x = 0;*),
- Un temps de compilation très court,
- Un système de gestion de packages (*go get ...*) et un système de documentation automatique de tous les packages publics (http://godoc.org),
- Un système de concurrence intégré au langage : des processus légers (*goroutines*), des canaux de communication (*channel*), et le mot clé *select* qui permet de faire l'équivalent d'un *switch* sur différents *channels*,
- Mot clé *go* qui permet de lancer une fonction en tâche de fond,
- Pas d'héritage de classe mais un héritage par interface,
- Un système de compilation qui génère des binaires sans dépendances externes (du coup les binaires peuvent faire plusieurs Mo, contrairement au C où les binaires sont généralement très légers),
- Pas de surcharge d'opérateur,

Fig.1

(De gauche à droite) Robert Griesemer, Rob Pike et Ken Thompson

- Pas de dépendances circulaires,
- Pas d'assertions,
- Pas d'arithmétique de pointeurs,
- Pas de programmation générique,
- Gestion de la mémoire automatique (avec un garbage collector).

POURQUOI CODER EN GO ?

Premièrement, le Go est un langage qui a été pensé pour s'exécuter sur des machines modernes ayant plusieurs processeurs et donc gérant le multi-threading. Il est à noter que le Go excelle dans les programmes concurrents. Petit rappel, la concurrence est le fait d'exécuter du code en arrière-plan, mais pas forcément dans un autre thread. En effet, il ne faut pas confondre le parallélisme et la concurrence : la concurrence est le fait d'avoir plusieurs processus indépendants s'exécutant en tâche de fond tandis que le parallélisme est le fait d'exécuter simultanément différents calculs possiblement liés entre eux.

Une autre réponse à cette question est simple : comparé à d'autres langages modernes comme Python, Ruby, PHP et JS, il est bien plus performant. Aujourd'hui, lorsqu'on cherche à développer des applications performantes dans des langages qui ne sont pas très rapides de base (comme le PHP par exemple), il faut toujours essayer d'optimiser, parfois au détriment de la lisibilité du code. En Go, c'est l'inverse : le langage est performant, donc vous pouvez faire en sorte de rendre votre code le plus lisible possible. Ce qui n'est pas difficile puisque le Go a une syntaxe simple même si elle diffère de la plupart des langages s'inspirant du C tels que le C++, Java, C# etc.

De plus, Go est installé avec un outil nommé **gofmt** qui permet de formater le code Go directement à la norme de Go. Cela semble être un détail, mais finie la guerre entre ceux qui mettent l'accolade à côté de l'ouverture d'un scope ou à la ligne. Cet outil va même supprimer les point-virgules que vous aurez mis par habitude (eh oui, il n'y a pas besoin de ";" en Go à la fin de chaque ligne). Du coup, vous n'aurez normalement que peu de problèmes à lire le code d'autres personnes sur Internet puisqu'il aura le même format que votre code.

De plus, la gestion de dépendances externes en Go se fait de manière très simple. Par exemple, il existe un dépôt sur Github nommé Pi qui appartient à l'utilisateur Gocarina. Ce dépôt contient une bibliothèque Go que je veux utiliser. Il suffit alors de faire appel à la commande **get** qui est incluse avec l'outil **go** lors de l'installation de Go :

```
go get github.com/gocarina/pi
```

Je peux alors l'importer normalement directement depuis mon code :

```
import github.com/gocarina/pi
```

Il n'y a pas besoin de fichiers de configuration : bye bye bower.json, package.json, requirements.txt et autres. Ensuite, le langage a une gestion

automatique de la mémoire avec un ramasse miette, donc on ne perd pas son temps à libérer la mémoire. Puis, le Go possède une bibliothèque standard moderne avec des analyseurs json, xml, csv, une bibliothèque réseau très complète (le package **net/http**), un package pour gérer le temps et bien d'autres choses encore.
Enfin, un grand atout est que le langage est compilé et génère un binaire statiquement lié, beaucoup plus lourd que d'autres langages (un simple hello world fait plus de 2000 Ko) et qu'il est donc totalement autonome, contrairement aux langages scriptés type Python ou PHP, où il est nécessaire d'avoir tout le code de l'application pour pouvoir l'exécuter.

LES CAS D'USAGES

Le Go a été conçu par Google pour réaliser des applications serveurs et est principalement utilisé pour réaliser des APIs. Il existe un nombre incalculable de bibliothèques qui aident au développement d'APIs REST (pour citer quelques exemples il y a **gorilla/pat**, **gorilla/mux**, **martini**, **gin**, **gocarina/pi**...). Go peut aussi être un très bon choix pour réduire les coûts liés à l'hébergement (la société Iron est passé de 30 serveurs tournant sur du Ruby On rails à seulement deux serveurs en Go, un seul aurait suffi mais ils avaient besoin d'un deuxième serveur pour la redondance). Il est également possible de créer des interfaces graphiques, et même de réaliser des jeux en Go, mais le langage n'est pas réellement fait pour ça, même s'il est possible de le faire; il vaut mieux utiliser des langages et des frameworks plus adaptés (Java ou .NET pour des interfaces graphiques lourdes ou C++ pour le jeux vidéo).
Par contre, le Go convient parfaitement pour le développement de moulinettes plus ou moins grosses pour traiter des données. Si avant vous utilisiez un langage de script quelconque pour ce faire (Python, Perl ou autre...), je ne peux que recommander de les faire en Go pour tirer parti de ses performances et de sa bibliothèque standard très fournie, que ça soit pour du traitement de fichiers JSON, CSV, XML, binaires ou autre, le Go pourra les traiter sans problèmes.

STRUCTURE ET SYNTAXE

Syntaxe de déclaration en Go

Généralement, la première chose qui trouble lorsqu'on se met au développement Go, c'est la déclaration de variables ou de fonctions.
Commençons par parler du C. Par exemple, pour déclarer un int en C, il suffit de faire :

```
int x;
```

A peu près tout le monde comprend qu'ici, on déclare x étant un int. En C, généralement, on met le type à gauche et l'expression à droite. Donc, les déclarations suivantes :

```
int *p;
int a[3];
```

spécifient que *p* est un pointeur sur int car "*p" est de type int et que *a* est un tableau de int car "a[3]" est de type int. Prenons maintenant le cas de fonctions :

```
int main(int argc, char *argv) { /* ... */ }
```

Ici, on voit que la fonction *main* retourne un int et prend deux paramètres : *argc* de type int et argv de type pointeur sur char.
La syntaxe du C est simple et astucieuse et fonctionne très bien pour les types simples mais elle peut vite devenir troublante. Prenons le cas des pointeurs sur fonctions en C :

```
int (*fp)(int a, int b);
```

Ici, *fp* est un pointeur sur fonction car si on écrit l'expression *(*fp)(a, b),* vous allez appeler une fonction qui retourne un int. Mais qu'est ce qui se passe si un argument de *fp* est une fonction ?

```
int (*fp)(int (*ff)(int x, int y), int b)
```

Ça commence à être difficile à lire... Enlevons le nom des paramètres pour plus de clarté :

```
int (*fp)(int (*)(int, int), int)
```

Ce n'est pas vraiment plus clair, et on commence à vite oublier que c'est une déclaration de pointeur sur fonction. Et là, prenons le cas extrême, si *fp* retournait un pointeur sur fonction :

```
int (*(*fp)(int (*)(int, int), int))(int, int)
```

Là, c'est même quasiment impossible de voir que cette déclaration déclare une varialbe *fp*. On pourrait s'amuser à rendre la chose encore plus compliquée (et si les paramètres étaient des structures plutôt que des int ? ou même des pointeurs sur fonction eux-mêmes ?), mais ce n'est pas le but. Reprenons les premiers exemples de C en Go :

```
var x int
var p *int
var a [3]int
```

Ces déclarations sont plus naturelles : on lit de gauche à droite "je déclare une variable x de type int", "je déclare une variable p de type pointeur sur int", "je déclare une variable a de type tableau de int".
Maintenant, regardons les fonctions :

```
func main(argc int, argv [] string) int {  /* --- */ }
```

Tout comme dans l'exemple précédent, la lecture se fait de gauche à droite : "je déclare une fonction main prenant comme paramètres argc de type int et argv de type tableau de string et retournant un int".
Et attention, si on prend l'exemple extrême de C :

```
var f func(func(int,int) int, int) func(int, int) int
```

Il est vrai que ça n'est pas très clair mais cette déclaration a le mérite d'offrir au lecteur la certitude que l'on déclare une variable *f*. Et si on s'amuse à la lire, cela donne :
"f est une fonction prenant en premier paramètre une fonction qui prend en paramètres deux variables de type int et retournant un int, et comme deuxième paramètre, un int et retourne une fonction prenant en paramètres deux int et retourne un int".
Difficile à lire à haute voix, certes, mais à écrire ça reste vraiment plus simple que les pointeurs sur fonction (si si, croyez-moi).

Les types *builtin*

- Les chaînes de caractères notées **string**,
- Les booleans notées **bool**,
- Les nombres entiers qui sont de types **uint**, **uint8**, **uint16**, **uint32**, **uint64**, **int8**, **int16**, **int32**, **int64**, **byte** (alias pour uint8), **rune** (alias pour uint32), **uint** (32 ou 64 bits en fonction du système) et **int** (même taille que uint),
- Les nombres flottants : **float32** et **float64**,
- Les nombres complexes : **complex64** et **complex128**,
- Les tableaux : **[32]int**, **[64][32]float64** etc.,
- Les slices, qui sont des tableaux à tailles variables (ils peuvent êtres agrandis après leur instanciation) : **[]int**, **[]float64** etc.,
- Les pointeurs : ***int**, ***[4]int** etc. Par défaut, les variables sont passées par copie à une nouvelle fonction, sauf lorsque la variable est de type

pointeur, **interface**, **channel** ou **slice**,

- Le type **nil** qui est l'équivalent du null en java. Par défaut, une map, un pointeur, un slice, une interface sont à nil,
- Les fonctions (le Go considère les fonctions comme un type, elles peuvent donc être passées en paramètres à d'autres fonctions) : **func(int, string, int) (int, string, int)** (fonction prenant un int, une string et un int en parametre et retournant un int, une string et un int). Il est a noter que les fonctions commençant par une majuscule sont exportées (publiques), et celles commençant par une minuscule ne le sont pas (elles sont privées),
- Les structures (qui font office de classe en Go).

Exemple de déclaration d'un nouveau type Animal en Go via une structure :

```
type Animal struct {
  Nom string
  pattes int // Comme les fonctions, le champ pattes ne sera pas exporté car il commence par une minuscule
}
```

Les channels, qui sont un type qui permet de communiquer entre différentes goroutines. Ils peuvent être apparentés à une Queue. Par exemple, un channel de boolean est noté **chan bool**. Nous en verrons plus tard, Les maps : **map[string]string**, **map[string]bool**, **map[int]string** etc., Les interfaces qui n'ont pas besoin d'être explicitement implémentées par un type. Prenons un exemple :

```
type Chien interface {
  Aboie() string
}

type Teckel struct {
  Nom string
}

func (teckel *Teckel) Aboie() string {
  return "ouaf"
}
```

Dans cet exemple, on déclare une interface *Chien*. Un *Chien* est n'importe quel type qui implémente la méthode *Aboie() string*. De ce fait, le type *Teckel* qui possède une méthode Aboie, qui ne prend pas de paramètres et qui retourne une string, implémente l'interface *Chien*.
Le dernier type à noter, qui est l'un des plus importants, est le type **error**. Ce type est en fait une interface :

```
type error interface {
  Error() string
}
```

En go, il n'y a pas d'exceptions. De ce fait, à chaque appel à une fonction qui peut échouer, on retourne une erreur en plus du résultat attendu. Par exemple, voici le prototype de fonction qui permet de créer un fichier (cette fonction est présente dans le package *os*) :

```
func Create(name string) (*File, error)
```

Hello World

Maintenant, voyons tout de suite à quoi ressemble un simple Hello World en Go :

```
package main

import (
  "fmt"
)

func main() {
  fmt.Println("Hello World")
}
```

Décomposons le programme :

- *package main* : cela indique que c'est le package principal du programme, ilcontient la fonction *main* qui est le point d'entrée du programme,
- *import ("fmt")* représente la liste des packages importés, ici nous utilisons le package *fmt* (pour formatted I/O) pour écrire sur la sortie standard,
- *func main* : représente la fonction main (qui ne retourne rien) ; c'est la première fonction appelée lors du démarrage du programme,
- *fmt.Println("Hello World")* appelle la fonction *Println* du package *fmt* ; cette fonction affiche donc "Hello World" ainsi qu'un retour à la ligne.

Il faut savoir que les accolades DOIVENT être au même niveau que la déclaration de fonction. Sinon, le programme ne compile pas.
Nous allons parcourir d'autres petits exemples de code pour voir les spécificités du Go par rapport aux langages plus classiques :

Déclaration de fonctions

```
// Fonction qui ne retourne rien
func doSomething() {
// Do something...
}

// Fonction qui prend un paramètre et qui ne retourne rien
// Notez que le nom de la variable et son type sont inversés par rapport aux langages plus classiques
func doSomethingWithThat(foo string) {
// Do something...
}

// Fonction qui prend plusieurs paramètres et qui retourne un paramètre
// Notez que foo et bar sont tous les deux des strings
// Notez également que le type retourné par la fonction est indiqué juste avant d'ouvrir l'accolade
func computeSomething(foo, bar string, baz int) int {
 // Do something...
 return 42
}

// Fonction qui retourne plusieurs paramètres (oui oui c'est possible)
func returnsALotOfThings() (int, int, string, int, float64) {
  return 24, 24, "toto", 42, 42.42
}
```

Déclaration de variables et inférence de type

```
var strSalut string = "salut"
strSalutations := "salutations"

var thisIsTrue bool = true
thisIsFalse := false

// Par défaut, le type de number sera de int
number := 42

// Par défaut, le type de numberFloat sera un float64
```

```
numberFloat := 42.42

// Les slices sont instanciés à l'aide du mot clé make
var sliceOfInt []int
sliceOfInt = make([]int, 0) // sliceOfInt est maintenant un slice de int de taille 0

// Pour ajouter des éléments à un slice, on utilise le mot clé append
sliceOfInt = append(sliceOfInt, 12, 24, 42)
sliceOfInt[0] == 12
sliceOfInt[1] == 24
sliceOfInt[2] == 42
sliceOfInt[3] // crash du programme puisque le tableau ne contient que 3 éléments

// Exemple de map:
m := map[string]int{
  "reponse": 42,
  "question": 21,
}

reponse := m["reponse"] // reponse == 42
question := m["question"] // question == 21

// Définition d'un objet Animal

type Animal struct {
  Nom string
  Pattes int
}

func main() {
// Déclaration de variables de type Animal :
var chien Animal // chien est de type Animal
chien.Nom = "Rex"

var chat *Animal // chat a la valeur nil par défaut car c'est  un pointeur non instancié
chat.Nom = "Tom" // crash du programme car chat == nil

// Instanciation d'un objet *Animal avec le mot clé new
var poisson *Animal = new(Animal)
poisson.Nom = "Bubble"

// Instanciation d'un object *Animal avec l'inférence de type (ma méthode préférée)
souris := &Animal{
      Nom: "Jerry",
  Pattes: 4,
}
// Cela marche aussi avec une structure qui n'est pas un pointeur :
cheval := Animal{
      Nom: "Jolly",
  Pattes: 4,
}
```

Structures de contrôles

```
func main() {
  // Déclaration d'un tableau de int de 5 éléments :
  arrayInt := []int{1, 2, 3, 4, 5}

  // Les conditions n'ont pas de parenthèses en Go
  if arrayInt [1] == 2 {

} else if arrayInt [3] != 2 {

} else {

}
// Déclaration d'un tableau de strings
arrayString := []string{"salut", "monsieur"}

// On peut switch sur des strings :
switch arrayString[0] {
case "salut":
// Le break est implicite en Go
case "monsieur":
  // Si on veut exécuter le bloc suivant, il faut utiliser le mot-clé fallthrough
  fallthrough
case "madame":

}

// Tout comme les conditions, les boucles n'ont pas de parenthèses :

for i := 0; i < 10; i++ [

}

// On peut boucler sur un array :

for index, valeur := range arrayString {
  fmt.Println(index, valeur)
}
// Dans l'ordre, ça affichera "0 salut" et "1 monsieur"
}
```

Méthodes et Héritage

```
// Définition d'un objet Animal
type Animal struct {
  Nom    string
  Pattes int
}

// Définition d'une méthode Crier pour le type Animal :
func (animal *Animal) Crier() string {
  return "AHHHHH!"
}

// Définition d'un objet Chien qui hérite d'Animal :
type Chien struct {
  Animal
}

// Surcharge de la méthode Crier pour le type Chien :
func (chien *Chien) Crier() string {
  return "OUAF!"
}

func main() {
```

```
    // Instantiation d'un pointeur sur Chien
    chien := &Chien{
        Animal{
            Nom:    "Rex",
            Pattes: 4,
        },
    }
    chien.Crier()       // retourne "OUAF!"
    chien.Animal.Crier() // retourne "AHHHHH!"
}
```

COMMENT CODER EN GO ?

Pour se faire une idée du langage, Go propose *Un Tour de Go* qui permet d'aborder le langage progressivement avec des explications et la possibilité de lancer son programme directement depuis le navigateur. Le tour se passe ici : http://go-tour-fr.appspot.com/welcome/1

Sinon, pour **vraiment** coder en Go, il faut commencer par l'installer. Il faut le télécharger ici : http://golang.org/dl/.

Pour les amateurs de linux, il est souvent disponible dans les dépôts mais c'est très souvent une vieille version, il est donc conseillé de le télécharger quand même (ou de le compiler soit même, c'est très rapide et ne demande que git et gcc).

Installation de Go

Sur Windows, le plus simple est de l'installer via le fichier MSI ; il suffit de suivre l'installation et le binaire **go** sera mis automatiquement dans le PATH. Sur OS X, il suffit de l'installer avec le fichier PKG, de suivre l'installation, et, comme sur Windows, **go** sera mis automatiquement dans le PATH. Faites attention, il faudra rouvrir les terminaux ouverts lors de l'installation pour que ce changement soit pris en compte.

Quel IDE pour Go ?

Le Go n'est pas sorti avec un IDE. Du coup, il existe un grand nombre de plugins pour des IDE déjà existants. Normalement, vous devriez pouvoir coder en Go directement avec votre éditeur préféré, même s'il existe un IDE dédié au Go. Voici la liste des plugins et IDE dédiés au Go :

- Emacs et le *go-mode*. Si vous avez l'habitude d'Emacs, ça peut suffire largement (mes six premiers mois de développement sur Go ont été faits sur Emacs). Ce plugin permet d'avoir une coloration syntaxique, un formatage du code avant chaque sauvegarde, pouvoir accéder à la documentation de n'importe quel package directement depuis emacs et encore d'autres fonctionnalités. Disponible ici : https://github.com/dominikh/go-mode.el
- Vim et le plugin *vim-go*. Ce plugin très puissant inclut les fonctionnalités suivantes : coloration syntaxique, autocomplétion, pouvoir lancer le code directement depuis l'éditeur, etc.. Le plugin est téléchargeable ici : https://github.com/fatih/vim-go
- Sublime Text et le plugin GoSublime. Il semble être l'un des éditeurs les plus utilisés et fournis : autocomplétion, coloration syntaxique, pouvoir lancer le programme depuis l'éditeur, formater le code, analyser les erreurs de syntaxes, pouvoir accéder à la définition d'une variable ou d'une fonction, voir la documentation d'une variable sans changer de fenêtre... Si vous avez l'habitude de Sublime Text, je vous recommande ce plugin : https://github.com/DisposaBoy/GoSublime
- L'éditeur LiteIDE. C'est un éditeur open-source développé en C++ fait spécialement pour Go. Fonctionnant sur à peu près tous les OS (Windows, Mac OS, Linux et OpenBSD), il permet de faire à peu près tout ce que les autres éditeurs de texte situés plus haut peuvent faire, mais rajoutez à tout ça la possibilité de pouvoir débuguer via GDB (à installer sur Windows via MinGW). Il est disponible ici : http://sourceforge.net/projects/liteide/files/
- Le plugin Goclipse pour Eclipse. Il inclut une coloration syntaxique, analyseur d'erreur; autocomplétion, débogage (via GDB), documentation (intégrée à l'autocomplétion) et compilation du programme. Pour l'installer, il suffit de se rendre ici : https://github.com/GoClipse/goclipse/blob/latest/documentation/Installation.md#installation
- Le plugin Go pour Atom qui offre une coloration syntaxique, autocompletion, analyseur d'erreur etc.. Il est disponible ici : https://github.com/joefitzgerald/go-plus
- Le plugin Go pour Intellij IDEA. Personnellement, c'est celui que j'utilise. Depuis quelques temps, le plugin est maintenu par des développeurs de JetBrains (c'est la société derrière l'IDE Intellij) et la version est maintenant en 1.0 alpha. Le plugin inclut coloration syntaxique, téléchargement des dépendances, autocomplétion, documentation via l'autocomplétion, pouvoir accéder à la déclaration de la variable ou fonction, analyseur d'erreur, etc... Il est compatible avec les éditeurs Intellij IDEA (version gratuite et Ultimate), PyCharm, WebStorm et est téléchargeable ici : https://github.com/go-lang-plugin-org/go-lang-idea-plugin/releases

Pour la suite, j'utiliserai Intellij.

Le workspace

L'outil *go* est fait pour travailler avec un code open source maintenu dans des dépôts publics. Même si le code n'est pas publié, la hiérarchie des dossiers sera la même. Le *workspace* consiste en trois dossiers à la racine :

- *src* qui contient toutes les sources organisées en packages (un package par dossier),
- *pkg* qui contient les fichiers générés par la compilation,
- *bin* qui contient les différents binaires.

Pour avoir une meilleur idée de ce à quoi ressemble un *workspace* en pratique, voici un exemple : Fig.2.

Ce *workspace* contient un dépôt Git (exemple) contenant deux exécutables (hello et world) et une bibliothèque (stringutil). Les bibliothèques et les exécutables sont générés depuis différents types de packages : les packages principaux (**main**) et tous les autres packages (par exemple le package **stringutil**). Lorsque nous définissons un package comme étant principal, il faut qu'une fonction main soit présente pour que la compilation s'effectue.

La variable d'environnement GOPATH

La variable d'environnement GOPATH spécifie l'emplacement de votre *workspace*. Il est possible d'avoir plusieurs *workspaces*, mais il est conseillé de travailler avec un seul pour plus de facilité.

Si votre GOPATH est **/user/vincent/projects/go** alors votre dossier go devra contenir les dossiers src, bin et pkg.

Par la suite, nous allons considérer que le GOPATH est **/user/vincent/projects/go.**

Les packages

Les packages de la bibliothèque standard ont des noms courts comme **fmt** ou **net/http**. Pour vos propres packages, essayez de choisir des noms qui ne risquent pas d'entrer en collision avec d'autres bibliothèques.

Votre premier programme

Nous allons réaliser un programme qui va utiliser toute la simplicité de Go pour résoudre des problèmes parfois trop compliqués pour pas grand-chose dans d'autres langages : le modèle du producteur -

```
bin/
    hello                       # éxécutable
    world                       # éxécutable
pkg/
    windows_amd64/
        github.com/programmez/exemple/
            stringutil.a        # fichier généré après compilation du package stringutil
src/
    github.com/programmez/exemple/
        .git/                   # Données du dépôt Git
        hello/
            hello.go            # fichier source d'un exécutable
        outyet/
            main.go             # fichier source d'un exécutable
            main_test.go        # fichier de test
        stringutil/
            reverse.go          # fichier source d'une library
            reverse_test.go     # fichier de test
```

Fig.2

consommateur. Premièrement, il faut créer le package principal. Nous allons le nommer **procons** et sera localisé dans le dossier **src** du GOPATH : **/user/vincent/projects/go/src/procons**.
Du coup, le package étant principal, nous allons créer un fichier **main.go** ne contenant rien de spécial pour l'instant :

```
package main

func main() {

}
```

Si vous compilez ce programme (**go install procons**), dans le dossier bin (à la racine du GOPATH) se trouvera un binaire qui ne fera strictement rien. Maintenant parlons du programme. Le modèle du producteur - consommateur est un modèle parfait pour mettre en avant les différents problèmes de synchronisation de ressources dans un environnement multi-threadé, et il permet donc d'apprendre à se servir de mutex. Pour rappel, un mutex est une sécurité permettant (notamment) de s'assurer que plusieurs threads ne modifient pas en même temps la même variable. Dans le cas de Go, ce modèle peut être réalisé sans avoir recours aux mutex grâce au type très spécial du Go qu'est le **channel**. Un channel est en fait une queue de données sur laquelle on peut lire et écrire en même temps. Pour commencer, nous allons créer un modèle : une banane, nous allons créer et consommer des bananes.
Nous allons créer un nouveau sous-package **models,** et dedans, nous allons créer un fichier **banane.go**.
Dans ce fichier, nous allons créer un objet Banane qui possède une couleur ainsi qu'une liste de couleurs possibles :

```
package models

const (
  BananeJaune = "jaune"
  BananeVerte = "verte"
  BananeNoire = "noire"
)

type Banane struct {
  Couleur string
}
```

Ensuite, nous allons créer un objet Consommateur qui s'occupera de manger des bananes, il aura une méthode Manger qui prendra une Banane en paramètre. Nous allons donc créer un fichier **consommateur.go** dans le package models :

```
package models

import "fmt"

type Consommateur struct {
 BananesMangées int
}

func (consommateur *Consommateur) Mange(banane *Banane) {
 switch banane.Couleur {
 case BananeVerte:
   fmt.Println("C'est pas mur!")
 case BananeJaune:
   fmt.Println("C'est parfait")
 case BananeNoire:
   fmt.Println("Beurk!")
 }
 consommateur.BananesMangées++
}
```

Nous n'avons plus qu'à réaliser le producteur maintenant : pareillement, dans un fichier **producteur.go** dans le package models. Ce producteur aura une méthode FabriqueBanane() qui produira une banane avec une couleur aléatoire :

```
package models

import "math/rand"

type Producteur struct {
}

func (producteur *Producteur) FabriqueBanane() *Banane {
 nombreAleatoire := rand.Intn(3)
 banane := &Banane{}
 switch nombreAleatoire {
 case 0:
   Banane.Couleur = BananeJaune
 case 1:
   banane.Couleur = BananeVerte
 case 2:
   banane.Couleur = BananeNoire
 }
 return banane
}
```

Après avoir créé tous ces fichiers, vous devriez avoir cette hiérarchie : Fig.3. Maintenant que nous avons tous les modèles, il faut réaliser la boucle de création. Il y aura un producteur qui fabriquera des bananes, et il y aura un consommateur qui mangera une banane dès que le producteur la lui fournira. La boucle de production sera infinie et le consommateur arrêtera de manger au bout de cinq bananes. La boucle de production sera dans un processus à part de la boucle de consommation qui sera dans le processus principal. Il faut donc lancer un processus à part. Non, nous n'allons pas utiliser la bibliothèque pthread, mais nous allons utiliser le mot-clé du langage Go : *go*. Lorsqu'il est placé devant

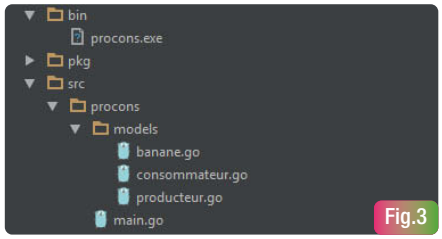


Fig.3

un appel de fonction, *go* permet de lancer cette fonction dans un processus à part; ce processus, très léger, s'appelle une goroutine. Retournons dans le package principal pour faire la boucle :

```
package main
import (
  "procons/models"
  "fmt"
)

func produit(queueBanane chan *models.Banane) {
  producteur := models.Producteur{}
  for {
      // On fabrique une banane
      banane := producteur.FabriqueBanane()
      fmt.Println("Banane produite !")
      // Cette syntaxe un peu bizarre signifie qu'on ajoute une banane dans le channel
      queueBanane <- banane
  }
}

func consomme(queueBanane chan *models.Banane) {
  consommateur := &models.Consommateur{}
  // On boucle sur le channel, à chaque fois qu'une banane sera ajoutée au channel, nous
réaliserons un tour de boucle
  for banane := range queueBanane {
    consommateur.Mange(banane)
    if consommateur.BananesMangées == 5 {
      fmt.Println("J'ai fini de manger !")
      return
    }
  }
}

func main() {
  // On instancie un channel de banane
  queueBanane := make(chan *models.Banane)

  // On lance dans un processus à part la production de bananes.
  go produit(queueBanane)
  consomme(queueBanane)
}
```

Si vous compilez le programme et que vous le lancez, vous aurez des messages de ce type : Fig.4. Votre affichage peut différer puisque nous avons fait un Random. Mais ce qu'on peut voir, c'est que le producteur n'attend pas que le consommateur ait fini de manger pour continuer de produire des bananes. Si on veut que ça soit le cas, il suffit de rajouter un autre channel qui servira à notifier au producteur qu'il peut produire une nouvelle banane :

```
package main

import (
 "fmt"
 "procons/models"
)

func produit(queueBanane chan *models.Banane, queueNourriture chan bool) {
 producteur := models.Producteur{}
 for {
   banane := producteur.FabriqueBanane()
   fmt.Println("Banane produite !")
   queueBanane <- banane
    // On attend que le consommateur finisse de manger pour continuer la boucle.
   <- queueNourriture
 }
}

func consomme(queueBanane chan *models.Banane, queueNourriture chan bool) {
 consommateur := &models.Consommateur{}
 for banane := range queueBanane {
   consommateur.Mange(banane)
    // On notifie le producteur qu'on a fini de manger.
   queueNourriture <- true
   if consommateur.BananesMangées == 5 {
     fmt.Println("J'ai fini de manger !")
     return
   }
 }
}
func main() {
 queueBanane := make(chan *models.Banane)
 // Dans ce cas, on se fiche du type du channel.
 queueNourriture := make(chan bool)
 go produit(queueBanane, queueNourriture)
 consomme(queueBanane, queueNourriture)
}
```

Maintenant, notre programme devrait afficher quelque chose de ce type : Fig.5. Voilà, vous avez créé votre premier programme en Go. Programme assez simple mais qui permet de voir beaucoup de particularités du Go par rapport aux autres langages : on a pu voir la création d'objets et de méthodes et l'instanciation de ces objets, la création et l'utilisation de channel, l'utilisation de packages et surtout la création des goroutines via le mot-clé *go*. Le langage est jeune, mais sa communauté est grandissante et le langage lui-même incite à faire de l'open source. Les sites Web http://godoc.org et http://go-search.org cherchent sur github, bitbucket et code.google.com les packages de n'importe quel utilisateur. Si vous créez un dépôt public sur github et que les fichiers sources sont à la racine du dépôt, votre package sera alors trouvable via ces deux sites. Ces outils permettent d'accélérer grandement le développement via la réutilisation d'outils déjà créés par d'autres utilisateurs. En bref, le Go est un langage simple, qui n'a peut-être pas toutes les fonctionnalités du C++ ou encore du Java ou C#, mais il n'a pas cette prétention, il se veut juste être le plus simple possible. Et c'est le cas, le Go, c'est simple à lire, à utiliser, et se maîtrise rapidement.

```
C:\Users\Vincent\IdeaProjects\PremiersPasGo\bin>procons.exe
Banane Produite !
Banane Produite !
Beurk!
C'est parfait
Banane Produite !
Banane Produite !
Beurk!
Beurk!
Banane Produite !
Banane Produite !
C'est pas mur!
J'ai fini de manger !
```

Fig.4

```
C:\Users\Vincent\IdeaProjects\PremiersPasGo\bin>procons.exe
Banane produite !
Beurk!
Banane produite !
C'est parfait
Banane produite !
Beurk!
Banane produite !
Beurk!
Banane produite !
C'est pas mur!
J'ai fini de manger !
```

Fig.5

Back from Build 2015

Florent SANTIN
fsantin@infinitesquare.com
http://blogs.infinitesquare.com/b/florent / @SantinFlo
INFINITE SQUARE

//Build/ est l'évènement Microsoft majeur consacré aux développeurs. Cet évènement a lieu chaque année depuis 2011, et est venu succéder à la PDC (Professional Developers Conference) créée en 1992 !
//Build/ a été créée pour le lancement de Windows 8, sous l'intitulé « Build Windows », mais adresse aujourd'hui l'ensemble des développeurs de l'écosystème Microsoft.
L'édition 2015 s'est déroulée du 29 Avril au 1er Mai au Moscone Center à San Francisco, avec toujours autant d'impatience de la part des développeurs afin de savoir ce que Microsoft prépare. Au niveau des annonces, l'évènement s'est déroulé autour du thème « la transformation de la plateforme Microsoft ». Cette transformation s'axe selon trois points : le Cloud avec Microsoft Azure, Office en tant que plateforme via Office 365, et, enfin, bien sûr, Windows 10.

La montée en puissance d'Azure

L'évènement a été l'occasion de présenter quelques chiffres sur l'évolution d'Azure. D'un point de vue purement infrastructure, Microsoft est l'opérateur de Cloud disposant du plus grand nombre de Datacenters au monde, avec 19 centres actifs déployés sur chaque zone continentale. Azure continue sa montée en puissance, et opère aujourd'hui plus d'1 million de serveurs dans le monde, ainsi qu'environ 1,5 million de bases de données. Chaque mois, ce sont 90 000 nouveaux abonnements (souscriptions) Azure qui sont activés. Côté développeur, ce qui est intéressant dans Azure, c'est sa Roadmap très active. Contrairement à Windows qui dispose d'un cycle d'annonces et de nouveautés maintenant annuel (même si c'est mieux que les 3 à 5 ans entre chaque version du passé), Azure est en constante évolution, avec des nouveautés au mois. En un an, il y a eu plus de 500 nouvelles releases des différents services d'Azure, de quoi couvrir un évènement dédié. Ce sont donc les annonces du mois de Mai qui ont été présentées et ont retenu l'attention des participants.

La simplification et l'évolution du PaaS

Au niveau de sa plateforme PaaS, Microsoft a simplifié l'existant et présenté son futur.
La simplification de l'existant est passée par un regroupement d'un ensemble de fonctionnalités d'Azure sous le nom **Azure Services**. Celui-ci propose des modules d'exécution de services Web, aujourd'hui au nombre de 4 : **Azure Web Apps**, anciennement Azure Websites pour héberger en quelques clics un site Web en mode Cloud ; **Azure Mobile Apps**, ex mobile services – un « back end as a Service » pour applications mobiles ; **Azure API Apps** - un nouveau template pour héberger, exposer, connecter des APIs et enfin **Azure Logic Apps** qui permet d'interconnecter visuellement dans des process des APIs et des services.

Windows 10 IoT sur Raspberry Pi 2, ça va piquer les yeux ! A suivre dans le numéro d'été de Programmez!

L'évolution de l'existant, et l'avenir de la plateforme PaaS, ont été présentés au travers du service **Azure Service Fabric**. Il s'agit du cerveau d'Azure, celui qui opère et isole déjà grand nombre de services sur Azure, tels que les WebJobs ou les bases de données SQL Azure... La nouveauté réside dans la mise à disposition de celui-ci au travers d'un SDK qui permet d'opérer des packages de déploiement pour Azure dans un Datacenter privé, avec un seul script de définition du groupe de ressources. Cette annonce a été complétée la semaine suivante lors de l'évènement Microsoft Ignite, équivalent de //Build/ pour les IT Pro, lors duquel a été annoncé Microsoft Azure Stack, permettant de disposer d'un mode d'hébergement identique à Azure dans son propre Datacenter.

Du côté de SQL...

Les bases de données dans Azure ont également eu droit à leur nombre d'annonces. Afin de toujours réduire la différence entre les bases de données hébergées en mode PaaS dans Azure et celles on-premise, Microsoft continue de faire converger les fonctionnalités des produits SQL (Azure) DataBase et SQL Server. Ceci s'illustre notamment par l'arrivée dans le Cloud de fonctionnalités jusqu'alors cruellement manquantes tel que le Full Text Search.
SQL DB Elastic Pool est une nouvelle fonctionnalité de SQL Database annoncée. Celle-ci permet de regrouper de manière intelligente des bases de données entre elles, et d'effectuer du sharding automatique en fonction de la consommation en performances de chaque base. Les bases de données sont ainsi regroupées ou isolées en fonction des besoins. Cela permet également de requêter l'ensemble des données des bases isolées dans le groupe, au travers d'une seule chaîne de connexion, et de gérer les évolutions de schéma pour chaque base en exécutant des jobs transverses. Coté Business Intelligence, **SQL Data Warehouse** est un nouveau service d'Azure qui permet de créer un Data Warehouse en quelques minutes, de gérer le flux d'imports de données, d'exploiter celles-ci via PowerPI, mais aussi d'ajouter de la prédiction de données grâce à Azure Machine Learning. Enfin, pour nos futurs objets connectés, **Azure Data Lake Service** est un système de stockage de données brutes de volumétrie infinie. Non relationnel, ce mode de stockage est dédié et pensé pour les scénarios de collecte massive de données IoT. Azure est depuis plusieurs années orienté **Open Source**, avec le support de tous types de langages (Python, Ruby) ou de produits (Hadoop, Redis, DocumentDB), mais ce mouvement s'est encore amplifié cette année, avec notamment la contribution apportée par Microsoft sur le projet **Docker**, afin de supporter cette technologie de conteneurs dans Azure.

Microsoft, développement et Open Source

La tendance à l'Open Source de Microsoft s'est également illustrée dans les démonstrations des technologies de développement, au travers d'exécution de code **ASP.NET sous Linux**, debuggé à distance depuis Visual Studio.
Le projet Apache Cordova, permettant de faire du développement mobile multiplateformes en HTML5 / CSS a également été mis à l'honneur, car maintenant nativement supporté par **Visual Studio 2015**. Pour simplifier le développement d'applications Android, l'éditeur de code de Microsoft se voit d'ailleurs doté de son propre émulateur **Android**. Enfin, Microsoft a annoncé la sortie de

Visual Studio Code, un éditeur de code **gratuit** fonctionnant à la fois sous OSX et Windows. Il propose quelques fonctionnalités de Visual Studio telles que l'intellisence, le debugging, la comparaison de code, et a surtout le grand intérêt de disposer d'une intégration native à Visual Studio Online pour le contrôle de source.

Office as a Platform

Les annonces autour d'Office 365 ont tourné autour de l'évolution de son modèle de développement côté client et côté serveur. Côté client, le nouveau mode d'applications office 365 permet de créer des Addins à Word, Excel, PowerPoint, Outlook qui, avec **un seul code**, fonctionnent sur toutes les versions d'Office : PC, Web (online dans Office 365), mais également sur la version iPad. Ces applications peuvent être distribuées et déployées publiquement au travers du Store d'Office 365 ou bien de manière privée dans un Store d'entreprise. Côté serveur, Office 365 dispose maintenant d'un système d'API unifiée pour être manipulée plus simplement par programmation. Il s'agit de **l'Office365 Graph**, pensée sur un modèle d'API REST / JSON, point d'entrée unique pour manipuler au travers de n'importe quel langage de développement l'ensemble des services actuels et futurs.

Une nouvelle génération de Windows

Windows 10 a été présenté comme une **nouvelle génération de Windows**, avec une plateforme vraiment unifiée en termes de développement. L'objectif de Microsoft annoncé, est, sous 3 ans, d'avoir Windows 10 déployé sur **1 milliard de devices**. Ce qui est plus que n'importe quelle autre plateforme. Par device est entendu ici tout objet capable de faire tourner du Windows 10, que cela soit du PC, de la tablette, du smartphone, mais également de l'objet connecté avec une version de Windows 10 dédiée. Avec Windows 10, Microsoft donne clairement un coup de balais sur Windows 8, et fait de même dans le monde du Web. Le **"Project Spartan"** devient **"Microsoft Edge"**. Il s'agit du navigateur Web, en charge de remplacer Internet Explorer, en repartant sur une base de fonctionnement plus simple, plus moderne, et respectant complétement les standards HTML.

Des applications vraiment universelles

La plus grosse annonce pour les développeurs Windows 10 concerne les 4 nouvelles façons de faire tourner du code pour la plateforme Universelle de Windows 10 :

- Publication d'un **site Web au travers du store Windows**. Un site Web peut être référencé sur le store et, une fois installé, va pouvoir tourner dans une App et non plus dans le navigateur Web. Ceci permet de mettre en place par exemple des notifications sur un site Web et de disposer de tuile dynamique ou de l'intégration de Cortana.
- **.NET et Win32** : il sera possible de packager une application ancienne génération pour la publier dans le store Windows. Ceci permet une installation rapide grâce à un mode d'exécution dans un contexte isolé. Finis donc les programmes d'installation qui trainent avec eux des dizaines de dépendances pour fonctionner, et qui laissent tout en place sur la machine lors de la désinstallation : cela devient transparent.
- **Android Java et C++**: une des plus grosses annonces concerne la possibilité de faire tourner du code ciblant Android sous Windows 10 Mobile. Ce code est exécuté sous Windows sans changement de code, avec juste l'ajout de fonctionnalités supplémentaires liées aux API Windows Phone et l'intégration du système de navigation Windows Phone.
- **Objective C / IOS** : autre grosse annonce, le code Objective C peut être compilé depuis Visual Studio pour devenir également une Application Universelle Windows.

Il est donc maintenant vraiment simple pour un développeur de cibler l'intégralité de la plateforme Windows (PC, Xbox, Hololens, Nano PC, Tablettes, Smartphone...) : Développement natif C#, Multiplateformes avec Xamarin ou Cordova, Packaging de site Web, développement Win32 / WPF ou compilation de code IOS / Android... tous les chemins sont couverts, et tous les outils sont disponibles.

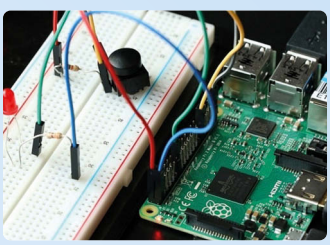
Expérience continue avec Continuum

Le concept **"Continuum"** a été présenté pendant //BUILD. Il s'agit de la logique de Windows 10 qui permet d'adapter l'écran et le rendu des applications en fonction du mode d'utilisation du device : en mode PC ou en mode tablette. Vu que le même Windows tourne maintenant sur téléphone et PC, les mêmes applis doivent pouvoir fonctionner avec les mêmes fonctionnalités, juste avec un rendu différent en fonction de l'usage. Une application en mode tablette met en avant le tactile et le stylet, en PC, elle s'appuie sur le clavier, sur un téléphone, elle change de mode de navigation. Une impressionnante démonstration de ce concept a été effectuée au travers d'un Windows Phone connecté à un écran et un clavier. Les applications comme Powerpoint et Outlook installées sur le téléphone se sont retrouvées utilisables comme sur un PC ! Téléphone + grand écran + clavier = PC pour Windows 10.

En résumé

Cette version 2015 de //BUILD/ est sans aucun doute la plus riche en termes d'annonces et d'avancées de Microsoft depuis la toute première édition et la sortie de Windows 8. Microsoft persiste et signe clairement dans sa stratégie d'ouverture vers **l'Open Source** et **l'Ouverture de son écosystème** pour permettre de valoriser ses **plateformes** (Windows, Azure, Office365) et ses **Hardwares** (Xbox, Surface, Lumia, Surface Hub, Hololens).

WINDOWS REMOTE ARDUINO ET VIRTUAL SHIELD



Le pont vers l'IoT ne passera pas uniquement par des applications Windows 10 sur des machines telles que le Raspberry Pi. Microsoft l'a bien compris et Windows 10 devient le premier système "Certifié Arduino". Un partenariat caractérisé par une intégration profonde de la couche IoT de Windows 10 avec Arduino.

Windows Remote Arduino

Les applications Windows 10 auront la faculté de communiquer avec des programmes tournant sur Arduino de manière bidirectionnelle délaissant par exemple les tâches purement électroniques aux cartes Arduino, et la puissance de calcul ou de stockage aux devices sous Windows 10. Cette communication permettra également la création d'écosystèmes de plusieurs appareils et cartes aux fonctions dédiées.

Windows Virtual Shield for Arduino

Plus étonnant encore, nous aurons la possibilité de créer des shields virtuels pour Arduino depuis Windows 10. Ici ce seront les cartes Arduino que l'on connectera à des appareils sous Windows 10, et ces PC, tablettes et même téléphones exposeront à nos programmes Arduino, leurs capteurs et autres capacités logicielles. Une idée maline pour exploiter les capteurs (GPS, accéléromètre, etc.) dont sont équipés nos téléphones, et pour prototyper sans se ruiner. Arduino étant un des leaders du prototypage et de l'IoT, ce mariage technique est une bonne idée de Microsoft pour consolider sa place dans l'Internet des Objets tout en s'appuyant sur une communauté et des standards déjà existants.

Windows 10 Universal Apps et IoT Core

En parallèle des annonces sur les Windows Platform Applications ou autres projets Centennial, Islandwood, etc, Microsoft a également annoncé un changement important dans sa roadmap des versions Windows. L'universalité des applications devient concrète et s'étend jusqu'aux objets connectés.

Michaël FERY
Consultant
mfery@infinitesquare.com
http://blogs.infinitesquare.com/mfery

Jonathan ANTOINE
Consultant
jantoine@infinitesquare.com
http://blogs.infinitesquare.com/jonathan

INFINITE SQUARE

Universal Apps

Avec Windows 10, Microsoft entend briser la segmentation de son système d'exploitation, et dans ce sens, de gros changements ont été annoncés côté interface graphique.

Quand Microsoft nous annonce que l'on aura la possibilité de créer des applications « universelles », il ne faut pas prendre cette annonce à la légère. Il faut entendre par là qu'une même application pourra être exécutée sur n'importe quel appareil faisant tourner Windows 10, quels que soient le matériel ou la taille de l'écran. Continuum, la nouvelle fonctionnalité de Windows 10 Mobile en est l'exemple flagrant puisqu'une application exécutée depuis un téléphone auquel on adjoindrait clavier/souris Bluetooth et écran se comportera comme si elle était exécutée depuis un PC Fig.1.

Côté développeur, cette convergence avait déjà débuté avec Windows 8.1 et Windows Phone 8.1 et les Universal Apps qui permettaient de factoriser une grande partie du code métier et IHM. Fort du succès de cette démarche auprès des développeurs, les équipes de Microsoft sont allées au bout de la convergence. Désormais, avec Windows 10, un seul développement (un seul et même code) pourra être réalisé pour cibler toutes les plateformes.

Au-delà d'unifier les stores et d'y apporter de la cohérence pour les utilisateurs, cela permettra aux développeurs de n'avoir à générer et déployer qu'un package. Alors que passer du développement WPF à Windows 8 ou Windows Phone était déjà aisé, il faut se dire que désormais, il n'y aura qu'un seul développement pour les applications natives. Une restriction toutefois, il faudra bien évidemment gérer les différentes résolutions comme le font déjà les développeurs de sites Web. Si cette considération était déjà existante pour Windows 8 PC et tablettes, il faudra désormais prendre en compte les téléphones et les objets connectés aux écrans bien plus restreints.

Un peu à la manière d'un site responsive, il faudra donc maintenant penser son application de manière réellement responsive, de l'écran 4K au téléphone, voire au Raspberry Pi 2 équipé d'un mini écran.

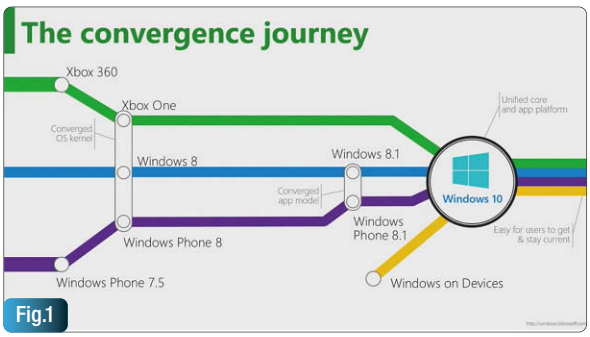


Fig.1

Windows 10 IoT Core

Nous devons également accueillir des nouvelles dans la liste des versions Windows. Similaires aux cartes programmables Arduino, les cartes Raspberry Pi 2, Galileo ou Minnowboard Max ont désormais leurs propres versions de Windows 10 leur permettant également de faire fonctionner les mêmes applications que leurs grandes sœurs.

Qu'est-ce que cela implique pour les développeurs ? Aucune nouvelle compétence ne sera nécessaire si vous souhaitez faire fonctionner vos applications Windows 10, puisque celles-ci tourneront sans problème sur ces petits appareils. Certaines limitations seront probablement présentes mais la majeure partie des applications n'auront aucun mal à tourner et ne nécessiteront aucun développement supplémentaire. En revanche, si vous le souhaitez, vous pourrez ajouter des fonctionnalités à vos applications s'exécutant sur ces plateformes.

Sous Windows 8 il était par exemple déjà possible d'adjoindre à nos applications l'utilisation du SDK Handled pour accéder aux fonctionnalités Point Of Sell de certains appareils équipés de lecteurs de cartes magnétiques, de code-barres ou autres. De la même manière, sous Windows 10, nous pouvons utiliser le SDK Windows 10 IoT Core. Disponible sous forme de package NuGet gratuit, il offre à nos applications Windows 10, les fonctionnalités supplémentaires d'accès aux entrées/sorties des broches d'un Raspberry Pi 2 ou toute autre carte compatible. Il vous est déjà possible de commencer à développer pour ces plateformes en y installant Windows 10 IoT Core Insider Preview et en développant depuis un PC sous Windows 10 avec Visual Studio 2015.

Vous trouverez toutes les étapes nécessaires à vos premiers pas sur le site windowsondevices.com

Puisque les exemples parlent mieux que les mots, Microsoft propose sur la plateforme hackster.io des exemples d'applications Windows 10 qui interagissent avec des composants électroniques physiques telles qu'une LED ou des capteurs météo. Je vous invite fortement à parcourir les exemples de code pour connaître l'étendue des fonctionnalités disponibles.
https://www.hackster.io/windowsiot

Comme il a été annoncé lors de la Build, « Si vous êtes développeur Windows 10, vous êtes déjà développeur IoT ».

Visual Studio Code : le nouvel IDE multiplateformes

Jérémy LANDON
Consultant Infinite Square

INFINITE SQUARE

Visual Studio fait partie des meilleurs IDE du marché, avec sa version 2015 Microsoft y apporte bon nombre de nouveautés et confirme son statut de référence dans le domaine. Dans son optique d'ouverture qui s'est confirmée avec les annonces de la Build, Microsoft s'offre un nouvel outil de développement qui s'ajoute à la suite Visual Studio nommé Visual Studio Code (VSCode).

Qu'est-ce que VSCode ?

VSCode, le nouveau dans la suite Visual Studio, est disponible sur code.visualstudio.com. Pour faire court, c'est un IDE qui est avant tout : simple, très léger, très rapide, **gratuit**, mais surtout utilisable sur Windows, Linux et Mac ! Visuellement très proche de Monaco, le rapprochement en termes de fonctionnalités et de base technique se fera vite sentir. Le socle technique de VSCode annonce la couleur; réalisé à l'aide du framework Electron Shell (et implicitement Chromium et io.js), utilisant Roslyn, TypeScript, et le moteur de debug Visual Studio, le nouvel IDE de Microsoft repose sur ce qui se fait de mieux.

Que permet-il de faire ?

La force de VSCode est d'offrir un large panel de technologies supportées :

- Coloration syntaxique et auto-complétion sur **25 langages** : Batch, C++, Closure, Coffee Script, DockerFile, F#, Go, Jade, Java, HandleBars, Ini, Lua, Makefile, Markdown, Objective-C, Perl, PHP, PowerShell, Python, R, Razor, Ruby, SQL, Visual Basic, XML,
- Fourni par Roslyn et OmniSharp, **l'Intellisense** est disponible pour le CSS, HTML, JavaScript, JSON, LESS, SASS, C# et TypeScript,
- Enfin l'aide au **refactoring** et la recherche de référence sont utilisables pour C# et le TypeScript. Fig.1.

Et la liste est sans limite, en effet VSCode offre un système d'extension de langage (non ouvert pour l'instant) permettant d'y greffer le support de n'importe quel langage. Actuellement en preview, VSCode ne permet pas de créer des projets (comme peut le faire Visual Studio), il n'est pour l'instant possible de réaliser que des fichiers et dossiers.

VSCode est un vrai IDE

VSCode ne se contente pas de l'aide à l'édition de code, comme peut très bien le faire notepad++ ou sublimeText. Le nouvel IDE de Microsoft offre toutes les fonctionnalités d'un vrai IDE.

Debugging

VSCode permet de réaliser du vrai Debugging pas-à-pas, avec possibilité d'analyser le contexte des variables durant le déroulement de l'application. Très puissant, il a l'avantage de se baser sur un fichier nommé launch.js (à inclure dans la solution) qui permettra potentiellement, à terme, d'y greffer n'importe quel système de debugging. Détail intéressant, VSCode étant basé sur Chromium, le mode développeur est toujours accessible (via la touche F12), celui-ci permettra en plus le débugging de faire le tour de l'IDE qui reste entièrement fait en Javascript/Html.

Contrôle de source

VSCode propose nativement la possibilité d'archiver son code sur GIT (et de ce fait sur Visual Studio Online). Tout repose sur du standard. Il est de ce fait possible de commander les actions en ligne de commande, mais

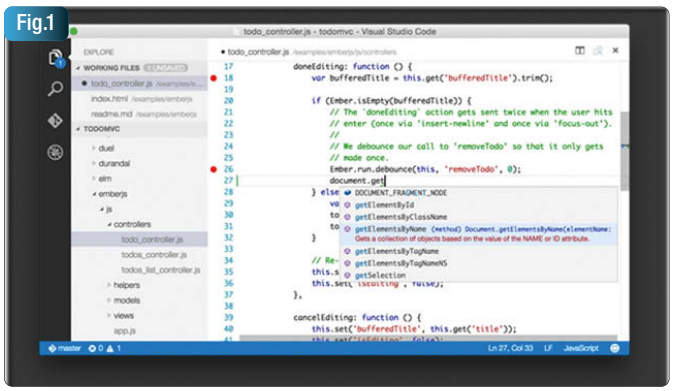


Fig.1

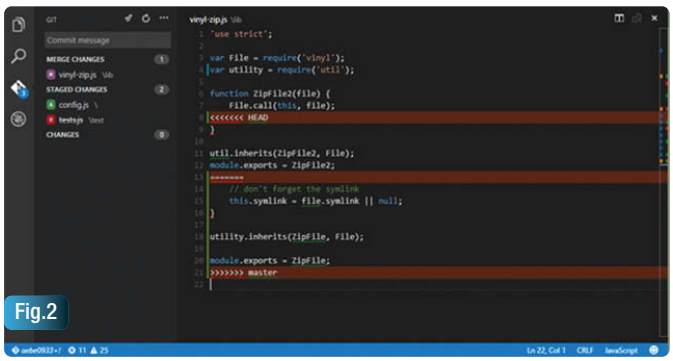


Fig.2

VSCode propose, comme Visual Studio, une interface permettant de réaliser les actions de base (commit, pull, gestion des conflits...). L'intégration est basique mais a l'avantage d'être très bien réalisée et facile à prendre en main Fig.2.

Un peu de Visual Studio et de nouveautés en plus

- Besoin de scripter certaines actions ? VSCode propose son propre système de Task runner, qui permet par exemple de réaliser facilement la génération des fichiers TypeScript, ou encore qui permet d'appeler des outils externes (Make, Gulp, MSBuild...).
- L'AutoSave permet d'enregistrer le fichier au fur et à mesure de son édition de façon automatique.
- Entièrement personnalisable :
  - Possibilité de modifier les thèmes de base ou de se créer les siens,
  - Possibilité de modifier et ajouter des raccourcis.
- Quelques fonctionnalités de Visual Studio :
  - Snippets,
  - Go to definition,
  - Peek Definition (et Hover : complémentaire à Peek Definition, il permet d'obtenir dans le cas du CSS les éléments sur lequel une valeur s'applique) Fig.3.

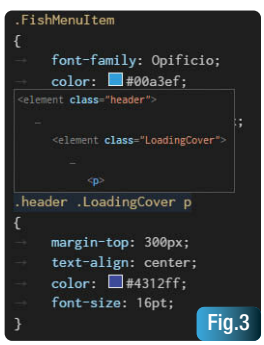


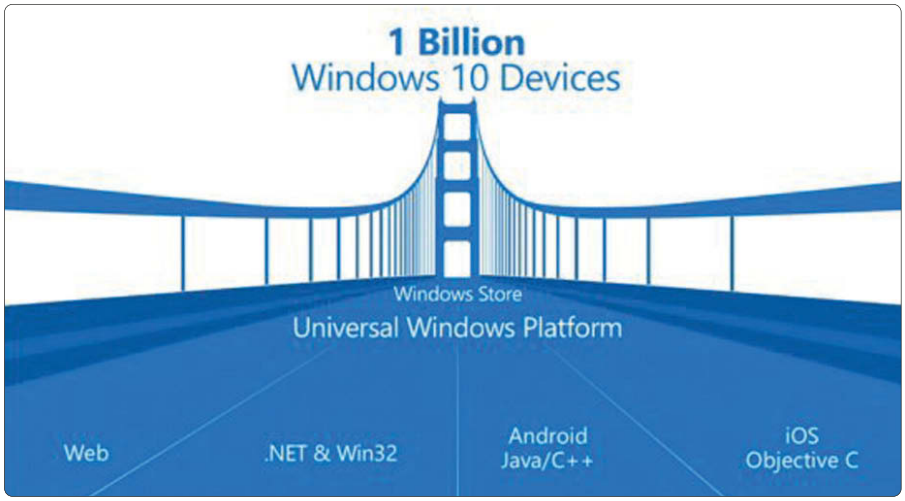
Fig.3

Un petit IDE qui a tout pour devenir grand

VSCode a tout pour devenir une référence. Extrêmement léger et rapide, son intégration à l'explorateur Windows permettra de l'utiliser dans n'importe quel contexte. Officiellement « fermé », il est fort à parier que Microsoft ouvrira cet IDE lorsqu'une version plus stable en sera disponible pour que tout le monde puisse l'accommoder à ses besoins. En résumé, une très bonne surprise de la part de Microsoft, et un produit dont il serait dommage de se passer.

Les Universal Apps en force

Lors de la //Build 2015, Microsoft a annoncé un grand nombre de nouveautés autour du concept « d'Universal Apps », qui, au final, va bien plus loin que le simple fait d'avoir une application qui s'exécute sur téléphone, tablette, PC, Raspberry, etc.



Thomas LEBRUN
Consultant
tlebrun@infinitesquare.com
http://blogs.infinitesquare.com/b/tom

Tout d'abord, il y a les besoins de réutilisation. Typiquement, un grand nombre d'applications Web continuent d'être développées à l'heure actuelle, et Microsoft a cherché un moyen de permettre aux développeurs de « packager » ces applications pour les rendre accessibles à travers le Store; ceci tout en leur permettant d'utiliser des fonctionnalités natives de l'OS (comme les notifications). L'idée est de permettre de simplifier la distribution et l'installation de ces applications. Dans la continuité de cette volonté, Microsoft a annoncé le projet « Centennial ». Derrière ce nom quelque peu « barbare » se cache la volonté d'exploiter au maximum le Store pour permettre aux utilisateurs d'installer également des applications Win32 ! Ainsi, si vous disposez d'une application Windows Forms, WPF ou VB, vous n'aurez plus besoin de vous compliquer la vie pour la partie déploiement : tout sera pris en charge automatiquement par le Store (Microsoft a d'ailleurs, pour les besoins de sa démonstration, montré comment il avait été possible de déployer Photoshop, donc une application Win32, depuis le Store).

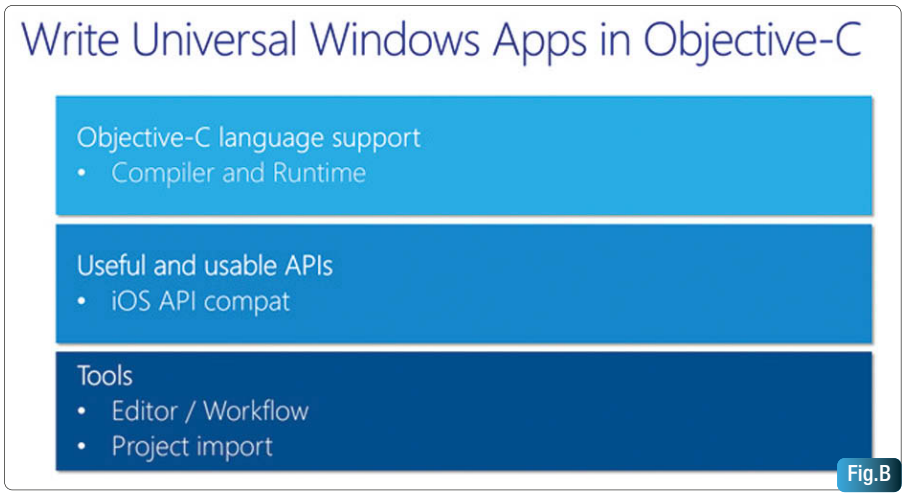


Fig.B

L'avantage de ce système ? Tout d'abord les mises à jour, qui seront entièrement automatiques puisque pilotées par le Store. Mais il y aussi le fait qu'il sera possible, en modifiant légèrement votre code, d'utiliser (comme pour les applications Web) les APIs de la plateforme pour intégrer des fonctionnalités Windows 10 au sein de votre application Win32. Sous le capot, le processus est divisé en plusieurs étapes :

- Génération d'un MSI (cette étape ne change pas de ce qui se faisait avant),
- Etude du MSI par un outil Microsoft, chargé d'analyser les dépendances et les impacts potentiels,
- Génération d'un APPX (comme on le connaît sous Windows 8.x) avec un fichier Manifest spécifique,
- Publication sur le Store. Fig.A.

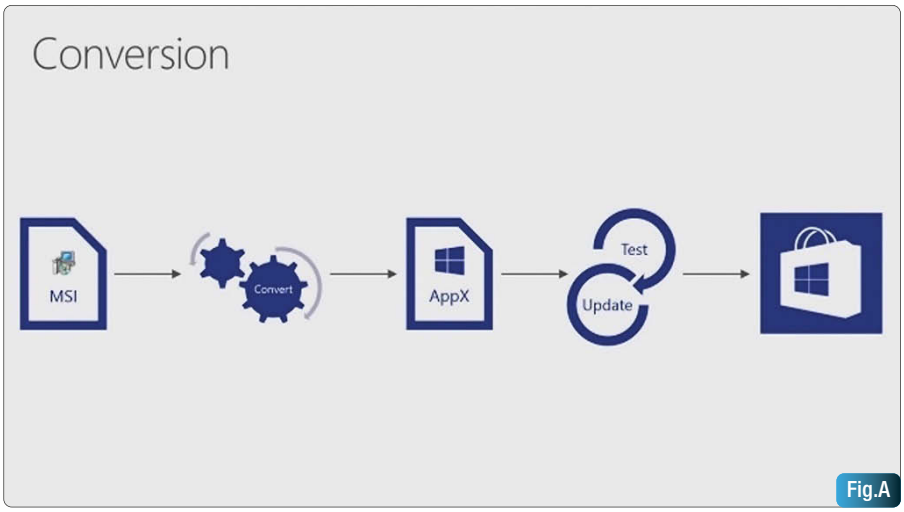


Fig.A

Pour en savoir plus sur le projet « Centennial », je vous recommande la lecture de l'article de Jonathan : http://goo.gl/jQnDsi

Java et Objective-C

Autres nouveautés qui ont fait énormément parler d'elles à la //Build (et maintenant encore) : les projets « Astoria » et « Islanswood ».
Concrètement, il s'agit de 2 projets Microsoft qui ont pour vocation de permettre aux développeurs de prendre du code Android ou iOS (Objective-C pour le moment), afin de le recompiler sous Windows pour disposer d'une application Windows (Phone) en utilisant la même base

de code (modulo quelques modifications légères) !

Dans le cas du projet Astoria par exemple, l'idée de l'outil est de prendre le code Java et de l'analyser pour voir comment il va être possible de mapper un composant sur sa version Windows. Typiquement, les contrôles graphiques vont être mappés sur leur version Windows, OpenGL va être transformé en DirectX grâce au projet Angle (https://code.google.com/p/angleproject/), et les appels aux APIs Google Play Services vont se voir traduits dans leur équivalent Windows, etc.

Là encore, il sera possible d'utiliser, dans une application Android/iOS, des fonctionnalités spécifiques à la plateforme Windows (telles que les notifications/tuiles par exemple) : il n'y aura pas d'impacts sur les autres plateformes. Cependant, en utilisant la même base de code, vous serez en mesure de disposer de votre application sur l'ensemble des plateformes, avec des fonctionnalités spécifiques si nécessaire.

Côté iOS, l'idée est la même bien sûr, à savoir proposer un compilateur et un runtime Objective-C, des APIs utilisables depuis le code Objective-C, etc. : Fig.B.

Attention, même si ce système peut sembler « magique », il ne faut pas s'attendre à trop de miracles non plus. Certes, le portage d'une application « simple » (ou d'un jeu, comme l'a fait Microsoft en portant le jeu Candy Crush de iOS à Windows Phone) fonctionnera pour ainsi dire directement.

Cependant, il est à noter (et cela a été confirmé par Microsoft) que des applications trop complexes, ou utilisant des briques trop pointues du système, ne pourront pas être portées (à minima pour l'instant, même si rien n'a été annoncé dans ce sens).

Là encore, n'hésitez pas à consulter cet article si vous souhaitez en savoir plus (vous pouvez également me contacter via mon email, blog ou Twitter) : http://goo.gl/lgLJlp

Grâce à ces nouveaux projets, il est possible de voir la direction que prend Microsoft lorsque l'on parle d'universalité : l'idée d'un code unique, tournant directement sur toutes les plateformes, est le rêve de tous les développeurs, et il semblerait que l'on s'en rapproche de plus en plus...

Avec Microsoft, le concept d'universalité n'aura jamais autant pris tout son sens !

Hololens, la réalité augmentée selon Microsoft

Thomas LEBRUN
Consultant
INFINITE SQUARE
tlebrun@infinitesquare.com
http://blogs.infinitesquare.com/b/tom

Ce produit, qui se veut bien plus qu'un simple casque de réalité augmentée, cherche à aller plus loin, à proposer une expérience utilisateur sans précédent, en permettant à celui qui le porte d'insérer des objets (hologrammes) dans la « scène », tout en pouvant interagir avec eux et le décor ! Le périphérique en lui-même est un casque qui se positionne sur la tête de l'utilisateur et qui se fixe au moyen de 2 cerceaux. D'après les premiers retours qui ont été faits par des personnes ayant eu l'opportunité de jouer avec lors de la //Build, le produit n'est pas trop lourd et se porte aussi bien avec des lunettes, que sans, pour pouvoir être utilisé par tout le monde, quelle que soit la configuration.

Sous le capot, pas de surprise : on retrouve un Windows 10 qui va pouvoir recevoir les applications qui seront développées avec Unity. Comme pour le Raspberry Pi, Hololens embarque un mini serveur Web qui permet, grâce à une interface Web, de pouvoir visualiser les packages/applications qui sont installés. C'est également au sein de cette interface que l'on peut charger une nouvelle application sur le casque ! En termes de développement, tout repose sur Windows 10 et Unity. Ainsi, les étapes de développement, racontées par les personnes ayant eu l'occasion de faire « l'Holographic Academy » (un workshop de 4 heures ayant eu lieu pendant l'évènement), se trouvent être très simples :

- Modélisation des objets 3D dans Unity,
- Ajout des behaviors (comportements), triggers et actions au travers de scripts C# ajoutés dans le projet Unity et édités depuis Visual Studio grâce à l'addin Unity.VS,
- Compilation du projet Unity pour le transformer en application Universal,
- Ouverture du projet Universal dans Visual Studio, Ctrl+F5 pour déployer depuis Visual Studio sur les lunettes,
- Déconnexion du câble USB pour pouvoir se balader et tester les scénarios et fonctionnalités possibles.

Au niveau des interactions entre l'utilisateur et le décor, tout a été vu pour les simplifier au maximum, tout en permettant de proposer une expérience utilisateur maximale. Ainsi, la position du curseur est représentée par l'orientation de la tête, la combinaison de sélection est très simple : on positionne le curseur dans le monde en face de soi en bougeant la tête, et on pince les doigts pour sélectionner. La courbe d'apprentissage est beaucoup plus simple que ce qu'elle a pu être sur Kinect (mouvement de main qu'il faut tenir pour Hold), que ce soit pour les utilisateurs mais également pour les développeurs. Il suffit de connaître C# et Unity pour pouvoir développer sa première application et créer ses propres actions/interactions. Si l'apprentissage de C# est déjà chose acquise pour vous ou vos développeurs, il ne reste que Unity qui, au final, s'apprend dans un laps de temps relativement court.

Le mapping dans l'espace, à savoir le fait de gérer l'adhérence d'un hologramme dans le monde réel pour faire en sorte qu'il ne « tombe » pas directement au sol, mais reste sur une table par exemple, est ce qui ressort comme étant l'un des points les plus impressionnants pour les testeurs. En effet, les retours montrent que les lunettes, qui cartographient les alentours en 3D, sont vraiment capables de détecter les bords et, par extension, de gérer les collisions.

Certes, il y a encore de grandes interrogations à l'heure actuelle pour le produit :

- Quel sera son prix ? Sera-t-il abordable au grand public ou est-on sur un produit pour le moment uniquement réservé au monde de l'entreprise?
- Quelle est son autonomie ?
- Il ne semble pas lourd certes, mais est-on sûr de pouvoir le garder sur la tête toute la journée ?
- Est-il possible de le faire fonctionner en extérieur ? Est-ce que le trop plein de lumière peut avoir un impact ?
- Etc.

Comme vous pouvez le voir, il existe encore un grand nombre d'interrogations ! Mais, dans l'absolu, le produit n'est pas encore disponible, donc il y a fort à parier que Microsoft nous réserve, une fois encore, de belles surprises pour la suite !

A noter que Microsoft a annoncé qu'il serait disponible « dans le timeframe Windows 10 » donc, autrement dit, il sera possible de s'en procurer un après la sortie Windows 10, mais personne ne connaît, pour le moment, la date exacte.

Quelques nouveautés autour des performances graphiques

La conférence BUILD a dévoilé de nombreuses évolutions et nouveautés sur la partie graphique et les outils / libraires dédiés. Petit tour du propriétaire à venir !

Sébastien Thevenin
SOAT

DIRECTX

Une des grosses nouveautés est bien entendue Direct3D 12 !

Comme il s'agissait d'un événement Microsoft, je vais commencer par traiter les nouveautés autour de DirectX au sens large, mais on verra dans la suite de l'article qu'il y a aussi beaucoup à dire sur le développement multiplateforme. Si vous avez suivi la deuxième plénière de la BUILD vous avez sûrement vu passer la démonstration sur DirectX12.

Je vous la remets ici, au cas où. Cette démonstration de Square ENIX est plutôt impressionnante (même si elle tourne sur une config que peu de monde possède à la maison).

Lors de cette session, Max McCullen, lead développeur dans l'équipe *Direct3D*, rappelle que DirectX 12 sera disponible uniquement sur Windows 10. Il nous informe que les API sont finalisées. C'est à dire qu'il n'y aura pas de nouvelle feature, ils travaillent maintenant sur des optimisations, bug fix, et prise en charge du maximum de périphériques. A ce sujet, il indique qu'actuellement 50% des joueurs possèdent une carte compatible DirectX12. Ils estiment que, cet été, ils arriveront à couvrir 67% du parc.

Côté support, la plupart des moteurs de jeux connus (il cite Unity 5 et Unreal 4), supportent désormais la dernière version de DirectX. Visual Studio 2015 et notamment son outil de débogage graphique, permet de profiler Direct3D 12.

Côté nouveautés, Max a longuement parlé du rapprochement du matériel en donnant plus de contrôle aux développeurs. Pour rappel, l'année dernière, ils avaient surtout parlé de la réduction de la charge CPU grâce au *Pipeline State Object*, des *Command Bundles*, *Resources Descriptor Heap*, et de la suppression de l'immediate context.

Cette fois-ci, il se concentre sur le MultiAdapter (on comprend mieux les 4 Titan X dans le pc de démonstration de la keynote). En gros, grâce au MultiAdapter, il est possible d'utiliser le potentiel de plusieurs GPU.

Il y a deux modes de MultiAdapter. On pourra utiliser le mode *Implicit* (celui utilisé par la démo de Square ENIX) pour dire au driver que l'on ne veut pas s'en occuper et il le gérera donc tout seul, en s'appuyant sur SLI et CrossFire. Le mode *Explicit,* quant à lui, permet au développeur d'adresser le GPU qu'il souhaite, d'allouer une ressource sur un GPU en particulier (et de les synchroniser si besoin) et de faire exécuter des commandes à un GPU en particulier.

Il y a deux cas dans le mode *Explicit*. Tout d'abord celui des Linked GPU, les cartes graphiques viennent du même vendeur et leurs spécifications sont proches, alors on ne verra qu'un seul GPU avec différentes zones mémoires et plusieurs processeurs de commandes. Max ne s'est pas attardé sur cette partie. Il a surtout parlé de l'autre cas, les Unlinked GPU.

Dans ce cas, les cartes graphiques ne sont pas forcément du même vendeur et de la même puissance. On peut, par exemple, imaginer de calculer la scène 3D sur la carte graphique dédiée et de calculer l'interface utilisateur à la carte graphique intégrée à la carte mère. Ou encore de laisser le GPU intégré s'occuper des tâches de post processing.

LES AUTRES ÉLÉMENTS

- DirectWrite : DirectWrite s'occupe de l'affichage et de la manipulation graphique du texte. Dans Windows 10, il y a désormais deux types de polices de caractères. Les « recommandées », qui seront présentes sur tous les périphériques tournant sous Windows 10, et les « optionelles », qui seront téléchargées à la demande d'une application.
- *Direct2D apporte aussi de nombreuses nouveautés telles que le chargement des images YCbCr simplifié, des nouveaux effets intégrés et l'optimisation des effets multiples grâce au Shader Linking*"
- Présentation du projet Angle : ce projet a été initié par Google en 2010. Le but du projet était d'apporter la compatibilité avec OpenGL pour les applications Win32. Il est notamment utilisé dans Chromium, Google Chrome, Firefox, principalement pour le support de WebGL. Sa particularité est de transformer les ordres OpenGL en Direct3D. Ce choix a été fait car, aux niveaux des drivers, Direct3D est beaucoup plus supporté. Microsoft a ensuite rejoint le projet en 2013. Leur arrivée a permis de rajouter le support de Windows Phone 8.1 ainsi que des applications Windows Store, et également de diminuer les prérequis matériels demandés.

La partie native C++ n'a pas été oubliée avec Visual C++ Cross-Platform Mobile. Il permet de créer des jeux utilisant OpenGL pour Android et iOS et avec un large partage du code. Et il sera possible combiner tout ceci avec le projet Angle.

Retour sur Devoxx

1ère partie

Pour cette nouvelle édition, Devoxx France s'est déplacé au Palais du Congrès, Porte Maillot, à Paris. Cette année tout a été plus : la superficie, le nombre de visiteurs, les partenaires. L'agenda a été comme chaque année très chargé avec de nombreux thèmes : cloud, mobile, futur du développeur, IoT, Java, JavaScript, frameworks divers et variés.

Nous vous proposons de faire un retour Devoxx avec une sélection d'une dizaine de sessions, sur tous les sujets. Comme toujours, Devoxx est frustrant car il y a jusqu'à 8 sessions en parallèles. Mais heureusement, les vidéos seront disponibles en ligne et de nombreux slides sont dores et déjà visibles sur des sites comme slideshare.

Enjoy !

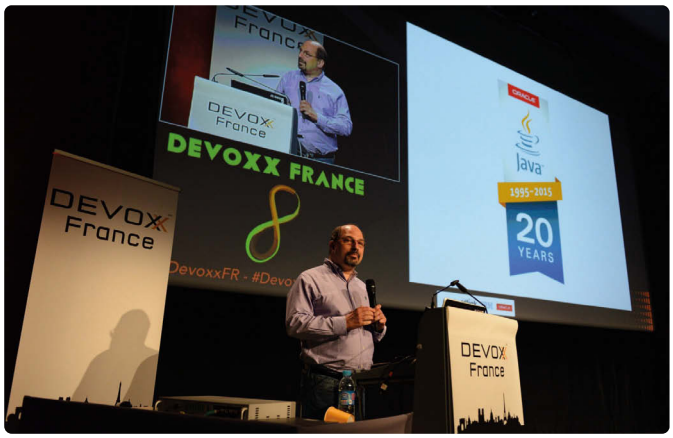
La rédaction.

Imaginons notre métier de développeur dans 20 ans

Début Avril se déroulait la 4ème édition de Devoxx France. Un évènement avec un goût de nouveauté car il se déroulait pour la première fois à la Cité des Congrès de Paris. Un cadre prestigieux permettant d'accueillir encore plus de développeurs et de conférenciers. Imaginez : 1800 participants, 200 speakers, et plus de 220 talks ! Tout cela, présenté sur trois jours de conférences.

Florent Dupont
(fdupont@sqli.com)
Expert Mobility & Internet of Things, Digital Industrialization chez SQLi
http://florent-dupont.blogspot.fr

Jean-François Garreau
(jfgarreau@sqli.com)
Responsable IOT & Senior Innovation Developper chez SQLi | Organisateur GDG Nantes
http://jef.binomed.fr



Cette année, les organisateurs ont souhaité nous faire réfléchir au métier de développeur informatique dans 20 ans. Chez SQLi, nous avons déniché une vieille carcasse de DeLorean, confectionné un convecteur spatio-temporel avec quelques Arduino (oui, oui, on peut, c'est écrit au fin fond du guide de référence), soudé quelques condensateurs : et hop ! Nous avons construit une machine à voyager dans le temps et avons été projetés jusqu'en 2035. Nous vous faisons un petit retour de ce que nous avons vu... à la sauce Future<Devoxx> !

Devoxx me up, Scotty(*)

Début 2000, Devoxx, était initialement une conférence basée à Anvers. Elle s'est ensuite ouverte à Paris au sein d'un Devoxx France en 2012, suivi de Londres en 2014, puis d'un Devoxx Pologne en 2015. Une nouveauté importante en 2015 : l'ouverture de Devoxx au continent Africain avec un nouveau pays Devoxxien : Devoxx Marocco.
En 2035, toutes les grandes villes européennes possèdent leur Devoxx. Après l'ouverture au continent Africain, c'est l'Asie qui s'y met rapidement et bientôt le concept Devoxx s'étend au monde entier. Plus d'infos sur cette conférence sur www.devoxx.ma/fr

(*) Scottie, télé-devoxx-portation !

Prod? Where we're going we don't need Prod !(*)

En 2035, dans la plupart des entreprises, les serveurs ne seront plus des ressources gérées directement en interne. La *Production* - telle qu'on la connaît en 2015 – n'est plus. Les serveurs sont entièrement externalisés vers des services de Cloud *(en 2015, on voit d'ailleurs de plus en plus d'acteurs présents sur le Cloud, Microsoft Azure qui étend son offre, IBM Bluemix*). Chaque exécution d'application se traduit par une demande "CPU/mémoire/stockage/bande passante" définie en amont par les développeurs. Ceux-ci ne se soucient donc plus de savoir où sont publiées physiquement leurs applications mais seulement de ce dont elles ont besoin.
Apache Mesos qui permet de distribuer les applications sur différents modes d'exécutions, selon les ressources disponibles sur un ensemble de serveurs. Les besoins des applications peuvent évoluer de manière élastique et les nouvelles ressources disponibles sont prises en compte en temps réel. Les serveurs sont dorénavant des ressources d'entreprise, payées à la demande,

au même titre que l'électricité ou le chauffage...
Quelques talks associés :

- "The end of server management : hosting have to become a commodity", par Q. Adam
- "Mesos University", A. Maury et al.
- "Bluemix, l'avenir du développement dans le Cloud", par J. Gauci
- "L'écosystème Docker sur Azure", par P. Chanezon.
- Scaling Docker with Kubernetes, par Carlos Sanchez

(*) La Prod? Là où on va, on n'a pas besoin de Prod

Everybody remembers where we (s)parked?(*)

Depuis longtemps, les frameworks et API aident à tirer parti de ces ressources, qu'elles soient distribuées sur plusieurs serveurs ou sur plusieurs cœurs. Les langages eux-mêmes sont pensés pour faciliter la distribution des calculs. Ils marient l'approche OOP à la démarche fonctionnelle et proposent nativement des instructions de parallélisation facilitant ainsi le développement d'applications réactives.
La généralisation des conteneurs (Docker menant toujours la danse), fait perdre l'intérêt des langages à compilation intermédiaire. Java, .NET, Scala, Groovy sont morts depuis longtemps en 2035... Tous les développeurs sont revenus sur des langages compilant en natif pour lesquels on compile pour une seule plateforme puisqu'elle sera conteneurisée. Tout le monde y gagne : les développeurs programment et compilent plus rapidement, les utilisateurs profitent d'exécutions plus rapides. En 2035, ces types de langage (dont *Go* et *Rust* ont été les précurseurs) sont couramment mis en place pour les grosses applications distribuées en production.
Quelques talks associés :

- Initiation à Spark avec Java 8 et Scala, par H. Saleh & O. Girardot
- Go Tutorial, par F.Campoy Flores
- Refactoring to functional, par Hadi Hariri
- Présentation de Rust, par M. Poumeyrol et P. Baillet
- API Asynchrones en Java 8, par J. Paumard

(*) quelqu'un sait où on spark ?

Sometimes the scariest things come from within...(*)

En 2035, JavaScript a disparu depuis quelques années... Mais il ne faut pas oublier qu'il a été LE langage de prédilection pendant un grande partie des années 2010 pour le développement Frontend... mais pas que. Il a été également présent côté serveur. Cette particularité assure ainsi au langage une capacité isomorphique. Pour aider cet isomorphisme, Browserify, a

permis de modulariser le code de manière cohérente entre la partie cliente et serveur.
L'année 2015 a vu le Web faire un grand pas en avant avec notamment l'introduction des Web Components (et tout particulièrement son implémentation Google : PolymerJS) et des Web Workers. Nos développements Web s'en sont trouvés facilités et toujours plus robustes.
Côté langage, en attendant la norme ES6, c'est TypeScript – une surcouche JS – qui se met en place, notamment dans un Framework très à la mode à l'époque : Angular 2. TypeScript offre l'avantage d'être typé, améliore la brièveté du code et sa lisibilité. C'est à cette époque que le JavaScript est finalement devenu l'assembleur du Web laissant ainsi la place à des langages typés, qui sont dorénavant pris en compte par les navigateurs.
Quelques talks associés :

- Se préparer dès maintenant à l'arrivée d'Angular 2, par R. Linsolas
- "Modulariser son application AngularJS avec Browserify", par A. Richard
- « Typescript, le javascript statiquement typé », par B. Lemoine
- "WebComponents, Polymer and Material Design" par H. Gonzalez

(*) Parfois, les choses les plus effrayantes viennent de l'intérieur

Where did he learn how to negotiate (TCP) like that?

En 2015, les premières implémentations de http/2 voient le jour dans les navigateurs Web modernes et ce, même si la spécification ne sera finalisée que quelques années plus tard ! HTTP2 est orienté stream, permettant entre autres, de faire enfin des appels HTTP en parallèle, nativement sécurisés. Les développeurs prennent également conscience qu'il est temps de suivre l'adage « le bon outil pour le bon usage ». Http est omniprésent mais pas le plus adapté, notamment pour les communications des objets connectés. En 2035, les nouveaux protocoles sont partout.
L'avènement des objets connectés en 2015 a forcé les entreprises à se tourner vers de nouveaux protocoles plus modulaires et plus réactifs que l'omniprésent Http. Les objets connectés s'appuient dorénavant sur des protocoles construits directement au-dessus de TCP ou UDP. C'est notamment le cas du protocole MQTT toujours aussi efficace pour faire du publish / subscribe.
Quelques talks associés :

- "HTTP/2: A deux c'est mieux! " par JF. Arcand
- "Comprendre l'IOT grâce à une boule... mais pas que", par L. Huet & P. Charrière
- De l'API au protocole, par G. Couprie

(*) Où est-ce qu'il a appris à négocier (TCP) comme ça?

Some (entreprise) things never change... some things do(*)

Malgré les nombreux ouvrages démontrant les mauvaises pratiques de management de projet - notamment décrites dans le livre de F. Brooks « The Mythical man months » datant de 1975 (sic !) - certaines idées reçues ont la vie dure ! En 2035 (comme en 2015 d'ailleurs), beaucoup de sociétés tombent encore dans le panneau des idées reçues de l'informatique : ajouter des développeurs en plus en fin de projet pour limiter le retard, faire plusieurs projets en parallèle, chasser les très gros projets informatiques...
Quelques talks associés :

- "Les idées reçues de l'informatique : quand nos erreurs de raisonnement nous limitent", par L. Cinquin.

(*) Certaines choses (d'entreprise) changent, et d'autres ne changeront jamais

Can a robot turn a canvas into a beautiful masterpiece?(*)

En 2035, le monde est fortement robotisé. Après l'utilisation massive de la robotique pour l'industrie, les robots commencent à faire leur apparition dès les années 2020 pour apporter du service à la personne. Aider les personnes en situation de handicap moteur afin de faciliter leur déplacement, les assister ; mais également les personnes âgées souffrant de pertes de mémoire, et pour lesquelles, les robots vont aider à maintenir une socialisation.
Les robots sont également présents en tant qu'intelligence, pour aider tous les médecins à diagnostiquer certaines maladies en s'appuyant sur des intelligences partagées dont Watson est précurseur.
Les robots apprennent d'abord isolément à identifier leurs environnements. Puis collectivement, en transmettant leurs nombreuses données d'apprentissages qui sont stockées en masse, puis analysées par des algorithmes de Machine Learning, afin d'améliorer les prédictions et donc, en retour, l'intelligence de chaque robot. Mais attention, toutes ces données collectées sur les propriétaires des robots sont-elles utilisées à bon escient ?
Quelques talks associés :

- "Le futur de la robotique personnelle", par R. Gelin
- "Un robot peut-il apprendre comme un enfant ?" par PY. Oudeyer
- "Créer des applications cognitives avec IBM Watson Engagement Explorer et Bluemix", par P. Comte
- La problématique du contrôle des technologies de l'information, par E. Fillol

(*) Est-ce qu'un robot sait transformer un bout de toile en magnifique tableau de maître ?

Always in motion the future is(*)

Nous sommes de retour dans le présent. Devoxx 2015 nous aura donné un avant-goût de ce que l'avenir nous réserve et nous aura permis d'anticiper notre futur de métier de développeur et plus généralement d'informaticien.
Cet article fait référence à une sélection des conférences que nous avons pu voir en tant que participants et que nous invitons à voir, ainsi que de nombreuses autres, toutes aussi intéressantes, sur le site officiel de Devoxx http://www.devoxx.fr, ainsi que prochainement sous forme de vidéo sur le site de Parleys : www.parleys.com.
Et comme disait le Doc' « Le futur n'est jamais écrit à l'avance pour personne ; votre futur sera exactement ce que vous en ferez, alors faites qu'il soit beau pour chacun de vous. »

(*) Toujours en mouvement le futur est

FIREBASE : REALTIME APPS

Emilien MURATON
Développeur JAVA chez
PALO IT

En préparant ma venue au Devoxx 2015, j'ai remarqué le Tools-in-Action « Jeu de rôle en ligne massivement multijoueur avec Firebase », animé par Alexis Moussine-Pouchkine et Thomas Guerin, qui semblait prometteur. D'une part grâce à la technologie abordée, Firebase, la start-up qui a été rachetée fin 2014 par Google pour consolider Google Cloud Platform, d'autre part par la promesse faite dans la description du Tools-in-Action de pouvoir créer un jeu multi-utilisateurs en seulement 30 minutes ! Les speakers ont présenté Firebase par le biais d'un jeu de prise de contrôle de gare qu'ils ont développé. Ce talk m'a donné un aperçu des capacités impressionnantes de Firebase : je vous propose de vous partager les recherches que j'ai effectuées afin de me rendre compte de la réelle valeur ajoutée de Firebase...

Firebase, c'est quoi ?

Firebase est une plateforme de développement d'applications en temps réel. Elle utilise un modèle BaaS (Back end as a Service). En d'autres termes, Firebase propose aux développeurs de ne coder que du code front-end et de laisser Firebase s'occuper du reste. Les données sont stockées en NoSQL sous la forme d'un fichier JSON au sein de SoftLayer d'IBM. Pour mieux comprendre, voici un exemple d'une BD Firebase : https://dinosaur-facts.firebaseio.com/.
Firebase s'appuie sur 3 piliers fondamentaux.

1. Une API pour manipuler, stocker et synchroniser les données en temps réel

La synchronisation des données est tout simplement bluffante. Les modifications effectuées depuis un client ou dans la BD Firebase sont presque instantanément mises à jour. Ainsi, on comprend facilement le surnom de Firebase comme étant la « Dropbox des développeurs ». OK, mais que se passe-t-il si le client perd sa connexion Internet ? Firebase conserve une version locale des données actives. Quand une donnée est modifiée, elle est mise à jour en local puis envoyée aux serveurs Firebase. Par conséquent, les applications Firebase restent relativement fonctionnelles en mode offline et synchronisent les données dès la reconnexion (par elles-mêmes, donc pas de code à écrire).
Un autre point très intéressant de Firebase : des librairies officielles, qui sont pour la plupart des frameworks / librairies Javascript et des librairies natives pour iOS et Android :

- AngularFire,
- EmberFire,
- ReactFire,
- BackboneFire,
- NodeJS,
- iOS SDK,
- Android SDK.

Et si votre plateforme n'est pas supportée, une API REST permet tout de même d'intégrer Firebase (de nombreuses librairies non officielles ont été créés par ce moyen et sont disponibles sur Git).

2. Un service d'authentification (sans toucher à une seule ligne de code côté back)

Firebase fournit un service d'authentification clé en main. L'API nous offre 4 principaux modèles d'authentification :

- Une authentification anonyme,
- Un couple d'email & password,
- Différents OAuth (Facebook, Twitter, GitHub, etc.),
- Ou une authentification personnalisée via JSON Web Tokens.

Firebase utilise des callbacks pour renvoyer les informations de l'utilisateur et son UID. De plus, Firebase conserve un payload de la connexion contenant le couple : mode de connexion (anonyme, Facebook, etc. et UID de l'utilisateur.

3. Sécurité des données

Le protocole SSL est utilisé par défaut pour transporter les informations, ainsi les ressources REST sont accessibles via HTTPS. Pour la gestion des droits de lecture/écriture, Firebase fournit des « règles de sécurité ». En synthétisant, ces règles correspondent à des mots clés à placer au sein de la structure JSON (donc du côté de la BD). Ils permettent de décrire et de limiter l'accès à nos données. Il existe quatre types de règles : read, write, validate et indexOn. Les deux premières servent à octroyer des droits de lecture et écriture aux données. La règle validate est utilisée comme une contrainte sur la donnée. IndexOn permet d'accélérer la vitesse des requêtes en précisant les clés à indexer. NB : les règles de type read and write appliquées à un nœud sont aussi appliquées à tous ses nœuds enfants. Ainsi, si un nœud parent a la permission read, tous ses noeuds enfants ont aussi obligatoirement cette même permission.

4. Pas de SQL : comment rechercher mes données ?

Les requêtes peuvent à elles seules constituer un article. De ce fait, nous n'allons voir qu'un aspect schématisé de cette notion, puis je vous proposerai quelques liens à visiter pour obtenir plus d'informations. Les données peuvent être recherchées selon leur emplacement dans l'arborescence JSON ou selon la priorité des nœuds JSON. Une priorité peut être un nombre ou string que l'on lie à un nœud du fichier JSON (par défaut les nœuds n'ont aucune priorité). Lors d'une requête sur les priorités, les règles suivantes s'appliquent :

- Les nœuds sans priorités s'affichent en premier,
- Les priorités de type nombre s'affichent en deuxième,
- Les priorités de type string s'affichent en dernier,
- Quand deux nœuds ont la même priorité, les nœuds sont triés selon leurs noms.

Dans la plupart des cas, on utilise l'emplacement du nœud comme élément de recherche : il faut donc faire très attention à la structure du JSON stockée dans Firebase.

Pour plus d'informations :

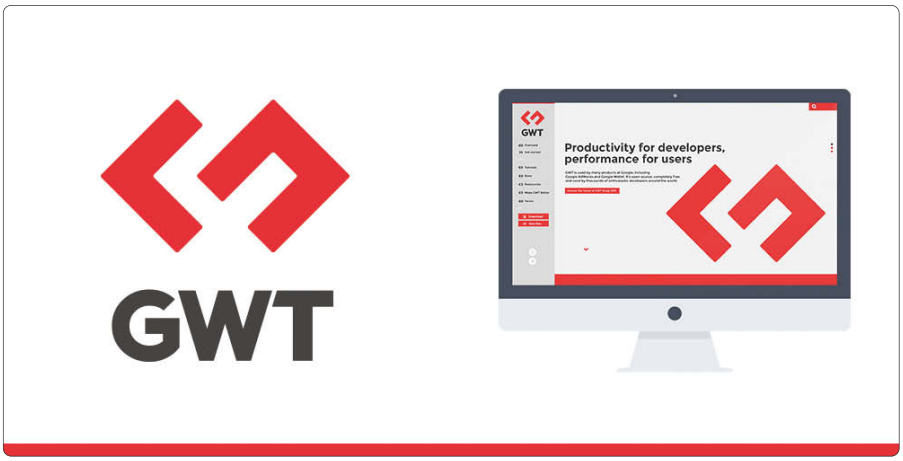
- Structuring data : https://www.firebase.com/docs/web/guide/structuring-data.html
- Priority : https://www.firebase.com/docs/web/guide/structuring-data.html
- Queries : https://goo.gl/5GQfOa
- Retriving data : https://www.firebase.com/docs/web/guide/retrieving-data.html

Prêt pour l'aventure Firebase ?

Vous pouvez dès à présent vous lancer dans l'aventure Firebase en vous inscrivant ici : https://www.firebase.com/login/.
Pensez également à consulter le pricing ici : https://www.firebase.com/pricing.html.
Pour plus de détails sur la conférence Devoxx : http://goo.gl/NakDfQ

« Le retour en force de GWT »

Guillaume PARAMELLE
Développeur Java GWT chez **PALO IT** *Toulouse*



Devoxx est le rendez-vous des nouvelles technos... et de celles qui durent ! Adepte de la première heure et convaincu de l'approche proposée par GWT, je me devais d'assister à cette session présentant les dernières évolutions du SDK et animée par Sami Jaber.

GWT est bel et bien là !

C'est la première chose soulignée lors de cette présentation. Malgré les rumeurs d'abandon du projet depuis que des solutions alternatives ont été proposées par Google comme Dart ou Angular, la communauté est active (les versions 2.8 et 3.0 sont en cours de développement) et c'est bel et bien GWT que Google a choisi pour développer son tout dernier client mail Google Inbox.

Java 8

Une des principales nouveautés attendues dans la prochaine release sera bien entendu le support de Java 8. En effet, les utilisations des lambda expressions prendront tout leur sens dans la gestion des évènements UI et des appels serveurs qui, de façon inhérente au JavaScript, sont asynchrones. Les projets GWT bénéficieront ainsi de cette nouvelle syntaxe plus concise :

```
button.addClickHandler((clickEvent) -> {
   dialogBox.setText("You click me !");
   dialogBox.show();
}) ;
```

JSInterop

L'objectif de JSInterop est, comme son nom le sous-entend, de faciliter l'interopérabilité Java/Javascript au sein d'une application GWT. Concrètement, il s'agit de faire correspondre, via un jeu d'annotations, une interface Java avec une implémentation en Javascript. Cette solution, plus intuitive que JSNI, devrait permettre de faciliter encore un peu plus l'intégration des librairies JS dans l'écosystème GWT.

Compilation et débogage

De gros efforts ont été fournis pour améliorer l'expérience de développement. La compilation incrémentale doit permettre de réduire les temps de compilation et de prendre en compte les modifications de code instantanément au sein du navigateur. Alliée au Super Dev Mode et Source Map, il sera ainsi possible de déboguer le code Java directement au sein du navigateur.

Singular

Initialement présenté lors de la dernière conférence GWT.Create, Singular est un framework qui doit apporter à GWT les principaux concepts qui ont fait le succès d'Angular : structuration du code, MVC, templates, double data-binding, injection de dépendances, approche déclarative des composants, etc. Ajoutez à ces concepts la philosophie d'un langage typé comme Java et d'un compilateur optimisé comme GWT, et on gagne les avantages suivants :

- La compilation habituelle du code Java (Model et Controller), mais également des templates HTML (View) permettent d'identifier au plus tôt les bugs éventuels et de bénéficier de l'auto-complétion de l'IDE,
- L'optimisation du code Javascript généré, au-delà de ce qui pourrait être fait manuellement,
- Le code Javascript produit est propre à la plateforme cible et n'embarque que ce qui lui est nécessaire.

C'est sur ce dernier point que Singular enfonce le clou. Il ne s'agit plus simplement de produire une application Web adressant tous les navigateurs du marché, mais désormais de proposer une solution pour créer des applications natives, avec pour principales cibles iOS et Android.
En effet, c'est tout naturellement que le couplage trop lâche entre la vue et le contrôleur ont amené les Développeurs du framework à pousser le concept jusqu'au bout : écrire d'un côté un code unique pour la partie contrôleur et de l'autre des templates propres à chaque plateforme cible (UiBuilder pour iOS, Layout XML pour Android, HTML pour le Web). Pour être tout à fait complet, Singular apportera avec lui son lot d'APIs permettant d'accéder aux fonctionnalités natives du terminal (géolocalisation, caméra, etc.).
Grâce à cette dernière nouveauté, il se pourrait bien que GWT vienne appuyer le slogan original de Java: « Develop once, run everywhere ».

Conclusion

Cet article reprend l'un des concepts qui m'ont paru les plus prometteurs annoncés lors de la conférence de Sami Jaber.
N'oublions pas que GWT, avec son concept original (et qui le reste 10 ans après !) est un incubateur d'idées, où les meilleures d'entre elles se voient pérennisées lors des releases suivantes.
Si vous êtes intéressé, je vous propose de regarder du côté de GSS (Less/Sass like), Elemental (HTML5 features) et de divers travaux qui émergent autour des Web Components, Polymer et de GWT.

DEVOXX 2015 : quickie « Asynchronismes en JavaScript »

Patrick DOS SANTOS
Architecte Java chez
PALO IT

Après avoir assisté au Devoxx, j'ai eu envie de rédiger un retour sur la conférence « Asynchronismes en JavaScript » présentée par Florent Le Gall. Pour comprendre pourquoi j'ai sélectionné cette présentation, il faut d'abord comprendre mon parcours : je vais donc me présenter.

Les limites de JavaScript

Je m'appelle Patrick Dos Santos et je suis Développeur/Architecte depuis de nombreuses années. J'ai essentiellement évolué dans le monde de JAVA et JEE. Je ne pense pas être le seul à avoir constaté à ses débuts les limites de l'utilisation de JavaScript :

- Le langage étant non compilé, il fallait déployer les scripts et les tester pour se rendre compte qu'il manquait un ";" ;
- A l'époque, il fallait reconstruire le WAR entier pour le déposer sur notre serveur local et attendre son déploiement. Désormais, ce n'est heureusement plus le cas : les différents serveurs gèrent le déploiement à chaud de nos applications WEB.
- Le code était téléchargé sur le poste client, ce qui posait quelques problèmes de sécurité ;
- Il était impossible de tester nos scripts et de vérifier que nous n'avions pas créé de régression sur notre code, ce qui nous obligeait du coup, à faire une recette de non régression à chaque fois que du code JavaScript était écrit.

Récemment, j'ai participé à une mission où nous n'avons utilisé que du HTML5 et JavaScript dans le but de créer une application PhoneGap, ce qui m'a fait découvrir NodeJS et son écosystème (task runner, jshint, etc.). Depuis, mon intérêt pour JavaScript s'est accru et c'est dans cette optique que je me suis rendu à la présentation de l'asynchronisme en JavaScript.

Asynchronismes en JavaScript

Florent Le Gall avait opté pour une présentation au format "Quickie" d'une durée de 20 minutes, pendant la pause déjeuner. Malgré le peu de temps imparti pour rentrer dans les détails, j'ai trouvé que le speaker a réussi l'exploit de nous faire une mini-formation sur les promesses de JavaScript.
En nous basant sur des méthodes "classiques" de gestion de l'asynchronisme (appel de méthodes callback), nous avons pu voir les difficultés lorsque celles-ci s'enchaînent. Effectivement, lorsqu'on appelle des méthodes asynchrones en cascade, le code devient de moins en moins lisible au fur et à mesure des enchaînements. Du coup, les Développeurs ont logiquement opté pour la gestion de promesses (en l'occurrence avec le framework Q) et c'est à ce moment que j'ai constaté la remontée fulgurante du JavaScript. Grâce à ces promesses, nous pouvons facilement écrire du code lisible indiquant le comportement à adopter sur un ensemble d'événements, voir chaîner nos promesses pour les rendre plus claires. Dans le monde du JAVA, une telle fonctionnalité ne s'est vue qu'en Java 8. A mon grand étonnement, JavaScript a ainsi rattrapé son retard assez rapidement.

Un peu de Java EE

Durant 50 minutes, nous avons eu droit à un tour de propriétaire du futur Java EE autour de 3 axes forts : HTML 5, Cloud, simplification du développement. La plate-forme ne sera pas disponible avant 18 mois (si tout va bien) et de nombreuses spécifications ne sont pas encore lancées. Pour les équipes, le support de HTML 5 est un élément important pour faciliter le déploiement d'applications web et suivre les évolutions. Un important travail sera fait pour supporter les dernières versions évolutions de JSON. Mais il est aussi intéressant de voir les discutions et les problèmes d'orientations pour certaines fonctions. Par exemple, MVC n'a pas été clairement tranché pour le moment, faut-il une alternative à JSF ou au contraire en être complémentaire ? Pour les templates, lesquels supporter (facelets, jsp, spi, les trois) sans en créer de nouveaux ? Idem pour la partie contrôleur : reprendre une technologie existante ou en créer une nouvelle ? D'autre part, il est aussi intéressant de comprendre comment une nouvelle spécification, un nouveau standard se retrouvent dans le monde Java. Http2 sera bien entendu supporté, mais une des questions était de savoir où ce support se ferait. Le choix a été fait dans le futur servlet 4.0... Du côté des EJB, une bataille se fait jour : va-t-il être remplacé à terme par JMS ? Possible ou pas... Occasion aussi de faire le point sur CDI 2 (CDI = Contexts and Dependency Injection)...

La rédaction

JavaScript, les bases du langage

J'ai pu assister à la conférence de M. Thierry Chatel intitulée « Comprendre enfin JavaScript ». Comme beaucoup de personnes, je suis un développeur qui ai commencé à utiliser le JS à travers JQuery qui, grâce à ses différents plugins, facilite les interactions utilisateur sur les pages Web.

Makhtar Diagne
Ingénieur Etudes et Développement JAVA
NETAPSYS ingénierie informatique
Source : http://goo.gl/u3uDel

JS est un langage orienté objet mais sans classes comme Java, non typé et fonctionnel. Cette conférence avait pour but de rappeler les fondements de JS, de mettre des noms sur des principes que nous utilisons tous les jours, et donner des Best Practices mises en place dans les Framework qui composent nos applications.

Mode strict : « use strict »

En tout début de fichier ou au début d'une fonction, cette instruction permet d'activer un mode rigoureux qui signale plus d'erreurs (fail-fast), comme des variables non déclarées, ne transforme pas this en objet (window n'est pas utilisé à la place de undefined comme objet global), et impose certaines règles comme l'unicité des propriétés dans un objet littéral, ou des noms d'arguments dans une fonction.

Var

Le mot clé var est utilisé pour déclarer une variable non typée.

```
var myVariable ;
```

Point-virgule (;)

Le point-virgule (;) marque la fin d'une instruction. Il peut être omis mais c'est déconseillé car JS ajoute automatiquement les points-virgules manquants et cela peut provoquer des situations cocasses.

```
function millenium() {
return
[
1000,
"M"
];
}
```

Que renvoie cette fonction ?
Undefined est renvoyée car un « ; » est ajouté juste après le return. Il est également ajouté automatiquement :

- En fin de ligne, de fichier, ou avant une accolade fermante (}),
- Seulement si le token suivant crée une erreur de syntaxe,
- Jamais dans l'entête d'une boucle for,
- Jamais quand le point-virgule crée une instruction vide,
- Insertion automatique là où une fin de ligne est interdite (une Postfix Expression ++/−, un continue/break, un return, un throw),
- L'insertion automatique des « ; » manquants ne peut être désactivée.

Scope

Une variable est déclarée pour toute la fonction qui la contient et elle existe même avant la ligne où se trouve le mot clé var.
Une variable déclarée hors de toute fonction devient une propriété du scope global. (window est le scope global dans les navigateurs Web).

Objets

Les objets sont des maps clés/valeurs. La clé est une chaîne de caractères correspondant au nom de la propriété et la valeur est quelconque non typée. Toutes les propriétés sont publiques. La valeur d'une propriété peut être accédée avec obj['cle'] ou obj.cle. Un objet peut être créé avec des accolades et les objets peuvent être imbriqués.

```
var objet = {
key1 : 3,
key2 : « test »,
test1 : function () {
return key2;
}
} ;
```

Ajout ou modification d'une propriété : obj.key3 = 'value3' ou obj[key3] = 'value3'
Suppression d'une propriété : delete obj.key3 ou delete obj[key3]
La recherche d'une propriété se fait d'abord dans l'objet courant, puis dans son prototype, puis dans le prototype de son prototype, puis dans le prototype du prototype de son prototype ... L'affectation d'une valeur à une propriété modifie ou crée la propriété toujours dans l'objet courant, même si c'est une propriété héritée d'un prototype.
Objetc.prototype est la racine des chaînes de prototypes.

Fonctions

Déclaration : function en début d'instruction, le nom de la fonction est obligatoire et aucune déclaration de valeur. La fonction est définie dès le début de la fonction qui la contient sinon dès de le début du fichier.

```
function aFunction (arg1, arg2) { // Déclaration
return function bFunction (arg1, arg2) { // Expression
…
} ;
}
```

Expression : le mot-clé function dans une expression n'est pas en début d'instruction, le nom est facultatif et la valeur retournée est la fonction. La fonction est définie seulement après l'évaluation des expressions.
Les fonctions sont des objets JS. Ils peuvent être mis dans une variable, dans la propriété d'un objet, dans un tableau. Ils peuvent renvoyer des fonctions ou passer comme paramètres d'une fonction.
aFunction.call(this, arg1, arg2, arg3); // appel de la function en fournissant comme paramètre this et les paramètres un par un.
aFunction.apply(this, [arg1, arg2, arg3]); // appel de la function en fournissant comme paramètre this et un tableau de paramètres.
Les fonctions call et apply sont utilisées pour des appels récursifs. Par exemple, pour parcourir différents éléments d'un élément et effectuer un traitement.

Closure

Toute fonction JS a accès aux données du scope dans lequel il est défini.

```
buildHome: function(container,query) {
var _this = this;
$('#login-box form').submit(function(event){
_this.fillValues();
}
}
```

_this est accessible dans la fonction interne car c'est une variable du scope dans lequel la function est définie. La fonction garde une référence vers son scope de définition et les données sont celles présentes au moment de l'exécution de la fonction. Cela peut provoquer des fuites de mémoire car JS purge le scope à la sortie de la fonction parente.

Anonymous Wrapper

Les fonctions anonymes sont exécutées immédiatement. Les arguments ne sont pas obligatoires.

```
(function(container,query) {
//…
}($(« #id »), « /login »));
```

OU

```
!function(container,query) {
//…
}($(« #id »), « /login »);
```

Ce sont des expressions de fonctions au lieu de déclarations, d'où l'utilisation de parenthèses. Cela permet de créer un scope contenant les variables et les fonctions. L'usage est d'encapsuler le contenu de chaque fichier JS dans un anonymous wrapper.

```
(function() {
//…
}) ();
```

Pattern Module

C'est une anonymous factory appelée immédiatement, permettant de créer et de renvoyer un objet avec des propriétés et des méthodes publiques, et qui a accès aux variables locales (données et fonctions privées) du module.

```
var module = (function() {
var privateVarA = « A »;
var privateMethod = function() {
// …
};
return {
moduleProperty : « B »,
get : function() {
// …
privateVarA = « B »;
privateMethod();
}
};
}());
```

Opérateurs

obj && obj.fn permettent d'accéder à une propriété si et seulement si l'objet est défini. Si obj est équivalent à true, renvoie obj.fn sinon renvoie obj.
arg1 || {} permet d'utiliser une valeur par défaut. Si arg1 est équivalent à true, renvoie arg1 sinon {}.

Exceptions

Un seul catch utilisé pour tout type d'exception. Le bloc finally est facultatif.

```
try {
// …
} catch(e) {
// …
} finally {
// …
}
```

Une exception peut être déclenchée avec throw « Erreur » ou throw new Error(message), Error étant un objet avec une méthode toString().

Patterns

Duck typing :

Les prototypes sont utilisés pour partager du code. Cela correspond à une fonction générique effectuant un traitement sans se baser sur le type.

Mixins :

```
function aFunction () {
myFramework.extend(arguments, {
map : Array.prototype.map
});
// return
}
```

Configuration object :

C'est utilisé pour passer un objet en argument avec un code beaucoup plus maintenable et lisible.

```
$myAjaxCall({
method : 'GET',
url : '/call'
}) ;
```

Au lieu de $myAjaxCall('GET', '/call') ;

Promesses :

Utilisé pour gérer l'asynchronisme, l'objet retourné correspond à un résultat asynchrone. Il ne contient jamais le résultat avec une méthode then pour enregistrer un callback de succès ou un callback d'erreur. On appelle then, que la méthode soit en attente ou déjà résolue.

```
this.fetch({
success : function(collection, response) {
//
},
error : function(collection, response) {
//
},
data : {
logout : »
}
});
```

Types

Il y a 5 types primitifs: boolean, number, string, null, undefined

- Undefined si aucune valeur n'est affectée,
- Null utilisée explicitement pour indiquer l'absence d'objet. Le type numérique est sur 8 octets et il n'y a pas de types caractères,
- Les primitifs : passage par valeur avec une copie de la valeur,
- Les objets : passage par référence,
- L'autoboxing est utilisée avec une conversion en Objets de types String, Boolean, Number. Ces types ne doivent pas être utilisés explicitement.

Comparaisons

Il existe deux types de comparaison :

- Comparaison stricte sans conversion automatique : « true » pour des données primitives du même type et même valeur, pour des références au même objet et « false » dans les autres cas.
- Comparaison avec conversion automatique de type : pour deux objets, une comparaison de références est réalisée. S'il y a un opérande primitif alors null == undefined si un opérande number ou un boolean, tout est converti en number, si un opérande string, tout est converti en string.

Tableaux

Les tableaux sont des objets initialisés avec des crochets ([]). Comme tous les objets, ils ont des propriétés dont les noms sont des chaînes de caractères.
Ex : var emptyArray = [] ; var myArray = ['java', 'script'];
myArray [1]; => script
La propriété length permet d'avoir la taille du tableau. Elle est augmentée lorsqu'on ajoute un index trop grand, et, lors de la modification de length, les index trop grands sont supprimés.

En conclusion, le Devoxx France est une occasion idéale pour revoir quelques fondamentaux comme j'ai pu le faire avec le Javascript, de découvrir un sujet qui vous intéresse à l'occasion d'un TP, et de voir la mise en œuvre d'un outil que vous pouvez utiliser dans vos projets.

Datomic, la base de données qui n'oublie rien

C'est au cours de la conférence « Datomic, la base de données qui n'oublie rien » présentée par Hiram MADELAINE sur Devoxx France 2015 que j'ai pris connaissance de cette nouvelle base de données, ou devrais-je dire de ce nouveau type de base de données.

Patrice Cavezzan
Ingénieur Etudes et Développement JAVA confirmé
NETAPSYS ingénierie informatique

Datomic est une base de données transactionnelle, distribuée et temporelle. Sa principale particularité est qu'elle n'efface aucune donnée. Elle marque les modifications de ces dernières dans le temps. Vous devinez donc qu'elle nécessite beaucoup de ressources (stockage et ram).

Un peu d'histoire

Pour comprendre ce besoin, Datomic nous replonge un peu dans l'histoire de l'informatique. Historiquement, nous n'avions pas beaucoup de mémoire car c'était une ressource onéreuse.

L'une des principales techniques était de supprimer au plus vite les anciennes données pour les remplacer par des plus récentes. L'ère du « Place Oriented Programming » (PLOP) est alors apparue. Le défaut de ce style de programmation est qu'il tue l'information. On se prive donc de l'historique de cette dernière. Et voilà l'une des raisons pour laquelle nos bases de données relationnelles actuelles gèrent incorrectement les historisations de nos données. Elles sont, en effet, nées sous cette ère.

Datomic et son datamodel

L'idée est donc d'arrêter de supprimer les données car en 2015, le stockage n'est plus un problème. Pour cela, Datomic conserve toujours la donnée car elle se base sur le fait. Un fait est immuable, atomique et temporel. Datomic gère des datoms dont le datamodel est le suivant :

```
Entité, Attribut, Valeur, Temps, Opération (true / false, insert / delete )
toto, aime, pizza, 1001, true
toto, aime, bière, 1001, true
toto, aime, pizza, 1002, false
```

Ce datamodel ne possède donc pas de tables. Il se rapproche des bases de données SchemaLess (Clé/Valeur) tout en conservant les capacités ACID des bases de données traditionnelles en gérant les transactions à l'aide d'un composant appelé le Transactor.

Conclusion

Datomic me parait être une base de données à exploiter si vos applications ont un besoin d'historisation.

L'un des avantages de cette solution est sa gestion clé en main de la temporalité de nos données, ainsi que de ses capacités de scalabilité en lecture, tout en conservant la gestion transactionnelle des données. A contrario, l'un des inconvénients que je vois également est sa limite en écriture en raison de ses caractéristiques ACID.

Source :
http://blog.netapsys.fr/datomic-la-base-de-donnees-qui-noublie-rien/

Modulariser son application AngularJS avec Browserify

Cet article est issu d'une session de live-coding ayant été présentée à Devoxx France, le 8 avril 2015.

Antoine Richard
Architecte Web chez SQLi

D'aveu de développeur, node.js est une plateforme très agréable à programmer. Elle bénéficie notamment :

- D'un gestionnaire de packages efficace - npm - et d'un dépôt public associé, portés par une société solide et une communauté Open Source prolifique et de qualité,
- D'un système de modularisation - basé sur le standard CommonJS - qui favorise la production de petits programmes spécialisés que l'on va ensuite combiner entre eux pour obtenir des comportements avancés.

Cet article présente les intérêts d'utiliser ces mécaniques venues du back-end, pour développer nos front-end Web, notamment avec le framework AngularJS.

Le code présenté dans cet article est issu d'une application d'exemple affichant une liste de "mets gastronomiques" (dishes en anglais) à l'écran. Elle est disponible sur github.com/antoine-richard/devoxx-browserify-angularjs

npm

Npm est le gestionnaire de packages de node.js (il est inclus dans l'installeur). L'équipe d'AngularJS publie chaque version du framework sur npm. Il est donc possible d'installer le framework dans notre projet ce cette façon :

```
$ npm install angular@1.3.x --save-dev
```

Angular est ainsi téléchargé dans un dossier node_modules et inscrit en tant que dépendance de développement dans le fichier package.json décrivant notre application. Npm a pour vocation d'héberger toutes nos dépendances Javascript, qu'elles soient destinées au back-end ou bien aux navigateurs. Il remplace ainsi avantageusement Bower.

CommonJS

Dans un environnement CommonJS comme node, chaque fichier constitue un module de code isolé des autres (ce qui n'est pas le cas dans nos navigateurs). Chaque fichier va alors déclarer ses dépendances à l'aide du mot clé require et exposer son API via exports ou module.exports (voir nodejs.org/api/modules.html#modules_modules).

Dans ce cadre, pour utiliser l'API d'AngularJS, par exemple pour créer un module, nous allons écrire ceci :

```
var angular = require('angular');
angular.module('dishes', []);
```

Nous pouvons également utiliser require pour nos propres fichiers en précisant un chemin relatif :

```
var truc = require('./truc.js'); // l'extension .js est facultative
```

Je vous propose d'utiliser ce mécanisme pour structurer notre code AngularJS de la façon suivante :

- Chaque élément (fonction) de notre logique métier sera implémenté en pur Javascript et stocké dans son propre fichier,
- L'assemblage de ces différents éléments via framework AngularJS aura lieu ailleurs, dans un fichier responsable de la configuration applicative.

Voici ce que nous obtenons concrètement sur une factory Angular tirée du projet d'exemple :

Le fichier src/dishes/data/dishes.js expose une fonction implémentant la logique de notre factory :

```
module.exports = function() {

var dishes = [
 { name: 'Galette-saucisse', type: 'Plat principal', image: 'images/1.jpg' },
 /* ... */
];

return {
 list: function() {
  return dishes;
 }
}

};
```

Le fichier src/dishes/index.js, crée un module AngularJS et instancie chacun de ses composants :

```
var angular = require('angular');

angular.module('dishes', [])
.factory('dishesData', require('./data/dishes'))
.controller(/* ... */);
```

Les avantages d'une telle structuration sont multiples :

- La maintenabilité de notre application est améliorée grâce à la séparation de la logique métier et de l'assemblage de ces différentes briques avec AngularJS,
- La logique métier étant uniquement portée par du code Javascript/CommonJS, elle ne nécessite pas de navigateur pour être testée unitairement : nos tests seront exécutables directement dans node (avec mocha par exemple),
- Si votre code serveur est écrit en node.js, vous partagez le même standard de modularisation entre vos sources client et serveur.

Browserify

Il nous reste un petit détail à régler : un code tel que présenté ci-dessus n'est pas exécutable directement dans nos navigateurs (qui ne supportent pas CommonJS)... C'est là qu'intervient la magie de Browserify ! Cet outil permet de transformer du code CommonJS en code exécutable par les navigateurs. Une fois Browserify installé (via npm install -g browserify), nous pouvons transformer notre code à l'aide de la commande :

```
$ browserify src/app.js > bundle.js
```

src/app.js étant le point d'entrée de notre application, et bundle.js le fichier transformé par Browserify. Ce "bundle" porte l'ensemble de notre code et ses dépendances. Pour l'utiliser, il suffit de le référencer dans notre index.html :

```
<script src="bundle.js"></script>
```

N'hésitez pas à aller plus loin en parcourant le code présent sur le dépôt Github.

Comprendre l'IoT avec une boule… mais pas que…

Philippe Charrière
SQLi

J'ai eu l'immense plaisir (et l'honneur) d'intervenir à la session 2015 de Devoxx France pour faire une présentation sur l'Internet des objets, intitulée "Comprendre l'IOT avec une boule… mais pas que…", en collaboration avec Laurent Huet, @lhuet35 de la team BreizhCamp.

L'objectif de notre présentation était de présenter notre vision de "l'internet of things" en parlant des grands principes (définitions, protocoles, objets, architectures, outils…), tout en les illustrant par 4 démonstrations avec de vrais objets connectés (montrer comment faire son propre Internet of things). Le 3ème protagoniste de cette présentation était une Sphero (une boule lumineuse et colorée, télécommandée par smartphone) qui a participé à 2 des 4 démonstrations. Chacune des démonstrations permettait de montrer différentes configurations d'objets connectés plus ou moins complexes et autonomes Fig.1.

Je vais tenter dans cet article d'extraire les idées les plus significatives de cette présentation. Je définirai donc ce qu'est (selon moi) un objet connecté, ce qu'est l'Internet des objets et expliquerai de quelle manière et avec quels outils nous avons réalisé certaines de nos démonstrations.

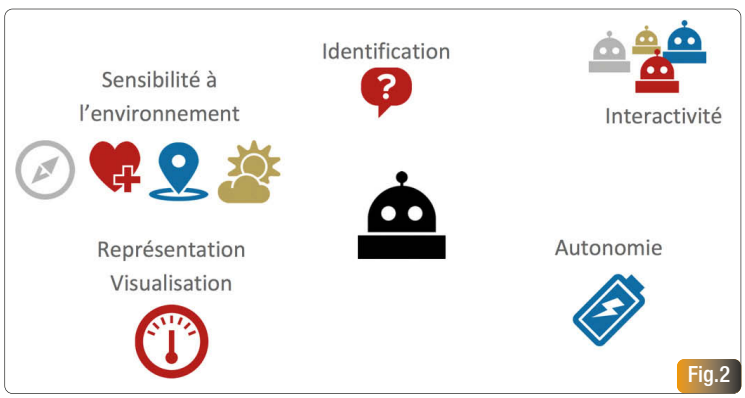


Fig.2

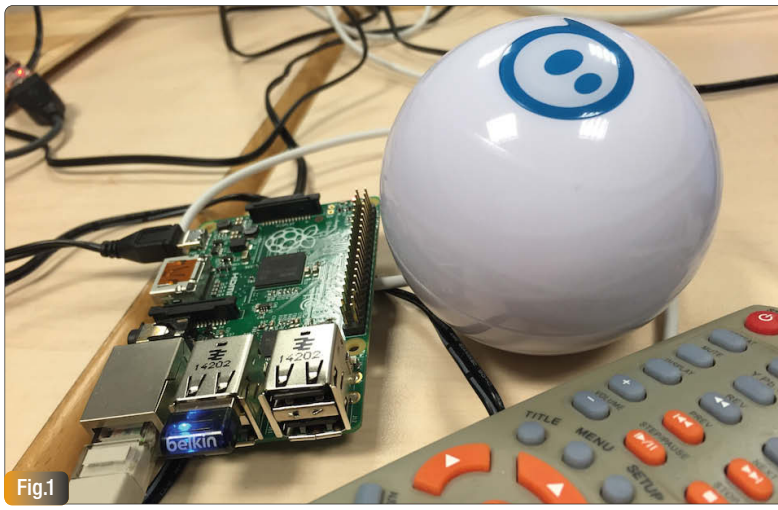


Fig.1

Qu'est-ce qu'un objet connecté ?

On parle aussi de "Thing". Donc, un objet connecté c'est une "chose" constituée d'un ensemble de composantes (des propriétés et des API). Cet objet a généralement une **sensibilité à l'environnement** par le biais de capteurs : position, température, humidité, hygrométrie, pulsations cardiaque (pensez aux montres connectées ou aux bracelets connectés). Ensuite par différents moyens, ce même objet est relié à un mode de **visualisation** embarqué (les informations de la montre par exemple, la couleur de la Sphero…) ou distant (des dashboards pour afficher la température de votre maison dans un navigateur ou sur votre smartphone). Cet objet va avoir également une **interactivité** plus ou moins poussée avec d'autres objets (la Sphero ou la montre connectée avec un smartphone, un maillage de capteurs…), mais aussi une **autonomie** plus ou moins grande (durée de vie d'une batterie, piles, secteur), et, enfin, mais non des moindres, une **identité** : pour gérer et reconnaître les objets il faut savoir les identifier Fig.2.

Qu'est-ce que l'Internet of things ?

On parle de "Internet of Things", mais aussi de "Web of Things", et même d'"Internet of Everything". Essayons d'expliquer ce que c'est en replaçant les "Things" dans leur contexte Fig.3.

Dans un premier temps, nous avions des objets isolés (bien que ce soit discutable pour une télévision), puis est apparu le concept du "Machine To

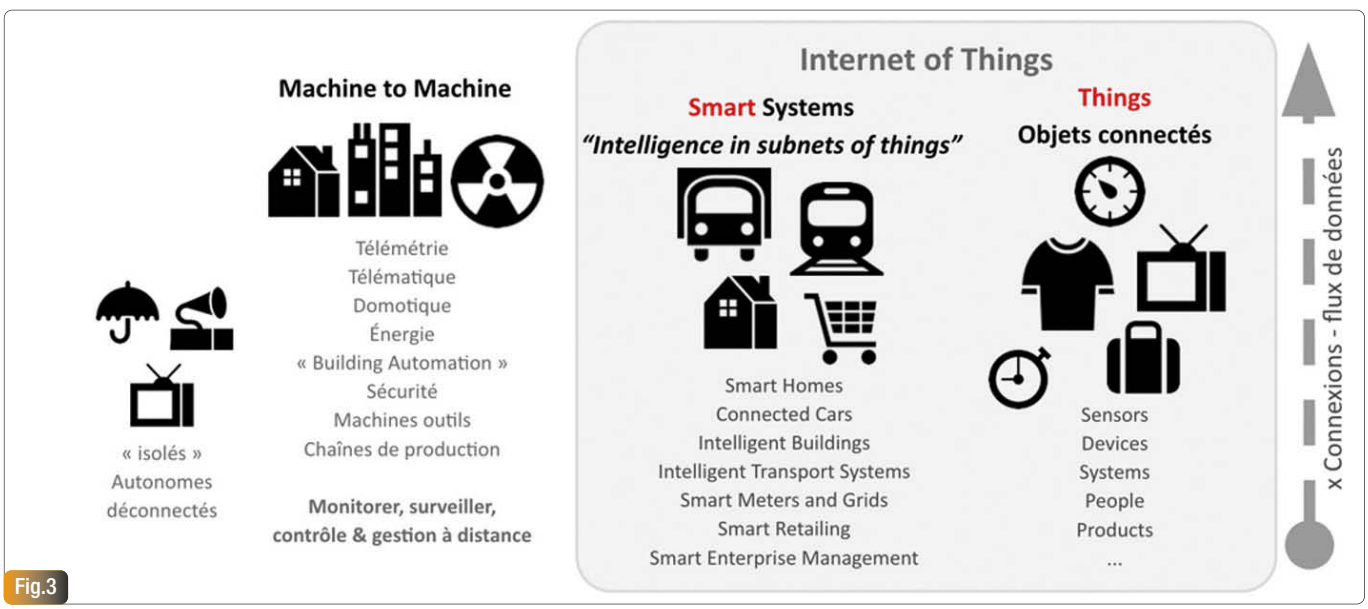


Fig.3

Machine" où la technologie permet de faire "discuter" les machines entre elles et de les contrôler à distance (pensez à tout ce qui est surveillance à distance par exemple), et enfin depuis peu, est apparu l'IoT : des objets du monde réel transmettent (et reçoivent pour certains) des informations à travers le réseau Internet, permettant ainsi de construire des "maillages d'objets" même à grande distance pour élaborer des systèmes de plus en plus intelligents (par exemple: les smart grids ou la distribution intelligente de l'électricité). Retenez, que tout est potentiellement "connectable" et en mouvement avec l'arrivée des montres connectées, des vêtements connectés... que votre domotique est en pleine mutation, et qu'il reste encore de nombreux cas d'usage à inventer.

Un marché en pleine expansion

L'IoT représente un marché à expansion rapide, notamment grâce à des modes de connectivité accrue : en effet, nous disposons tous du Wifi, de la 3G, de la 4G et d'appareils Bluetooth. Mais d'autres facteurs techniques, tels que l'augmentation des performances et de la puissance de calcul, la miniaturisation, l'autonomie (pensez à votre smartphone) contribuent aussi fortement à cette expansion. Les facteurs économiques sont également favorables, nous pouvons observer une baisse des coûts de production de matériel (regardez les prix des composants électroniques, des micro-contrôleurs comme Arduino ou encore des nano-computers comme le Raspberry PI). Le phénomène du Crowdfunding permet aussi l'émergence de nombreux projets qu'il n'y a encore pas si longtemps, n'auraient pas pu voir le jour.

La diversité des "Things" remporte l'adhésion du grand public et contribue donc grandement au marché de l'IoT Fig.4.

Les objets connectés deviennent de plus en plus complets et puissants, avec des usages multiples (de l'utile au purement récréatif, la Sphero par exemple, qui est une boule télécommandée par votre smartphone) et, j'en parlais plus haut, certains ont la capacité d'utiliser des langages et des technologies avancées ; prenez par exemple le Raspberry PI, c'est un ordinateur pas plus gros qu'un paquet de cartes à jouer qui embarque un système d'exploitation Linux complet, ou même du Windows 10 dans sa dernière version.

Mais comment ça fonctionne ? Et comment le faire soi-même ?

Nous disposons ainsi de nombreux objets susceptibles de communiquer et d'agir, mais pour que cela ait une quelconque utilité, il faut mettre en place des systèmes, des architectures capables d'échanger des informations avec les Things, de les identifier mais aussi capables de stocker, chiffrer, restituer... toutes les données générées par les things (des capteurs de température qui émettent toutes les 30 secondes pendant des mois, des montres connectées couplées à des smartphones qui comptent des pas, qui enregistrent des pulsations cardiaques, qui diffusent des coordonnées géographiques...). Lors de notre présentation, nous avons illustré ce qu'était un projet IoT à partir de démonstrations avec des objets réels et notamment en commençant (et finissant) avec une Sphero (d'où le titre de notre présentation). La Sphero, dont nous avons détourné l'usage initialement ludique, était donc l'objet connecté parfait pour une présentation. Vous allez comprendre pourquoi.

Un projet "IoT", peut être découpé en 4 parties principales :

- Les things (objets, périphérique...),
- Les infrastructures (les serveurs qui vont communiquer avec les things, dans le Cloud ou ailleurs),
- Le stockage des données,
- L'utilisation des données Fig.5.

Pour les besoins de la présentation, une des démonstrations mettait en œuvre ces 4 parties, de manière plus ou moins avancée, pour illustrer et faire comprendre les concepts.

Remarque: pour simuler Internet, nous avions notre propre routeur WiFi.

Echanger des données avec une Sphero à travers le Web

Partie 1 : les "Things"

Dans le rôle de l'objet connecté, nous avions choisi la Sphero pour les avantages suivants :

- Elle peut se déplacer et changer de couleur lorsqu'elle reçoit des informations, elle peut donc recevoir des informations,
- Elle embarque des capteurs (gyroscope, accéléromètre...) et peut transmettre des données,
- Elle communique avec l'extérieur en Bluetooth Low Energy qui est un des protocoles de communication de l'IoT (mais vous avez aussi le WiFi, zigbee, etc.).

Le plus souvent, les Things n'ont pas la capacité de communiquer directement avec les infrastructures (donc se connecter à Internet), il est donc nécessaire de mettre en place une "Gateway" ou "Hub", pouvant à la fois communiquer avec les Things et les infrastructures.

Dans le rôle de la "Gateway", nous avions choisi un Raspberry PI allié aux éléments suivants :

- Un dongle Bluetooth afin de communiquer avec la Sphero,
- Un dongle WiFi (ou un cable RJ45) pour se connecter "à l'extérieur".

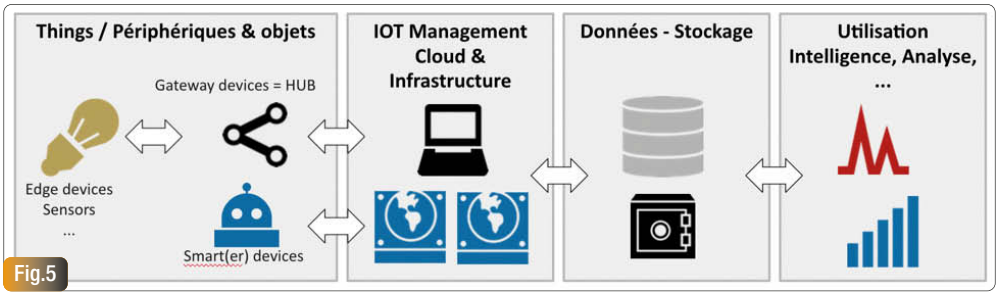


Fig.5

Fig.4

Communication entre la Sphero et la Gateway

D'un point de vue logiciel, le Raspberry PI communiquait avec la Sphero grâce au framework CylonJS (http://cylonjs.com) qui s'appuie sur NodeJS (JavaScript est de plus en plus présent dans le monde de l'IoT, même si les Things se programment beaucoup en C, Python ou bien Lua). CylonJS est un projet qui propose les API et les drivers nécessaires pour communiquer avec de nombreux objets, tels l'AR Drone (de la société Parrot), mais aussi la Sphero.

Le "code de pilotage" ressemble à ceci :

```
Cylon.robot({
 connections: {
  sphero: {adaptor: 'sphero', port: config.spheroPort()},
```

```
},
devices: {
 sphero: {driver: 'sphero'}
},
work: function (my) {
 // start sphero
 my.sphero.roll(5, Math.floor(Math.random() * 360));

 after((1).seconds(), function () {
  var opts = {n: 200, m: 1, pcnt: 0};
  my.sphero.setDataStreaming(["locator", "accelOne", "velocity"], opts);
  my.sphero.setBackLED(192);
  my.sphero.stop();
 });


 my.sphero.on("data", function (data) {
// get some data
 });
}
}).start();
```

Communication entre la Gateway et le serveur

Il existe plusieurs protocoles de communication dans le monde l'IOT, pour les besoins de notre démonstration, nous avions choisi le protocole MQTT (Message Queue Telemetry Transport), dont les initiateurs principaux sont IBM et Eurotech. MQTT a été standardisé à l'OASIS en novembre 2014. Nous l'avons choisi car c'est un standard simple et léger, reposant sur TCP/IP, utilisant un modèle évènementiel, et il est "content agnostic".

Le modèle évènementiel de MQTT repose sur l'application du pattern "publish / subscribe", il faut donc un serveur de messages (on parlera de broker MQTT) et des clients MQTT pour publier des messages ou s'abonner à des channels de messages, on parlera de topics Fig.6.

Un autre avantage de MQTT est qu'il existe déjà de nombreuses implémentations de brokers et de clients dans différents langages.

Dans le cadre de notre démonstration nous avons utilisé Mosca (http://www.mosca.io/) comme broker MQTT (il repose sur NodeJS), donc installé sur un laptop qui faisait office de serveur.

Côté "Gateway" (sur le Raspberry PI donc), il se trouve que le projet CylonJS propose aussi un driver MQTT (http://cylonjs.com/documentation/platforms/mqtt/), le code de pilotage de la

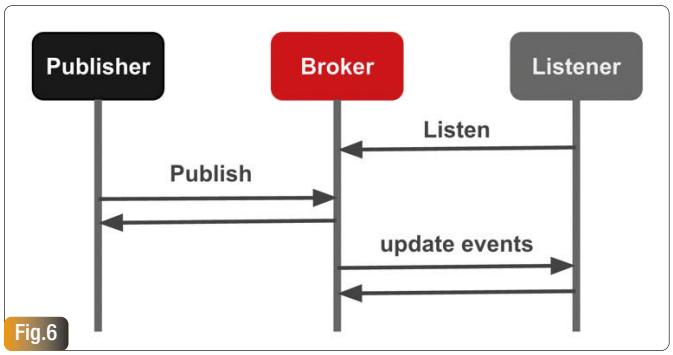


Fig.6

Sphero a donc été très facile à modifier :

```
Cylon.robot({
 connections: {
  sphero: {adaptor: 'sphero', port: config.spheroPort()},
  server: {adaptor: 'mqtt', host: mqttServer}
 },
 devices: {
  sphero: {driver: 'sphero'}
 },

 work: function (my) {
  // S'abonner à tous les topics de type /sphero/*
  my.server.subscribe('/sphero/+');

  // start sphero
  my.sphero.roll(5, Math.floor(Math.random() * 360));

  my.server.on('message', function (topic, data) {
   // à la réception d'un message en provenance du serveur,
   // je déclenche les mouvements de la Sphero
   data = JSON.parse(data)
   if (topic == "/sphero/move-sphero") {
    my.sphero.roll(data.speed, data.angle);
   }
   if (topic == "/sphero/stop-sphero") {
    my.sphero.stop();
   }
  });
```

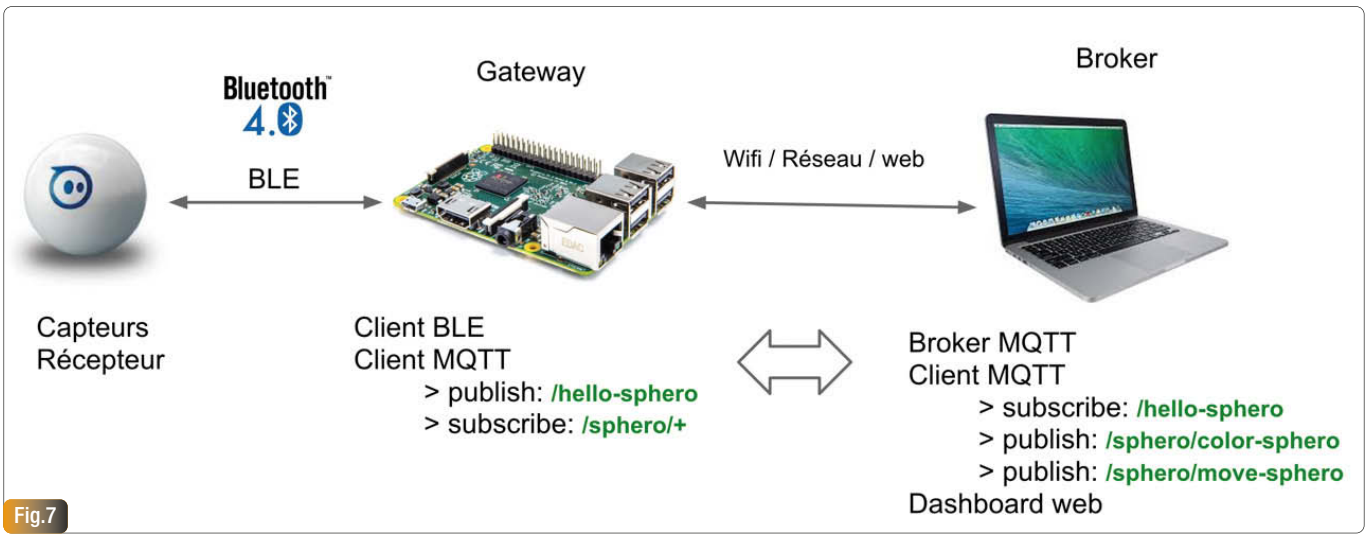


Fig.7

```
my.sphero.on("data", function (data) {
  // lorsque la Sphero se déplace je renvoie les informations au serveur (broker)
  // sur le topic hello-sphero
   my.server.publish("hello-sphero", JSON.stringify(data));
 });
}
}).start();
```

Parties 2, 3 & 4 : le serveur, les données et les dashboards

Le serveur de la démonstration (un portable) comportait plusieurs composants :

- NodeJS pour faire fonctionner le broker MQTT et le serveur d'application
- Le broker MQTT Mosca, présenté plus haut,
- Une application Web développée avec ExpressJS qui fournissait à la fois une API (obtenir la liste des objets connectés et leurs informations via une API REST, par exemple : http://my_iot_server/connected/things) et le front Web,
- Une base Redis (qui contenait la liste des objets connectés et leurs informations),
- Une webapp développée avec Polymer (http://www.polymer-project.org) qui fournissait des informations graphiques (les dashboards) à propos de la Sphero (grâce au framework Epoch (http://fastly.github.io/epoch/) à base de D3.js),
- Pour permettre des échanges temps réels sur toute la chaîne de communication de la Sphero jusqu'au navigateur, nous avons utilisé côté serveur le framework socket.io (http://socket.io/) et côté navigateur, le client socket.io.js du même projet: donc tous les messages MQTT en provenance de la Sphero via la Gateway et le broker étaient transmis via socket au navigateur, et de la même façon, l'interface Web permettait d'envoyer des informations de pilotage à la Sphero pour la faire bouger ou changer de couleur.

Ces éléments permettaient d'illustrer un cheminement complet de l'objet vers le broker, puis le client, et ensuite de renvoyer des données dans le sens inverse, vers l'objet : Fig.7 et 8.

La Sphero est un objet ludique, mais imaginez à la place un éclairage de ville dont on pourrait facilement gérer l'intensité, la couleur (ambiance), ou même détecter des problèmes physiques (collision avec véhicule...) et vous comprendrez l'utilité de l'IOT.

Il ne faut pas oublier non plus que les objets sont plus ou moins "intelligents" (logiciels embarqués), communicants et puissants : si la Sphero a besoin d'une Gateway qui permet de faire "le pont" vers Internet, mais

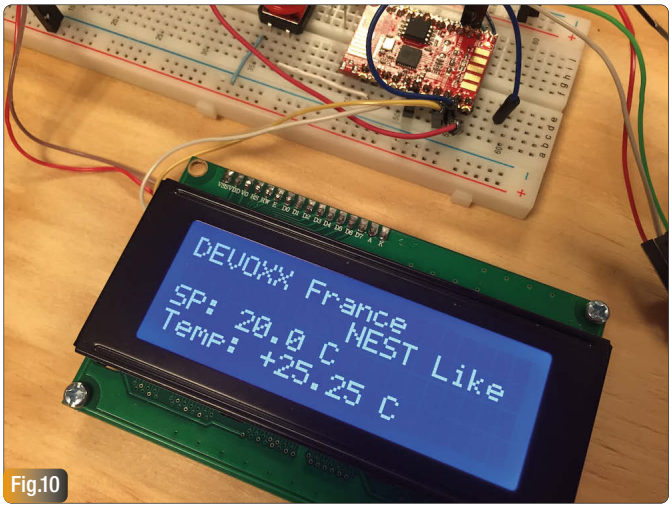


Fig.10

Nest Like par @lhuet35

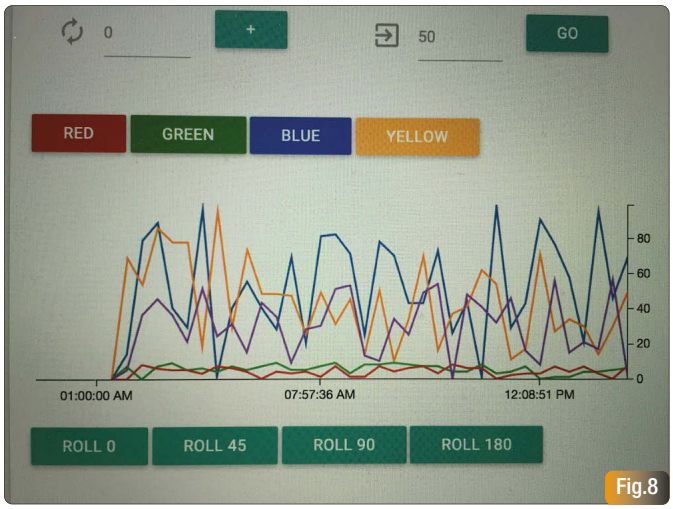


Fig.8

Le dashboard "sphero" en pleine action

aussi qui "embarque" le client MQTT pour échanger avec le serveur, certains objets peuvent communiquer directement en WiFi et embarquer la partie logicielle (donc plus besoin de Gateway).

Ce cas de figure était l'objet d'une autre démonstration qui mettait en œuvre un micro contrôleur ESP 8266 à peine plus gros qu'un sucre embarquant un client MQTT codé en C, et capable de communiquer directement en WiFi (et pour un prix dérisoire de l'ordre de 5$) Fig.9.

Fig.9

l'ESP 8266

L'ESP 8266 était relié à un capteur de température et à un écran à cristaux liquides. Sur le même principe que la démonstration précédente il était possible d'interagir à distance avec l'ESP 8266 (envoyer la température vers le serveur, envoyer des informations du serveur vers l'écran : globalement nous avions un début de NEST). La seule différence était que ce nouvel objet se passe de Gateway, car il embarque à la fois les moyens de communication et l'intelligence Fig.10.

Notre objectif à travers ces démonstrations était double : illustrer l'Internet des Objets, mais aussi démontrer que, simplement, on peut être acteur de ce nouveau monde et faire son propre Internet des Objets.

Quelques idées pour aller plus loin

Il faudrait beaucoup plus de pages et de temps pour être exhaustif, mais voici quelques points supplémentaires si vous souhaitez creuser plus en profondeur le sujet :

- Il existe d'autres protocoles, et notamment CoAP (Constrained Application Protocol) pensé pour les réseaux LoWPAN (Low-Power Wireless Personal Area Network), des réseaux de "toutes petites" machines,
- Tous ces objets génèrent beaucoup de données qu'il faut pouvoir exploiter. Après avoir assisté à la présentation de Nicolas Muller à Devoxx France 2015 sur InfluxDb et Grafana, je ne saurai que trop vous conseiller de jouer avec,
- Les composants et les kits de démarrage "électroniques" avec des micro-contrôleurs ou des nano-computers sont de plus en plus abordables et vous permettent de prototyper des objets avec des capteurs très facilement; par exemple, mon prochain jouet sera certainement un "Grove Starter Kit" (ou comment faire de l'électronique façon Lego lorsque l'on n'y connait rien du tout).

Des licences… plus ou moins libres !

Dans le monde du logiciel, on rencontre toutes sortes de licences, dont les termes décrivent les possibilités d'utilisation du code source et du logiciel, souvent compilé, en résultant : l'exécutable. Et depuis les années 1990, la "mode" de la licence Libre (ou encore "Open Source"), a séduit de nombreux développeurs, voire des entreprises qui y voient des intérêts plus ou moins grands. Car les logiciels libres offrent plus de droits à leurs utilisateurs, et notamment celui de pouvoir accéder au code source, afin de pouvoir éventuellement le modifier, et aussi le redistribuer.

Gaël Duval

Gaël Duval - Promoteur du logiciel libre, entrepreneur IT. Actif dans la communauté du logiciel libre depuis 1995, Gaël est à l'origine de la création des sociétés Mandraksoft (devenu Mandriva) qui éditait la distribution Linux "Mandrake Linux", puis de la société Ulteo, qui a développé une solution libre et multiplateforme de gestion et de délivrance d'applications. Il s'intéresse actuellement au développement HTML5/JS et à l'Intelligence Artificielle. http://www.gaelduval.com

La vague est si importante que l'on a vu naître, dans la mouvance de Linux ou encore d'Apache - deux logiciels libres phares - des projets à grand succès qui sont couverts en grande partie par des licences libres. Par exemple... Android, le système d'exploitation qui équipe des centaines de millions de smartphones dans le monde (lui-même dérivé de Linux, il ne faut pas l'oublier). Et plus récemment encore, une information incroyable a fait date : un responsable de Microsoft a annoncé qu'il n'était pas exclu que Windows lui-même soit un jour diffusé sous licence libre !

Un peu d'histoire...

Historiquement, c'est à dire dans les années 50/60, lorsque l'informatique était une discipline naissante encore hyper-confidentielle, tous les logiciels étaient libres, dans le sens où les développeurs se les échangeaient et les modifiaient sans arrière-pensée. A l'époque c'était plutôt le matériel qui focalisait l'attention et les enjeux. Mais petit à petit, avec la naissance d'une industrie du logiciel dans les années 1970, des sociétés ont commencé à "serrer la vis" et à vouloir utiliser des restrictions, évidemment pour des raisons mercantiles. En parallèle les lois américaines ont commencé à prendre en compte la notion de copyright pour les logiciels. Au début des années 80, la société IBM a enfoncé le clou en instaurant une politique de diffusion des logiciels uniquement sous forme de binaires, c'est à dire, sans la recette de cuisine qui permet de les créer.

C'est suite à ces changements profonds de pratiques que l'idée du "Free Software" est née. Et en particulier, peu avant le milieu des années 1980, certains hackers de l'époque, et en particulier un certainRichard Stallman, offusqués de ne pas pouvoir débuguer ou améliorer eux-mêmes certains logiciels propriétaires, ont décidé de créer un système d'exploitation compatible avec UNIX qui n'aurait pas les contraintes d'un logiciel propriétaire. Le projet GNU (acronyme récursif de "GNU's Not Unix") était né, et bientôt les premières licences libres, dont la fameuse licence "GPL" avec lui.

Mais si la GPL reste une des licences libres les plus connues, notamment car elle couvre Linux pour laquelle elle fut historiquement l'une des toutes premières créées(1), une autre licence extrêmement répandue est apparue à la même époque, et même un peu avant : la toute première licence BSD (pour "Berkeley Software Development") est en effet apparue en 1988.

Libre et "Copyleft"

Par la suite, ces licences se sont améliorées, afin d'adresser des cas particuliers ou des situations nouvelles (comme le logiciel en mode SAAS), mais elles ont également vu naître un grand nombre de licences plus ou moins dérivées, et plus ou moins libres.

Quoi qu'il en soit, la comparaison entre les deux licences GPL et BSD est intéressante : en effet, bien qu'étant deux licences reconnues comme libres (aussi bien par la Free Software Foundation que par l'Open Source Initiative), elles diffèrent fondamentalement par un aspect que l'on appelle le "CopyLeft".

En effet, si les libertés de base garanties par un logiciel libre sont de pouvoir accéder au code source et de pouvoir le redistribuer, éventuellement modifié et/ou compilé, la licence GPL, contrairement à la BSD, impose à celui qui redistribue de ne pas changer la licence initiale. C'est à dire que du code source initialement couvert par la GPL doit garantir des droits identiques aux utilisateurs de ce code source modifié et redistribué. Alors que dans un logiciel couvert par la BSD (ou licence équivalente), la seule contrainte sera de publier une notice comportant un certain nombre d'informations, dont les auteurs. C'est pour cette raison qu'Apple, par exemple, éditeur de logiciel propriétaire devant l'éternel, qui interdit la publication sur son App Store(2) de logiciels en licence GPL, ne se prive pas pour utiliser un grand nombre de composants libres dans ses logiciels, par exemple dans iOS. Evidemment il n'utilise que des composants sous licence BSD-like. Les licences de style BSD sont appelées des licences libres "permissives".

Alors... Propriétaire ou libre ? Copyleft ou pas Copyleft ?

Tout est affaire de... goût ! Et de convictions. Mais aussi de contraintes : même si les licences libres n'empêchent en rien une utilisation commerciale, la peur de voir un investissement de R&D important récupéré par un tiers à peu de frais peut faire peur. Idem pour la peur de voir son modèle commercial s'effondrer car le produit est disponible "gratuitement" sur Internet. C'est là qu'il faut se poser la question du "business-model" : est-ce que je vends du logiciel ou le service qui va avec ? Mais aussi se poser la question de l'avantage d'utiliser du libre pour développer son propre logiciel (accélération du processus de R&D par exemple).

Une question importante également, dans le cas d'une société commerciale, est la notion de "propriété du code", ou plus exactement du "copyright sur le code". En effet, diffuser sous licence libre n'empêche pas le droit d'auteur sur le code distribué. Si la société récupère des contributions externes qu'elle intégre à son code source, elle a tout intérêt à demander aux

(1) Sa première version fut réalisée en 1989 à partir de plusieurs licences libres spécifiques à certains développements de R. Stallman, notamment l'éditeur de textes EMACS

(2) Deux exemples concrets : le très populaire VLC a été interdit d'App Store pendant de très longs mois car il était originellement publié en GPL. Tout le code GPL a dû être réécrit "en aveugle" par des auteurs différents et sous une licence plus permissive, afin que VLC puisse faire son retour sur App Store.
Le client iOS d'Ulteo, s'est vu retirer d'App Store par un concurrent pour la raison qu'il était également couvert par la GPL. Une nouvelle version, utilisant seulement du code appartenant à Ulteo et au projet FreeRDP, a pu être republiée en licence Apache (BSD-like)...

auteurs de leur céder leurs contributions, pour une raison très simple : si un jour la société souhaite changer la licence, ou publier son logiciel en double-licence par exemple, le refus d'un seul des auteurs du code concerné bloquera tout le processus. Alors que si la société possède 100% du code source qu'elle diffuse, même sous licence libre, elle aura tout loisir si elle le désire, soit de changer de licence soit d'adopter une double-licence, voire même de générer une version "propriétaire", par exemple pour un client qui ne veut pas de logiciel libre (ça arrive !). D'un autre coté, dans l'optique d'un projet très communautaire, adopter une licence libre de type GPL sera très bien vu de la communauté qui sera alors plus encline à donner de son temps et à contribuer au projet : Linus Torvalds l'a bien compris quand il a créé Linux, alors qu'à l'origine il le diffusait sous une licence plus restrictive.

Quant au Copyleft, c'est encore une affaire de contexte. Il est certain que ce type de licence favorise une plus large diffusion, y compris dans des logiciels propriétaires. Mais c'est aussi le risque de se voir "piquer" son logiciel sans aucune contrepartie derrière... sauf évidemment la fierté de se retrouver éventuellement cité dans les termes des licences Apple ou Microsoft... Une situation certes probablement valorisante sur un CV.

Au final, le choix d'une licence libre est vraiment affaire de contraintes et de jugement personnel. Il peut être également plus sûr d'utiliser une licence bien connue, et ayant fait ses preuves comme GPL ou BSD. Il sera plus compliqué, en cas de problème, de trouver de la jurisprudence ou un avocat qualifié concernant "My Own Home Made Public License".

Mais comment faire respecter une licence libre ?

Licence libre ou pas, il n'en reste pas moins que lorsque les sources sont diffusées, il peut être utile de pouvoir contrôler qu'aucun logiciel tiers ne respectant pas les termes de la licence utilisée pour la diffusion puisse être distribué ou commercialisé. Pendant longtemps, lorsqu'un doute était permis, il fallait fouiller dans les exécutables pour essayer de repérer des "traces" pouvant constituer des preuves, comme des commentaires dans le code ou des headers de bibliothèques. Une autre approche consistait à essayer de reproduire des comportements singuliers du logiciel, par exemple : un bug connu ! Puis des outils plus perfectionnés sont apparus, comme "BlackDuck" qui permettent d'identifier les licences du code utilisé pour assembler un logiciel.

Et la question ne se pose plus : les licences libres sont bien reconnues par les tribunaux. Plusieurs plaintes ont donné lieu à des condamnations, ou ont réussi à faire se conformer des sociétés abusant du logiciel libre : on retiendra par exemple la société Free qui, suite à une plainte de la FSF(3) France, a fini en 2011 par publier les sources des logiciels libres parfois modifiés qui servent à faire fonctionner la Freebox. ⊞

(3) Free Software Foundation

LES PETITES CONTROVERSES

Il ne faut pas penser que lorsque l'on parle entre partisans du Logiciel Libre, la discussion est forcément calme et sereine. Elle peut même être orageuse, voire violente. L'un des épisodes les plus violents qui a eu lieu dans la communauté du Logiciel Libre et dont je me souvienne, fut probablement la confrontation des utilisateurs de KDE et de GNOME, qui a culminé vers 1999. A l'époque, le problème était Qt, la bibliothèque graphique à partir de laquelle l'environnement graphique KDE était programmé.

Qt, de la société Troll Tech, fut d'abord une bibliothèque graphique propriétaire : bien que l'on puisse avoir accès à ses sources et la recompiler, les termes de sa licence interdisaient de la modifier et d'en redistribuer des versions modifiées. C'était tout le problème ! La plupart des membres de la communauté du libre de l'époque n'étaient pas très à l'aise qu'une brique logicielle qui devenait de plus en plus proéminente dans les systèmes GNU/Linux/X11 ne fusse pas libre. Et cela posait un problème certain à tous les éditeurs de distributions Linux.

Devant ce problème, les utilisateurs ont réagi de manières différentes. La première façon de faire fut de ne pas s'en soucier, ce qui fut d'ailleurs sans doute l'attitude adoptée par le plus grand nombre, pas forcément conscients des enjeux.

La deuxième attitude, adoptée par certains, fut d'encourager vivement la société Troll Tech à passer Qt en licence libre, en essayant de leur prouver qu'ils avaient intérêt à le faire.

La troisième attitude, la plus visible et la plus violente, a été de refuser Qt et KDE en bloc à cause de ce problème. Ce qui a été à l'origine du développement rapide de l'embryonaire environnement graphique GNOME, basé sur la bibliothèque graphique GTK+ qui avait déjà servi à programmer le logiciel de retouche d'image « The Gimp ».

Très rapidement, les hostilités furent déclarées et les engueulades devinrent franches entre les partisans de Qt/KDE et ceux de GTK/GNOME, principalement sur les groupes et listes de discussion. A l'époque, devant l'ampleur des mécontentements et devant la pression GTK/GNOME, Troll Tech créa une nouvelle licence pour Qt, appelée « QPL ». Une licence "quasiment libre", qui répondait aux critères de l'« Open-Source », mais pas à ceux, plus restrictifs, du « Free Software ».

La situation devint même parfois relativement malsaine : on pouvait souvent lire à l'époque des messages frisant la mauvaise fois, essayant par exemple de démontrer que Gtk, alors un toolkit encore jeune et assez frustre, était déjà supérieur techniquement à Qt, alors que sa maturité était bien supérieure.

Finalement, nous avons été plusieurs dans la communauté à souhaiter que cette agitation fratricide et nuisible pour le mouvement du libre cesse, et que KDE et GNOME puissent cohabiter en paix.

Bien heureusement, Troll Tech, qui à l'époque avait contacté des acteurs du libre pour leur demander leur point de vue, a fini par suivre nos suggestions de passer Qt en licence vraiment libre. Ils choisirent une double licence dont la GPL. La « guerre » s'acheva donc, en laissant quelques traces malgré tout : la plupart des « anciens » ont continué à défendre bec et ongle leur "meilleur environnement graphique" pendant des années.

Plus récemment, en 2008, une polémique concernant les licences a de nouveau vu le jour entre Linus Torvalds et Richard Stallman. En effet, Linus a refusé d'appliquer à Linux la GPL v3 (qui a été conçue afin d'éviter des contournements abusifs de la GPL v2 via des techniques de DRM(4), en arguant du fait qu'étant donné le nombre d'auteurs de Linux, leur faire accepter à chacun cette modification générale de licence était juste impossible...

Il faut en effet savoir que le noyau Linux est distribué sous les termes de la licence GPL v2 à l'exception de la phrase "or any later version".

(4) Le constructeur d'enregistreurs numériques TiVo est bien connu pour avoir utilisé ce genre de procédé

Linux en tant que poste de développeur

Aurélie Vaché
Développeuse Web chez **atchikservices**

Lorsque j'ai été embauchée par atchikservices il y a 9 ans, J'ai commencé avec une Fedora Core 3 pendant quelques années, puis je suis passée à la distribution Ubuntu. Il a fallu s'habituer à l'environnement de bureau Unity, mais on s'y fait vite :-). Je développe et maintiens des applications en production et je pense qu'il n'y a pas mieux que Linux pour cela. J'ai un double écran et avoir par exemple, mon IDE Eclipse, mon navigateur ou mon logiciel de messagerie Thunderbird sur l'écran de gauche, et, sur celui de droite, l'explorateur de package d'Eclipse ou le terminal, c'est excellent.
Vous allez vous rendre compte dans cet article qu'avoir un environnement de Dev sur Linux permet d'améliorer la productivité. Il est plus facile par exemple de gérer différentes versions de vos outils préférés (java, maven ...), de compiler vos projets, et vous pourrez même avoir un environnement de dev semblable à celui de production.

Une distribution tu choisiras

Vous avez choisi de passer votre environnement de développement à Linux, c'est un bon choix ! En premier lieu, il faut que vous choisissiez une distribution, et, parmi la multitude des distributions Linux qui existent, je suis d'accord il y a de quoi être perdu et ne pas savoir quoi choisir. Afin d'aiguiller votre choix, il faut savoir qu'en ce moment il y a 3 distributions qui sont prisées par les développeurs :

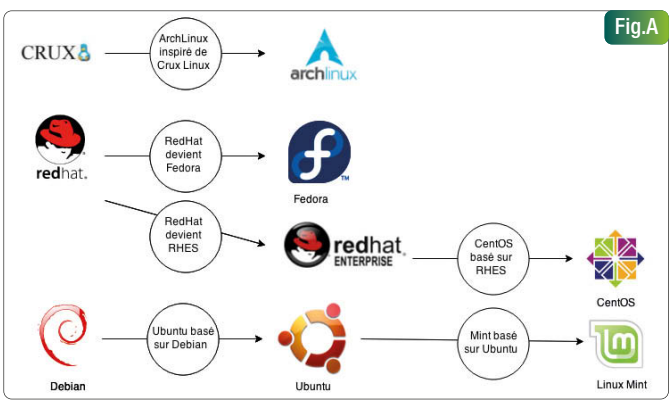
- Ubuntu
- Linux Mint
- ArchLinux Fig.A

Si vous avez déjà entendu parler de Kali Linux, sachez que cette dernière est basée sur Debian et est plus réservée aux tests de sécurité.
On peut choisir une distribution pour son Desktop différente de celles des serveurs de tests et de production : je code sur une Ubuntu et les applications que je déploie tournent sur une CentOS 7. Pourquoi choisir une distribution et pas une autre ? Je dirais que cela dépend de vos besoins, du contexte, de votre affinité avec une distribution, si vos collègues codent déjà sur une distribution particulière, si vous désirez une interface graphique ou non ... Bref, c'est a vous de choisir ! :-)

Des paquets tu installeras :

Vous avez choisi et installé une distribution Linux. Parfait! Maintenant il faut s'attaquer aux paquets/aux outils que vous devez installer.
La liste des paquets que vous devrez installer dépendra de vos besoins ainsi que des technologies et langages que vous utilisez. Je vais vous en lister quelques-uns mais si vous avez un doute sur le paquet à installer, pas de panique, Internet regorges de tutoriels :-).
Avant toute chose, les exemples que je vous donne ci-dessous sont basés sur Ubuntu donc j'utilise apt-get pour installer les paquets (car il s'agit d'un OS basé sur Debian tout comme Linux Mint). Mais si vous avez installé un système basé sur une Red Hat telle que Fedora, il vous faudra utiliser yum à la place ou encore pacman pour Arch Linux.

Fig.A



Apache

```
$ sudo apt-get install apache2
```

MySQL

```
$ sudo apt-get install mysql-server
```

MongoDB

En premier lieu il faut importer la clé publique dans votre machine :

```
$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10
```

Ensuite il faut ajouter le repository de mongodb :

```
$ echo "deb http://repo.mongodb.org/apt/ubuntu "$(lsb_release -sc)"/mongodb-org/3.0
multiverse" | sudo tee /etc/apt/sources.list.d/mongodb.list
```

Et pour finir, vous pouvez installer le paquet :

```
$ sudo apt-get update
$ sudo apt-get install mongodb
```

PHP

```
$ sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt php5-mysql php5-soap
```

Java

Vous pouvez installer plusieurs versions de la JVM si vous le désirez et surtout si vous en avez besoin. Pour cela il faut que vous alliez sur le site d'Oracle, que vous téléchargiez l'archive, que vous la déplaciez dans le répertoire /opt/ ou /usr/lib/jvm/, par exemple, et, ensuite vous pouvez l'installer :

```
$ cd /usr/lib/jvm/
$ tar zxvf jdk-8u45-linux-i586.tar.gz
$ sudo update-alternatives --install "/usr/bin/java" "java" "/usr/lib/jvm/jdk1.8.0_45/bin
/java" 500
```

On peut de cette manière installer plusieurs versions de Java sur sa machine et activer à tout moment celle que l'on souhaite :

```
$ sudo update-alternatives --config java
```

Il existe 5 choix pour l'alternative java (qui fournit /usr/bin/java).

Sélection	Chemin	Priorité	État
0	/usr/lib/jvm/java-7-openjdk-i386/jre/bin/java	1071	mode automatique
1	/opt/java/bin/java	1	mode manuel
2	/opt/java/bin/javac	1	mode manuel
3	/usr/lib/jvm/java-7-openjdk-i386/jre/bin/java	1071	mode manuel
* 4	/usr/lib/jvm/jdk1.7.0_25/bin/java	1	mode manuel
5	/usr/lib/jvm/jdk1.8.0_45/bin/java	500	mode manuel

Appuyez sur <Entrée> pour conserver la valeur par défaut[*] ou choisissez le numéro sélectionné :
Vous pouvez également activer la version de Java que vous souhaitez si vous la connaissez :

```
$ sudo update-alternatives --set java /usr/lib/jvm/jdk1.8.0_45/bin/java
```

De cette manière il est super simple d'utiliser java 7 pour un projet, java 8 pour un autre qui utilise les lambda expressions ou java 6 pour un vieux projet par exemple.

Python

```
$ sudo apt-get install libssl-dev openssl
$ wget https://www.python.org/ftp/python/3.4.1/Python-3.4.1.tgz
$ tar -xvf Python-3.4.1.tgz
$ cd Python-3.4.1/
$ ./configure
$ make
$ sudo make install
```

Maven

```
$ sudo apt-get install maven
```

Subversion

```
$ sudo apt-get install subversion
```

Git

Si vous souhaitez installer Git, il vous faudra installer au préalable les bibliothèques suivantes : curl, zlib, openssl, expat et libiconv. Si vous n'avez pas déjà installé la librairie libssl-dev qui est nécessaire pour Python notamment, n'oubliez pas de l'installer pour Git.

```
$ sudo apt-get install libcurl4-gnutls-dev libexpat1-dev gettext libz-dev libssl-dev
```

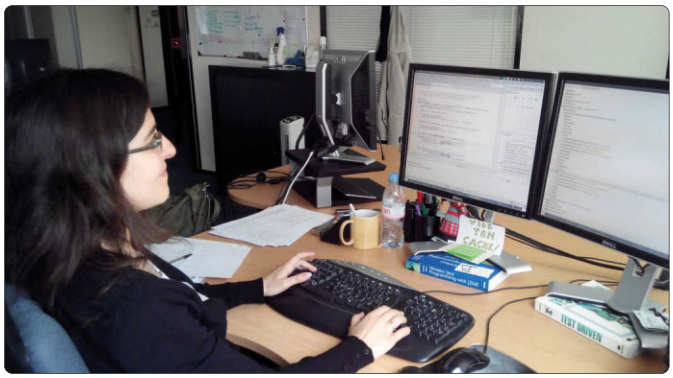
Puis il ne vous reste plus qu'à installer le package de Git :

```
$ sudo apt-get install git
```

Android

Si vous souhaitez développer des applications natives sous Android et que vous êtes sur un Ubuntu 64 bits, il vous faudra installer la librairie de compatibilité en 32 bits :

```
$ sudo apt-get install ia32-libs
```



De la configuration tu feras

Pour que votre poste de travail ait toutes les bonnes variables d'environnement configurées, les bons paths, il faut créer un fichier .profile et l'appeler dans votre fichier .bashrc :

```
$ cat ~/.bashrc
...
source ~/.profile
```

Voici un aperçu de mon fichier .profile :

```
$ cat ~/.profile
...
# some more ls aliases
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

JAVA_HOME="/usr/lib/jvm/jdk1.7.0_25/"
JAVACMD="$JAVA_HOME/bin/java"
PATH="/usr/lib/jvm/jdk1.7.0_25/":"/home/my_user/applis/android-sdk-linux":$PATH
export JAVA_HOME
export PATH

export MAVEN_OPTS="-Dmaven.wagon.http.ssl.insecure=true -Dmaven.wagon.http.ssl.
allowall=true"

# The next line updates PATH for the Google Cloud SDK.
source /home/my_user/applis/google-cloud-sdk/path.bash.inc

# The next line enables bash completion for gcloud.
source /home/my_user/applis/google-cloud-sdk/completion.bash.inc

export ANDROID_EMULATOR_FORCE_32BIT=true

export ANDROID_HOME="/home/my_user/applis/android-sdk-linux/"
```

```
# Choose the default editor for crontab edition
export EDITOR=vim
```

Lorsque vous effectuez une modification du .bashrc, si vous ne voulez pas sortir de la session, il vous suffit de le sourcer :

```
$ source ~/.bashrc
```

Il est également possible de personnaliser le prompt de son shell ou celui des serveurs que l'on administre via ssh, c'est pratique et tout se passe dans le fichier .bashrc :

```
#prompt changes default color to red in production servers
DEFAULT="\e[00m"
RED="\e[31m"
PS1="$RED[\u@\h \W]\$ $DEFAULT";
```

```
[    @prod01 ~]$
[    @prod01 ~]$
[    @prod01 ~]$ uptime
 15:08:08 up 48 days, 11:52,  4 users,  load average: 0.03, 0.12, 0.13
[    @prod01 ~]$
```

A noter que sur Ubuntu, au lieu de setter une valeur pour la variable EDITOR, comme vu dans le fichier .profile, vous pouvez re-sélectionner à tout moment votre éditeur par défaut :

```
$ select-editor

Select an editor.  To change later, run 'select-editor'.
  1. /bin/ed
  2. /bin/nano      <---- easiest
  3. /usr/bin/vim.basic
  4. /usr/bin/vim.tiny

Choose 1-4 [2]:
```

Pour vérifier que les dernières versions par exemple de Java, de maven ou de php que j'ai installées sont bien présentes dans mon PATH, il suffit de le demander en ligne de commande :

```
$ java -version
java version "1.7.0_25"
Java(TM) SE Runtime Environment (build 1.7.0_25-b15)
Java HotSpot(TM) Server VM (build 23.25-b01, mixed mode)

$ mvn -v
Apache Maven 3.0.5
Maven home: /usr/share/maven
Java version: 1.7.0_25, vendor: Oracle Corporation
Java home: /usr/lib/jvm/jdk1.7.0_25/jre
Default locale: fr_FR, platform encoding: UTF-8
OS name: "linux", version: "3.13.0-49-generic", arch: "i386", family: "unix"

$ php -v
PHP 5.6.7-1+deb.sury.org~trusty+1 (cli) (built: Mar 24 2015 11:13:27)
Copyright (c) 1997-2015 The PHP Group
Zend Engine v2.6.0, Copyright (c) 1998-2015 Zend Technologies
  with Zend OPcache v7.0.4-dev, Copyright (c) 1999-2015, by Zend Technologies

$ mongo --version
MongoDB shell version: 3.0.0
```

Pour me connecter aux différents serveurs que l'on gère je me connecte en ssh. Et si l'on veut se faciliter la vie et gagner du temps, au lieu de saisir à chaque fois le port du serveur (si différent du port 22) et le login, on peut gérer tout ce petit monde dans le fichier config de SSH :

```
$ cat .ssh/config
Host dev01
   HostName dev01
   Port 5555
   User dev

Host dev02
      HostName dev02
      Port 5555
      User dev

Host rec01
      HostName recette01
      Port 5554
      User dev
```

Et pour éviter d'avoir à chaque fois à saisir le mot de passe, il suffit d'insérer votre clé publique dans le fichier ~.ssh/authorized_keys du serveur distant.

Des outils tu te muniras

Pour installer des outils, vous le savez, il existe le gestionnaire d'installation et de désinstallation de paquets en ligne de commande, mais il existe également une version graphique (la Logithèque Ubuntu par ex. ou Synaptic), et il est également possible d'installer un logiciel via son package RPM. Quotidiennement j'utilise une multitude d'outils sur Linux, en voici une liste non exhaustive :

Éditeurs de texte :

- VI(M). J'adore VI personnellement, c'est un éditeur de texte super léger et très pratique. C'est mon partenaire du quotidien. En quelques secondes et en tapant quelques caractères seulement, il est possible d'effectuer des actions sur un fichier texte.

Imaginez que vous avez un fichier contenant des IDs, il est possible de rajouter au début de chaque ligne du fichier une commande SQL SELECT :

```
:%s#^#SELECT name from Client where client_id='#g
```

Et vous pouvez fermer votre instruction SELECT également :

```
:%s#$#';#g
```

Si vous voulez supprimer 10 lignes d'un coup dans un fichier :

```
10dd
```

Ce n'est pas le sujet de l'article donc je vais arrêter là, mais pour moi il s'agit d'un outil indispensable :-).

- Gedit

IDE :

- Eclipse
- Netbeans
- IntelliJ

Outils en ligne de commande :

- file : permet de connaître les informations d'un fichier,
- wc : permet d'obtenir des statistiques sur un fichier comme par

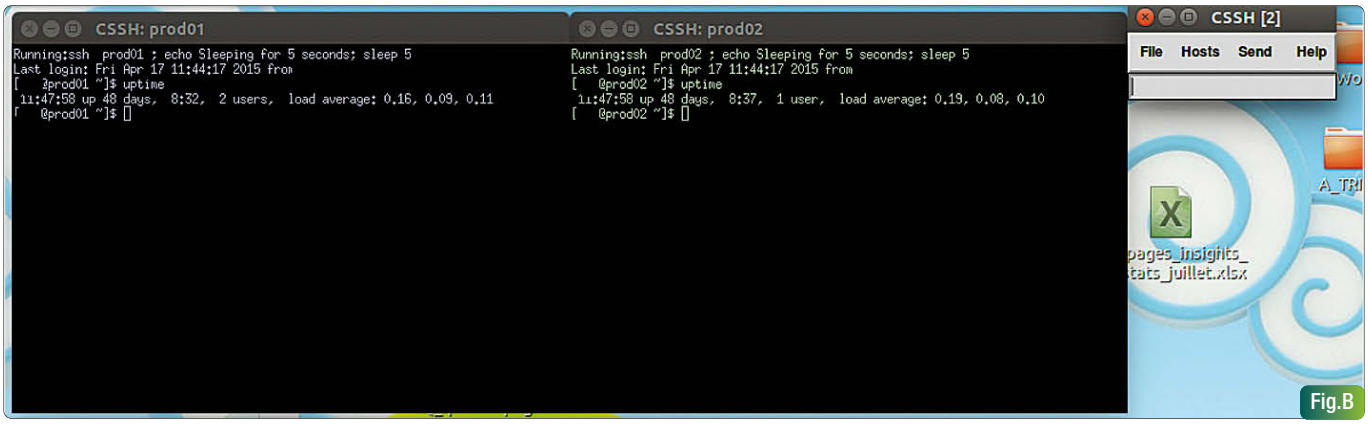


Fig.B

exemple le nombre de lignes, le nombre de mots ...,
- find/grep : pour rechercher,
- split : pour découper un fichier en plusieurs,
- tar/zip/bzip2/gzip : permet de compresser et décompresser un ensemble de fichiers
- sort : permet de trier,
- head/tail/less : permet de lire un fichier, respectivement les premières, les dernières lignes et son contenu entier,
- cat : n'a aucun rapport avec les chats ;) et permet de concaténer plusieurs fichiers dans un seul fichier par exemple,
- cut : permet d'extraire des zones d'un fichier,
- diff : permet de comparer deux fichiers,
- top : permet d'obtenir la liste dynamique des processus classés par occupation mémoire, CPU ...,
- ps : permet d'obtenir la liste statique des processus qui tournent au moment où vous lancez la commande,
- sed : imaginons que l'on veut remplacer une URL par une autre dans plusieurs fichiers d'un site Web par exemple, on peut faire un petit script shell utilisant sed, c'est super simple :

```
$ cat updater.sh
# !/bin/sh
########################################################################
##                                    ##
## Ce script remplace un mot par un autre mot sur plusieurs fichiers ##
##                                    ##
########################################################################

FILES=`grep -rl programmez_185 /home/<your_user>/conf /home/<your_user>/bin
 /home/<your_user>/local/install/*.crontab`
for i in $FILES
do
  echo "replacing programmez_185 by programmez_186 in $i"
  sed -r "s:programmez_185:programmez_186:g" $i > $i.new && mv -f $i.new $i
done
```

- awk : très utile pour les logs Apache par exemple...
- wget : permet de télécharger un fichier d'après son url
- ...

Utilitaires :

- Pour rapidement effectuer des captures écran qui sont stockées dans le presse papier, il faut appuyer en même temps sur les touches CTRL+ALT+Impr écran, puis sélectionner ce que vous voulez capturer. C'est super pratique !
- Libre Office
- Cluster SSH : cet outil permet de lancer des commandes sur plusieurs serveurs en même temps.

```
$ cssh prod01 prod02
```
Fig.B

- Remmina : il s'agit d'un visionneur de bureau distant. C'est pratique pour se connecter à distance sur des serveurs/PC qui sont sous Windows par exemple.
- Wine : permet d'exécuter *certains* programmes Windows sous Ubuntu. C'est un outil pratique pour installer Photoshop par exemple. La liste des logiciels compatible est disponible sur Internet.

Navigateurs :

- Firefox
- Chrome
- Opera
- ...

Graphisme :

- Gimp
- Inkscape

FTP :

- FIleZilla
- Il existe également le plugin FireFTP pour Firefox et on peut se connecter à un serveur en FTP en ligne de commande aussi.

Virtualisation :

Il peut arriver que l'on ait besoin d'utiliser des logiciels tournant uniquement sur Windows, pas de problème, les outils de virtualisation sont faits pour ça, ils permettent d'utiliser Windows dans Linux. Les outils de virtualisation permettent de créer une machine virtuelle et d'y installer n'importe quel système d'exploitation.

- VirtualBox
- VMWare

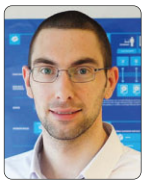
Conclusion

Comme le dit le dicton : "Il y a une application pour tout", il s'agit de la même chose sur un environnement Linux pour le Développement. Et contrairement au mythe, non il ne faut pas recompiler le noyau Linux tous les jours ;-).

```
$ cowsay "J'adore coder sur Linux !"
 ____________________________
< J'adore coder sur Linux ! >
 ----------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Déployer un container Docker contenant ASP.NET 5 sur Microsoft Azure

On entend beaucoup parler de Docker ces derniers temps. En effet, cette nouvelle technologie qui permet de créer des « micro-services » que l'on peut déployer sur différentes plateformes, est très à la mode. Microsoft, quant à lui, a fait de sa plateforme Microsoft Azure une plateforme très ouverte à l'Open Source. Il est d'ailleurs désormais possible de faire tourner des applications ASP.NET 5, jusque-là réservées aux plateformes Windows, sur Linux et OSX.



Mickaël Mottet

Leader technique chez SQLI Enterprise, Mickaël est également reconnu MVP (Most Valuable Professional) sur Azure, la technologie de Cloud Computing de Microsoft. Il anime d'ailleurs depuis plusieurs années la communauté francophone ZeCloud sur celle-ci. Curieux et toujours à l'écoute des dernières tendances, il aime tester les nouvelles solutions techniques et les évangéliser autour de lui.

Voyons comment toutes ces technologies s'utilisent ensemble, en déployant notre premier container Docker contenant une application ASP.NET 5 hébergée dans une VM Ubuntu dans Microsoft Azure.

Prérequis

Ce tutoriel a été réalisé sous Windows (mais il est également réalisable sur Mac ou Linux). Un compte Microsoft Azure est nécessaire pour déployer une machine virtuelle sur la plateforme. Si vous n'en disposez pas encore, vous pouvez vous y inscrire gratuitement pour une période d'essai à cette adresse : http://azure.microsoft.com/fr-fr/

Installation des outils d'administration Microsoft Azure

Nous allons utiliser les outils en ligne de commande qui sont basés sur Node.JS pour effectuer le provisionnement de notre machine virtuelle sur Azure. Le principal avantage de cet outil par rapport à Powershell est qu'il est disponible sur les trois systèmes d'exploitation : Windows, Mac et Linux. Ainsi, si vous n'êtes pas fidèle à un seul système d'exploitation, vous ne serez jamais dépaysé puisque les commandes seront toujours les mêmes.

Les outils en ligne de commande sont téléchargeables à cette adresse : http://azure.microsoft.com/fr-fr/downloads/ **Fig.1.**

Pour ma part, je suis sous Windows, l'installation est effectuée grâce à Web Platform Installer **Fig.2**. Il suffit alors de cliquer sur « Installer » pour que l'outil s'installe sans intervention manuelle. Pour vérifier que l'outil est bien installé, il suffit de lancer une invite de commande et de lancer la commande :

```
azure
```

Si l'outil est correctement installé, vous devez obtenir un résultat identique à celui-ci : **Fig.3**.

Connexion à Microsoft Azure

Nous allons maintenant relier notre compte Azure à notre outil. Pour cela, nous allons télécharger le fichier de publication contenant l'ensemble des informations concernant notre abonnement, puis l'importer. Il faut lancer la commande suivante :

```
azure account download
```

L'utilitaire va ainsi ouvrir un navigateur Internet, nous demander de nous identifier, et nous rediriger vers l'URL :
https://manage.windowsazure.com/publishsettings/index?client=xplat
Un fichier à télécharger va être soumis au téléchargement. Je vous conseille de bien noter où vous l'enregistrez. Pour ma part, je l'ai enregistré à la racine de mon disque D: sous le nom « azure.publishsettings ». Ensuite, nous allons importer le fichier de publication précédemment enregistré en lançant la commande :

```
azure account import D:\azure.publishsettings
```

Fig.4.

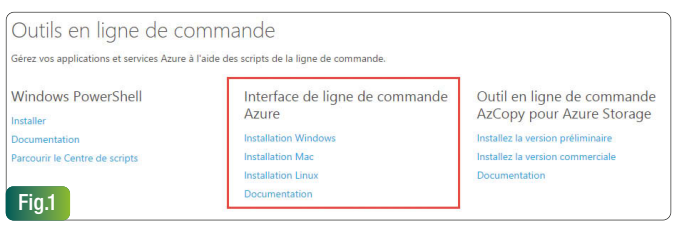


Fig.1

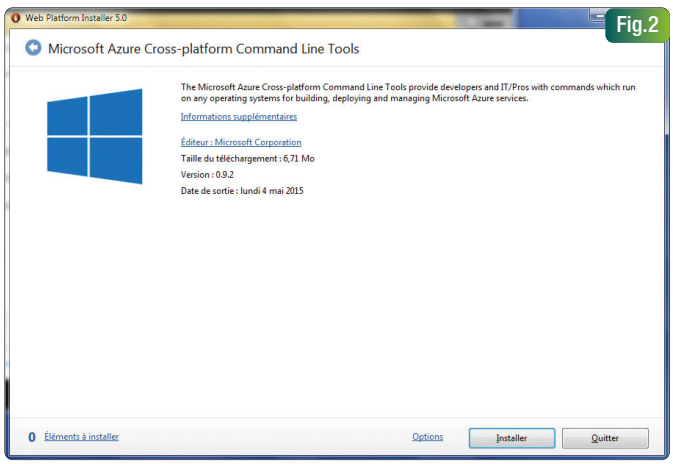


Fig.2

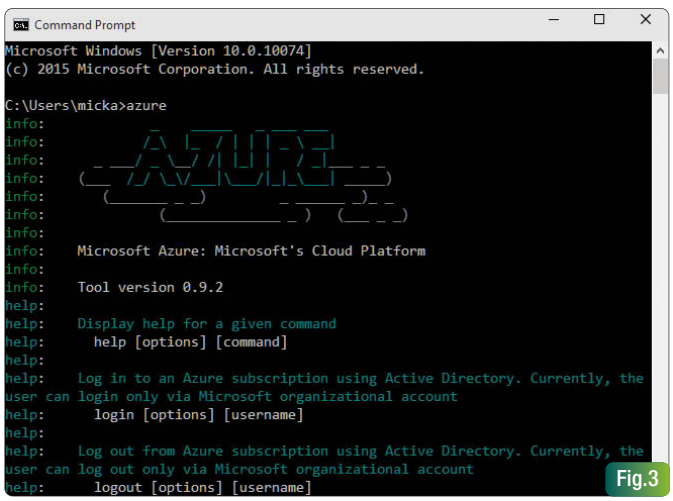


Fig.3

Une fois l'import effectué, vous pouvez vérifier que tout s'est correctement déroulé en lançant la commande suivante, qui va lister l'ensemble de vos abonnements Azure :

```
azure account list
```

Fig.5.

Si votre abonnement Azure est bien listé, vous êtes prêt à déployer votre machine virtuelle Ubuntu.

Déploiement de notre hôte Docker

Un container Docker nécessite d'être déployé sur un hôte disposant du serveur Docker. Dans ce tutorial, nous allons utiliser une distribution Ubuntu car l'image est déjà présente dans la galerie Azure. Nous y installerons ensuite Docker et notre application.
Microsoft Azure propose une galerie d'images de différents systèmes d'exploitation (Windows, Ubuntu, CoreOS, Suse...) ainsi que plusieurs versions. Nous allons tout d'abord choisir une image de Ubuntu.
Pour lister l'ensemble des images disponibles, il faut utiliser la commande suivante :

```
azure vm image list
```

Une longue liste apparaît alors et il est difficile de faire son choix. Pour limiter la sélection aux machines Ubuntu, on peut alors utiliser la commande « grep » sur Linux ou « findstr » sur Windows.

```
azure vm image list | findstr Ubuntu-15_04
```

J'ai ainsi choisi la dernière image Ubuntu disponible à la date de cet article :

```
« b39f27a8b8c64d52b05eac6a62ebad85__Ubuntu-15_04-amd64-server-20150422-en-us-30GB »
```

La commande azure vm create va nous permettre de créer rapidement une machine virtuelle sur laquelle nous pourrons nous connecter par la suite et y installer docker ainsi que notre application. Ainsi, en utilisant l'image que nous avons sélectionnée, il suffit de lancer la commande suivante :

```
azure vm create --ssh 22 --location "West Europe" < vm-CloudServiceName> "b39f27a8b8c64d52b05eac6a62ebad85__Ubuntu-15_04-amd64-server-20150422-en-us-30GB" <username> <password>
```

où

- --ssh 22 : permet d'ouvrir par défaut le port 22 qui va nous permettre d'administrer notre machine avec SSH ;
- --location "West Europe" : datacenter Azure où la VM sera déployée ;
- <vm-CloudServiceName> : DNS du Cloud Service de votre choix ;
- <username> : login de l'utilisateur root créé par défaut sur la VM ;
- <password> : mot de passe de l'utilisateur (au moins 8 caractères avec une minuscule, une majuscule et un caractère spécial comme !@#$%^&+=);

Dans mon cas, j'ai lancé la commande suivante :

```
azure vm create --ssh 22 --location "West Europe" ubuntudocker-mcklmt "b39f27a8b8c64d52b05eac6a62ebad85__Ubuntu-15_04-amd64-server-20150422-en-us-30GB" mcklmt Ubuntu!2015
```

Azure va tout d'abord créer un compte de stockage puis y déposer une copie de notre image Ubuntu. Enfin, une machine virtuelle basée sur cette image de disque va être instanciée.
Si tout se passe correctement, vous devez obtenir un résultat similaire à celui-ci : Fig.6.
La commande suivante va vous permettre de vérifier que votre VM a bien été créée :

```
azure vm list
```

Fig.7.

Connexion à la machine virtuelle en SSH

Nous allons nous connecter à la machine virtuelle en utilisant SSH. Si vous êtes sous Windows, vous pouvez utiliser Putty : http://www.putty.org/
En lançant Putty, saisissez l'adresse de votre Cloud Service que vous venez de créer : Fig.8. En cliquant sur « Open », Putty peut vous demander une confirmation de connexion telle que celle-ci : Fig.9.
Cliquez sur « Yes » pour accepter la connexion.

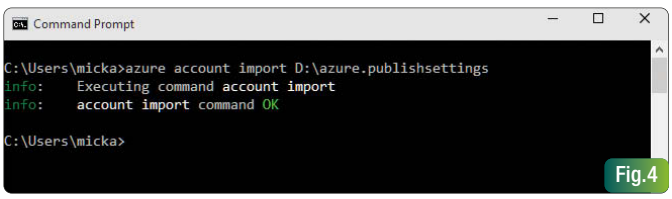


Fig.4

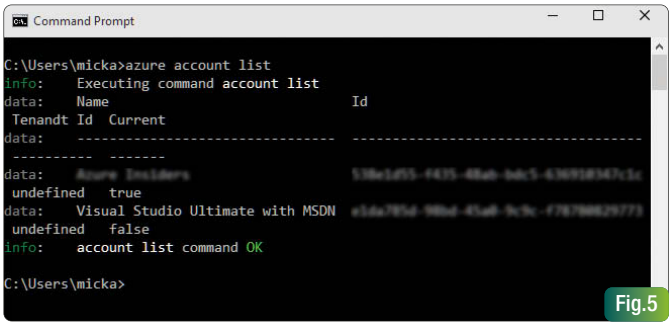


Fig.5

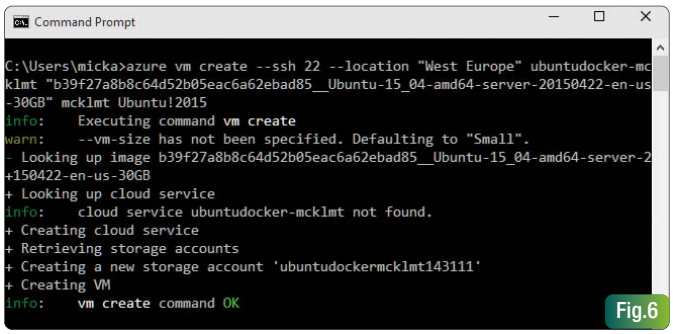


Fig.6

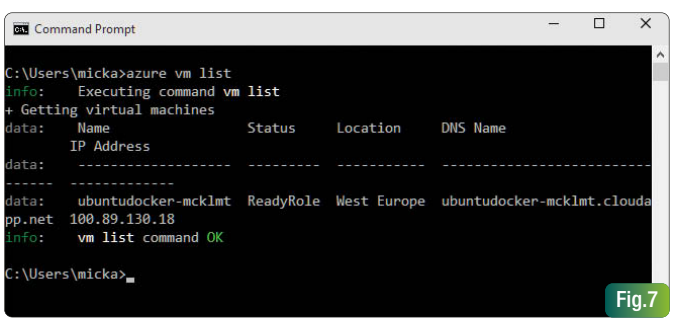


Fig.7

Utilisez ensuite le couple login / mot de passe que vous avez choisi précédemment. Vous voilà connecté ! Fig.10.

Installation de Docker

La commande est assez simple et se résume à :

```
sudo apt-get install docker.io
```

Si une confirmation vous est demandée, cliquez simplement sur "Y". Lorsque la commande est terminée, vous devez obtenir un message similaire à celui-ci : Fig.11.

Création de notre container Docker

Pour créer un container Docker, il est possible de créer un container à partir de rien, ou bien de partir d'une image de base sur laquelle nous allons ajouter notre application. C'est cette seconde possibilité que j'ai retenue.
J'ai donc choisi de déployer une application ASP.NET 5.
Pour récupérer cette application, j'ai choisi de télécharger directement les sources à partir du dépôt officiel du projet sur GitHub.
La commande suivante va permettre de cloner le dépôt officiel dans le répertoire aspnet-Home sur notre machine virtuelle :

```
git clone https://github.com/aspnet/Home.git aspnet-Home
```

Fig.12.
Rendons-nous dans le répertoire de l'application pour lister les fichiers :

```
cd aspnet-Home/samples/latest/HelloWeb
dir
```

Ces commandes doivent vous permettre d'observer le contenu du répertoire suivant : Fig.13.
La création d'un container Docker est régie par la présence d'un fichier « Dockerfile » qui permet de préciser le contenu du container.
Créons simplement ce fichier en lançant la commande :

```
nano Dockerfile
```

Cette commande va permettre de lancer la création du fichier « Dockerfile » et de lancer l'éditeur de texte « nano ».
Ensuite, saisissez le contenu du fichier « Dockerfile » :

```
FROM microsoft/aspnet:1.0.0-beta4

COPY . /app
WORKDIR /app
RUN ["dnu", "restore"]

EXPOSE 5004
ENTRYPOINT ["dnx", ".", "kestrel"]
```

Enfin, validez par les commandes « CTRL + X », la touche « Y » pour enregistrer les modifications et enfin la touche « Entrée » pour valider le nom du fichier Fig.14.

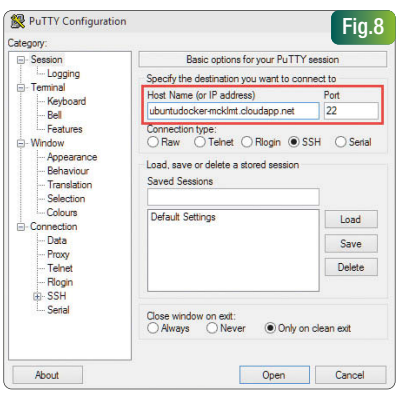
Explorons le contenu de ce fichier Dockerfile, ligne par ligne :
FROM microsoft/aspnet:1.0.0-beta4 : permet de préciser à Docker quelle image est la base de notre container. J'ai volontairement sélectionné une version plutôt que de prendre la dernière car le développement ASP.NET n'est pas encore stabilisé.

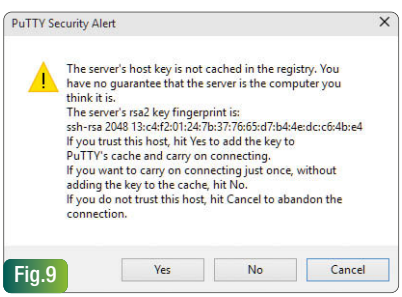
COPY . /app et WORKDIR /app : Docker va copier le contenu de notre application dans le container et changer le répertoire de travail.

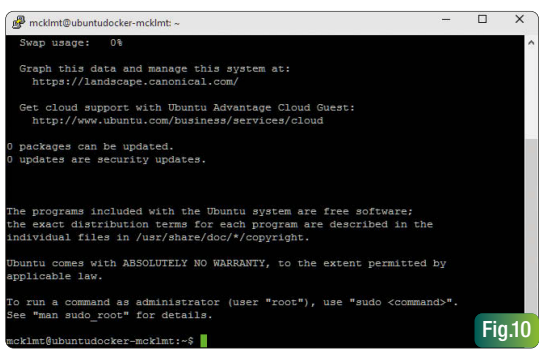
RUN ["knu", "restore"] : Docker va lancer la restauration des dépendances qui sont présentes dans l'application.

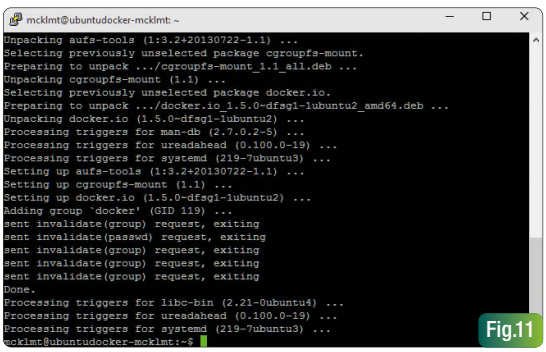
EXPOSE 5004 : Docker va ouvrir le port 5004 afin que l'application puisse écouter les requêtes entrantes.

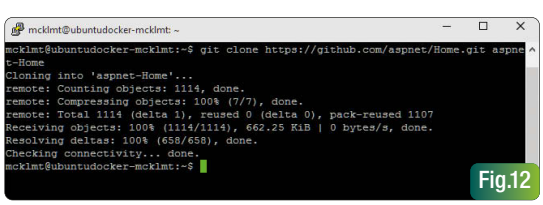
Enfin, la dernière commande ENTRYPOINT ["dnx", ".", "kestrel"] va permettre au serveur ASP.NET de se lancer dans le répertoire de notre application.

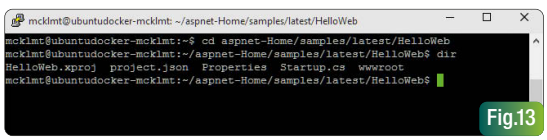


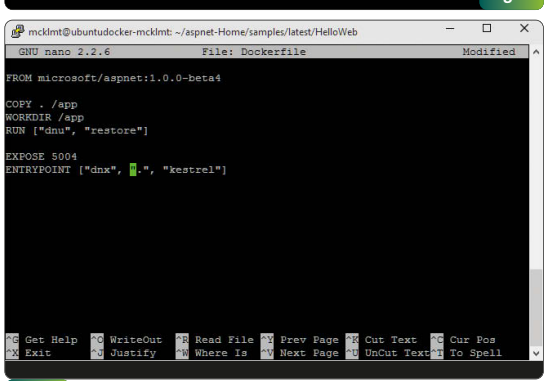












Il nous faut maintenant construire notre container. Rien de plus simple, il nous suffit de lancer la commande :

```
sudo docker build -t mcklmt-app .
```

où mcklmt-app est le nom de mon container.

Si vous obtenez un message similaire à celui-ci : Fig.15.

C'est simplement que le service docker est arrêté sur votre machine. Dans ce cas, il vous suffit de démarrer le service en lançant la commande : sudo service docker start et de relancer la construction de votre container.

Docker va télécharger l'image de base que nous avons choisie, ici, l'image de Microsoft ASP.NET va restaurer les dépendances de notre application et construire le container.
Lorsque Docker a terminé son travail, vous devez apercevoir un message similaire à celui-ci : Fig.16.

La commande sudo docker images va permettre de vérifier que notre image est bien construite : Fig.17.

Lançons maintenant notre container :

```
sudo docker run –t –p 80 :5004 mcklmt-app
```

Fig.18.
La commande sudo docker ps nous permet de vérifier que tout s'est correctement déroulé et que tout fonctionne correctement.
Fig.19.

Exposer notre application sur Internet

Nous avons presque terminé. Notre application est déployée dans notre container Docker. Il faut maintenant ouvrir le port 80 sur lequel notre machine virtuelle attend les requêtes sur le firewall Azure.
Pour cela, revenons sur notre poste de travail où nous allons utiliser les outils en ligne de commande Azure pour créer un point de terminaison sur notre machine virtuelle.
Pour cela, lançons la commande :

```
azure vm endpoint create ubuntudocker-mcklmt 80 80
```

où ubuntudocker-mcklmt est le nom de la machine virtuelle.
Fig.20.

Test depuis le navigateur

Vous pouvez désormais ouvrir votre navigateur à l'adresse de votre Cloud Service et vous devez apercevoir ceci : Fig.21.

Conclusion

Dans ce tutoriel, vous avez appris à manipuler les outils en ligne de commande Microsoft Azure pour administrer et automatiser votre infrastructure dans le Cloud. Si vous souhaitez déployer votre container sur une autre plateforme Cloud comme Amazon ou Google, c'est tout à fait possible en minimisant les actions de déploiement. Il vous suffit de ne redéployer que le container, ce qui est moins lourd que de déplacer une machine virtuelle.
Et pourquoi ne pas aller plus loin et déployer votre container sur un cluster Docker Swarm ?

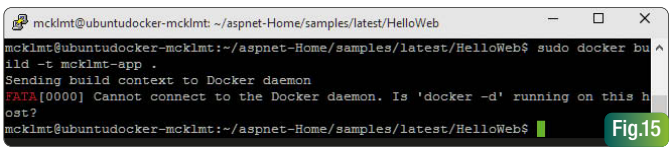


Fig.15

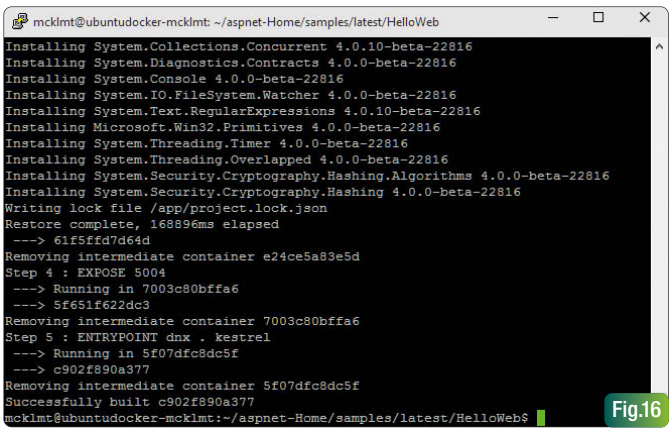


Fig.16

Fig.17

Fig.18

Fig.19

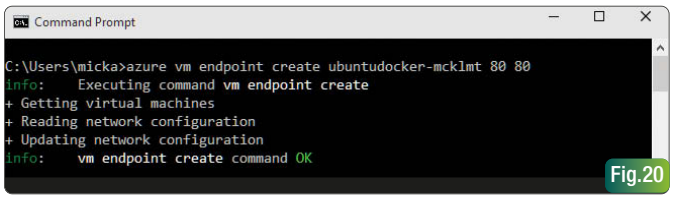


Fig.20

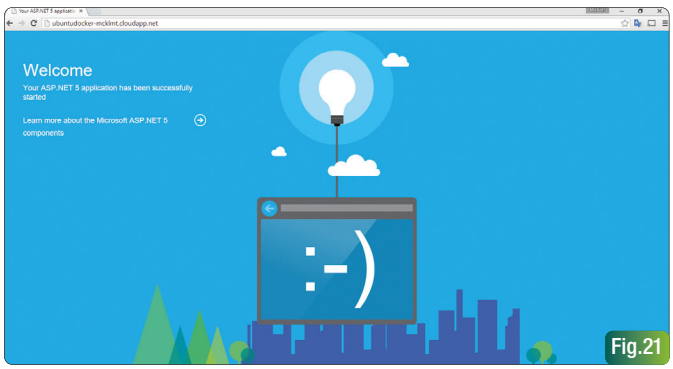


Fig.21

Web Workers : le multithread débarque dans le navigateur !

Proposant une expérience utilisateur toujours plus riche, les applications Web sont écrites en Javascript. Alors que leurs fonctionnalités les rapprochent des applications natives, elles souffrent depuis toujours d'une limite de taille liée au modèle d'exécution monothread du Javascript au sein du navigateur. Ainsi, les opérations lourdes réalisées côté client se traduisent souvent en message d'avertissement du navigateur conseillant de tuer le script en question ce qui est du plus mauvais effet. Avec les Web Workers, HTML 5 embarque une spécification apportant le multithread directement au sein du navigateur. Découverte d'une technologie prometteuse qui va accélérer encore un peu plus les applications Web.

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

Né au milieu des années 90, Javascript est rapidement devenu le compagnon indispensable d'HTML. A cette époque, les ordinateurs étaient dotés de processeurs simple cœur et leur modèle d'exécution, basé sur un thread unique, se prêtait plutôt bien à ses usages. En effet, les sites de l'époque étaient à des années lumière de qui se fait aujourd'hui avec des possibilités restreintes comparées aux applications natives.

Ainsi, l'exécution d'un script n'engage qu'un seul thread au niveau du navigateur. Plus connu sous le nom d'UI Thread, il est également en charge de l'affichage et du rendu. Avec la complexification des applications Web, les traitements Javascript n'ont cessé de s'alourdir conduisant quelques fois à des IHM bloquées. Les navigateurs disposent d'un système de protection permettant d'avertir l'utilisateur lorsque l'exécution d'un script est trop longue lui proposant de le tuer (Fig.1) ce qui se révèle désastreux en termes d'expérience utilisateur.

Néanmoins, jusqu'alors, les scénarios d'utilisation où cela posait de réels problèmes étaient limités, bien que les applications Web avec leurs contenus toujours plus riches aient commencé à être pénalisées. Pour dépasser ces limitations historiques, les développeurs avaient trouvé des parades permettant de simuler des traitements parallèles. Basées sur les fonctions setTimeout / setInterval, l'utilisation de l'objet XMLHttpRequest pour les appels HTTP asynchrones ou encore le système évènementiel du DOM, ces approches étaient non bloquantes mais n'apportaient rien au niveau de l'accélération des traitements. Alors que la norme est aux ordinateurs multi-cœurs et qu'HTML 5 propose des possibilités toujours plus puissantes, la présence du multithread dans Javascript est une nécessité indispensable aujourd'hui et plus encore pour les années à venir.

Un peu de théorie

Afin d'adresser cette problématique, le W3C a intégré au sein d'HTML 5 la spécification Web Worker qui propose une API pour exécuter des scripts Javascript en tâche de fond. Il devient alors possible de créer des threads séparés à partir de l'UI Thread d'une application Web. Pour éviter les problèmes inhérents aux partages de ressources entre threads, tels que les locks ou les races conditions, le cadre d'exécution des Web Workers est une sandbox relativement stricte au niveau des ressources accessibles. Ils ne peuvent ainsi pas accéder aux objets Javascript standards comme document, window, console, parent et également le DOM. Impossible de faire accéder à un élément via un getElementById ou d'utiliser la fonction bloquante alert.

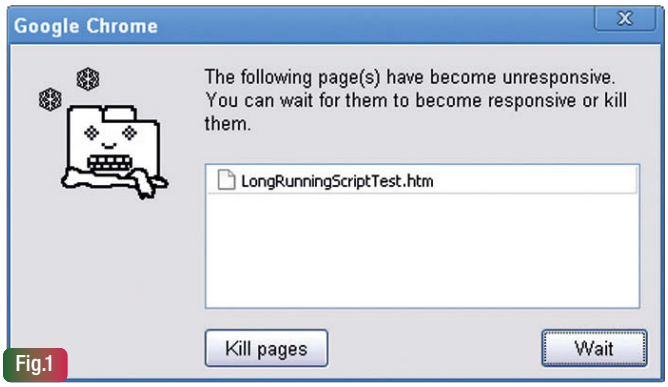


Fig.1

Avertissement script trop long

Ces restrictions risquent de frustrer les développeurs habitués au multithreading sur des environnements natifs, mais elles permettent de préserver les développeurs Web des nombreux écueils du monde parallèle. Malgré ces limitations, les possibilités sont nombreuses et le développeur peut interagir avec les éléments suivants dans un Web Worker :

- L'objet navigator,
- L'objet location en lecture seule,
- La fonction importScripts() pour importer des scripts issus du même domaine,
- Les objets Javascript de base : Object, Array, Date, Math et String,
- L'objet XMLHttpRequest pour les requêtes HTTP ,
- Les fonctions setTimeout et setInterval,
- La base de données Indexed DB,
- L'objet self représentant les objets dans le scope du worker courant.

La spécification introduit 3 types de Web Workers. Le Dedicated Worker correspond à un nouveau thread exécuté en arrière-plan ne bloquant donc pas le travail de l'UI Thread. Associé à ce type, le Sub Worker est un Dedicated Worker un peu particulier puisque créé au sein d'un autre worker. Enfin, on retrouve le Shared Worker utilisable par plusieurs pages via des connexions multiples sur le même domaine d'origine.

Communication et gestion des erreurs

Pour conclure cette introduction théorique aux Web Workers, il est important de parler du mode de communication retenu entre les workers et le thread principal. Implémenté sous la forme de messages, il est réalisé via la fonction postMessage sur un worker depuis l'UI Thread ou directement à l'intérieur du worker pour envoyer un message à l'UI Thread. La fonction postMessage prend en entrée n'importe lequel des objets primitifs du Javascript et également des objets JSON. Au sein du script associé à un

worker, il faut s'abonner à l'évènement message avec un callback appelé lors des passages de messages depuis l'UI Thread. Le travail du thread démarre ainsi au premier appel à la fonction postMessage depuis la page principale.
Une fois un worker terminé, sa fermeture reste à la charge du développeur. Compte tenu de la mémoire prise au sein du navigateur, il est important de réaliser cette opération. Pour cela, on peut appeler la méthode terminate sur l'instance de worker au sein de la page principale ou bien directement clore le worker dans son script via un appel à self.close().
Au niveau de la gestion des erreurs, le worker propose l'évènement error auquel il faudra s'abonner pour récupérer les informations détaillées associées à une erreur du code d'un worker. L'objet ErrorEvent passé au callback de cet évènement propose le nom du worker, le message d'erreur remonté ainsi que la ligne correspondante au sein du script.

Cas d'utilisation

Avec les applications Web actuelles nécessitant des interfaces utilisateurs riches et réactives, les scénarios d'utilisation pour la mise en place des Web Workers sont nombreux. Bien que leurs restrictions réduisent leur portée, on peut citer les cas suivants :

- **Traitement de gros volumes de données**. Remontés via XMLHttpRequest, ces volumes peuvent être traités au sein de workers soulageant par la même l'UI Thread,
- **Traitement d'image**. Les possibilités offertes par le Canvas ou la balise video apportent de nouveaux cas d'utilisation nécessitant une grande puissance de calcul. La parallélisation de ces calculs est une option avantageuse,
- **Traitements lourds**. Que ce soit pour de l'analyse de texte au sein de suites bureautiques en ligne ou bien pour réaliser des calculs financiers poussés sur des sites boursiers, les workers auront leur mot à dire,
- **Utilisation concurrente de la base de données Indexed DB**. Si l'accès aux données du cache local n'est pas autorisé dans les workers, la base de données Indexed DB est totalement opérationnelle,
- **Jeux**. Les moteurs physiques ou bien les IA de jeux HTML 5 sont également d'excellents candidats.

Aussi alléchants que soient ces scénarios, il est bon de garder à l'esprit que la programmation parallèle est une problématique complexe nécessitant du pragmatisme. Ainsi, tous les traitements ne pourront tirer partie des Web Workers. En sus, la création d'un worker est une opération coûteuse qui peut prendre plus de temps que le gain réalisé à l'exécution du traitement via un worker pour des calculs trop modestes. Toujours rayon performance, le coût mémoire supplémentaire induit devra être soigneusement quantifié.

Premier Worker

Pour mettre en œuvre un premier Dedicated Worker, nous allons prendre l'exemple classique de la recherche de nombres premiers. Une telle recherche est un traitement plutôt lourd qui bloquerait l'UI Thread et rendrait donc inutilisable l'IHM pour l'utilisateur. L'utilisation d'un worker est tout à fait à propos pour éviter ce phénomène. Mieux encore, ce traitement se prête parfaitement à la parallélisation puisque chaque recherche de nombre premier est indépendante de la précédente. Nous pouvons donc créer un script pour le worker nommé prime.js prenant en entrée une plage de nombres pour laquelle la primalité sera testée :

```
self.onmessage = function messageHandler(event) {
 var wid = event.data.wid;
 var n = event.data.start;
 var threshold = event.data.end - event.data.start;
 var total = 0;
 var found = 0;

 while (total < threshold) {
  n += 1;
  var max = Math.sqrt(n);
  var ok = true;

  for (var i = 2; i <= max  &&  ok; i += 1) {
   if (n % i == 0) {
    ok = false;
   }
  }

  if (ok) {
   found++;
  }

  total++;

  if (total % event.data.step == 0) {
   this.postMessage({wid : wid, total : total, found : found});
  }
 }
}
```

On s'abonne à l'évènement message en plaçant un callback sur la propriété self.onmessage. Le nombre d'entiers à tester est défini dans la variable threshold en calculant la différence entre les bornes de début et de fin. Enfin, on stocke le nombre d'entiers premiers trouvés et le nombre d'entiers testés. Le paramètre step de l'objet JSON récupéré via event.data.step définit la fréquence à laquelle on doit avertir l'UI Thread de l'avancée de la recherche. La communication des données vers ce thread principal s'effectue via un appel à la fonction postMessage avec en entrée un objet JSON proposant également l'id du worker ayant envoyé le message. Il reste maintenant à créer les différents workers au sein de la page principale en passant en entrée de chacun d'entre eux les plages de valeurs à tester :

```
var width = 0;
var percentage = 0;
var total = 0;
var found = 0;
var t = [];
var f = [];
var sum = function(a, b) { return a + b;};

function messageHandler(event) {
 t[event.data.wid] = event.data.total;
 f[event.data.wid] = event.data.found;
 total = t.reduce(sum);
 found = f.reduce(sum);

 // affichage UI
 var prime = document.getElementById('prime');
 prime.innerHTML = 'Premiers trouvés: ' + found;
 percentage = Math.round(total * 100 / 1000000);
 width = Math.round(310 * percentage / 100);
 var progressbar = document.getElementById('progressbar');
 progressbar.innerHTML = percentage + '%';
```

```
 progressbar.style.width = width + 'px';
}

function processFor(number, workers) {
 var block = number / workers;

 for (var i = 0; i < workers ; i += 1) {
  var worker = new Worker('prime.js');
  worker.onmessage = messageHandler;
  worker.postMessage({wid : i, start : i * block, end : (i + 1) * block, step : block / 10});
 }
}
```

La fonction processFor démarre la recherche d'entiers premiers avec un seuil défini en choisissant le nombre de workers à utiliser. Chaque worker est ensuite créé au sein d'une boucle avant de lui passer en entrée la plage de valeurs. La fréquence de mise à jour est primordiale ici puisqu'une mise à jour à chaque entier testé reviendrait à bloquer l'UI Thread ! Pas à cause du calcul des nombres premiers mais à cause de la mise à jour de l'affichage de l'UI réalisé au sein de la fonction messageHandler. En effet, la recherche d'éléments de la page au sein du DOM pour leur mise à jour est une des opérations les plus coûteuses en Javascript. L'exécution du script sur 1 million d'entiers met en lumière l'efficacité des workers puisque l'UI reste parfaitement réactive (Fig.2).

Traitement d'image

Comme vu précédemment, les Web Workers se prêtent particulièrement bien au traitement d'images dans le navigateur et c'est un usage qui va se répandre avec l'arrivée du composant Canvas et de l'élément video dans HTML 5. Pour réaliser un exemple de ce type, nous partons d'une image dessinée au sein d'un Canvas sur laquelle pourront être effectués des traitements d'image simples tels qu'un effet Sepia ou un effet de niveaux de gris. Travaillant pixel par pixel, ces traitements sont de fait de parfaits candidats à la parallélisation. Il suffit de découper l'image à traiter en sous parties qui seront traitées au sein de workers dédiés.
Un worker n'ayant pas accès à l'élément Canvas, il va être nécessaire de travailler directement sur les données binaires de l'image chargée en découpant l'image en autant de régions que de workers à employer :

```
var canvas = document.getElementById("target");
canvas.width = source.clientWidth;
canvas.height = source.clientHeight;
var tempContext = canvas.getContext("2d");
var len = canvas.width * canvas.height * 4;
// img initiale
tempContext.drawImage(source, 0, 0, canvas.width, canvas.height);
```

Fig.2

Calcul de nombres premiers via Worker

```
// parallélisation
var workersCount = 4;
var finished = 0;
var segmentLength = len / workersCount;
var blockSize = canvas.height / workersCount;

for (var index = 0; index < workersCount; index++) {
 var worker = new Worker("pictureProcessor.js");
 worker.onmessage = onWorkEnded;
 var canvasData = tempContext.getImageData(0, blockSize * index, canvas.width, blockSize);
 worker.postMessage({ data: canvasData, index: index, length: segmentLength });
}
```

Suivant le nombre de workers paramétrés, les différentes régions de l'image à traiter définies et envoyées ensuite à chacun des workers via la fonction postMessage. En entrée de celle-ci, on retrouve ainsi les données binaires de la région de l'image à traiter ainsi que la longueur de cette région. Contenu dans le fichier pictureProcessor.js, le code du worker exécute le traitement d'image souhaité :

```
importScripts("tools.js");

self.onmessage = function (e) {
 var canvasData = e.data.data;
 var binaryData = canvasData.data;
 var l = e.data.length;
 var index = e.data.index;

 processSepia(binaryData, l);
 // processGrayscale(binaryData, l);

 this.postMessage({ result: canvasData, index: index });
};
```

A la réception du message, le worker se met en route appliquant le traitement d'image sur les données passées en entrée. On note ici l'appel de la fonction importScripts pour importer le script tools.js contenant les fonctions de traitement d'image en question. Une fois le traitement exécuté, le worker envoie un message à l'UI Thread contenant les données binaires transformées ainsi que l'index de la région concernée. Au niveau de la page principale, la réception de ce message est faite au sein du callback onWorkEnded :

```
var onWorkEnded = function (e) {
 var canvasData = e.data.result;
 var index = e.data.index;
 // img resultat
 tempContext.putImageData(canvasData, 0, blockSize * index);
 finished++;

 if (finished == workersCount) {
  // fin traitement
 }
};
```

Les données transformées sont affichées au sein du Canvas afin de présenter le résultat de la transformation à l'utilisateur. Un compteur permet de déterminer la fin d'exécution du traitement et de proposer éventuellement un retour à l'utilisateur. Non détaillé ici, le recours à bibliothèque Javascript Modernizr et sa propriété Modernizr.webworkers offre la possi-

bilité de proposer un traitement dégradé sans Web Workers dans le cas où le navigateur cible ne supporterait pas cette nouvelle fonctionnalité HTML 5. Non détaillées ici car n'étant pas l'objet premier de l'article, les fonctions de traitement d'image utilisées au sein de tools.js sont disponibles avec le code source des exemples de cet article. Enfin, l'exécution du script permet de constater que le navigateur utilise bien 4 threads pour traiter en parallèle l'image d'origine (Fig.3).

Shared Worker

Second type de workers, les Shared Workers peuvent être partagés entre pages issues d'un même domaine au sein du navigateur ce qui implique qu'il n'est pas rattaché uniquement au script qui l'a lancé. Ainsi, il peut recevoir plusieurs connexions. Outre cette différence d'implémentation fondamentale, les Shared Workers utilisent également le concept de port à partir duquel il faudra travailler. Le démarrage de l'écoute d'un worker de ce type se fait une fois l'appel à la méthode start de l'objet port réalisé. Fort logiquement, les variables globales définies au sein d'un Shared Worker sont accessibles lors des autres appels de ce worker. Enfin, la communication se fait également via postMessage mais appelé depuis l'objet port. Un exemple simple illustrant le fonctionnement des Shared Workers consiste à mémoriser le nombre de pages s'y connectant et de renvoyer à la page appelante son numéro de connexion :

```
var num = 0;

self.onconnect = function (e) {
 var port = e.ports[0];
 port.postMessage("Connexion  #" + num);
 num++;

 port.onmessage = function (e) {
   // réponse ...
   port.postMessage("Réponse : " + e.data);
 };

 port.start();
};
```

Petite subtilité avec un Shared Worker, il est nécessaire de s'abonner à l'évènement de connexion connect pour ensuite se mettre en attente de réception de messages.
Au niveau de la page appelante, le démarrage du Shared Worker s'effectue aisément avec pour simple précaution de bien utiliser le concept de port lors des manipulations afférentes au worker :

```
var w = new SharedWorker("shared.js");
w.port.onmessage = function(e){
 // msg dans e.data
};

w.port.start();
w.port.postMessage("Bonjour");
```

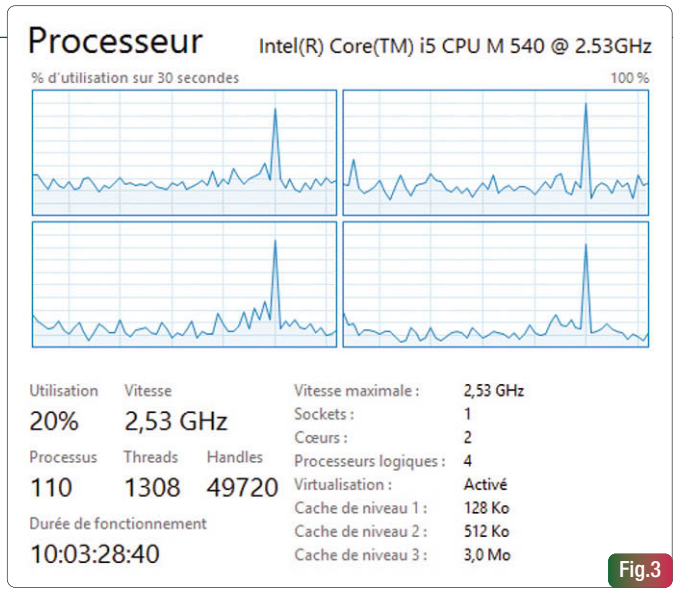


Fig.3

Exécution du script avec 4 Workers

L'exécution du script met en exergue le principe de fonctionnement du Shared Worker puisque le numéro de connexion est bien affiché au sein de la page appelante. L'existence du Shared Worker n'est pas liée à la page ou à l'UI Thread dont il est issu. En fait, il continue à tourner tant qu'au moins une page qui y est connectée reste ouverte. Sa destruction s'effectue donc à la fermeture de la dernière page connectée ou bien lors de l'appel explicite aux fonctions close ou terminate selon que l'on soit au sein du worker ou de l'UI Thread.

Conclusion

Comblant un manque historique de Javascript lié à son modèle d'exécution monothreadé, les Web Workers vont rendre les applications Web encore plus réactives. Les esprits le plus chagrins regretteront les limitations dues à leur exécution au sein d'une sandbox mais c'est un choix d'implémentation logique au vu des écueils induits par la programmation parallèle. Les possibilités demeurent cependant assez importantes pour en tirer profit sur de nombreux cas d'utilisation. La facilité de manipulation de l'API associée aux Web Workers ne doit néanmoins pas perdre de vue que la parallélisation des traitements reste un domaine complexe pas forcément applicable à tous les cas de figure. Enfin, compte tenu de leur temps démarrage et de leur consommation mémoire, il sera nécessaire d'évaluer correctement si les gains apportés par leur emploi se révèlent bien supérieurs à ces temps de fonctionnement.

Débuter avec LabVIEW

Développé en 1986 par Jeff Kodosky, LabVIEW (Laboratory Virtual Instrumentation Engineering Workbench) est un langage de programmation au cœur de la plateforme de développement graphique de National Instruments. Créé à l'origine sur Macintosh, LabVIEW est utilisé principalement pour la mesure par acquisition de données, le contrôle d'instruments et l'automatisme industriel. La plateforme de développement s'exécute sous différents systèmes d'exploitation tels que Windows, Linux et OS X. LabVIEW peut générer du code sur ces systèmes d'exploitation, mais également sur des plateformes temps réel, des systèmes embarqués ou des composants de types FPGA ou DSP propres à NI.

Emmanuel ROSET
Ingénieur Produits
National Instruments
emmanuel.roset@ni.com

Quel est le principe de fonctionnement de LabVIEW ?

LabVIEW est un langage graphique basé sur le principe du flux de données, c'est-à-dire qu'il utilise des fils pour véhiculer les variables du programme. Il est compilé et génère un code exécutable optimisé qui peut être disponible sous forme de DLL ou d'un exécutable autonome .exe allié à un moteur d'exécution (Run-Time). Le cœur du compilateur utilise un moteur multitâche afin d'exécuter rapidement du code en parallèle avec une gestion automatisée des priorités et des affinités d'attribution des cœurs du processeur sur Windows. Le respect des priorités et des attributions de cœurs dépend du système d'exploitation sur lequel le code est déployé (système d'exploitation temps réel ou non, par exemple). L'avantage principal de ce langage réside dans sa nature intuitive et la rapidité à laquelle il permet d'obtenir une application fonctionnelle. Pour ce faire, les déclarations mémoire, une étape incontournable propre à tous les langages, est ici prise en charge par LabVIEW. Les variables sont directement utilisables par le code avec une valeur de type par défaut (modifiable par la suite). De plus, le programmeur utilise des icônes qui représentent des symboles intuitifs des fonctions proposées. Le code graphique est une autre manière de programmer mais le principe reste exactement le même qu'un langage textuel (Fig.1 et Fig.2). Avec un peu de pratique, les couleurs associées au type des fils de données facilitent le débogage graphique. LabVIEW est ouvert à l'intégration de toutes sortes de programmes réalisés par des langages de programmation externes. Il peut encapsuler des .DLL et des .NET, et également incorporer des langages mathématiques textuels par des fenêtres graphiques et des icônes. Cela permet de réutiliser un programme déjà réalisé, et d'éviter de le traduire en langage graphique.

Aux fonctions de structures d'un langage classique sont intégrées des fonctions mathématiques et de traitement du signal ainsi qu'un certain nombre de bibliothèques d'analyse et de conversion de chaînes de caractères. Ces dernières simplifient l'utilisation intensive de la communication avec des instruments, en permettant par exemple d'extraire des données numériques de la réponse textuelle d'un instrument, et de les convertir rapidement pour les positionner dans un graphe.

Que peut-on réaliser avec LabVIEW ?

Il est possible de réaliser toutes sortes de programmes, au même titre qu'un langage textuel générique. Cependant, les fonctions sont très orientées mesures et analyses des données. Les applications sont destinées à en savoir plus sur le comportement d'un système physique par l'utilisation de capteurs, ou à contrôler des paramètres en envoyant des ordres de commande par l'intermédiaire des sorties physiques (ex : piloter des moteurs) ; le but étant de mieux connaître un système pour le mettre au point, l'optimiser ou simplement le surveiller. Malgré tout, de nombreux développeurs utilisent LabVIEW pour des programmes n'ayant aucun rapport avec la mesure ou même l'industrie; notamment pour simuler des algorithmes mathématiques, créer un distributeur de boissons ou contrôler un jeu de flipper mécanique.

Comment installer et démarrer LabVIEW ?

Pour utiliser LabVIEW, un PC avec peu de ressources peut suffire. La configuration requise n'est pas très gourmande même si, pour le confort visuel, il est préférable d'avoir un écran de bonne résolution, le code étant graphique. Pour connaître les ressources : http://www.ni.com/labview/requirements/f/

Toutefois, le code est encapsulé par des sous-programmes sous forme d'icônes, et il est recommandé de ne pas créer de diagrammes de code sur plus d'une largeur d'écran.

Une fois les DVD installés, une fenêtre d'accueil apparaît et propose de créer un projet (Fig.3). Comme pour tout logiciel il est intéressant, pour commencer, de voir un programme qui fonctionne déjà pour l'analyser, plutôt que de créer des éléments sans avoir de guide. Pour ce faire, il suffit d'aller dans le menu *Aide » Recherche d'exemples...* Le menu qui apparaît propose plusieurs choix en fonction de ce que l'on souhaite apprendre ou approfondir (Fig.4). Il suffit donc de se laisser guider. La plupart des utilisateurs ouvrent les codes du menu *Analyse, traitement du signal et mathématiques* ou du menu *Fondamentaux*, qui proposent des exemples fonctionnels liés à la structure du langage et à chaque fonction. Pour lancer les codes sans connaissances

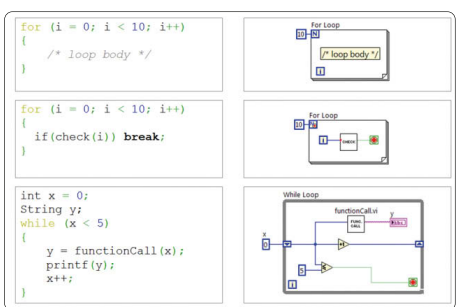


Fig.1

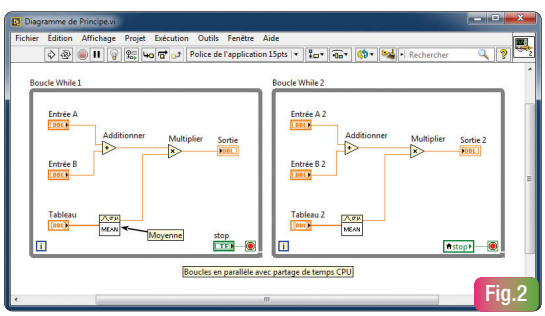


Fig.2

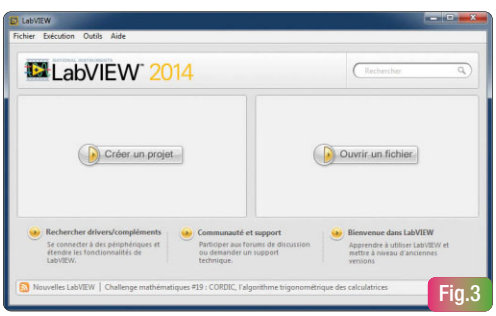


Fig.3

particulières, il suffit de cliquer sur la flèche en haut à gauche qui affiche le mot *Exécuter*, dans la fenêtre sur fond gris qui contient les objets interactifs. Cliquez sur Stop dans un programme ou sur l'icône *Abandonner l'exécution* de la barre d'outils à coté du bouton *Exécuter* (Fig.5). Il faut savoir que la compilation est la plupart du temps quasi instantanée. Ceci est dû au fait que la compilation s'effectue au fur et à mesure, à chaque étape de la création de fils de données.

Comment écrire un code à partir d'une page blanche ?

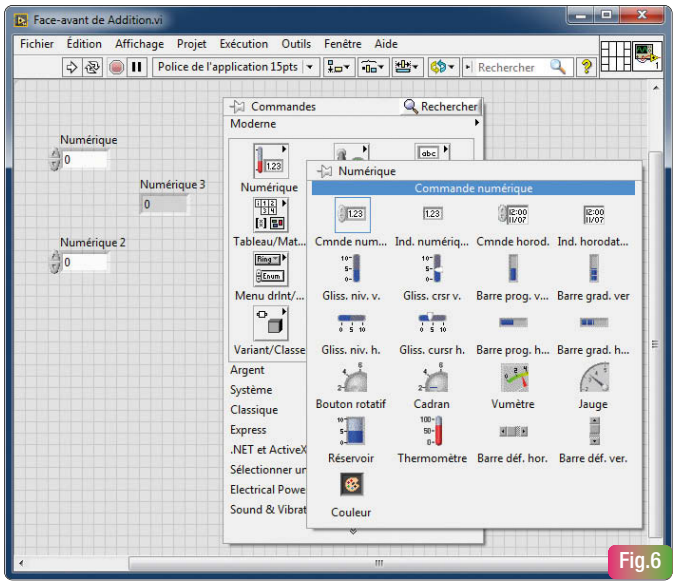
Fermez toutes les fenêtres d'exemples ouvertes et cliquez sur *Fichier » Nouveau VI*
Le principe est d'utiliser d'une part une fenêtre de face avant sur laquelle on vient placer des objets d'interface graphique (qui sert ensuite d'interface homme-machine principale), et d'autre part une fenêtre de diagramme dans laquelle le code graphique prend place.
Par un simple clic avec le bouton droit de la souris, il est possible de faire apparaître la fenêtre de sélection des objets graphique de la face avant, ou la palette des fonctions sur le diagramme.
Pour un premier programme, il est préférable de commencer par la manipulation de quelques fonctions simples afin de prendre en main les outils du langage. Depuis la palette de la face avant, prenez deux entrées (nommées commandes) de format numérique et une sortie (nommée indicateur). Laissez les noms des variables (nommés étiquettes) par défaut ; il est évidemment possible, voire recommandé, de les modifier en utilisant un nom de variable pertinent (Fig.6 et Fig.7).
Dans la fenêtre du diagramme apparaissent des icônes orange qui sont reliées aux données saisies dans la face avant. Passez dans la fenêtre blanche (le diagramme) et placez une fonction d'addition en la sélectionnant dans la palette *Programmation » Numérique » Additionner*. Ensuite, rapprochez-vous avec la souris des entrées et sorties et reliez les variables avec l'outil bobine qui se présente automatiquement. Il ne reste plus qu'à exécuter le code comme précédemment. Les fonctions ne se limitent pas à celles de la palette. En ouvrant la palette *Connectivité*, on constate que l'on peut lancer des exécutables externes, utiliser du code .NET et ActiveX, utiliser des Services Web, encapsuler des DLL et bien d'autres ouvertures.
Après quelques essais, il devient évident que l'on ne peut pas tout créer en une seule interface de taille infinie en résolution. La solution consiste à créer des sous-programmes en reliant les entrées/sorties du code à l'icône en haut à droite de la fenêtre, puis de les placer dans la palette de fonctions, au besoin. Le VI s'organise alors de manière hiérarchique, comme présenté dans le menu *Affichage » Hiérarchie du VI*.
Tout le cœur du langage est là. Le reste est une suite de fonctions intuitives à étudier au travers d'exemples de programmes.

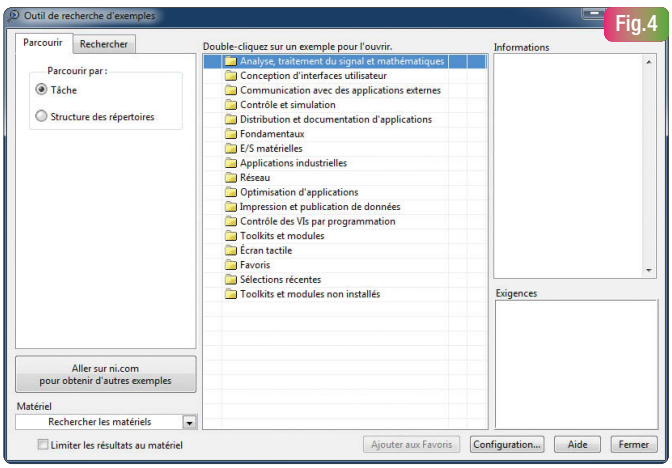
Et après ? Comment créer des exécutables autonomes ou distribuer le code ?

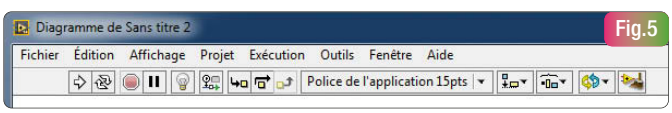
Pour créer des exécutables, il est nécessaire d'avoir la version professionnelle ou d'obtenir l'outil Application Builder. Dans une fenêtre plus haut niveau appelée Fenêtre de projet, placez tous les composants et les sources nécessaires, puis utilisez l'une des options de génération de *Spécification de construction*. Ce qui peut être généré est soit un exécutable lié au système d'exploitation sur lequel l'ordinateur fonctionne, des .NET ou des DLL, entre autres, soit des installeurs et des distributions de code sources.
Pour la mise au point de programmes plus conséquents, il existe des outils logiciels complémentaires permettant de tester du code automatiquement et en équipe.

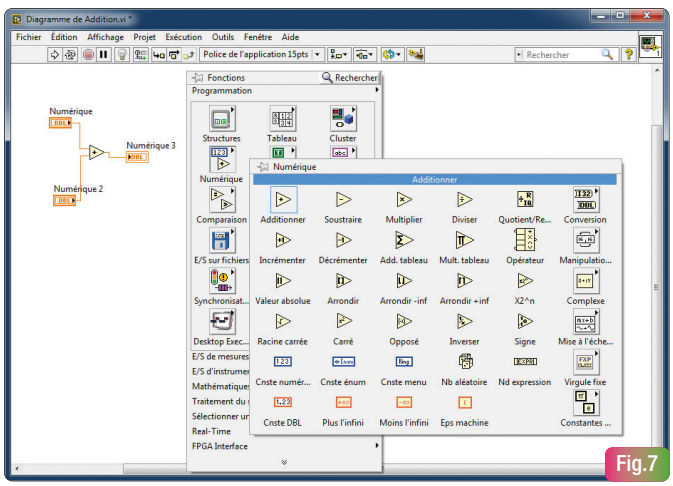
Conclusion

Nous avons vu les concepts principaux afin de mieux comprendre le fonctionnement et les principes de base de LabVIEW. Vous pouvez installer une version d'évaluation, pour vous faire une petite idée des menus de base. Vous constaterez par vous-même que ce langage n'a rien de complexe, et qu'il permet de réaliser des applications fonctionnelles rapidement et intuitivement.





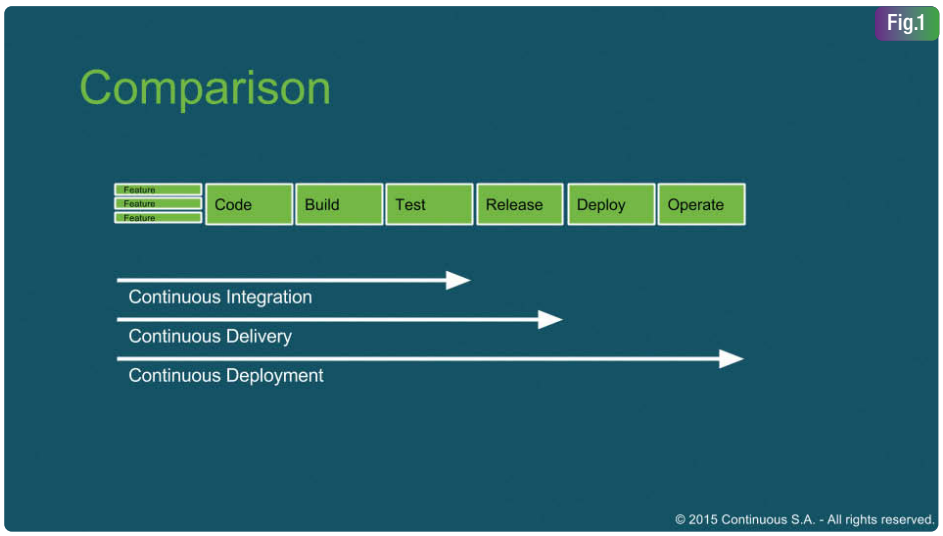




Industrialisation PHP en continu

L'intégration continue est une pratique plutôt développée pour de nombreux langages de programmation, mais assez récente dans l'écosystème PHP. L'intérêt de mettre en place une solution de livraison continue est de délivrer rapidement et régulièrement de nouvelles fonctionnalités sans faire régresser l'application. Pour y parvenir, cette pratique permet de réduire les tâches manuelles, d'exécuter automatiquement des tests, et de construire des builds.

Frédéric Dewinne
CTO de continuousphp, est un expert du Continuous Delivery/Deployment spécifique à PHP.



De nombreuses solutions existent pour mettre en place l'intégration continue, mais très peu sont pensées pour PHP.
La plupart sont multi-langages et ne prennent pas en compte les besoins spécifiques des développeurs PHP. Dans cet article qui vous présentera toutes les choses à savoir pour mettre en place facilement l'intégration continue pour vos projets, nous utiliserons continuousphp pour illustrer notre propos.

Les pratiques

Il existe plusieurs pratiques, dont l'intégration continue, pour vous aider à améliorer vos processus d'assurance qualité. Prenons le temps de faire un point sur celles-ci.

Intégration Continue (CI)

L'intégration continue est une pratique qui consiste à merger continuellement les branches des développeurs dans une branche commune. Il s'agit là d'une partie de la méthodologie Extreme Programming (XP), une méthode agile plus particulièrement orientée sur l'aspect réalisation d'une application. La mise en pratique de l'intégration continue demande de construire la base de code et de la tester à chaque commit, afin de prévenir toute régression liée aux problèmes d'intégration. Cette version de la base de code peut ensuite être automatiquement déployée sur un serveur d'intégration afin d'y opérer des tests manuels complémentaires.

Livraison Continue (CD)

La livraison continue est la suite logique de l'intégration continue. Tout ce qui s'applique à cette dernière vaut également pour cette pratique. A noter qu'on y trouve une notion supplémentaire de package. Chaque commit va déclencher un workflow (le build) qui va construire la base de code, la tester et créer un package. Ce package devient le livrable de la nouvelle version du code. Chaque commit peut être livré à n'importe quel moment.

Déploiement Continu

Le déploiement continu met en pratique le workflow le plus extrême. Tout en appliquant les principes de la livraison continue, chaque build stable est déployé dans un environnement de production. De cette manière, toute nouvelle fonctionnalité terminée et fonctionnelle sera mise à disposition des utilisateurs finaux le plus rapidement possible. Il s'agit de la pratique permettant d'accélérer l'ajout de Business Value en production. Une très bonne assurance qualité est primordiale pour mettre en œuvre cette pratique Fig.1.

Mise en route

Dans notre exemple, nous allons nous pencher sur le cas du déploiement continu et reprendre les différentes étapes à mettre en place.

Le build

La première chose à prendre en considération est la gestion des dépendances logicielles. Quelles dépendances sont nécessaires aux tests et quelles sont celles qui sont nécessaires au bon fonctionnement de l'application ? Inutile de s'encombrer de frameworks de test sur un serveur de production. En revanche, il faut s'assurer que les bibliothèques tierces utilisées par l'application seront bien disponibles dans la bonne version. Pour ce faire, Composer est votre meilleur ami.
Ensuite, vient la question de l'automatisation. Plusieurs possibilités s'offrent à vous, mais nous devons garder à l'esprit que les 2 cas de figures (environnement de test et environnement de production) doivent être couverts dans un même build. Si vous utilisez déjà continuousphp, Composer est supporté de façon complètement automatique et créera 2 packages ; l'un pour le test avec les dépendances de développement, l'autre pour le déploiement sans ces dépendances additionnelles. Sinon il vous faudra gérer Composer au même titre que les autres dépendances de configuration traitées ci-après.
Les dépendances logicielles sont donc maintenant prises en charge. Mais le travail ne s'arrête pas là ! Dans certains cas, nous devrons créer/modifier des fichiers de configuration, compiler des fichiers statiques (CSS, JavaScript...). Pour ce faire, nous pourrons utiliser Phing, un outil d'automatisation de tâches, qui a le grand avantage d'être écrit en PHP... pour PHP. Il nous sera aisé d'étendre Phing si des tâches additionnelles devaient s'avérer nécessaires, même si un grand nombre de tâches existe déjà nativement Fig.2.

Les tests

On ne pouvait pas parler de déploiement continu sans aborder les tests ! Pour ce qui est des tests unitaires, différents frameworks existent mais le plus utilisé reste PHPUnit. Comme les tests unitaires ne nécessitent pas l'usage de base de données ni d'autres dépendances systèmes, il

est facile de les mettre en œuvre dans une chaîne d'intégration continue. Nous pouvons donc laisser le serveur de build prendre le relai pour son exécution.

La partie tests fonctionnels sera plus compliquée à mettre en place. Comme les tests fonctionnels peuvent faire appel à certaines dépendances systèmes, il sera alors nécessaire de provisionner ces dépendances en vue d'exécuter les tests.

Dans ce cas, Phing est de nouveau un outil intéressant qui nous permettra de gérer facilement cette étape nécessaire à l'exécution des tests. Si nous devons provisionner la base de données, il nous faudra scinder cette étape en deux sous-étapes. La première pour créer/mettre à jour le schéma tout au long de la vie de l'application via des outils de migration (Doctrine Migration, Phinx, DBDeploy...), et la seconde pour y insérer des fixtures (données de test).

Pour écrire ces tests, Behat reste un outil incontournable car c'est le seul qui permet de réaliser efficacement du Behavior Driven Development (BDD, développement conduit par les spécifications) en PHP.

La notation Gherkin, utilisée pour l'écriture des tests, rend ceux-ci tout de suite plus accessibles. Ils peuvent donc servir de base de spécifications aux différentes fonctionnalités.

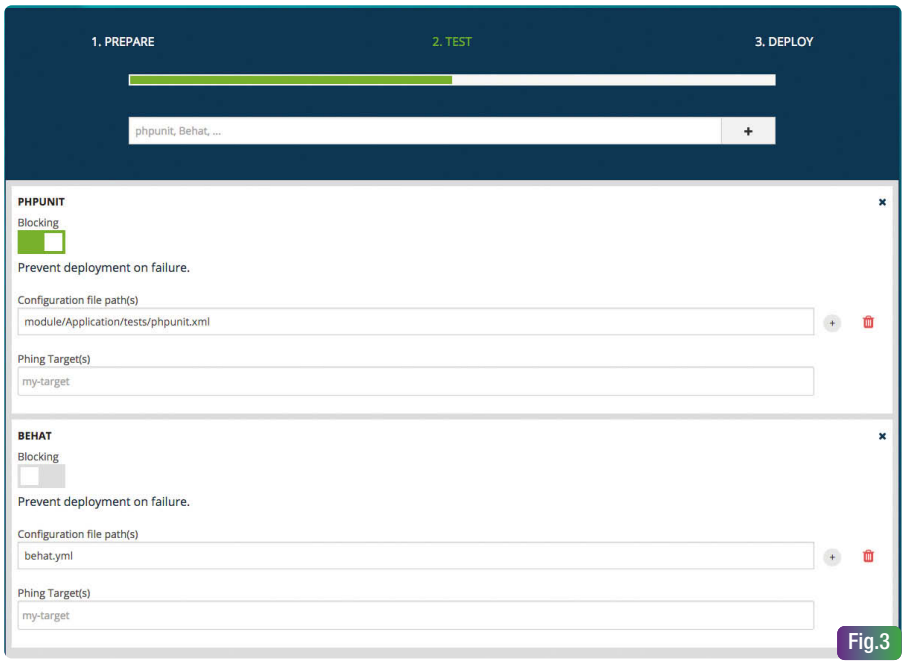


Fig.3

(Configuration d'un pipeline de déploiement dans continuousphp)

Le déploiement

Cette étape est la finalité de toute application. La rendre fonctionnelle et disponible sur un serveur applicatif. Ainsi, la création d'un package reste indispensable si l'on veut pouvoir exécuter des déploiements et/ou un rollback d'une version précise de l'application et de ses dépendances. Ce package peut aussi bien servir au déploiement d'une nouvelle version sur des serveurs existants, qu'au déploiement de la version actuelle sur de nouveaux serveurs (dans un environnement Cloud par exemple), ou d'une version antérieure si un rollback est nécessaire. Au final le package n'est en général qu'un fichier compressé (.zip, .tar.gz...) contenant l'intégralité de l'application et ses scripts d'installation/migration. Une fois ce package disponible, il nous faudra l'installer sur le(s) serveur(s) applicatif(s). Il existe plusieurs outils de déploiement travaillant avec des packages.

L'un des plus complets dans le monde PHP est sans doute Zend Server mais d'autres outils, tel que le récent Amazon CodeDeploy, sont également très prometteurs.

De même, il est possible de gérer ces déploiements applicatifs via des outils de provisioning (Chef, Puppet, Ansible...).

Le pipeline de déploiement

L'étape suivante consiste à utiliser une plateforme de build (Jenkins, continuousphp...) afin de mettre toutes ces étapes bout à bout et obtenir un pipeline de déploiement qui déclenchera un build après chaque commit. En créant différents pipelines selon les branches/tags à l'origine des builds, nous pourrons facilement travailler de manière agile et multiplier les environnements à la volée. Il sera cependant nécessaire d'utiliser une plateforme vous permettant d'exécuter plusieurs builds en parallèle, afin de ne pas être freiné par de longs temps d'attente après chaque commit Fig.3.

Pour conclure cet article, nous pouvons considérer que la mise en place de pipeline de déploiement n'est pas beaucoup plus compliquée qu'une chaîne d'intégration continue, mais est beaucoup plus flexible car elle gère toutes les étapes du développement d'un projet. De plus, elle permet aux développeurs de mieux s'impliquer dans la partie déploiement et appréhender ainsi les méthodes DevOps. Enfin, nous pourrons par ce biais délivrer de la valeur ajoutée à l'application pour l'utilisateur final beaucoup plus rapidement, tout en diminuant drastiquement les périodes d'indisponibilité.

```
<project name="my-project" default="provision" basedir=".">
    <property file="./build.properties"/>

    <target name="compass">
        <if>
            <isset property="env"/>
            <then>
                <if>
                    <contains string="dist" substring="${env}" casesensitive="true"/>
                    <then>
                        <exec command="${compass} compile -e production" passthru="true" dir="."/>
                    </then>
                    <else>
                        <exec command="${compass} compile" passthru="true" dir="."/>
                    </else>
                </if>
            </then>
            <else>
                <exec command="${compass} compile" passthru="true" dir="."/>
            </else>
        </if>
    </target>

    <target description="Setup Application Dependencies" name="provision" depends="compass"/>
</project>
```
Fig.2

Lectures complémentaires :

Tutoriel continuousphp :
http://bitly.com/tutocontinuousphp

Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation (Addison-Wesley Signature Series - Martin Fowler) - Jez Humble et Al., 2010

The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win – Gim Kim et Al., 2014

Une API de streaming en 3 clics

Streamdata.io permet de transformer n'importe quelle API JSON en un flux de données (streaming API) sans aucun développement serveur.

@StreamdataIO
community@streamdata.io

STREAMDATA.IO

Le streaming permet de transmettre aux devices (smartphones, tablettes, browser Web ou objets connectés) tout changement de données source sans que le device n'ait besoin de « poller » la source de données directement : le device n'a plus besoin de lancer régulièrement des requêtes afin de mettre à jour les données. Les utilisateurs de bénéficient également d'un cache optimisé, qui factorise les requêtes vers une même API. Avec streamdata.io, ce n'est plus chaque device qui doit aller poller l'API, mais c'est streamdata.io lui-même qui le fait une seule fois pour tous les utilisateurs de la même API, ce qui permet de limiter la charge sur les systèmes d'information, et ainsi, de supporter des millions d'utilisateurs. Il possède un algorithme qui ne renvoie que le delta des données afin d'envoyer uniquement les données qui ont changé et non pas l'ensemble des données. Cela économise la bande passante du flux, et donc la vitesse de présentation des données sur les mobiles et objets connectés. Et cela en restant dans les standards présents et futurs du Web: HTTP et HTTP/2 (spdy), ce qui permet de réutiliser les infrastructures Web existantes pour mobiles, Web et objets connectés avec, de plus, une grande facilité de mise en œuvre grâce au Cloud.

Ce service est disponible en SaaS pour permettre une intégration simple par les développeurs. Il est ouvert aux éditeurs d'API, pour fournir à leurs développeurs clients des APIs performantes. Le service est aussi directement utilisable par les développeurs consommateurs d'APIs pour intégrer des données dynamiques dans leurs interfaces utilisateur. Jusqu'à présent, seules les applications mobiles à destination des traders intégraient des données dynamiques : les prix des actions qui évoluent en direct sans avoir besoin de rafraîchir sa page. Aujourd'hui, les données dynamiques sont dans toutes les nouvelles UX : les statuts des réseaux sociaux, les news des journaux, les stocks de promotions ponctuelles, le temps d'attente d'un taxi, d'un train, etc. Afin de garder son consommateur sur sa page ou de lui proposer le plus vite possible un article/un service disponible en quantité très faible, ... l'UX doit être dynamique. Or, le web a été construit sur une architecture stateless: chaque demande d'information est indépendante l'une de l'autre. Pour afficher une donnée en temps réel, un polling de la source de données doit être effectué depuis chaque device. Mais cette architecture présente des limites d'échelle et de performance pour des millions de devices avec les solutions de contournement de type long polling, et une croissance forte des devices. Les géants du Web s'y préparent en publiant des API de streaming pour l'accès à leur propre service, avec en premier lieu Twitter, qui en a (https://dev.twitter.com/streaming/overview). Streamdata.io ouvre cette possibilité à tout fournisseur ou consommateur d'API.

Streaming d'une API financière

Prenons une source de données financières JSON : xignite.com, avec par exemple l'API XigniteGlobalCurrencies (https://www.xignite.com/forex/developers). Elle peut être appelée avec par exemple trois taux de changes sur une monnaie, par exemple le BAM (http://www.xe.com/symbols.php donne une liste exhaustive), sur l'URL suivante : Fig.1.

Résultat de la commande curl sur cette URL : Fig.2.

La même source peut être appelée au travers du proxy streamdata.io en ajoutant simplement un préfixe, c'est à dire à l'URL suivante : Fig.3.

Le résultat obtenu est un stream dynamique : le client reçoit d'abord le premier jeu de données (snapshot), puis les incréments en JSON-Patch (http://jsonpatch.com/). On vient de transformer un API standard en API streaming incrémental sans code côté publication d'API ! Cette même donnée peut désormais être publiée sur des millions de devices grâce au cache. Maintenant que vous pouvez envoyer votre donnée dynamique vers votre audience sans surcharger le backend grâce au proxy cache streamdata.io, vous pouvez vous demander si la technologie diff incluse apporte de l'accélération. Pour cela, saisissez l'URL dans notre site Web (http://streamdata.io/), qui vous donne un aperçu de la quantité de données échangées économisées.

streamdata.io: comment l'intégrer ?

Les étapes ci-dessus vous montrent comment faire un test simple du résultat obtenu en appelant votre API au travers de streamdata.io. Pour l'intégrer dans votre application, un SDK Javascript est disponible sur GitHub (), avec une démo live en AngularJS sur Codepen.io (http://codepen.io/streamdataio/pen/RNXvdm). Pour exécuter cette démo, et utiliser le SDK Javascript dans votre application, vous aurez besoin de déclarer des clés de sécurités qui vous seront fournies dès que vous aurez créé un compte (gratuit) sur https://portal.streamdata.io Streamdata.io est gratuit jusqu'à 10 utilisateurs simultanés et 1 million de messages par mois. Au-delà, le prix est de 10€ par million de messages. Un message correspond à l'envoi des données initiales et aux patches successifs vers les devices. N'hésitez pas à nous contacter : sales@streamdata.io.

```
https://globalcurrencies.xignite.com/xGlobalCurrencies.
json/GetRealTimeRate?Symbol=BAMAUD,BAMAZN,BAMBSD&[YOUR_TOKEN]
```

Fig.1


```
data: [{"Outcome":"Success","Message":null,"Identity":"Request","
BaseCurrency":"BAM","QuoteCurrency":"AUD","Symbol":"BAMAUD","Date":"
04/15/2015","Time":"8:46:40 AM","QuoteType":"Calculated","Bid":0.715676,"
Mid":0.715905,"Ask":0.716135,"Spread":4.58747E-4,"Text":"1 Bosnia and
Herzegovina convertible mark = 0.715905 Australian dollar","Source":"Rate
calculated from AUD:BAM"},{"Outcome":"Success","Message":null,"Identity":
null,"BaseCurrency":"BAM","QuoteCurrency":"AZN","Symbol":"BAMAZN","Date":"
04/15/2015","Time":"8:45:00AM","QuoteType":"Calculated","Bid":0.567176,"
Mid":0.570778,"Ask":0.57438,"Spread":0.00720407,"Text":"1 Bosnia and
Herzegovina convertible mark = 0.570778 Azerbaijanian manat","Source":"
Rates calculated by crossing via USD"}
```


Fig.2

```
https://proxy.streamdata.io/https://globalcurrencies.xignite.
com/xGlobalCurrencies.json/GetRealTimeRate?Symbol=BAMAUD,BAMAZN,BAMBSD&
[YOUR_TOKEN]
```

Fig.3

Coder sa première API avec Python. 2e partie

Dans le n° 185 nous avions développé notre premier service Web exposant une API orientée RESTful à l'aide de Python. Dans cette partie nous traiterons les problématiques non-fonctionnelles.

Aurélien Moreau
Co-fondateur d'Angus.ai – Human Perception for Machines.
Blog : http://www.angus.ai/blog/

Fig.1

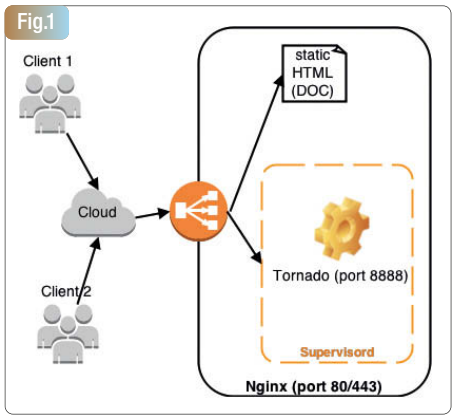
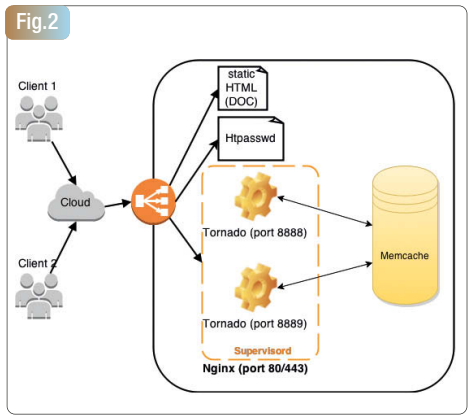


Fig.2



Dans la première partie de ce tutoriel nous avions proposé de développer votre premier service Web exposant une API orientée RESTful à l'aide de Python. Nous nous étions efforcés de rendre un service d'optimisation de sac à dos avec le minimum de configuration possible. Nous allons voir dans cette seconde partie comment répondre à des problématiques non fonctionnelles telles que la sécurité, la performance et la scalabilité. Notre précédent service répond en HTTP à toutes les requêtes sans discrimination du client. La première chose que nous allons mettre en place est un filtre à l'entrée pour garantir que les clients interagissant avec le service sont connus. De plus ce service ne se lançait que sur un seul cœur et stockait les résultats en mémoire. Nous allons montrer comment exploiter convenablement l'ensemble des cœurs du système afin de maximiser le nombre de requêtes parallèles que le service peut supporter sur une seule machine. Cela nous amènera à utiliser un serveur de cache qui nous permettra par la suite d'augmenter le nombre de machines soutenant le service et ainsi effectuer un passage à l'échelle. Nous repartons du service précédent d'optimisation du sac à dos. Il nous permettra de tester le gain en puissance de nos solutions, le service d'optimisation étant assez gourmand en calcul lorsqu'on augmente la taille du problème. La figure suivante montre les modifications que nous allons effectuer pour répondre à ces problématiques non fonctionnelles : Fig.1 devient Fig.2.

SECURITE

Nous disposons de plusieurs possibilités pour sécuriser une API. Le choix doit se faire en fonction du type de client et du type de flux qui sera mis en place au sein des processus métier complets. Avant de mettre en place une solution complexe du type OAuth2, nous proposons ici de commencer avec une authentification très simple basée sur le standard HTTP1.1 : la « Basic Authentication. ».

Chiffrement

Avant tout nous devons chiffrer les communications entre le client et le serveur, les paquets pouvant être interceptés et les informations de connexion facilement récupérables. Nous allons utiliser le reverse-proxy *nginx*. Pour les besoins de la démonstration, nous allons configurer les certificats auto-signés qui sont livrés avec votre distribution. Ici pour une *Ubuntu* 14.04 :

```
server {
    listen 443;
    ssl on;
    ssl_certificate /etc/ssl/certs/ssl-cert-snakeoil.pem;
    ssl_certificate_key  /etc/ssl/private/ssl-cert-snakeoil.key;
(...)
}
```

Authentification Basique

Le principe est le suivant : le client doit ajouter un header HTTP à ses requêtes avec la clé Authorization. La valeur doit commencer par Basic puis être suivie de l'encodage Base64 de la chaîne <username>:<password>.
Par exemple pour le login toto et le mot de passe totopass il faut rajouter à toutes les requêtes le header suivant :

```
Authorization: Basic dG90bzp0b3RvcGFzcw==
```

Cette procédure est grandement simplifiée par la bibliothèque requests que nous avons précédemment utilisée dans notre client de test. Il suffit de construire un objet HTTPBasicAuth et de le passer en argument aux méthodes. L'appel à la création d'un problème sur notre service devient donc :

```
auth=requests.auth.HTTPBasicAuth('toto', 'totopass')
result = requests.post("https://localhost/api/problems", data=json.dumps(data), auth=auth)
```

De même, la récupération d'un résultat devient :

```
auth=requests.auth.HTTPBasicAuth('toto', 'totopass')
result = requests.get(link, auth=auth)
```

Notre client étant prêt à s'authentifier, nous allons mettre en place la stratégie du côté serveur. Nous avions utilisé un reverse proxy (*nginx*) pour absorber les requêtes, nous allons continuer à l'utiliser pour immédiatement rejeter les clients inconnus. *Nginx* offre la routine auth_basic_user_file pour ce faire. Il faut alors indiquer le chemin d'un fichier sous le format htpasswd. Ce fichier peut être généré online (http://www.htaccesstools.com/htpasswd-generator/) ou en utilisant divers utilitaires. Pour notre unique utilisateur (toto) le contenu du fichier est le suivant :

```
toto:$apr1$kQYKDynq$TU092QcnJJinhHyBPmzoX1
```

Il suffit de rajouter alors les routines dans le fichier nginx, sans oublier de recharger la configuration :

```
auth_basic "You need a access token to use these API";
auth_basic_user_file /tmp/htpasswd;
```

A partir de là le serveur va rejeter tout client qui ne s'authentifie pas convenablement.

Contrôle d'accès

A ce stade, seul un client enregistré (ici toto) peut accéder à la plateforme, mais cela n'empêche pas d'aller récupérer les résultats des autres utilisateurs. Ce contrôle d'accès à la ressource doit se faire au niveau « métier » du serveur. Ici nous allons imposer qu'un client ne puisse lire que les résultats produits par ses problèmes. Dans le service Python nous allons tout d'abord rajouter une fonction d'extraction du client qui récupère le login du user :

```
def extract_user(handler):
  auth_header = handler.request.headers.get('Authorization')
  auth_header = base64.decodestring(auth_header[6:])
  user = auth_header.split(':', 2)[0]
  return user
```

Ensuite nous allons rajouter à la ressource le login du user qui l'a créé lors de la définition du problème :

```
  def post(self):
    data = json.loads(self.request.body)
(...)
    user = extract_user(self)
    result["user"] = user
(...)
```

Ainsi nous sommes capables de vérifier que le demandeur d'une ressource est bien son créateur lors d'un GET :

```
  def get(self, prob_id):
    result = self.storage[prob_id]
    user = extract_user(self)
    if result["user"] == user:
      self.write(json.dumps(result))
    else:
      self.set_status(401)
      self.finish("You have no permission to access to this resource")
```

Conclusion sur la sécurité

Nous avons vu comment sécuriser le canal de communication client/serveur en utilisant HTTPS, puis nous avons mis en place un système d'authentification permettant d'assurer que le client soit connu et qu'il est bien celui qu'il prétend être grâce à la méthode « Basic Authentication » de l'HTTP. Enfin nous avons commencé à implémenter un système de contrôle d'accès permettant de vérifier que la ressource est visible par l'utilisateur. Par la suite, nous allons nous attaquer aux problématiques de performance.

PERFORMANCE

Le problème du sac à dos peut devenir gourmand en temps de calcul lorsque la taille du problème croît (nombre d'objets disponibles et poids maximal). Avec la mise en œuvre actuelle, le service ne peut prendre qu'une seule requête à la fois. Pendant le traitement, le service ne peut pas répondre à d'autres requêtes. Cela est un point bloquant si nous voulons offrir ce service à plusieurs clients simultanément, les taux de disponibilités risquant d'être très médiocres. Les machines (qu'elles soient virtuelles ou physiques) disposent très souvent de plusieurs cœurs que nous pouvons utiliser pour augmenter les capacités du service. Il est habituel d'utiliser plusieurs threads pour paralléliser les calculs. Seulement dans le cas de Python, les restrictions du GIL (Global Interpreter Lock) font qu'il n'est pas possible d'utiliser les threads pour paralléliser le calcul. Il faut donc utiliser plusieurs processus afin de bénéficier au maximum des capacités machines. L'avantage de cette approche est qu'elle permet de préparer le terrain futur de l'utilisation de plusieurs machines en relâchant la contrainte de la mémoire partagée.

Accès aux ressources interprocessus

Afin de simplifier la première partie du tutoriel nous avions stocké les résultats des calculs en mémoire. En revanche, dans le cadre d'un service supporté par plusieurs processus, les espaces mémoire sont distincts. Il faudrait donc garantir que la récupération (GET) d'un résultat utilise le même processus qu'il a créé (POST). Cette contrainte est très forte et difficile à gérer sans établir des sessions qui vont à l'encontre des principes « stateless » de l'approche RESTful. De plus, garder en mémoire les résultats n'est pas pérenne. Qu'arrive-t-il si le processus redémarre ? L'ensemble des résultats sont perdus. Nous allons donc rajouter une couche de cache permettant de stocker temporairement les données et de les récupérer quel que soit le processus. Nous allons utiliser pour ceci *memcache*, un serveur de cache très populaire avec de nombreuses bibliothèques clientes pour Python. Nous partons du principe qu'un serveur *memcache* tourne sur votre serveur et écoute sur le port par défaut (11211).

```
> pip install pymemcache
```

Pour éviter de modifier le code déjà écrit nous allons substituer le simple dictionnaire `storage` initial par une classe faisant appel à un client *memcache* :

```
from pymemcache.client import Client
class Storage(object):
  def __init__(self):
    self.client = Client(('localhost', 11211))

  def __getitem__(self, key):
    return json.loads(self.client.get(key))

  def __setitem__(self, key, data):
    self.client.set(key, json.dumps(data))
storage = Storage()
```

Nous avions donné un identifiant unique par résultat en incrémentant un compteur (plus exactement en regardant le nombre de ressources stockées) :

```
prob_id = len(storage)
```

Il nous faut globaliser l'identifiant, le plus simple étant d'utiliser un uuid. Ce dernier garantit un identifiant unique :

```
import uuid
(...)
class ProblemsHandler(tornado.web.RequestHandler):
(...)
  def post(self):
    data = json.loads(self.request.body)
    prob_id = str(uuid.uuid1())
(...)
```

Le reste du service est inchangé. Dorénavant, l'ensemble des résultats de calculs sont déposés sur le serveur *memcache* et les données disparaîtront en cas de redémarrage de ce serveur. Si nous voulons garantir une vie encore plus longue pour ces ressources il conviendra de les stocker à terme dans une base de données. Nous ne verrons pas cette partie dans ce tutoriel.

Lancer plusieurs processus.

Maintenant que nous avons partagé les données avec tous les processus, il reste à en lancer plusieurs, chacun écoutant sur un port différent.
Nous allons modifier dans un premier temps le lancement de la boucle *tornado* en allant chercher le port à écouter sur la ligne de commande :

```
if __name__ == "__main__":
  if len(sys.argv) > 1:
    port = int(sys.argv[1])
  else:
    port = 8888
  application.listen(port)
```

```
tornado.ioloop.IOLoop.instance().start()
```

Cela nous permet de changer la configuration de Supervisord de la manière suivante :

```
(...)
[program:knapsack-api]
command=/home/amoreau/Projects/Programmez/env/bin/python application.py %(process_num)s
process_name=knapsack%(process_num)s
numprocs=2
numprocs_start=8888
```

Ce qui permet de lancer deux processus, respectivement de nom knapsack8888 et knapsack8889 écoutant sur les ports 8888 et 8889.
Une fois les modifications faites, ne pas oublier de relancer Supervisord :

```
> supervisorctl -c supervisor.conf reload
```

Nginx est un très bon distributeur de charge (loadbalancer). Cela veut dire qu'il est à même de ventiler les requêtes sur un certain nombre de « backend » c'est à dire des url cibles. Nous allons utiliser cette fonctionnalité pour utiliser nos deux processus et les cacher derrière une unique adresse. De cette manière les différents processus lancés sont invisibles pour le client. Nous allons définir un nouveau upstream pour *nginx*, qui correspond à la liste des urls cibles sur lesquelles ventiler les requêtes:

```
upstream backend {
   server 127.0.0.1:8888 max_fails=3 fail_timeout=1s;
   server 127.0.0.1:8889 max_fails=3 fail_timeout=1s;
}
```

Pour l'instant le reverse-proxy s'exécute sur la même machine que les processus du service, nous utilisons donc l'adresse 127.0.0.1. Puis nous utilisons cette nouvelle définition « backend » comme adresse cible pour la procédure « proxy_pass » :

```
server {
(...)
    location ~ ^/api(.*)$ {
        proxy_pass http://backend/api$1$is_args$args;
    }
}
```

Ne pas oublier de relancer le serveur :

```
> sudo service nginx reload
```

Dorénavant, toutes les requêtes sont ventilées sur les deux processus écoutant sur le port 8888 et 8889.

Vérification

Pour vérifier le bon fonctionnement de l'ensemble de la chaîne de traitement nous allons stresser le service et voir comment il réagit. Pour cela nous allons utiliser l'outil ab pour apache benchmark, qui permet de générer un grand nombre de requêtes et de mesurer les résultats. Générons une description de problème assez conséquente pour que le temps CPU soit significatif. Un petit programme Python suffit :

```
import json
import random
objects = [{"name": "o%s"%i, "value": random.randint(1, 10), "weight": random.randint
(1, 20)} for i in range(500)]
sack = { "max-weight": 1000 }
print json.dumps({"objects": objects, "sack": sack}, indent=2)
```

On génère ainsi 500 objets aléatoires pour un poids maximum de 1000. Si nous enregistrons ce résultat dans un fichier data.json, nous pouvons alors stresser le serveur de la manière suivante :

```
> ab -n 30 -c 2 -p data.json -A toto:totopass https://localhost/api/problems
```

Cela correspond à 30 requêtes avec une concurrence de 2 clients simultanés, en utilisant le fichier que nous venons de générer et en s'authentifiant.
La figure suivante fait varier la concurrence et le nombre de processus lancés par Supervisord pour voir l'impact sur les performances en requêtes par seconde. Comme le montre la Fig.3, le contrat est rempli, nous utilisons pleinement les possibilités de la machine sur laquelle tourne le service.

Aller plus loin sur la performance

Pour augmenter les performances et éventuellement la robustesse, il est possible alors de distribuer les processus, le reverse-proxy, ainsi que le serveur de cache sur des machines différentes. Les modifications apportées le permettent sans problème. Il faut s'assurer de remplacer dans tous les fichiers de configuration le localhost par la machine effective.
La figure suivante montre une architecture possible : Fig.4.

CONCLUSION

Après avoir déposé les bases fonctionnelles du service Web, nous nous sommes concentrés sur deux points essentiels de la qualité de vos APIs à savoir la sécurité et les performances. Nous avons vu comment mettre en place une stratégie basique d'authentification ainsi qu'un contrôle d'accès aux ressources simples. Puis nous avons rendu notre programme compatible avec une approche distribuée permettant de lancer plusieurs processus sur une ou plusieurs machines, permettant ainsi d'augmenter la capacité du service à prendre en charge plusieurs clients simultanément.

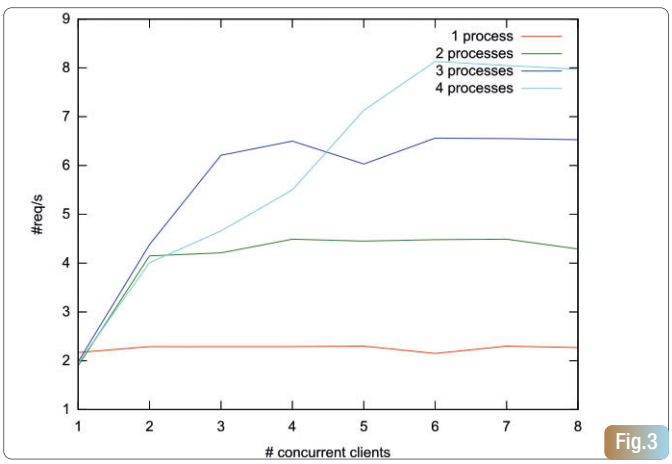


Fig.3

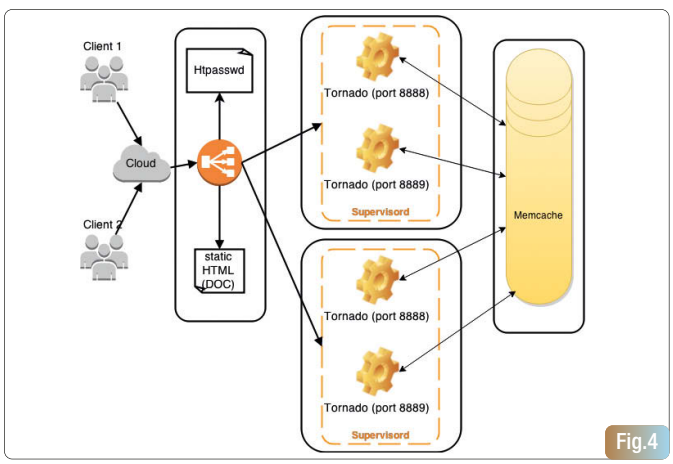


Fig.4

Apple Watch : développer avec le WatchKit

Comme tous les ans, Apple a tenu en Septembre 2014 une conférence lors de laquelle de nombreuses annonces ont été faites. L'iPhone 6 et l'iPhone 6 Plus ont fait leur apparition au catalogue et Apple Pay veut changer la manière de régler ses achats en utilisant son smartphone.

Stéphane Cordonnier
http://blog.beecomedigital.com

Mais hormis si l'on écoute attentivement les rumeurs, la véritable bombe de cette conférence a été l'annonce de l'Apple Watch. Ce nouveau périphérique dans l'écosystème Apple permet à la firme de Cupertino de faire son entrée sur le segment des objets connectés grâce à la commercialisation d'une montre.

Outre le fait de donner l'heure, c'est tout l'environnement iOS qui est mis en avant autour de cette montre. Celle-ci sert en effet d'extension aux applications iOS existantes en leur permettant de pouvoir ajouter des fonctionnalités directement utilisables sans avoir à sortir son téléphone de sa poche.

Se pose donc logiquement la question de savoir comment faire pour développer une application à destination de l'Apple Watch. C'est ce que nous vous proposons de découvrir au travers de cet article de découverte du développement sur la 1ère montre Apple.

Qu'est-ce qu'une application Apple Watch ?

Comme nous l'avons évoqué ci-dessus, la première chose à retenir quand on regarde ce qu'est une application Apple Watch, c'est qu'il s'agit obligatoirement d'une extension pour une application iOS existante. Et qui dit application iOS existante dit nécessité de posséder un iPhone auquel sera couplé la montre.

En effet, pour des raisons qui sont certainement aussi bien techniques que marketing, il n'est pas possible pour l'heure de développer des applications totalement autonomes, c'est à dire qui s'exécuteraient sur l'Apple Watch sans avoir recours à un iPhone (nous reviendrons là dessous plus tard).

Cela signifie que pour développer une application à destination de la montre, vous devez déjà développer une application à destination de l'iPhone, et étendre celle-ci afin de la doter d'un module permettant d'afficher des données sur votre montre.

Ceci est extrêmement important car cela va dicter la manière de concevoir et d'exposer les fonctionnalités d'une application entre des choses visibles sur l'iPhone, d'autres sur la Watch et d'autres sur les deux à la fois.

Un SDK dédié aux applications Apple Watch

Comme Apple nous en a donné l'habitude au fil des mises à jours d'iOS, de nombreux SDK (UIKit, StoreKit, HealthKit...) sont mis à disposition des développeurs au fur et à mesure des mises à jour. La nouvelle née ne déroge pas à cette règle et un nouveau SDK est donc proposé aux développeurs.

WatchKit est donc le SDK dédié au développement d'applications à destination de l'Apple Watch. Comme tous les SDK proposés par Apple, celui-ci propose tous les éléments (interface graphique, contrôleurs, notifications...) dont les développeurs auront besoin pour réaliser les applications qui s'exécuteront sur la montre.

Bien que dans la pratique beaucoup d'éléments de l'écosystème d'Apple tels que Foundation, soient utilisés pour développer des applications à destination de la Watch, ce nouveau SDK engendre des différences fondamentales pour les développeurs habitués à réaliser des applications iOS.

Des différences notables entre les applications iOS et WatchKit

Parmi les différences importantes que l'on notera entre le développement iOS et le développement WatchKit, commençons par parler des éléments permettant de mettre en œuvre une interface graphique.

Là où sur iOS nous avons l'habitude d'utiliser UIKit pour manipuler des images, des boutons, des listes de données, des cartes, etc... WatchKit arrive avec son propre jeu d'éléments graphiques, beaucoup moins riche tant en nombre qu'en possibilités de personnalisation.

La liste des éléments d'interface graphique pouvant être affichés sur une montre est réduite à sa plus simple expression à savoir :

Conteneur (WKInterfaceGroup)

- Table (WKInterfaceTable)
- Image (WKInterfaceImage)
- Separateur (WKInterfaceSeparator)
- Bouton (WKInterfaceButton)
- Switch (WKInterfaceSwitch)
- Slider (WKInterfaceSlider)
- Label (WKInterfaceLabel)
- Date (WKInterfaceDate)
- Timer (WKInterfaceTimer)
- Carte (WKInterfaceMap)
- Menu
- MenuItem

Cette liste réduite et différente des habituels contrôles utilisés sur iOS (UIButton, UIImage, UILabel...) déroutera les développeurs au premier abord car l'utilisation et les possibilités de personnalisation de ces contrôles sont différentes de leur équivalent UIKit.

Il en est de même pour ce qui est de l'écriture de la logique de l'application et du découpage via le traditionnel pattern MVC (Model View Controller). Même si ce pattern est également utilisé au sein de WatchKit, une nouvelle fois la mise en œuvre diffère par rapport à UIKit.

Oubliez les habituels UIViewController car vous devrez désormais utiliser leur équivalent WatchKit à savoir les WKInterfaceController. Comme pour les contrôles d'interface graphique, leur utilisation et leurs possibilités de personnalisation diffèrent grandement par rapport à ce que l'on a l'habitude de faire sur iOS, même si la philosophie reste identique.

Pour donner un dernier exemple des différences fondamentales entre iOS et WatchKit, attardons nous sur l'écriture du code, notamment lorsqu'il s'agit de manipuler les éléments présents à l'écran.

Même si cela a un peu changé depuis l'apparition des storyboard et de la fonction « Auto-Layout » permettant de construire plus simplement ses interfaces graphiques, afin que celles-ci s'adaptent aux différentes tailles d'écran des périphériques iOS, les développeurs ont encore souvent recours au fait de dessiner ou changer la position des contrôles visibles à l'écran via du code. Sur WatchKit, oubliez cette possibilité car il n'est pas

permis (sauf cas très particulier) de manipuler par code la hiérarchie des éléments graphiques au sein d'une vue, ou bien encore de surcharger les méthodes des vues pour en gérer soit même la mise en forme.
Pour gérer la hiérarchie des éléments affichés dans l'application, tout doit être fait en amont au moment de la conception. Lors de l'exécution de l'application, vous ne pouvez plus changer ce qui aura été fait au préalable dans Xcode, au sein d'un storyboard.

Architecture d'une application WatchKit

Comme nous l'avons évoqué précédemment, une application WatchKit est obligatoirement une extension d'une application iOS existante. Cela signifie que l'architecture de l'application sur iOS est composée au minimum de deux parties qui s'exécuteront sur l'iPhone :

- L'application iOS principale
- Une extension d'application

Le but de l'extension d'application est d'embarquer tous les éléments qui seront utilisés par l'application WatchKit et notamment la logique de l'application (le code Objective-C / Swift) et des ressources éventuelles (images, sons...).
Mais une application WatchKit est également composée d'une partie qui sera installée sur la montre. Cette partie n'embarque, et c'est la chose importante à retenir, que les interfaces graphiques (construites dans un storyboard au sein de Xcode) et les éventuelles ressources utilisées par celles-ci (images, sons...).
Mais si la partie installée sur l'Apple Watch n'embarque pas de code, comment l'application sait-elle ce qu'elle doit exécuter vous demanderez-vous ? C'est là qu'entre en jeu WatchKit car c'est lui qui va faire le lien entre la montre et l'iPhone.
En effet, lorsque vous installez une application sur l'Apple Watch, la logique de l'application est exécutée sur l'iPhone. Seuls les résultats de l'exécution sont affichés sur la montre. WatchKit se charge d'échanger les données entre les deux périphériques via la connexion Bluetooth établie lors de l'opération d'appairage au paramétrage initial de la montre. L'architecture d'une application WatchKit ressemble donc à ceci : **Fig.1.**

Créer et tester une application WatchKit dans Xcode

Pour rendre une application iPhone compatible avec la Watch, Apple fournit bien évidemment de l'aide aux développeurs et a mis à jour Xcode (version 6.2 minimum) afin de proposer de nouveaux modèles de projets, permettant de paramétrer tous les éléments nécessaires afin de démarrer au mieux le plus rapidement possible.
Pour ce faire, commencez par ouvrir le projet de son application existante au sein de Xcode. Une fois le projet ouvert, il suffit d'ajouter une nouvelle cible au sein de celui-ci (File > New > Target).
Apparaît alors la boîte de dialogue habituelle qui vous invite à choisir un type de projet pour votre nouvelle cible. Comme vous pouvez le noter dans la capture ci-dessous, une nouvelle section intitulée Apple Watch est désormais présente **Fig.2**.
En sélectionnant le type de projet WatchKit App, Xcode opèrera toutes les modifications nécessaires dans votre projet afin de :

- Créer une extension pour votre application iPhone existante,
- Créer une nouvelle application WatchKit,
- Créer un profil (scheme) pour compiler et tester l'application.

Pour tester ensuite l'application WatchKit, rien de plus simple. Sélectionnez le nouveau profil créé puis sélectionnez Product > Run afin de lancer l'application. Le simulateur iPhone va se lancer comme à l'accoutumée mais il se pourrait que vous ne soyez pas en mesure de voir l'application WatchKit.
Afin de pouvoir visualiser le rendu de l'application WatchKit, il faut aller dans le menu du simulateur et sélectionner l'option Hardware > External Displays > Apple Watch - 38mm (ou 42mm).

Concevoir l'interface graphique via un storyboard

Comme nous l'avons vu dans l'architecture de l'application, la partie installée sur la Watch comprend un storyboard, celui-ci ayant été créé par l'assistant Xcode utilisé à l'étape de création d'une nouvelle cible. Ce storyboard comme tous les éléments de ce type depuis quelques années, permet de concevoir l'interface graphique de l'application et l'enchaînement entre les écrans. Pour cela vous avez à disposition un éditeur visuel permettant de déposer les contrôleurs (1 contrôleur = 1 écran), les composants affichés par chaque contrôleur (boutons, images, labels...) et de définir les propriétés de chacun de ces éléments.
Pour avoir une idée de comment réaliser une application WatchKit, Apple a mis à disposition deux exemples sur le portail réservé aux développeurs. L'exemple nommé « WatchKit Catalog » permet de pouvoir tester l'intégralité des composants d'interfaces disponibles et de voir comment construire une première application simple (enchaînement d'écran, notifications, animations d'images...).
Vous pouvez apercevoir ci-dessous un exemple du storyboard de cette application qui montre une partie des composants graphiques disponibles sur Apple Watch **Fig.3**.
Un point important à noter dans la conception d'une interface graphique pour WatchKit concerne la position des composants au sein du contrôleur où, là encore, une différence notable existe par rapport à iOS. Vous n'avez pas la possibilité de positionner les composants comme vous le souhaitez au sein d'un contrôleur. Ceux-ci doivent obligatoirement être positionnés comme si vous conceviez un tableau

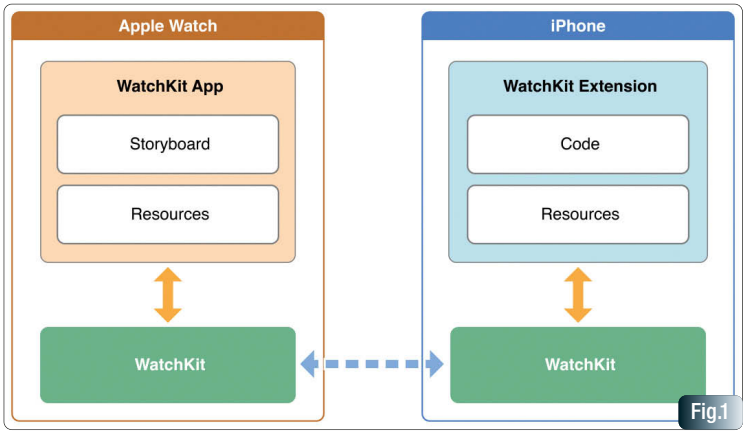


Fig.1

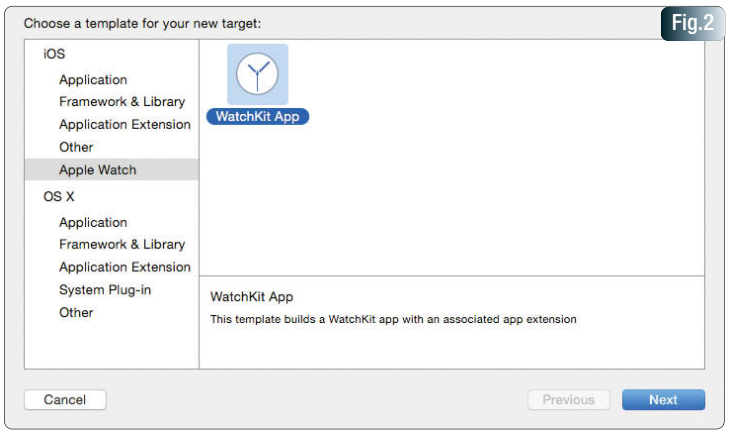


Fig.2

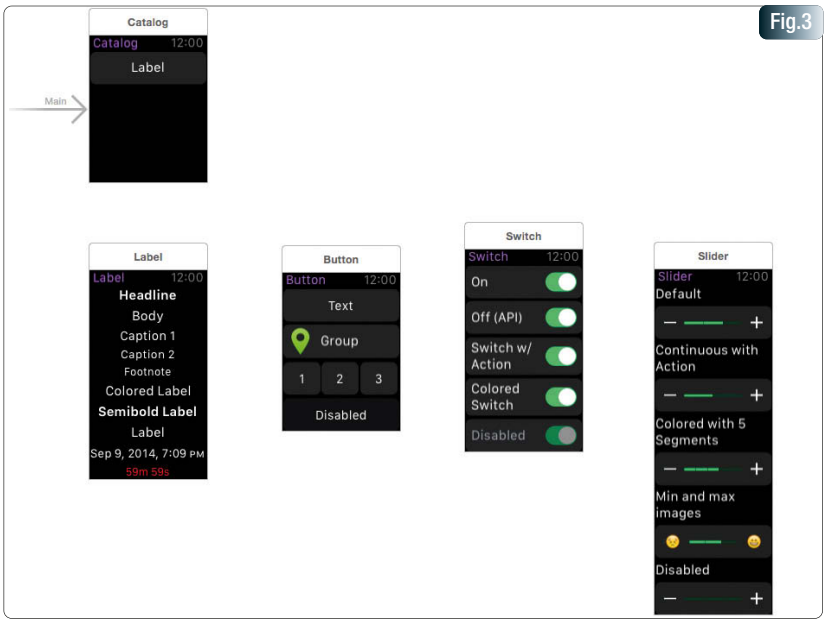


Fig.3

avec des lignes et des colonnes pour gérer les alignements.
Par défaut tous les éléments sont systématiquement positionnés les uns sous les autres. Il est également possible via des conteneurs, de les positionner les uns à côté des autres. En revanche il est impossible de positionner librement les composants (ex : mettre un bouton à 16px du haut et 24px de la gauche).

Afficher des données dans une liste

Parmi les composants visuels disponibles sur la Watch, on retrouve bien évidemment la liste des données (WKInterfaceTable) qui reprend le même principe que sur iPhone, à savoir, afficher des lignes d'informations les unes sous les autres et pouvant être sélectionnées.
Mais comme nous l'avons évoqué précédemment, les composants WatchKit s'utilisent différemment de leurs homologues UIKit. La plus grosse différence sur WatchKit réside dans la gestion des données qui doivent être entièrement connues avant de pouvoir être affichées, là où, sur iOS, elles n'ont besoin d'être connues qu'au moment où l'utilisateur souhaite les afficher (quand il fait défiler la liste).
Dans l'application d'exemple dont nous parlions, nous pouvons voir le code suivant dans le contrôleur gérant la liste de données :

```
- (instancetype)init {
    self = [super init];
    if (self) {
        [self loadTableData];
    }
return self;
}

- (void)loadTableData {
    self.cityNames = @[@"Cupertino", @"Sunnyvale", @"Campbell", @"Morgan Hill"];
    [self.interfaceTable setNumberOfRows:self.cityNames.count withRowType:@"default"];

    [self.cityNames enumerateObjectsUsingBlock:^(NSString *citName, NSUInteger idx, BOOL *stop) {
        AAPLTableRowController *row = [self.interfaceTable rowControllerAtIndex:idx];
        [row.rowLabel setText:citName];
    }];
}
```

Lors de l'initialisation du contrôleur, la liste de données est chargée dans une variable et l'on remarque que le nombre de lignes de données doit être défini (self.interfaceTable setNumberOfRows) avant que les lignes ne soient ensuite affichées dans la liste (via le bloc d'énumération des lignes).
Cela peut avoir un impact important sur une application puisque, comme nous l'avons dit précédemment, les données transitent entre l'application et la montre en Bluetooth. Dans cet exemple qui n'affiche que 4 lignes pas de soucis, mais pour une liste qui devrait afficher des centaines ou des milliers de lignes, cela pourrait engendrer une certaine latence désagréable pour l'utilisateur et donnant une impression de non fluidité de l'application.
Cette différence avec UIKit (qui ne charge les données que lorsqu'elles sont sur le point d'être affichées à l'écran) est fondamentale car elle impacte au premier plan la manière de concevoir son application. La solution si l'on souhaite afficher des centaines ou des milliers de lignes (la mémoire vive de la montre étant limitée) serait de découper la liste en plusieurs écrans.
Ensuite si l'on souhaite savoir sur quelle ligne l'utilisateur a appuyé afin de réagir en conséquence (ex : afficher un 2ème écran), il suffit d'écrire la méthode ci-dessous :

```
- (void)table:(WKInterfaceTable *)table didSelectRowAtIndex:(NSInteger)rowIndex {
…
}
```

Pour les développeurs habitués à manipuler les listes de données (UITableView) disponibles dans UIKit, ils verront également que la signature de cette méthode est différente de celle qu'ils ont l'habitude de manipuler.

Afficher un contrôleur à l'écran

Parmi les autres fonctionnalités importantes que vous aurez besoin de mettre en œuvre au sein d'une application WatchKit, nous parlerons de la possibilité d'afficher/changer le contrôleur actuellement visible à l'écran.
Pour cela, la première chose est d'avoir défini plusieurs contrôleurs au sein de son storyboard comme cela était le cas sur la capture vue précédemment, mais surtout d'avoir nommé ces contrôleurs pour pouvoir ensuite demander à WatchKit d'afficher celui-ci ou celui-là.
Toujours dans l'application d'exemple fournie par Apple, on peut voir cela dans un autre contrôleur de liste de données lorsque l'utilisateur sélectionne une ligne :

```
- (void)table:(WKInterfaceTable *)table didSelectRowAtIndex:(NSInteger)rowIndex {
    NSDictionary *rowData = self.elementsList[rowIndex];
    [self pushControllerWithName:rowData[@"controllerIdentifier"] context:nil];
}
```

Dans cet exemple, la ligne sélectionnée par l'utilisateur est récupérée dans un tableau de données sous la forme d'un dictionnaire. Dans ce dictionnaire est ensuite récupéré le nom du contrôleur devant être affiché à l'utilisateur (cf. rowData[@"controllerIdentifier"]).
Pour afficher le contrôleur souhaité, il suffit ensuite d'appeler la méthode pushControllerWithName à partir du contrôleur courant (self).
A noter qu'il est possible si cela est nécessaire, de passer des objets entre les contrôleurs en précisant un contexte lors de l'appel de la

méthode. Le résultat de l'appel de cette méthode est d'afficher le nouveau contrôleur par dessus le contrôleur courant en prenant toute la place disponible à l'écran (équivalent du mode modal sur iPhone).

Utilisation des « Glances »

Lorsque vous installez une application sur une Apple Watch, il existe 2 manières de l'utiliser :

- En lançant l'application et en naviguant dans ses écrans,
- En affichant un écran résumant les informations importantes de l'application.

Ce deuxième mode d'utilisation est réalisé via la mise en œuvre de ce qu'Apple appelle les « Glances ». Ces éléments permettent de définir un contrôleur spécifique au sein de votre application qui sera en charge d'afficher les données importantes, sans que l'utilisateur ait besoin d'exécuter votre application.

L'exemple typique de cette possibilité concerne l'application calendrier fournie en standard. Quand elle est exécutée, vous pouvez naviguer par jour, voir l'intégralité de vos rendez-vous, etc.

L'écran « Glance » de cette application permet de vous afficher vos prochains rendez-vous sous forme d'un mini tableau de bord de votre journée.

Pour inclure ce type de fonctionnalité au sein de votre application, il suffit dans le storyboard, d'inclure un contrôleur en utilisant le modèle dans la boîte à outils disponible au sein du storyboard dans Xcode comme on peut le voir en **Fig.4**.

Utilisation des notifications

Tout comme sur iOS, une des fonctionnalités importantes des applications WatchKit repose dans leur capacité à afficher des notifications permettant à l'utilisateur de savoir qu'un événement important vient de se produire (ex : rappel d'un rendez-vous sur le point de démarrer).

Pour définir ce que doit afficher votre application en fonction de la notification reçue, il est possible d'inclure dans le storyboard un ou plusieurs contrôleurs dédiés à cet usage (Notification Interface Controller comme indiqué sur la capture précédente).

Ces contrôleurs peuvent afficher des données aussi bien statiques (toujours identiques à ce qui est défini dans le storyboard) que dynamiques (le contenu change en fonction de la notification).

En fonction du type de notification que vous souhaitez utiliser (statique vs. dynamique), différentes méthodes du cycle de vie de l'application sont à votre disposition pour intervenir et effectuer des traitements complémentaires si nécessaires.

Le diagramme ci-dessous vous montre le cycle de vie et les différentes méthodes mises en œuvre dans le cadre du traitement de notifications dynamiques **Fig.5**.

Comment bien démarrer ?

Maintenant que vous avez un aperçu sur la manière de mettre en œuvre une application WatchKit avec la création d'un projet, l'implémentation d'une interface graphique, l'affichage de données sous forme de liste, ou comment passer d'un contrôleur à un autre, vous avez probablement envie d'en apprendre plus et d'aller plus loin.

Pour ce faire et comme nous l'avons évoqué précédemment, le meilleur moyen est de se baser sur les applications d'exemples fournies par Apple sur le portail réservé aux développeurs.

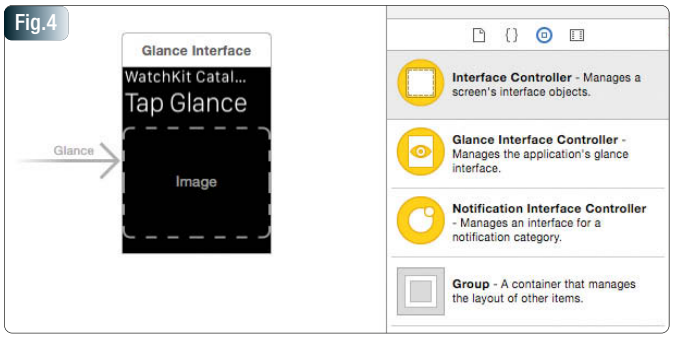
Outre l'application évoquée dans cet article (WatchKit Catalog), vous trouverez une autre application nommée « Lister (for Apple Watch, iOS, and OS X) » qui a notamment servi à de nombreuses démonstrations durant la dernière WWDC.

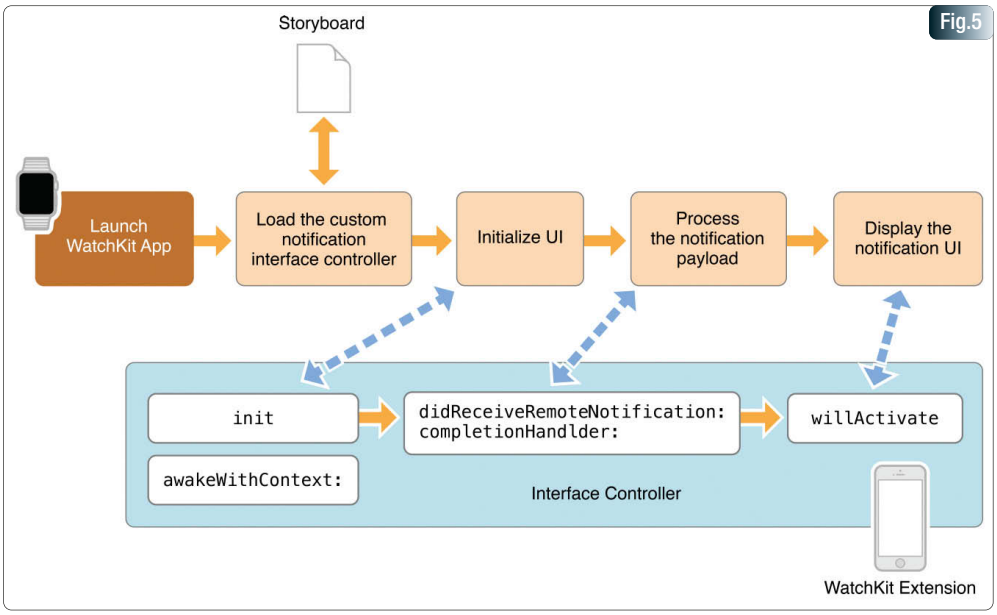
Elle permet de voir avec des exemples concrets, comment mettre en œuvre une vraie application iOS, avec une extension pour l'Apple Watch, et présentant tous les concepts importants pour ce type d'application (stockage de données, notifications...).

Celle-ci a été modifiée au fil des mois pour prendre en compte les dernières évolutions chez Apple, notamment le support de Swift 1.2, afin de bien démarrer sur la réalisation d'applications WatchKit quel que soit le langage que vous souhaitez utiliser.

Nul doute également que la prochaine WWDC qui aura lieu en Juin, fera la part belle au développement WatchKit.

Les vidéos qui seront mises en ligne par Apple dans la foulée de la conférence, serviront d'excellents tutoriaux pour vous aider à appréhender les bonnes pratiques de développement sur la dernière née d'Apple.





Intégration du DataTables Jquery dans ASP.NET MVC 5

1ère partie

Le « DataTables Jquery » est un outil open source JavaScript permettant l'affichage de données sous forme de tableau HTML au sein de votre application Web. Ce plugin présente l'avantage d'intégrer nativement des fonctionnalités courantes aux tableaux HTML, et offre une meilleure interaction avec les données manipulées en quelques lignes de code. Nous allons voir comment l'intégrer et l'utiliser dans une application ASP.NET MVC 5.

Georges DAMIEN
Consultant .NET/WEB chez
Cellenza
Blog : www.georgesdamien.com

Pourquoi l'utiliser ?

Tout d'abord, parce que le plugin « DataTables » est open source, mais surtout parce qu'il intègre nativement un ensemble de fonctionnalités récurrentes que l'on vous demandera, en général, d'implémenter dans un tableau HTML :

- Tri,
- Pagination,
- Fonction de recherche,
- Filtre sur le nombre d'éléments à affiche,r
- Personnalisation des filtres,
- "Responsive",
- Gestion des langues,
- Personnalisation de toutes les fonctionnalités proposées,
- Chargement via Ajax.

Afin d'éviter de « recoder » perpétuellement ces fonctionnalités de base que l'on retrouve dans l'affichage d'une liste d'éléments (chose qu'on retrouve sur une majorité des projets de développement Web en entreprise), nous allons donc utiliser le plugin « DataTables Jquery».

Comment l'utiliser ?

Le « DataTables » s'utilise sur tout type de site Web HTML : il a juste besoin d'un conteneur du DOM (un div en général) afin d'y injecter dynamiquement le tableau à afficher. Passons à la pratique :

Tout d'abord, il faut télécharger la librairie sur http://www.datatables.net/

La version que l'on utilisera ici est la dernière en date (1.10.4). Une fois le package téléchargé, vous y trouverez tout un ensemble d'éléments dont les fichiers JavaScript et CSS de base, des exemples d'utilisation ainsi que des extensions supplémentaires **Fig.1**.

Le contenu du dossier média contient l'essentiel pour utiliser la librairie : les CSS, les JS et quelques images. Une fois ces fichiers ajoutés sur votre site, l'utilisation basique du datable ressemble au code suivant :

```
<!DOCTYPE HTML>
<HTML>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="initial-scale=1.0, maximum-scale=2.0">
  <title>DataTables exmple</title>
  <link rel="stylesheet" type="text/css" href="../../media/css/jquery.dataTables.css" />
  <script type="text/javascript" language="javascript" src="../../media/js/jquery.js"></script>
  <script type="text/javascript" language="javascript" src="../../media/js/jquery.dataTables.js"></script>
  <script type="text/javascript" language="javascript" src="../resources/syntax/shCore.js"></script>
  <script type="text/javascript" language="javascript" src="../resources/demo.js"></script>
</head>

<body>
  <table id="example" cellspacing="0" width="100%"></table>

  <script type="text/javascript" language="javascript" class="init">
    var dataSet = [
    ['Trident', 'Internet Explorer 4.0', 'Win 95+', '4', 'X'],
    ['Gecko', 'Firefox 1.0', 'Win 98+ / OSX.2+', '1.7', 'A'],
    ['Webkit', 'S60', 'S60', '413', 'A'],
    ['Presto', 'Opera 7.0', 'Win 95+ / OSX.1+', '-', 'A'],
    ['KHTML', 'Konqureror 3.1', 'KDE 3.1', '3.1', 'C'],
    ['Tasman', 'Internet Explorer 4.5', 'Mac OS 8-9', '-', 'X'],
    ['Misc', 'NetFront 3.1', 'Embedded devices', '-', 'C']
    ];

    $(document).ready(function () {
      $('#example').dataTable({
        "data": dataSet,
        "columns": [
        { "title": "Engine" },
        { "title": "Browser" },
        { "title": "Platform" },
        { "title": "Version", "class": "center" },
        { "title": "Grade", "class": "center" }
        ]
      });
    });
  </script>
```

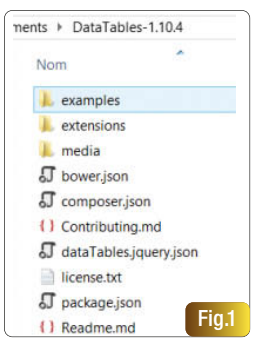


Fig.1

Contenu du package DataTables

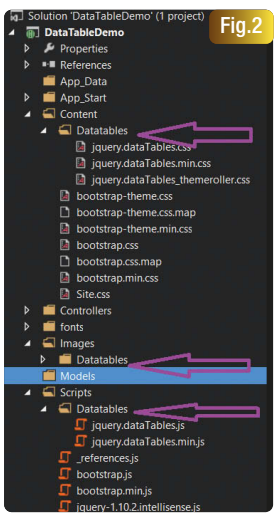


Fig.2

Exemple d'intégration des fichiers « DataTables » dans une solution ASP.NET MVC 5

```
</body>
</HTML>
```

Intégration dans ASP.NET MVC 5

Pour cette démo, je vais utiliser le template ASP.NET MVC 5 sans mode d'authentification. Une fois mon projet initialisé, je rajoute respectivement dans les répertoires « Content », « Images » et « Script » les fichiers CSS, images et scripts JS contenus dans le répertoire « media » du package du DataTables. Au passage, j'ai créé dans chacun de ces répertoires un sous-répertoire « DataTables » pour bien distinguer cette librairie.

NB : cette réorganisation des fichiers engendre des erreurs de récupération des images (servant, par exemple, pour les tris). Il faut donc penser à corriger les liens dans les fichiers CSS :

Par exemple, dans mon cas, la CSS suivante :

```
table.dataTable thead .sorting {
    background: url("../images/sort_both.png") no-repeat center right;
}
```

Devient :

```
table.dataTable thead .sorting {
    background: url("../images/Datatables/sort_both.png") no-repeat center right;
}
```

Voici à quoi ressemble notre arborescence dans Visual Studio : **Fig.2**.

Cette décomposition est, bien entendu, à titre illustratif. Chaque projet a son architecture à part entière : du moment où vos références aux fichiers de la librairie sont correctes là où elles sont appelées, cette dernière fonctionnera correctement. Dans notre exemple ASP.NET MVC 5, je vais directement les référencer dans le fichier « Bundle.Config » :

```
public static void RegisterBundles(BundleCollection bundles)
{
    bundles.Add(new ScriptBundle("~/bundles/jquery").Include(
    "~/Scripts/jquery-{version}.js"));

    bundles.Add(new ScriptBundle("~/bundles/jqueryval").Include(
    "~/Scripts/jquery.validate*"));

    bundles.Add(new ScriptBundle("~/bundles/modernizr").Include(
    "~/Scripts/modernizr-*"));

    bundles.Add(new ScriptBundle("~/bundles/bootstrap").Include(
    "~/Scripts/bootstrap.js",
    "~/Scripts/respond.js",
    "~/Scripts/Datatables/jquery.dataTables.min.js"));

    bundles.Add(new StyleBundle("~/Content/css").Include(
    "~/Content/bootstrap.css",
    "~/Content/site.css",
    "~/Content/Datatables/jquery.dataTables.min.css"));

    BundleTable.EnableOptimizations = true;
}
```

Prérequis

Vous l'aurez compris, le principal prérequis pour l'utilisation de cette librairie est le chargement préalable de la librairie « JQUERY » : cela tombe bien dans notre cas, puisqu'elle est déjà intégrée au projet ASP.NET MVC 5.

Utilisation dans notre View ASP.NET MVC 5

Voici le contenu de ma « View mvc » utilisant le « DataTables » :

```
@{
 ViewBag.Title = "Home Page";
}
<div class="jumbotron">
  <h4>DEMO DATATABLES JQUERY</h4>
</div>
<div class="row">
  <table id="example" class="table table-striped table-hover dt-responsive" cellspacing
="0" width="100%"></table>
</div>

@section scripts
{
<script>
  var dataSet = [
  ['Trident', 'Internet Explorer 4.0', 'Win 95+', '4', 'X'],
  ['Trident', 'Internet Explorer 5.0', 'Win 95+', '5', 'C'],
  ['Trident', 'Internet Explorer 5.5', 'Win 95+', '5.5', 'A'],
  ['Trident', 'Internet Explorer 6', 'Win 98+', '6', 'A'],
  ['Trident', 'Internet Explorer 7', 'Win XP SP2+', '7', 'A'],
  ['Trident', 'AOL browser (AOL desktop)', 'Win XP', '6', 'A'],
  ['Gecko', 'Firefox 1.0', 'Win 98+ / OSX.2+', '1.7', 'A'],
  ['Gecko', 'Firefox 1.5', 'Win 98+ / OSX.2+', '1.8', 'A'],
  ['Gecko', 'Firefox 2.0', 'Win 98+ / OSX.2+', '1.8', 'A'],
  ['Gecko', 'Firefox 3.0', 'Win 2k+ / OSX.3+', '1.9', 'A'],
  ['Gecko', 'Camino 1.0', 'OSX.2+', '1.8', 'A'],
  ['Gecko', 'Camino 1.5', 'OSX.3+', '1.8', 'A'],
  ['Gecko', 'Netscape 7.2', 'Win 95+ / Mac OS 8.6-9.2', '1.7', 'A'],
  ['Gecko', 'Netscape Browser 8', 'Win 98SE+', '1.7', 'A'],
  ['Gecko', 'Netscape Navigator 9', 'Win 98+ / OSX.2+', '1.8', 'A'],
  ['Gecko', 'Mozilla 1.0', 'Win 95+ / OSX.1+', 1, 'A'],
  ['Gecko', 'Mozilla 1.1', 'Win 95+ / OSX.1+', 1.1, 'A'],
  ['Gecko', 'Mozilla 1.2', 'Win 95+ / OSX.1+', 1.2, 'A'],
  ['Gecko', 'Mozilla 1.3', 'Win 95+ / OSX.1+', 1.3, 'A'],
  ['Gecko', 'Mozilla 1.4', 'Win 95+ / OSX.1+', 1.4, 'A'],
  ['Gecko', 'Mozilla 1.5', 'Win 95+ / OSX.1+', 1.5, 'A'],
  ['Gecko', 'Mozilla 1.6', 'Win 95+ / OSX.1+', 1.6, 'A'],
  ['Gecko', 'Mozilla 1.7', 'Win 98+ / OSX.1+', 1.7, 'A'],
  ['Gecko', 'Mozilla 1.8', 'Win 98+ / OSX.1+', 1.8, 'A'],
  ['Gecko', 'Seamonkey 1.1', 'Win 98+ / OSX.2+', '1.8', 'A'],
  ['Gecko', 'Epiphany 2.20', 'Gnome', '1.8', 'A'],
  ['Webkit', 'Safari 1.2', 'OSX.3', '125.5', 'A'],
  ['Webkit', 'Safari 1.3', 'OSX.3', '312.8', 'A'],
  ['Webkit', 'Safari 2.0', 'OSX.4+', '419.3', 'A'],
  ['Webkit', 'Safari 3.0', 'OSX.4+', '522.1', 'A'],
  ['Webkit', 'OmniWeb 5.5', 'OSX.4+', '420', 'A'],
  ['Webkit', 'iPod Touch / iPhone', 'iPod', '420.1', 'A'],
  ['Webkit', 'S60', 'S60', '413', 'A'],
  ['Presto', 'Opera 7.0', 'Win 95+ / OSX.1+', '-', 'A'],
  ['Presto', 'Opera 7.5', 'Win 95+ / OSX.2+', '-', 'A'],
  ['Presto', 'Opera 8.0', 'Win 95+ / OSX.2+', '-', 'A'],
  ['Presto', 'Opera 8.5', 'Win 95+ / OSX.2+', '-', 'A'],
  ['Presto', 'Opera 9.0', 'Win 95+ / OSX.3+', '-', 'A'],
  ['Presto', 'Opera 9.2', 'Win 88+ / OSX.3+', '-', 'A'],
  ['Presto', 'Opera 9.5', 'Win 88+ / OSX.3+', '-', 'A'],
  ['Presto', 'Opera for Wii', 'Wii', '-', 'A'],
```

```
  ['Presto', 'Nokia N800', 'N800', '-', 'A'],
  ['Presto', 'Nintendo DS browser', 'Nintendo DS', '8.5', 'C/A<sup>1</sup>'],
  ['KHTML', 'Konqureror 3.1', 'KDE 3.1', '3.1', 'C'],
  ['KHTML', 'Konqureror 3.3', 'KDE 3.3', '3.3', 'A'],
  ['KHTML', 'Konqureror 3.5', 'KDE 3.5', '3.5', 'A'],
  ['Tasman', 'Internet Explorer 4.5', 'Mac OS 8-9', '-', 'X'],
  ['Tasman', 'Internet Explorer 5.1', 'Mac OS 7.6-9', '1', 'C'],
  ['Tasman', 'Internet Explorer 5.2', 'Mac OS 8-X', '1', 'C'],
  ['Misc', 'NetFront 3.1', 'Embedded devices', '-', 'C'],
  ['Misc', 'NetFront 3.4', 'Embedded devices', '-', 'A'],
  ['Misc', 'Dillo 0.8', 'Embedded devices', '-', 'X'],
  ['Misc', 'Links', 'Text only', '-', 'X'],
  ['Misc', 'Lynx', 'Text only', '-', 'X'],
  ['Misc', 'IE Mobile', 'Windows Mobile 6', '-', 'C'],
  ['Misc', 'PSP browser', 'PSP', '-', 'C'],
  ['Other browsers', 'All others', '-', '-', 'U']
  ];

  $(document).ready(function () {
    $('#example').datatable({
      "data": dataSet,
      "columns": [
      { "title": "Engine" },
      { "title": "Browser" },
      { "title": "Platform" },
      { "title": "Version", "class": "center" },
      { "title": "Grade", "class": "center" }
      ]
    });
  });
</script>
}
```

Je charge ici une liste JSON d'éléments instanciés dans le script en fin de page. L'instanciation du « DataTables » s'effectue simplement en appelant la fonction « datatable » (sans le S à la fin) à partir d'un tableau du DOM (ici c'est le tableau avec l'ID 'exemple'). Il existe tout un ensemble de fonctions permettant la 'customisation' du tableau. Dans l'exemple précédent, j'ai rajouté les entêtes de colonnes de mon tableau via la propriété "columns" du 'DataTables'. Ma source de données provient de mon JSON instancié ici en dur dans la variable "dataset". Je pourrais, bien évidemment, charger cette liste directement depuis un appel Ajax. Au passage, j'ai rajouté quelques classes css 'bootstrap' au tableau, histoire d'avoir un aperçu acceptable.

```
class="table table-striped table-hover dt-responsive"
```

Sans spécification particulière, le « DataTables » activera ses fonctionnalités par défaut (tri, pagination, recherche, etc.). On aura en résultat un tableau déjà "opérationnel". Nous pourrons cependant personnaliser ou désactiver chacune des fonctionnalités proposées Fig.3.

Internationalisation

Pour adapter les différents titres dans votre langue, vous trouverez ici votre bonheur. Vous pourrez cependant personnaliser vous-même les traductions. Voici un exemple pour la langue française :

```
// Initialisation de la langue française
  "language": {
    "sProcessing": "Traitement en cours...",
    "sSearch": "Rechercher :",
    "sLengthMenu": "Afficher _MENU_ &eacute;l&eacute;ments",
    "sInfo": "Affichage de l'&eacute;lement _START_ &agrave; _END_ sur _TOTAL_ &eacute
;l&eacute;ments",
    "sInfoEmpty": "Affichage de l'&eacute;lement 0 &agrave; 0 sur 0 &eacute;l&eacute;ments",
    "sInfoFiltered": "(filtr&eacute; de _MAX_ &eacute;l&eacute;ments au total)",
    "sInfoPostFix": "",
    "sLoadingRecords": "Chargement en cours...",
    "sZeroRecords": "Aucun &eacute;l&eacute;ment &agrave; afficher",
    "sEmptyTable": "Aucune donn&eacute;e disponible dans le tableau",
    "oPaginate": {
      "sFirst": "Premier  ",
      "sPrevious": "Pr&eacute;c&eacute;dent  ",
      "sNext": "Suivant",
      "sLast": "  Dernier"
    },
  "oAria": {
    "sSortAscending": ": activer pour trier la colonne par ordre croissant",
    "sSortDescending": ": activer pour trier la colonne par ordre d&eacute;croissant"
  }
  },
  // désactivation de la pagination
"paginate": false,
}
```

Au passage, j'ai désactivé la pagination à titre illustratif (propriété « paginate » à « false »). Nous avons pu voir comment instancier de façon simple un « DataTables Jquery », ainsi que quelques options.
Voyons maintenant comment interagir avec le serveur, et notamment comment récupérer des données via Ajax et comment passer des paramètres de filtre au serveur.

Cas approfondi

Jusque-là, nous avons pu voir comment intégrer facilement ce plugin qui permet l'affichage de données sous forme de tableau dans un projet web ASP.NET MVC 5. Nous allons maintenant aller en profondeur dans notre démarche d'intégration de ce plugin et voir comment cela se présente dans un vrai projet.
Dans un contexte d'entreprise, nous nous retrouvons à gérer des données grandissantes dans le temps (en tout cas nous devons le prévoir). Ces informations sont, en général, stockées dans des bases de données. Ainsi, il nous faudra charger nos données depuis notre serveur et penser également à limiter le nombre de lignes à afficher dans notre tableau « DataTables » afin d'optimiser le chargement de notre page.
Dans notre projet MVC de démo, nous allons garder notre exemple et en faire un modèle MVC qui nous servira tout au long des exemples à venir.

La suite dans Programmez! n°187.

Afficher 10 éléments Rechercher : ar

Engine	Browser	Platform	Version	Grade
Webkit	Safari 1.2	OSX.3	125.5	A
Webkit	Safari 1.3	OSX.3	312.8	A
Webkit	Safari 2.0	OSX.4+	419.3	A
Webkit	Safari 3.0	OSX.4+	522.1	A

Affichage de l'élement 1 à 4 sur 4 éléments (filtré de 57 éléments au total) Précédent 1 Suivant

Exemple d'affichage du « DataTables »

Fig.3

Le Bluetooth Low Energy dans les applications universelles

2e partie

Dans la 1ère partie de cet article nous avons vu ce qu'est le Bluetooth Low Energy et son couplage fort avec les objets connectés. Nous avons aussi découvert le SensoTag de Texas Instruments ainsi que les concepts de base concernant l'utilisation du Bluetooth Low Energy dans les applications Windows Universelles.

Stéphane Sibué
Directeur technique chez GUYZMO
stephane.sibue@guyzmo.fr - www.guyzmo.fr

GUYZMO

SensorBase, le capteur générique

Tous les capteurs du SensorTag fonctionnent de la même manière, il est donc fortement recommandé de créer une classe générale à partir de laquelle nous pourrons créer tous les différents types de capteurs supportés par le SensorTag. Pour être en mesure d'utiliser un capteur du SensorTag, il faut 3 informations de base :

- L'UUID du service,
- L'UUID de la caractéristique de configuration,
- L'UUID de la caractéristique de données.

Ces 3 informations seront donc à fournir dans le constructeur et stockées sous la forme de GUID. Il faudra aussi garder une trace du service et des caractéristiques pendant leur utilisation :

```
using Windows.Devices.Bluetooth.GenericAttributeProfile;

protected Guid pServiceUUID;
protected Guid pConfigurationUUID;
protected Guid pDataUUID;

protected GattDeviceService pDeviceService;
protected GattCharacteristic pConfigurationCharacteristic;
protected GattCharacteristic pDataCharacteristic;

public SensorBase(string serviceUUID, string configurationUUID, string dataUUID)
{
    pServiceUUID = new Guid(serviceUUID);
    pConfigurationUUID = new Guid(configurationUUID);
    pDataUUID = new Guid(dataUUID);
}
```

La première opération à effectuer sur le capteur est de l'initialiser. Cette opération peut se passer correctement ou non. Nous allons donc gérer l'état de l'initialisation sous la forme d'une énumération et coder la méthode d'initialisation qui pourra éventuellement être surchargée par les capteurs qui hériteront de **SensorBase**. Cette méthode retournera **DeviceNotFound** si aucun SensorTag n'est trouvé pour le service demandé ou **Ok** si tout se passe bien. On en profite aussi pour mettre en place la « tuyauterie » qui permettra la récupération de la valeur portée par le capteur (température, pression, etc) grâce aux notifications. Dans la classe **SensorBase** l'implémentation est vide car elle est différente d'un capteur à un autre :

```
public enum InitResult
{
    Ok,
    DeviceNotFound
}

public virtual async Task<InitResult> Init()
{
    // On recherche un device supportant le service du sensor
    // Si le device trouvé est un SensorTag on retourne le service
    // Sinon on retourne null

    var deviceService = await GetDeviceService(pServiceUUID);

    if (deviceService == null)
    {
        // Aucun SensorTag ne supporte ce service

        return InitResult.DeviceNotFound;
    }

    pDeviceService = deviceService;

    // On a le DeviceService
    // On peut maintenant se brancher sur les caractéristiques
    // Configuration et Data

    var configCharacteristic = GetCharacteristic(pDeviceService, pConfigurationUUID);

    if (configCharacteristic == null)
    {
        return InitResult.DeviceNotFound;
```

```
    }

    var dataCharacteristic = GetCharacteristic(pDeviceService, pDataUUID);

    if (dataCharacteristic == null)
    {
      return InitResult.DeviceNotFound;
    }

    // On peut stocker les caractéristiques trouvées

    pConfigurationCharacteristic = configCharacteristic;
    pDataCharacteristic = dataCharacteristic;

    // On peut se brancher sur l'événement de changement de valeur des données

    await pDataCharacteristic.WriteClientCharacteristicConfigurationDescriptorAsync(GattClient
CharacteristicConfigurationDescriptorValue.Notify);

    pDataCharacteristic.ValueChanged += pDataCharacteristic_ValueChanged;

    return InitResult.Ok;
}

void pDataCharacteristic_ValueChanged(GattCharacteristic sender, GattValueChangedEvent
Args args)
{
    OnValueChanged(args.CharacteristicValue);
}

protected virtual void OnValueChanged(IBuffer buffer)
{

}
```

*Note : La méthode **GetDeviceService** a été vue dans la partie 1 de cet article*

Une fois initialisé (en gros une fois qu'on est en lien avec le capteur) il faut l'activer, car, par défaut, les capteurs sont arrêtés (économie d'énergie). Là encore, c'est la même chose avec tous les capteurs du SensorTag, il suffit d'écrire la valeur numérique 1 dans la caractéristique de configuration. La valeur 0 quant à elle arrête le capteur. Les fonctions **Start** et **Stop** peuvent donc être générales :

```
public virtual async Task<bool> StartSensor()
{
    if (pConfigurationCharacteristic != null)
    {
      using (var writer = new DataWriter())
      {
        writer.WriteByte(1);
        var status = await pConfigurationCharacteristic.WriteValueAsync(writer.DetachBuffer());
        return (status == GattCommunicationStatus.Success);
      }
    }

    return false;
}
```

```
public virtual async Task<bool> StopSensor()
{
    if (pConfigurationCharacteristic != null)
    {
      using (var writer = new DataWriter())
      {
        writer.WriteByte(0);
        var status = await
pConfigurationCharacteristic.WriteValueAsync(writer.DetachBuffer());
        return (status == GattCommunicationStatus.Success);
      }
    }

    return false;
}
```

Le binding XAML

Comme nous sommes en train de développer une application universelle, le XAML est au centre de la définition de l'interface graphique. Le XAML permet de binder directement des propriétés d'objets à celles de contrôles visuels, ce qui est très pratique et très puissant.
Pour simplifier les choses, les classes de gestion des capteurs seront directement compatibles avec cette fonctionnalité. Ainsi la classe **SensorBase** hérite de **ObservableObject** qui permet d'envoyer une notification au moteur de binding XAML à chaque fois qu'une valeur destinée à être affichée est modifiée. Dans la station météo que nous sommes en train de réaliser nous devons afficher la valeur de chaque capteur dès qu'elle change, c'est pourquoi l'implémentation prend directement en charge le binding. Voici le code de **ObservableObject** :

```
public abstract class ObservableObject : INotifyPropertyChanged
{
    public event PropertyChangedEventHandler PropertyChanged;

    protected virtual async void NotifyPropertyChanged(string propertyName)
    {
      var propertyChanged = this.PropertyChanged;

      if (propertyChanged != null)
      {
        await CoreApplication.MainView.CoreWindow.Dispatcher.RunAsync(CoreDispatcher
Priority.Normal, () =>
  {
    propertyChanged(this, new PropertyChangedEventArgs(propertyName));
  });
      }
    }
}
```

TemperatureSensor

Maintenant que nous avons une base, nous pouvons créer une classe pour gérer chaque type de capteur présent dans le SensorTag. Nous allons voir en détails celle qui gère le capteur de température, **TemperatureSensor** qui hérite, vous l'avez deviné, de **SensorBase**. Pour ce capteur il n'y a que 3 choses à ajouter :

Une propriété pour accéder à la température

C'est la base, il nous faut pouvoir lire la température actuelle retournée par le capteur. Cette propriété est de type double, en lecture seule depuis l'extérieur :

```
protected double pCurrentTemperature;

public double CurrentTemperature
{
  get { return pCurrentTemperature; }

  private set
  {
    pCurrentTemperature = value;
    NotifyPropertyChanged("CurrentTemperature");
  }
}
```

Un constructeur adapté

Le constructeur de la classe parent **SensorBase** permet de donner en paramètres les UUID nécessaires, il suffit de l'utiliser avec les paramètres liés au capteur de température :

```
public TemperatureSensor()
 : base("F000AA00-0451-4000-B000-000000000000",
"F000AA02-0451-4000-B000-000000000000",
"F000AA01-0451-4000-B000-000000000000")
{

}
```

Le code de calcul de la température

A chaque fois que la valeur de température change, une notification est lancée et la méthode définie à la base dans **SensorBase** est exécutée. Il suffit de surcharger cette méthode et le tour est joué :

```
protected override void OnValueChanged(IBuffer buffer)
{
  DataReader wReader = DataReader.FromBuffer(buffer);

  using (wReader)
  {
    byte[] b = new byte[4];
    wReader.ReadBytes(b);

    var ambientTemperature = BitConverter.ToInt16(b, 2) / 128.0;

    double Vobj2 = BitConverter.ToInt16(b, 0);
    Vobj2 *= 0.00000015625;

    double Tdie = ambientTemperature + 273.15;

    double S0 = 5.593E-14;
    double a1 = 1.75E-3;
    double a2 = -1.678E-5;
    double b0 = -2.94E-5;
    double b1 = -5.7E-7;
    double b2 = 4.63E-9;
    double c2 = 13.4;
    double Tref = 298.15;
    double S = S0 * (1 + a1 * (Tdie - Tref) + a2 * Math.Pow((Tdie - Tref), 2));
    double Vos = b0 + b1 * (Tdie - Tref) + b2 * Math.Pow((Tdie - Tref), 2);
    double fObj = (Vobj2 - Vos) + c2 * Math.Pow((Vobj2 - Vos), 2);
    double tObj = Math.Pow(Math.Pow(Tdie, 4) + (fObj / S), .25);
    CurrentTemperature = tObj - 273.15;
  }
}
```

Les données sont stockées dans un tableau de 4 octets, les 2 premiers utilisés pour stocker la température de l'objet (température IR) et les 2 derniers pour la température ambiante. Les formules à utiliser pour convertir ces valeurs en températures exploitables sont expliquées dans la documentation en ligne du SensorTag mais c'est bien plus clair sur ce blog : http://tinyurl.com/lbkx38f

HumiditySensor

Après la température, passons au capteur d'humidité. Il fonctionne exactement comme le capteur de température, il faut juste lui donner les bons paramètres lors de son initialisation (les UUID) et faire les bons calculs de conversion (l'humidité est en fait un taux d'humidité exprimé en %) :

```
protected double pCurrentHumidity;

public double CurrentHumidity
{
  get { return pCurrentHumidity; }

  private set
  {
    pCurrentHumidity = value;
    NotifyPropertyChanged("CurrentHumidity");
  }
}

public HumiditySensor()
: base("F000AA20-0451-4000-B000-000000000000",
"F000AA22-0451-4000-B000-000000000000",
"F000AA21-0451-4000-B000-000000000000")
{

}

protected override void OnValueChanged(IBuffer buffer)
{
  using (var wReader = DataReader.FromBuffer(buffer);
  {
    byte[] b = new byte[4];
    wReader.ReadBytes(b);

    int hum = BitConverter.ToUInt16(b, 2);
    hum = hum - (hum % 4);
    CurrentHumidity = -6.0 + 125.0 * (hum / 65535.0);
  }
}
```

Conclusion

Dans cette 2ème partie nous avons vu comment accéder de manière simple aux capteurs du SensorTag qui fonctionnent tous sur le même modèle. Dans la 3ème et dernière partie nous verrons comment implémenter le capteur de pression qui est légèrement différent des autres et comment intégrer tout ce petit monde dans une interface XAML universelle afin d'obtenir notre station météo maison.

Responsive Design : qu'est-ce que c'est ?

Responsive Design est une notion qui se réfère au développement de sites Web. Elle signifie littéralement « Design Adaptatif »; le responsive design facilite la construction d'un site unique qui fonctionnera sur plusieurs résolutions d'écran. Par exemple un site Web adaptatif (qui peut être aussi désigné comme réactif ou responsive, ou encore évolutif) pourra lors de son accès par des appareils divers, afficher plusieurs colonnes sur un écran de PC de bureau/portable alors que sur une tablette ce même site va afficher deux colonnes, et sur un smartphone, une seule.

Aymen CHOUGUIAT
Supinfo

Responsive Web : pour ou contre ?

On va commencer par ce qui est avantageux. Tout d'abord le coût; vous disposez d'un seul et unique site Web à développer, ce qui réduit considérablement les frais de développement et de mise en place que vous auriez à faire dans le cas où vous voudriez lancer un site Web classique, c'est-à-dire avoir différents sites pour chaque appareil (PC / Tablettes / Montres / Smartphones)

Le deuxième point sur cette liste des « pour » est la maintenance: vous n'avez pas plusieurs sites Web à gérer. Mettre à jour le contenu sur votre site vous permettra d'afficher ce même contenu sur tous les appareils qui accèderont à ce dernier.

Créer un site Web adaptatif apporte de la cohérence, ainsi les visiteurs de votre site obtiennent la même expérience utilisateur qu'ils utilisent leur téléphone, leur tablette ou leur PC ; c'est un avantage mais ce peut aussi être un inconvénient qu'on verra plus tard. Sur ce même point un site réactif se veut aussi versatile sur la partie logicielle avec la compatibilité sur la majorité des navigateurs: le rendu d'un site Web adaptatif est généralement bien retranscrit quel que soit le navigateur utilisé.

Le Responsive Design a une large base de soutien, des centaines de milliers de développeurs actifs travaillent constamment à adapter et à améliorer les fonctionnalités principales des outils de créations, rendant le concept de Web design évolutif comme étant un standard de facto.

Passons maintenant aux inconvénients relatifs à ces sites qu'on peut considérer comme « touche-à-tout »

On en parlait précédemment : l'expérience utilisateur sur un site réactif est par définition la même quel que soit l'appareil que vous utilisez. C'est bien quand c'est fonctionnel, mais pas tant que ça quand ça ne l'est pas ; certaines pages peuvent s'afficher parfaitement sur un écran de PC de bureau mais pas aussi bien sur un smartphone quand celles-ci sont redimensionnées. De plus l'affichage d'un site Web sur un téléphone mobile peut nécessiter une expérience complètement différente de celle d'un ordinateur de bureau. Avoir un seul site restreint pour vous la possibilité d'avoir cette flexibilité.

Le développement d'un site en Responsive Design demande énormément de temps, et vous oblige à solliciter des développeurs aguerris ayant déjà une expérience sur différentes plateformes mobiles : cela implique une multitude de tests à effectuer selon les différents cas de figure et une anticipation sur la mise en page sur les différents supports et navigateurs.

Mettre en place un site en Responsive Dsign implique de prendre le temps de comprendre et de modéliser les besoins du client. Une vision bien spécifique est alors nécessaire pour répondre aux attentes de celui-ci. L'étape de la conception est par conséquent plus lourde car il faut réfléchir aux différentes plateformes, aux contenus, et aux bases fondamentales du design de votre site.

Quand vous vous lancerez dans le développement d'un site Web responsive vous devrez faire un choix, celui de proposer une expérience utilisateur (UX) novatrice ou de faire un site qui met l'accent sur le confort général en termes de lecture. En effet, le responsive design ne tient pas compte des fonctionnalités de chaque support, il impose une uniformité et une globalité sur l'ensemble des supports.

Seule une adaptation automatique est mise à votre disposition. L'impact sur le temps de production est considérable selon le choix que vous ferez. Dans le premier cas de figure, vous ne pourrez pas exploiter le potentiel de votre créativité a cause des contraintes qu'apporte le Responsive Design; dans ce cas la conception prendra relativement moins de temps. L'alternative vous permettra d'avoir plus de libertés mais le temps nécessaire pour faire naître votre projet augmentera drastiquement.

Un des atouts du Responsive Design est son faible coût, mais sous certaines conditions, le facteur argent dépend en grande partie de l'organisation et du plan de départ.

Le coût de développement d'un site responsive est nettement plus élevé que pour la réalisation d'un site classique si le projet n'a pas été bien pensé dès le départ, et, comme on l'a vu, les facteurs qui influent sur le coût sont nombreux (conception, maintenance, objectifs...etc).

Dans un site Web adaptatif vous ne serez pas toujours amenés à utiliser tous les éléments disponibles dans un responsive Framework.

Ceci signifie qu'il pourrait y avoir du code non utilisé et non nécessaire, que ce soit pour un projet personnel ou professionnel ; c'est un gros point noir qu'il faudra prendre en considération avant de commencer à développer !

Structures logicielles adaptatives

Les structures logicielles ou « Framework » d'un site Web adaptatif sont des grilles qui sont utilisées comme des blocs de construction pour les sites Web évolutifs.

Cela fait gagner du temps en donnant à l'utilisateur les blocs nécessaires pour construire un site Web ; en gros c'est un ensemble de code qui peut

être réutilisé afin que les développeurs ne soient pas obligés à chaque fois de tout commencer en partant de zéro.

Media Queries

Avant d'entrer dans les détails on va voir ce que sont les Medias Queries.
Les Media Queries sont des règles que l'on peut appliquer dans certaines conditions et qui font partie intégrante du langage CSS. Concrètement, vous allez pouvoir dire dans votre code « Si la résolution de l'écran de l'internaute est inférieure à tant, alors applique les propriétés CSS suivantes » , cela vous permet de changer l'apparence du site dans certaines conditions, vous pourrez augmenter la taille du texte, changer la couleur de fond, positionner différemment votre menu dans des résolutions spécifiques, etc.
Un des outils en rapport avec les medias queries et dans la création d'un site Web adaptatif en général est **« Adapt.js ».** Adapt.js est un fichier JavaScript allégé qui détermine quel fichier CSS charger avant que votre navigateur Internet affiche une page. Si le navigateur déborde ou redimensionne les éléments affichés, Adapt.js vérifie sa largeur et délivre le fichier CSS qui est nécessaire, lorsque cela est nécessaire.

Point de vue : Code

Pour comprendre pourquoi développer un site Web responsive prend autant de temps de nos jours, on va revenir un peu en arrière, bien avant l'ère des smartphones et des tablettes, et essayer de savoir ce qui s'est passé.
Avant on disposait toujours du HTML et du CSS comme aujourd'hui. On avait aussi les medias queries, même si ces outils étaient plus destinés à l'impression et à l'accessibilité qu'à autre chose. Il est donc normal que les normes d'efficacité et la base sur laquelle s'organise le processus de conception d'un site Web aient évolué. Avec l'avènement des appareils mobiles, la demande et les objectifs ont changé : le processus de conception est devenu plus complexe, plus long à réaliser.
C'est pour cela qu'il faut adopter une méthodologie dans la conception où la norme est que les règles CSS inline soient totalement responsables de l'implémentation d'un Web design responsive.
Plus concrètement, on devrait pourvoir effectuer ça :

```
#mon-element-id {/* Styles */} //Règles à adopter pour les styles bureau
.phone #my-element-id {/* Styles */} //Règles à adopter pour les styles Smartphones
.tablet #my-element-id {/* Styles */} //Règles à adopter pour les styles Tablettes
```

Au lieu d'avoir une structure comme celle-ci :

```
#my-element-id {} // Règles à adopter pour les styles bureau

@media only screen and (min-width : 320px) and (max-width : 480px) {
/* Règles à adopter pour certains appareils mobiles*/
}

@media only screen and (min-width : 321px) {
/* Règles à adopter pour d'autres appareils mobiles */
}

/** ...Directives principales de vos medias queries **/
```

Les medias queries font partie du CSS, mais dans ce cas précis comme on peut le voir ci-dessus, ils plombent sérieusement votre productivité, principalement à cause des répartitions illogiques des balises de zones qui seront introduites dans votre CSS en guise de directives medias; en gros votre base de code séquentielle est maintenant divisée en plusieurs zones, dont chacune que vous aurez à gérer et à entretenir séparément. Cela vous prendra beaucoup de temps et par conséquent, vous coûtera de l'argent.
Pour faciliter grandement la maintenance, ou la mise à niveau d'un site classique vers un site responsive (ce qui n'est pas très conseillé à moins que vous n'ayez pas le choix), on va s'intéresser à un outil javascript qui s'appelle « Restive.js »

Restive.js

Tout en restant dans l'optique d'optimisation et de gain de temps sur l'ensemble de vos travaux, le principe derrière restive.js est de faire du responsive Web design avec moins de code.
Restive.js est un plugin jQuery (ou framework) qui vous permet d'ajouter des fonctionnalités responsive a un site Web presque instantanément, il effectue des opérations assez lourdes en background pour vous donner plus de flexibilité, il dispose de beaucoup de fonctions que vous pourrez explorer quand vous aurez le temps.
Dans le contexte de la construction de votre site Web responsive, il vous permet de définir l'ensemble de vos balises « Responsives » CSS inline. L'installeur est très simple, et tout ce dont vous avez besoin pour l'installer et le configurer dans le cas qu'on va voir tient dans une dizaine de lignes de code :

```
<!-- Installe jQuery version 1.7 ou superieur -->
<script type="text/javascript" src="jquery.min.js"></script>

<!-- Installe Restive.js -->
<script type="text/javascript" src="restive.min.js"></script>

<!-- Configuration Restive.js -->
<script>
  $( document ).ready(function() {
    $('body').restive({
      breakpoints: ['10000'],
      classes: ['nb'],
      turbo_classes:'is_mobile=mobi,is_phone=phone,is_tablet=tablet,is_portrait=
portrait,is_landscape=landscape'
    });
  });
</script>
```

Restive.js a le format de base de la plupart des plugins jQuery, dans le code ci-dessus nous avons défini trois option, **breakpoints, classes** et **turbo_classes.**
Breakpoints et classes sont les principales options dans restive.js. Leur fonction est de configurer des points d'arrêt déclarés, puis d'appliquer des classes à un sélecteur donné pour que la largeur de la fenêtre de l'appareil qui navigue sur votre site corresponde à une plage spécifique de pixels.
Donc si on définit des points d'arrêts : ['240' , '320'], et les classes ['240-css' , '320-css'], le plugin va configurer les plages des points d'arrêt de 0 à 240 pixels, et de 241 à 320 pixels, il ajoutera ensuite la classe 240-css au <body> de votre fichier html (qui est notre sélecteur) si la largeur de la fenêtre de l'appareil se situe entre 0 et 240 pixels, sinon il ajoutera la classe 320-css si la largeur de la fenêtre de l'appareil se situe entre 241 et 320 pixels. Voilà comment fonctionnent à peu près ces deux options complémentaires.
Turbo_classes est une caractéristique particulière de restive.js qui ajoute des classes, en plus de celles définies dans l'option classes à notre balise <body> lorsque certaines conditions prédéfinies sont remplies, par

exemple lorsque l'appareil est un smartphone, lorsque l'appareil est une tablette...etc. Il y a en tout neuf conditions spécifiques au total, dans notre exemple on en utilise 5 :
Ls_landscape=landscape indiquera au plugin d'ajouter la class landscape a la balise <body> si l'appareil est orienté en mode paysage. Et comme Restive.js agit en temps réel, cette classe sera supprimée si l'appareil repasse en mode portrait. Dans le code au-dessus nous n'avons pas besoin de breakpoints (qui pour rappel sont des points d'arrêt définis) classiques, ce code est un exemple qui vise plus à comprendre la forme, nous nous contenterons de définir les breakpoints : ['10000'] et classes : ['nb'] pour créer une plage de 0 a 10 000 pixels qui correspondra à tous les appareils, puis nous utiliserons turbo_classes pour appliquer les classes pertinentes à la création de votre site

Les outils à adopter

De nombreux outils qui ne cessent d'évoluer sont mis à disposition des développeurs pour créer et mettre en place un site Web adaptatif, on va s'intéresser aux outils indispensables qui constituent la base pour la création de votre site Web.

Les Framework

On en parlait tout à l'heure les Framework ne sont pas indispensables pour créer un site Web adaptatif mais si vous vous lancez dans le développement de votre premier site Web, les Framework vous permettront de gagner du temps et d'accélérer le développement de votre site. Ceci n'est pas négligeable quand vous êtes face à une deadline serrée.

Bootstrap

L'un des Framework les plus répandus et les plus populaires est Bootstrap aussi connu sous le nom Twitter bootstrap Fig.1.
Bootstrap utilise la bibliothèque JavaScript « JQuery », c'est un ensemble qui contient des codes HTML et CSS, il regroupe de nombreux éléments graphiques tel que des formulaires, des boutons, des outils de navigation et d'autres éléments interactifs ainsi que des extensions JavaScript (en option). Le projet bootstrap a gagné en popularité sur la plateforme de gestion de développement GitHub. Aujourd'hui devenu incontournable, bootstrap est le Framework le plus populaire dans le développement de sites Web adaptatifs ou non.

Simple Grid Fig.2

Simple Grid est un outil très complet qui permet de simplifier la mise en page d'une grille par l'utilisation d'une classe par élément, l'utilisation de cet outil est relativement simple et intuitive ! Simple grid est composé de quatre gammes distinctes de taille d'écran :
En dessous de 720 pixels, < 720 px
Au-dessus de 720 pixels, > 720 px
Au-dessus de 985 pixels, > 985 px
Au-dessus de 1235 pixels, > 1235 px
Le but étant de limiter les barres de défilement horizontales, les internautes recevront automatiquement une mise en page adéquate à la taille de leur fenêtre de navigation.

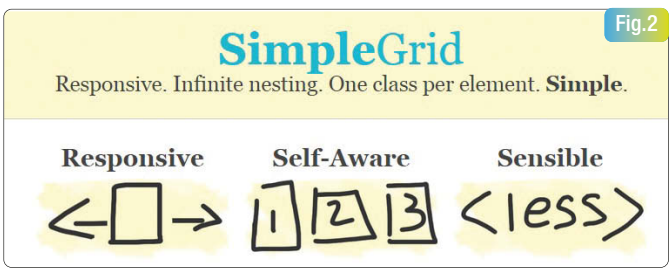


Fig.2

Fig.1

Foundation Fig.3

Foundation est un Framework très complet de création de prototypes de sites Web qui utilise la bibliothèque JQuery. Plus léger et plus simple d'utilisation, Foundation est une alternative a bootstrap qui permet de créer rapidement un prototype de votre futur site, il dispose de nombreux éléments graphiques.
Foundation inclut un outil de ligne de commande pour une création et une mise en place de projets assez rapide avec des modèles html optimisés qui permettront de créer des grilles et les éléments interactifs qui vont avec simplement, tout en étant efficaces et fluides.

Les Images Responsive

La gestion des images peut vite devenir problématique. Il faut faire attention a la taille en pixels, au poids de l'image, ainsi qu'à son format. Un développeur aguerri peut faire tout cela en implémentant les scripts nécessaires dans son code, mais il existe des outils conçus spécialement pour les néophytes et les développeurs débutants qu'on peut considérer comme « User Friendly » qui permettent d'effectuer cette tâche plus facilement et plus simplement.

Adaptive Images

Adaptive Images est un outil qui génère automatiquement des images optimisées et flexibles en fonction de la résolution d'écran de l'internaute, une solution en somme toute simple mais très efficace et assez pratique Fig.4.

FitText Fig.5

Pour clore cette partie consacrée aux outils pratiques pour développer un site responsive l'outil FitText pour aller de pair avec l'outil précédent qui est desti-

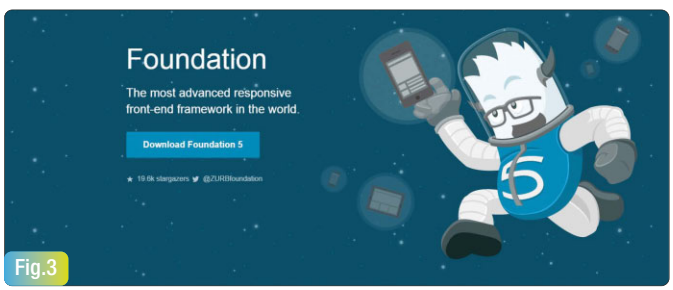


Fig.3

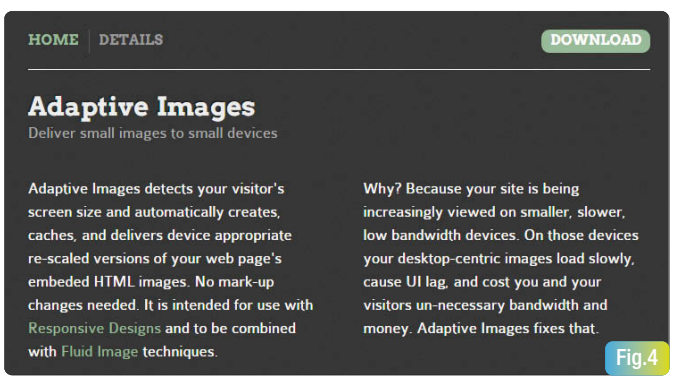


Fig.4

né aux images, FitText est un plugin JQuery qui permet d'ajuster Automatiquement la taille des textes et des titres en fonction de la résolution d'écran.

Les bonnes pratiques

Si vous entreprenez le projet de créer un site Web adaptatif il y'a plusieurs habitudes à prendre lors de la conception graphique notamment concernant la navigation, qui en général sur les écrans aux diagonales assez larges est axée sur une surface horizontale, afin de limiter le scrolling dans une page. A l'inverse sur des écrans de petite taille où la largeur est beaucoup moins importante, il serait plus pertinent de revoir ce principe de navigation, l'implémentation d'un site Web mobile est en grande partie contextuelle. Chez vous sur votre bureau ou dans les transports en communs, ou en marchant dans la rue, l'internaute aura besoin d'une interface claire où il peut naviguer et lire rapidement. Il faut donc adopter une approche différente lors de la création d'un site Web qui prend en charge le format mobile. On se concentrera principalement sur une conception simple qui regroupera l'essentiel de votre contenu, et on laissera de côté les effets superficiels. Partir d'un site existant pour le transformer en Responsive est une mauvaise idée, surtout si le site Web en question est complexe. Fournir des informations adaptées, cohérentes et qui ne sont pas omniprésentes sera une tâche fastidieuse à réaliser dans ce cas de figure. De plus, vous allez forcément rencontrer des bugs d'interface en procédant ainsi. La bonne stratégie consiste à partir sur une base responsive avant même la conception du site, en créant une maquette de celui-ci qui sera votre point de départ pour votre projet. Avec le développement des réseaux 3g+ et 4g le temps de chargement d'un site Web sur une plateforme mobile est devenu relativement rapide, mais le facteur du poids (de votre site web) est un élément à prendre en compte; la légèreté de votre site apportera un confort de navigation non négligeable aux internautes.

Fig.5

Il faudra aller à l'essentiel de votre contenu, pensez également à supprimer les medias et les appels de scripts qui ne sont pas indispensables, pour vos images par exemple, au lieu de les redimensionner on préfèrera créer une alternative, cela vous permettra d'avoir des images moins lourdes et d'exploiter moins de ressources (Mémoire, processeur) en évitant le procédé de redimensionnement. Pour résumer l'ergonomie est le maître mot quand il s'agit de navigation sur des appareils mobiles. Le contenu proposé devra offrir des informations pertinentes et un confort de lecture quel que soit le contexte d'utilisation.Concernant la typographie, on prendra soin d'augmenter la taille des polices pour les résolutions élevées et de les diminuer pour les basses. Ceci afin d'avoir un nombre de mots par ligne relativement consistant, ce qui améliorera considérablement la lisibilité de votre site. Au niveau des contenus médias, il faudra avoir recours à des scripts pour recadrer et diminuer en taille les images afin de les adapter aux différentes tailles d'écran. Des bibliothèques de scripts pour la majorité basés sur JQuery peuvent faciliter grandement cette tâche.

Enfin les requêtes media CSS permettront de définir les conditions à appliquer en fonction de la résolution de l'écran d'un appareil, par exemple en utilisant une syntaxe comme celle-ci :

@media screen and (min-width : 720 px) (max-width : 1920 px)

Pour spécifier une taille de texte et la faire varier en fonction des paramètres de largeur minimale (min-width) et maximale (max-width).

Il existe une quantité assez importante de scripts la plupart du temps en Javascript ou basés sur JQuery qui permettent de concevoir un site adaptatif de manière simple et créative et d'offrir un confort d'utilisation aux internautes quel que soit le type d'appareil utilisé.

Responsive Design Drupal avec Bootstrap

Le Responsive Design (RWD Responsive Web Design), ou le Web adaptatif, n'est plus à considérer comme un effet de mode, mais plutôt, comme un élément central dans la conception des sites Web. L'expérience utilisateur connaît un réel engouement et devient un élément central dans le monde digital depuis la démocratisation des tablettes et des Smartphones. Offrir aujourd'hui une expérience utilisateur de qualité est devenu un enjeu économique majeur. l'UX (expérience utilisateur) a révolutionné l'usage du Web et c'est la conséquence que chaque site Web doit s'adapter à chaque support.

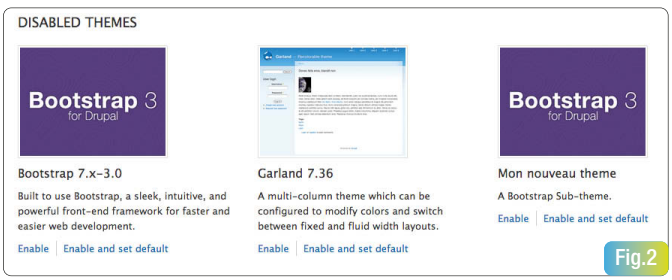
Tarik Larbi
Responsable technique TrainedPeople

Le design réactif, n'est plus une technique expérimentale, il est devenu un procédé collaboratif par des équipes de designers et de développeurs. Le résultat, c'est une expérience utilisateur qui répond à son contexte. Ce qu'il voit n'est pas une version réduite du site Web : le design s'adapte tout simplement aux caractéristiques techniques de l'appareil qui l'utilise.

L'objectif de cet article n'est pas de proposer un mode d'emploi ou d'une introduction au Web adaptatif, mais plutôt une manière de procéder, comment choisir et configurer des modules, des thèmes, et des techniques Responsive Design dans le contexte de Drupal.

Ce n'est pas un scoop, mais tout le monde sait que Drupal 7 n'intègre pas, nativement, un fonctionnement adaptatif, que ça soit du backend ou du frontend, contrairement à Drupal 8. La communauté Drupal a mis à disposition plusieurs thèmes et modules pour combler ce manque. Dans cet article nous aborderons comment utiliser ces concepts et quelle est la pratique d'intégration dans un thème responsive Drupal.

Les terminaux mobiles ou fixes permettant de surfer sur le Web se multiplient, et par conséquent les caractéristiques techniques se complexifient.



Mobile first (Mobile d'abord !)

Le terme "mobile first" est devenu une stratégie numérique incontournable dans la conception des sites Web dotés d'une forte expérience utilisateur. Pour simplifier les choses, le mobile first signifie qu'il faut commencer par la conception d'une version mobile. Une fois que c'est fait, nous commençons à étendre cette conception vers une version Desktop (Ordinateur de bureau). Avec cette approche, cela permettra de prendre des décisions conceptuelles de mise en œuvre. C'est plus facile de traduire une conception mobile à Desktop. A l'inverse, c'est beaucoup plus laborieux de traduire une conception Desktop à mobile.

Définir une stratégie numérique est un élément essentiel dans l'élaboration d'un produit ou d'une marque réussie. Une stratégie de l'expérience utilisateur d'un site Web orienté Desktop diffère d'une stratégie mobile. Traditionnellement l'expérience utilisateur d'un site Web Desktop est conçue pour des interactions d'un clavier et d'une souris.

L'approche "mobile first" existe pour une bonne raison. La première media queries (320 pixels et plus) va enlever tous les éléments de la mise en page de notre thème Drupal, dont on ne veut pas ou dont on n'a pas besoin et repositionner le reste.

Quel thème choisir pour votre site Drupal ?

Le choix du thème responsive est la pierre angulaire de la structure de votre contenu. Nous savons que Omega, Adaptive Thème et Zen sont les favoris de chacun. Sans compter que Omega et Adaptive Thème ont une quantité infinie d'options de configuration.

Bootstrap (version 3) Fig.1, sera notre choix du thème responsive Drupal.

Les autres sont certainement flexibles, mais leur configuration graphique peut paraître parfois un peu déroutante. Le choix d'un thème responsive dépend vraiment des besoins spécifiques de chaque projet et de la stratégie numérique mise en oeuvre, de sorte que Bootstrap pourrait ne pas ne pas être la réponse à votre prochain projet.

Création de notre sous-thème responsive

Tout d'abord, nous commençons par télécharger Bootstrap et créer par la suite notre sous-thème custom. Avec la commande Drush, on télécharge Bootstrap.

```
$ drush dl bootstrap
```

Après avoir exécuté cette commande, il suffit d'aller dans le répertoire sites/all/themes/bootstrap et copier le répertoire bootstrap_subtheme et de placer cette copie dans le répertoire thème. Renommer celui-ci par le nom de votre thème custom sans oublier de renommer aussi le fichier bootstrap_subtheme.info.starterkit par [le-nom-de-votre-thème].info.

Sur le panneau de configuration Drupal (admin/appearance) cliquez sur activer et configurer par défaut (Enable and set default) Fig.2. Pour plus d'informations sur la création d'un sous-thème Bootstrap consultez la documentation de la communauté https://www.drupal.org/node/1978010.

Officiellement Bootstrap est packagé et livré avec le pré-processeur sass. Il n'est donc pas besoin de conserver le répertoire less; le remplacer par un répertoire nommé bootstrap-sass afin que nous puissions utiliser toute la puissance des mixins (ils permettent de réaliser des sorties CSS paramétrables) qui viennent avec SASS & Compass.

Pré-processeur CSS

Il existe un certain nombre de pré-processeurs CSS : LESS, SASS, Stylus, etc. Les pré-processeurs correspondent essentiellement à la compilation du code CSS. Cette nouvelle approche de l'intégration qui se rapproche de la programmation dynamique permet de créer des règles CSS imbriquées et un ensemble de propriétés réutilisables. Le but étant de rendre le travail des intégrateurs (développeur front-end Drupal) plus organisé et maintenable.

Configuration du sous-thème avec le pré-processeur SASS

Cette partie consiste à télécharger la versions SASS de Bootstrap depuis cette adresse http://getbootstrap.com/getting-started/#download. Après, copiez le répertoire stylesheets qui se trouve dans le répertoire bootstrap-sass/assets/ et mettez-le dans le répertoire de notre sous-thème custom. Renommez celui-ci par bootstrap-sass. Ensuite, créez un nouveau répertoire nommé custom-sass, dans votre sous-thème custom, et ajoutez-y un fichier SCSS nommé style.scss. Dans ce dernier (style.scss) nous importons le bootstrap.scss en ajoutant cette ligne :

```
@import '../bootstrap-saas/bootstrap.scss';
```

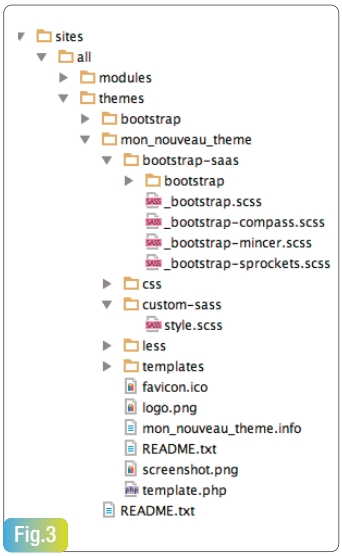


Fig.3

Votre site Drupal devrait maintenant être configuré avec Bootstrap, et votre structure de dossier devrait maintenant ressembler à : Fig.3.

Mise en place de Gruntfile.js et package.json

Passons maintenant à la configuration de Grunt pour le bon fonctionnement de la compilation des fichiers CSS et JavaScript. Grunt a besoin de deux fichiers, un fichier nommé package.json qui définit toute les dépendances dont on a besoin et Gruntfile.js pour configurer nos plugins.

Prérequis GRUNT

Grunt nécessite la présence de Node.js et Ruby dans votre système pour son fonctionnement. Sans entrer en détail dans l'installation et la configuration de Grunt, la suite de cet article fera en sorte que tout l'environnement est configuré et prêt à lancer la commande npm install à savoir :

- *L'installation de Grunt,*
- *L'installation de Compass,*
- *L'installation de SASS.*

Dans votre sous-thème custom, créer un fichier JSON nommé package.json et ajoutez le code suivant :

```
{
  "name" : "mon_nouveau_theme" ,
  "version" : "1.0.0" ,
  "auteur" : " Votre Nom ",
  " page d'accueil ":" : http //monsite.com ",
  "moteurs" : {
    "noeud" : "> = 0.8.0"
  },
  "devDependencies" : {
     "grunt-contrib-watch" : "v0.9.0" ,
     "grognement-contrib-sass" : "v0.7.3" ,
     ...
  }
}
```

Dans ce fichier (ci-dessus), vous pouvez ajouter n'importe quels plugins les mieux adaptés à votre projet Drupal dans la section devDenpendencies. Consultez la liste complète des plugins officiels sur le site http://gruntjs.com/plugins pour plus d'information. Ensuite, après avoir configuré notre fichier package.json, lancez la commande suivante :

```
$> npm install
```

Cette commande regarde le fichier package.json et installe tous les plugins dont on a besoin. Si vous souhaitez ajouter un autre plugin Grunt à partir du fichier *package.json*. Vous devez relancer cette commande.
Une fois après l'exécution de cette commande, vous remarquerez un nouveau dossier dans le répertoire du sous-thème custom, appelé node_modules qui stocke tous les plugins. Si vous utilisez un outil de versionning (git ou svn) dans votre projet Drupal, assurez-vous d'ignorer ce dossier. Maintenant passons à la configuration de Grunt afin d'utiliser les plugins et d'automatiser la compilation. Dans le répertoire du sous-thème custom, créez un fichier nommé Gruntfile.js et mettez le code suivant :

```
module.exports = function (grunt) {
  grunt.initConfig({
    watch: {
      src: {
        files: ['**/*.scss', '**/*.php'],
        tasks: ['dev']
      },
     options: {
        livereload: true,
      },
    },
    compass: {
      dev: {
        options: {
          sassDir: 'custom-sass',
          cssDir: 'css',
          imagesPath: 'assets/img',
          noLineComments: false,
          outputStyle: 'compressed'
        }
      }
    }
  });
  grunt.loadNpmTasks('grunt-contrib-compass');
  grunt.loadNpmTasks('grunt-contrib-sass');
};
```

Pour plus de détails et voir les options disponibles, je vous invite à consulter cette documentation à l'url https://github.com/gruntjs/grunt-contrib-watch.

Installer et configurer LiveReload

Télécharger et activer le module Drupal LiveReload qui permet d'appliquer dynamiquement vos changements des CSS et JavaScript sans rafraîchir votre page sur votre navigateur Web. Une fois le module activé, il faut actualiser la fenêtre de votre navigateur pour charger la bibliothèque livereload.js. Par défaut, il faut être connecté comme administrateur pour que le chargement automatique prenne effet; vous pouvez changer cela dans les permissions du module. Voilà tout est prêt pour lancer votre compilation Grunt avec la commande suivante :

```
$> grunt watch
```

Maintenant, chaque fois que vous éditez un fichier SCSS, Grunt se lancera automatiquement pour compiler votre fichier SCSS et l'intégrer dans le fichier style.css.
Voilà votre sous-thème est configuré et prêt à l'emploi pour des bonnes pratique d'intégration. Cependant, il reste plusieurs aspects de Grunt qu'on n'a pas évoqué étant donné que cette technologie peut faire l'objet d'un sujet à part entière.

Back-office Drupal mobile-friendly

Faciliter l'administration d'un site Drupal 7 sur un appareil mobile passe forcément par quelques ajustements. D'abord, nous commençons par télécharger le module navbar qui offre une barre d'administration mobile plus conviviale (voir **Fig.4**). Celui-ci, nécessite le module libraries et quelques bibliothèques JavaScript.
Drush, pour installer les modules et leurs dépendances.

```
$ drush dl libraries
$ drush en -y libraries
$ drush dl navbar
$ drush en -y navbar
```

Désactiver les deux modules du core Drupal 7, toolbar et overlay. La désactivation du module overlay n'est pas une recommandation, mais plutôt un choix personnel.

```
$ drush dis -y overlay
$ drus dis -y toolbar
```

A ce stade, la barre de navigation est mise en place mais pas fonctionnelle et ce, en raison des bibliothèques JavaScript manquantes : Backbone.js http://backbonejs.org, Underscore.js http://underscorejs.org, Modernizr.js http://modernizr.com.
Pour Backbone et Underscore il suffit de télécharger les deux fichiers JavaScript et de les copier dans le répertoire sites/all/libraries. Cependant, pour la bibliothèque Modernizr, il faut utiliser l'outil de build de Modernizr pour sélectionner uniquement les fonctionnalités que vous souhaitez tester. Vous pouvez installer le module contrib Drupal JavaScript Libraries Manager https://www.drupal.org/project/javascript_libraries.

Modernizr

Modernizr a été écrit par Ates Farouk en 2009 et, au moment de l'écriture de cet article, est disponible dans sa version 2.8.3, mais également en version 3 beta. Modernizr est un petit paquet de JavaScript qui offre une fonctionnalité simple de détection des composants HTML5 jusqu'à CSS3 en passant par les API JavaScript comme Drag and Drop ou Local Storage.
Il fonctionne de deux manières distinctes : en premier lieu, en ajoutant des classes à l'élément <html> : en deuxième lieu, en fournissant une API JavaScript.
Lorsqu'on charge un thème Drupal qui inclut Modernizr, la bibliothèque exécute sa série de détection de fonctionnalités et utilise les résultats pour ajouter des attributs de classe à l'élément <html> de la page native des navigateurs Web.
Beaucoup de ces caractéristiques sont déjà en place sur les navigateurs modernes. Modernizr vous donne une qualité incroyable de contrôle et s'assure que chaque utilisateur est servi avec une expérience de qualité.

Configurer la balise Meta Viewport

La définition de cette balise pourrait faire un sujet d'article à lui tout seul. Pour faire simple, la balise meta viewport indique au navigateur mobile comment il doit interpréter votre page Web. La plupart des sites web optimisés pour le mobile utilisent une balise comme la suivante :

```
<meta name="viewport" content="width=device-width, user-scalable=no">
```

La propriété width contrôle la taille du *viewport*. Elle peut être réglée à une valeur précise de pixels, comme width=600, ou bien à la valeur spéciale device-width qui correspond à la largeur de l'écran en pixels CSS à l'échelle 100%. Dans notre sous-thème Bootstrap la modification de cette

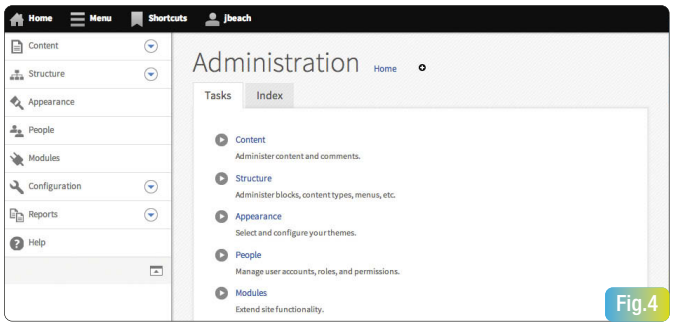


Fig.4

propriété peut se faire à travers un module contrib Add to Head https://www.drupal.org/project/add_to_head, ou à partir de la fonction thème theme_preprocess_html.

```
function mon_nouveau_theme_preprocess_html(&$variables) {
 $viewport = array (
  '#tag' => 'meta' ,
  '#attributes' => array (
   'name' => 'viewport' ,
   'content' => 'width=device-width, user-scalable=no' ,
  ),
 );
 drupal_add_html_head($viewport, 'viewport');
}
```

Définition des Breakpoint et Media Queries

Les media queries sont au coeur du développement responsive. Elles permettent d'indiquer quand on doit appliquer des propriété CSS. Cela peut-être configuré dans Drupal comme ça pourrait l'être en dehors. Définir les points de rupture (Breakpoint) repose, généralement, sur une approche de 3 ou 4 vues (portables, tablettes, Netbooks, Desktops). Les médias queries permettent de modifier l'apparence d'une page HTML en fonction de la résolution ou de la densité de pixels d'un appareil.
La version 3 de Bootstrap orientée Mobile-First, intègre nativement une nouvelle grille avec des points de ruptures préconfigurés. Ainsi, connaître les tailles de grille est indispensable pour l'exécution de vos médias queries. Pour plus d'information il faut se rendre à cette adresse http://getbootstrap.com/css/#grid pour voir la documentation officielle.

```
/* Small devices (tablets, 768px and up) */
@media (min-width: @screen-sm) {
 .header-btn {
  display: none;
 }
}

/* Medium devices (desktops, 992px and up) */
@media (min-width: @screen-md) {
 .slogan {
  display: none;
 }
}
/* Large devices (large desktops, 1200px and up) */
@media (min-width: @screen-lg) {

}
```

Pages et templating avec les Grilles

Toute grille doit être suffisamment complète pour supporter les types de contenu spécifiques au projet Drupal. Le 960 Grid System est très populaire, car il possède cet équilibre : on peut avoir un grand nombre de colonnes

divisées équitablement et qui s'adaptent facilement aux besoins de la plupart des contenus.

Pour garder les choses bien organisées dès le début, nous devons commencer par mettre en place notre grille. Un concepteur bien nanti saura toujours concevoir indépendamment d'une grille. Mais avec tellement de choses à considérer dans une conception adaptative, la grille sera d'une aide majeur à la mise en page.

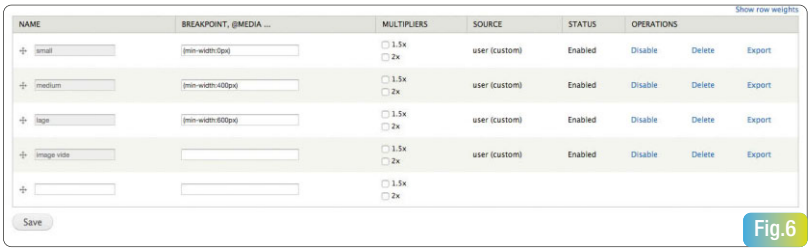


Fig.6

Une grille de 12 colonnes Fig.5, est particulièrement utile car elle se divise en 2, 3, 4 et 6 colonnes, et fournit donc une bonne répartition de contenu et des combinaisons sur le plan de diffusion sur les objets connectés.

OPTIMISATION DE LA TYPOGRAPHIE RESPONSIVE

L'optimisation de la typographie sur tous les types de supports disponibles est un sujet récurrent. Ainsi, rendre la typographie lisible sur tous les supports est une tâche délicate.

Les em sont privilégiés par rapport aux pixels pour mettre en place la typographie des sites responsive. Les em s'adaptent plus facilement aux tailles d'affichage et d'écran, alors que les pixels ne se redimensionnent pas toujours bien.

Les différences de résolution d'écran entre les appareils affectent la lisibilité du texte, y compris sur les appareils de même taille. Configurez l'échelle typographique de base sur autant d'appareils de taille identique que possible.

DES IMAGES RESPONSIVE

Adapter les images vise d'abord avant tout à améliorer l'expérience utilisateur, en délivrant l'image qui convient le mieux. Inutile d'envoyer une image trop lourde ou trop grande si l'utilisateur consulte votre site sur Smartphone avec un petit écran et une mauvaise connexion Internet ! En utilisant Drupal, nous pouvons automatiser la création des versions nécessaires d'une seule image, ce qui rend la création de contenu beaucoup plus facile. Premièrement, nous devons créer les styles d'images pour chaque point d'arrêt (breakpoint). Dans notre exemple de sous-thème Bootstrap nous avons défini 4 groupes des points d'arrêt. (large, small, medium, vide); il devrait y avoir un style d'image pour chaque, avec un effet de resize (crop and scale).

Fig.5

Drupal doit être au courant de la stratégie des breakpoint mise en place et pour cela nous utilisons le module contrib breakpoint https://www.drupal.org/project/breakpoints. Le module breakpoints a deux façons de définir les points d'arrêt, à partir de l'interface de configuration admin/config/media/breakpoints ou dans le fichier .info de notre custom thème Fig.6.

Ensuite, il faut créer un groupe de points d'arrêt. Imaginez que vous ayez encore plusieurs points de rupture définis, mais que vous vouliez choisir ceux spécifiques à appliquer à une composante du site. Avec les groupes des points d'arrêt, vous avez cette capacité.

Bien que le module breakpoints gère bien l'organisation des points d'arrêt, le module contrib picture https://www.drupal.org/project/picture utilise les groupes des points d'arrêt pour créer le balisage. Après son téléchargement et son installation, activer le module "ajouter une nouvelle cartographie". Après avoir choisi votre groupe de points de rupture, vous avez quelques options qui devraient se produire à chaque point d'arrêt.

Dans notre cas, nous voulons faire correspondre chaque point d'arrêt au style d'image approprié quenous avons créé précédemment.

CONCLUSION

D'après un article paru le 07 avril dans le JDN (Journal du Net), Google a annoncé que la compatibilité mobile sera un critère de pertinence sur la recherche sur Mobile. En effet, à partir du 21 avril 2015, le géant américain revoit son classement des pages Web optimisées pour le mobile afin de bénéficier du label "mobile friendly". D'après l'article, "*un tiers des sites Web du CAC 40 ne sont pas prêts pour la mise à jour du 21 Avril*".

Le responsive design n'est pas le seul critère pour l'obtention de ce label. Le géant de l'Internet a mis en place un outil appelé "Page Speed Insight" permettant d'analyser un site Web selon deux critères : la vitesse et l'expérience utilisateur (UX).

La notion RWD est simple : il s'agit de créer des sites Web qui réagissent instantanément en fonction de l'appareil qu'utilise l'internaute, à sa taille, sa résolution, ses capacités techniques, etc. Drupal 7 offre plusieurs outils de l'écosystème responsive afin d'offrir un produit meilleur avec une forte expérience utilisateur.

Pour conclure, voici quelques statistiques qui indiquent cette prolifération qui ne fera que continuer :

- En 2014, le trafic mobile sur des sites Web a dépassé celui des ordinateurs de bureau
- 86% des utilisateurs d'Internet mobile utilisent leur appareil mobile tout en regardant la télévision (Yahoo 2011)
- Insight Express a constaté que 43% de l'accès à Internet mobile se produit à la maison

Commencer avec le Microsoft Band SDK

Il y a peu, Microsoft a sorti la première version du Microsoft Band SDK. Ce SDK est disponible pour iOS (7), Android (API 17) et Windows Phone 8.1 en version native. Si vous êtes un développeur C#, sachez aussi que Xamarin a porté ce SDK en C# pour iOS et Android.

John Thiriet
Consultant .NET
Microsoft MVP Client Platform Development
Cellenza

Cellenza

Avant toute chose, il est important de noter que ce SDK ne permet pas de développer des applications pour le Microsoft Band, mais plutôt des applications mobiles dialoguant avec ce dernier.
Ce SDK offre 3 grandes fonctionnalités :

- Créer des tuiles et envoyer des notifications associées,
- Lire les capteurs,
- Personnaliser la tuile principale (alias Me Tile).

Quel que soit le téléphone choisi, il doit être appairé en Bluetooth avec le Band pour pouvoir lancer et tester des applications.
Dans cet article, on utilisera donc un téléphone Windows Phone 8.1 puisque c'est ce que j'utilise avec mon Band. L'environnement de développement sera ainsi celui d'une application Windows Phone classique : Windows 8.1 avec Visual Studio 2013.

Installation du SDK

L'installation du SDK en lui-même est très simple car elle consiste juste à l'ajout d'un paquet Nuget **Fig.1 et 2**.
Après l'installation, il faut aller éditer le fichier Package.appmanifest et rajouter les lignes suivantes dans la section Capabilities pour autoriser les connexions Bluetooth avec le Band **Fig.3**.
Il faudra ensuite aussi éditer les « capabilities » de l'application pour rajouter le support des capteurs de proximité. On peut le faire via les propriétés du projet ou à la main avec la ligne suivante : **Fig.4**.

Connexion au Band

Un téléphone pouvant techniquement être appairé à plusieurs Band, la première opération à effectuer consistera à lister les Band actuellement accessibles.

```
IBandClientManager clientManager = BandClientManager.Instance;
IBandInfo[] bands = await clientManager.GetBandsAsync();
```

Ensuite, on choisit un Band (ici le premier) et on tente une connexion.

```
try
{
    using (IBandClient client = await clientManager.ConnectAsync(bands[0]))
    {
        // Traitement en cas de succès
    }
}
catch (BandException)
{
    // Traitement en cas d'échec
}
```

Fig.1

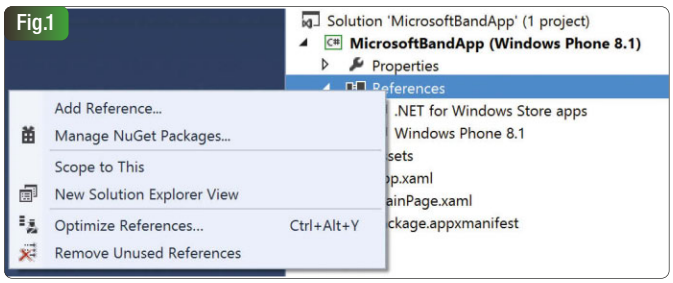
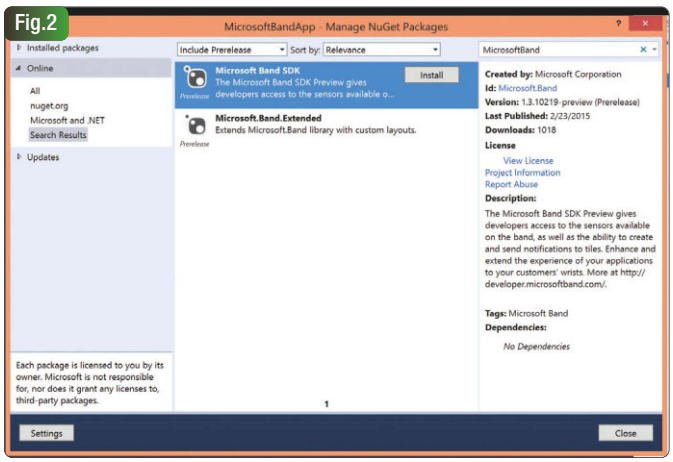


Fig.2




```
<m2:DeviceCapability Name="bluetooth.rfcomm">
  <m2:Device Id="any">
    <!-- Used by the Microsoft Band SDK Preview -->
    <m2:Function Type="serviceId:A502CA9A-2BA5-413C-A4E0-13804E47B38F" />
    <!-- Used by the Microsoft Band SDK Preview -->
    <m2:Function Type="serviceId:C742E1A2-6320-5ABC-9643-D206C677E580" />
  </m2:Device>
</m2:DeviceCapability>
```

Fig.3

```
<DeviceCapability Name="proximity" />
```

Fig.4

Numéro de version

Il est parfois utile de connaître les numéros de version des Band avec lesquels on communique. Récupérer le numéro de version du Band est, par ailleurs, un bon candidat pour vérifier que les étapes précédentes se soient bien passées.

```
string hwVersion = await client.GetHardwareVersionAsync();
string fwVersion = await client.GetFirmwareVersionAsync();
```

Les capteurs

Le Band propose un grand nombre de capteurs que nous pouvons interroger avec le SDK :

- Accéléromètre,
- Gyroscope,
- Distance,
- Rythme cardiaque,
- Podomètre,
- Température de la peau,

- UV,
- Contact de la peau.

Cette interrogation se fait au travers d'un abonnement à un événement spécifique à chaque capteur. Certains capteurs délivrent leurs valeurs à intervalles réguliers, d'autres avec un intervalle que l'on peut spécifier, et, enfin les derniers ne les délivrent que lorsque leurs valeurs changent.
Attention cependant, chaque abonnement à un capteur utilise de la batterie sur le Band, certains plus que d'autres. Aussi, il est fortement conseillé de ne s'abonner que lorsque cela est vraiment pertinent et indispensable au fonctionnement de l'application.
Par exemple, voici comment cela fonctionne avec le capteur de rythme cardiaque :

```
// Récupération du capteur
var heartRateSensor = client.SensorManager.HeartRate;

// Abonnement à l'évènement de mise à jour des valeurs du capteur.
heartRateSensor.ReadingChanged += (o, args) =>
{
   Debug.WriteLine("Rythme : {0}, Qualité : {1}",
      args.SensorReading.HeartRate, args.SensorReading.Quality);
};

// Récupération du premier intervalle de mise à jour.
var interval = heartRateSensor.SupportedReportingIntervals.FirstOrDefault();

// Affectation de cet intervalle au capteur.
heartRateSensor.ReportingInterval = interval;

// Début de lecture des valeurs du capteur.
await heartRateSensor.StartReadingsAsync();

// On lit les valeurs pendant 20 secondes.
await Task.Delay(20000);

// On arrête la lecture.
await heartRateSensor.StopReadingsAsync();
```

La logique pour les autres capteurs est toujours la même :

- Abonnement à l'événement,
- Spécification de l'intervalle (si disponible),
- Démarrage de la lecture,
- Arrêt de la lecture.

Il n'est donc pas très compliqué de lire et de traiter les capteurs.

Tuiles

Il est possible de créer et de gérer des tuiles sur le Band. Ces tuiles permettent d'ajouter des fonctionnalités sympathiques à l'application téléphone comme l'envoi de notifications ou de messages directement sur le Band. Les tuiles sont composées d'un logo blanc sur fond transparent, de son nom ainsi qu'optionnellement d'un thème, le thème par défaut étant celui du Band. Elles peuvent supporter ou non, à la discrétion du développeur, la notion de badge indiquant un compteur de notifications non lues.
Les images des tuiles sont de deux dimensions :

- 46x46 pour la tuile principale,
- 24x24 pour l'image affichée au-dessus du badge.

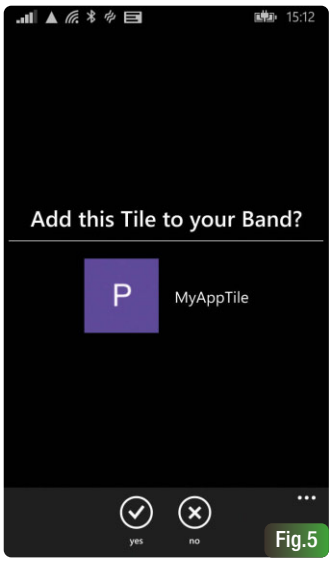


Fig.5

Création et suppression

Le Microsoft Band ne prend en charge que 13 tuiles séparées, toutes applications confondues. Il faut donc récupérer le nombre de tuiles libres avant toute opération de création de tuile.

```
int capacity = await client.TileManager.GetRemainingTileCapacityAsync();
```

Si le compte est supérieur à zéro, la création d'une tuile est possible.
Afin d'ajouter une tuile, il faut récupérer son image associée. Dans le cas présent, l'image est située dans le dossier Assets de l'application Windows Phone.

```
// On crée un bitmap de la taille de la tuile.
WriteableBitmap tileIconBitmap = new WriteableBitmap(46, 46);

// On récupère l'image de la tuile dans le dossier Assets.
var tileUri = new Uri("ms-appx:///Assets/MicrosoftBandAppLogo.png");
var tileFile = await StorageFile.GetFileFromApplicationUriAsync(tileUri);

// On lit l'image et on définit son contenu comme source du bitmap.
using (var stream = await tileFile.OpenReadAsync())
   await tileIconBitmap.SetSourceAsync(stream);

// On convertit le bitmap en un format compatible avec le Band.
BandIcon tileIcon = tileIconBitmap.ToBandIcon();
```

La manipulation est identique pour la petite tuile, excepté pour la taille qui est de 24x24. On passe à présent à la création de la tuile en elle-même.

```
// On crée le Guid de la tuile.
var guid = Guid.Parse("FAFF5F9B-E43C-4686-855E-BB1971938606");

// On crée la tuile
var newTile = new BandTile(guid)
{
   Name = "MyAppTile",
   TileIcon = tileIcon,
   SmallIcon = smallTileIcon,
   IsBadgingEnabled = true
};

// On pousse la tuile sur le Band
await client.TileManager.AddTileAsync(newTile);
```

Si tout se passe bien, le Microsoft Band SDK va alors demander une confirmation à l'utilisateur et si celui-ci l'accepte, le Band va se synchroniser Fig.5.
Pour supprimer une tuile, il suffit d'avoir l'identifiant de la tuile et d'exécuter le code suivant :

```
// Récupération des tuiles de l'application
var tiles = await client.TileManager.GetTilesAsync();
```

```
// Récupération de la tuile à supprimer
var tile = tiles.First(it => it.TileId == guid);

// Suppression de la tuile
await client.TileManager.RemoveTileAsync(tile);
```

Notifications

Boîtes de dialogue

Les boîtes de dialogue sont des messages s'affichant immédiatement sur le Band. Ils peuvent être ignorés par l'utilisateur et ils ne sont pas sauvegardés après que l'utilisateur les ait lus.
Pour afficher un message de dialogue, il suffit d'avoir l'identifiant de la tuile et d'exécuter le code suivant :

```
await client.NotificationManager.ShowDialogAsync(guid, "Titre", "Contenu");
```

Messages

A la différence des boîtes de dialogue, les messages sont sauvegardés (à hauteur de 8 notifications par tuiles). Par défaut ils ne s'affichent pas directement à l'utilisateur et celui-ci doit se rendre sur la tuile pour voir le message. Il est cependant possible de demander au SDK d'afficher ce message à la manière d'une boîte de dialogue.
Le code suivant envoie un message sous forme de dialogue :

```
await client.NotificationManager.SendMessageAsync(
  guid, "Titre", "Contenu",
  DateTimeOffset.Now, MessageFlags.ShowDialog);
```

Vibrations haptiques

Il est également possible de faire vibrer le Band. Il existe plusieurs types de vibrations que l'on peut choisir lors de l'appel de la méthode suivante :

```
await client.NotificationManager.VibrateAsync(VibrationType.NotificationAlarm);
```

Personnalisation du Band

Le SDK offre quelques possibilités de personnalisation. On peut modifier l'image de fond de la tuile principale ou encore changer le thème par défaut du Band.

Tuile principale

Le SDK permet de récupérer et changer la couleur ou l'image de fond de la tuile.

Récupération de l'image de fond

```
var bandImage = await client.PersonalizationManager.GetMeTileImageAsync();
var bitmap = bandImage.ToWriteableBitmap();
```

Modification de l'image de fond

Pour modifier l'image de fond, il faut une image d'un format de 310x102 :

```
await client.PersonalizationManager.SetMeTileImageAsync(bitmap.ToBandImage());
```

Thème principal

Comme pour l'image de fond, on peut récupérer le thème actuel, le modifier ou en définir un nouveau.

Récupération du thème

```
var bandTheme = await client.PersonalizationManager.GetThemeAsync();
```

Modification du thème

```
var bandTheme = new BandTheme
{
  Base = Colors.MediumPurple.ToBandColor(),
  HighContrast = Colors.Magenta.ToBandColor(),
  Highlight = Colors.Orchid.ToBandColor(),
  Lowlight = Colors.Plum.ToBandColor(),
  Muted = Colors.Thistle.ToBandColor(),
  SecondaryText = Colors.LightPink.ToBandColor()
};
await client.PersonalizationManager.SetThemeAsync(bandTheme, CancellationToken.None);
```

Conclusion

Bien que l'on ne puisse pas encore développer d'application fonctionnant sur le Band à proprement dit, le Microsoft Band SDK ouvre un certain nombre de possibilités intéressantes.
Le Microsoft Band contient un très grand nombre de capteurs qui sont très importants lorsque l'on souhaite développer des applications orientées santé, bien-être ou sport.
Même sans utiliser tous les capteurs disponibles, ajouter un peu de vie à vos applications en proposant l'envoi de notifications sur le Band de vos utilisateurs est toujours quelque chose qui sera apprécié.
C'est maintenant à vous de jouer et de montrer tout ce que ce petit appareil peut apporter !

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

Ont collaboré à ce numéro : S. Saurel.

Secrétaire de rédaction : Olivier Pavie
Experts : J. Dollon, G. Madison, C. Grandval, T. Leriche-Dessirier, F. N'Guessan, V. Nëel, F. Santin, T. Lebrun, J. Antoine, M. Fery, J. Landon, S. Thevenin, F. Dupont, J-F. Garreau, E. Muraton, G. Paramelle, P. Dos Santos, M. Diagne, P. Cavezzan, A. Richard, P. Charrière, G. Duval, A. Vaché, M. Mottet, E. Roset, f. Dewinne, Streamdata, A. Moreau, S. Cordonnier, G. Damien, S. Sibué, A. Chouguiat, T. Larbi, J. Thririet,

Photos/illustrations : D.R., des éditeurs, des auteurs, Valérie Turmel, François Tonic, Apple, Stock vector © bagiuiani

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.

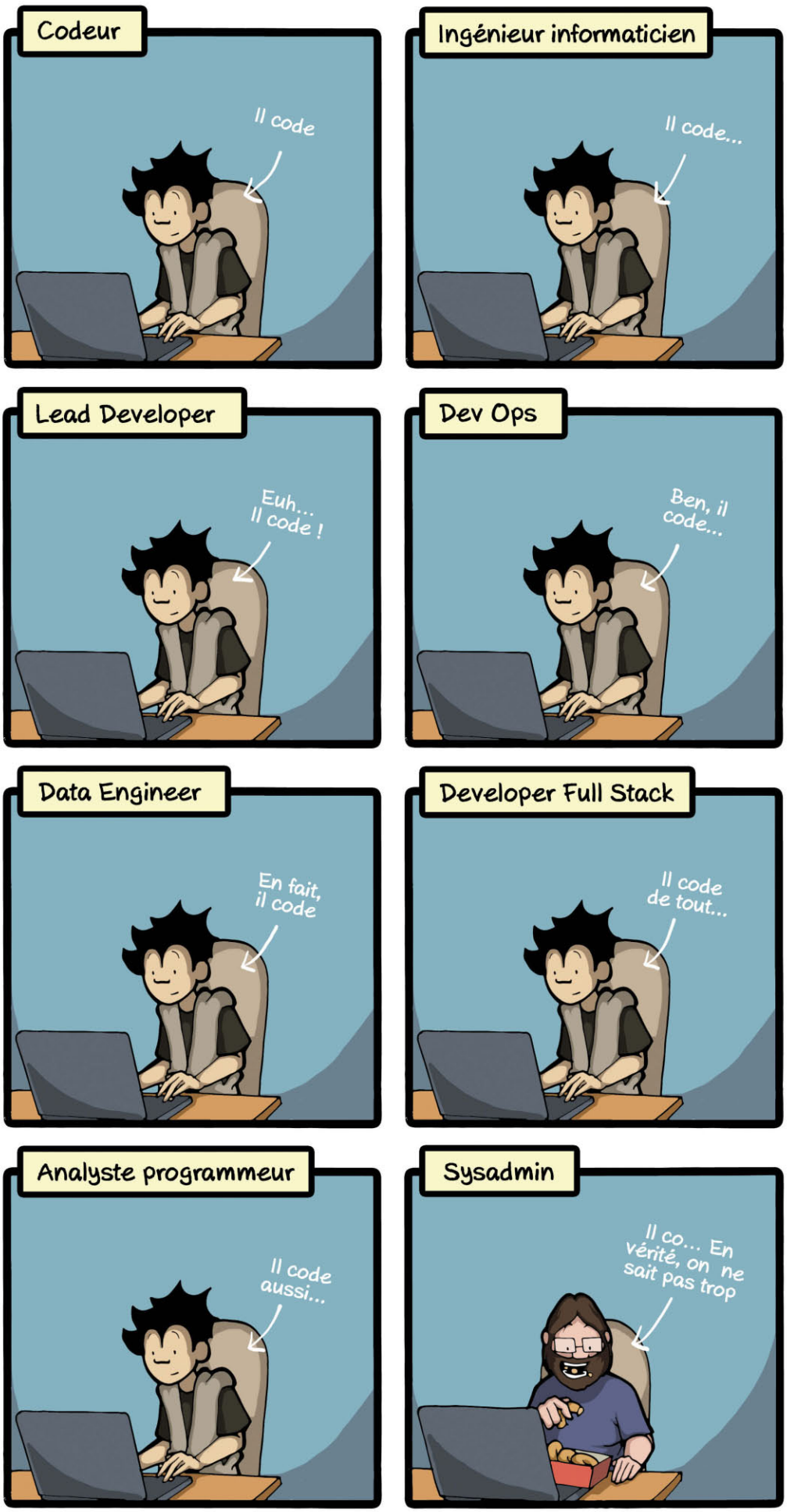
Marketing et promotion des ventes :
Agence BOCONSEIL - Analyse Media Etude

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

Quand je dois expliquer les différents métiers IT



CommitStrip.com

Sur abonnement ou en kiosque

Le magazine des pros de l'IT

Mais aussi sur le web

Ou encore sur votre tablette

www.linformaticien.com

L'INFORMATICIEN

PROFITEZ DE L'OPÉRATION
« POUR 1 EURO DE + »
JUSQU'AU 3 JUILLET

COMMANDEZ WINDEV MOBILE 20

OU WEBDEV 20 OU WINDEV 20

ET RECEVEZ LE NOUVEAU GALAXY S6 EDGE

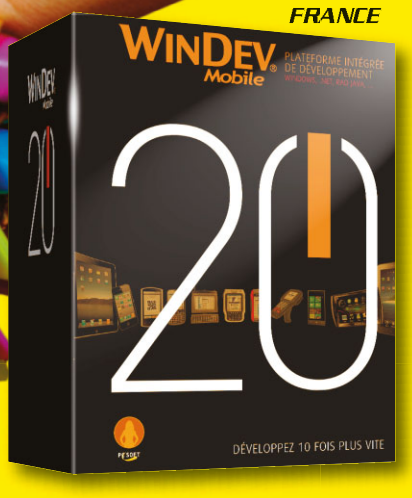
INCLUS: UN CHARGEUR À INDUCTION !

SAMSUNG
Galaxy S6 edge

Concept Mono-bloc • Ecran incurvé capacitif Super Amoled QHD 5.1" • Résolution 2560 x 1440 • Quadribande • 3G/4G • Android Lollipop 5.0 • APN 16 MP Autofocus f/1.9 • APN frontal 5.0 MP • Résolution vidéo en enregistrement UHD 4K (3840 x 2160) • Mémoire 32 Go • Processeur Octo Core 2.1 GHz • Bluetooth • Wifi • Batterie Li-ion 2600 mAh • USB 2.0 • Recharge rapide • GPS • NFC • ... *Livré avec un chargeur à induction*

Ou choisissez **1 Galaxy S6 64 Go** ou **2 Tablettes Galaxy Tab S 10,5"** ou **3 Tablettes Galaxy Tab 4 10,1"** ou encore **1 Télé Samsung Full HD Internet 140cm**.

WINDEV AGL N°1 en FRANCE



PROFITEZ DE L'OPÉRATION « POUR 1 EURO DE + »

Pour bénéficier de cette offre exceptionnelle, il suffit de commander WINDEV Mobile 20 (ou WINDEV 20, ou WEBDEV 20) chez PC SOFT au tarif catalogue avant le 3 juillet 2015.
Offre réservée aux sociétés, administrations, mairies, GIE et professions libérales..., en France métropolitaine. Aucun abonnement n'est à souscrire pour bénéficier de cette offre.
Le développement pour Android et iOS s'effectue avec WINDEV Mobile ou WEBDEV. Le développement pour Windows et Linux s'effectue avec WINDEV ou WEBDEV. Voir tous les détails et des vidéos sur : **www.pcsoft.fr**
Le Logiciel et le matériel peuvent être acquis séparément; merci de vous connecter au site www.pcsoft.fr pour consulter la liste des prix et les dates de disponibilité. Tarifs modifiables sans préavis.

WINDEV Mobile 20 permet de créer facilement et rapidement des applications natives pour iOS, Android, Windows Phone et Windows Mobile. Liaison facile à votre SI et à toute base de données.
Les applications WINDEV sont directement importées.

Aucun abonnement à souscrire pour bénéficier de cette offre.

Fournisseur Officiel de la Préparation Olympique
www.pcsoft.fr

Document non contractuel. Ceci est la 23ème opération «pour 1 euro de +» organisée par PC SOFT.